TOWARDS DEVELOPING EFFECTIVE HUMAN-ROBOT SYSTEMS

by

Jacob W. Crandall

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Jacob W. Crandall

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Michael A. Goodrich, Chair
Date	Dan R. Olsen, Jr.
Date	Quinn Snell

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Jacob W. Crandall in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Michael A. Goodrich
Chair, Graduate Committee

Accepted for the Department

David Embley
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

TOWARDS DEVELOPING EFFECTIVE HUMAN-ROBOT SYSTEMS

Jacob W. Crandall

Department of Computer Science

Master of Science

There is a rising need for robust human-robot systems. Such systems require that humans and robots work together efficiently. In order for humans and robots to work together efficiently, these systems must include both efficient interfaces and robots that can carry out some tasks autonomously. These two elements of system intelligence are captured in two notions: interface efficiency and neglect tolerance. In this thesis, we develop metrics for these two concepts. These metrics can be used to measure and compare the effectiveness of human-robot systems.

In order to obtain the measures of neglect tolerance and interface efficiency for a human-robot system, we develop a measurement technology. This measurement technology calls for a large number of user experiments to be performed. From these experiments, random processes that measure the neglect tolerance and interface efficiency of a system can be estimated nonparametrically. The measurement technology requires the development of two other related metrics. They are instantaneous robot performance and world complexity, which we describe in this thesis.

To validate the usefulness of the interface efficiency and neglect tolerance metrics, we compare three different systems for a certain navigation task. In the task, a human helps a robot navigate through its world to specified goal positions. A user study involving

40 subjects is used to estimate the interface efficiency and the neglect tolerance of these systems. The systems are then compared based on these measures.

While the measurement technology described in this thesis requires that a large number of user experiments be performed, we show that the neglect tolerance and interface efficiency of a system can be approximated with only a few user experiments. This allows more practical application of the neglect tolerance and interface efficiency metrics described herein.

ACKNOWLEDGMENTS

I would like to thank my parents, who provided me with a wonderful life and opportunities to learn and develop. These gifts to me came at great sacrifices to them. My advisor, Dr. Michael Goodrich, also deserves my gratitude. In addition to providing much advice and input to this thesis, he has provided me with many opportunities. So many other people have influenced me over the years that I certainly can not name them all, but I appreciate what they have done for me. Most of all, however, I would like to thank my wife, Tawna, who makes my life so much better.

Contents

Chapter 1

Introduction

In many applications, it is desirable to allow a human to interact with multiple independent robots. These applications include search-and-rescue, exploration, hazardous waste clean-up, and so on. In such applications, efficient interactions between humans and robots are necessary for such systems to perform proficiently. Two components that, in part, determine the effectiveness of human-robot systems are the efficiency of the interface between humans and robots, and the capabilities of the robots' artificial intelligence. These two components can be captured in the concepts of *interface efficiency* and *neglect tolerance*.

In this thesis, we present a method for measuring the interface efficiency and neglect tolerance of human-robot systems. In order to do this, we first develop two related metrics. They are instantaneous robot performance and world complexity. Second, we develop a measurement technology from which measures of the interface efficiency and neglect tolerance can be estimated nonparametrically. These measures can be used to assess the effectiveness of a human-robot system.

1.1 Motivations

Human-robot systems face various challenges. In particular, a gulf exists between humans and robots that impedes system performance. This is largely because humans do not understand robots, and robots do not "understand" humans. We will call this a *barrier of*

understanding (related closely to the gulfs of evaluation and execution described in [37]). This so called *barrier of understanding* can be overcome by identifying the interaction needs of both humans and robots.

Interaction is largely for the purpose of overcoming and avoiding limits. Both robots and humans have limits that contribute to the *barrier of understanding*. For example, a robot is limited by its hardware and software (artificial intelligence) capabilities. A human, on the other hand, has, in general, better hardware and uncanny reasoning skills. However, we have limits in attention, cognition, physical strength, and brain power. Good combinations of robot and human capabilities can allow problems to be solved more efficiently. Such combinations are more easily obtained when human-robot interactions are efficient. The metrics of neglect tolerance and interface efficiency can be used to compare various human-robot systems. They can also serve to identify the strengths and weakness of these systems.

1.2 Thesis Statement

In this thesis, we will develop the metrics of neglect tolerance and interface efficiency for use in human-robot systems. We will also present a measurement technology that can be used to estimate these metrics. Furthermore, we will validate the usefulness of these metrics. These metrics (a) provide a foundation for evaluating human-robot systems for performance and operator workload and (b) show how existing human-robot systems can be improved.

1.3 Thesis Organization

The next chapter of this thesis gives an overview of related work to this thesis. In chapter 3, we provide results from a case study on teleoperation that provides a foundation for this thesis. In chapter 4, we discuss interface efficiency and neglect tolerance, their metrics, and other related metrics. Additionally, we describe a measurement technology that can be

used to obtain these measures. The rest of the thesis focuses on validating the measurement technology and the usefulness of the metrics. In chapter 5, we describe and discuss the human-robot systems we will use for validation. In chapter 6, we discuss the environment in which we performed a large user study for this validation. In chapter 7, we report a user study and show how measures of neglect tolerance and interface efficiency for the human-robot systems discussed in chapter 5 are obtained. In chapter 8, we show how neglect tolerance and interface effiency for an interaction scheme can be approximated quickly. Finally, we conclude and discuss future work in chapter 9.

Chapter 2

Related Literature

An article in *IEEE Robotics and Automation Magazine* [10] by leaders in the field of robotics proposed goals for the capabilities of robots in urban search and rescue missions by the year 2050. The article lists six desired levels of competency that included various levels of autonomy. Ultimately, efficient human-robot interaction is desired, in which a robot should be able to carry out many tasks semi-autonomously (some autonomously and others with the help of the human operator). Past research has explored some important elements necessary to obtain these levels of competency. A brief summary is included below.

Human-Centered Robotics. Conway et. al. in [11] presents a taxonomy of human-machine interaction. The taxonomy includes teleoperation, shared control, traded control and supervisory control. Of these four, teleoperation has been studied, perhaps, the most. Sheridan's work [48], however, is the seminal work in the area. Although useful for many situations, pure teleoperation is not desireable for many applications because it is not sufficently tolerant to being neglected by a human operator.

Shared control has become popular in recent years. It has been studied in different forms. Safeguarding (vetoing actions that endanger the robot) is one form [19, 29]. Another form of shared control is when the robot combines operator input with its own assesment of the environment [4, 3, 45].

To avoid undo burden on the operator, traded control has become popular [22]. Point-

to-point and scripted interactions (used in this thesis) are traded control methods. Challenges exist, however, in traded control. One of these areas of concern is the transition of control, both from the human's perspective and the robot's perspective [27].

Sheridan's book [48] contains a good description of supervisory control. A good example of supervisory control for autonomous robotics was RALPH (Rapidly Adapting Lateral Position Handler), a semi-autonomous vehicle [44]. RALPH was able to steer autonomously on large stretches of highway. When RALPH failed, a human took over the steering of the vehicle. Other examples of supervisory control, among numerous others, include [1, 51].

Robot Teams. Human-centered robot teams have become a popular study subject. Much of the work is related to cooperative robotics. Arkin's group has done a lot of work in this area. Such work includes the teleoperation of a group of robots by a single input from an operator [2, 5]. This same idea was used in [26] for telemanipulation. Goldberg's work in [21] is related to this idea. However, instead of having one operator control multiple robots, Goldberg has many operators control one robot. This is important because it provides a foundation for multiple user/multiple robot interactions.

Many-robot systems have been studied in depth. Work on swarms [15, 20] and formation management (platooning) [16, 31] has been done. Problems for which using many small, simple, and cheap agents is more efficient than using a few bigger, complex, and more expensive agents are discussed in [15]. Utah State University recently used this method in the search and rescue competition at RoboCup 2001 [32].

Cooperative robotics has also been combined with other concepts. Balch and Arkin combined cooperative robotics with behavior-based robotics [7]. Parker provides a hybrid approach to distributed cooperation among heterogeneous robots that combines elements of behavior-based robotics and higher-level reasoning [40]. Such hybrid approaches typify much of the current development on autonomous robot design [4, 33].

Interface Technologies. Interface technologies are obviously important to human-robot interactions. Interface assistants have potential in human-robot interactions since

they can lower the workload of the human user/operator [34]. Additionally, personal user interfaces [17], gesture recognition [28, 52], emotive computing [8], virtual reality-based displays [50], and predictive displays [30] can be useful and important to human-robot interactions. Another use of interfaces is having robots learn from human demonstrations [36, 49, 53].

Adjustable Autonomy and Mixed Initiatives. A powerful principle for human-robot systems is the principle of adjustable autonomy. The term adjustable autonomy captures the notion that the autonomy level of a robot can be changed. This principle was developed by Dorais et. al in [14], and is being discussed and used extensively in the literature in various forms [39] and with various kinds of agents [43, 46, 42]. An important principle related to adjustable autonomy is that of mixed-initiatives [18, 42], which poses the question of who has control (a human or a robot) in a system at a given moment as well as who is responsible for initiating control transitions.

Neglect Tolerance and Performance. In this thesis, the idea of neglect tolerance and interface efficiency curves is introduced and used. These curves are similar to performance resource function curves in [54]. In this work, Wickens cites how performance on a task can be measured as a function of resources demanded by the task. Additionally, neglect tolerance and interface efficiency curves call to mind attention-operating-characteristic-plots [41] that cross-plot performance on two independent tasks as a function of attention spent with each task. Related to these ideas is the concept of operator workload. Operator workload can be used to measure the effectiveness of human-robot interactions. Much work on measuring operator workload has been done, but Boer's work relating workload and entropy [35] is of particular relevance to this thesis.

Work Impact. We presented preliminary versions of the neglect tolerance and interface efficiency metrics presented in this thesis in [12, 24]. These metrics were used by Scerri et al. [47] to calculate when an autonomy mode change should be made. They calculated expected utilities to decide whether an agent or a human should should make decisions or perform actions. Other places that this work has been cited include [9, 25].

Chapter 3

Foundational Work - A Case Study

In this chapter, we present a user study that compares a shared-control teleoperation method[1] with traditional manual-control teleoperation. This experiment was performed to demonstrate the way that robot control schemes affect human-robot systems. We first describe the user study, after which we explain criteria and results.

3.1 Experiment Description

The primary task of the user study was for a human to guide a robot via a joystick through a typical building (we used the top floor of the BYU computer science department). The human operator used a joystick to drive a Nomad Superscout II through the environment using shared-control teleoperation and traditional manual-control teleoperation. Four users were selected from various disciplines. None of these subjects had any prior experience controlling the robot. In addition to driving the robot around the building, the users were asked to perform a secondary task (arithmetic problems) that imposed cognitive load on the subjects.

Conditions of the user study are depicted in Figure 3.1. The subject viewed a two-digit arithmetic problem and the robot video stream on separate windows of the same

[1]Described in detail in chapter 5, section 2 of this thesis. The method uses sonar to assist navigation. The term *shared control* is used throughout this thesis to refer to this method.

display. They selected their answers to the arithmetic by clicking on the arithmetic display via the mouse (operated with the left hand) while controlling the robot via the joystick with the right hand. The secondary task used in this experiment is difficult because it requires response-selection attention, occupies working memory, and requires a manual control output with the non-dominant hand (for most subjects). We used a self-paced implementation, meaning the operator chose when a new arithmetic problem was presented (by clicking a button). However, once the arithmetic problem was presented, the operator had only five seconds to answer it.

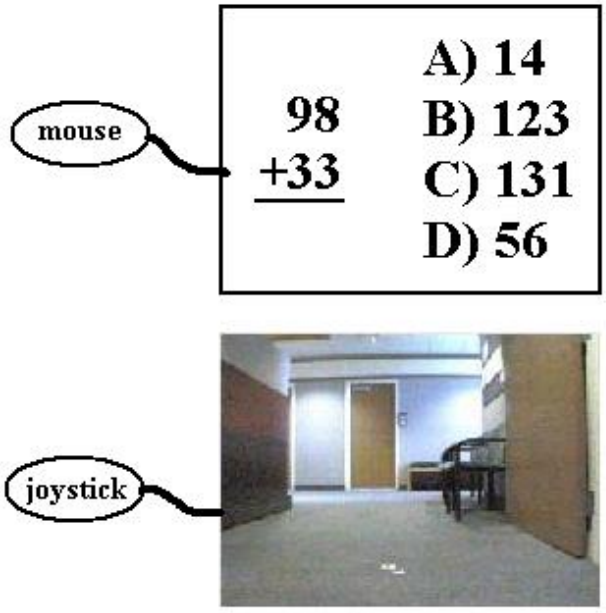

Figure 3.1: *HRI secondary task experiment.*

For each participant, the following steps were followed:

Step 1. The math proficiency level of the participant was determined. Two-digit addition problems were displayed on the screen along with four multiple choice answers (only one being the correct answer). The participant was given five seconds to answer the question. A log of math proficiency was kept. After the participant answered the question, he or she could proceed to a new problem by clicking on a button. This proficiency test lasted for two minutes. If the participant could not successfully

complete 60% of the problems, the difficulty level was reduced to adding a two-digit number to a one-digit number.

Step 2. Next, the participant was trained to guide the robot using a particular autonomy mode. Scheme S was the *S*hared-control teleoperation scheme, and Scheme D was a traditional *D*irect-control teleoperation scheme. In order to not bias results, some participants were trained and tested on Scheme S first, and others were trained and tested on Scheme D first. After completing initial training, the participant was asked to guide the robot through the course as quickly as possible. While doing so, he or she was told to look out for the safety of the robot. Training was completed when the subject had successfully guided the robot through the course one time.

Step 3. The participant was again asked to guide the robot through the course. This time, the participant was also asked to do math problems as he or she controlled the robot. The participant was instructed to guide the robot through the course as quickly as possible, and to answer as many math problems in this time as he or she could, while making sure the robot was safe.

Steps 4–6. The participant repeated steps 2–3 using the other control scheme. That is, if the participant started with Scheme S, then he or she was next trained and tested on Scheme D and vice versa.

3.2 Evaluation Criteria and Results

Ideally, the robot would be able to perform at 100% and the average time-off-task (time not spent on the primary task of driving the robot) would also be 100%. In general, increased time-off-task means increased neglect of that task. In self-paced tasks, this means that for a particular level of neglect, a more neglect-tolerant system will have higher performance; similarly, for a particular level of performance, a more neglect-tolerant system will have more secondary tasks performed. These results for the two autonomy modes (shared-control and manual control teleoperation) are shown in Figure 3.2, which shows robot

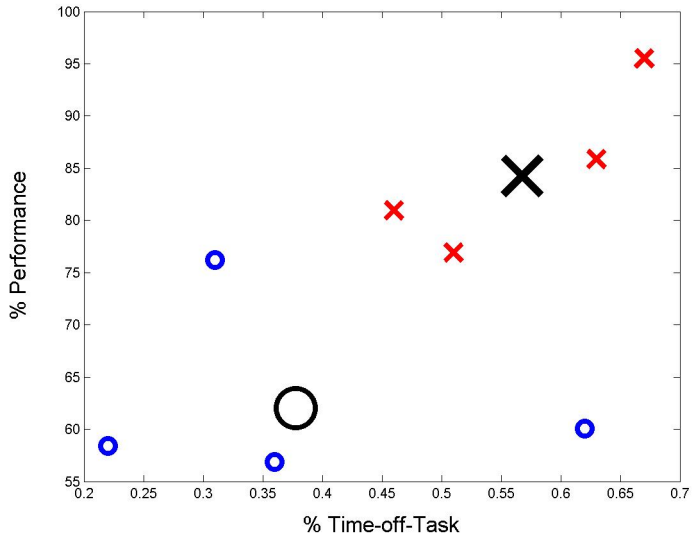

Figure 3.2: *A plot of robot effectiveness verses neglect rate. The vertical axis represents robot performance (as a percentage of maximum effectiveness) and the horizontal axis represents neglect (in terms of percentage of time-off-task). The x's represent results from shared-control, and the o's from manual-control. The large x and large o represent the mean values for each scheme.*

effectiveness verses neglect for the task performed in the experiment. It is interesting to note from this graph that, on the navigation task tested, the shared-control system (represented by the x's) dominates the direct-control sytem (represented by the o's) not only on average across participants (the mean points), but also for each participant.

The relevant measurements for the four subjects are shown in Table 3.1, and are described in detail below.

Neglect Rates. Neglect time is the amount of time spent doing other tasks. Thus, neglect is the time spent solving arithmetic problems[2] divided by the total time of the trial run.

Under the shared-control scheme, the four participants spent an average of 57% of their time doing math problems (and, therefore, neglecting the robot). In contrast, an average of

[2]The time spent solving arithmetic problems is assumed to be the total amount of time arithmetic problems appeared unanswered on the screen.

Shared-Control Results

Participant	A	B	C	D	Ave.
% Neglect	51%	67%	46%	63%	57%
% Performance	77%	96%	81%	86%	85%
# per min.	9.5	18.9	8.9	10.6	12.0
% Correct	74%	98%	94%	66%	83%
Entropy	0.56	0.42	0.51	0.35	0.46

Direct-Control Results

Participant	A	B	C	D	Ave.
% Neglect	36%	31%	22%	62%	38%
% Performance	57%	76%	58%	60%	63%
# per min.	6.4	9.1	3.9	9.8	7.3
% Correct	72%	85%	79%	61%	74%
Entropy	0.72	0.79	0.67	0.63	0.70

Table 3.1: *Results for four subjects (A, B, C, and D) in the user study.* % Neglect *is how much the robot was neglected (the percent of time the operator spent doing arithmetic problems),* Performance *is how efficently the primary task was completed (as a percentage of maximum possible performance),* # per min *is how many arithmetic problems were attempted per minute,* % Correct *is the percentage of attempted arithmetic problems the subject answered correctly, and* Entropy *is the joystick steering entropy; lower values indicate smoother driving. As can be seen, the shared-control system dominates the manual-control system in every category for each participant on the tested task.*

38% of the subject's time was spent neglecting the robot under the direct control scheme. Thus, subjects neglected the robot an average of 50% more using shared-control than direct-control.

Joystick Steering Entropy. We obtain the joystick steering entropy for each participant using the algorithm described in [35]. Entropy ratings range between 0 and 1. A high entropy rating means that joystick movements are choppy and thus indicates that the operator is under high workload; lower entropy ratings indicate that the operator has a lower and more manageable workload.

Under the direct-control system, joystick streering entropy on this task was 0.7, in contrast to 0.46 under shared-control. On average, the direct control system had an entropy just over 50% higher than shared-control. Since entropy is a measure of workload, this indicates that the cognitive workload was substantially higher for direct-control than shared-control. These results are consistent with the 50% increase in time spent neglecting the robot, which is important since workload and neglect rates should have a significant negative correlation.

Primary Task Effectiveness. This is how well the participant did in driving the robot through the course. To keep things simple and objective, the judgement of how well a task was performed is established simply by how much time it takes to get the robot around the building. The distance the robot is required to travel and maximum robot speed dictates that it take at least 170 seconds to get through the course. We base performance off this number: $Performance = \frac{170}{TimeElapsed} \times 100$.

Under shared-control, subjects performed the task at about 85% on average, whereas under direct-control they achieved only 63%. Thus, performance levels for the shared-control system exceeded performance levels of the direct-control system by an average of about 35% which indicates that, for the given world, the shared-control system was much easier to use than the direct-control system.

Secondary Task Effectiveness. This is a measurement of how well the participant performed on the arithmetic problems. Both the number of problems completed per minute

and the math proficiency are important. Since each participant's math abilities differ, only comparisons between how well a participant performed in different control schemes is relevant. We submit that participants should perform better on the secondary task when they have a lower workload imposed by the primary robot control task.

In the experiments, the secondary task results correlate with the results of all the other recorded data for this experiment. The average arithmetic proficiency on the shared-control system was 83%, meaning that subjects answered 83% of the questions correct. By contrast, under direct control, subjects only answered 74% of the questions correct. Thus, the average arithmetic proficiency of shared-control exceeded direct-control by 9%. Additionally, the average number of arithmetic problems attempted per minute increased from 7.3 problems per minute when participants used the direct-control system to 12.0 problems per minutes when participants used the shared-control system. That represents an increase of about 65%, and indicates that subjects were able to turn attention away from the priumary task more frequently with shared-control.

Subjective Rating. Each participant was asked to tell which system was better. The judgement criteria of what is better should be based on a general perception of how the participant felt they did on each scheme.

In the experiments, the participants in the experiment unanimously indicated that the shared-control system was better than the manual-control system.

3.3 Case Study Conclusions

From the case study, we conclude that shared-control teleoperation is more tolerant to neglect than is direct-control teleoperation for the navigation task tested, even though both used the same joystick input and information presentation. We also conclude that measures of workload are important in determining which interaction schemes are best.

While the case study was limited in its scope since it involved testing only two different interaction schemes and was tested on only one simple task, we can reason that varying human-robot systems changes the way that humans and robots interact. Furthermore,

it changes the performance of a system. In the next chapter, we will develop two metrics for measuring how changes in human-robot systems change interactions and system performance.

Chapter 4

Two Metrics for Human-Robot Systems

In this chapter, we will begin by formally defining an interaction scheme. Next, we will discuss *neglect tolerance* and *interface efficiency* in human-robot systems, as well as the related concepts of instantaneous robot performance and world complexity. Then, we will develop metrics for evaluating neglect tolerance and interface efficiency. Finally, we will present a measurement technology for estimating these measures.

4.1 Interaction Scheme

The interaction between a human and a robot is diagrammed in Figure 4.1 for a situation in which a human interacts with a remote robot over a communication network[1]. In the figure, there are two loops involving three different agents: the human, the robot, and the interface between the human and the robot. The top loop involves the human and the interface. Information about the robot and its environment is delivered from the interface to the human. The human processes this information and determines a course of action

[1]When robots are remote from human operators, all information about the world comes to the human via sensors, but when humans and robots are co-located, humans gain information directly from the environment as well.

that he/she believes should be done. This action is communicated to the interface through a control element. The way in which information is presented to a human (remember the gulf of evaluation [37]) and the way in which a human communicates to the robots (remember the gulf of execution [37]) determines the effectiveness of an interface. The bottom loop involves the robot and the interface. The robot receives input from the human via the interface. It then combines this input with its artificial intelligence (in this thesis referred to as its autonomy mode) to act in its world. The robot receives information about the world through its sensors and forwards it to the interface.

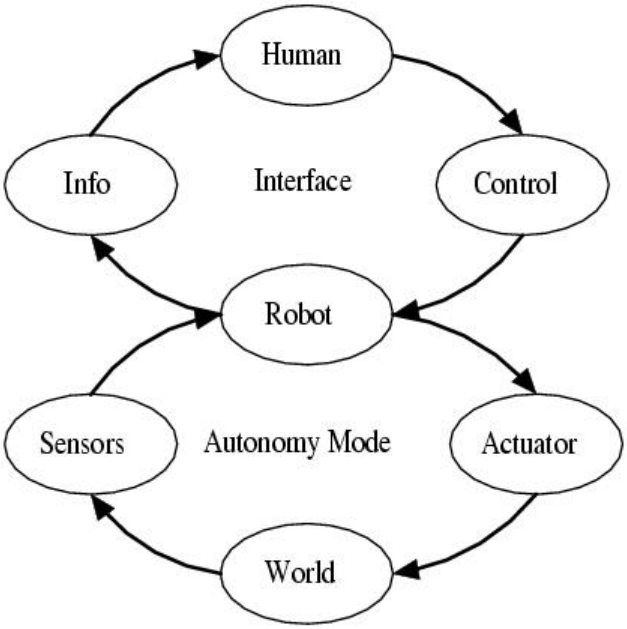

Figure 4.1: *The interface loop and autonomy loop for human-robot interaction.*

A lesson learned from process autonomation is that designing a system without consideration for human factors frequently fails [6], even when humans are well-trained and highly motivated. Therefore, attention should be focused on making the interface and the robot more intelligent in the sense that they support human interactions. Within this context, we define an *interaction scheme* as an *autonomy mode* (bottom loop) of the robot and an *interface* (top loop) between human and robot. The interface consists of an *information-presentation element* through which a human operator receives information from a robot and a *control element* through which the human communicates with the

robot. The interface (information-presentation and control) or the autonomy mode of the robot can be manipulated to create new interaction schemes. Changing an interaction scheme can cause both the frequency and duration of efficient human-robot interactions to change. These changes in interactions determine, in large measure, the effectiveness of an interaction scheme.

4.2 Neglect Tolerance

Neglect tolerance is a measure of a robot's autonomy mode. This term is used to refer to the way that a robot's performance changes when it is neglected by humans (i.e., when human attention is focused elsewhere). As a general trend, as neglect increases, robot performance decreases. How robot performance decreases depends on the interaction scheme that is being employed. Figure 4.2 conceptualizes how one might expect neglect to affect robot performance for different kinds of interaction schemes. In the figure, the performance of a robot using a teleoperation interaction schemes degrades quickly as the human neglects the robot. The performance of an autonomous robot[2] does not tend to degrade much over time, although its peak performances usually would not be expected to be as high as a teleoperated robot.

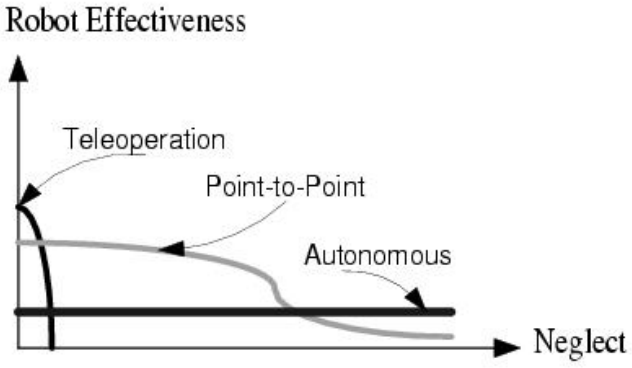

Figure 4.2: *Hypothesized neglect tolerance (time-off-task) for interaction schemes with various autonomy modes for a world of constant complexity.*

[2]We live in a world in which no robot is completely autonomous due to reprogramming, re-enginerring, etc., but we use the term to denote a robot that generally runs autonomously.

Teleoperated and fully autonomous modes lie on the extremes of human-robot interactions. There exist a large number of autonomy modes that require different degrees of interactions and are represented in Figure 4.2 by a point-to-point scheme in which a robot is a given a command, such as "turn left at the next intersection," and is then expected to carry that command out autonomously, after which more interactions are required.

4.3 Interface Efficiency

Interface efficiency is a measure of the effectiveness of an interface. When a human operator's attention is turned to a robot [3], we would expect the robot's performance to change, hopefully for the better. The way that the robot's performance changes during servicing depends on the interaction scheme being employed. The interface of an interaction scheme, through its information-presentation and control elements, affects the time it takes for a human to gain relevant situation awareness, decide on a course of action, determine the inputs to give to the robot, and then communicate those inputs to the robot.

A poorly designed information-presentation system may cause the process of gathering information to become a task in and of itself. Consider an extreme example in which information about obstacles around a robot is communicated to the human operator via text. In such a situation, the human operator must read the information and create a mental representation of the world around the robot (which could take considerable time) before generating a plan about how to deal with the obstacles. Thus, an interface from which information extraction is difficult extends the time it takes for the operator to switch from one task to another.

Similar to the way that information-presentation to the human from the robot can change the characteristics of interaction between a human and a robot, the way of giving information to the robot from the human can also change the characteristics of interaction. As an example, consider controlling an airplane with the movements of a mouse. It may be difficult for the human to determine in what ways mouse movements translate into

[3] We use the term *servicing the robot* to describe this action.

airplane movements in the air. In general, control should be intuitive to the human operator, otherwise the process of presenting information to the robot also becomes a task in and of itself.

Figure 4.3 shows how interface efficiency (time-on-task) could hypothetically affect the performance of a robot for different interaction schemes. The figure expresses the idea that different interaction schemes affect the way that the performance of a robot changes during interactions.

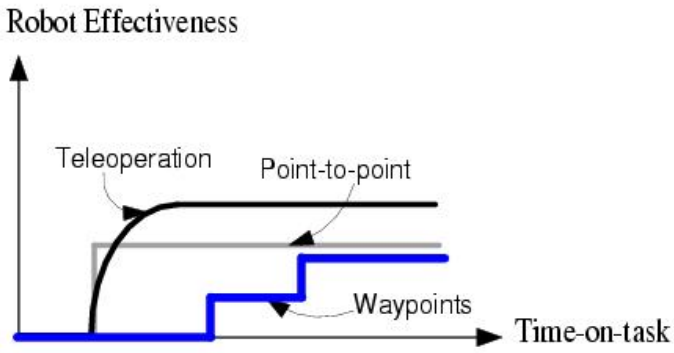

Figure 4.3: *Qualitative representations of interface efficiency for various presentations of information.*

4.4 World Complexity

Up to this point, we have ignored the effects of world complexity on neglect tolerance and interface efficiency. Consider, however, the two worlds shown in Figure 4.4. It seems obvious that it would be easier for a robot to navigate through world b than to navigate through world a. Thus, the complexity of the robot's environment affects robot performance. Interaction schemes that are designed for a particular level of world complexity may not perform well for other world complexities. Intuitively, robot performance generally decreases as world complexity increases.

Some interaction schemes scale better to the effects of world complexity than do others. An interaction scheme that scales well to complexity (i.e., robot performances changes little

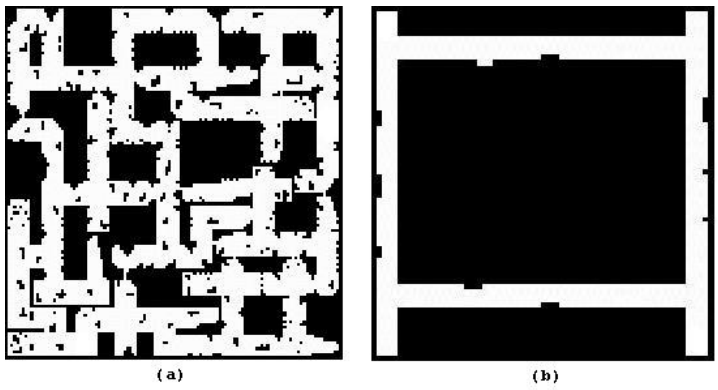

Figure 4.4: *Two worlds with differing world complexities.*

with changing world complexity) is said to be *complexity tolerance*.

4.5 Combining Neglect Tolerance and Interface Efficiency

The performance of a semi-autonomous robot declines as human attention is spent on other tasks and/or the complexity of the world increases. Additionally, effective human-robot interactions should cause robot performance levels to remain increase. This implies that interactions must be frequent enough and last long enough to maintain sufficiently high robot performance levels.

To illustrate this, consider Figure 4.5. In the figure, (moving from left to right along the horizontal axis), a robot begins at performance level zero (or from stand-still). A human operator begins to interact with the robot (Task 1). When this occurs, performance is modeled as an interface efficiency curve (see Figure 4.3). When a human terminates the interaction and turns his/her attention to another task (Task 2), the robot's performance level begins to deteriorate and is modeled as a neglect tolerance curve (see Figure 4.2). Before the robot's performance level drops below acceptable levels, the human must again turn its attention to the robot (Task 1) and interact with the robot. Because of time for context switches, these interactions must begin some time before a robot's expected performance declines below an acceptable level.

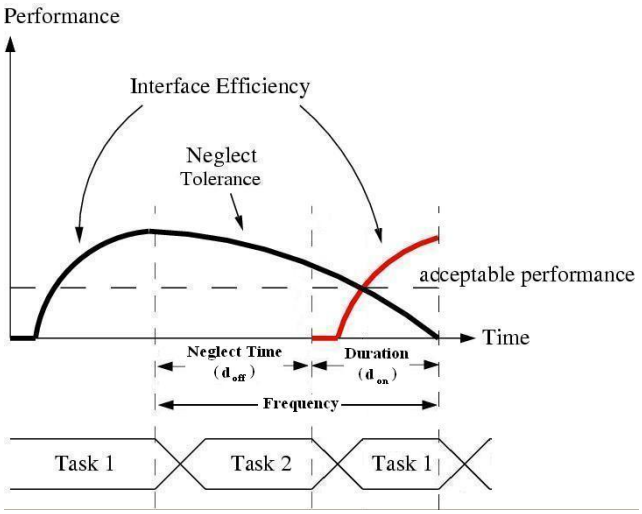

Figure 4.5: *The average frequency and duration of human-robot interactions can be determined by a combination of interface efficiency and neglect tolerance.*

Thus, Figure 4.5 shows how acceptable interaction rates can be derived from the neglect tolerance and interface efficiency measures. An *acceptable interaction rate* is a frequency and duration of interactions such that expected robot performance does not drop below a certain threshold. The figure shows that the duration of an interaction should be d_{on} in a situation in which the frequency of the interaction is $d_{\text{on}} + d_{\text{off}}$. Varying the minumum acceptable performance level (MAPL), or the threshold, changes the acceptable interaction rate. Additionally, it also changes the average performance of the robot.

As an example, consider lowering the MAPL in Figure 4.5. This would cause d_{off} to become larger, which in turn could change d_{on}. Thus, both the frequency and duration of the acceptable interaction rate would change. However, lowering the MAPL would also cause the robot's average performance to decrease since the robot would spend more time performing at lower performance levels. Similarly, increasing the MAPL in the figure would make d_{off} smaller, as well as affecting d_{on}, and would cause the robot's average performance to increase.

4.6 Mathematical Measures of Usefulness

The performance of a robot employing interaction scheme, π, is defined by a random process indexed by time, t, world complexity, c, and the duration of the previous lapse in interactions, t_N (neglect time), between a human and the robot. More formally, the performace p of a robot is defined as

$$p = \mathcal{V}(\pi, \mathcal{T}; t, c, t_N) \tag{4.1}$$

where $\mathcal{T}$ is the task being performed and c is an estimate of world complexity (obtained using a complexity metric; see section 4.7.2).

In equation (4.1), we used the generic time term t. However, a different time variable is used by the neglect tolerance metric than by the interface efficiency metric. The neglect tolerance metric uses time-off-task t_{off}, which denotes the time elapsed since the robot was last serviced. The interface efficiency metric uses time-on-task t_{on}, which denotes the time elapsed since servicing began. Thus, if the robot is currently being serviced, then t is t_{on}. If the robot is being neglected, then t is t_{off}. Therefore, equation (4.1) becomes

$$p = V(\pi; t, c, t_N) = \begin{cases} V_S(\pi; t_{\text{on}}, c, t_N) & \text{if being serviced} \\ V_N(\pi; t_{\text{off}}, c) & \text{otherwise} \end{cases} \tag{4.2}$$

where the variables are defined as before. Thus, $V_S(\pi; t_{\text{on}}, c, t_N)$ is a measure of the interface efficiency of π and $V_N(\pi; t_{\text{off}}, c)$ is a measure of the neglect tolerance of π. Notice that neglect tolerance is not dependent upon t_N. This is because we assume that interactions will always bring expected robot performance up to peak levels, independent of the previous neglect time, which means that $V_N(\pi; t_S = 0, c)$ is independent of t_N. For simplicity, we often refer to $V(\pi; t, c, t_N)$, $V_S(\pi; t_{\text{on}}, c, t_N)$, and $V_N(\pi; t_{\text{off}}, c)$ as $V(\pi)$, $V_S(\pi)$, and $V_N(\pi)$ respectively.

The random process $V(\pi)$ contains a lot of information about the interface efficiency and neglect tolerance of the interaction scheme π. First, $V_S(\pi)$ shows what happens to robot performance levels when a robot interacts with a human. Answers to questions such as, "How long does it take for the robot to return to peak expected performance levels?"

are available in $V_S(\pi)$. Second, $V_N(\pi)$ shows what happens to robot performance levels when a robot is neglected. As an example, $V_n(\pi)$ answers the question, "How long does it take for the robot's expected performance level to drop below 50% of peak expected performance levels?"

$V(\pi)$ also indicates the average frequency and duration of interactions that should take place between a human and a robot for any minimum acceptable performance level. This information is obtained by using the technique shown in Figure 4.5. Additionally, the average performance of a robot employing interaction scheme π can be estimated using these acceptable interactions. Such calculations can be used to identify the strengths and weaknesses of interaction schemes.

4.7 A Note on Related Metrics

We mentioned previously that measures of neglect tolerance and interface efficiency are dependent on two metrics. These two metrics are *instantaneous performance* and *world complexity*.

4.7.1 Performance Metrics

In the context of this paper, the term *performance metric* is, perhaps, a little awkward. The performance metric discussed in this session is necessary to estimate the neglect tolerance and interface efficiency of an interaction scheme nonparametrically, and should not be confused with the performance prediction that the interface efficiency and neglect tolerance metrics perform.

In this paper, the performance of a robot is the *work* done by a robot with respect to that robot's, or, perhaps, some other object's, *capacity* to perform work. Therefore, robot *performance* is simply the ratio $\frac{work}{capacity}$. Note that performance can be either positive or negative and can take on any value in the range [-1, 1].

It can be difficult to measure the performance of a robot on an ongoing (continuous) basis. In many instances, it is very easy to measure the performance of a robot after it

has completed a task, but it is difficult to measure a robot's performance while the task is in progress. In this thesis, however, we assume that performance can be measured or estimated continuously, and leave situations in which performance can not be measured or estimated continuously to future work.

The way that a robot's performance is measured can be different for each task. The neglect tolerance and interface efficiency metrics require only that at any given time, an estimate of the *instantaneous performance*[4] of the robot be available. This implies that we must be able to estimate *instantaneous work* and *instantaneous capacity* for work was well. Assuming we have these estimates, we have

$$ip_t = \frac{iw_t}{ic_t} \tag{4.3}$$

where ip_t is the instananeous performance at time t, iw_t is the instantaneous work performed at time t and ic_t is the instantaneous capacity for work at time t.

Robot performance for a whole task should be given by

$$performance = \sum_t ip_t \tag{4.4}$$

where ip_t is given by equation (4.3). However, since ip_t is only required to be an estimate, equation (4.4) is not required to be perfect in practice. We do, however, note, that the neglect tolerance and interface efficiency metrics are most effective when the right side of equation (4.4) does indeed equal, or approach, the left side of the equation.

To summarize, for this thesis, a performance metric is valid if it is able to estimate the instantaneous performance of a robot at each time t. No specification, at this point, is made about how performance is measured other than it is some ratio of work performed and work capacity. Later, we will present one candidate technique for specifying these measurements.

[4]We use the term instantaneous performance to indicate the performance of a robot over a small time interval.

4.7.2 World Complexity Metrics

Like performance, world complexity is also difficult to measure. World complexity is, in fact, somewhat subjective. A world can be considered relatively simple or very complex, depending on the task being performed. Additionally, to one set of robot abilities a world may be considered complex, whereas to another set of robot abilities the same world may be considered quite simple.

This being said, world complexity metrics are an important part of the neglect tolerance and interface efficiency metrics. However, we do no specify how world complexity must be measured since such a specification would be practical for only one task. We only say that a way of estimating the complexity of a robot's world is required. How this is done is left to the system designer. Good world complexity metrics, however, tend to assign high complexity estimates to environments that make a task difficult for a robot to perform, and low complexity estimates to environments that make tasks easy for a robot to perform.

In section 7.1, we provide an example of how world complexity can be measured for a navigation task. We also provide analysis of this world complexity metric in section 7.3.5.

4.8 Measurement Technology

Previously, we discussed the random process $V(\pi)$, which is a measure of the neglect tolerance and interface efficiency of the interaction scheme π. In this section, we discuss how this random process can be estimated nonparametrically by designing and performing user experiments that suffiently sample the domain space of the random process $V(\pi)$.

The domain of the performance random process consists of time, t, neglect time, t_N, and world complexity, c. As we discussed in the previous section, time, t, is separated into time-on-task, t_{on}, and time-off-task, t_{off}. To sufficiently sample the time domain, we need users to spend time both servicing and neglecting a robot. To do this, we require that the user perform secondary tasks in addition to performing the primary task of servicing the robot. In this way, the robot will be neglected when the user focuses his/her attention

on the secondary task. To sample the neglect time domain thoroughly, we must vary how long the robot is neglected. This is achieved by varying the length of time that a user must perform a secondary task before returning to service the robot. The complexity domain can easily be sampled by simply performing the user experiments in worlds of various complexities.

Since the domain of the random process is continuous, it must be discretized so that it can be sampled sufficiently. Each data sample from the user study (which is a robot's estimated performance at a particular time, neglect time, and world complexity) is placed in a bin defined by the discretized domain to form a nonparametric estimate of the random process $V(\pi)$.

Even after discretizing the domain of the random process, an impractical number of test subjects must be used in order to sufficiently sample the domain in this manner. This is because each world complexity estimate is a sample from an unknown distribution. We address this problem by applying a gaussian filter to the data. Such an approach is justified by the central limit theorem. A large number of test subjects must still be used, but not nearly as many.

To summarize, the measurement technology requires that humans and robots must actually interact in real systems to measure the neglect tolerance and interface efficiency of these systems. Secondary tasks must also be used to thoroughly sample the domain space of the random processes.

4.9 Chapter Summary

In this chapter, we developed metrics for neglect tolerance and interface efficiency in human-robot systems. These metrics identify the interactions that should occur to maintain acceptable robot performance levels and can predict the performance of a robot on a given task given these interactions and the world's complexity. These metrics have potential use in a number of areas, including human-robot system design.

Thus, given a random process $V(\pi)$ for a given task, the performance of the robot

is known (probabilistically through a random variable). We need only obtain V for each interaction scheme π and each task $\mathcal{T}$. This is discussed in chapter 7. Before doing that, we will describe some interaction schemes that we will use to show how the neglect tolerance and interface efficiency metrics can be approximated.

Chapter 5

Developing Interaction Schemes with Varying Degrees of Autonomy

In this chapter, we describe three interaction schemes that we will use to show how the metrics and measurement technology described in the last chapter work. The interaction schemes are built for the purpose of performing a navigation task in indoor environments.

The three interaction schemes have autonomy modes with varying degrees of autonomy. While the control element of the interface for the three interaction schemes vary, the information-presentation element is largely the same for all three interaction schemes. The three interaction schemes are described in Table 5.1. The table lists the autonomy mode and the interface used for each interaction scheme.

In the rest of this chapter we describe the three interaction schemes, beginning with the information-presentation element and the shared control algorithm that is used by each of the schemes. We then describe each of the interaction schemes separately.

5.1 Information-Presentation Element

In this section, we describe the information-presentation element used for each of the interaction schemes used in this thesis. As we stated previously, the information-presentation element is part of the interface of an interaction scheme. It presents information about

Interaction Schemes

Interaction Scheme	Autonomy Mode	Interface	
		Information	Control
Teleop	Shared-Control Teleoperation	Typical; joystick vector icon	Joystick
P2P	Point-to-Point Mode	Typical; arrow indicator	Mouse (Click on Video Feed)
Scripted	Scripted Mode	Typical; goal icons	Mouse (Click on Map)

Table 5.1: *Shows the three interaction schemes used to show how the interface efficiency and neglect tolerance metrics are obtained and used. The information-presentation element (*Information*) has the attribute* Typical *listed in each scheme to indicate that the same basic information-presentation element is used for each scheme with the exception of feedback given of inputs/commands sent from the human to the robot.*

the robots in the system (and their environments) to human operators.

Figure 5.1 shows a snap shot of the graphical user interface used to present information about the robots to a human operator using the scripted interaction scheme. The main portion of the GUI shows a topographical map of the robots' world. On the topographical map, the position of each robot is marked by a triangular objects (the objects are shown bigger when the robot is currently begin serviced) and the robots' goals are marked by a rectangle (the color of the goal matching the color of the robot it corresponds to). On the right hand side of the GUI various system indicators are listed, such as team performance, time-based workload, time elapsed, number of goals found, and the operators math proficiency. Directly below the topological map of the world, the performance of each individual robot is shown, as well as an estimate of the robot's current world complexity.

At the bottom of the GUI, the sensory information of the robot currently being serviced is displayed. The color of the background of this window corresponds with the color of

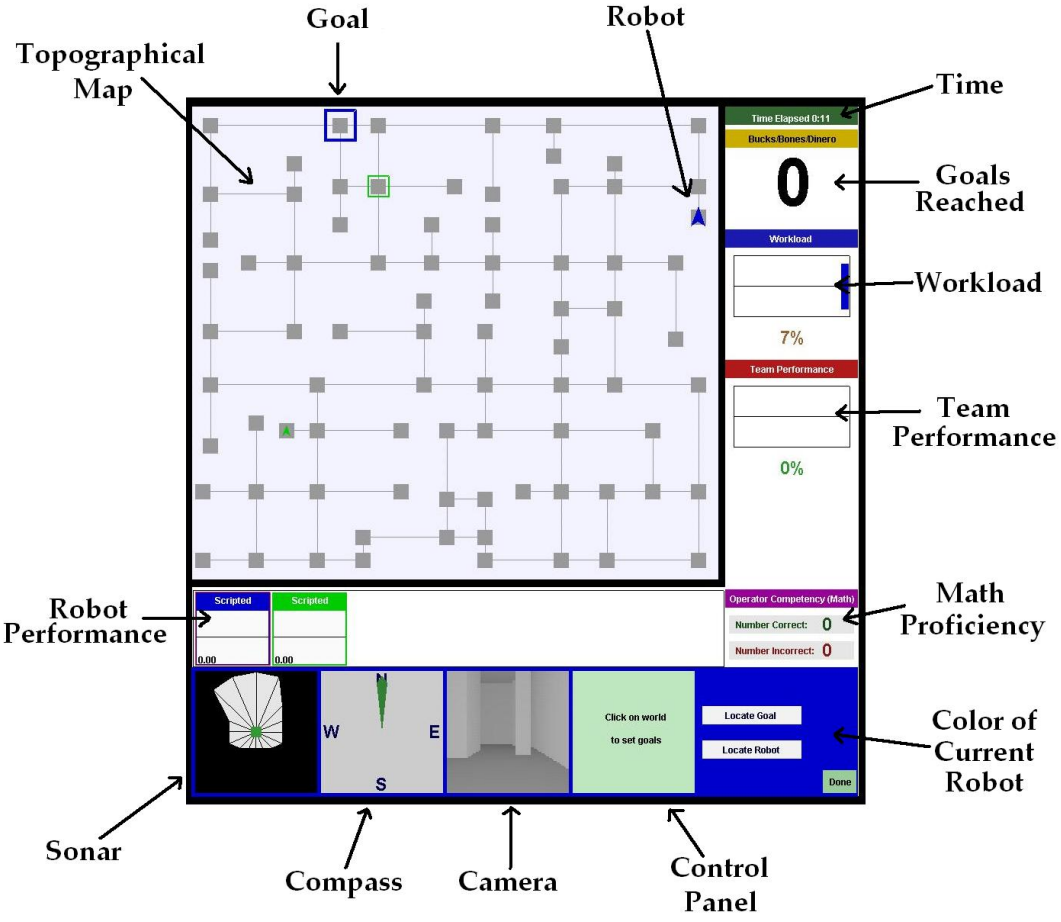

Figure 5.1: *The Graphical User Interface (the information-presentation element of the interface) for the three interaction schemes used in this thesis.*

the robot shown in the global topological map of the world. The robots we used for the experiments presented in this thesis are equiped with a sonar ring of sixteen sonar which encompass the robot, a compass, and a black and white video camera. Graphics of these sonars are all displayed. First is a graphic of the sonar readings, followed by the compass reading, followed by the video display.

Also in this bottom portion of the GUI is the control panel, which is interaction scheme dependent. At the lower right hand corner are various buttons: a *locate robot* button (used to help the operator locate the current robot on the global map), a *locate goal* button (used to help the operator locate the current robot's goal position), and a *done* button (used to terminate the current task and move on to another task; the interface decided the task that was to be executed next).

When no robot is currently being serviced, the current robot's sensor information, the control panel, and the control buttons are replaced by an arithmetic display. In this display is a two-digit addition or subtraction problem, with four multiple choice answers. The operator answers the problem by clicking on one of the answers.

5.2 Shared Control

Each interaction scheme's autonomy mode uses the shared-control algorithm described in this section. The algorithm requires as input a robot centered vector that indicates the general desired direction the robot should go. The algorithm outputs a vector indicating the direction the robot should move, provided that the algorithm can find such a vector which will not cause the robot to collide with objects the robot can see (with its sonar). If no such vector is found, the robot is simply directed to spin in place towards the direction with the nearest open space (as defined by the robot's sonar).

Our approach to shared-control teleoperation uses a variant of potential fields. In the algorithm, the vector of each sonar is associated with a behavior. Sonars that measure nearby obstacles return repelling behaviors, and sonars that measure open spaces return attracting behaviors. More specifically, sonar distances are classified into three categories:

repelling, neutral, and attracting. If a sonar returns a distance greater than a pre-defined *safe distance* (65 inches in our implementation) then the corresponding behavior is categorized as an attracting behavior. If a sonar returns a distance less than a pre-defined *risk distance* (40 inches in our implementation) then the corresponding behavior is categorized as a repelling behavior. Otherwise, the corresponding behavior is categorized as a neutral behavior.

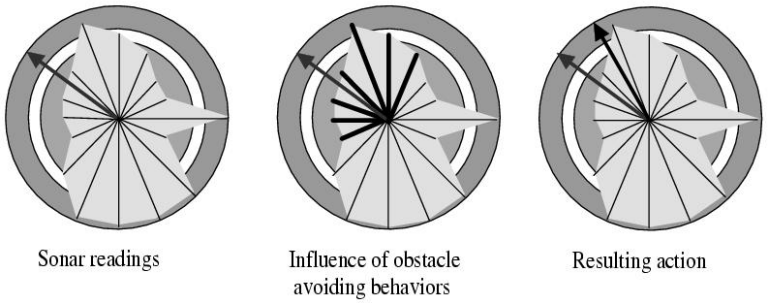

Figure 5.2: *A graphical depiction of the algorithm for a robot positioned in a hallway with a open door on the robot's right. Raw sonar readings (left) are translated into relevant behaviors (middle) and combined with the human input to produce the actual robot action (right).*

Additionally, each sonar (or behavior) is assigned a weight based on how relevant it is to the direction that the robot is asked to move. This is illustrated in Figure 5.2. In the figure, the human tells the robot to go forward and left (see the image on the left). Sonar readings that are relevant are identified (see the image in the middle). As a rule, the seven sonars that represent vectors closest to the input vector are considered relevant and are weighted (based on how closely their vectors are like the input vector), while the other sonars have zero weight in the calculation. The output vector is then obtained by summing these weighted vectors. In the example shown in the fiugre, the robot still moves forward and left, but does not move as far to the left as suggested by the human.

More formally, weights are assigned to the behaviors in the following way. Let $A = (a_0, a_1, a_2, a_3)$ and $R = (r_0, r_1, r_2, r_3)$. Let the *center sonar* be the sonar with the angle closest to the angle of the input vector and let n be the number of sonars between sonar i

and the *center sonar* plus one. If n is greater than three, then sonar i's weight is zero. If sonar i corresponds to an attracting vector, then its weight is a_n. If sonar i corresponds to a rejecting vector, then its weight is r_n. Otherwise, its weight is zero. For our interaction schemes, we used $A = (0.8, 0.7, 0.4, 0.1)$ and $R = (-1.0, -1.0, -0.45, -0.1)$ and 1.4 as the weight of the input vector (behavior). The output vector o of the algorithm is given by

$$o = (iw)(iv) + \sum_i w_i v_i$$

where iw is the weight to the input vector, iv is the input vector, w_i is the weight assigned to sonar i and v_i is the vector formed by sonar i.

The direction indicated by the output vector o will usually but not alway cause the robot to avoid obstacles. For this reason, we implement safeguarding control on the output vector. Safeguarding is done by simply calculating where the robot will be after a certain time t. If the robot will depart from the open region defined by the sonar (see Figure 5.2), then movement by the robot is stopped, and the robot spins in place towards the direction with the nearest open space (again, defined by the sonar). Otherwise, the robot moves in the direction pointed by o.

Various autonomy modes can be obtained from this algorithm by simply altering the way that the input vector is obtained. We specify the way that these input vectors are obtained for each interaction scheme in the following sections.

5.3 Teleoperation Interaction Scheme

In this section, we describe the remaining details of the teleoperation interaction scheme (*Teleop*) used in this thesis.

5.3.1 *Teleop*'s Interface

The control element of the interface of the *Teleop* interaction scheme is a Microsoft Force Feedback II Joystick (no force-feedback was given). The joystick was mapped to try to emulate the feeling of driving a car. Figure 5.3 depicts the mapping of vector direction to

wheel movements. For example, moving the joystick forward to the right causes the left wheel to turn faster than the right, thus causing a right turn.

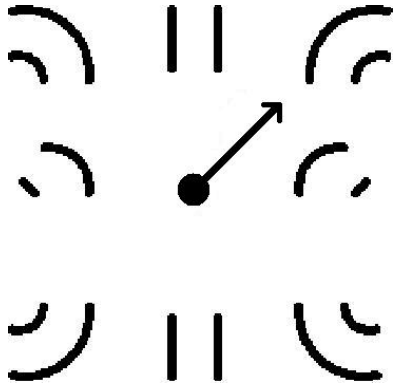

Figure 5.3: *Depicts the mapping of the direction of the output vector (robot centered) to wheel movements on the robot. The robot is located in the center of the figure. As an example, the sample vector shown in the figure would cause the robot's wheels to follow the tire tracks in the upper righthand corner. So the robot would move forward and to the right.*

The information-presentation element of the teleoperation interaction scheme is identical to that shown in Figure 5.1 except for the control panel. For *Teleop*, the vector supplied to the robot via the joystick was depicted graphically in the control panel. This helped the operator to understand the direction they were telling the robot to go.

5.3.2 *Teleop*'s Autonomy Mode

The autonomy mode for *Teleop* is identical to the autonomy mode used in the case study described in chapter 3. The input vector to the shared-control algorithm is simply the angle and magnitude of the joystick. The magnitude of the joystick (how hard the operator pushes on the joystick in the desired direction) changes the speed that the robot moves.

Teleoperation autonomy modes are potentially very complexity tolerant because the human operator makes the high level decisions (and is thus able to reason through complex situations). However, if the information-presentation element does not provide the human operator with adequate awareness of the robot's environment, increased world complexity

can lead to decreased robot performance.

5.4 Point-to-Point Interaction Scheme

In this section we describe the remaining details of the point-to-point interaction scheme (*P2P*) used in this thesis.

5.4.1 *P2P*'s Interface

The control element of the interface of *P2P* is a mouse. The operator simply clicks on buttons telling the robot what to do at the next intersection (such as *go straight*, *turn left*, and *turn right*). Additional buttons are also provided for control (*spin left*, *spin right*, *stop*, and *back up*). These buttons, shown in Figure 5.4, are placed in the control panel of the information-presentation element. As feedback, the button currently employed is made darker to let the operator know what to expect from the robot. After the robot believes that it has fulfilled the current command, it notifies the operator by switching back to the *go straight* action.

5.4.2 *P2P*'s Autonomy Mode

The autonomy mode used for this interaction scheme employs the shared-control algorithm described previously. It differs from the teleoperation mode by the way that the input vector to the shared-control algorithm is obtained. The operator clicks on a button to indicate the command that he/she wants the robot to carry out at the next intersection or at its next opportunity (such as *go straight*, *turn left*, or *turn right*).

To traverse a hallway, a vector pointing straight ahead (robot centered) is input into the shared-control algorithm. If the command is to turn right (or left) the next chance it gets, the robot looks for an opening to its right (or left). An *opening* is when the sum of three adjacent sonar exceeds 180 inches[1]. When the robot finds an opening, it inputs

[1] A single sonar may contribute no more than 72 inches.

Figure 5.4: *Graphic of the buttons that an operator pushes when the* P2P *interaction scheme is being employed. The operator tells the robot what to do at the next intersection (go straight, turn right, turn left). Additional buttons are provided to stop the robot, have it spin in place, or move backward. The currently employed button is darkened on the display to help the operator know what to expect from the robot. This display replaces the control panel shown in Figure 5.1.*

a vector 45° to the right (or left) of the robot's straight ahead position. This causes the robot to turn in the desired direction (according to the mapping shown in Figure 5.3). The robot must decide when it has turned the correct amount. When it believes that it has[2], it resumes a *go straight* behavior and communicates to the interface that it is now performing a *go straight* behavior.

There are several observations to be made about this autonomy mode. First, the algorithm assumes that the robot is correctly oriented in the hallway so that when a robot is asked to traverse a hallway, the straight ahead input vector will direct a robot straight down the hall. For this purpose, the operator may need to use the *spin left* and *spin right* buttons to correctly orient the robot. It should be noted that the algorithm is robust enough that the orientation of the robot need only be very general, and doesn't require exactness.

Second, hallway clutter can cause the orientation of the robot in the hallway to get turned around, since the shared-control algorithm will seek to avoid this clutter. Since a

[2]This is determined by dead reckoning.

history of directions is not kept by the robot, if the robot must turn too much to go around an obstacle, it could cause the robot to turn around and move in the opposite direction. The robot is not equipped with a way of knowing it is now going in the wrong direction. Thus, this autonomy mode does not promise to be complexity tolerant, since clutter can certainly contribute to the world complexity measure in such a task as indoor navigation.

Third, when frequent turns are necessary, this algorithm requires more frequent human-robot interactions (because after each intersection is passed the robot generally needs new input from the user). Since the branching factor (number of intersections per area) is part of indoor navigation, it should contribute to a good world complexity measure. Again, this autonomy mode does not promise to be complexity tolerant.

5.5 Scripted Interaction Scheme

In this section, we discuss the scripted interaction scheme (*Scripted*) used in this thesis. We first discuss its interface, followed by its autonomy mode.

5.5.1 *Scripted*'s Interface

The information-presentation element of *Scripted*'s interface is the same as it was for the other interaction schemes with two exceptions. First, the GUI has a statement in the control panel telling the operator to click on the global map to set goal markers for the robot to traverse. Second, when the operator clicks on the global map, an icon appears that indicates where a goal marker has been set on the map. Each goal marker is numbered according to the order it was placed on the screen, which indicates the order that the robot will traverse the goals. Figure 5.5 shows a situation in which four goal markers have been placed on the map of the world directing a robot to its goal.

The control element of the interface for this scheme uses a mouse. An operator issues commands to the robot by simply clicking on the topological map. Any number of goal markers can be set. The robot traverses the goal markers in order. The next goal marker for the robot to reach is labeled "1", the next is labeled "2", etc. When the robot reaches

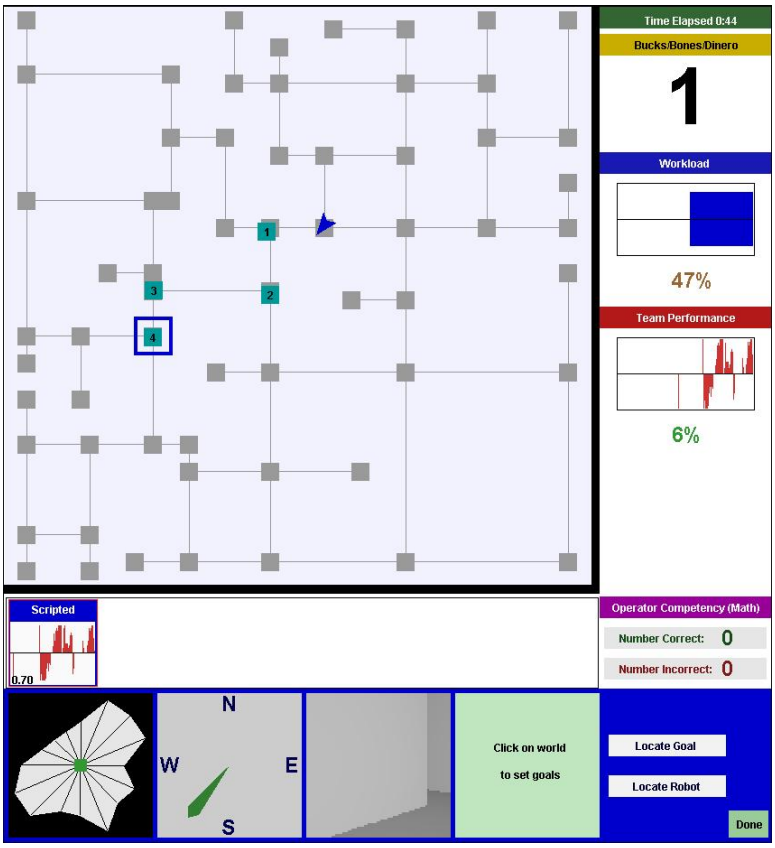

Figure 5.5: *Shows a situation in which four goal markers have been placed on the map of the world (by the operator) to direct a robot to its goal.*

goal marker "1", goal marker "2" is relabeled "1", goal marker "3" is relabeled "2", etc. A goal marker can be deleted by clicking on the goal marker and dragging it off of the global map. Similarly, a goal marker's location can be moved by simply clicking and dragging that icon to the desired location.

5.5.2 *Scripted*'s Autonomy Mode

As mentioned previously, a robot employing *Scripted* traverses the goal markers set by the human operator. The robot accomplishes this by again using the shared-control algorithm defined previously. Again, only the method of obtaining the input vector is different from the previous autonomy modes.

The input vector is obtained by using the goal marker labeled "1." The relative vector, V_g, between the goal marker and the robot is calculated. This vector is compared to the vector V_d (which is a vector indicating the direction that the robot is facing). If the angle between these vectors is greater than 45°, then the robot simply spins in place (in the direction that decreases the angle between the two vectors). If the angle is less than or equal to 45°, then the robot simply inputs V_g into the shared-control algorithm. If there is no goal marker placed, the robot stays still.

Note that the only effect that branching factor has on this algorithm is that the human operator must drop more goal markers. This, in many instances, does not take a lot of time, so it would be expected that this autonomy mode would be relatively tolerant to this kind of world complexity. However, if obstacles are between the robot and the next goal marker, it is possible that a robot could get "stuck." Thus, this interaction scheme is not necessarily tolerant to this kind of world complexity. Operators are required to check for themselves whether "blocking" obstacles will be encountered.

It should be noted that this interaction scheme is somewhat ineffective when the topological map of the world is unknown, or when the map is not very good (whereas the previous interaction schemes could still be reasonably effective, it would appear). Thus, it is important to reemphasize that both interface and robot autonomy mode affect the

efficiency of an interaction scheme.

5.6 Summary

In this chapter, we discussed the interaction schemes that are used in the user studies presented in this thesis. In the next chapter, we describe the environment used in the studies.

Chapter 6

Validation Environment

We used simulated worlds to validate the usefulness of the metrics and measurement technology described in chapter 4. In this chapter, we first discuss why we use simulated worlds. We then describe the simulator we used.

6.1 Why Simulated Worlds?

Using simulated worlds for a user study rather than real worlds offers several advantages. First, it allows that worlds with varying degrees of complexity can easily be created. In the real world this would be more difficult to do since it would require that worlds be constructed out of physical objects. This, in and of itself would not be extremely difficult, except that we would have to create 20 such worlds and have them always fully operational so as to not compromise the experiments[1]. Thus, using simulated worlds simplifies the experimental process significantly.

Second, the reliability of robotic hardware becomes a critical issue since a large number of experiments must be performed. In chapter 8, we describe ways that measures of neglect tolerance and interface efficiency can be estimated using a small number of experiments, and thus less wear-and-tear on robots occurs. However, it is necessary to perform a

[1]See the next chapter. It would compromise the experiments if this was not done because it would not allow for the random selection of worlds.

large number of experiments to show that the random process can be estimated correctly using only a small number of experiments. In the future, robot hardware technologies will, hopefully, become more robust. For the time being, however, such wear-and-tear on the robots significantly complicates the experiments (since robots can break down in the middle of an experiment). Therefore, we use simulated robots.

6.1.1 Restrictions

Despite the above arguments for the use of simulated worlds in these experiments, they are not valid if experiments in the simulated world compromise the validity of the results. In this thesis we seek only to validate the usefulness of the neglect tolerance and interface efficiency metrics. The results should show how and why the metrics are useful, but the actual random processes obtained should not be used in the real world.

The simulated worlds are sufficient to validate the metrics only if they generate results similar to those that would be obtained in the real world. In this context, "similar" means that the same general trends exist in both the simulated and the real world. The simulated world should show the same trends exibited in user workload and robot performance.

6.2 The Simulator

In this section, we describe the simulator that we have developed. We will describe the simulator and then show that it does indeed produce the same trends in user workload and robot performance as does the real world.

6.2.1 Description

The simulated robots were designed to mimic the Nomad Superscout used in the user study described in chapter 3. They are equiped with 16 sonar encompassing the robot, a compass, and a black and white video image. The way the robot responds to input is also designed to be identical, with little exception. The biggest differences between the simulated robot

and the real-world robot are in noise; in movements and sensory information. No noise is introduced into the simulated worlds. Another way that the simulated world is different from the real world is that network delays are greater in the real world.

It is acknowledged that these differences can be significant. For this reason, simulated results do not translate directly to the real world. As mentioned previously, however, a simulator need only show that similar trends exist in order for the measurement technology to be valid for these experiments.

6.2.2 Trends

To validate that the simulated world produces acceptable trends, we ran the same experiment described in chapter 3 in a simulated world made to look similar to the real world. Table 6.1 shows the results of the experiments for seven users.

Workload. Time-based workload[2] (shown in Table 6.1 as *% Neglect*) is higher for manual-control teleoperation than shared-control teleoperation by an average of 72% to 64%. Workload, as indicated by entropy, is also shown to by higher in manual-control teleoperation than shared-control teleoperation by an average of 0.68 to 0.41. These trends are similar to those found in the real world (see Table 3.1), where time-based workload and entropy were both higher in manual-control teleoperation (57% to 38% and 0.70 to 0.46). Thus, trends in workload are similar in the simulator to those found in the real world.

Robot Performance. In the simulator, the average performance (as a percentage of capacity) was higher for shared-control teleoperation than manual-control teleoperation (93% to 84%). This is consistent with the performance trend in the real world in which shared-control teleoperation outperformed manual-control teleoperation by 85% to 63%. In the real world, all participants did better performing shared-control teleoperation than manual-control teleoperation. In the simulator, 5 of the 7 participants did better with shared-control teleoperation. Thus, while the simulated world results are not as strong

[2]Calculated, again, by the percentage of time the operator spends doing arithmetic problems.

Shared-Control Results

Participant	A	B	C	D	E	F	G	Ave.
% Neglect	74%	72%	77%	61%	73%	72%	74%	72%
% Performance	97%	88%	94%	98%	85%	92%	97%	93%
# per min.	12.0	12.4	10.3	12.1	13.8	16.3	15.8	13.2
% Correct	71%	63%	39%	94%	85%	88%	78%	74%
Entropy	0.37	0.49	0.45	0.32	0.39	0.55	0.29	0.41

Direct-Control Results

Participant	A	B	C	D	E	F	G	Ave.
% Neglect	65%	70%	70%	34%	70%	68%	73%	64%
% Performance	83%	74%	96%	96%	88%	75%	81%	84%
# per min.	10.2	12.5	9.8	6.4	11.5	12.7	13.4	10.9
% Correct	57%	63%	38%	79%	71%	88%	77%	67%
Entropy	0.68	0.77	0.69	0.57	0.66	0.72	0.67	0.68

Table 6.1: *Results for seven subjects (A, B, C, D, E, F, and G) in the experiment.* % Neglect *is how much the robot was neglected (the percent of time the operator spent doing arithmetic problems),* Performance *is how efficently the primary task was completed (as a percentage of maximum possible performance),* # per min *is how many arithmetic problems were attempted per minute,* % Correct *is the percentage of attempted arithmetic problems the subject answered correctly, and* Entropy *is the joystick steering entropy calculated; lower values indicate smoother driving. Again, the shared-control system dominates the manual-control system in every category for most participants. Trends are similar to those observed in the real world.*

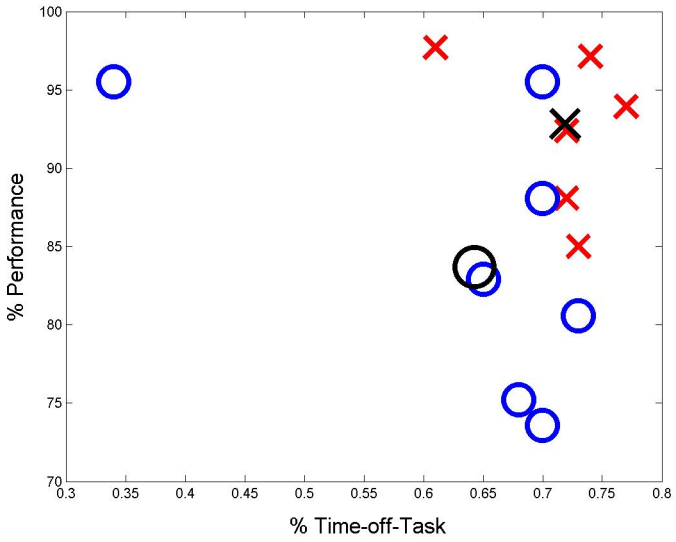

Figure 6.1: *A plot of robot performance verses operator workload. The vertical axis represents robot performance (as a percentage to maximum effectiveness) and the horizontal axis represents operator workload (in terms of percentage of time-off-task). The x's represent results from shared-control, and the o's from manual-control. The big x and big o represent the mean values. Note that only six x's appear in the plot because the results from two of the test subjects overlapped on shared-control teleoperation.*

in showing the dominance of shared-control, they still support the same trends in performance.

Figure 6.1 shows performances verses time-off-task (time-based workload) for the experiments performed in the simulator. This figure should be compared to Figure 3.2. Notice that there appears to be slightly better separation between the two classes (manual-control vs. shared-control) in the real world. However, similar trends exist in workload and robot performance in both worlds. Thus, we conclude that the simulator is sufficient for validating the trends in neglect tolerance and interface efficiency.

6.3 Chapter Summary

In this chapter, we described the simulated world that the experiments described in this thesis were performed. Results obtained in the simulator were shown to have similar trends in workload and robot performance to results obtained in the real world. This was shown by comparing the results presented in chapter 3 with results of the same experiment performed in the simulated world.

In the next chapter, we report the user study designed and performed according the to the specifications of the measurement technology described in chapter 4 to obtain measures of the neglect tolerance and interface efficiency of the three interaction schemes described in chapter 5.

Chapter 7

Validation of Usefulness

In chapter 4, we defined metrics for neglect tolerance and interface efficiency in human-robot systems. These metrics are random processes that can be estimated via the measurment technology, also discussed in chapter 4. In this chapter, we validate the usefulness of these metrics by comparing and contrasting the three interaction schemes described in chapter 5 in a navigation task in the environment described in chapter 6.

In this chapter, we first define the instantaneous performance and world complexity metrics used by the neglect tolerance and interface efficiency metrics. Next, we describe the user study performed to estimate the neglect tolerance and interface efficiency random processes for each interaction scheme. Finally, we report the results and compare and contrast the three interaction schemes, as well as analyze the world complexity metric that we used.

7.1 Instantaneous Performance and World Complexity for Navigation

In this section, we describe the instantaneous performance and world complexity metrics used to derive $V(Teleop)$, $V(P2P)$ and $V(Scripted)$ for a navigation task.

7.1.1 Instantaneous Performance Metric

As described in chapter 4, instantaneous performance, ip, is defined as instantaneous work, iw, divided by instanteous capacity for work, ic. For the navigation of a robot through a maze world towards a goal position, the instantaneous capacity of the robot is the work that the robot would do if it moved optimally[1] towards its goal at top speed. Since the simulated robots can travel at the rate of 30 inches per second, we define instantaneous capacity for work as

$$ic = 30t_\epsilon \tag{7.1}$$

where t_ϵ is the time elapsed, usually a small amount of time. The instantaneous work done by a robot in this task is how much closer it is to its goal after time t_ϵ. Let d_i be the distance the robot is from its goal at time i. Then, the instantaneous work iw performed at time i is

$$iw_i = \frac{d_i - d_{i-t_\epsilon}}{t_\epsilon}. \tag{7.2}$$

Using Dijkstra's Algorithm, we can obtain d_i by calculating the shortest path from the robot to its goal at time i using the topographical map of the world, which contains information about the distance between nodes (i.e., intersections) in the world. Thus, by combining equations 7.1 and 7.2 the instantaneous performance of a robot at time i is

$$ip_i = \frac{iw_i}{ic} = \frac{d_i - d_{i-t_\epsilon}}{30t_\epsilon^2}, \tag{7.3}$$

where the variables are defined as before.

We note that since "cutting corners" can perhaps decrease the shortest path to the goal and we don't want to worry about calculating the optimal way to cut corners for our distance measure, it is possible for ip_i to be greater than 1 (or less than -1). In such a case, ip_i is truncated to 1 (or -1). Thus, the sum of all ip over a period of time is not necessarily equal to the performance of the robot over that same period of time. It is, however, close to the overall performance, which makes it a good performance metric.

[1]Optimality, in this context, ignores clutter that might exist in the path from the robot to its goal.

Thus, we have a performance metric that returns the value of the robot's instantaneous performance, which is a value between -1 and 1. This performance metric meets the requirements for performance metrics we discussed in chapter 4.

7.1.2 World Complexity Metric

Loosely speaking, world complexity for navigation in a maze consists of those things that make navigation through the maze difficult. The two dominant factors that make navigation through a maze difficult are the branching factor of the environment and the amount of clutter in the environment. Branching factor indicates the number of decision points (e.g., intersections) per area, and clutter refers to the amount of obstacles in the environment per area through which a robot must travel.

If we take branching factor B and clutter L as the two elements that affect the complexity of an environment for navigation through a maze, then we can define world complexity $\mathcal{C}$ at any given state S, where S can be a sequence of states[2], as some function of B and L,

$$\mathcal{C}(S) = f(B(S), L(S)).$$

We must find B, L and the function f that maps B and L to C.

Estimating Branching Complexity

$B(S)$ is the average number of directions of travel afforded to a robot over a defined area. Moving down a straight hall at any given moment there are two afforded directions of travel: forward and backward. Likewise, there are three afforded directions of travel at a "T" intersection, four for a four-way intersection, etc. We define a straight hallway (or two afforded directions or less) to have zero branching complexity. Let A_p be the number of afforded directions to travel at position p, which is a coordinate in the robot's world. Then, branching complexity at that moment, or instantaneous branching complexity, is

[2]A *state* in this context is a snapshot of the conditions of the robot (sensor information, etc.) and the conditions of its world.

equal to $A_p - 2$. By assuming that there are no more than four afforded directions of travel at a time, this value is scaled to be between 0 and 1 by dividing it by two, except in the case in which A_p is equal to one. Thus, the instantaneous branching complexity at position p, ib_p, is given by

$$ib_p = \frac{A_p - 2}{2}. \tag{7.4}$$

The average branching complexity $B(S)$ at a given time is the weighted average of the instantaneous branching complexities over a certain distance of travel. More formally, let S be the set of states, or robot positions, that the robot has visited in its last D units of travel. Then,

$$B(S) = \sum_{p \epsilon S} \frac{(D - ||P_c - p||) ib_p}{\frac{D|S|}{2}}, \tag{7.5}$$

where P_c is the robot's current position, $||P_c - p||$ represents the distance the robot traveled between positions P_c and p and $|S|$ is the number of states in S. In our implementation, we set D equal to 800 inches[3]. It is possible that $B(S)$ could be less than zero since A_p can be less than zero. If it was, then $B(S)$ was set to zero since branching factor can never be negative.

The number of afforded directions of travel can be determined using the sonar signatures of the robot. A cluster of adjacent sonars that each exceed a given threshold indicates an afforded direction of travel. For example, the sonar signature shown in Figure 7.1(a) has three different clusters, each cluster consisting of three adjacent sonars that have values that exceed the threshold. Figure 7.1(b), however, shows a sonar signature that has just two clusters, one of which includes three sonars and the other includes six sonars. The cluster with six sonars probably represents two afforded directions of travel. Therefore, if the number of sonars in a cluster is more than three, then that cluster is considered to be two clusters. Likewise, a cluster of sonars greater than six is considered to be three

[3]800 inches represents about 25-30 seconds of robot travel (for the robots used in the experiments). This is enough to time to gather sufficient information to estimate the number of afforded directions of travel per area in the robot's current environment.

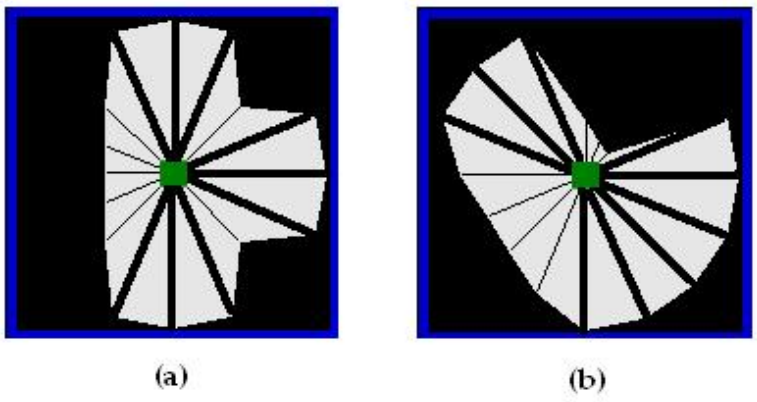

Figure 7.1: *Two different sonar signatures, each consisting of three (a) and two (b) clusters of sonars that exceed the threshold value. At right, the larger cluster is divided into two clusters since its size indicates that it probably represents two afforded directions of travel. Thus, both sonar signatures represent three afforded directions of travel.*

clusters, greater than nine is four clusters, etc. Thus, the number of afforded directions of travel at any given time is determined to be the number of clusters of sonars that exceed a given threshold.

Estimating Clutter Complexity

To measure the clutter of an environment, L(S), we use (a) directional entropy, $E_\theta(S)$, (b) change in velocity over time, $E_V(S)$, and (c) change in sonar values over time, $E_S(S)$. Directional entropy, $E_\theta(S)$, is found by using the method described in [35] to compute the entropy in the direction the robot faced. We modify the method slightly so that more recent robot behavior has more impact on the directional entropy calculation. This is done by assigning weights to each robot movement, where the most recent movement receives the highest weight. Weights assigned to past movements are given in a linearly decreasing fashion according to the time since each movement was made. Change in velocity over time, $E_V(S)$, can be calculated by averaging the robot's change in velocity over its last 30 movements. This value is then scaled to be between 0 and 0.5. We obtain change in sonar values over time, $E_S(S)$ by calculating the sum of the sonar values for the current sonar

readings, S_c, and the previous sonar readings, S_p. From these values, we get

$$E_S(S) = \begin{cases} \frac{3|S_c - S_p|}{S_p} & \text{if } |S_c - S_p| <= S_p \\ 0.25 & \text{otherwise} \end{cases} .$$

From these estimates, we can obtain an estimate of clutter complexity via $L(S) = E_D + E_V + E_\theta$.

Combining Branching and Clutter Complexities

There are many ways that $B(S)$ and $L(S)$ can be combined, but for this thesis we made $\mathcal{C}(S)$ a weighted sum of $B(S)$ and $L(S)$. Formally,

$$C(S) = \begin{cases} 1.0 & \text{if } w_B B(S) + w_L L(S) > 1 \\ w_B B(S) + w_L L(S) & \text{otherwise} \end{cases} , \tag{7.6}$$

where w_B and w_L are positive weights assigned to B and L respectively. Notice that the S used in $B(S)$ may be different than the S used in $L(S)$. However, the S in $B(S)$ is a superset of the S used in $L(S)$ and, therefore, we use this S in equation 7.6, and L simply ignores the states of S that it does not need. We use $w_B = 1.0$ and $w_L = 0.71$ in our metric since these weights give branching and clutter estimates, on average, equal weight. This returns a value between 0 and 1.0.

Notice that because a robot tends to move differently for each interaction scheme, world complexity estimates may be slightly different for each interaction scheme because of differing autonomy modes and control elements. This is true of both the branching and clutter estimates. In practice, however, world complexity estimates throughout the worlds were similar for all the interaction schemes used.

Figure 4.4 shows two worlds used in the experiments described in this chapter. Using this world complexity metric with a teleoperation interaction scheme, the world shown in Figure 4.4(a) has an average world complexity of 0.373 and the world shown in Figure 4.4(b) has an average world complexity of 0.216. These numbers indicate that, indeed, the world complexity metric returns a notably higher estimate for a world that would be described as more complex than for a more simple world.

Figure 7.2: *Secondary two-digit addition and subtraction problems were used as one of the secondary task used in the user study. The above display replaced the sensor information in the GUI shown in Figure 5.1 when the operator was to perform this task.*

7.2 The User Study

In this section, we describe the user study performed to estimate the random processes $V(Teleop)$, $V(P2P)$, and $V(Scripted)$ for the navigation task in a maze environment. The user study was performed according to the measurement technology described in chapter 4.

7.2.1 User Study Design and Protocol

The measurement technology in chapter 4 uses secondary tasks to sample random processes. In this user study, we used two secondary tasks. The first secondary task that the human operator was asked to perform was to control a second robot. This made it possible to gather twice as much data during each test session, so less test subjects were needed. The other secondary task was to perform two-digit addition and subtraction problems. The part of the GUI that displays robot sensory information (see Figure 5.1) was replaced by the math display shown in Figure 7.2 when this secondary task was to be performed. Thus, the current task to be performed by the operator was displayed in the bottom portion of the GUI.

When one of the robots was being serviced, the operator was allowed to service that robot as long as he/she desired. When he/she was done, he/she clicked a button and was given a different task. A random neglect time was then assigned to this robot and the

operator was not allowed to service this robot again until the neglect time had elapsed. After the neglect time had elapsed, the task of servicing this robot was reassigned to the operator provided that the operator was not currently servicing the other robot. When both of the robots were being neglected, the operator was assigned the task of solving arithmetic problems until it was time to service one of the robots again.

We created 21 different worlds, each of different make-up and complexity. The first world, the *training world*, was used to train test subjects on the interaction schemes they were to use. This world included a wide variety of world complexities. The other 20 worlds were selected for use randomly during test sessions, but restrictions were made on how many times a world could be used.

Instructions on how each test subject was to proceed with the experiment was read from a prepared script (included in appendix A). The protocol for the experiment was as follows:

1. The interaction scheme to be used in each of the three sessions was selected randomly. Although selections were made randomly, certain stipulations were made. First, exactly two interaction schemes were used. This was done to try to keep the test subject from getting too tired of doing the same task over and over, and also limited the amount of training that was necessary. Second, we needed to be certain that, after the 120 sessions have been completed, the correct number of sessions were used for each interaction scheme. Scheduling of the interactions schemes to be performed was facilitated by creating a schedule (randomly) of interaction schemes to be used by each subject.

2. The interaction scheme corresponding to the current session number was selected (chosen in step 1).

3. The subject (i.e., the operator) was trained on the current interaction scheme if he/she had not been trained on it previously. He/she was trained by navigating a robot through the *training world* towards its goal position using the current control

scheme. Additionally, the operator was allowed to practice neglecting the robot by clicking on a button, at which time the operator was asked to perform math problems for a short time. When the robot reached its goal position, another goal was selected at random in the same world and the process was repeated. The operator was given as much time as he/she felt was necessary to learn the interaction scheme well. Tips on how to better control the robot using the current interaction scheme were read from the prepared script.

4. A world was selected at random from the set of worlds. Again, random selection was constrained to use all the worlds equally often, and a test subject was not allowed to see the same world twice. We again facilitated this by creating a schedule of the order that worlds were to appear. The test subject was asked to perform the same task described in the training step for two robots simultaneously according to the specifications described previously. This process continued for 10 minutes, during which time the following information was recorded: time, robot sensory information, environmental complexity estimates, robot performance, and operator movements (e.g., mouse clicks, joystick movements, etc.). In order to encourage high operator effort, test subjects were told that they were to receive a score based how well they did[4].

   Since a robot's performance immediately drops to zero when it is neglected using *Teleop*, this step of the experiment was changed slightly for this interaction scheme. With *Teleop*, one of the robots was serviced for about ten seconds, after which the operator performed one arithmetic problem after which the test subject again serviced one of the two robots, selected at random, for ten seconds, etc. This allowed the random process of this interaction scheme to be sampled thoroughly.

[4]This score was calculated based on average robot performance, operator math proficiency, and world difficulty.

5. The subject was given a short break.

6. Steps 2 through 5 were repeated until all three sessions were completed.

Each test subject took part in three ten-minute test sessions, using a total of two different interaction schemes. A total of forty test subjects were used in all, so 120 test sessions were performed. Of these sessions, 15 were dedicated to the *Teleop* interaction scheme, 48 to the *P2P* interaction schemes, and 57 to the *Scripted* interaction scheme.

As mentioned in chapter 4, the domain space of the random processes, consisting of the variables t_N, t, and c, must be properly discretized[5]. In order for t_N to be sampled sufficiently for each interaction scheme, some neglect times must be extended until the expected performance of the robot approaches zero. This is a different length of time for each interaction scheme so t_N must be discretized differently for each interaction scheme. For *Teleop*, t_N took on only one value since robot performance immediately dropped to zero upon being neglected. For *P2P*, t_N was divided into bins of 5, 10, 15, 20, 25, and 30 seconds. For *Scripted*, t_N was divided into bins of 10, 20, 30, 40, 50, and 60 seconds. The time dimension of the domain space was discretized into half second increments and the complexity dimension of the domain space was discretized into chunks of 0.05 units.

7.3 Results

In this section, we analyze the three interaction schemes according to the measures of neglect tolerance and interface efficiency estimated by the user studies. First, we describe the number of samples obtained for each domain space for the three interaction schemes. Second, we discuss $V(Teleop)$, $V(P2P)$, and $V(Scripted)$ individually. Third, we compare the three interaction schemes. Last, we analyze the world complexity metric that we used

[5]Recall that t_N is the time the robot was neglected prior to its last servicing, t is the current time (t_{on} or t_{off}), and c is world complexity.

Teleop

t_N	0	8	26	57	61	82	64	50	50	44	39	22	26	26	18	17	5	11	5	4
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**

P2P

30	0	7	11	21	26	22	27	24	22	23	14	11	13	12	12	4	1	3	3	2
25	1	0	31	0	33	0	44	2	33	1	27	0	23	0	11	1	4	0	2	0
20	0	6	0	33	0	31	0	22	0	18	0	14	0	10	0	4	0	2	0	0
15	1	0	26	4	31	0	39	2	33	0	26	1	20	1	11	1	5	0	2	0
10	0	6	1	17	0	19	1	11	1	8	0	7	1	8	0	5	0	1	0	0
5	0	0	17	2	40	0	19	0	20	0	11	0	10	0	7	0	3	0	0	0
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**

Scripted

60	0	8	5	45	10	42	5	36	6	41	4	28	8	24	2	15	3	4	0	4
50	0	0	29	0	51	1	44	0	37	0	26	0	23	0	22	0	4	0	3	0
40	0	4	0	36	0	30	0	37	1	30	0	21	1	19	0	11	0	6	0	1
30	0	0	22	0	49	1	39	0	31	1	26	0	25	0	20	1	7	0	8	0
20	0	7	0	25	0	33	1	36	0	19	2	17	3	19	1	9	0	3	0	4
10	0	0	34	1	47	0	46	1	30	0	35	0	31	0	18	1	12	0	3	0
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**100**

Table 7.1: *Number of samples for each tuple (c, t_N), where t_N is along the vertical axis and c is along the horizontal axis. For simplicity, c is represented by the values 0 through 100. We note that for Teleop only one neglect time t_N is listed (t_N), which indicates that all of the values are lumped together.*

to estimate world complexity.

Following the discretization method described previously, $V(Teleop)$, $V(P2P)$, and $V(Scripted)$ were estimated nonparametrically from the data obtained from the user study. Table 7.1 shows the number of samples for each tuple (c, t_N) for the three interaction schemes. We do not display the t variable for visability's sake, but the number of samples along t is fairly uniform.

Table 7.1 shows that at the extremes of complexity, very little sampling occured. The reason for this is that world complexity rarely reaches these levels. Additionally, the table shows that for $P2P$ and $Scripted$, only every other bin was sampled. This is not a problem since interpolation and filtering can fill in the gaps. In general, the tables show that the domain space of the random processes were, for the most part, sampled sufficiently.

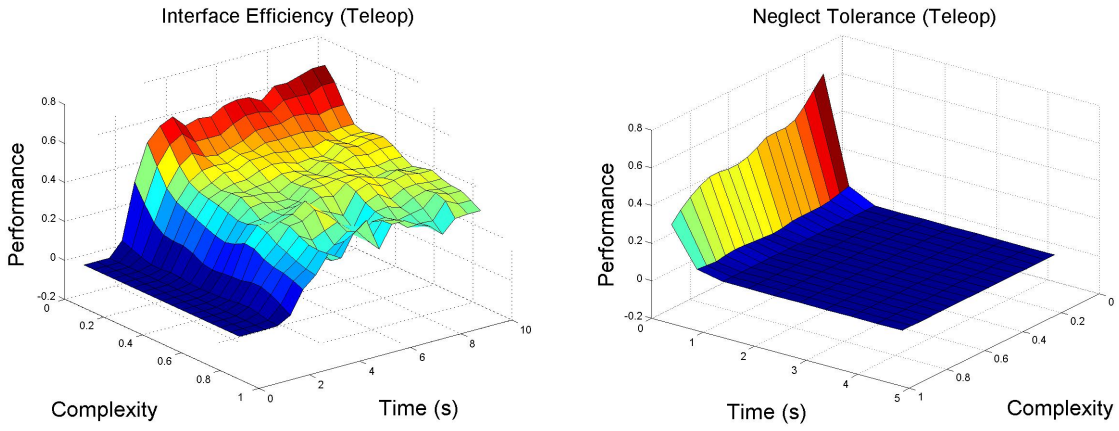

Figure 7.3: *A plot of the mean of the estimated random process $V(Teleop)$ consisting of $E[V_S(Teleop)]$ (left) and $E[V_N(Teleop)]$ (right).*

7.3.1 *Teleop* Results

As mentioned previously, 15 ten-minute sessions were dedicated to estimate the $Teleop$ interaction scheme for the navigation task. The mean of the estimated random process $V(Teleop)$ is shown in Figure 7.3. On the left-hand side of the figure is $E[V_S(Teleop)]$ and on the right-hand side of the figure is $E[V_N(Teleop)]$.

Interface Efficiency

Figure 7.3(left) shows the expected value of interface efficiency for the $Teleop$ interaction scheme. Notice that for all levels of complexity, it takes a few seconds before the operator is able to make the context-switch from a secondary task to driving the robot with the joytick. After the first few seconds, the expected performance of the robot rises very quickly for all levels of complexity until it reaches peak levels and then levels out.

Figure 7.4 gives a clearer presentation of interface efficiency for a few different complexity levels. Although the results are noisy, the general trends are obvious. The trends are as expected. Indeed, increased complexity does tend to lower performance. However, even at high levels of complexity, expected performance has decayed very little, from a peak performance of between 0.5 and 0.6 for a world complexity of 0.20 to about 0.4 for a world complexity of 0.70. This decrease in expected performance is expected since increased

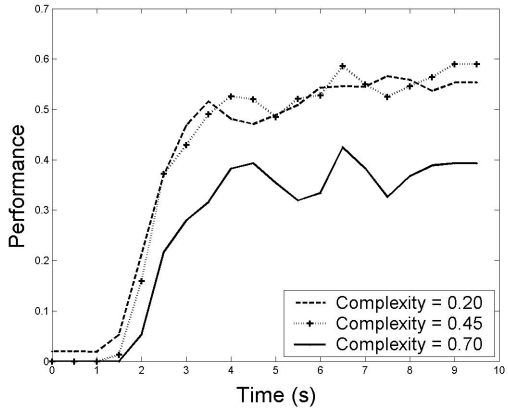

Figure 7.4: *Comparison of expected performance of $V_S(Teleop)$ for various levels of complexity.*

branching and clutter cause the operator to need to turn and move around obstacles, which slows the robot's progress toward its goal.

Neglect Tolerance

The effect that neglect has on the *Teleop* interaction scheme is not very interesting because the performance of the robot quickly goes to zero when the human completely neglects it. Figure 7.3(right) shows this expected result.

Combining Interface Efficiency and Neglect Tolerance

We learned from chapter 3 that moving from manual-control teleoperation to shared-control teleoperation can reduce the amount of attention that the operator gives to the human; however, the human must still continue to give continous input to the robot. This means that the combination of neglect tolerance and interface efficiency requires continous human-robot interactions, which is consistent with what a combination of neglect tolerance and interface efficiency specifies unless the minimum acceptible performance level (MAPL) is defined to be below zero, which would be impractical. Thus, the expected performance of a robot employing *Teleop*, given that these interactions occur, is defined, at each level of world complexity as the peak expected performance level. This peak expected performance

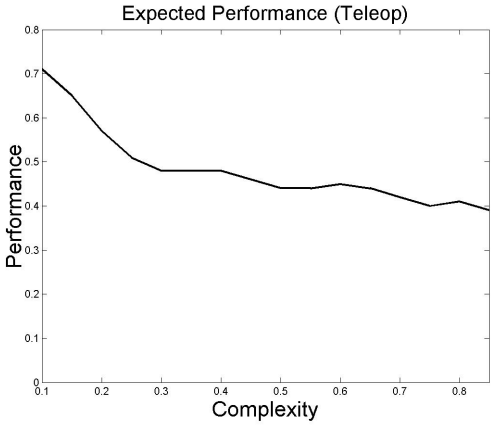

Figure 7.5: *Shows the peak expected performance for Teleop for most complexities. These peak performance levels correspond to the robot's expected performance at any given time (given that interactions are constant, which is what the combination of the measures neglect tolerance and interface efficiency recommends for this interaction scheme).*

level for most complexities is shown in Figure 7.5.

Confidence Measurements

Up to this point we have looked only at the expected value of $V(Teleop)$. However, in some circumstances there is more reason to be concerned about whether a robot's performance is above a certain level rather than what its expected performance is. $V(Teleop)$, being a random process, also provides this information.

Figure 7.6 shows the probability that performance will be greater than or equal to 0.2, 0.4, 0.6, and 0.8 respectively for $V_S(Teleop; t_{\text{on}}, c = 0.25, t_N)$ and $V_N(Teleop; t_{\text{on}}, c = 0.45)$. The same information is available through $V(Teleop)$ for all levels of defined world complexity. The figure shows that, in an environment with estimated complexity of 0.25, after a robot has been serviced for about ten seconds, a robot's performance level will be greater than or equal to 0.2 nearly 70% of the time. The majority of the cases in which the performance is below 0.2 for teleoperation is when the robot's performance is zero, or when the robot is turning in place or moving around obstacles. On the other extreme, a robot performs at top speed (i.e., above 0.8) only about 30% of the time after ten seconds

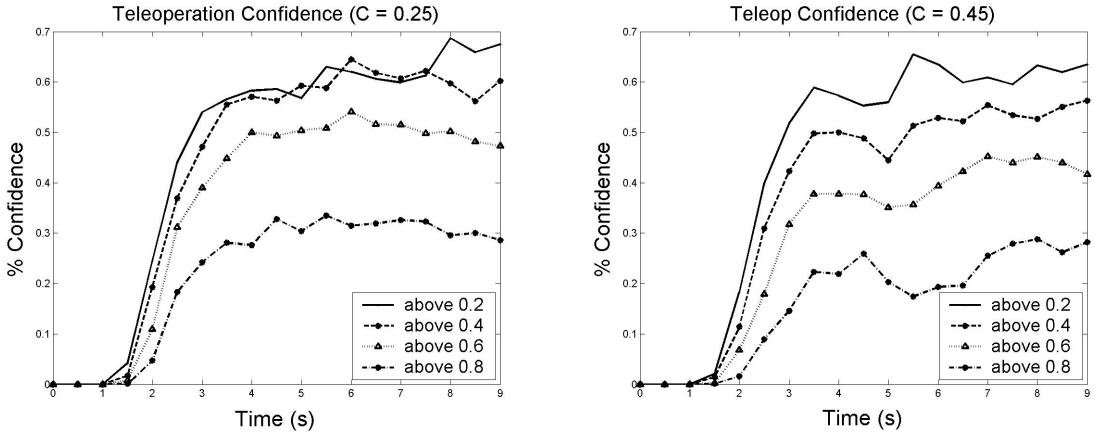

Figure 7.6: *Shows the probability that performance will be greater than or equal to 0.2, 0.4, 0.6, and 0.8 of* $V_S(Teleop; t_{\text{on}}, c = 0.25, t_N)$ *(left) and* $V_N(Teleop; t_{\text{on}}, c = 0.45)$ *(right).*

of interactions with a human using *Teleop*. Moving to higher complexity levels lowers these confidence levels slightly (see Figure 7.6(right)).

Strengths and Weaknesses

Teleop has obvious strengths and weaknesses. Its two greatest weaknesses are, first, that it requires continuous input from a human operator and, second, it requires that the human (during interactions) must provide fairly specific details. The latter of these weaknesses is also a strength since because a human can give specific details to the robot, he/she is better able to guide the robot in cases that other automation fails. Additionally, it appears that the price of context switching is relatively low, as a robot is brought from a standstill to fairly high performance levels in only a few seconds.

7.3.2 *P2P* Results

There were 48 ten-minute sessions dedicated to $P2P$. Figure 7.7 shows $E[V(P2P)]$, consisting of $E[V_S(P2P)]$(left) and $E[V_N(P2P)]$(right), for the navigation task. The function is shown for neglect time $t_N = 30$ seconds.

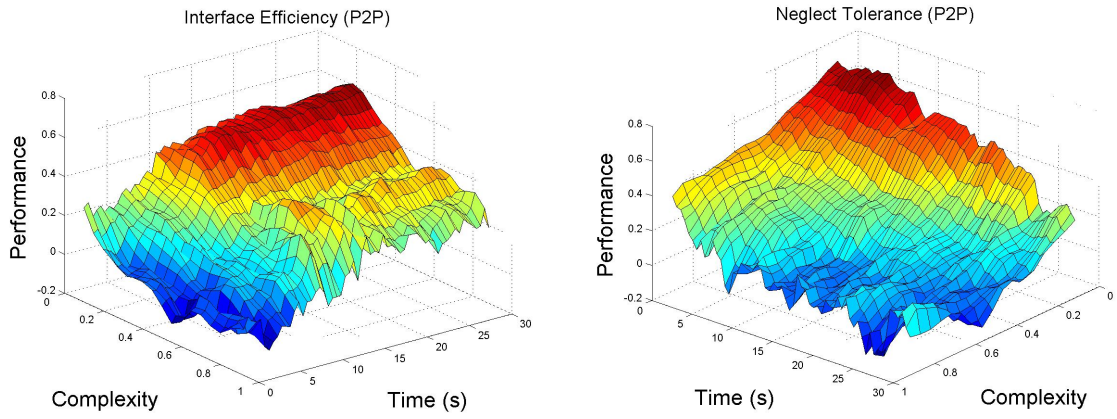

Figure 7.7: *A plot of the mean of the estimated random process* $V(P2P; c, t, t_N = 30)$ *for the navigation task, consisting of* $E[V_S(P2P); c, t_{\text{on}}, t_N = 30]$ *(left) and* $E[V_N(P2P; c, t_{\text{off}})]$ *(right).*

Interface Efficiency

Figure 7.7(left) shows the expected performance level of a robot using $P2P$ during interactions with a human after having been neglected for 30 seconds. Notice that the general trends are as we hypothesized. That is, as world complexity increases, expected robot performance decreases. Also, the function shows the property that robot performance increases as it interacts with a human. It takes the human a couple of seconds to make the context switch to controlling the robot, after which expected robot performance gradually begins to increases.

There is a slight trend shown in the figure that we did not predict. The graph shows a slight valley, or decreased expected performance, for medium range world complexities. This appears to be caused by the ineffectiveness of our world complexity measure. We discuss this in more detail later.

Figure 7.8 shows the expected performance of a robot being serviced in a world with complexity of 0.35 after various neglect times ($t_N = 10$, $t_N = 20$, and $t_N = 30$ seconds). Interestingly, after only about 4 or 5 seconds, the three plots show little difference. Thus, at this level of complexity, longer neglect times seem to have little impact on the efficiency of human-robot interactions with this interaction scheme. This indicates that the human

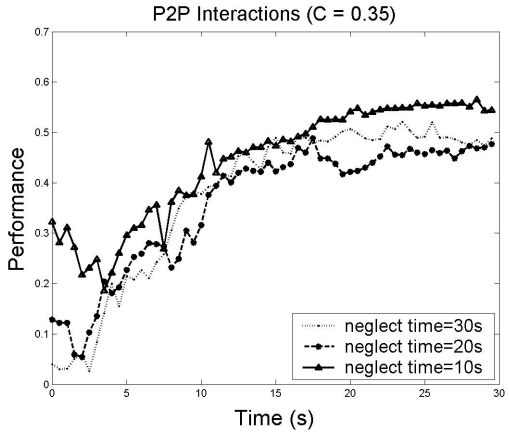

Figure 7.8: *Plots of expected performance during servicing after various neglect times ($t_N = 10$, $t_N = 20$, and $t_N = 30$) with a complexity of 0.35.*

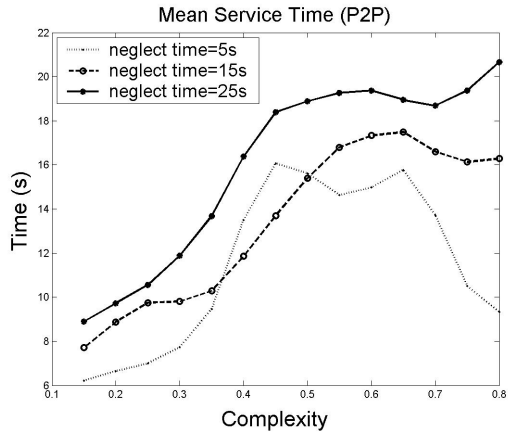

Figure 7.9: *Mean service time after the robot was neglected for various times t_N.*

does not seem to be relying on working memory directly after context switches. Rather, the human is relying on the interface to bring him/her up to date with what is going on with the robot.

To better understand the strengths of the interaction scheme, we also incorporate an analysis of the average interaction time of operators servicing robots employing $P2P$ for various world complexities. This is shown in Figure 7.9 after various neglect times t_N. Trends for each t_N are similar, with the exception of $t_N = 5s$ at high world complexities. Table 7.1 shows that few samples are available for $t_N = 5s$ at high world complexities, so this unexpected result can be attributed to noise. From the figure, as world complexity increases, service time increases significantly since the robot's were not proficient at carry-

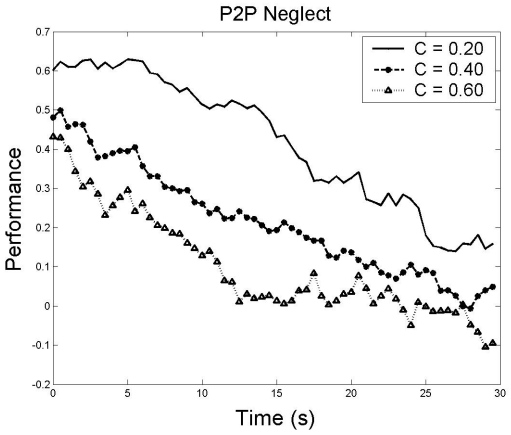

Figure 7.10: *Expected performance during time that a robot is neglect in environments with various world complexities for P2P.*

ing out the commands they were given. This caused operators to distrust the robots and monitor them more closely. Figure 7.9 also shows that interactions are not effective for high environmental complexities since the length of interaction time increases, yet performance still decreases. Thus, during interactions, the $P2P$ interaction scheme is not very complexity tolerant.

Neglect Tolerance

Figure 7.7(right) shows $E[V_N(P2P)]$ for the navigation task. As can be seen, the general trends we hypothesized for neglect tolerance hold true for this interaction scheme. When a robot is employing the $P2P$ interaction scheme, its expected performance decreases as it is neglected. Furthermore, its expected performance decreases faster at higher world complexities. Figure 7.10 further shows neglect tolerance for various levels of world complexity. In the figure, a robot's expected performance (using $P2P$) degrades in 30 seconds from 0.6 to 0.15 in a world with complexity 0.2. Meanwhile, a world with complexity estimate 0.6 causes a robot's expected performance, which began at about 0.42, to decay to nearly zero in about 12 seconds[6].

[6]The expected robot performance of 0.0 represents purely random behavior with $P2P$, since a robot acts randomly when the last command given to it has been carried out or is no longer applicable. For this reason, expected performance tends to level out at 0.0 when a robot currently has no operator input.

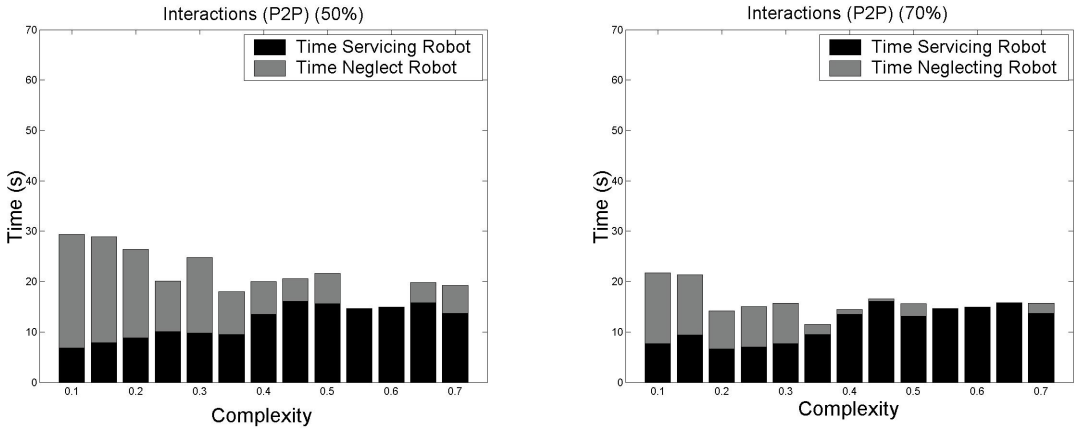

Figure 7.11: *Shows the average frequence and duration of interactions necessary to keep a robot above 50% (left) and 70% (right) of peak expected performance levels for different world complexities.*

Combining Interface Efficiency and Neglect Tolerance

Combining neglect tolerance and interface efficiency makes it possible to derive the average frequency and duration of interactions that must occur between a human and a robot to maintain high performance levels. The method for obtaining these interactions was discussed in chapter 4 and is depicted in Figure 4.5.

Figure 7.11 shows the average frequency and duration of interactions derived with minimum expected performance levels (MAPLs) of 50% and 70% of the peak expected performance levels obtained by the robot at each complexity level. The expected trend follows, that as world complexity increases, average interaction time, d_{on}, must increase and neglect time, d_{off}, must decrease to keep expected performance above a given threshold.

An important concept in generating the plots is the concept of context-switching. Since expected performance continues to drop after servicing has begun, concern must be taken that servicing begins some time before the expected performance level falls below the MAPL (see Figure 4.5). Because of this, any neglect of the robot is unacceptable for some of the higher world complexity levels.

Figure 7.11 illustrates the ineffectiveness of $P2P$, especially for intermediate world

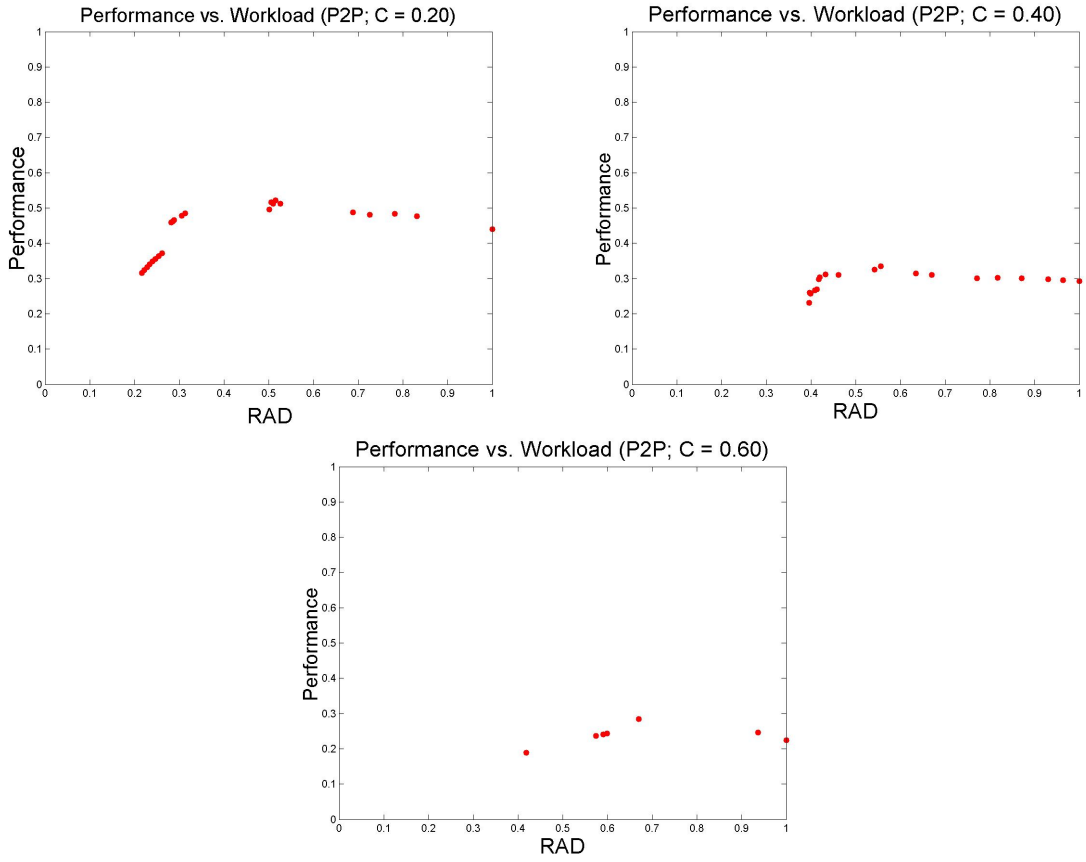

Figure 7.12: *Robot performance plotted against Robot Attention Demand for three levels of complexity. Each point represents the average robot performance and RAD corresponding to a single MAPL.*

complexity levels and higher. At these high world complexity levels, $P2P$ becomes merely an awkward teleoperation scheme, since neglecting the robot means that the robot will begin to perform poorly almost instantaneously. However, when branching factor and clutter are not extremely high, the $P2P$ interaction scheme can be quite useful. An example of such an environment is the hallways of a normal building, where there is little clutter and only occasional intersections. In such environments, an operator may easily give a robot a command and then neglect the robot, confident that it will carry out the command effectively.

Choosing the best MAPL is a key part of finding the best average frequency and duraction of human-robot interactions for a system. Figure 7.12[7] illustrates why this is

[7]These plots are closely related to the attention-operating-characteristic plots in [41].

the case. In the figure, robot performance is plotted against a time-based workload metric called *Robot Attention Demand* [38], or RAD, for three levels of complexity. The RAD is given by $\frac{d_{\text{on}}}{d_{\text{on}}+d_{\text{off}}}$. In the plots, each point represents the average robot performance and RAD obtained from a single MAPL. Interactions corresponding to high robot performance and low RAD are the most desireable. Observe from the plots at each world complexity that there exists a "sweet spot" at which point increasing the RAD yields very little increase in robot performance, and even sometimes decreases it. For example, when world complexity is 0.2, average robot performance increases until the RAD is about 0.5. When the RAD is increased above this point, robot performance actually decreases. Observe also that, although robot performance is the highest when the RAD is about 0.5, using a RAD of 0.3 yields only a small decrease in robot performance. Therefore, depending on system conditions it may be desireable to decrease operator workload at a small expense in robot performance.

While the percentage of workload is important, it is not everything. The length of the interaction cycle, or service time and neglect time added together, is also important. Whether a short interaction cycle or a long interaction cycle is more desireable depends, again, on the makeup of the system. Figure 7.11 also shows these cycle lengths for a couple of different MAPLs.

Confidence Measurements

We again emphasize that more information than just the expected performance of the robot is available from the interface efficiency and neglect tolerance metrics. Figure 7.13 show the confidence that the robot's performance level will be above certain levels for world complexities of 0.20 and 0.40. The graphs are relatively straight forward, and simply show information about the random processes $V_S(P2P; t_N = 30 seconds)$ and $V_N(P2P)$.

Strengths and Weaknesses

One of the biggest weaknesses of $P2P$ is that it is not very complexity tolerant. It seems that this is caused by two different reasons. First, only a single command could be given

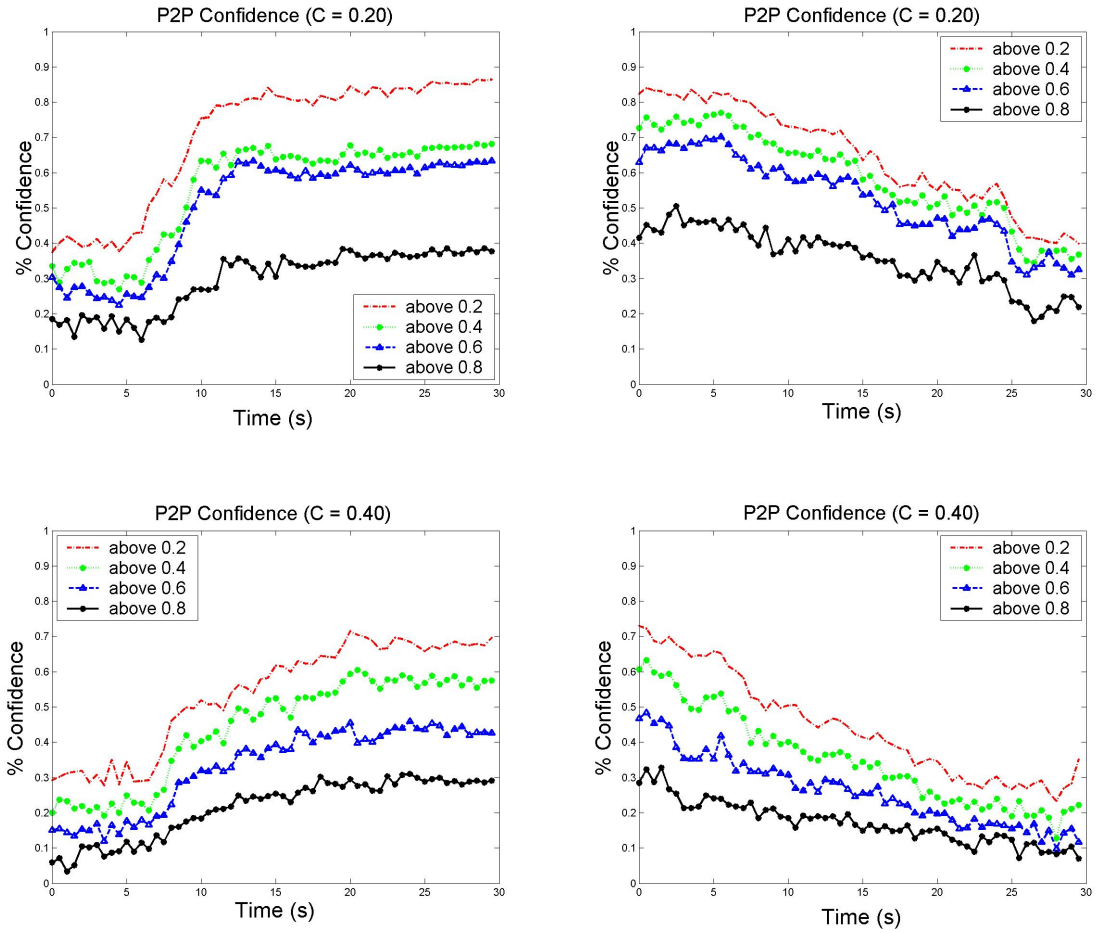

Figure 7.13: *Shows the probability that a robot (employing P2P) will perform above various levels at world complexities of 0.20 (above) and 0.40 (below). The graphs on the left are from interface effiency measures and the graphs on the right are from neglect tolerance measures.*

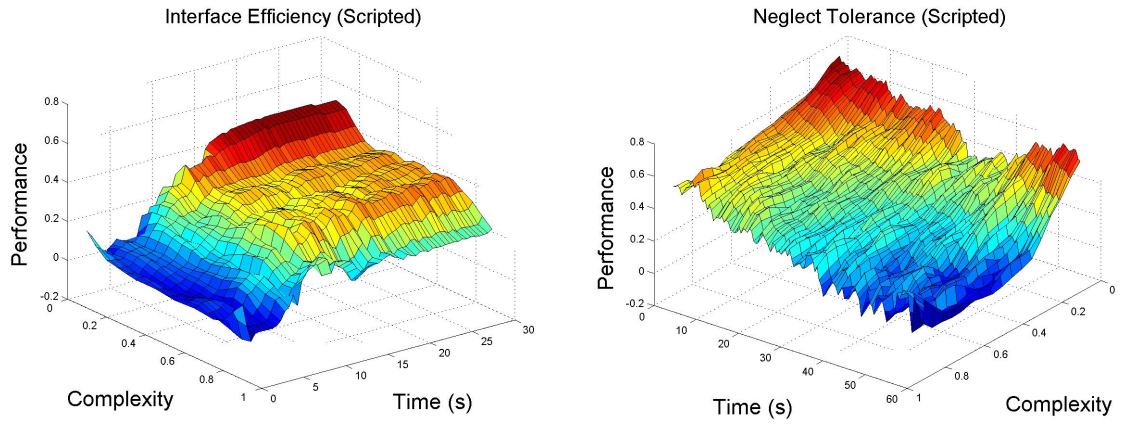

Figure 7.14: *A plot of the mean of the estimated random process* $V(Scripted; c, t, t_N = 60)$ *consisting of* $E[V_S(Scripted; c, t_{\text{on}}, t_N = 60)]$*(left) and* $E[V_N(Scripted; c, t_{\text{off}}, t_N = 60)]$*(right).*

to the robot at a time. By allowing the human to give multiple commands (e.g., turn right and then turn left), the interaction scheme could be improved. The second cause of $P2P$'s complexity intolerance can be linked to its inability to carry out commands in cluttered environments. The obstacles "confuse" the robot so that it frequently loses its way while trying to avoid the obstacles.

As mentioned previously, however, $P2P$ does indeed provide the ability to neglect the robot and still maintain high performance levels for some time in less complex environments. This gives it an advantage over *Teleop* in some circumstances.

7.3.3 *Scripted* Results

There were 57 ten-minute sessions dedicated to the *Scripted* interaction scheme. Figure 7.14 shows estimated $E[V(Scripted)]$, consisting of $E[V_S(Scripted)]$(left) and $E[V_N(Scripted)]$(righ for the navigation task. The figure is shown for neglect time $t_N = 60$ seconds.

Interface Efficiency

Figure 7.14(left) shows the mean of the random process $V_S(Scripted; c, t_{\text{on}}, t_N = 60sec.)$. Again, the expected trends occur. As a robot employing *Scripted* interacts with a human,

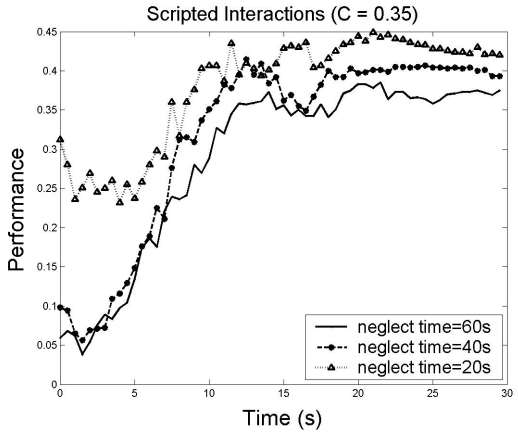

Figure 7.15: *Plots of expected performance during servicing after various neglect times (t_N = 20, 40, and 60 seconds respectively) with a complexity of 0.35.*

its expected performance increases. Additionally, expected performance peaks at lower levels for high world complexities than it does for low world complexities. Interestingly, like the *Teleop* interaction scheme, the peak occurs at about the same time for all levels of world complexity. Also, just like *P2P*, robot performance with *Scripted* seems to remain constant through the mid-range complexities. Only at very low and very high world complexities do changes in complexity seem to affect performance significantly.

Figure 7.15 gives an idea of what happens to the random process after smaller neglect times for a world complexity of 0.35. From the figure we see that when t_N is small (e.g., t_N = 20 seconds) performance has not dropped as far as for the other values of t_N. However, it takes nearly the same amount of time for the expected performance to peak as it does for the other values of t_N. In all cases, it takes more than ten seconds for the expected performance to peak. All this is similar to the interface efficiency of *P2P* (see Figure 7.8).

Like we did for the *P2P* interaction scheme, we would like to show the mean service time for various levels of complexity and neglect time. Figure 7.16 shows the mean servicing time. Interestingly, the mean servicing times for when t_N = 20 and 40 seconds are similar. However, mean interaction time for all middle and upper complexity levels increases significantly after a robot has been neglected for 60 seconds. We attribute these

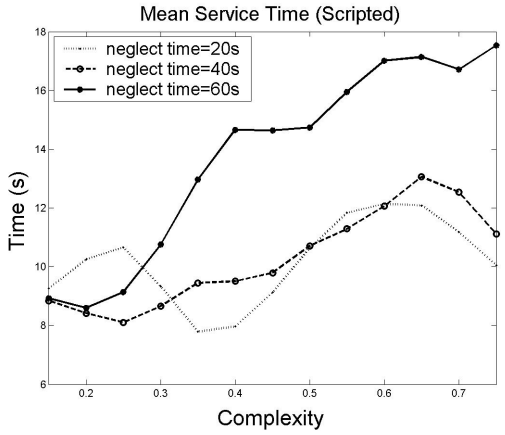

Figure 7.16: *Mean service time after the robot was neglected for various neglect times* t_N.

results to command queueing. Since the operator can send a large number of commands to the robot at a time, many interactions using *Scripted* consists of only a brief checkup by the operator on the robot. If the robot is proceeding as expected with the queued commands, the operator simply moves on to another task. However, after long neglect times, the robot is more likely to have reached its goal, so the next interaction requires the operator to issue a new sequence of commands to get the robot to its new goal.

The plots for $t_N = 20$ and 40 seconds have some unexpected behaviors. We attribute such behaviors to noise, since they generally occur at the extremes of world complexity were there is a deficiency in data samples.

Neglect Tolerance

Figure 7.14(right) shows the expected performance of a robot based on world complexity and neglect time. The trends we hypothesized are also born out in this figure. As neglect time increases, the expected performance of the robot decreases. Additionally, as complexity increases, the expected performance of the robot decreases at a faster rate when it is neglected.

Figure 7.17 shows the decay of performance over time for a few complexity levels in a clearer manner. These plots show a pecular trend. Just after interactions between the human and robot cease, expected performance continues to increase for a few seconds

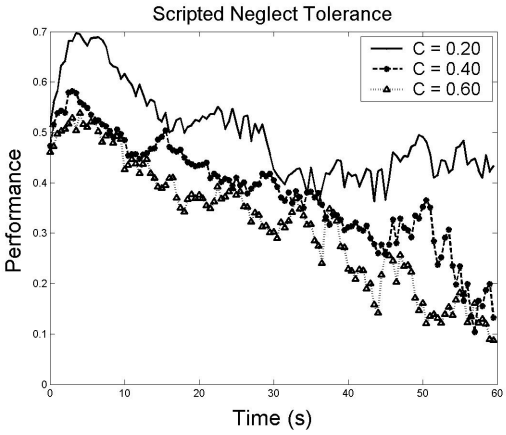

Figure 7.17: *Expected performance during time that a robot is neglect in environments with various complexities for Scripted.*

before beginning to decay. When a robot is given a command, it takes a couple of seconds for the robot to begin to move, since it must spin to face its next target. The sharp increase in performance can be attributed to this and indicates that the operators tended to trust that the robot would fulfill the command successfully. Thus they did not check to make sure the robot was performing the command properly before neglecting the robot.

The figure also indicates that performance degrades quite slowly over time. Even after 60 seconds, the graphs for all complexities show positive expected robot performance. Although world complexity does have an impact on performance, expected performance is relatively the same for world complexities of 0.4 and 0.6. However, when world complexity is very low (e.g., 0.2), performance degrades much less.

Combining Interface Efficiency and Neglect Tolerance

As we did with $P2P$, we can use interface efficiency and neglect tolerance to obtain the average frequency and duration of interactions needed to maintain high expected robot performance levels. These average interactions for MAPLs of 50% and 70% of peak performance levels are shown in Figure 7.18. The average interactions are significantly more desireable than those required by $P2P$ for all levels of complexity. Notice that interaction cycles are smaller when the MAPL is 70% rather than 50%.

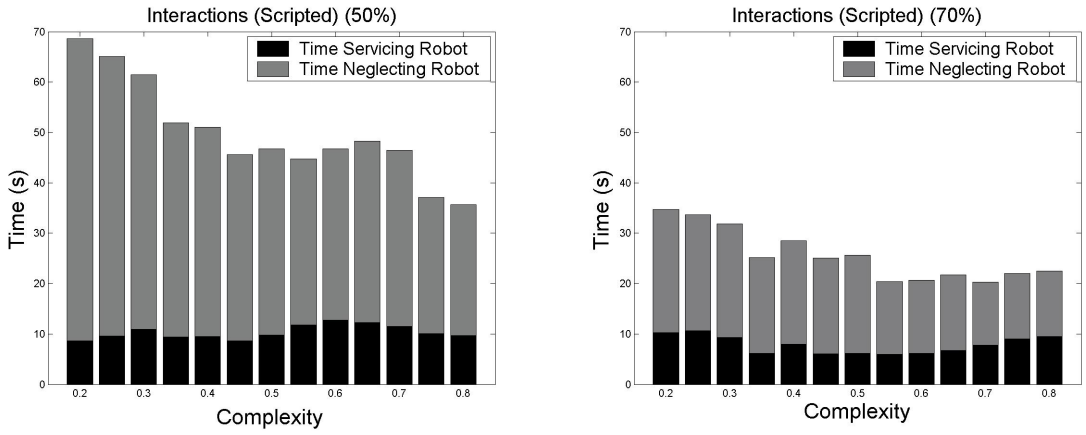

Figure 7.18: *Average interaction rates for Scripted with MAPLs of 50% (left) and 70% of peak expected performance. The black represents time spent on-task, and the gray represents time spent off-task.*

Figure 7.19 shows how changing the MAPL affects RAD and average robot performance for several world complexities. The graphs show that as world complexity increases, average robot performance decreases and RAD increases. However, such changes are not as drastic as those in $P2P$ (see Figure 7.12).

Confidence Measurements

Figure 7.20 shows the probabilities that a robot's performance level will be greater than or equal to certain levels for world complexities of 0.20 and 0.40 respectively. The general trends in the graphs are the same for those seen in the expected performance plots (see Figures 7.15 and 7.17).

Strengths and Weaknesses

The *Scripted* interaction scheme has many strengths. First, robot performance can remain high after the robot is neglected for some time due to the ability the operator had to queue commands. Second, low operator workload is required. Finally, *Scripted* performs fairly well when world complexity is high.

Scripted, however, is not without its weaknesses. Its main weakness is that it requires

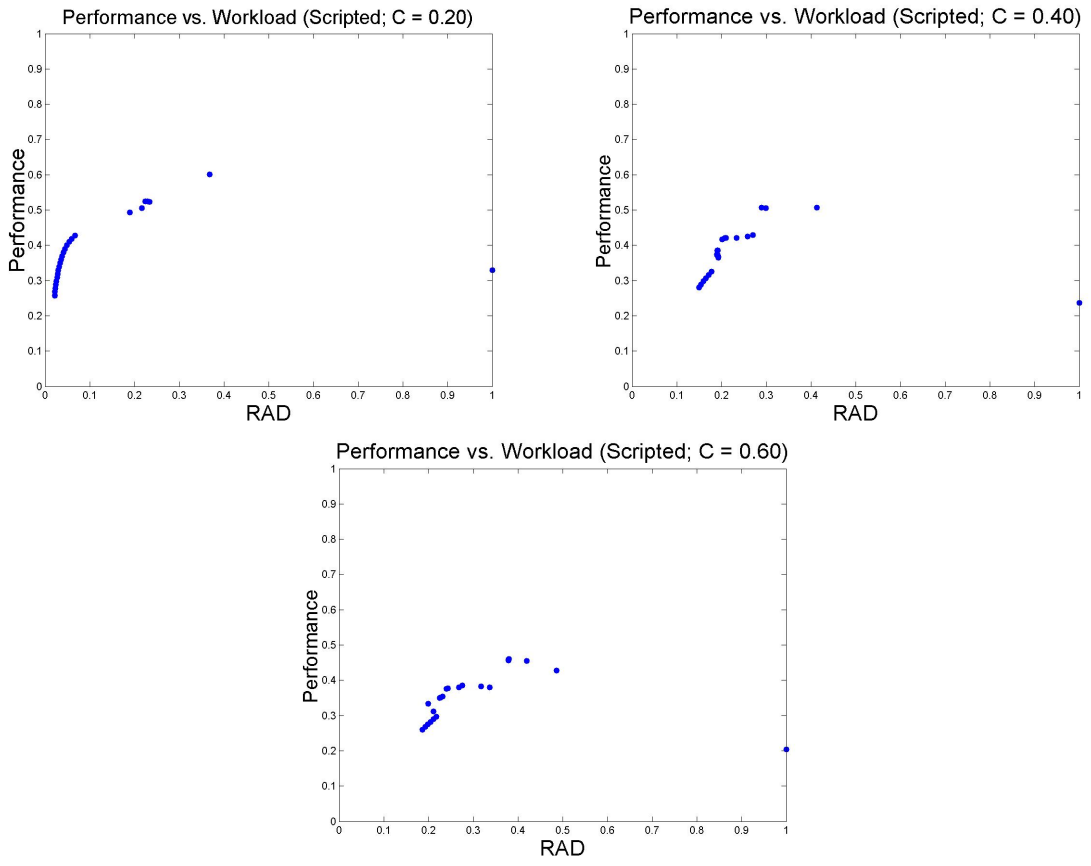

Figure 7.19: *Plots of percent operator workload verses performance for various MAPLs for world complexities of 0.20, 0.40, and 0.60.*

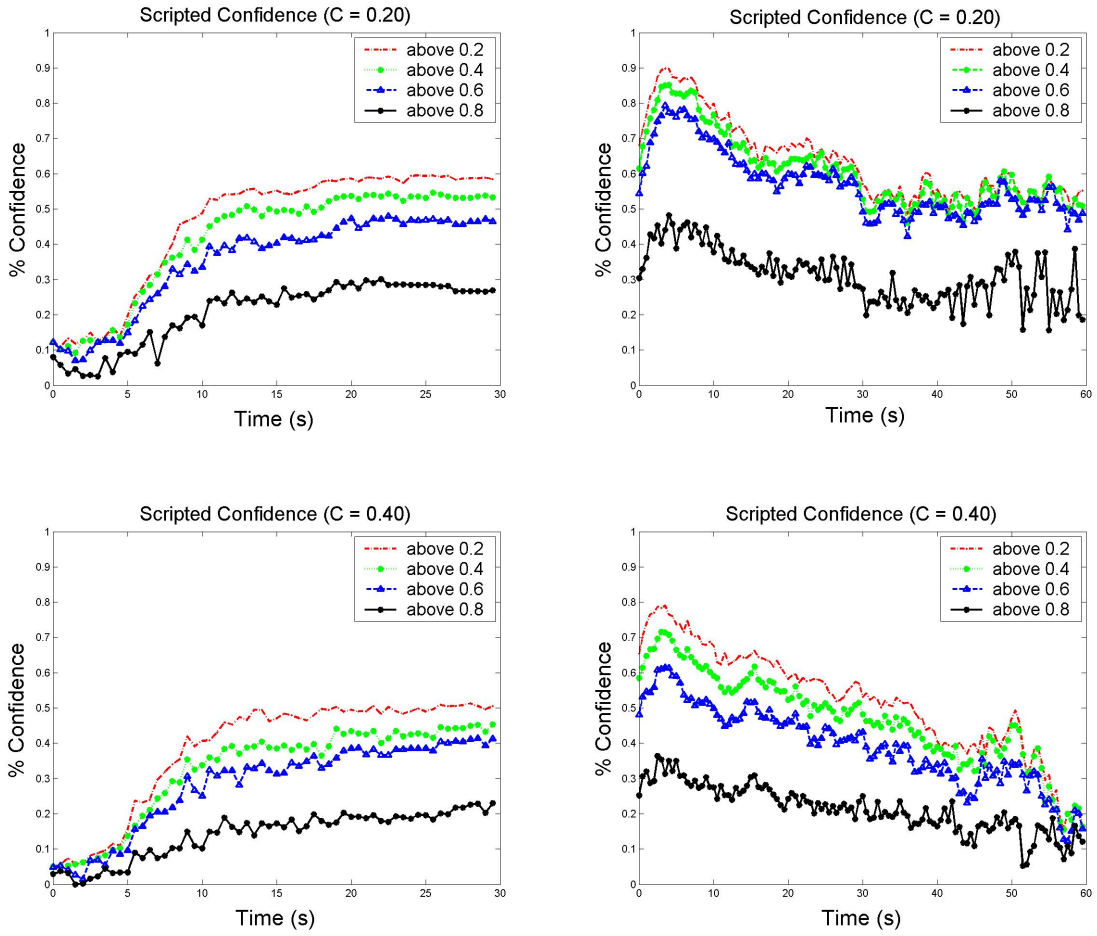

Figure 7.20: *Shows the probability that a robot (employing Scripted) will perform above various levels at world complexitis of 0.20 (above) and 0.40 (below). The graphs at left show probabilites during interactions ($t_N = 60$ seconds) and those at right show probabilities when the robot is being neglected.*

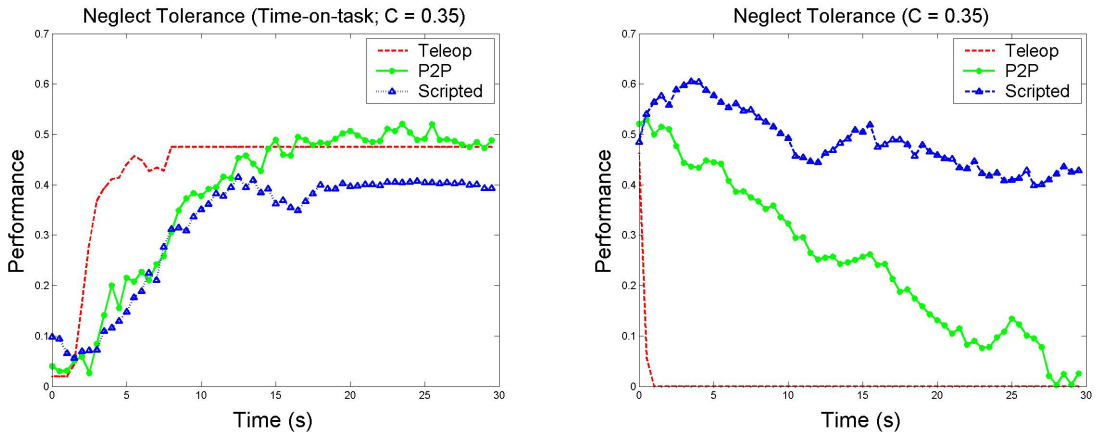

Figure 7.21: *Plots comparing the expected performance levels of robots employing the three interaction schemes at a world complexity of 0.35. The plot at left shows expected robot performance levels during interactions (after significant neglect times), and the plot at left shows expected robot performance when the robots are being neglected.*

a global map of the world. Without such a map, the interaction scheme is more difficult to use. Additionally, robots employing *Scripted* occasionally could not find their way around obstacles. Improving this aspect of the robot's autonomy mode would make the interaction scheme much more desireable.

7.3.4 Interaction Scheme Comparison

In this subsection, we compare the three interaction schemes according to their measures of neglect tolerance and interface efficiency.

Figure 7.21 compares the three interaction schemes at a world complexity of 0.35. The plots show the expected performance of robots employing each of the three interaction schemes. Figure 7.21(left) shows the expected performance of robots during interactions after significant neglect time. From the figure, the performance of a robot employing *Teleop* reaches peak expected performance levels much quicker than do robots employing the other two interaction schemes. The other two interaction schemes peak at about the same time. *P2P* peaks at about the same level that the *Teleop* interaction scheme does. However, the *Scripted* interaction scheme peaks at lower levels. This result can be

somewhat deceiving. This graph was created by assuming that an operator quit servicing a robot when the robot had reached peak performance levels. This led to the assumption that if servicing had continued, the robot would have continued performing at this same level. This assumption is false with the *Scripted* interaction scheme, as seen in Figure 7.21(right), which shows expected performance level during times of neglect. In this figure, the expected performance level of a robot employing *Scripted* continues to increase after interactions ceased.

Figure 7.21(right) shows the differences in neglect tolerance for the three interaction schemes. After 30 seconds of being neglected, a robot employing *Scripted* is still expected to be performing at about 40% of capacity, while robots employing the other two interaction schemes have already reached, or approached, zero.

Figure 7.22 shows the time it takes for a robot employing each of the three interaction schemes to reach peak expected performance levels during interactions after significant neglect times. In this case, "peak" is defined as 85% of the highest performance levels reached. As can be seen, *Teleop* peaks much quicker than do the other interaction schemes for all levels of complexity. Robots employing *Scripted* and *P2P* each take about the same amount of time at very low complexity levels. However, at medium and high complexity levels, it takes a few more seconds for a *P2P* robot to peak than it does for a *Scripted* robot to peak. Notice the spike in the graph of *P2P* at medium complexity levels. At these points, the robot reaches fairly high expected performance levels just as quickly as at other complexity levels, but did not reach 85% of peak performance for some time.

The last two figures indicate that performance peaks during interactions much quicker with *Teleop*. Additionally, *P2P* has no advantage in this way over *Scripted* (for this task), since *Scripted* takes less time to reach peak performance levels, for most complexities, while having much more tolerance to neglect.

We next compare the average frequency and duration of interactions that the combination of interface efficiency and neglect tolerance derives for a couple of different MAPLs. Figure 7.23 shows the average interactions for a MAPL of 50% (above) and of 70% (be-

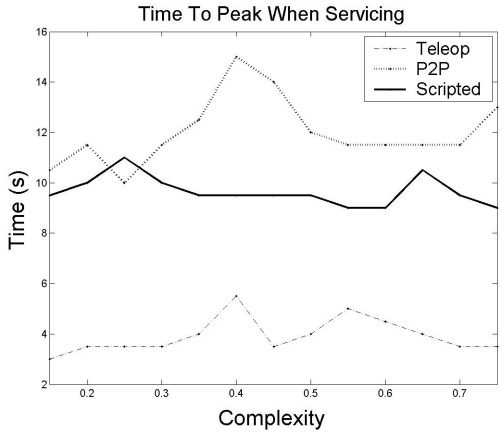

Figure 7.22: *The time it takes for a robot's expected performance to reach 85% of the peak expected performance when servicing begins. For* P2P *and* Scripted*, data is collected after a neglect time of 30 seconds, while data from* Teleop *is calculated after various neglect times.*

low). *Teleop* interaction times are based off of the time it takes for the robot to reach peak expected peak performance levels. Again, *Scripted* dominates $P2P$. When *Scripted* is employed, average interactions are shorter and tolerance to neglect is much greater than when $P2P$ is employed for all levels of world complexity, except interaction duration at very low world complexities. While bringing a robot up to peak performance levels is much shorter for *Teleop* than the other two interaction schemes, it allows for no neglect. Therefore, it is not desireable for many circumstances.

Figure 7.24 compares the three interaction schemes in terms of RAD and average robot performance. Plots are shown for world complexities of 0.20, 0.40, 0.60, and 0.70. In the plots, each of the points is taken from the MAPL that maximizes the robot's average performance for each of the interaction schemes. Again, interaction schemes are typically better if their points are in the upper left-hand corner of the plots, and are not as good if they are towards the bottom right-hand corner of the plots. As can be seen, $P2P$ approaches the bottom right-hand corner as complexity increases much faster than does *Scripted*. The expected performance levels of both *Teleop* and *Scripted* do not decrease much (comparatively) with increased world complexity. Again, however, *Teleop* requires

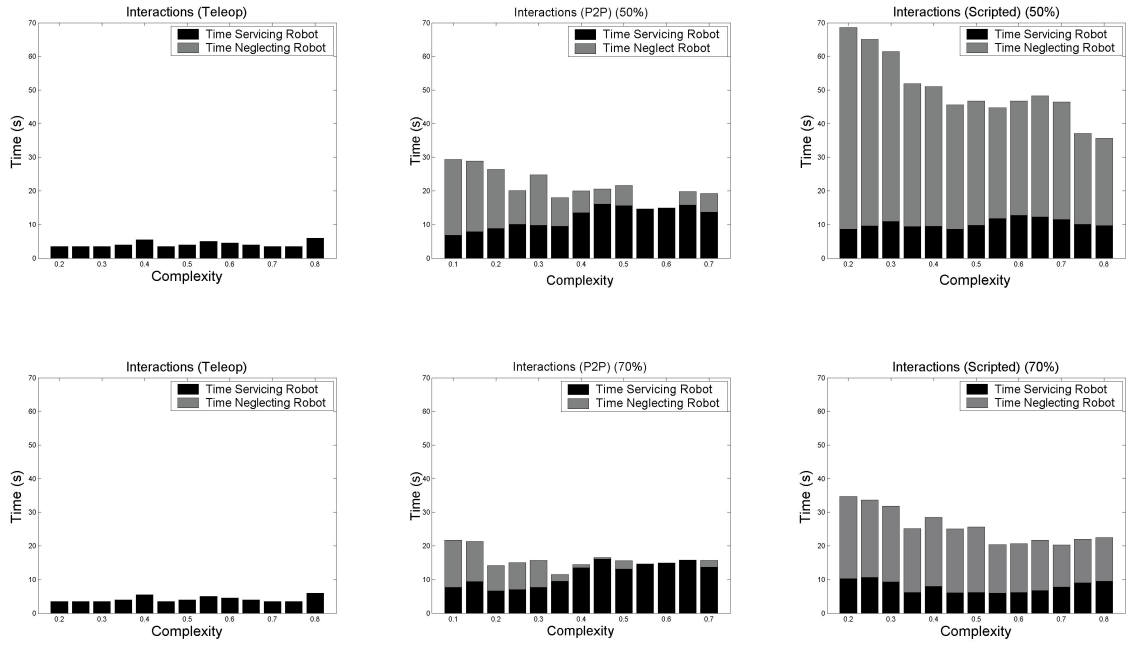

Figure 7.23: *Shows a comparison of the average interactions of each of the three interaction schemes with a minimum acceptable performance level of 50% (above) and of 70% (below). Teleop interaction times are based off of the time it takes for the robot to reach peak expected peak performance levels.*

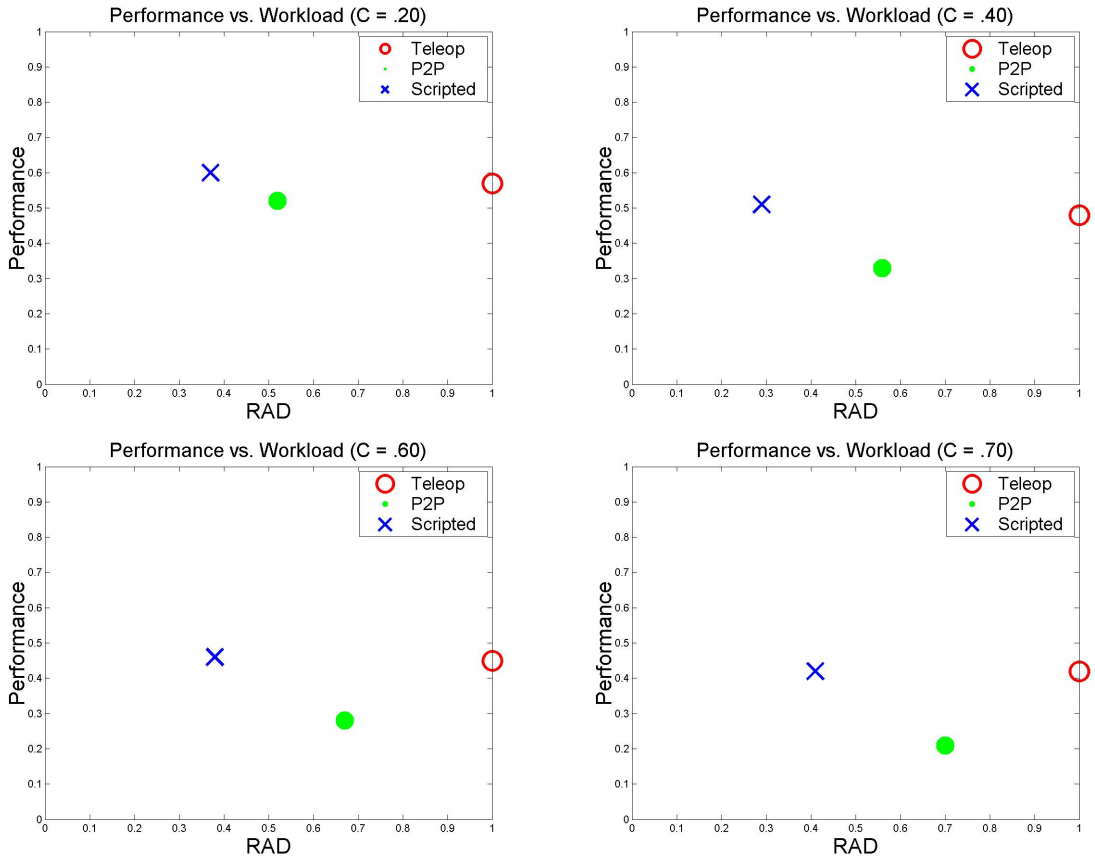

Figure 7.24: *Comparison of the three interaction schemes in terms of operator workload and performance for four different world complexities. In the graphs, points in the upper left-hand corner are more desireable than points in the lower right-hand corner.*

100% operator workload, while *Scripted* requires much less.

From these graphs and discussions, we have shown that for the navigation task tested in the simulated world, *Scripted* almost completely dominates $P2P$. Additionally, it obtains as high or higher average robot performance levels as does $Teleop$, while requiring much less operator workload. While these results are true in a simulated world, this does not necessarily mean that they will be true in the real world, although we would expect the same general trends to exist.

7.3.5 Analyzing the World Complexity Metric

As mentioned previously, some of the data seems to show that there is perhaps a small problem with the world complexity metric that we have been using. Such problems seem to appear somewhat in Figures 7.7(left) and 7.14(left). In these two figures, complexity significantly affects robot performance during interactions at the extremes of world complexity. However, through medium complexity levels, expected performance evens out, and even exhibits slightly increased expected performance for some higher world complexities than lower world complexities.

We explain why this happens by showing the mean performance of the random processes for *Teleop*, *P2P*, and *Scripted* using different world complexity metrics. Figure 7.25 shows the mean of the random processes if only the clutter estimate from our world complexity metric is used to estimate the complexity of a robot's environment. With the exception of noise, caused by a lack of data samples at the extremes of complexities, the trends shown are what we would expect in all the random processes. The trends actually seem to be followed even more closely than do the random processes obtained using the original complexity measure. So why not just use this world complexity metric instead? We answer this question later.

Figure 7.26 shows the mean of the random processes if only the branching estimate from our world complexity metric is used to estimate the complexity of a robot's environment. The results shows that *Teleop* is affected somewhat by branching complexity, although certainly not to the same extreme that it was affected by clutter complexity. *P2P* shows the same trend, although a little bit more strongly than does *Teleop*. During interactions (bottom-left), *Scripted* shows almost a wave-like behavior, with performance dropping quickly at the high extreme of world complexity. We cannot explain this, except to say that it appears that branching complexity affects *Scripted* very little during interactions, if at all. The neglect tolerance measure of *Scripted* shows no wave-like behavior, and follows the trend that expected robot performance degrades more quickly in complex environments as neglect time increases.

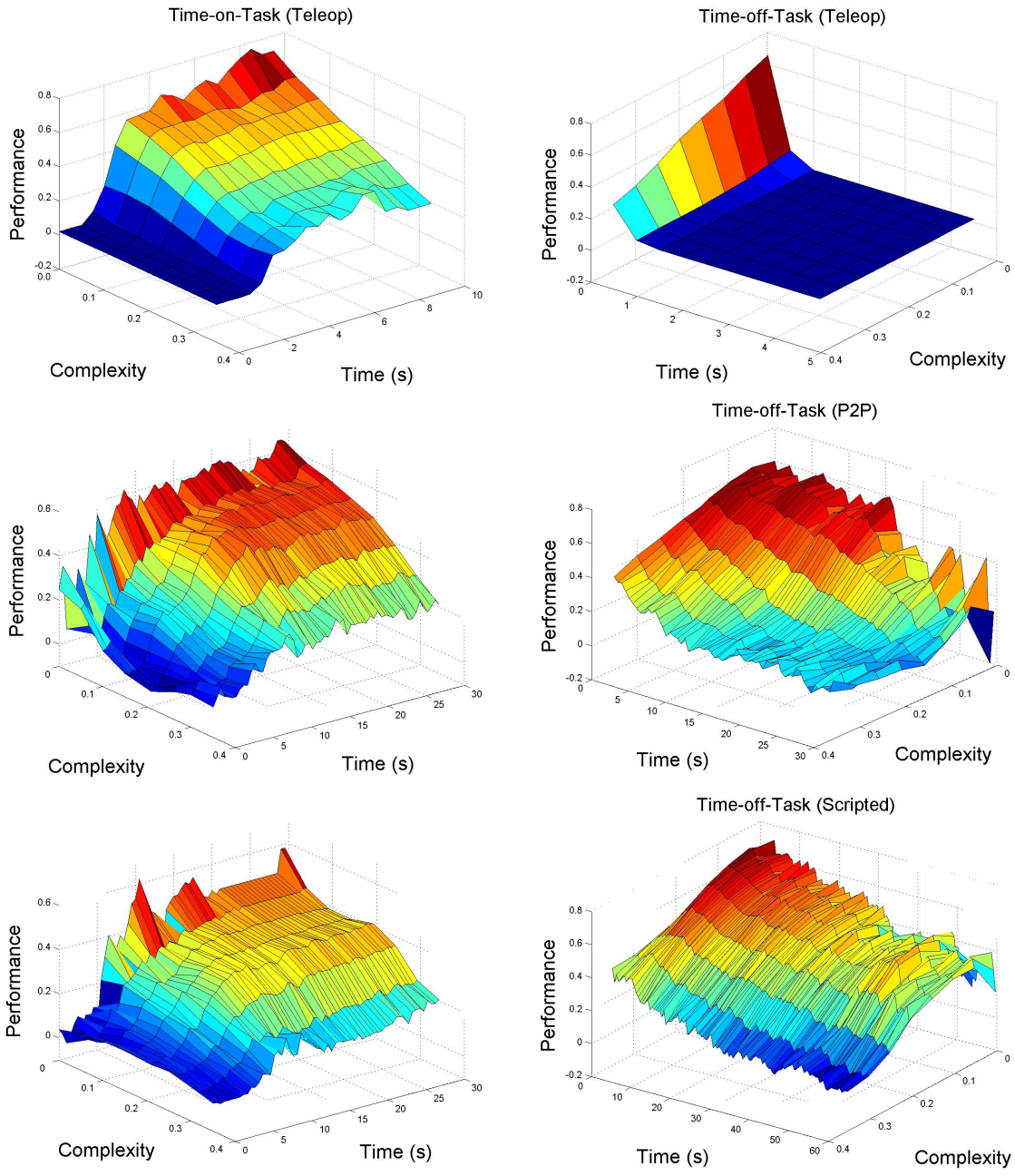

Figure 7.25: *The mean of the random processes* $V(Teleop)$, $V(P2P; t, c, t_N = 30sec.)$, *and* $V(Scripted; t, c, t_N = 60sec.)$ *using only the clutter estimate (omitting the branching estimate) for the world complexity metric.*

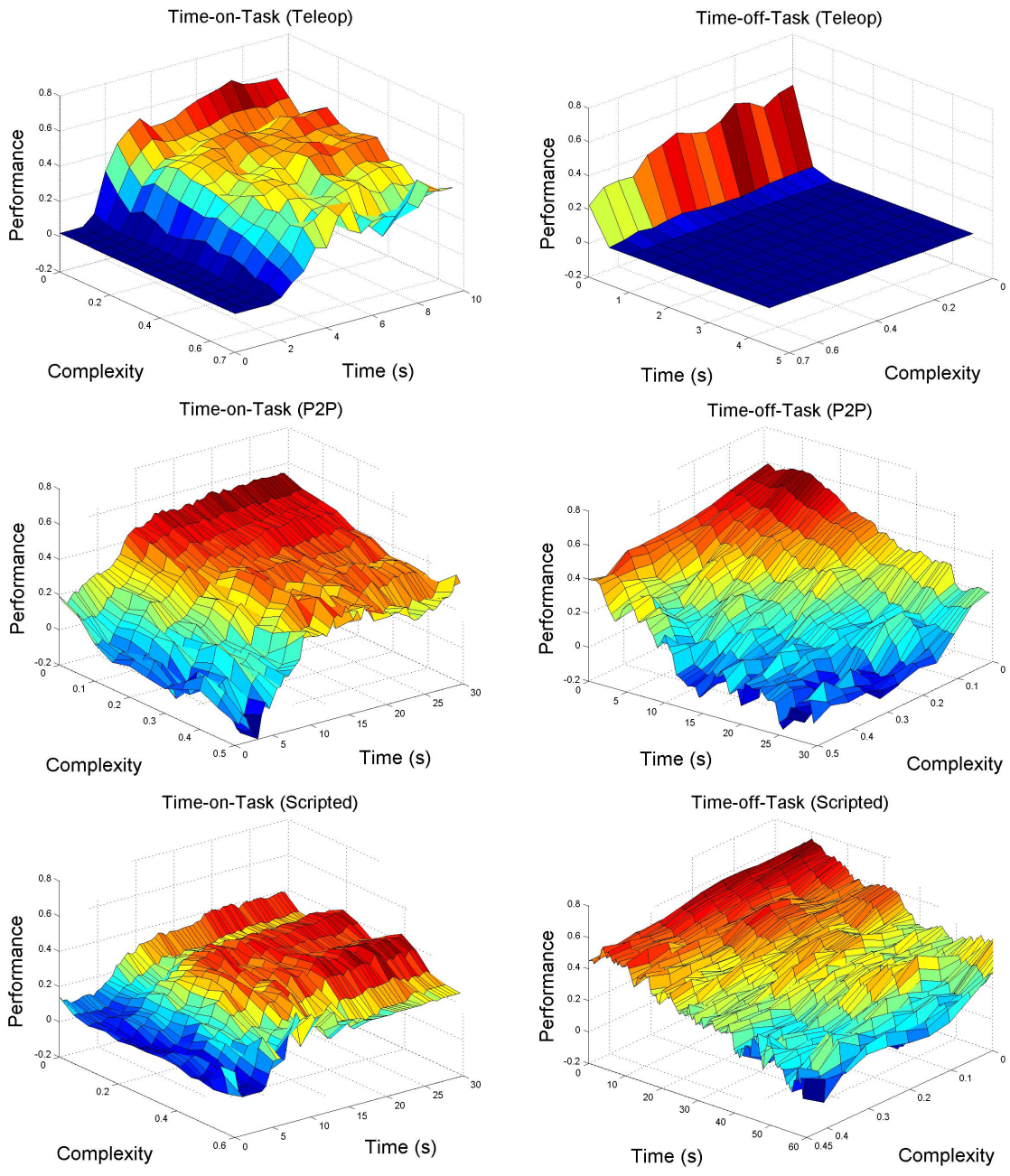

Figure 7.26: *The mean of the random processes $V(Teleop)$, $V(P2P; t, c, t_N = 30sec.)$, and $V(Scripted; t, c, t_N = 60sec.)$ using only the branching estimate (omitting the clutter estimate) for the complexity metric.*

Since, in general, both clutter and branching factors do indeed make the task of navigation more difficult for a robot, we are not justified in omitting the one or the other from our complexity metric. However, as has been seen from the results, a weighted sum of the two estimates is not perfect either and this appears to be the biggest problem with our world complexity metric. However, it is still a useful world complexity metric for the navigation task since it is able to estimate world complexity fairly well.

7.4 Chapter Summary

In this chapter, we presented the results of a user study involving 40 test subjects. From the data logged during the user study, we were able to measure the neglect tolerance and interface efficiency of three interaction schemes for a navigation task. The measures of neglect tolerance and interface efficiency helped identify the strengths and weaknesses of each interaction scheme.

The neglect tolerance and interface efficiency metrics showed that *Scripted* was almost universally superior to $P2P$ for the navigation of a robot through a maze world. Also, *Scripted* was able to obtain nearly the same performance levels as $Teleop$.

We also analyzed the world complexity metric that was used. While the complexity metric is not perfect, it appears to be sufficient to allow comparison of different interaction schemes at different world complexities.

In the next chapter, we will discuss how the neglect tolerance and interface efficiency metrics can be estimated using only a few user experiments.

Chapter 8

Approximating Neglect Tolerance and Interface Efficiency

It took 40 test subjects, each performing three ten-minute test sessions, to estimate the neglect tolerance and interface efficiency of the three interaction schemes described in chapter 5. In many situations, however, such a large number of experiments are costly and impractical. In this chapter, we discuss how approximations of neglect tolerance and interface efficiency can be made using only a few user experiments. First, we will describe the algorithm used to make the approximations, after which we will show the results.

8.1 Approximation Algorithm

To approximate measures of neglect tolerance and interface efficiency for an interaction scheme, the same user experiments described in the previous chapter are performed, only not as many are needed. For good results, these test sessions should be run in worlds of varying complexities.

Once the experiments are run, measures of interface efficiency and neglect tolerance are obtained as before, with three exceptions. First, in the previous chapter, a sample obtained at time t_0, complexity c_0, and neglect time t_{N_0} contributed only to the nonparametric estimate of the random variable $V(\pi; t = t_0, c = c_0, t_N = t_{N_0})$. However, when not enough

samples are available, we can make the assumption that a data sample obtained with neglect time t_{N_0} would be similar to data sample obtained with neglect time t_{N_1}. To give this affect in the random process $V(\pi)$, each random variable $V(\pi; t = t_0, c = c_0, t_N = t_{N_1})$ is influenced by a sample occurring at time t_0, complexity c_0, and neglect time t_{N_0} by the amount $\frac{M}{|t_{N_1} - t_{N_0}| + M}$, where M is the distance between the *bins* along the neglect time axis, t_N of the discretized random process $V(\pi)$ (see Section 7.2.1). For example, with *Scripted*, the distance between the bins along t_N is ten seconds, so M equals ten for *Scripted*.

Second, in chapter 7, a gaussian filter with a small variance of 0.05[1] was used along the complexity axis to smooth the data. Since less samples are available when we only use a few user experiments, we must use a gaussian filter with a larger variance to adequately smooth and distribute the data. In this thesis, we used a variance of 0.15[2]. Increasing the variance on this gaussian filter makes the assumption that there are no sharp changes in performance along the complexity axis of the random process.

Third, a filter is also applied along each value of the time axis t of the random process $V(\pi)$. This filter is [1, 2, 1], meaning

$$V(\pi; t_i, c_0, t_{N_0}) = \frac{V(\pi; t_{i-1}, c_0, t_{N_0}) + 2V(\pi; t_i, c_0, t_{N_0}) + V(\pi; t_{i+1}, c_0, t_{N_0})}{4}.$$

This smooths the data along this axis.

8.2 Results

Using the method described above, we obtained the results shown in Figure 8.1. The figure shows the means of the approximated random processes $V(Teleop)$, $V(P2P)$, and $V(Scripted)$. To approximate $V(Teleop)$, three ten-minute test sessions were performed, each with a different test subject. $V(P2P)$ and $V(Scripted)$ were approximated using

[1]This variance was chosen since it seemed to work well. This value can change depending on the distribution from which the world complexity estimates are drawn. Since this distribution is unknown, we have no way of specifying what the variance should be for the generic case.

[2]Again, this value is arbitrarily chosen and seems to work well for the data obtained for this thesis. However, different variances can be used.

three test subjects each, where each test subject performed two ten-minute test sessions. Sessions from the case study in the previous chapter were randomly selected for use in this case study.

The trends discussed throughout this thesis are generally born out in Figure 8.1. As can be seen, for the most part, as world complexity increases, robot performance decreases. There are a few exceptions to this. For example, the interface efficiency graph of *Scripted* does not demonstrate this phenomenon. However, overall, the graphs show the same basic results that Figures 7.3, 7.7, and 7.14 show.

Figure 8.2 shows the difference between the mean of the random processes obtained using the measurement technology described in chapter 4 (and shown in Figures 7.3, 7.7, and 7.14), and the mean of the random processes obtained using the approximation algorithm. The figure shows that the differences in the mean between the two sets of random processes is usually bounded by 0.2.

Since the small number of samples obtained are not statistically significant, we also show a second set of experiments, also randomly selected, to demonstrate the differences that different users and different worlds cause to occur. Figure 8.3 shows the means of these estimated random processes. The results shown in Figure 8.3 are similar to those shown in Figure 8.1. While there are differences, which are to be expected, the trends in interface efficiency, neglect tolerance, and complexity tolerance are generally born out in the results.

Figure 8.4 shows the difference between the mean of the random processes obtained using the measurement technology described in chapter 4 (and shown in Figures 7.3, 7.7, and 7.14), and the mean of the random processes obtained using the approximation algorithm tih the second set of experiments (Group2). Like the difference plots using the first set of experiments (Group1), the figure shows that the differences in the mean between the two sets of random processes is usually bounded by 0.2. Notice, however, that some of these difference plots show the trend of higher differences around the extremes of world complexity because the approximation algorithm has more difficulty approximating

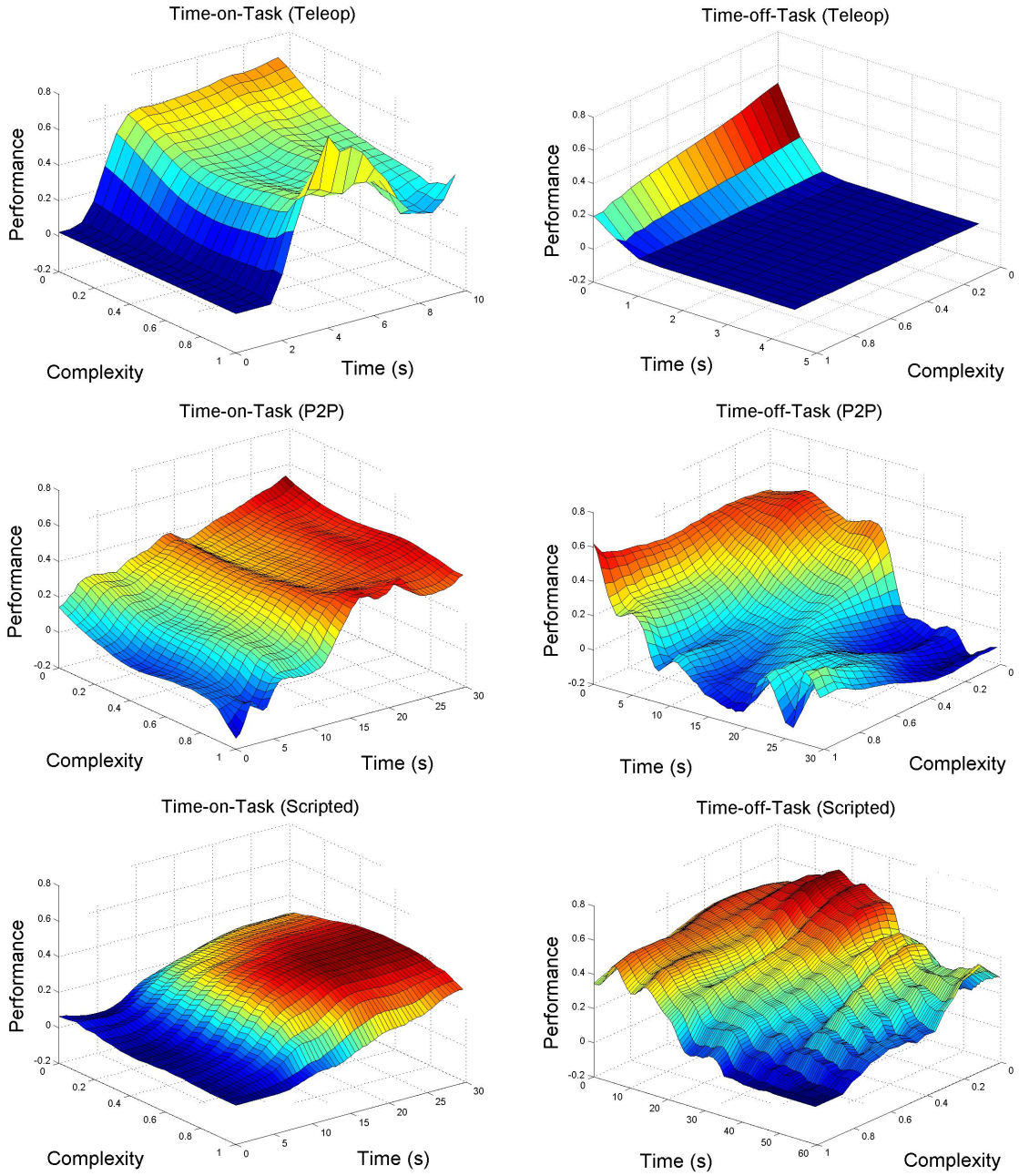

Figure 8.1: *Shows the mean of the random processes* $V(Teleop)$, $V(P2P)$, *and* $V(Scripted)$ *obtained by the approximation algorithm for one set of experiments.*

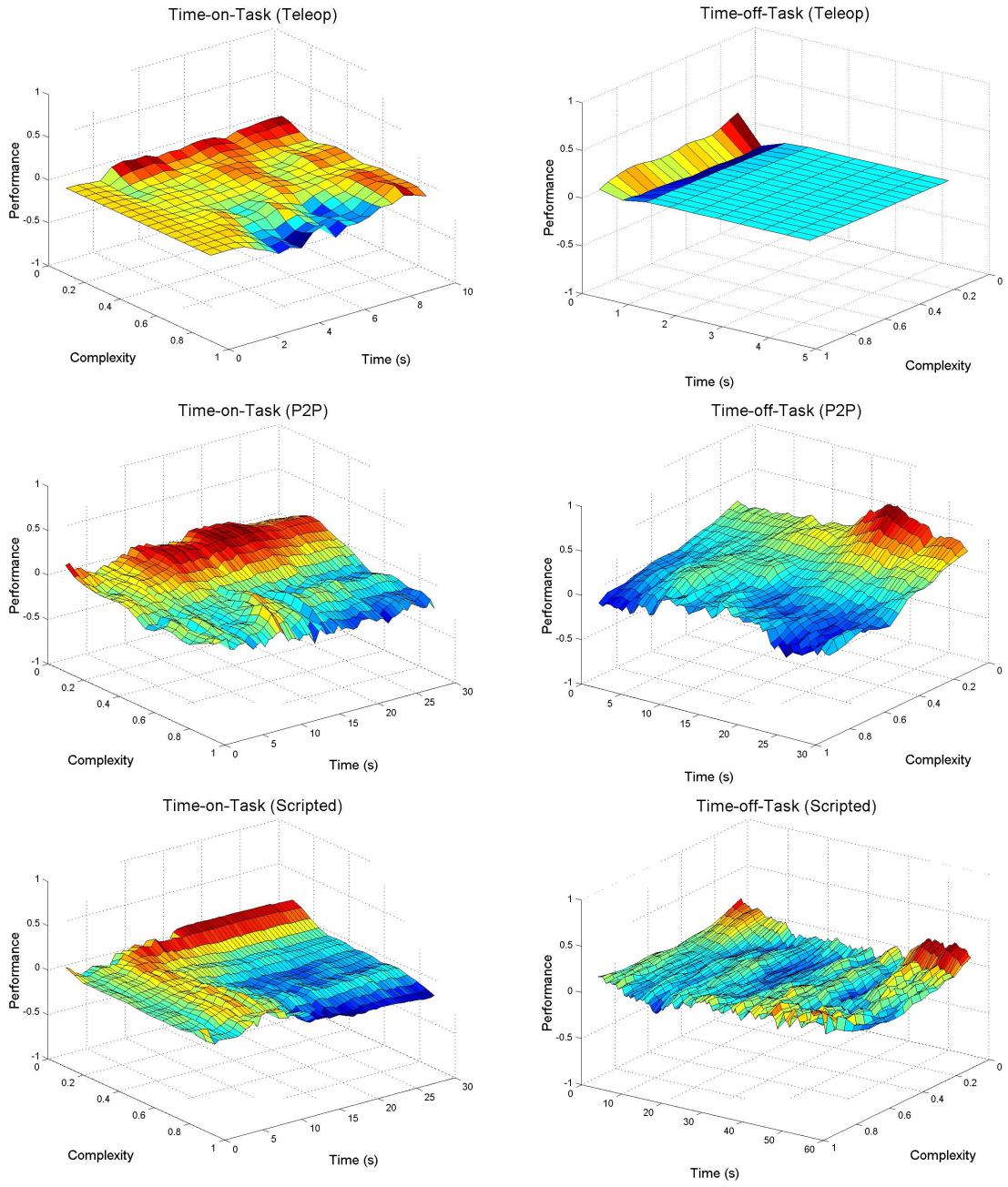

Figure 8.2: *Shows the difference between the mean of the random processes obtained using the measurement technology described in chapter 4 (and shown in Figures 7.3, 7.7, and 7.14), and the mean of the random processes obtained using the approximation algorithm for the first set of experiments used (shown in Figure 8.1).*

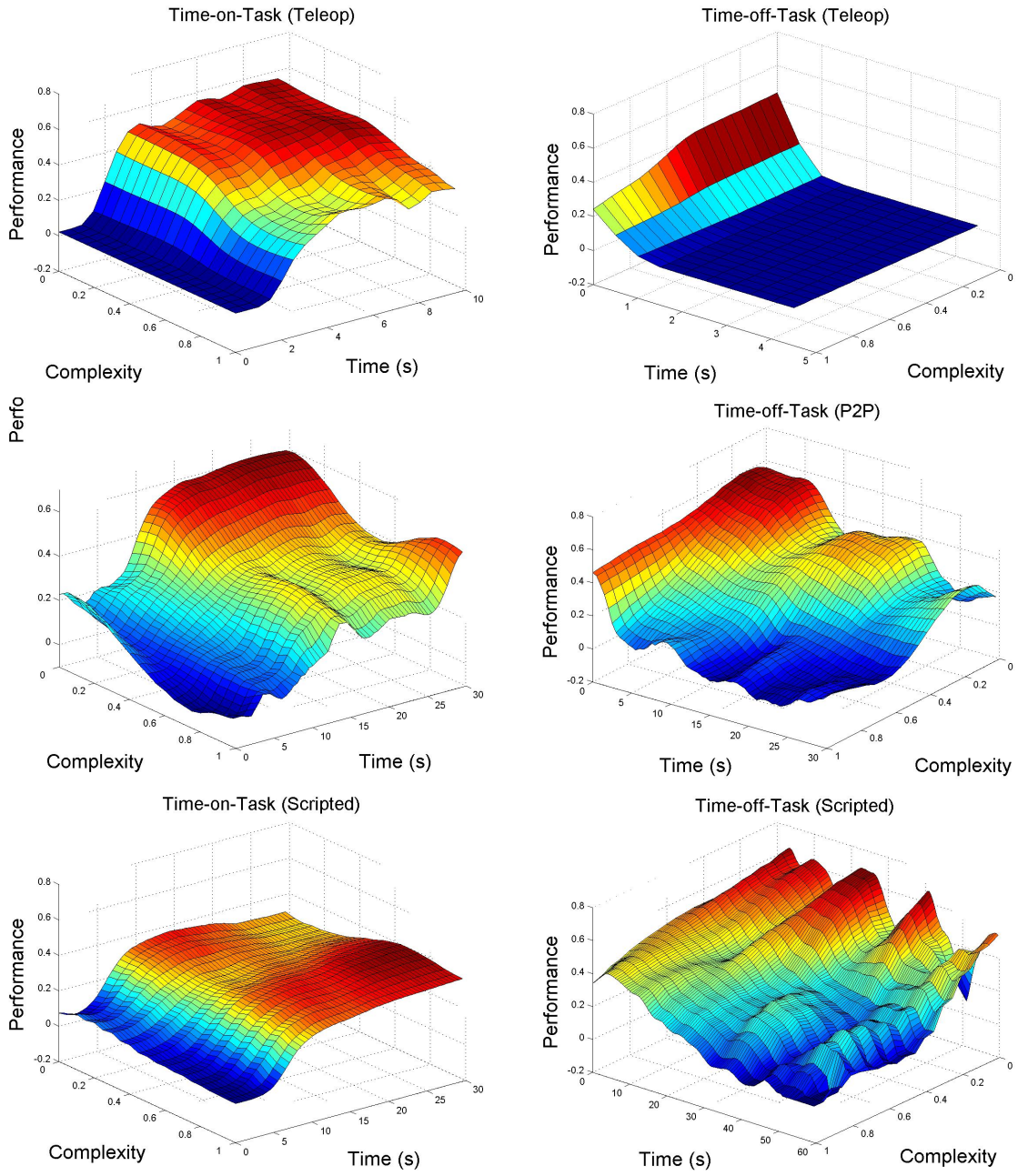

Figure 8.3: *Shows the mean of the random processes* $V(Teleop)$, $V(P2P)$, *and* $V(Scripted)$ *obtained by the approximation algorithm for a second set of experiments.*

expected robot performance at the the extremes of world complexity. However, in general, it does a pretty good job of approximating neglect tolerance and interface efficiency.

Figures 8.5 and 8.6 show the average frequency and duration of interactions that should occur between human and robot, with a MAPL of 50% of peak robot performance levels, for the two different sets of user experiments. The results are similar to those shown in Figure 7.23, and thus indicate that the approximation algorithm is quite effective. Just like the results from the larger user study, $P2P$ is basically dominated by *Scripted*.

Figures 8.7 and 8.8 show average robot performance plotted against RAD for MAPLs that maximize average robot performance. Again, the trends are quite similar to those obtained in chapter 7 (see Figure 7.24) for both both test groups.

Thus, while the the results presented in this chapter are not identical to those presented in the previous section, they are similar. Therefore, the approximation algorithm presented in this chapter appears to be an adequate solution (in some circumstances) to the cost of performing the large number of user experiments that is required by the algorithm presented in chapter 4 and shown in chapter 7. However, when high precision is needed, the approximation algorithm may not be useful.

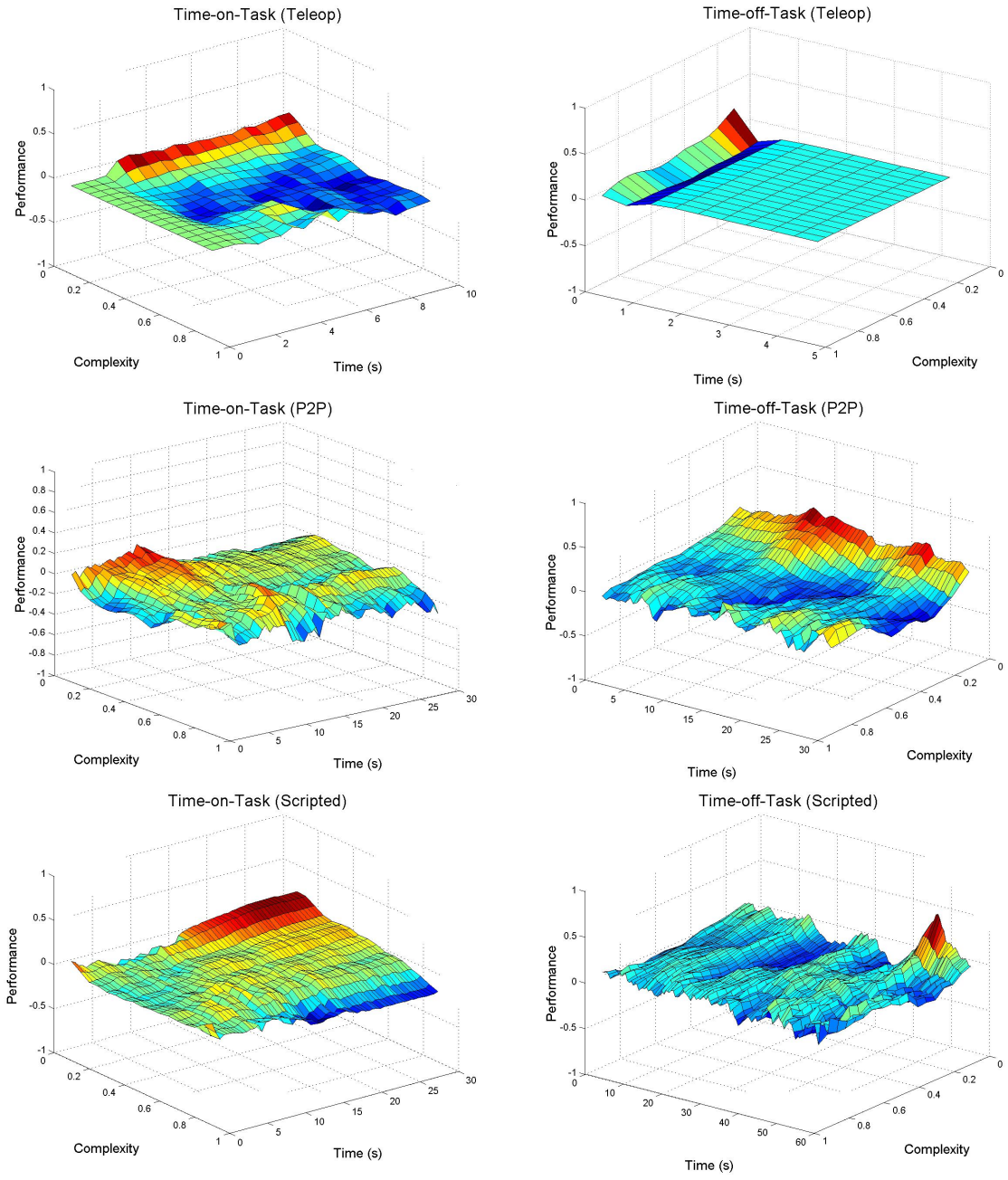

Figure 8.4: *Shows the difference between the mean of the random processes obtained using the measurement technology described in chapter 4 (and shown in Figures 7.3, 7.7, and 7.14), and the mean of the random processes obtained using the approximation algorithm for the second set of experiments used (shown in Figure 8.3).*

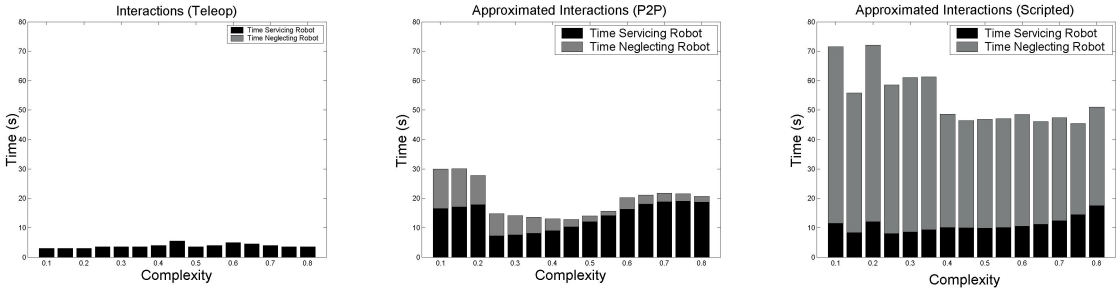

Figure 8.5: *Shows the average frequency and duration of interactions that should occur for the three interaction schemes when a MAPL of 50% of peak robot performance levels is used. This data was gather from Group1.*

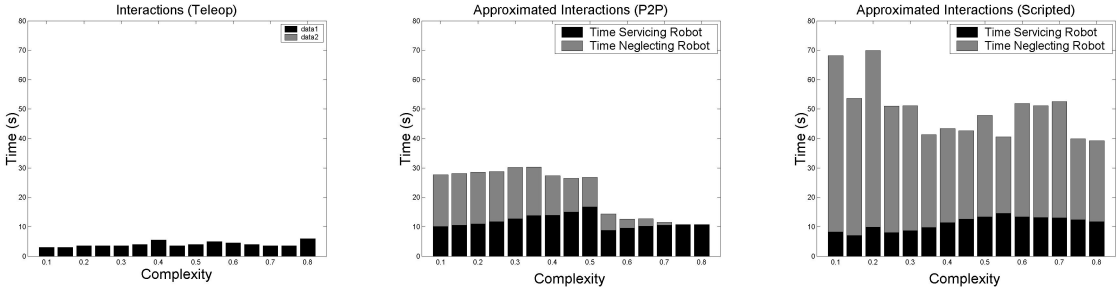

Figure 8.6: *Shows the average frequency and duration of interactions that should occur for the three interaction schemes when a MAPL of 50% of peak robot performance levels is used. This data was gather from Group2.*

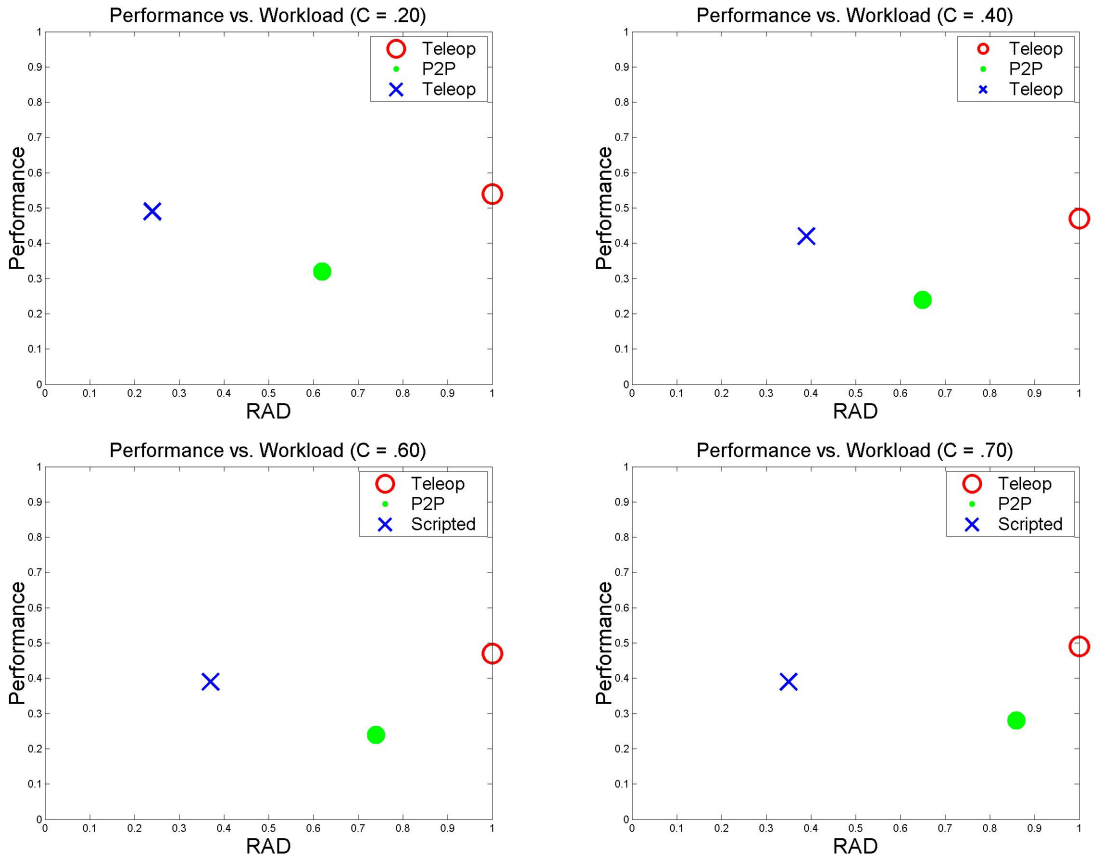

Figure 8.7: *Comparison of the three interaction schemes in terms of operator workload and performance for four different world complexities using Group1. In the graphs, points in the upper left-hand corner are more desireable than points in the lower right-hand corner. The results are similar to those shown in Figure 7.24.*

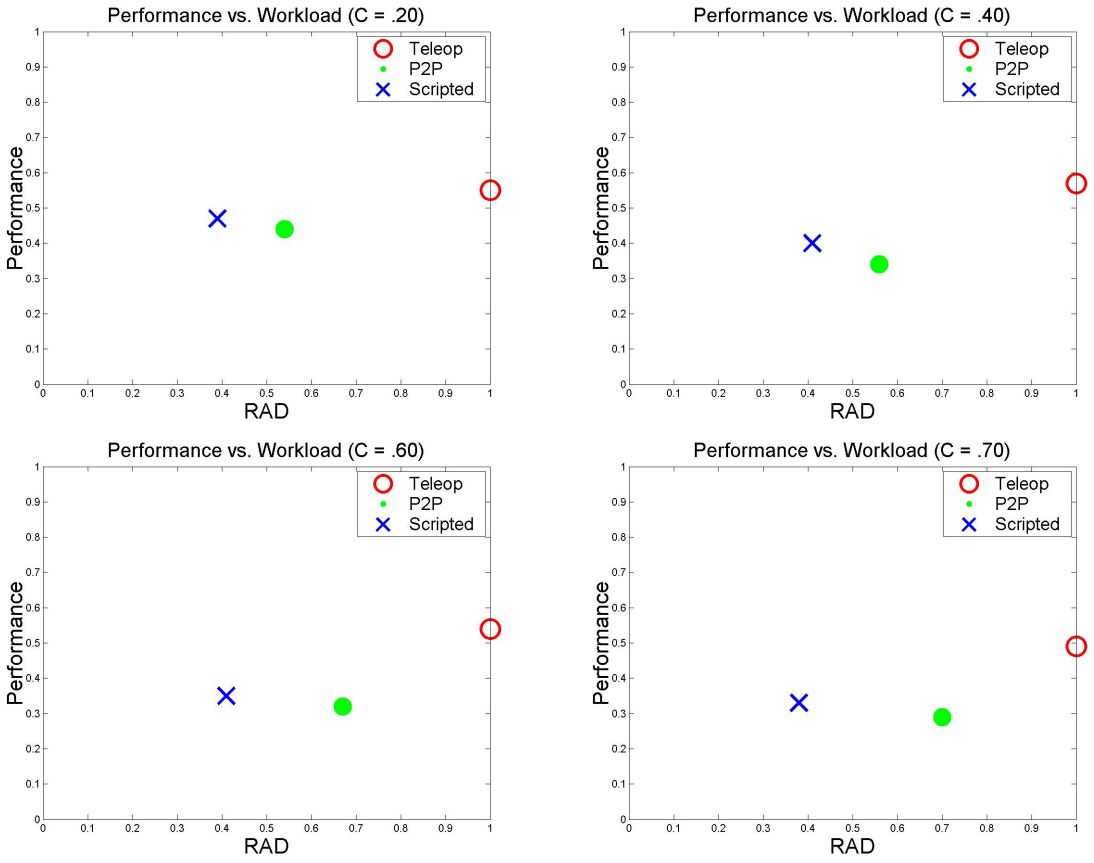

Figure 8.8: *Comparison of the three interaction schemes in terms of operator workload and performance for four different world complexities using Group2. In the graphs, points in the upper left-hand corner are more desireable than points in the lower right-hand corner. The results are similar to those shown in Figure 7.24.*

Chapter 9

Conclusions and Future Work

In chapter 3 we reported a user study that compared shared-control teleoperation with manual-control teleoperation. In the user study, a human operator navigated a mobile robot through an indoor environment using the two teleoperation modes. The shared-control method increased robot performance over the manual-control method, as well as lowering operator workload. This user study indicates that changing a interaction scheme, which is the combination of a robot's *autonomy mode* and the *interface* between human and robot, changes both system performance and operator workload.

In chapter 4 we defined metrics for interface efficiency and neglect tolerance in human-robot systems. Measures of interface efficiency and neglect tolerance for human-robot interaction schemes are estimated via a measurement technology that uses secondary tasks in user studies. These measures can be used to predict the average human-robot interactions that must take place, predict the performance of robots employing these interactions, and identify the strengths and weaknesses of an interaction scheme.

In chapter 5 we described three different interactions schemes that were used to validate the usefulness of the neglect tolerance and interface efficiency metrics. The three interaction schemes were shared-control teleoperation ($Teleop$), point-to-point ($P2P$), and scripted ($Scripted$).

In chapter 6 we described the environment in which we performed a large user study to validate the usefulness of the neglect tolerance and interface efficiency metrics. We used

simulated worlds because they allowed the use of many worlds with varying complexities. We performed in the simulated world the same experiment performed in the real world that was reported in chapter 3. The results from this experiment showed that the same trends in robot performance and operator workload exist in the simulated world as exist in the real world. Therefore, the simulated world is sufficient to validate the usefulness of the neglect tolerance and interface efficiency metrics.

In chapter 7 we described a user study involving 40 test subjects. The user study was performed in accordance to the measurement technology discussed in chapter 4. From the user study, data was gathered to measure the neglect tolerance and interface efficiency of each of the three interaction schemes described in chapter 5. We then analyzed the strengths and weaknesses of each interaction scheme as well as the average frequency and duration of human-robot interactions that should occur for each interaction scheme. We also showed the trade-offs in operator workload (RAD) and average robot performance for each interaction scheme.

In chapter 8 we described how interface efficiency and neglect tolerance can be approximated using only a few test subjects. The approximated measures show the same trends as those obtained in chapter 7. Thus, the metrics can be used even when large user studies are not practical.

The main contribution of this work is the neglect tolerance and interface efficiency metrics, as well as the accompanying measurement technology. From the metrics, characteristics of human-robot systems such as robot performance, operator workload, and acceptable human-robot interactions can be estimated and predicted. These metrics can be used for human-robot system design and other applications.

The metrics can also be powerful in conjunction with the principles of adjustable autonomy. Because robot performance and the necessary operator workload of an interaction scheme can estimated via these metrics, these metrics can be used by robots to adjust their autonomy. In [13] we extend the work done in this thesis to begin to facilitate this, but we leave the main portion of this idea to future research.

While we showed that the metrics and measurement technology described in this thesis can be useful, it would be desirable for humans and robots to be able to learn the proper interaction schemes that should be employed. For robots, this would mean the development of a learning algorithm to determine the proper interaction schemes to employ at a given time (see [23]). Such a learning algorithm would rely implicitly on the concepts of interface efficiency and neglect tolerance discussed in this thesis.

Appendix A

Neglect Tolerance Experiments Script

Included here is the script which was read to the test subjects during the experiments explained in chapter 7. Comments meant for the experiment presenter but not meant to be read are included in *italics*.

Introduction

You will be asked to guide two robots at a time through three different worlds for ten minutes each. You will use a total of two control methods, and you will be trained appropriately on each. The total duration of the experiment should last about 1 hour (give or take a few minutes). If you don't currently have that much time, we would appreciate it if you would reschedule a time that you would have that much time.

Make sure the test subject has at least 1 hour to take the experiment. If they do, have them read and sign the form. Record their name.

In addition to navigating the robots through the world, you will be asked to perform two-digit addition and subtraction problems. Multiple choices will be provided. You will be given a score based on how well you control the robots and how well you complete math problems. Scores will be kept anonymous, however, so you need not fear that you will be made fun of for your efforts. If you desire, however, you may ask for your score and how

you compare with others that have taken the experiment. We will now proceed with the experiments.

GUI Explanation

Fire up the program.

Enter the first and last name you wish to be known by. The first name and last name must be separated by a space. Press the return key when you have finished.

Make sure they do it correctly. When they have done it correctly, a window should pop up which will have a big "Train Me" button on it.

When you are ready to proceed with training on the first control method, press the button on the screen labeled "Train Me."

Wait for them to push the button.

I will now explain the user interface that is before you on the screen. The main portion of the user interface shows a map of the world. The map is a connected graph of the world. Each node is in effect a landmark (or intersection), and indicates general directions that the robot should be able to go from that position in the world. This graph isn't perfect, but it is reliable. Blue and green triangle-like objects on the map indicate the positions of the robots in the world. (The robots are not actually this shape, however.) Additionally, blue and green colored squares around nodes in the world indicate goal positions for the robots respectively. When a robot reaches the goal position, it will automatically be given a new goal position. Currently, only the blue robot is in the world, but in testing conditions, both robots will be present.

The lower portion of the GUI shows the sensory information of the robot that you are currently controlling. The background color of this portion of the screen indicates which robot you are currently controlling. If you are having problems locating the robot or its goal on the map, you may click on the "locate robot" and "locate goal" buttons in the lower right hand corner of the GUI. A line will appear that goes from the clicked button to the robot or goal. This line will disappear after a predetermined amount of time. Feel free to click on these buttons to get accustomed to them.

Pause to give them time to play around with the buttons for a second if they desire.

The robot is equipped with a camera, sixteen sonars (which are arranged in a ring around the robot), and a compass. A display of each of these sensors is displayed in the lower portion of the GUI. At the left is a graphic of the sonar readings. The light regions indicate areas of that the robot thinks it can go without hitting walls and obstacles, and the dark areas indicate places the robot thinks are obstacles. The robot is located in the center of this region and is facing up with reference to this graphic.

Make sure the user understands this display. It is important.

Next is a graphic of the compass on the robot. North is up on the map. Next is a black and white camera image from the robot. The camera is pointed in the direction the robot is facing.

Also included in this lower portion of the GUI is a control panel (to the right of the camera image). This will be different for each control mode. At the bottom right of the GUI is a button labeled "Done." This button is to be clicked when you are done servicing the robot. When you are done servicing the current robot, you should push this button so you can attend to other tasks such as answering math problems and controlling the other robot when it is time to service it.

Go ahead and click on the "Done" button. When you do so, the math display replaces the robot view in the lower portion of the GUI. A math problem (addition or subtraction) is displayed, and there are four answers to its right. To answer, simply click on the answer you think is correct. If you are correct, the answer will be displayed in green. If you are incorrect the answer will be displayed in red. When it is time for a robot to be serviced, this display will automatically be removed, and the robot view of the robot to be serviced will be displayed.

Directly above the robot view are small icons indicating the performance of each robot over the last 30 seconds or so. On the right side of the GUI (to the right of the map) are other various system indicators. At the top of this panel the time for this section is indicated (you will go until 10 minutes are up). Below that, the number of goals that

your robots have reached is displayed, followed by a display of time-based workload (which indicated the time when you serviced each robot). Under that is a display of overall robot team performance. 100 % indicates that all robots are operating absolutely perfectly, and isn't achieved realistically. Don't panic if this percentage drops very low (although you want to keep it high). Your math performance is also displayed on the right hand side of the GUI. It isn't incredibly important that you understand all of these system indicators completely, but if you have any questions about any of them, feel free to ask now.

Make sure they don't have any questions about the system indicators.

Next, you will train them on a control mode. Locate the script for the current control mode they are to be trained on. After you are done with that, return to this spot in the script.

Repeat steps 1 - 6 three time

1 Remember, your goal is to get the robots to reach as many goals as possible and answer as many math problems as possible in the allotted time. You must perform well on both to get a high operator rating.

2 *Read the script above the button to the test subject.*

3 *Let the test subject perform the experiment. Only help the test subject if they are having problems that are so obviously stupid you can't help it (like them forgetting for a long while to push in the trigger on the joystick). Try not to distract them in any way.*

4 *When the test subject is finished with this test session, have them stand up and take a break for a few minutes. They may want to proceed very quickly, but make sure they at least stand and stretch for a second.*

5 *The program will next indicate whether the test subject should be trained on a different control scheme, or if they should proceed with another test session on a previously performed control mode. If they need to be trained, locate the script for the current control mode they are to be trained on.*

6 *Jump to the beginning of the loop and proceed*

Thank you for participating as a test subject.

Teleoperation Training

If you haven't already, press the button labeled "Train Me."

You will now be trained on how to operate the robot for the coming session. In this session, you will use a joystick to control the robots.

The robots are equipped with assisted control when operated by the joystick. This means that the robots will automatically avoid obstacles. It won't crash into them. If you direct a robot towards an obstacle, the robot will turn away from it. With the joystick, you need only direct the robot in the general direction that you desire it to go.

To control the robot, push the trigger with your index finger, (help the test subject to find it) and point the robot joystick in the direction you want the robot to go. A red arrow will appear in the robot view control panel (point on the screen to where that is) that indicates the direction and the magnitude that you are giving the robot. Direction is with respect to the robot. You can increase the magnitude by pushing harder on the joystick. If you want the robot to spin in place, let go of the trigger and twist the joystick in the direction that you want it to spin.

Practice controlling the robot until you feel comfortable driving it through the training world. Practice driving the robot to its goal position. When you feel that you are ready to proceed with the test session, tell me.

Let the test subject drive the robot around until he/she feels comfortable, and indicates that they are ready to proceed.

When you are controlling the robot in this control mode, you will drive one of the robots for a predetermined amount of time. Control of this robot will be taken from you and you will be asked to answer a math problem. You will then be asked to control the next robot. The process will continue in this manner for ten minutes.

Close the GUI window. A new window now will pop up which is the window in which you will perform a ten-minute test session.

Point-to-Point Training

If you haven't already, press the button labeled "Train Me."

You will now be trained on how to operate the robot for the coming session. In this session, you will use a mouse to control the robots. This control method is called point-to-point control.

With this control method you control the robot by giving it commands of what it should do when it reaches the next intersection. For example, at the next intersection you may want to tell the robot to go left, or to proceed straight through the intersection. Commands are given to the robot by clicking on the buttons in the control panel *(if need be, point on the screen to where that is)*. To tell the robot to go straight at the next intersection, push the button labeled "Straight." To have the robot turn right the next chance it gets, push the button labeled "R." To have the robot turn left the next chance it gets, push the button labeled "L." After any of these buttons are pushed, the robot may be neglected. This is done by pressing the "Done" button in the lower right-hand corner of the GUI. This will allow you to attend to other tasks while the robot carries out the command.

Note that the robot will not always fulfill the commands as you feel that it should. This is because the robot isn't perfect. However, it should do pretty well in most circumstances. Practice will help you to predict how the robot will behave. Notice the sonar signature in the robot view. This could be of help, as the robot will turn (when commanded) when the sonars allow it to.

If need be, additional commands can be given to the robot. Push the button labeled "Stop" to tell the robot to stop moving. Push the button labeled "SL" to tell the robot to spin in place turning counterclockwise (or left), and push the button labeled "SR" to tell the robot to spin in place turning clockwise (or right). The robot can be made to go backward by pushing the button labeled "Back."

The basic protocol you should follow with this control scheme is to service the current robot (i.e., give it the command or sequence of commands that will get it moving towards

its goal). Then press the "Done" button to move to a different task (either performing math calculations or servicing the other robot). When it is time to service a robot again, you will be given control (robot neglect times vary and are determined by the computer). Changes in current task will be indicated by changes in the robot view.

Go ahead and practice driving the robot through the current world using this control method. Be sure to try to learn how the robot reacts to commands. You should also practice neglecting the robot by pushing on the "Done" button after giving the robot commands (when you feel that it is appropriate to do so). The "Done" button will give you control of a different task (such as servicing another robot or doing math problems) until it is time to again service that robot.

When you feel that you are ready to proceed with the test section, tell me and we will move on.

Wait for the test subject to practice.

Close the GUI window. A new window now will pop up which is the window in which you will perform a ten-minute test session.

Scripted Training

If you haven't already, press the button labeled "Train Me."

You will now be trained on how to operate the robot for the coming session. In this session, you will use a mouse to control the robots. This control method is called scripted control.

This control mode allows you to drop a sequence of goals or markers on the map to tell the robot where to go. These goals can be dropped on the map by simply clicking the mouse on the spot on the graph you want the robot to go to. Each goal will have a number associated with it that tells you the order the robot will traverse the goals.

If you want to move a goal, simply click on it and drag it to the location you want to move it to and then drop it. If you want to delete a goal, simply drag it to a position on the screen that is not on the map of the world.

There is one thing that you must be weary of when using this control mode. Sometimes

obstacles get in the robots path (on its way to the goal markers) which the robot has trouble getting around. You can help the robot to get past these obstacles by dragging and dropping the goal marker labeled "1" to a place that will help the robot get around the obstacles. After you have helped the robot around the obstacle, move the goal marker labeled "1" back to the place you want the robot to move to.

The basic protocol you should follow with this control scheme is to service the current robot (i.e., give it the command or sequence of commands that will get it moving towards its goal by dropping goals in progress on the map). Then press the "Done" button to move to a different task (either performing math calculations or servicing the other robot). When it is time to service a robot again, you will be given control (robot neglect times vary and are determined by the computer). Changes in current task will be indicated by changes in the robot view.

Go ahead and practice moving the robot around its world using this method. Make sure you learn how the goal markers make the robot behave. Remember to also practice neglecting the robot by pushing the "Done" button and moving on to other tasks. When it is time to control the robot again, control will be given back to you. When you are ready to proceed with the next test section, tell me.

Wait for the test subject to indicate they are ready to proceed.

Close the GUI window. A new window now will pop up which is the window in which you will perform a ten-minute test session.

Bibliography

[1] J. A. Adams and R. P. Paul. Human supervisory control of multiple mobile agents. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, pages 3298–3303, Vancouver, British Columbia, Canada, 1995.

[2] K. S. Ali and R. C. Arkin. Multiagent teleautonomous behavioral control. In *Machine Intelligence and Robotic Control*, volume 1(2), pages 3–10, 2000.

[3] R. C. Arkin. Reactive control as a substrate for telerobotic systems. In *IEEE AES Systems Magazine*, volume 6(6), pages 24–31, 1991.

[4] R. C. Arkin. *Behavior-Based Robotics.* MIT Press, 1998.

[5] R. C. Arkin and K. S. Ali. Integration of reactive and telerobotic control in multi-agent robotic systems. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 473–478, Brighton, England, 1994.

[6] L. Bainbridge. Ironies of automation. In *Automatica*, volume 19, pages 775–779, 1983.

[7] T. Balch and R. C. Arkin. Behavior-based formation control for multi-robot teams. In *IEEE Transactions on Robotics and Automation*, volume 14(6), pages 926–939, 1998.

[8] C. Breazeal. A motivational system for regulating human-robot interaction. In *Proceedings of the National Conference on Artificial Intelligence*, pages 54–61, Madison, WI, 1998.

[9] D. J. Bruemmer and M. Walton. Collaborative tools for mixed teams of humans and robots. In *Multi-Robot Systems Workshops*, Washington, D.C., 2003.

[10] H. Burkhard, D. Duhaut, M. Fujita, P. Lima, R. Murphy, and R. Rojas. The road to robocup 2050. In *IEEE Robotics and Autonomation Magazine*, volume 9(2), pages 31–38, 2002.

[11] L. Conway, R. A. Volz, and M. W. Walker. Teleautonomous systems: Projecting and coordinating intelligent action at a distance. In *IEEE Transactions on Robotics and Autonomation*, volume 6(2), pages 146–158, 1990.

[12] J. W. Crandall and M. A. Goodrich. Experiments in adjustable autonomy. In *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1624–1692, Tuscon, AZ, 2001.

[13] J. W. Crandall, C. W. Nielsen, and M. A. Goodrich. Towards predicting robot team performance. In *Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics*, pages 906–911, Washington, D.C., 2003.

[14] G. A. Dorais, R. P. Bonasso, D. Korenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Proceedings of the First International Conference of the Mars Society*, Boulder, CO, 1998.

[15] D. D. Dudenhoeffer and D. J. Bruemmer. Command and control architectures for autonomous micro-robotic forces. In *FY01 Project Report: Idaho National Engineering and Environmental Laboratory*, Idaho Falls, ID, 2001.

[16] M. Egerstedt and X. Hu. Formation constrained multi-agent control. In *IEEE Transactions on Robotics and Automation*, volume 17(6), pages 947–951, 2001.

[17] T. Fong, N. Cabrol, C. Thorpe, and C. Baur. A personal user interface for collaborative human-robot exploration. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Montreal, Canada, 2001.

[18] T. Fong, C. Thorpe, and C. Baur. Collaborative control: a robot-centric model for vehicle teleoperation, 1998.

[19] T. Fong, C. Thorpe, and C. Baur. A safeguarded teleoperation controller. In *Proceedings of the IEEE International Conference on Advanced Robotics*, Budapest, Hungary, 2001.

[20] D. W. Gage. Command control for many-robot systems. In *Proceedings of Nineteenth Annual AUVS Technical Symposium*, Huntsville, AL, 1992.

[21] K. Goldberg, B. Chen, R. Solomon, and S. Bui. Collaborative teleoperation via the internet. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2019–2024, San Francisco, CA, 2000.

[22] M. A. Goodrich and E. R. Boer. Designing human-centered automation: trade-offs in collision avoidance system design. In *IEEE Transactions on Intelligent Transportation Systems 1(1)*, volume 1(1), pages 40–54, 2000.

[23] M. A. Goodrich, J. W. Crandall, and J. R. Stimpson. Neglect tolerant teaming: Issues and dilemmas. In *2003 AAAI Spring Symposium*, Palo Alto, CA, 2003.

[24] M. A. Goodrich, D. R. Olsen Jr., J. W. Crandall, and T. J. Palmer. Experiments in adjustable autonomy. In *Proceedings of the IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, Seattle, WA, 2001.

[25] H. Hexmoor. A model of absolute autonomy and power: Toward group effects. In *2002 AAAI Workshop on Autonomy, Delegation, and Control: From Inter-Agent to Groups*, Edmonton, Canada, 2002.

[26] A. Kheddar, P. Coiffet, T. Kotoku, and K. Tanie. Multi-robots teleoperation-analysis and prognosis. In *6th IEEE International Workshop on Robot and Human Communication*, Piscataway, NJ, 1997.

[27] D. Korenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost. Traded control with autonomous robots as mixed initiative interaction. In *AAAI Spring Symposium on Mixed Initiative Interaction*, Palo Alto, CA, 1997.

[28] D. Korenkamp, E. Huber, and R. P. Bonasso. Recognizing and interpreting gestures on a mobile robot. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 915–921, Portland, OR, 1996.

[29] E. Krotkov, R. Simmons, F. Cozman, and S. Koenig. Safeguarded teleoperation for lunar rovers: from human factors to field trials. In *IEEE Planetary Rover Technology and Systems Workshop*, Minneapolis, MN, 1996.

[30] J. C. Lane, C. R. Carignan, and D. L. Akin. Advanced operator interface design for complex space terobots. In *Autonomous Robots*, volume 11(1), pages 49–58, 2000.

[31] J. R. T. Lawton, R. W. Beard, and B. J. Young. A decentralized approach to formation maneuvers. In *IEEE Transactions on Robotics and Automation*, volume 19(6), pages 933–941, 2003.

[32] P. Lima, T. Balch, M. Fujita, R. Rojas, M. Veloso, and H. A. Yanco. Robocup 2001. In *IEEE Robotics and Automation Magazine*, volume 9(2), pages 21–23, 2002.

[33] R. R. Murphy. *Introduction to AI Robotics.* MIT Press, 2000.

[34] R. R. Murphy and E. Rogers. Cooperative assistance for remote robot supervision. In *Presence*, volume 5(2), pages 224–240, 1996.

[35] O. Nakayama, T. Futami, T. Nakamura, and E. R. Boer. Development of a steering entropy method for evaluating driver workload. In *Presention for the International Congress and Exposition*, Detroit, MI, 1999.

[36] M. N. Nicolescu and M. J. Matarić. Learning and interacting in human-robot domains. In *Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, volume 31(5), pages 419–430, 2001.

[37] D. A. Norman. *The Design of Everyday Things.* Doubleday, 1990.

[38] D. R. Olsen and M. A. Goodrich. Metrics for evaluating human-robot interactions. In *Performance Metrics for Intelligent Systems Workshop*, Washington, D.C., 2003.

[39] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. In *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, volume 30(3), pages 286–297, 2000.

[40] L. E. Parker. Alliance: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 776–783, Munich, Germany, 1994.

[41] H. E. Pashler. *Engineering Psychology and Human Performance.* The MIT Press, 1997.

[42] D. Perzanowski, A. C. Schultz, W. Adams, and E. Marsh. Goal tracking in a natural language interface: Towards achieving adjustable autonomy. In *IEEE International Symposium on Computational Intelligence in Robotics and Autonmation*, Monterey, CA, 1999.

[43] M. E. Pollack, I. Tsamardinos, and J. F. Horty. Adjustable autonomy for a plan management agent. In *AAAI Spring Symposium on Adjustable Autonomy*, Palo Alto, CA, 1999.

[44] D. Pomerleau. Ralph: Rapidly adapting lateral position handler. In *IEEE Symposium on Intelligent Vehicles*, Detroit, MI, 1995.

[45] T. Rofer and A. Lankenau. Ensuring safe obstacle avoidance in a shared-control system. In *Proceedings of the 7th International Conference on Emergent Technologies and Factory Autonation*, pages 1405–1414, Barcelona, Spain, 1999.

[46] P. Scerri, D. Pynadath, and M. Tambe. Adjustable autonomy in real-world multi-agent environments. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 300–307, Montreal, Canada, 2001.

[47] P. Scerri, D. V. Pynadath, and M. Tambe. Why the elf acted autonomously: Towards a theory of adjustable autonomy. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 857–864, Bologna, Italia, 2002.

[48] T. B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control.* MIT Press, 1992.

[49] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 903–910, Palo Alto, CA, 2000.

[50] F. Steele and T. Blackmon. An operator interface for a robot-mounted, 3d camera system: Project pioneer. In *Proceedings of the American Nuclear Society*, pages 126–132, Houston, TX, 1998.

[51] I. H. Suh, H. J Yeo, J. H. Kim, J. S. Ryoo, S. R. Oh, C. W. Lee, and B. H Lee. Design of a supervisory control system for multiple robotic systems. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 332–339, Osaka, Japan, 1996.

[52] R. M. Voyles and P. K. Khosla. Tactile gestures for human/robot interaction. In *Proceedings of IEEE/RSJ Intelligent Robots and Systems Conference*, pages 7–13, Pittsburgh, PA, 1995.

[53] R. M. Voyles and P. K. Khosla. A multi-agent system for programming robots by human demonstration. In *Integrated Computer-Aided Engineering*, volume 8(1), pages 59–67, 2001.

[54] C. D. Wickens and J. G. Hollands. *Engineering Psychology and Human Performance, third edition.* Prentic Hall, 2000.