SLIDING SCALE AUTONOMY AND TRUST IN HUMAN-ROBOT INTERACTION

BY

MUNJAL DESAI

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Signature of Author: _____ Date: 05/10/2007

Signature of Thesis Supervisor: _____
Holly Yanco, Ph.D.

Signatures of Other Thesis Committee Members: _____
Fred Martin, Ph.D.

_____
Jill Drury, Sc.D.

SLIDING SCALE AUTONOMY AND TRUST IN HUMAN-ROBOT INTERACTION

BY

MUNJAL DESAI

ABSTRACT OF A THESIS SUBMITTED TO THE FACULTY OF THE
DEPARTMENT OF COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL
2007

Thesis Supervisor: Holly Yanco, Ph.D.

Assistant Professor, Department of Computer Science

Autonomy defines what robots can do independently: the greater the autonomy, the more robots can do. However, giving more autonomy to robots does not always mean they can perform better. Adjustable autonomy systems solve this problem by providing multiple autonomy levels from which to select. The system designers select the autonomy levels based on what they think would be appropriate for the application. However there are circumstances where a required autonomy level is not provided. We designed a sliding scale autonomy system that provides a continuum of autonomy levels. We included a trust slider to ensure that the robot would take appropriate initiative in accordance with the level of trust that the user has of the robot. Our system was tested against two different adjustable autonomy systems for performance. Overall, users had fewer hits and lower run times with the sliding scale autonomy systems compared to the other two systems.

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor Professor Holly Yanco for her support throughout the process and also for guiding me in the right direction every time I went astray. I would also like to thank Professor Fred Martin and Dr. Jill Drury for not only being on my thesis committee but also for providing valuable insights which would otherwise have been missed. I would also like to thank Mark Micire for reviewing each and every thesis draft and also for providing valuable suggestions. Thank you to Kate Tsui for managing every aspect of the user tests, especially given the very short period of time we had. Thanks to all the members of robotics lab at UML, particularly to Andrew Chanler, Mike Baker, and Brenden Keyes. Thanks to my friends for their support and patience. Finally I would like to thank my parents for their love and support throughout this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Autonomy

Autonomy defines what robots can do independently; the greater the autonomy, the more robots can do. However, giving more autonomy to robots does not always mean they can perform better on their own. In fact, it can even be counterproductive at times (Cummings and Mitchell 2006) (Goodrich, Jr., Crandall, and Palmer 2001). Ideally, the autonomy level should be adjusted based on the environment and robot state.

There are many constraints that govern the amount of autonomy that a robot can possess.

- Computational power: The amount of autonomy is limited by the computational power of the robot platform. This is becoming less of a problem with the reduction in physical size of processors along with the increase in computational capability.

- Sensor arrays: Unless the robot has a good array of sensors, it will not be able to perceive the environment properly and would have to rely on the operator's judgment, thus limiting the autonomy that it can have.

- Sensor processing: Without good sensor modeling it is not possible to generate an accurate model of the environment. Better sensor models have been developed that help provide a more accurate model of the environment.

- Physical environment: Physical environment is very important. The level of autonomy is usually directly proportional to the structure of the environment. For example, robots have been created that can autonomously navigate an office building (Nourbakhsh, Powers, and Birchfield 1995) (Burgard, Cremers, Fox, Hahnel, Lakemeyer, Schulz, Steiner, and Thrun 1998), but the state-of-the-art urban search and rescue (USAR) systems are teleoperated.

- Communication delay: Delays that may be encountered due to communication links requires that the robots have more autonomy.

- Application domain: Robots that are designed to operate in an environment that involves interacting with humans usually have the level of autonomy restricted. For example robotic wheelchairs generally do not have very high levels of autonomy. But an industrial arm are almost always fully autonomous due to lack of close interaction with humans.

Most of the systems in application domains such as urban search and rescue are teleoperated. This is the lowest level of autonomy that a robot can operate. In these systems, operators do all of the cognitive work. Teleoperation is not always bad, especially if the robot is in direct view of the operator, but this is not the case with USAR. However, there are applications where this would not be feasible. For example, unmanned ground vehicles (UGV), unmanned aerial vehicles (UAV), and Mars rovers, all suffer from some of the reasons mentioned above, particularly the communication delay. These applications generally demand the robot be given some autonomy for better performance.

Dynamic autonomy or adjustable autonomy systems provide solutions to these problems. They have autonomy levels ranging between teleoperation and full autonomy. Systems that have two or more autonomy levels within this range are called discrete autonomy systems. The system designers select the intermediate levels based

on what they think would be appropriate for the application/system and let operators switch between the autonomy levels. An example would be the Idaho National Laboratory (INL) robot system that has four autonomy levels: teleoperation, safe, shared, and full autonomy (Bruemmer, Dudenhoeffer, and Marble 2002).

1.2 Trust

Whenever two people work towards a common task, there has to be some level of trust between them. It is the same when it comes to robots and operators. Currently, and for the foreseeable future, there will be an autonomy void that needs to be filled by an operator. The human presence in a robotic system with some autonomy is called human-in-the-loop control. Since the operator and the robot are the two entities working together, it is important to consider their interaction. The robot trusting the operator is implied. The tricky part is to handle how the operator trusts the robot. The operator needs to be assured that if he tells the robot to do something, then the robot will perform with a certain degree of accuracy, or it will notify the operator that it cannot carry out the task. It is not possible to have teamwork without trust.

Once trust has been established, the operator may transfer some cognitive load to the robot and not pay attention to what it is doing (Yanco and Drury 2004). The operator may not know the decision model used by the robot, and hence may not be able to understand the reasons for the decisions taken by the robot.

1.3 Problem Statement

For optimal operation, the level of autonomy needs to reflect the complexity of a robot's operating environment. Irrespective of the autonomy level, robot systems still require inputs from the operator. The presence of the user necessitates that robot systems consider the operator as part of the system, thus requiring the user to

trust the robot. Stress and cognitive limitations of operators demand that the robot system take over as many operations as possible. No existing robot architecture optimally incorporates the human-robot interaction (HRI) elements of sliding scale autonomy and trust building and enhancement.

1.4 Contributions of the Thesis

- Designed an architecture for human - robot interaction that increases the performance of novice users and expert users over existing dynamic autonomy systems.

- Defined levels of trust at which an autonomous system can operate, based upon Sheridan's levels of autonomy.

- Implemented the sliding scale autonomy and trust scale portions of the architecture. The sliding scale autonomy system lets the user and the robot involved change the autonomy level to the desired value, which is something that is not possible with the existing autonomy systems. The trust scale provides a means to increase the user's trust on the robot at a comfortable pace and to a suitable level.

- The sliding scale autonomy system with the trust scale was tested with novice and expert users. We found that there was a reduction in hits and run time for novice and expert users. The system was compared with a discrete autonomy system and a multiple slider system.

CHAPTER 2

REVIEW OF RELEVANT RESEARCH

2.1 Robot Architectures

A principled approach to design is being applied to robot projects in a wide array of domains ranging from underwater robots to unmanned aerial vehicles. This approach is sometimes called as architecting (Bayouth, Nourbakhsh, and Thorpe 1997). Robot architectures are central to any reliable robot system, and hence the study of robot architectures plays an important role in the development of a new generation of autonomous robots that are required to meet real-time constraints and exceed particular safety minima (Bayouth, Nourbakhsh, and Thorpe 1997). Over a period of many years, endeavors by researchers have resulted in a variety of architectures. Typically, architectures have been classified as either hierarchical e.g. (Nilsson 1984), reactive e.g. (Brooks 1986), (Arkin 1987) or hybrid deliberate/reactive e.g. (Dorais, Bonasso, Kortenkamp, Pell, and Schreckenghost 1998).

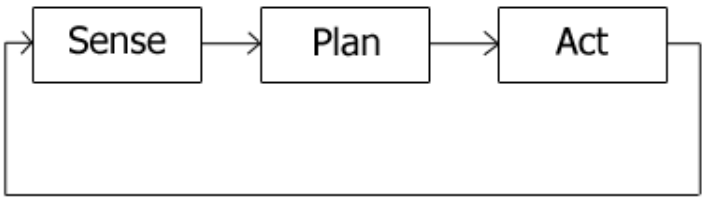

Figure 1. Sense plan act paradigm used by hierarchical architectures.

The hierarchical architecture was one of the first types of robot architectures. These architectures were based on the paradigm of sense-plan-act as shown in Figure 1. The robot system would read the sensor values, decide its course of action and execute it. These systems were slow because every time the environment changed, the model of the environment had to be recomputed and a new plan would be generated.

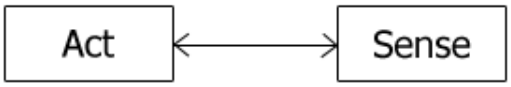

Figure 2. Plan act paradigm used by reactive architectures.

The subsumption architecture is a classic example of a reactive architecture (Brooks 1986). The subsumption architecture was designed to overcome the problems faced by the hierarchical architectures. These types of systems do not have a planning layer. The robot simply senses the environment and acts based on the sensed information as shown in Figure 2. This allowed the systems to react to changing environment in real time.

The subsumption architecture decomposes a behavior into many less complex layers. Each layer can subsume or override the underlying layer. Two main disadvantages of this model are the inability to modularize the system and the rather low flexibility at runtime.

ATLANTIS was an attempt to combine reactive and deliberative types of architectures in such a way that they compliment each other (Gat 1996). Gat did this by having the deliberative operations of the architecture running asynchronously with the rest of the system. This allowed the system to perform deliberation on a problem, yet respond to something in the environment demanding immediate attention. Figure

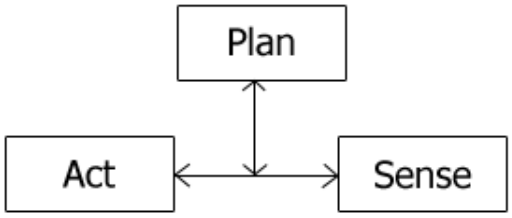

Figure 3. Sense plan act paradigm used by hybrid architectures.

3 shows the paradigm used by hybrid architectures. The 3T robot architecture is another example of hybrid architecture (Gat 1997).

The 3T architecture has three layers, each with its own set of functions. The deliberative layer does planning and problem solving; the execution layer translates goals into task networks and executes them; the sensory-motor skills interact with the world (Dorais, Bonasso, Kortenkamp, Pell, and Schreckenghost 1998).

2.2 Dynamic Autonomy

Some of the first fully autonomous robots were Grey Walter's tortoise robots, which were fairly simple reactive robots (Walter 1961). Since then, much has been done in the field of robot autonomy. One of the most influential works has been done by Sheridan (Sheridan, Parasuraman, and Wickens 2000) where they define 10 autonomy levels for generic autonomous systems. These ten generic autonomy levels could be applied to any robotic system. There have also been some attempts to create taxonomies for autonomy in multi-agent and single-agent systems (Dudek, Jenkin, and Wilkes 1993), (Huang, Albus, Messina, and Wade 2004) .

Some robot architectures have dynamic autonomy and support reactive and deliberative layers. However, there exists no such system that is fully autonomous in the true sense. Some groups specializing in the domain of USAR have done significant

Table 1. Sheridan's levels of autonomy, from (Sheridan, Parasuraman, and Wickens 2000).

Level	Description
High 10	The computer decides everything, acts autonomously, ignoring the human
9	Informs the human only if it, the computer, decides to
8	Informs the human only if asked
7	Executes automatically, then necessarily informs the human
6	Allows the human a restricted time to veto before automatic execution
5	Executes that suggestion if the human approves
4	Suggests one alternative
3	Narrows the selection down to a few
2	The computer offers a complete set of decision/action alternatives
Low 1	The computer offers no assistance: human must take all decisions and actions

amounts of work in the field in dynamic autonomy, including INL (Bruemmer, Dudenhoeffer, and Marble 2002) and our lab at UMass Lowell (Desai and Yanco 2005). These systems allow the users to switch between the available autonomy modes but to do not let them operate at a level between them.

In the INL architecture, each reactive behavior runs independently and can have a range of reactive and deliberative capabilities that operate in parallel. INL had also incorporated deliberative behaviors, which exploit a world model and function at a level above the reactive behaviors. INL planned to have reactive behaviors, which once satisfied would let the deliberative behaviors take control. The INL architecture consists of four discrete autonomy modes (Bruemmer, Dudenhoeffer, and Marble 2002):

- Teleoperation: In this mode, the user controls the robot directly without any interference from robot autonomy. In this mode, it is possible to drive the robot into obstacles.

- Safe: In this mode, the user still directly controls the robot, but the robot detects obstacles and prevents the user from bumping into them.

- Shared: In this mode, the robot drives itself while avoiding obstacles. The user, however, can influence or decide the robots travel direction through steering commands.

- Autonomous: The robot is given a goal point to which it then safely navigates.

Further research in the field of dynamic autonomy based on the above systems resulted in continuous dynamic autonomy, also called by some researchers as sliding scale autonomy (Desai and Yanco 2005). In (Desai and Yanco 2005), we explain a method to convert a discrete autonomy system into a sliding scale autonomy.

We define sliding scale autonomy as the ability to create new levels of autonomy between existing, preprogrammed autonomy levels. However, the term sliding scale autonomy does not have a fixed definition in the robotics domain. Some researchers interpret discrete autonomy systems as sliding scale autonomy systems. Sliding scale autonomy should not be confused with sliding autonomy which is frequently used for systems which have adjustable autonomy (Brookshire, Singh, and Simmons 2004), (Sellner, Simmons, and Singh 2005), (Heger, Hiatt, Sellner, Simmons, and Singh 2005), and (Bruemmer, Dudenhoeffer, and Marble 2002).

2.3 Trust

A panel at CHI-04 (Bruemmer, Few, Goodrich, Norman, Sarkar, Scholtz, Smart, Swinson, and Yanco 2004) indicated that a system must be designed such that the possibility of counterproductive interaction because of mixed-initiative autonomy must be reduced. Goodrich has put another theory forth: he states that too much trust leads to the user ignoring the system and hence poor performance (Olsen and Goodrich 2003). However, too little trust leads to over-monitoring, which may lead to an inability to perform a secondary task (Goodrich, Jr., Crandall, and Palmer 2001). He

also suggests that training should be used as a means to build trust and believes that trust can be negotiated through a task.

"System trust can only be enhanced when the system is designed to meet the actual user's needs, abilities, and limitations within the constraints of the task" (Marble, Few, and Bruemmer 2004). This work also provides details about testing such systems. Some of the important guidelines state that the system must be tested in an environment that reflects the complexities of the real world and must incorporate uncertainties. The researchers also mention that the operator's ability to trust the robot as part of a team must be evaluated.

Most research in this domain has been restricted to cases where the robot is not allowed to make changes to the autonomy level. Since there are very few systems that let the robot change the level of autonomy, that aspect of trust has been largely unexplored.

CHAPTER 3

APPROACH

3.1 Introduction

Currently, robot systems can have multiple autonomy levels (e.g., teleoperation, safe teleoperation where sensors are used to stop the robot as obstacles approach, shared control where the robot follows a corridor or wanders while the operator can use a joystick to influence the direction of motion, and fully autonomous control). The robot must operate in one of these modes; there is no notion of giving a little more control to the robot or a little more control to the user. By blending human and robot inputs, we can create autonomy levels between the few pre-programmed levels.

However, simply creating additional autonomy modes is not sufficient. In user testing, people select an autonomy level they are comfortable with, then stick with it throughout the tests, even when another autonomy level would improve task performance. We investigated methods for changing autonomy levels, both when giving the user more control and when giving the robot more control.

I implemented a sliding scale autonomy architecture which evaluates the robot's task performance in real time and determines how autonomy should change (block 1 in Figure 4).This decision is presented on a slider control, which the user can modify (block 2 in Figure 4). The user's trust of the system is entered in a trust slider (block 3 in Figure 4). If the user trusts the system, autonomy changes are made automatically. If the user does not trust the system, then suggestions for autonomy level are presented.

Figure 4 shows the robot architecture of the system that has been implemented. At its core are the following three subsystems: condition extraction system, system variable agents and arbitration system.

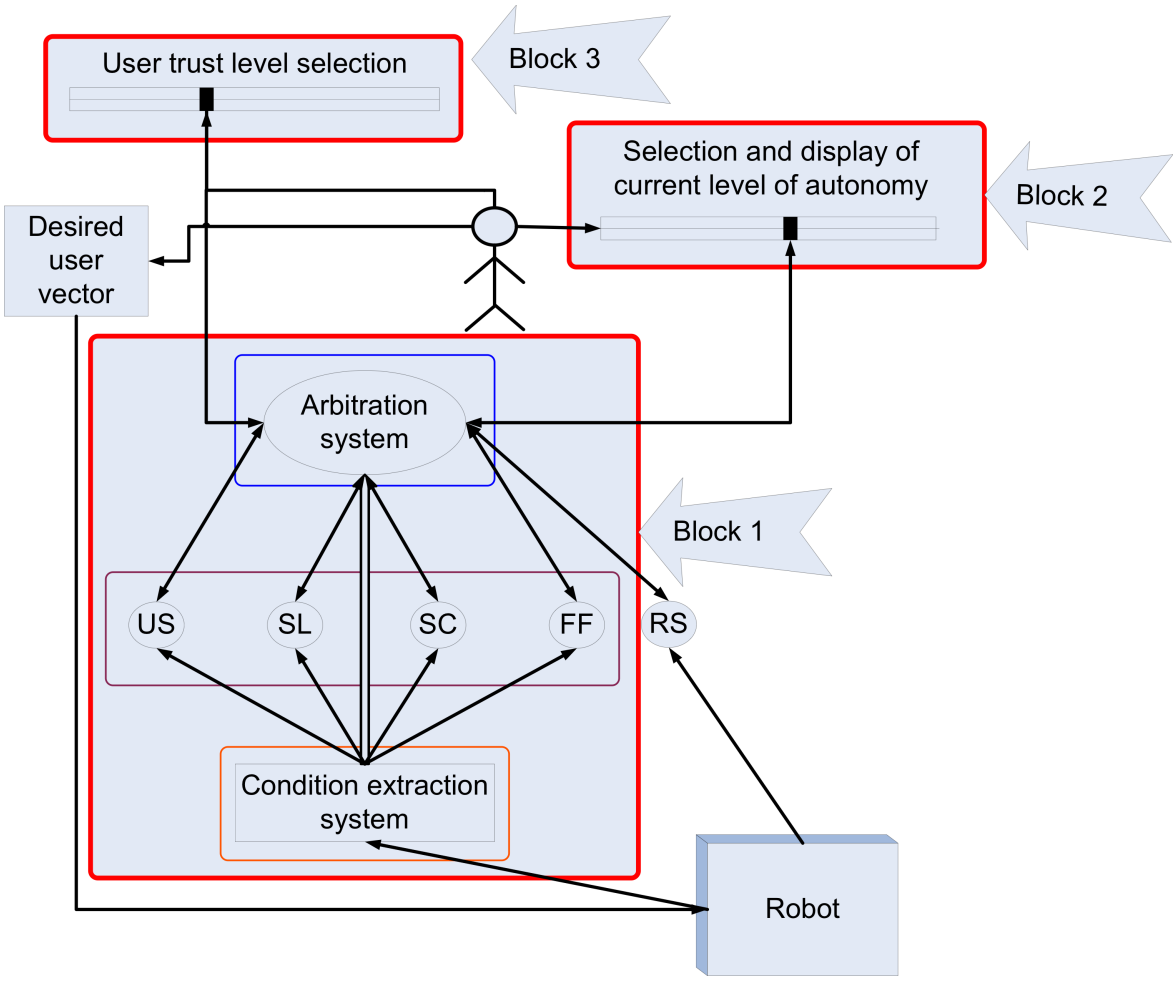

Figure 4. Implemented architecture for human-robot interaction.

The condition extraction system (CES) collects information from the robot's sensors and looks for certain conditions that might be true. It then informs the arbitration system and the system variable agents about these conditions.

Currently the system has four system variables. System variables are the characteristics that define autonomy. Each system variable has a system variable agent (SV-agent) that decides what values for the system variable is appropriate, based on the conditions received from the condition extraction system.

The changes recommended by the system variable agents are arbitrated by the arbitration system (AS). The arbitration system does this in accordance with the trust that the operator has of the robot. The operator expresses his trust of the robot using a trust scale.

3.2 System Overview

Figure 5 shows the overall working of the system. The whole system can be viewed as a set of three processes that run sequentially. Even though the processes run sequentially, the three sections have their own threads. This makes modifying the system easy, especially adding more functionality to any section.

The condition extraction system (CES) first reads the information from the various sources and processes it. It updates the status of all the conditions after processing the information. CES then informs all the system variable agents (SV-Agents) that the conditions have been updated and waits for a signal from the SV-Agents indicating that one cycle has been completed. The cycle as shown in Figure 5 starts with CES updating all the conditions, the SV-Agents generating the suggestions, and the arbitration system processing the suggestions.

Once CES signals all the SV-Agents, the SV-Agents generate the suggestion vector based on the value of each condition. The SV-Agent simply waits for the next signal from CES after it finishes generating the suggestion vector. Before going to sleep, the last SV-Agent signals the arbitration system (AS) indicating that all the suggestion vectors have been updated. The SV-Agent is not explicitly aware of the number of the other SV-Agents, making the whole process of multi-threading easy and convenient.

The AS reads and processes the suggestion vectors. Since the AS is aware of the various system variables and their SV-Agents, it arbitrates between the various suggestions. A function is associated with each condition. These functions check

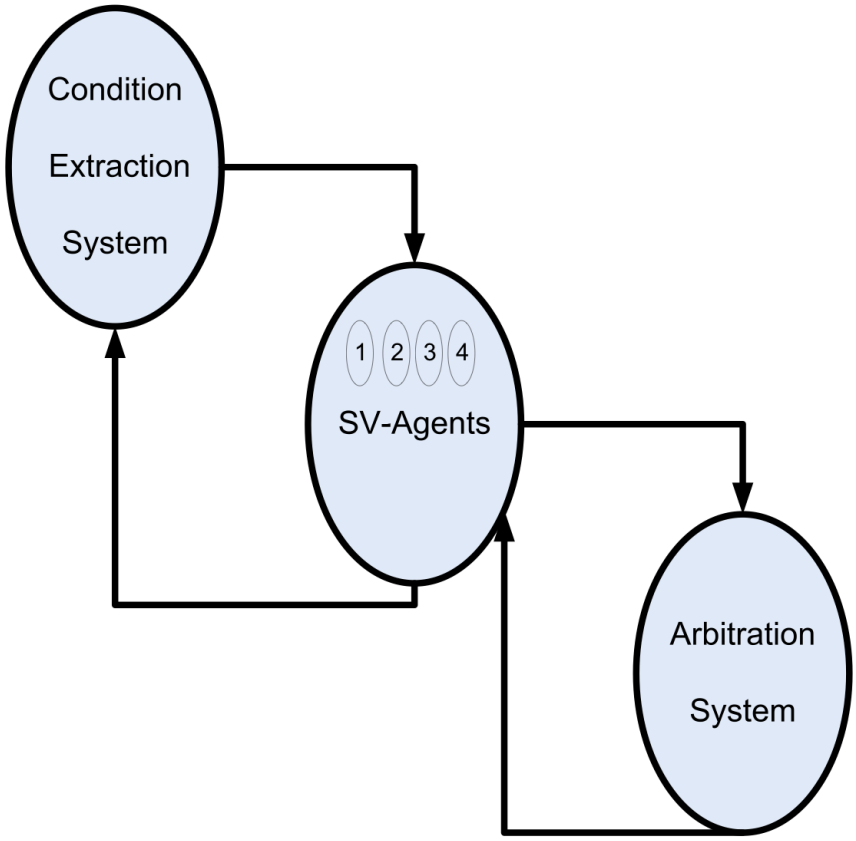

Figure 5. The figure shows the sequence of operations. The condition extraction system (CES) generates the conditions and then informs the system variable agents (SV-Agents). Once all the SV-Agents finish generating the suggestion vectors the arbitration system (AS) arbitrates between them and decides on the final system variable values. After this is done the cycle restarts.

the suggested values by the different SV-Agents and make corrections if necessary. Once all the functions have been executed the system variable values are calculated. Each condition or each SV-Agent has an associated weight pre-defined in the AS. The weight is a scalar value between 0 and 1. Using these weights the system variable values are calculated. Then the system variable values are updated. The AS signals the SV-Agents and waits for the next arbitration cycle once it is finished. The SV-Agents then signal the CES to begin the next cycle.

3.3 Condition Extraction System (CES)

The Condition Extraction System (CES) collects information from the robot and converts it into conditions that can later be used by other subsystems as shown in Table 2. It reads the information from the input device, sensor information from the robot, the system variable values, etc.

Table 2. List of conditions monitored.

Condition	Description
1	Open space
2	Open space relative
3	Middle space
4	Cluttered space
5	Expert user
6	Inept driver
7	Driving at maximum speed
8	Driving at minimum speed
9	Bumping into objects
10	Force field blocking

Once the information required to generate conditions is read it is processed based on predefined criteria. Each criterion has three parts: the result presentation, the sampling time and the input processing method. The result can be a binary value indicating that the threshold was met, the difference in the current and last value or the percent difference in the current and the last value. The sampling time specifies if the data processing will be in real time or batch processed after the specified time. The input processing methods indicate if the values need to be averaged over time or the difference in the values need to be averaged over time.

3.4 System variables (SV)

System variables are the characteristics that define autonomy levels. By changing the values of the system variables, it is possible to transition from one autonomy level to another. The system has five system variables: user speed, speed limiter,

speed contribution, force field and robot speed. These system variables were found after examining a discrete autonomy system developed by (Baker, Casey, Keyes, and Yanco 2004) for urban search and rescue.

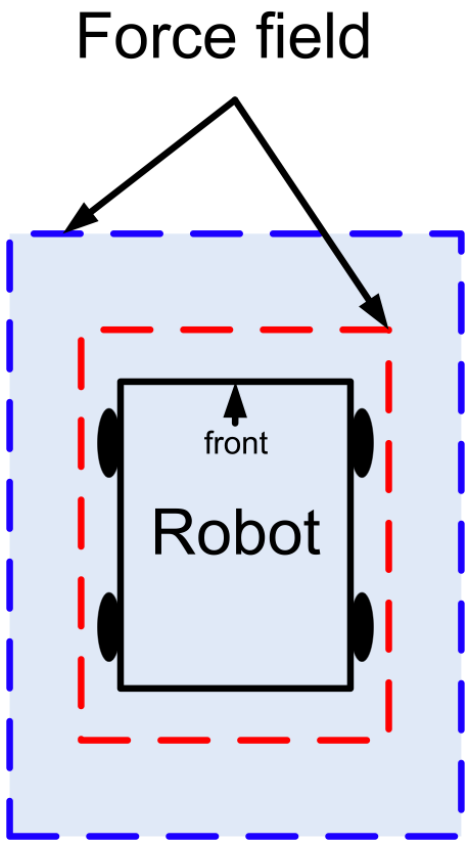

Figure 6. Different force field settings. The inner force field is appropriate when the robot is moving more slowly; the outer field would be used when the robot is traveling faster.

3.4.1 Force field (FF)

The force field system variable describes the minimum safety distance that must be maintained around the robot at all times. There are four force fields, one in each compass direction as shown in Figure 6. Whenever any object comes in contact with a force field, the movement of the robot in that particular direction is not allowed; however, the robot can still move in other directions. The values for force fields

typically range between 0 - 4 robot lengths. A robot length is equal to length of the robot, which is a means to abstract away the size of the particular robot being used.

3.4.2 User speed (US)

The user speed system variable defines a scalar on the speed at which the user can drive the robot. The scalar value in effect limits the maximum speed at which the user can drive the robot. This value is a float ranging from 0 to 1 and this value acts as a multiplier to the input provided by the user. The operator is not aware of this variable and hence cannot change this value.

3.4.3 Robot speed (RS)

The robot speed system variable defines the speed scalar. The robot speed is determined by the robot behavior. The robot's behavior also determines its travel vector. This value is a float ranging from 0 to 1.

3.4.4 Speed contribution (SC)

The SC can be used to switch from the user having full control of the final speed to the robot having full control of the speed or any point in between, without changing the user speed and robot speed values. Figure 7 shows some speed profiles that result from varying the speed contribution value.

3.4.5 Speed limiter (SL)

Because of inertia and traction, robots do not always stop immediately when their force field touches an object; this is especially true when they are going at high speeds. The speed limiter controls the operator's contribution to speed by deciding when to start slowing down the robot and at what rate. The value ranges between 0 - 1. For example, when the robot is traveling in a narrow hallway with the user speed

set to 1 and a speed limiter value greater than 0, if the user commands the robot go forward at full speed, the speed limiter will slow down the robot based upon the current distance to the hallway walls and the force field that has been set. Similarly, if there is an obstacle in the path of the robot, the robot will start to slow down at a rate dependent on the value of the speed limiter. The robot will then come to a stop when the force field comes in contact with the object. A fairly robust linear equation is used to determine when to slow the robot as it approaches objects. The equation is:

$$NewTranslateValue = \frac{(CurrentTranslateValue) * ((RangeInFront) - (ForceField))}{(SpeedLimiter) * 2.5} \tag{1}$$

While performing this calculation it takes into account the amount of force field. If the force field is set to zero then the speed limiter value has no effect on the robot's speed. Changing the speed limiter value essentially changes the slope of the line plotted by the equation.

3.5 Slider representation

A solution to some of the problems posed by discrete autonomy is provided by sliding scale autonomy (SSA). SSA is also the next logical step from discrete autonomy systems. SSA provides the operator and the autonomous agent with the capability to generate the desired autonomy level on the fly, in effect allowing them to have a little more or a little less autonomy. Each point on the autonomy scale represents a particular autonomy level.

We have conducted earlier tests to find a mapping between the various system variables to a single scalar value based on perceived autonomy by expert users. The results from previous tests showed that people perceive autonomy in different ways. More information about those tests is provided in the Appendix. The speed contri-

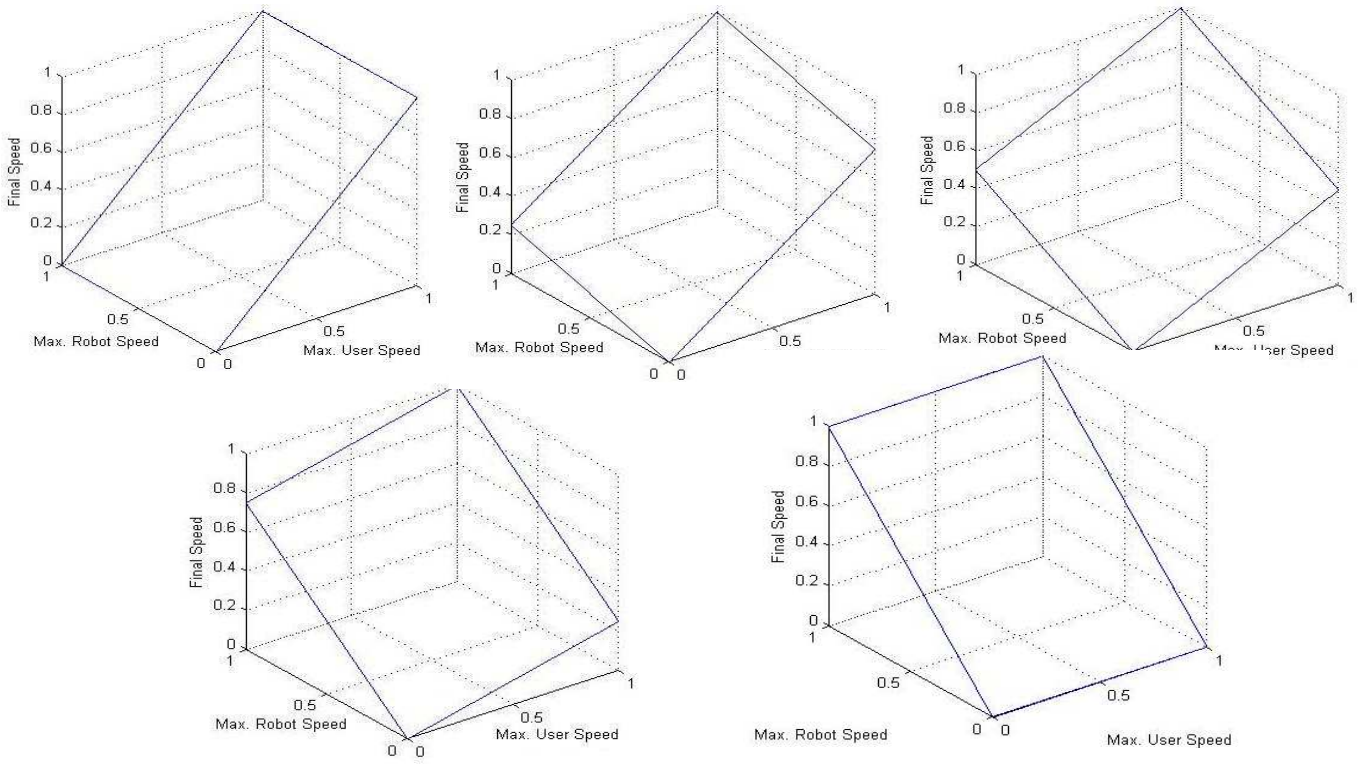

Figure 7. Figures show the varying speed profiles based upon the percentage of speed contribution for the user and the robot. The robot speed and the user speed are combined using the speed contribution to determine the final speed of the robot. The top left speed profile shows the user having full control over the speed, top center shows the user having 75% control, top right shows user having 50% control, bottom left shows user having 25% control and bottom right shows user having no control over speed.

bution system variable had the highest correlation to the perceived autonomy level of all the possible combinations of system variables.

As shown in Figure 8, we made several attempts to develop ways in which the autonomy level would encompass all the system variables while not binding the system variable values to certain ranges. Different methods were proposed, however all of them fell short of providing complete range of freedom to the system variables. Most of those methods revolved around assigning each system variable a fixed amount

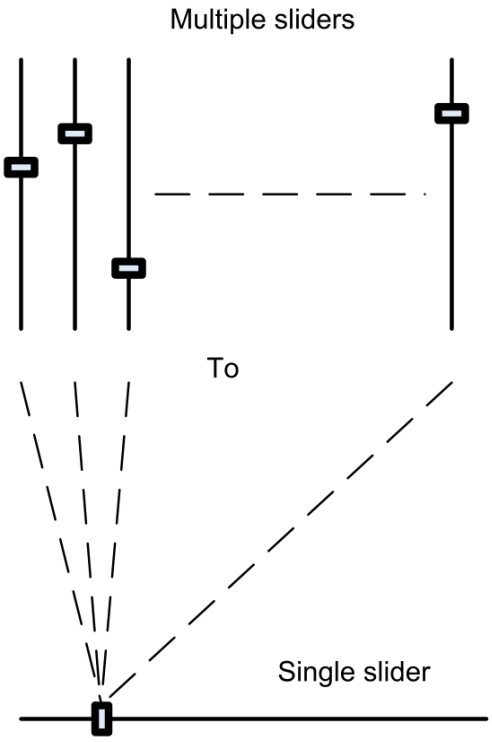

Figure 8. We tried to find a mapping from multiple sliders to a single slider without compromising the range of freedom of the system variables.

of contribution to the autonomy level. Since speed contribution had the highest correlation to the autonomy level, it was assigned the maximum amount of contribution. Every time a system variable changed its value, it would affect the autonomy level. When the robot did not take initiative, the system variable values could only change proportional to the change in the autonomy level. Binding system variables to the autonomy level restricted the possible range of values for the system variables. From the user's perspective, changing the autonomy level might not have the intended effect. The best solution was to let the autonomy level be directly mapped to the speed contribution system variable and it also seemed to be a better solution as it is nearly impossible for the users to indicate the desired change in a particular variable through the autonomy slider and vice versa. From the users' responses during the test it was clear that the users were able to understand what the autonomy slider was meant for (see section 5 for a discussion of user tests).

3.6 System variable agents (SV-agents)

There is a system variable agent for each system variable. The system variable agent (SV-Agent) determines the optimum value for that system variable given the current state of conditions. Each SV-agent has a pre-defined list of conditions that it processes. For example, the SV-agent for user speed may subscribe to the condition "open space." Every time the robot enters open space, the CES will update the value of the open space condition and be read by the SV-Agent for user speed. Based on the values of the conditions, the SV-agent decides if the current value of the system variable is appropriate or it needs to be modified. For each condition, there exists a function that processes the condition and suggests a scalar value. The collection of these scalar values forms the suggestion vector. While making the decisions, a function does not take into account what other functions may be doing. This feature makes adding and removing system variables or modifying the list of conditions it processes easy. It then updates the suggestion vector which read by the arbitration system.

3.7 Arbitration system (AS)

The arbitration system is the most important part of the system. It performs the following functions:

- It receives requests from the SV-agents to change system variables of which they are in charge. Based on the requests that it gets from the SV-agents and the current state of the system, it arbitrates, fine tunes, or even rejects the requests and informs the SV-agents about it.

- It has to adhere to the trust setting while performing the above mentioned functions.

- If changes approved for the system variable demand a change in the autonomy level, then the arbitration system must indicate that on the SSA scale.

Instead of sending a final computed value, the SV-agents send an array where they clearly indicate what the value of corresponding system variable should be for each condition. Passing all values makes it easy for the AS to fine tune as well as understand the recommendations. Additionally, this method allows arbitration to be done condition wise as shown in Table 3.

To make the process of arbitration easy to understand consider the following scenario. Assume that the robot is in a big room and the user has been pushing the joystick forward for some time. This triggers the "Open space," "Expert user," and "Driving at maximum speed" conditions. The "Open space" condition is active because the robot is in an open area above a certain size. Since the user has been pushing the joystick forward for sometime it also means that the joystick has been steady. Whenever the joystick is held without major movements, the "Expert user" condition is triggered. The "Driving at maximum speed" condition has been triggered because the user is trying to go forward at maximum possible speed. It is assumed that the user is trying to go forward based on the fact that the joystick has been pushed forward as much as possible.

Table 3. Sample inputs received by AS.

	Conditions										
SV-Agent	1	2	3	4	5	6	7	8	9	10	Values after arbitration
US	0.75	#	#	#	0.618	#	0.618	#	#	-	0.662
SC	0.6	#	#	#	0.42	#	-	-	-	-	0.51
SL	0.6	#	#	#	-	-	-	-	#	#	0.6
FF	0.8	#	#	#	-	-	-	-	#	#	0.8

Table 3 shows sample vectors generated by the four SV-Agents in response to the different conditions. The conditions numbered from 1 to 10 in Table 3 represent conditions listed in Table 2 in the same order. A "-" entry under a condition

indicates that the corresponding SV-Agent does not subscribe to that condition. Not all conditions might be active at the same time. The conditions that are not active are marked as "#" in Table 3. Conditions that are not active are ignored by the SV-Agents while producing the suggestion vectors. In the current example only conditions "Open space" (1), "Expert user" (5), and "Driving at maximum speed" (7) are active.

In response to the condition "Open space" the US SV-Agent sets the value of user speed (US) to 0.75. For the conditions "Expert user" and "Driving at maximum speed," the SV-Agent wants to increase the current value of US system variable by 3% each. Based on a previous value 0.6 for US the SV-Agent calculates 0.618.

In response to the condition "Open space," the speed contribution (SC) SV-Agent selects a predefined value of 0.6. For the condition "Expert user," the SV-Agent wants to increase the current value of SC system variable by 5%. Based on a previous value 0.4 for SC, the SV-Agent calculates 0.42. In response to the condition "Open space," the SV-Agents for SL and FF select a pre-defined value of 0.6 and 0.8 respectively.

The AS then looks at all the suggestion vectors each active condition at a time. For the condition, "Open space," the suggestions by the SV-Agents are found to be within limits and are all accepted. The other suggestions by SV-Agents for "Expert user" and "Driving at maximum speed" are also found to be within limits and accepted. The final values are computed by taking the average of suggested values. The averaged values are shown in the last column of Table 3. The availability of suggestion vectors makes it easy to arbitrate. If instead of the suggestion vectors, only final values were sent to the AS as shown in Table 4, then the AS would not know how the SV-Agents decided upon those values, making arbitration difficult.

Table 4. Arbitration process can be very difficult without knowing the reasons for the suggested values.

SV-Agent	Suggested values
US	0.662
SC	0.51
SL	0.6
FF	0.8

3.8 Sheridan's levels of "Trust"

Table 5 shows the ten autonomy levels defined by Sheridan that could be applied to any autonomous system (Sheridan, Parasuraman, and Wickens 2000). These levels are permutations of the automation of decision and action selection. To create a system that operates in accordance with the trust that the user has of the robot system, we converted Sheridan's levels of autonomy into levels of trust. Converting a decision-action model to a trust-action model provides a good scale for trust. Table 6 shows is a list of the levels of trust at which the system might operate. One of the differences between our levels of trust based on Sheridan's levels of autonomy and Sheridan's autonomy levels is that autonomy levels are operator centric, but the trust levels are more robot centric.

Table 5. Sheridan's levels of autonomy, from (Sheridan, Parasuraman, and Wickens 2000).

Level	Description
High 10	The computer decides everything, acts autonomously, ignores the human
9	Informs the human only if it, the computer, decides to
8	Informs the human only if asked
7	Executes automatically, then necessarily informs the human
6	Allows the human a restricted time to veto before automatic execution
5	Executes that suggestion if the human approves
4	Suggests one alternative
3	Narrows the selection down to a few
2	The computer offers a complete set of decision/action alternatives
Low 1	The computer offers no assistance: human must take all decisions and actions

Table 6. Our levels of trust, based upon Sheridan's levels of autonomy.

Level	Description
Max Trust 1	The user lets the robot do whatever it wants
2	User lets the robot execute automatically and inform the user only if the robot feels appropriate
3	User lets the robot execute automatically and the robot only responds when asked
4	User lets the robot execute automatically, then requires it to inform the user
5	User lets the robot execute automatically if it is not vetoed within some time
6	User lets the robot execute the suggestion if approved
7	User asks for only one alternative
8	The user asks for a narrowed down version of decision/action alternatives
9	The user asks the robot for a complete set of decision/action alternatives
Min trust 10	The user makes all decisions and actions

The system that we desgned implements the two cases on either end of the trust scale. The low trust system is the one in which the user has complete control over the autonomy levels. The robot can only process and suggest the autonomy level to the user. In the other extreme the user completely trusts the robot and lets the robot change the autonomy level. The implementation of additional trust levels is left for future work.

The low trust system provides the user with a chance to get the feel of the robot at his or her pace. The user can try what the robot wants to do while having full control over the autonomy level. Once the user understands the robot's behavior under different circumstances, it is easier to trust the robot and so they would be more comfortable with switching to a higher trust mode.

CHAPTER 4

METHODOLOGY

4.1 Robot Hardware

Figure 9. Pioneer robot used for testing.

The Pioneer 3 DX robot from ActivMedia Robotics (now know as MobileRobots, Inc) shown in Figure 9 was used. It has an onboard Pentium III CPU and RedHat 7 operating system. It is capable of wireless communication using the 802.11b protocol. It has 16 sonar sensors around it, 8 in front and 8 in back. It also has a SICK "Lms200" laser range finder. It has two wheels on the sides and a caster behind,

allowing it to turn in place. The overall geometry of the robot is that of a rectangle due to the added gripper. This increases the chances of the robot hitting a wall when turning close to it. It also has Canon VC-C4 pan-tilt-zoom camera, which was not used for CES, but was used for user testing.

4.2 Robot Software

The software to control the robot was written in C++ using the C++ client libraries provided by Player 1.6.4. The three interfaces were developed using Java 1.4. Java media framework (JMF) was used to transmit video in real time from the robot to the interface.

4.3 Test environment

A total of three arenas through which the users would drive the robot were set up. The arenas consisted of sharp to moderate turns and the width ranged between narrow to moderate. In the narrow parts of the arena, the robot had maximum of two to three inches on either side. This forced the users to drive very carefully to avoid hitting the walls. The moderate width had up to a foot and a half on each side of the robot. The arenas were designed such that no combination of width and curve would persist for more then a short distance. These variations required the user to change modes frequently.

There were three different autonomy systems: the discrete autonomy system shown in Figure 13, the multiple slider system shown in Figure 14, and the single slider system shown in Figure 15. The single slider system could be operated in low and high trust modes, where it was assumed that the human trusted the robot less and more, respectively. The underlying autonomy system in the high trust and the low trust systems were the same so they were grouped as a single system. As there

Figure 10. Map A.

were three systems that were to be tested, three different arenas were set up. The courses were designed to be of the same length and difficulty.

4.4 Experiment participants

A total of 18 subjects participated in the tests. There were 12 males within the age range of 18 to 40 and the remaining 6 were women aged between 18 and 42. Of the 18 subjects, 12 were novice users and the remaining 6 were expert users, 5 male and 1 female. The age range for novice users was 18 to 42 and the age range for expert users was 24 to 26 . Subjects who understood the concept of robot autonomy as opposed to simply knowing how to drive were considered expert users.

Figure 11. Map B.

4.5 Experimental design and procedure

We conducted a within subjects study. In order to avoid any learning effect, the sequence of maps and interfaces was randomized. The arenas were labeled as A, B and C and the interfaces were labeled as '1' - discrete autonomy system, '2' - multiple slider system, '3.1' - low trust single slider system, and '3.2' - high trust single slider system. The ordering is shown in the Table 7.

Each combination of interface and arena was run six times. The first twelve runs were with novice users and the last six runs were with expert users to make sure that within each group the interface - arena combinations were properly distributed. In the end each interface - arena combination was run four times for novice users

Figure 12. Map C.

and two times for expert users. The runs for interface 3.1 and 3.2 were alternated to provide even coverage.

After signing the Informed Consent Form, the users were asked to fill a pre-test questionnaire shown in Appendix A. Then we explained the robot system and the task to be performed. They were informed that the most important criteria were safety and time, i.e.: to drive as fast as possible without hitting anything. Then they were introduced to the first interface. They were allowed to drive the robot in a test arena that was set up until they became comfortable with the system. The training arena was in a different room from the user so that the user could not see the robot. After the training run, the actual run began. The users were informed that there was one camera recording the interface and the second camera that would record their interaction with the joystick. They were also asked to think out loud, so that their

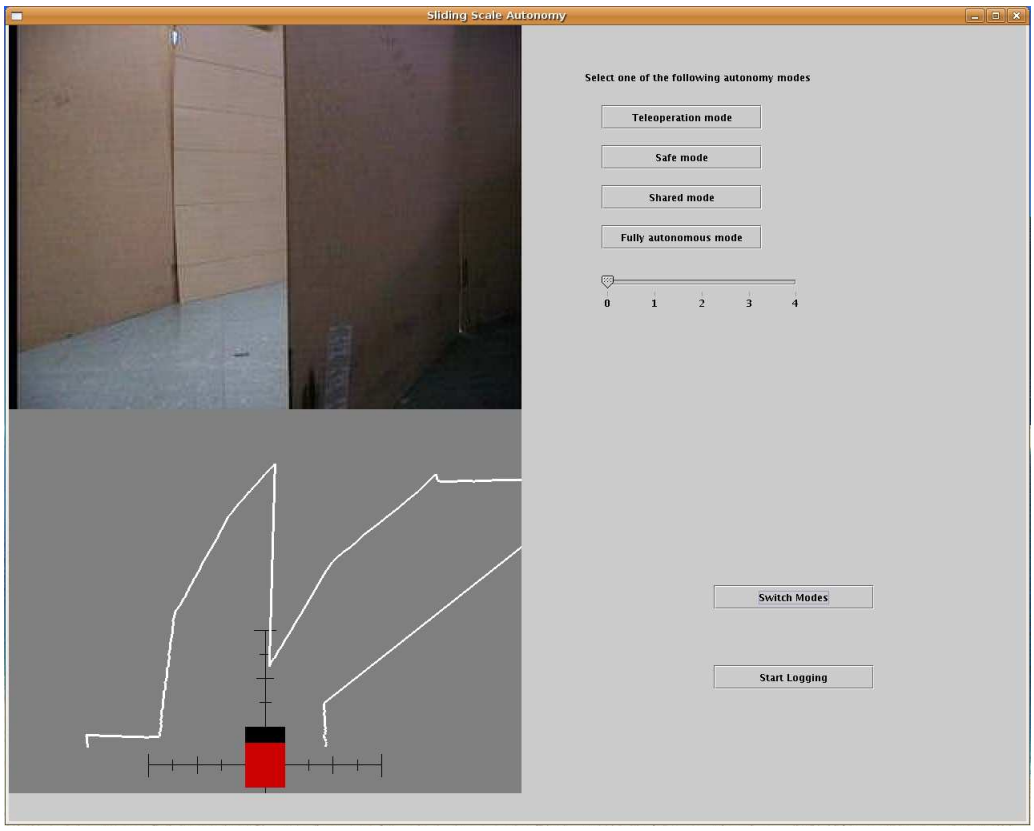

Figure 13. Interface for discrete autonomy system.

comments could be recorded for later analysis. Once all the runs concluded, the users were asked to fill out a post-test questionnaire. The same person interacted with all the users.

There were two observers who monitored the robot and recorded all the hits along with other information such as starting and ending times. This information was being recorded in the critical events sheet that allowed the observer to quickly note down information. The second observer videotaped the robot's progress through the arena.

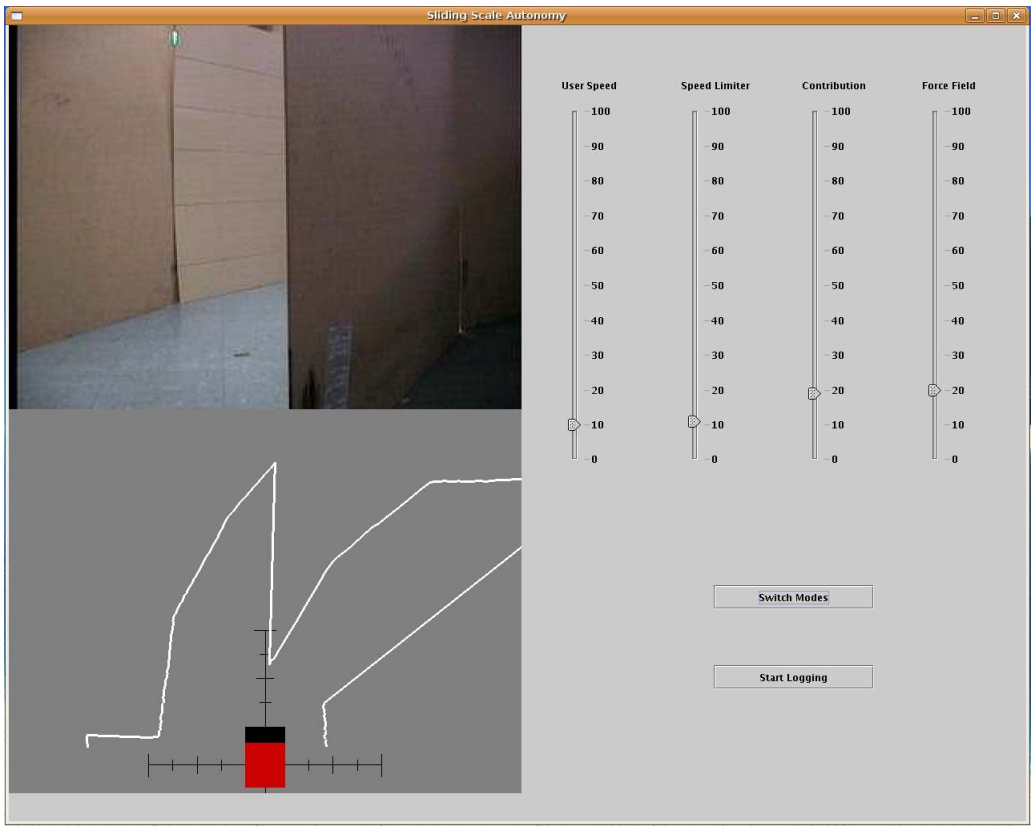

Figure 14. Interface for multiple slider system.

4.6 Interfaces

There were three interfaces used: one for discrete autonomy system, one for multiple slider system, and one for single slider system. These interfaces are described in the following sections in more details.

4.6.1 Discrete autonomy system (DAS)

The discrete autonomy system is an adjustable autonomy system with four autonomy levels as shown in Figure 13 and Figure 16. The autonomy levels are listed below in the increasing order of autonomy.

- Teleoperation mode: In this mode the user has complete control of the robot. There is no force field or speed limiter and the robot's inputs are suppressed. This makes it possible for the user to hit objects while driving the robot.

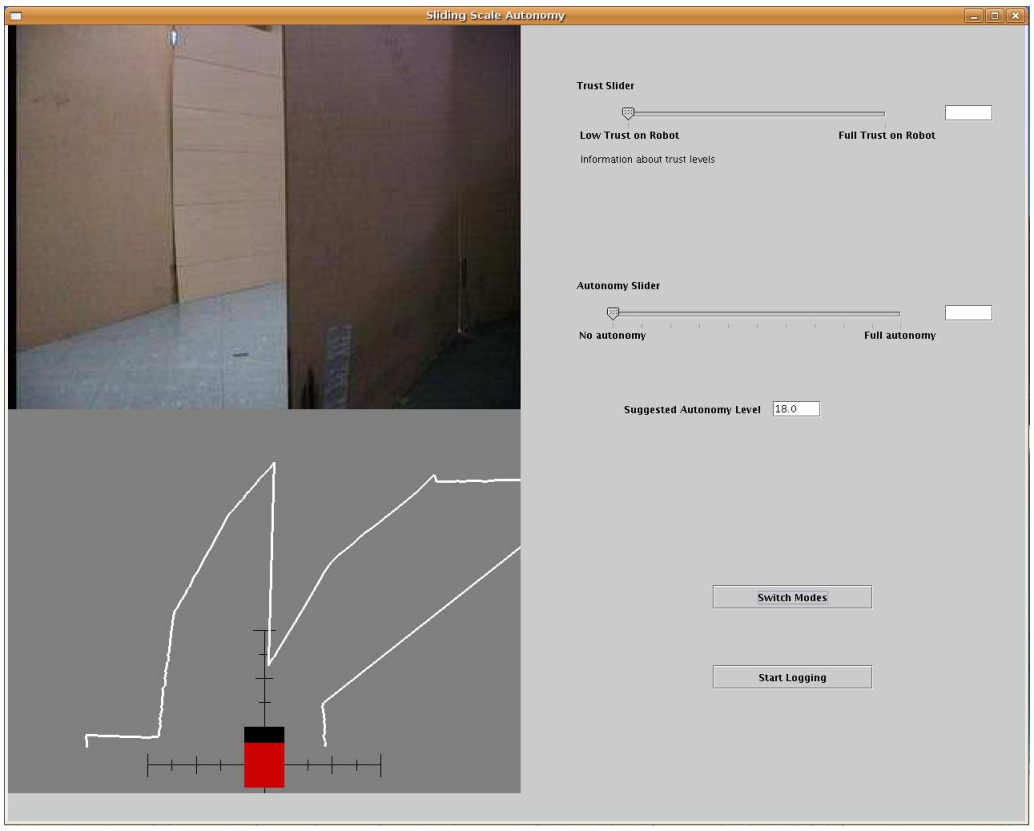

Figure 15. Interface for single slider systems.

- Safe mode: In this mode the force field and the speed limiter are set to a fixed value. This prevents the robot from hitting objects. However, it also prevents the robot from navigating in narrow spaces. The robot's inputs are ignored.

- Shared mode: In this mode the speed contribution, force field, and speed limiter are each set to fixed values. The speed contribution blends the user's input with the robot's input.

- Full autonomy mode: In this mode the user has no control of the robot. The force field and speed limiter are each set to specific values. The speed contribution is set to completely ignore user's inputs.

For each of the following modes the user can control the user speed through a slider that has five speed levels. These modes are similar to the autonomy modes used in the USAR robot systems developed by the (Baker, Casey, Keyes, and Yanco 2004).

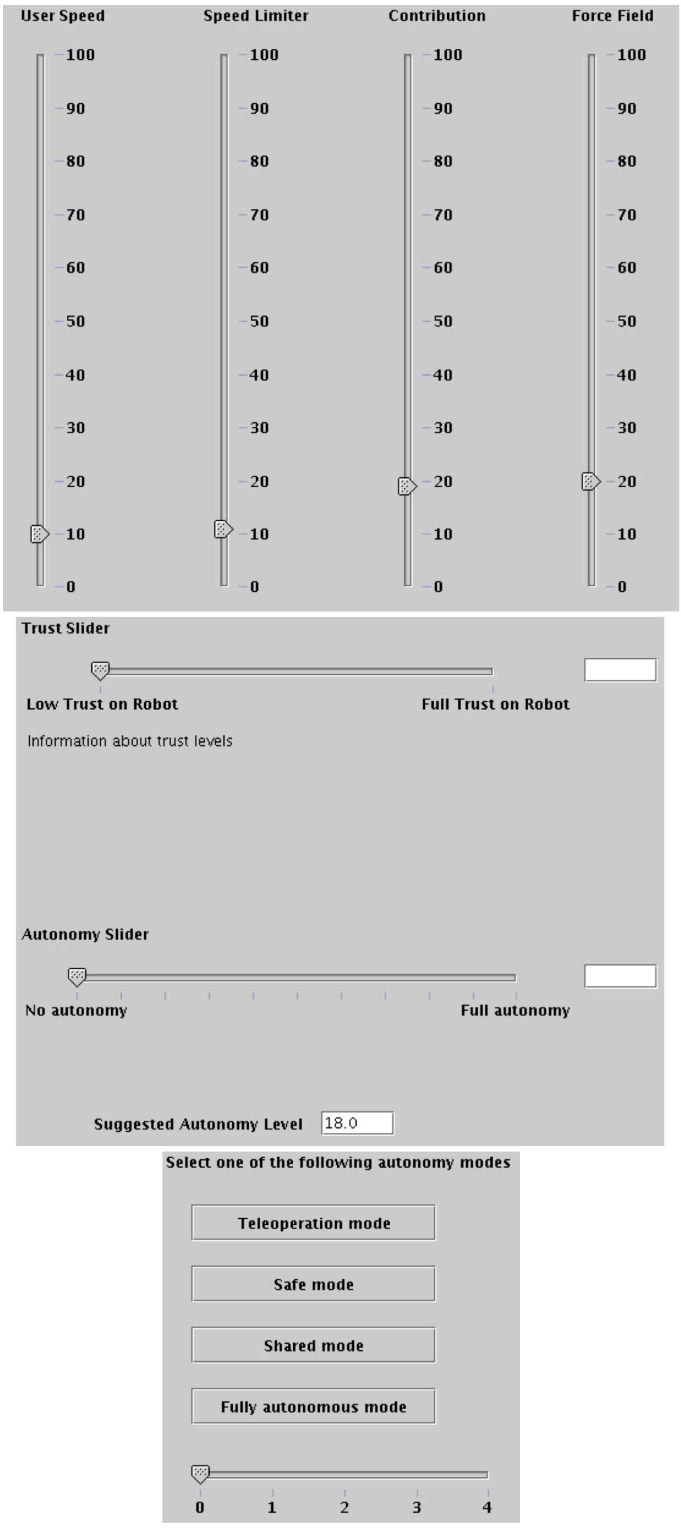

Figure 16. These figures show the differences between the three interfaces. The multiple slider system is shown in the top, the discrete autonomy system is shown in the bottom right, and the single slider system is shown in the bottom left.

Table 7. Map - Interface run sequencing. The first 12 runs were with novice users and the last 6 runs were with expert users. During each test run the users had to drive the robot in one of three maps (A, B, and C) with one of the three interfaces (1, 2, and 3).

Subject #	Run 1	Run 2	Run 3
1	1A	2B	3C
2	1B	3A	2C
3	2A	1C	3B
4	2C	3A	1B
5	3B	1C	2A
6	3C	2B	1A
7	1A	2B	3C
8	1B	3A	2C
9	2A	1C	3B
10	2C	3A	1B
11	3B	1C	2A
12	3C	2B	1A
13	1A	2B	3C
14	1B	3A	2C
15	2A	1C	3B
16	2C	3A	1B
17	3B	1C	2A
18	3C	2B	1A

4.6.2 Multiple slider system(MSS)

The multiple slider system, shown in Figure 14 and Figure 16, allows the user to change the system variable values over the entire range. This feature can be very helpful yet intimidating at the same time. It lets the user set the desired value for the system variables, but it also requires them to know the internal workings of the system.

4.6.3 Single slider system

As shown in Figure 15 and Figure 16, there are only two sliders in the interface. The slider on top is the trust slider and the slider below it is the autonomy slider. Currently the trust slider has two settings: low trust and high trust. The single slider low trust (SS Low) system and the single slider high trust (SS High) system can be

selected using the trust slider. The autonomy slider has a range from 0 to 1 with increments of 0.01. The autonomy slider reflects and controls the autonomy level.

The low trust mode indicates that the user has low trust on the robot and so the robot does not take the initiative to change the autonomy level. In the low trust mode, the user can set the autonomy level to the desired level. The robot does continuously keep recommending its desired autonomy level as a suggestion to the user.

The high trust mode indicates that the user trusts the robot. The robot takes the initiative to change the autonomy level to its desired value. Due to this the user is unable to change the autonomy level.

CHAPTER 5
RESULTS AND DISCUSSION

I present the results that were obtained from the user tests conducted, along with the analysis of those results in this chapter. The results and detailed analysis of the user's performance is presented in the first two sections. The performance was judged based on the number of hits during the test runs and the time required to finish the test runs. In the later sections information about learning effects along with information about trust are presented.

5.1 Hits per Interface

We considered hits to be an important metrics of performance. In certain application domains such as urban search and rescue it is important to have as few hits as possible. The result and analysis of the hits is presented in four sections, one for novice users, one for expert users, all the users combined, and finally comparison between the expert and novice users.

5.1.1 Novice Users

For the novice users, we hypothesized that the single slider systems would result in fewer hits than the discrete autonomy system (DAS) and multiple slider system (MSS). More hits were expected in the multiple slider system because the novice users did not fully understand the system variables and hence could not change the system variable values to best suit the existing situation. More hits were expected in DAS because of the system's inability to automatically adjust to the changing

environment and the limited options provided. Fewer hits were expected with the single slider systems because they could change the system variables in response to the changing surroundings.

Table 8 shows the mean hits in each autonomy system by novice users, along with the standard deviation and percent hits for each interface. Figure 17 shows the box plots for the same data set and column 3 of Table 10 shows the significance levels for differences between the four systems.

Table 8. Hits per Interface for Novice users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	5.50	4.00	3.42	2.83
σ	5.45	3.59	3.31	2.04
%	34.92	25.40	21.69	17.99

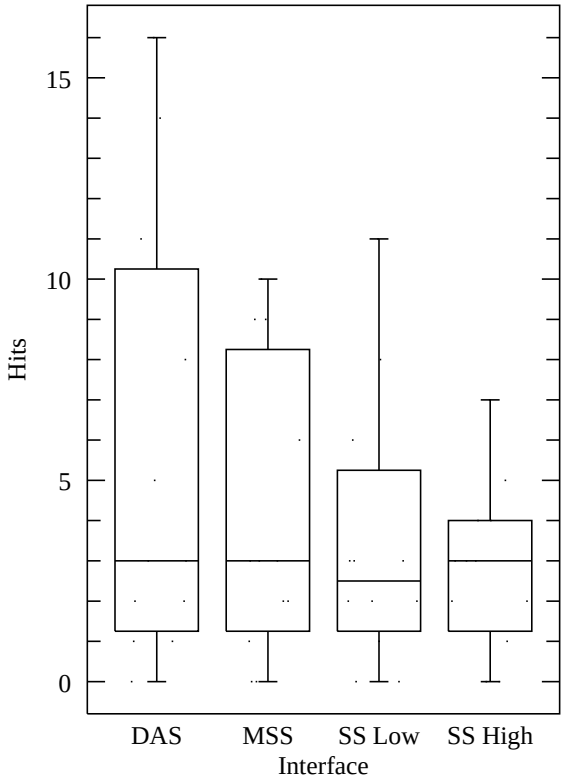

Figure 17. Hits per interface for novice users.

As expected, the least number of hits by novice users were with the single slider high trust system (HIGH) (μ = 2.83, 17.99%). The highest numbers of hits were with the DAS (μ = 5.5, 34.92%). However, no significant difference was found between the number of hits in HIGH and the DAS (ρ = 0.086). [1]

5.1.2 Expert Users

For the expert users we hypothesized that the number of hits with single slider systems would be the same as the multiple sliders. The expert users were expected to change the system variable values using sliders in response to the changing environment, just as the single slider systems would automatically change them. So only a slight improvement in the single slider system over the multiple slider system was expected.

The expert users had the least number of hits overall (μ = 1.0, 10.53%) with the low trust single slider system. However, the high trust single slider system (HIGH) resulted in the highest number of hits (μ = 4.17, 43.86%) for expert users and was significantly higher ($\rho \leq 0.05$) than single slider low trust system (LOW) and DAS. The primary reason for this unexpected result might be the disparity between the expected autonomy level by the expert users and actual autonomy level set by the sliding scale system. The low trust system, on the other hand, let them set the autonomy level to their desired value and only changed the other settings. The difference in hits between the low trust and the high trust systems was found to be significant for expert users (ρ = 0.008).

The expert users in the low trust system had at most half as many hits as the discrete autonomy system and multiple slider system. Five out of six users had 1 or no hits, with only one user having 4 hits. Table 9 shows the mean hits in each autonomy system by expert users, along with the standard deviation and percent hits for each system. Figure 18 shows the box plots for the same data set and column 4 of

[1] All the tests were performed using paired 1-tail t-test, unless otherwise mentioned) or any other combinations of autonomy systems.

Table 10 shows the significance levels for differences between the four systems. Even though the low trust single slider system resulted in the fewest number of hits, there was no significant difference in hits between the discrete autonomy system ($\rho = 0.12$) and multiple slider system ($\rho = 0.21$).

The expert users also had significantly ($\rho = 0.03$) more hits in the high trust mode than in the discrete autonomy mode. This high number of hits in HIGH was another unexpected result and leads us to the conclusion that expert users have very specific expectations from an autonomous agent and that their performance degrades significantly if those are not met. They were very familiar with the discrete autonomy system as it modeled one of our research systems and in the multiple slider system they could change the robot's contribution easily.

Table 9. Hits per Interface for Expert users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	2.33	2.000	1.00	4.17
σ	2.07	2.28	1.55	2.64
%	24.56	21.05	10.53	43.86

5.1.3 All users

In general, we hypothesized that users would have fewer hits with both single slider systems than with the discrete autonomy system and the multiple slider system. As expected, the single slider low trust system ($\mu = 2.61$, 19.11%) and single slider high trust system ($\mu = 3.28$, 23.98%) had fewer hits than the discrete autonomy system ($\mu = 4.44$, 32.52%) and multiple slider system ($\mu = 3.33$, 24.39%). As shown in Table 10, no significant difference in hits between the autonomy systems was found for all users combined. Table 11 shows the mean hits in each autonomy system by all the users, along with the standard deviation and percent hits. Figure 19 shows the box plots for the same data set. For all users combined, the low trust system had the least number of hits ($\mu = 2.61$, 19.11%), however, this was not significant.

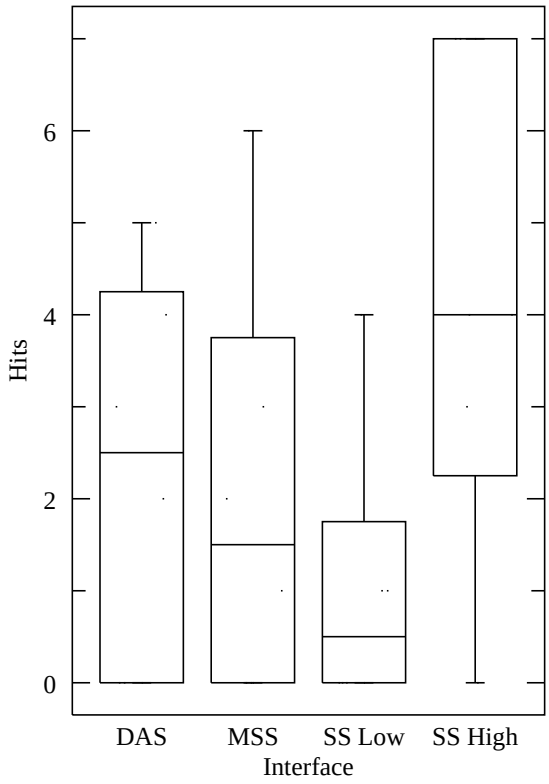

Figure 18. Hits per interface for expert users.

Table 10. Significant of hits between different interfaces for all, novice and expert users (using paired 1-tail t-test).

Difference in hits between	All users (ρ)	Novice users (ρ)	Expert users (ρ)
DAS v Multiple	0.17	0.17	0.42
DAS v Low	0.06	0.11	0.12
DAS v High	0.19	0.09	0.03
Multiple v Low	0.19	0.30	0.20
Multiple V High	0.48	0.16	0.14
Low v High	0.20	0.27	0.008

Table 11. Hits per interface for all users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	4.44	3.33	2.611	3.28
σ	4.78	3.29	3.032	2.27
%	32.52	24.39	19.11	23.98

5.1.4 Expert vs. Novice Users

Expert users using the single slider low trust mode had fewer hits than any other combination of autonomy system and user. It was also found that the expert users

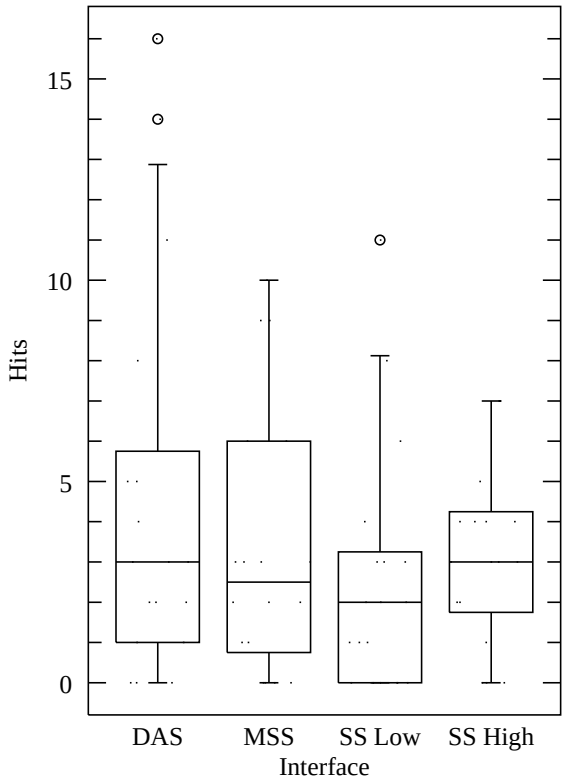

Figure 19. Hits per interface for all users.

had significantly ($\rho = 0.037$ using unpaired 1-tail t-test) fewer hits in multiple slider system than novice users had while using discrete autonomy system. Unpaired t-tests were used as there were only six expert users and twelve novice users.

The results in this subsection highlight the differences. Expert users performed significantly better in the low trust system than novice users in any other system. On the other hand, expert users with the multiple slider system only performed significantly better than novice users in discrete autonomy system. Table 12 presents the significance in the difference in hits between systems tested by expert users and novice users.

Table 12. Comparison of hits between expert and novice users with related significance levels (using unpaired one - tailed t-test).

Expert users		Novice users		
System	μ	System	μ	ρ
Low	1.00	DAS	5.5	0.009
Low	1.00	Multiple	4.0	0.01
Low	1.00	Low	3.42	0.02
Low	1.00	High	2.83	0.03
Multiple	2.00	DAS	5.5	0.03

5.2 Time per Interface

We considered time to be another metrics of performance. The results and analysis of the time is presented in three sections, one for novice users, one for expert users, and finally all the users combined.

5.2.1 Novice Users

We hypothesized that the single slider systems would require the least run time. The single slider systems would adjust the system variables including the user speed automatically to the optimal values, making it easier for the users to drive the robot and hence lower the run time. This turned out to be true as users took less time in low trust (μ = 180.25, 22.96%) and high trust systems (μ = 168.0, 21.4%) as compared to discrete autonomy (μ = 236.0, 30.06%) and multiple slider systems (μ = 200.92, 25.59%). Only the difference between the high trust system and the discrete autonomy system was found to be significant (ρ = 0.0196). Table 13 shows the mean run time in each autonomy system by expert users, along with the standard deviation and percent run time for each system. Figure 20 shows the box plots for the same data set. The novice users took less time in the high trust system because the robot generally tended to maintain a minimum level of autonomy. The robot's initiative, when added to the user's input, increased the overall speed of the robot.

No correlation was found between the run time of novice users in high trust mode and the number of hits.

Table 13. Time per Interface for Novice users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	236.00	200.92	180.25	168.00
σ	87.33	91.46	78.79	92.82
%	30.06	25.59	22.96	21.40

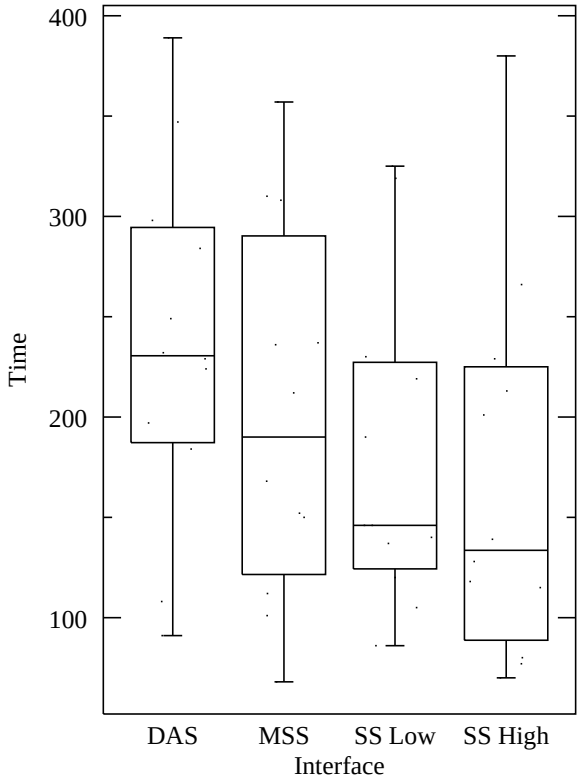

Figure 20. Time per interface for novice users.

5.2.2 Expert Users

The performance of expert users was consistent with that of novice users. Expert users required less time in the low trust (μ = 196.5, 23.17%), high trust mode (μ = 178.5, 21.05%) than the discrete autonomy system (μ = 273.5, 32.25%) and multiple slider system (μ = 199.5, 23.53%). Like the novice users, the expert users took significantly

more time in the discrete autonomy system than the low trust system ($\rho = 0.0354$) and high trust system ($\rho = 0.0198$). The experts might have taken significantly more time in the discrete autonomy system than the single slider systems due to the same reasons for novice users. Table 14 shows the mean run time in each autonomy system by expert users, along with the standard deviation and percent run times for each system. Figure 21 shows the box plots for the same data set.

Table 14. Time per Interface for Expert users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	273.50	199.50	196.50	178.50
σ	81.07	73.08	75.90	67.94
%	32.25	23.53	23.17	21.05

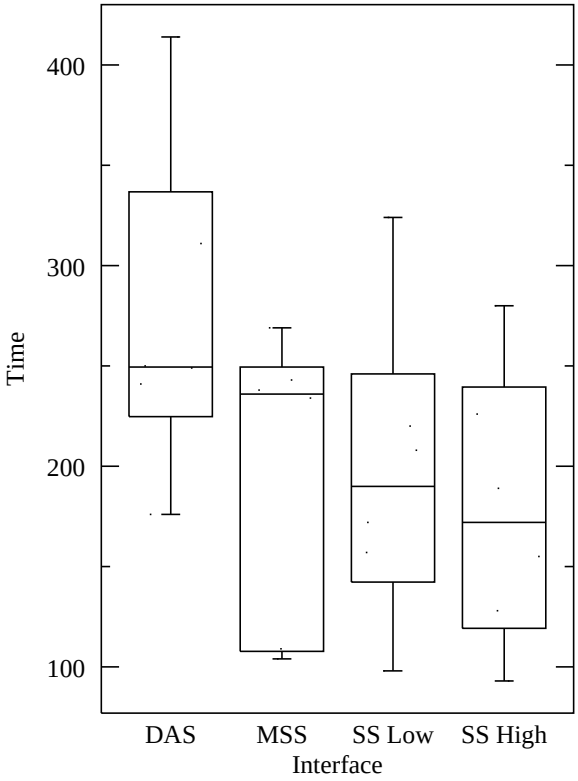

Figure 21. Time per interface for expert users.

5.2.3 All Users

We hypothesized that the single slider systems would provide lower run times than the discrete autonomy system and the multiple slider system. It was found that users took less time in the single slider high trust system than any other system, as can be seen in Table 15. Table 15 also shows the mean run time in each autonomy system by all users, along with the standard deviation and percent run time for each system. Figure 22 shows the box plots for the same data set. Users took significantly (ρ = 0.0029) less time in the high trust system (μ = 171.5, 21.27%) than the discrete autonomy system (μ = 248.50, 30.83%). Even though the users took less time in the high trust system (μ = 171.5, 21.27%) than the multiple slider system (μ = 200.44, 24.87%), the difference was not significant (ρ = 0.0774). In line with our hypothesis, the low trust system (μ = 185.67, 23.03%) also took significantly (ρ = 0.0104) less time than the discrete autonomy system (μ = 248.5, 30.83%). We expected that users would take more time in the multiple slider system (μ = 200.44, 24.87%). However, users took more time in the discrete autonomy system (μ = 248.5, 30.83%). This difference was not significant (ρ = 0.077).

Table 15. Time per Interface for All users.

	DAS	MSS	SS Low Trust	SS High Trust
μ	248.50	200.44	185.67	171.50
σ	84.84	83.57	75.98	83.41
%	30.83	24.87	23.03	21.27

5.3 Hits per Map

Map C had more hits than map A and map B in all three catogeries (all users, novice users, and expert users). The number of hits in map C was significantly more than that in map A for all users (ρ = 0.0147) and novice users (ρ = 0.0361). All users also had significantly more hits (ρ = 0.0341) in map C than in map B. The difference in

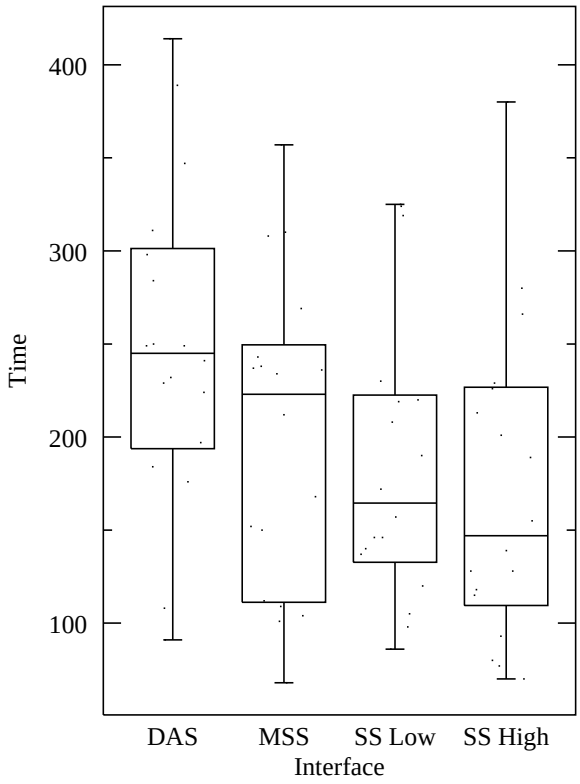

Figure 22. Time per interface for all users.

the number of hits for map C and map A were not significant for expert users ($\rho = 0.1240$). It is clear from this analysis that map C was more difficult. This might be in part because map C had a fork in the course that most users found hard to navigate and in part because map C had narrower sections than other maps. We cannot figure out the apparent reason for significantly more hits in map B than map A since both were roughly the same. Table 16 presents the mean hits, standard deviation and percent hits for novice, expert and all users in the three maps. Figure 23 presents the same information in box plots. Since the maps were rotated for each run, these effects were averaged across interface types.

Table 16. Comparison of hits by novice, expert and all users in different maps.

		Map A	Map B	Map C
Novice	μ	1.67	2.83	3.38
	σ	2.55	3.69	4.33
	%	21.74	37.27	40.99
Expert	μ	1.08	1.25	2.42
	σ	2.11	1.82	2.57
	%	18.75 3	4.38	46.88
All	μ	1.47	2.31	3.06
	σ	2.40	3.25	3.82
	%	21.24	36.79	41.97

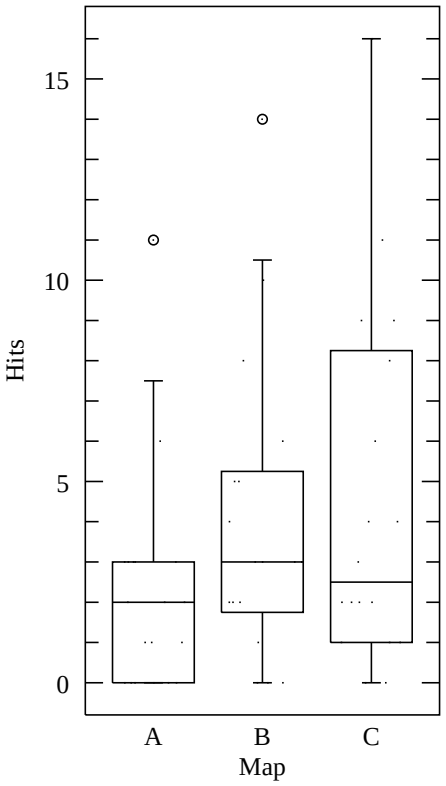

Figure 23. Hits per map.

5.4 Time per Map

All users combined and expert users alone users took significantly less time in map A than map B and map C. The three catogeries of users also took more time in map B, but the time difference between map B and map C was not significant for all categories. Table 17 presents the level of significance for the time difference between

the 3 maps for novice, expert and all users combined. Figure 24 present the box plots for the same. Table 18 shows the mean run time, standard deviation and percent run times for novice, expert and all users in the 3 maps.

One of the possible reasons for users taking the maximum amount of time in map B could be one specific turn that most users found difficult. The turn was difficult as it was not in a clearly visible location and it had a very narrow opening. Since all the maps had almost the same length, the only reason for the significant difference in the run time between map C and map A could be because users found map C more difficult. Since the maps were rotated for each run, these effects were averaged across interface types.

Table 17. Level of significance of difference in run time between different maps for all, novice and expert users.

	All users (ρ)	Novice users (ρ)	Expert users (ρ)
A v B	0.0039	0.0669	0.0068
A v C	0.0057	0.0392	0.0477
B v C	0.3222	0.4349	0.1812

Table 18. Comparison of run time by novice, expert and all users in different maps.

		Map A	Map B	Map C
Novice	μ	153.13	237.75	198.00
	σ	77.19	79.08	93.20
	%	27.77	36.67	35.55
Expert	μ	140.13	273.86	227.78
	σ	51.95	33.49	76.88
	%	22.40	40.75	36.84
All	μ	148.79	248.74	208.72
	σ	68.89	69.70	87.24
	%	25.89	38.11	36.01

5.5 Learning Effect

To simplify the calculation process, the four runs were grouped into 3 runs; as each map had equal number of extra runs. Based on the results, there was no significant

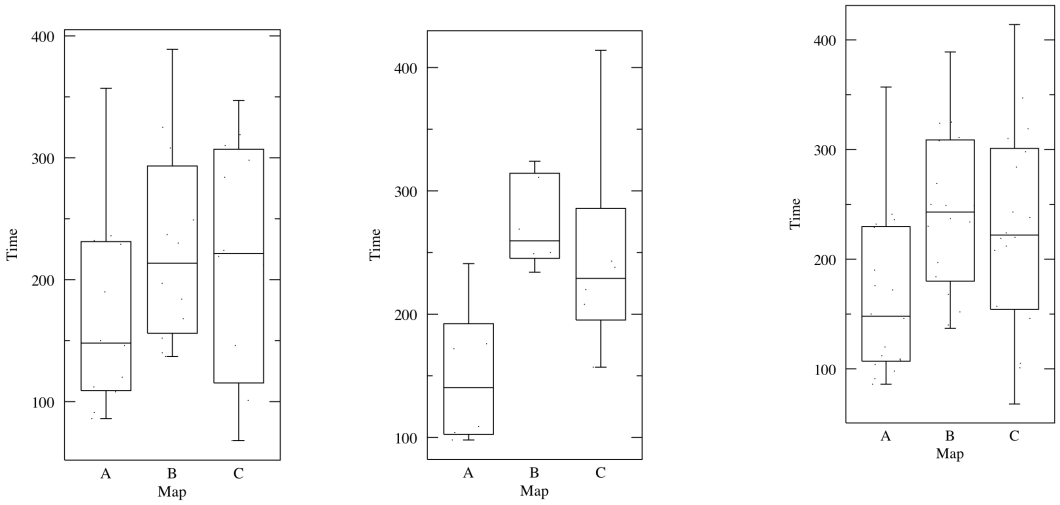

Figure 24. Time per map for novice (left), expert (center), and all users (right).

difference in run time and hits between runs as can be seen in Table 19. In fact, the percent hits and time for each run for all users combined was around 33% and is represented in the box plots in Figure 25 and Figure 26. There was no learning effect as there was almost no difference within runs for time and hits, let alone any significant difference.

Table 19. Level of significance of difference in run time and hits between different the 3 runs.

Run vs Run	Hit per run (ρ)	Time per run (ρ)
1 v 2	0.3844	0.2092
1 v 3	0.4526	0.208
2 v 3	0.447	0.4002

5.6 Experience with Joysticks

The pre-test questionnaire asked if the users had any previous experience with joysticks. This information was later compared with their performance measured in run time and hits. There were 5 users who had no prior experience with joysticks. We found that there was medium correlation (r = -0.468) between previous experience

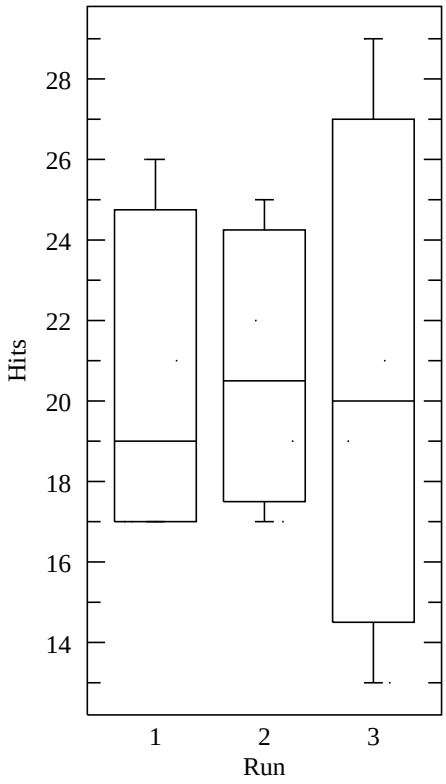

Figure 25. Hits per run.

with joysticks and the number of hits. No significant correlation was found between previous joystick experience and run time (r = 0.178). The average number of hits by users who had prior experience with joysticks ($\mu = 11.23$) was less than that of those who did not have prior joystick experience ($\mu = 20.0$). But this difference was not significant ($\rho = 0.0512$ using unpaired 1-tail t-test). Relative to run-time there was no significant difference ($\rho = 0.2843$ using unpaired 1-tail t-test) either. Since all of the users who had no prior joystick experience were novice users this result was to be expected. Another factor might be that many users listed using joysticks even when they only had rarely used joysticks with old game consoles.

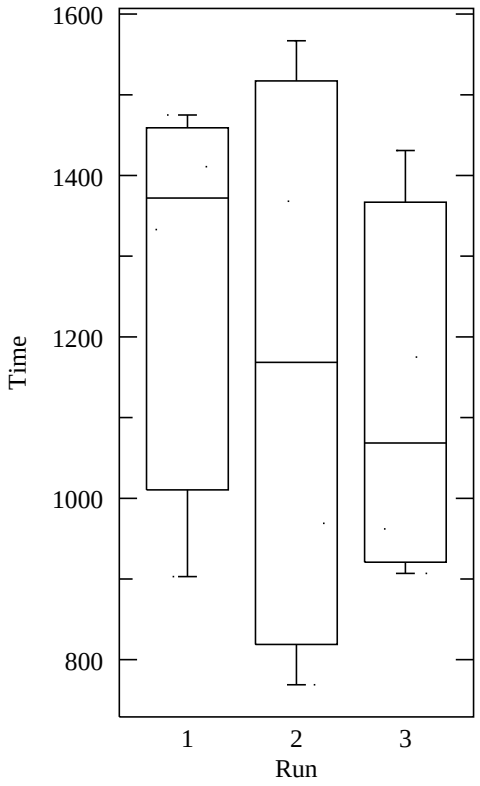

Figure 26. Time per run.

5.7 Experience with Video Games

As part of the pre-test questionnaire users were asked if they played games along with which genre of games they played. First person shooter (FPS) is a genre of video games. FPS and flight simulator games have characteristics like soda straw views that are also typical of mobile robots (Woods, Tittle, Feil, and Roesler 2004). There were eight users, who played either FPS games or flight simulator games. Of these eight users, six were expert users and the remaining two were novice users. Absolutely no correlation was found between the two groups with respect to run time (r = 0.0383) and there was medium correlation for hits (r = -0.5906). Users who played those games ($\mu = 8.3$) had significantly ($\rho = 0.0037$ using an unpaired 1-tail t-test) lower hits than those who did not ($\mu = 18.1$). Since 6 out of 8 users who played FPS or flight simulator games were expert users, the results were not surprising.

5.8 Expert vs. Novice users

The performance in terms of total run time and total hits for expert users was expected to be better than that of novice users. From Table 20, it is clear that expert users (μ = 9.5) had significantly ($\rho = 0.0379$ using an unpaired 1-tail t-test) fewer hits than novice users ($\mu = 15.75$). The total run time for expert users ($\mu = 848$) was greater than that of novice users ($\mu = 785.17$), but not significantly different ($\rho = 0.2595$ using unpaired 1-tail t-test). An interesting trend that can be observed in Table 20 is that the novice users took less time than expert users, but at the cost of more hits. Some correlation was found between the total run time and total hits for all users (r = -0.426).

Table 20. Mean hits and run time for novice, expert and all users.

		Hits	Run time
Novice	μ	15.75	785.15
	σ	9.62	229.36
Expert	μ	9.5	848.0
	σ	4.32	166.52
All	μ	13.67	806.11
	σ	8.64	207.66

5.9 Run time vs. Hits

The relationship between hits and time for novice users is directly proportional (r = 0.99), as can be seen in Figure 27. For the expert users, there is no such correlation. The most likely reason for this is the unexpectedly high number of hits that the expert users had in the single slider high trust system. This effect can be seen in Figure 27 as the right most point on the curve for expert users. If there had been fewer hits in the high trust mode by expert users like the novice users then that data point would have been to the left of the leftmost point. In that case the curve, though not linear, would represent a monotonically increasing function of time and hits. Even if that

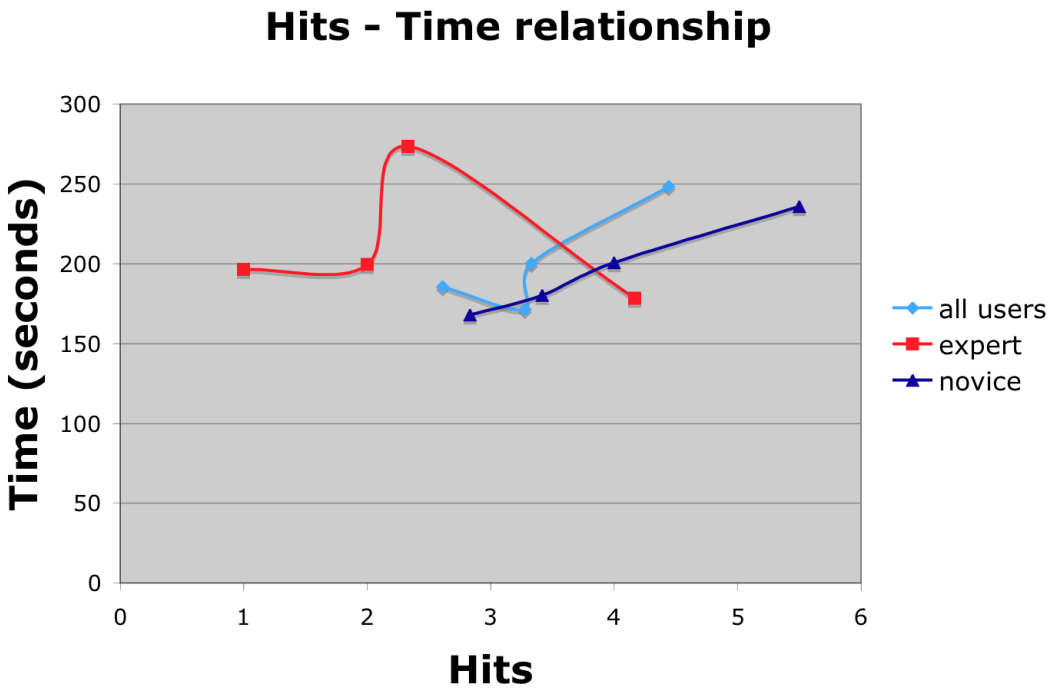

Figure 27. Time vs Hits by novice, expert and all users.

were the case, it is not possible to deduce anything from this alone, because these are data points from different systems and not the same system.

Figure 28 shows the relationship between hits and the different interfaces for novice, expert, and all users combined. Expert users have fewer hits than the novice users in all the autonomy systems, except for the previously mentioned anomaly. The low trust single slider system has fewer hits than DAS and MSS for novice and expert users.

Figure 29 shows the relationship between runtime for the different interfaces. As mentioned above, the expert users took more time than the novice users in all the interfaces. One interesting point to observe is the expert and novice users both

took the same amount of time for the multiple slider system, but the novice users had twice the number of hits than the expert users.

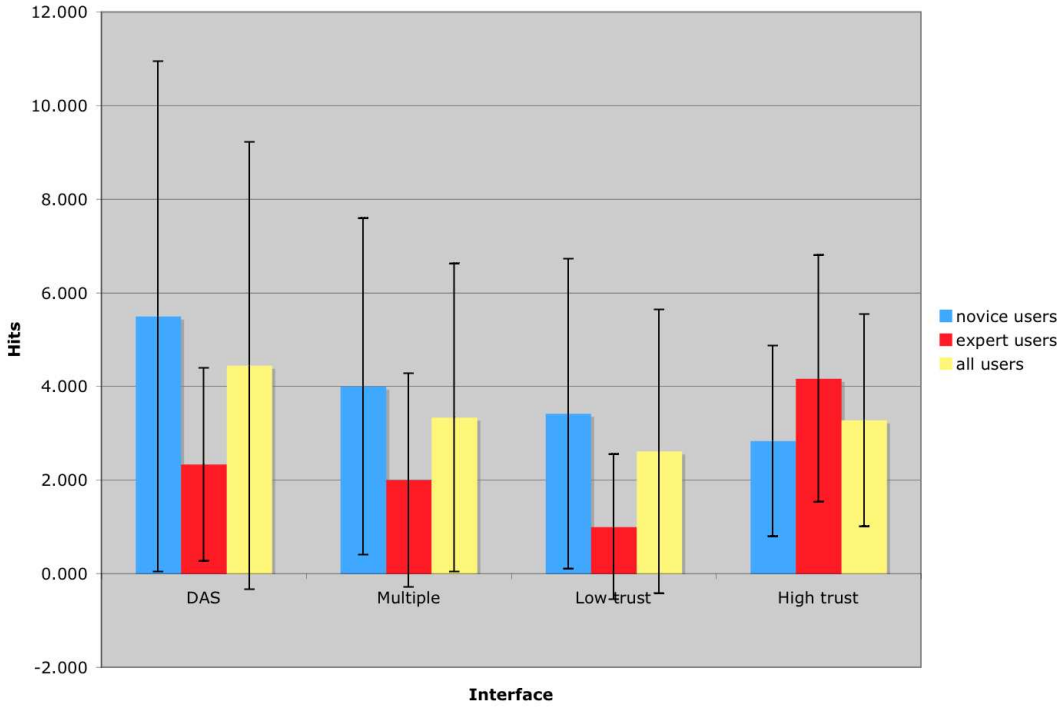

Figure 28. Hits by novice users, expert users and all users in all four interfaces.

5.10 Trust

As part of the post-questionnaire (shown in Appendix A) the users were asked to indicate how much they trusted each autonomy system on a scale of 0 to 10. These results were then ranked and the results are presented in Table 21 and Figure 30. The novice users trusted the single slider autonomy systems (μ =1.67 and μ = 2.17) more than the discrete autonomy system (μ = 2.67) and the multiple slider system (μ = 2.25). Statistical significance was only found for the low trust - discrete autonomy

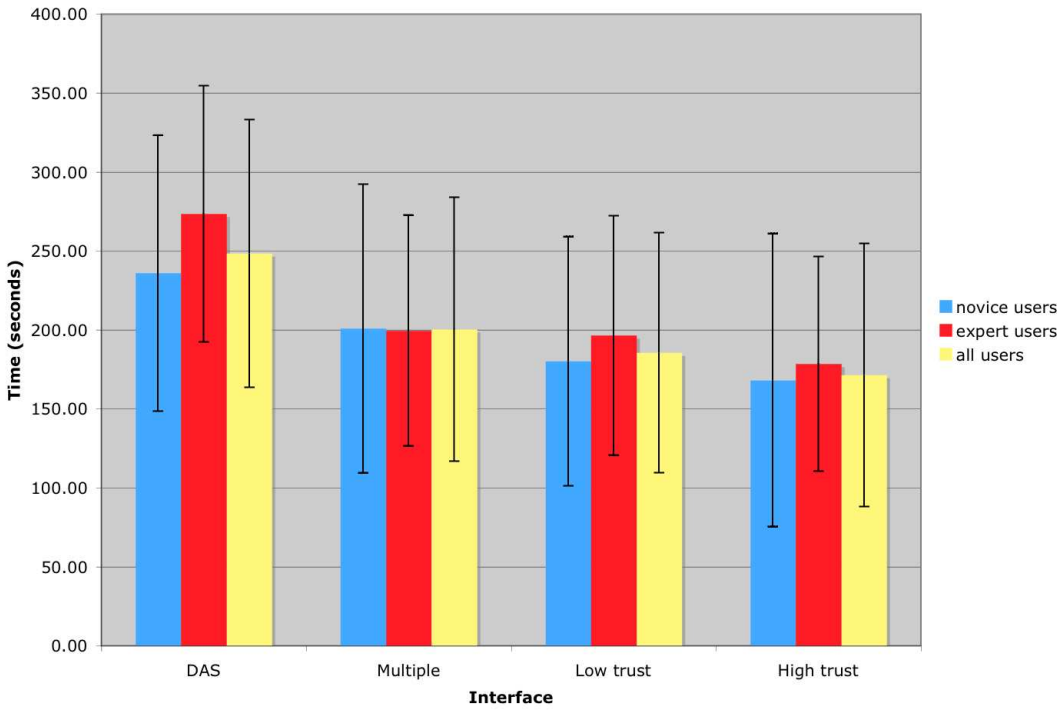

Figure 29. Time taken by novice, expert and all users in all four interfaces.

system ($\rho = 0.0194$) and the low trust - multiple slider system ($\rho = 0.445$). The expert users trusted the multiple slider system the most ($\mu = 1.67$). They ranked the high trust slider system the least, which goes to show that they were uncomfortable trusting the robot to change its own autonomy levels. However none of the results were found to be significant, as shown in Table 22.

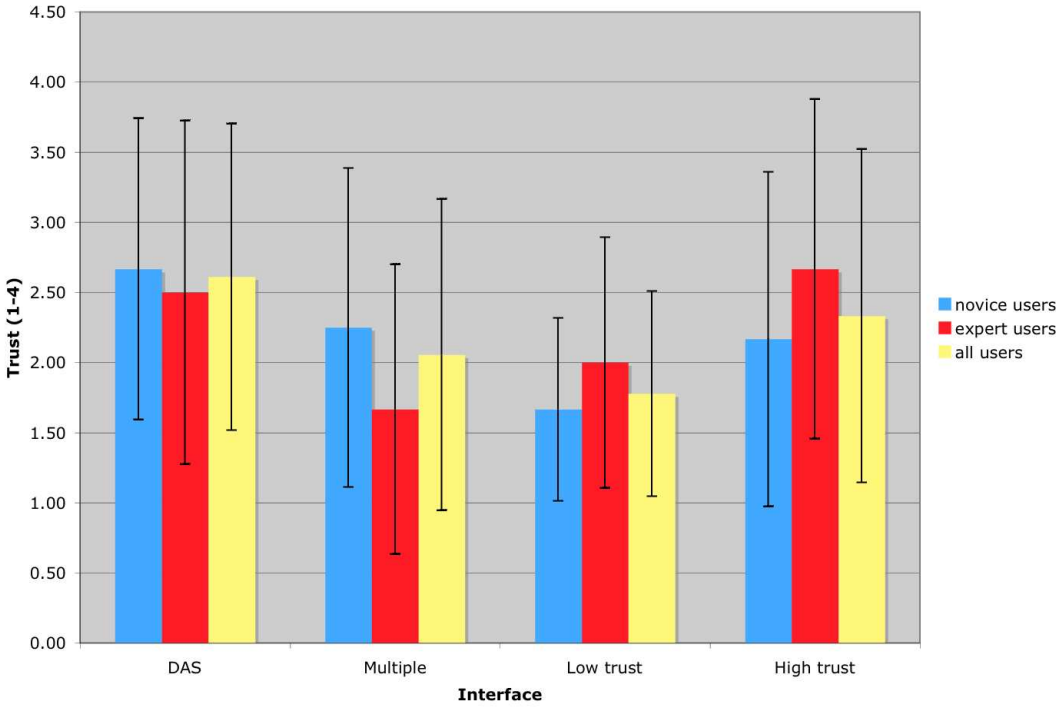

Figure 30. Trust shown by novice, expert and all users in all four interfaces.

Table 21. Mean trust per interface for novice, expert and all users.

		DAS	MSS	SS Low Trust	SS High Trust
Novice	μ	2.67	2.25	1.67	2.17
	σ	1.07	1.14	0.65	1.19
Expert	μ	2.50	1.67	2.00	2.67
	σ	1.22	1.03	0.89	1.21
All	μ	2.61	2.06	1.78	2.33
	σ	1.09	1.11	0.73	1.19

Table 22. Level of significance in difference between user's trust in the different interfaces.

	All (ρ)	Novice (ρ)	Expert (ρ)
DAS v Multiple	0.0580	0.1791	0.0926
DAS v Low	0.0102	0.0194	0.1816
DAS v High	0.2617	0.1545	0.4309
Multiple v Low	0.1922	0.0445	0.3191
Multiple v High	0.2720	0.4432	0.1016
Low v High	0.0677	0.1463	0.1638

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 Future Work

We would like to implement all of the levels of trust mentioned in Table 6. This would allow the users to get comfortable with the system at their pace and ultimately increase their performance. But as Olsen (Olsen and Goodrich 2003) mentioned, too much trust of the robot can be detrimental to the performance. Olsen states that more trust of robots results in higher neglect levels, which decreases the performance. This decrease in performance is in part because one of the side effects of neglect is reduced situation awareness (SA) of the robot's past and present. We plan to add a state summarization system (SSS) to the current architecture to negate that effect; the full architecture is shown in Figure 31. This system will continuously keep track of the robot's actions, the current state of the environment as perceived by the robot and the actions taken by robot and the user in response to the environment.

The users will also be able to query the state summarization system at different levels of granularity about the actions performed by the robot. The state summarization system will continuously monitor the suggestion vectors from the system variable agents, the system variables values after arbitration, and the conditions from the condition extraction system. This feature will allow it to answer the user's questions by linking the mentioned action to a change in the system variable and backtracking from there. This feature will also allow the summarization system to provide a detailed summary or a high level summary.

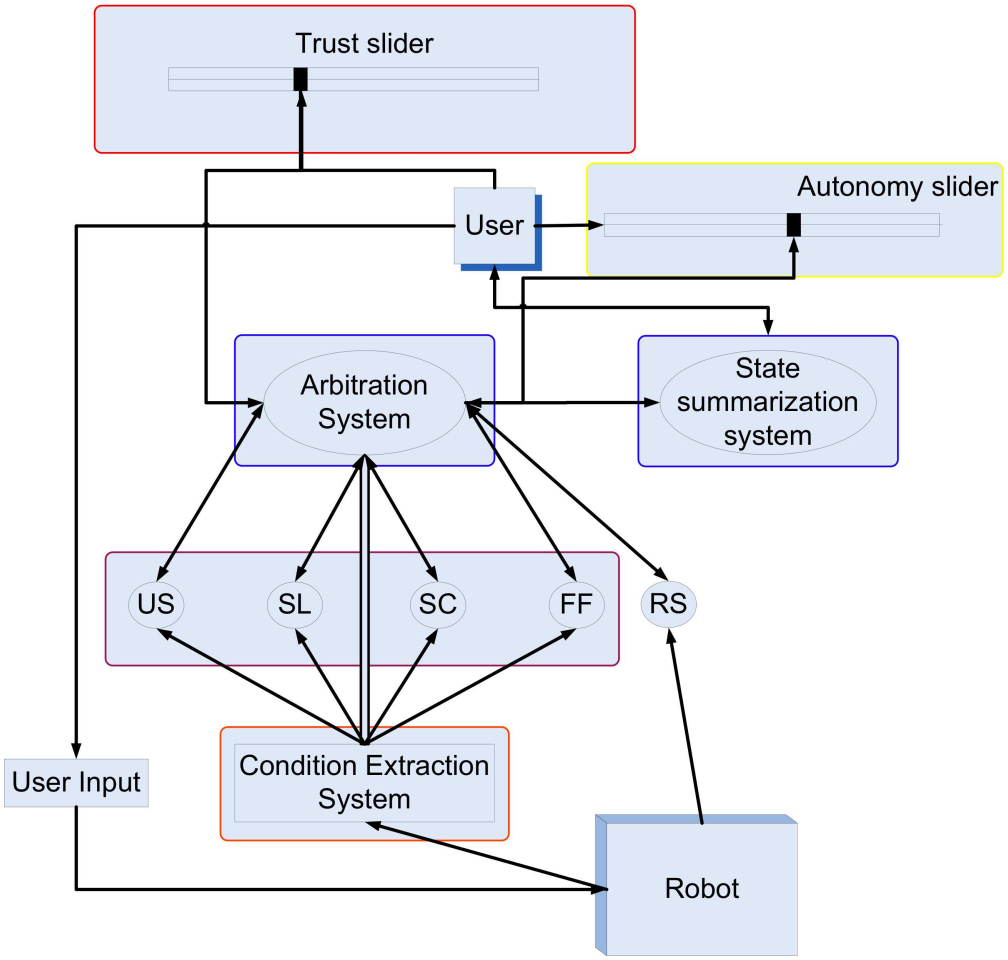

Figure 31. Architecture for future systems

State summarization can be a very useful feature in any autonomous mobile robot system. When the robot operates with some level of autonomy, the user can only guess the reasons for the robot's behavior. Having a state summarization system will eliminate this and provide the users with a better understanding of the robot's behaviors. It can also be useful in autonomous systems by providing a high level summary of events that took place when the user was not paying close attention to the robot. The state summarization system should also be useful in multi-agent systems where the user cannot continuously keep track of all the robot's actions while they were attending to other systems and frequently require a quick high level summary.

6.2 Conclusions

Sliding scale autonomy (SSA) systems improve mobile robot systems that share control between a user and a robot. SSA systems increase user performance by reducing the number of hits and by reducing the amount of time required to drive the robot. This improved performance can be useful in a wide array of application domains, but even more so in some domains such as urban search and rescue (USAR) where the robot is located at a distance from the user. In such situations, users are more likely to lose SA; thus operators would benefit from a system with flexible autonomy.

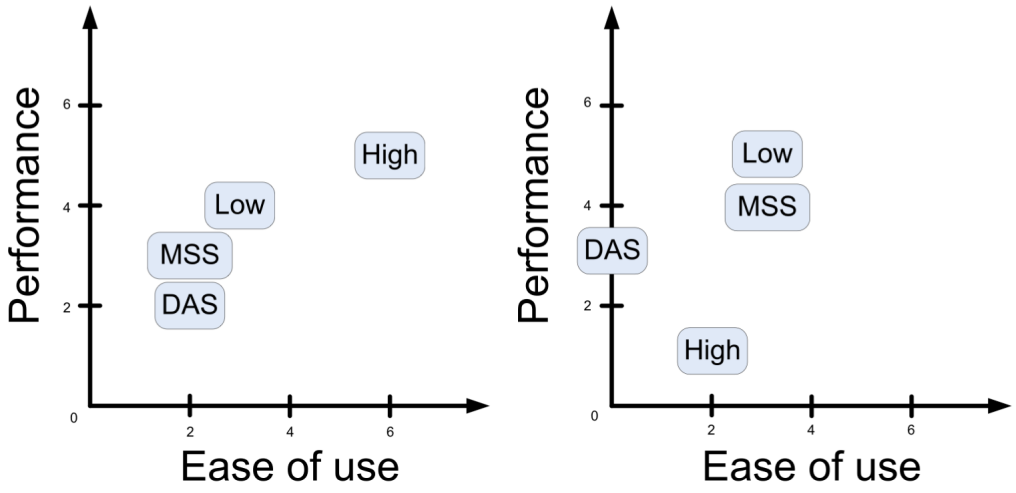

Figure 32. The two graphs are based on the data collected from the post-test questionnaire and the test runs. The graph on the left represents the data from the expert users and the one on the right is based on the data from novice users. In the graph for novice users it is easy to observe that the novice users had better performance using the single slider systems MSS and DAS. They also found the single slider systems to be easier to use than MSS and DAS.

A sliding scale autonomy system with trust provides the best of both worlds: the better performance than multiple slider system and easier to use than the discrete autonomy system. Even though multiple slider systems are sliding scale autonomy systems, we only refer to single slider systems as sliding scale autonomy systems. Figure 32 shows the performance versus ease of use graph for novice and expert users. The data about performance was gathered from the hits during the test runs and the

ease of use data was collected from the post-test questionnaire. More information can be found in Appendix B

The important contributions of this thesis are the sliding scale autonomy system which provides better performance than other autonomy systems and levels of trust at which an autonomous robot systems can operate.

Bibliography

Arkin, R. (1987). Reactive/reflexive navigation for an autonomous vehicle. *AIAA Computers in Aerospace Conference, 6 th, Wakefield, MA*, 298–306.

Baker, M., R. Casey, B. Keyes, and H. Yanco (2004). Improved interfaces for human-robot interaction in urban search and rescue. *IEEE Conference on Systems, Man and Cybernetics*.

Bayouth, M., I. Nourbakhsh, and C. Thorpe (1997). A hybrid human- computer autonomous vehicle architecture. In *Third ECPD International Conference on Advanced Robotics, Intelligent Automation and Control*.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre-1988] 2*(1), 14–23.

Brookshire, J., S. Singh, and R. Simmons (2004). Preliminary results in sliding autonomy for coordinated teams. *Proceedings of The 2004 Spring Symposium Series, March*.

Bruemmer, D., D. Dudenhoeffer, and J. Marble (2002, August). Dynamic autonomy for urban search and rescu. In *AAAI Mobile Robot Workshop*.

Bruemmer, D., D. Few, M. Goodrich, D. Norman, N. Sarkar, J. Scholtz, B. Smart, M. Swinson, and H. Yanco (2004). How to trust robots further than we can throw them. *CHI Extended Abstracts*, 1576–1577.

Burgard, W., A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun (1998). The interactive museum tour-guide robot. *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence*.

Cummings, M. and P. J. Mitchell (2006). Automated scheduling decision support for supervisory control of multiple uavs. *Journal of Aerospace Computing, Information, and Communication 3*(6), 294–308.

Desai, M. and H. Yanco (2005, August). Blending human and robot inputs for sliding scale autonom. In *Proceedings of the 14th IEEE International Workshop on Robot and Human Interactive Communication*.

Dorais, G., R. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost (1998). Adjustable autonomy for human-centered autonomous systems on mars. *Proceedings of the First International Conference of the Mars Society*, 397–420.

Dudek, G., M. Jenkin, and D. Wilkes (1993). A taxonomy for swarm robots. In *International Conference on Intelligent Robots and Systems*.

Gat, E. (1996). News from the trenches: An overview of unmanned spacecraft for ai researcher. In *Workshop Notes, Planning with Incomplete Information for Robot Problems, AAAI-96 Spring Symposium*.

Gat, E. (1997). On three-layer architectures. In D. Kortenkamp, R. Bonasso, and R. Murphy (Eds.), *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pp. 195–210. Menlo Park: MIT/AAAI Press.

Goodrich, M., D. O. Jr., J. Crandall, and T. Palmer (2001). Experiments in adjustable autonomy. In *Technical report version of paper in proceedings of the IJCAI01 workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*.

Heger, F., L. Hiatt, B. Sellner, R. Simmons, and S. Singh (2005). Results in sliding autonomy for multi-robot spatial assembly. *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 5–8.

Huang, H.-M., J. Albus, E. Messina, and R. Wade (2004). Specifying autonomy levels for unmanned systems: Interim report. In *SPIE Defense and Security Symposium*.

Marble, J., D. Few, and D. Bruemmer (2004). I want what you've got: Cross platform usability and human-robot interaction assessment. In *Proceedings of PerMIS'04*.

Nilsson, N. J. (1984). Shakey the robot. In *SRI Technical Note no. 323 (1984)*.

Nourbakhsh, I., R. Powers, and S. Birchfield (1995). Dervish - an office-navigating robot. *AI Magazine 16*(2), 53–60.

Olsen, D. and M. Goodrich (2003). Metrics for evaluating human-robot interactions. *Proceedings of PERMIS 2003*.

Sellner, B., R. Simmons, and S. Singh (2005). User modelling for principled sliding autonomy in human-robot teams. *Multi-Robot Systems: From Swarms to Intelligent Automata. Springer*.

Sheridan, T., R. Parasuraman, and C. Wickens (2000). A model for types and levels of human interaction with automation. In *IEEE Transactions on Systems, Man and Cybernetics Part A: Systems and Humans, Vol. 30, No. 3*.

Walter, W. (1961). *The Living Brain*. Am Psychosomatic Soc.

Woods, D., J. Tittle, M. Feil, and A. Roesler (2004). Envisioning human–robot coordination in future operations. *IEEE transaction on Systems, Man and Cybernetics—part A: Systems and Humans—Part C:applications and reviews 34*(2).

Yanco, H. and J. Drury (2004). "Where Am I?" acquiring situation awareness using a remote robot platform. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*.

APPENDICES

Appendix A
Questionnaires

A.1 Introduction

The users were asked to answer two questionnaires. The users answered the pre-test questionnaire after signing the IRB form and before being introduced to the robot and other aspects of the test. The users answered the post-test questionnaire once they finished testing the different interfaces.

A.2 Pre-test questionnaire

Gender: Female Male

Age:

Occupations:

1. What types of computer do you normally use? Check all that apply, and state the approximate number of years you've used each.

____ Macintosh Years: ____

____ PC Years: ____

____ Unix or Linux Years: ____

____ Others: _____ Years: ____

2. Have you ever operated a robotic system before? If yes, please explain.

Appendix A (Continued)

3. Approximately how much time do you spend using computers?

____ 0 - 3 hours

____ 3 - 10 hours

____ 10 - 20 hours

____ over 20 hours

4. What applications (e.g., Microsoft Word) do you use on a regular basis?

5. How would you rate your level of computer expertise? Please choose the description that matches the closest match.

____ Casual: Primarily occasional e-mail and web-browsing

____ Moderate: I do lot of regular work or leisure activities on a computer

____ Expert: I troubleshoot and upgrade applications or operating systems

____ Guru: Other come to me for solving their problems

6. How many hours a week do you play video games? ____

7. Which of the following platforms do you own (and use)?

____ PC

____ Playstation 1

Appendix A (Continued)

____ Playstation 2

____ Playstation 3

____ XBox

____ XBox 360

____ Gamecube

____ Gameboy / PSP

____ Others (Please specify) _____

8. Have you ever used a joystick before? If so please explain the context in which it was used (e.g., Flight simulator, etc)

9. What genre(s) of video games do you play the most?

___ First person shooter (Doom, Halflife, etc)

___ Action / Adventure (Tomb Raider, Grand Theft Auto, etc)

___ Real-Time Strategy (Starcraft, Age of Empires, etc)

___ Sports (Maiden, etc)

___ Racing (Grand Turismo, Formula 1, etc)

___ Puzzle / Board games (Chess, Tetris, etc)

___ Role-Playing games (Everquest, Baulder's Gate, etc)

___ Other(s) - Please specify below

_____

Appendix A (Continued)

10. How would you rate your ability at video games?

Bad Fair Average Good Excellent

11. How would you rate your ability to multitask?

Bad Fair Average Good Excellent

A.3 Post-test questionnaire

1. Which system did you like the most? Why?

2. Which system did you like the least? Why?

3. Which feature(s) did you like the most?

4. Which feature(s) did you not like?

Appendix A (Continued)

5. Which system was the easiest to use in confined spaces?

6. Which system was the easiest to use overall?

7. If you were to create a system how would it be?

8. In which system do you think you performed the best?

9. On a scale of 0 to 10 how much do you trust the robot in each system and why?

Appendix A (Continued)

Discrete autonomy mode:_____

Multiple Sliders:_____

Single slider with low trust:_____

Single slider with high trust:_____

Appendix B
Performance and ease of use data

Table 23. Ease of use versus performance (hits) for novice users. The numbers in column 2 shows the number of users that found the corresponding interface easy to use and the number in column 3 shows the number of users that had the least number of hits in the corresponding interface.

Interface	Ease of use	Performance (hits)
DAS	2	2
MSS	2	3
LOW	3	4
HIGH	6	5

Table 24. Ease of use versus performance (hits) for expert users. The numbers in column 2 shows the number of users that found the corresponding interface easy to use and the number in column 3 shows the number of users that had the least number of hits in the corresponding interface.

Interface	Ease of use	Performance (hits)
DAS	0	3
MSS	3	4
LOW	3	5
HIGH	2	1

Table 25. Ease of use versus performance (time) for novice users. The numbers in column 2 shows the number of users that found the corresponding interface easy to use and the number in column 3 shows the number of users that had the least run time in the corresponding interface.

Interface	Ease of use	Performance (time)
DAS	2	1
MSS	2	3
LOW	3	1
HIGH	6	7

Appendix B (Continued)

Table 26. Ease of use versus performance (time) for expert users. The numbers in column 2 shows the number of users that found the corresponding interface easy to use and the number in column 3 shows the number of users that had the least run time in the corresponding interface.

Interface	Ease of use	Performance (time)
DAS	0	1
MSS	3	2
LOW	3	1
HIGH	2	2