The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

TOWARDS IMPROVING HUMAN-ROBOT INTERACTION

A Thesis in

Computer Science and Engineering

by

Urmila Kukreja

© 2004 Urmila Kukreja

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2004

I grant The Pennsylvania State University the nonexclusive right to use this work for the University's own purposes and to make single copies of the work available to the public on a not-for-profit basis if copies are not otherwise available.

Urmila Kukreja

The thesis of Urmila Kukreja was reviewed and approved* by the following:

Frank Ritter
Associate Professor of Information Sciences and Technology
Associate Professor of Computer Science and Engineering
Thesis Advisor

Rajeev Sharma
Associate Professor of Computer Science and Engineering
Thesis Co-Advisor

Raj Acharya
Professor in Computer Science and Engineering
Head of the Department of Department or Graduate Program

*Signatures are on file in the Graduate School

ABSTRACT

Robotics has great potential and a broad range of application; robotic devices can be used in medical procedures, in urban search and rescue, and in a wide variety of other areas, to perform complicated tasks. However, controlling robotic equipment can be a difficult task, and functional interfaces must be developed to aid users in robot operation. Thus, we need to study and evaluate these interfaces to determine what components and functions should be incorporated and what can be left out. As an example, to evaluate the user interface on board the ER1 Robot, experiments were conducted with a number of participants, recording their interactions with the device. The amount of information the participants were provided about the ER1 robot varied to simulate users with varying levels of training with the system. A way in which the robot interface was further analyzed was create a cognitive model — a computer program that simulates human behavior using known aspects of human cognition and knowledge represented as rules that trigger sequences of events, much like our own thoughts do. The model was written in ACT-R 5.0 and used SegMan as its eyes and hands, to view the screen and to interact with the interface in the same way a user did. The model made predictions about how novice users may use the interface and its performance fit well with human data. It also indicated some problems with existing robot interfaces and how they can be improved. The model has some limitations, but it raises the bar of existing models in a way that it directly interacts with a robot interface.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I take this opportunity to express my deep sense of gratitude to my thesis advisor Dr. Frank Ritter, for his academic and financial support without which this work would not have been possible. He guided and encouraged me at every stage to make the whole process a very rewarding learning experience. I would also like to thank Dr. Rajeev Sharma, who also helped me in my thesis. His invaluable advice helped me complete my research. I thank Dr. Ling Rothrock who initiated my interest in the field of Human-Computer Interaction and helped my research find direction. I thank Dr. Rob St. Amant for his assistance and guidance whenever I needed it. I would also like to thank all the members of the Applied Cognitive Science Lab especially Andrew Reifers and Adrienne Bonski, without whose help I would have faced numerous difficulties in completing my research. I thank my parents and family whose constant support and love encouraged me all the way. Finally I would like to thank all my friends. I especially thank Sameer along with Smita, Jeshal, Parag, Dhiraj, Anupam, and Vinit for their immense help and moral support.

Chapter 1

Introduction

Urban Search and Rescue involves victim detection and rescue from collapsed structures after a calamity or attack. The search and rescue operation has both high mobility and high perceptual demands. The confined space and destroyed interiors in the rubble make maneuvering within it difficult and dangerous for humans or dogs. Identifying victims in time-critical situations is very stressful and due to fatigue, rescue workers may miss victims. Small mobile robots are hence very useful in USAR (Casper and Murphy, 2001), and their usability is important.

While research is being conducted toward building autonomous robots, most robots today are built to work with humans. In the exploration, surveillance, and medical assistance that robots are used for, humans interact with them in different roles. Scholtz (2003) identifies three main roles in which human-robot interaction can occur: humans act as a supervisor for the robot, as an operator, or as a team member. Certain tasks like victim detection are done autonomously by the robot sensors while certain others, like recovering victims from trapped locations, require human control. Heterogeneous teams of humans and robots in USAR operations perform better than only robots or humans (Murphy, Casper, Micire and Hyams, 2000). These teams correctly identify more victims and are able to navigate to challenging locations. A current goal for robot use thus is to create a synergetic team of humans and robots taking advantage of the skills each team member.

Research has been conducted along multiple domains (Kortenkamp and Dorais, 2000) to improve the way in humans and robots synergize in rescue and other operations. Some of these areas include research in mixed-initiative planning, machine-learning, and distributed artificial intelligence. Researchers in cognitive robotics (Kawamura, Rogers and Ao, 2002) believe that robot interaction could be improved if a robot recognized the person it was interacting with and behaved accordingly. The way the user interacts with the robot depends greatly on the design of its interface. It is thus important that the design and layout of an interface be intuitive so users can easily and quickly accomplish their tasks. This research is directed toward user modeling and improving robot interfaces for better human-robot interaction.

Human-machine interaction is studied in both sciences of automation and human-computer interaction (Amant and O.Reidl). However principles from neither of these can be directly applied in designing human-robot interaction (HRI). In HCI, the human is assumed to be in control, whereas in automation, the machine is in control, but monitored and supervised by the human. Unlike computers where humans request an action and see a deterministic result, robots exhibit a fair amount of autonomy and cognition in executing their tasks (Scholtz, 2003). Robots are used in harsh environments. In many situations to perform their task, the operator must be co-located in the confusing and stressful surroundings that the robots perform their operations. Thus, while certain general principles of HCI can be applied in robot interface design, HRI is different from HCI and a theory for the way humans interact with robots needs to continue to be developed.

Evaluating any interface can be a difficult and time-consuming process. The turnaround time for the design-build-test-fix for software is long and to have different users test interfaces each time is difficult and expensive. Ideally, we would like an automated tool that could test the usability and effectiveness of an interface(Ritter and Young, 2001; Byrne and Gray, 2003; Emond and West, 2003; Ritter, Rooy, Amant and Simpson, 2003). Here an attempt is made to create a model of a user for testing an interface. Building such a model has an initial hindrance of building the correct model, but once the model is built, it can be used repetitively reducing both cost and time(St. Amant and Ritter, 2005). Models once created can be used in assistance and training and in certain situations as substitute users.

Automatic models have been used successfully in the past to evaluate computer interfaces. In Project Ernestine (Gray, John and Atwood, 1992) Goals, Operators, Methods, and Selection Rules (GOMS) models of toll and assistance operators (TAO) were created for the existing and a proposed interface. The validated models correctly predicted that the new interface would slow down performance by 4% on average. The new interface was not purchased. The cost of the interface and the lower operator performance its use would have caused saved the company 2.4 million dollars per year. In this situation, an automatic model was both financially and scientifically useful.

While constructing user models to evaluate an interface, it is important that the model interact with the interface in the same way as humans do. Embodied models (Ritter and Young, 2001) are built in a cognitive framework and have eyes and hands to interact with an interface in a manner that is plausible to human behavior. Here the model is built using ACT-R (Anderson and Lebiere, 1998) as the cognitive architecture.

Cognitive architectures are composed of theories of human cognition that are fixed across tasks and individuals. ACT-R is extended with SegMan (Segmentation and Manipulation), an image processing tool that has routines to process a bitmap of an interface and motor routines to simulate human mouse and keyboard actions. The system of ACT-R and SegMan has previously been used to model user behavior in a driving game (Ritter et al., 2003).

To obtain information on how users interact with a robot, a study was conducted and the data collected. The model data was validated against human data. The limitations of the model and areas where it can be improved are then discussed. Ultimately, the goal is to provide a quantitative tool to guide the design process of human-robot interfaces. In this work the embodied models interact directly with a robot interface, and serve as explanations of users' behavior.

With a model that can directly interact with a fairly complex robot interface, it was possible to make useful predictions on how humans drive and program robots. It was found that good eye-hand co-ordination is extremely important to quickly and correctly drive the robot. It was also found that victim detection and obstacle avoidance are hard tasks, but knowledge of vehicle state and a global picture can help in improving performance in these operations. Based on the difficulties the model and the subjects faced, some suggestions are made to improve robot interfaces for better human-robot interaction.

Chapter 2

The Study

In order to understand human-robot interaction and to provide some simple tasks for users and models, a study was conducted of how users drive and program a robot.

2.1 Subjects

Thirty subjects were recruited for the study. Half of these were women and half men. These were all students between the age of eighteen and twenty-five. The users were asked to perform three tasks with the robot and took between 30-45 minutes to complete them. Because human subjects were involved, prior Institutional Review Board (IRB) approval was obtained from the Office of Research Protection. The participants were paid $7 each as compensation. All participants were extensively computer literate and were comfortable working in a Windows environment.

2.2 The ER1 Robot

This experiment was designed specifically to look at the Human-Robot Interface on the Evolution Robotics–1 robot (ER1) (www.evolution.com/er1). While the ER1 is an entry level robot lacking highly specialized technological capabilities, it definitely illustrates many analogous features and can perform similar tasks as more expensive robots used in the field.

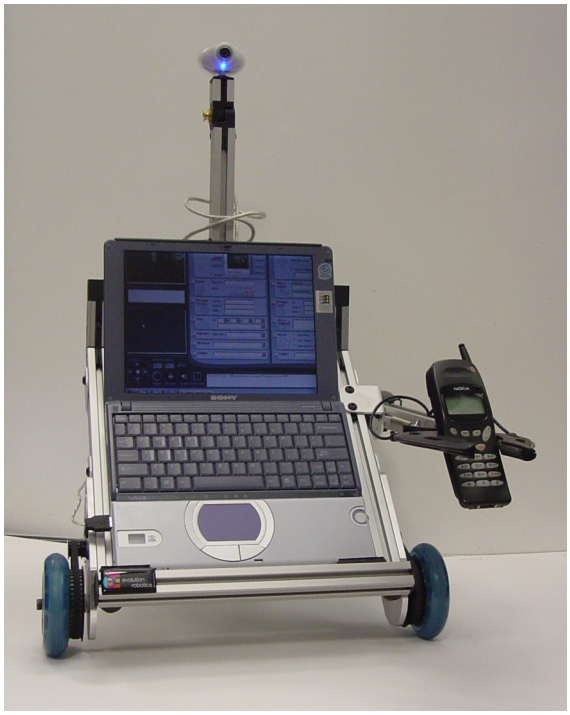

Figure 2.1 The ER1 robot.

On this 24 x 16 x 15 inch robot, a Dell laptop was placed that ran Windows XP to perform the robots on-board computing. The robot was tele-operated via another laptop, using wireless internet in a Virtual Private Network (VPN). The ER1 robot control center software was installed on both the laptops, providing the user interface on the fixed

platform for the subject and connection to the robot's camera, gripper, and motors to the other laptop via USB cables. The motors ran on a small battery on board the robot.

The ER1 Robot Control Center is used to control the robot's actions (see Figure 2.2). Using this interface, operators can observe the robot environment via the robot's camera, they can open and close the gripper, maneuver the robot through a room, as well as perform a variety of other tasks. The robot can be under manual control or can have behaviors saved that can later be executed.

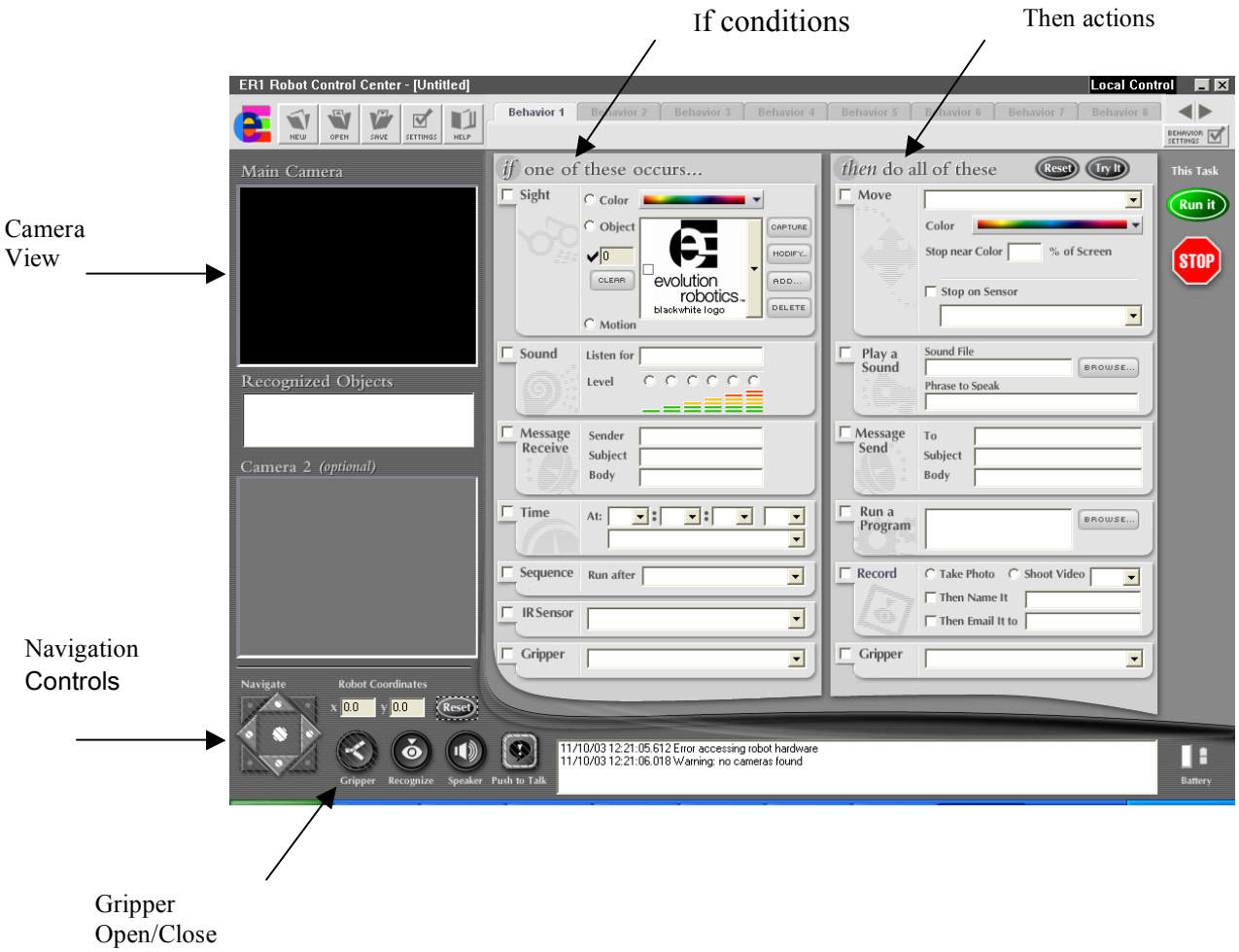

Figure 2.2 The ER1 Interface.

2.3 Collecting Data

As the subjects interacted with the robot, data needed to be collected to understand how users interact with the robot and develop a model of user actions. Timing records are useful for building and testing user models. With mouse movement and mouse click data the number of errors can be found which can be helpful in improving the interface.

There are three different ways in which user information can be obtained (Westerman, Hambly, Adler, Wyatt-Millington, Shryane, Crawshaw and Hockey, 1996). The first method, video recording is not feasible. Camtasia by TechSmith is an example of a tool to record and replay user interaction as a video file. However, timing information cannot be directly obtained and the tool creates logs that are resource intensive because they are videos. The second method, "instrumentation", cannot be used because the robot cannot be modified. The third method is to include an unobtrusive application that exists in the background, which can be used generically across all applications. Most of the applications currently available exist as spy ware and provide only keystroke data. A tool that provided user logs has been developed (Westermann et al., 1996) for the Windows 3.1 platform but it appears to be no longer available. Another tool developed (Trewin, 1998) can be used to obtain user interactions across generic interfaces but works only on the Macintosh platform.

Because no existing tool was available to obtain user interaction data, Recording User Interaction (RUI) (Kukreja and Ritter) was developed. RUI (see Figure 2.3) records mouse and keyboard data and logs it in a text file (see Figure 2.4) as timestamp, action, and argument. To study the data collected, it can be replayed faster or slower than real-

time. RUI is developed in the .NET framework using C#. Mouse and keyboard events of the Win32 API are used to capture mouse click, and keystroke data. To record mouse moves, a thread runs in the background checking if there is any change in the position of the mouse, if so, a mouse move event is raised.

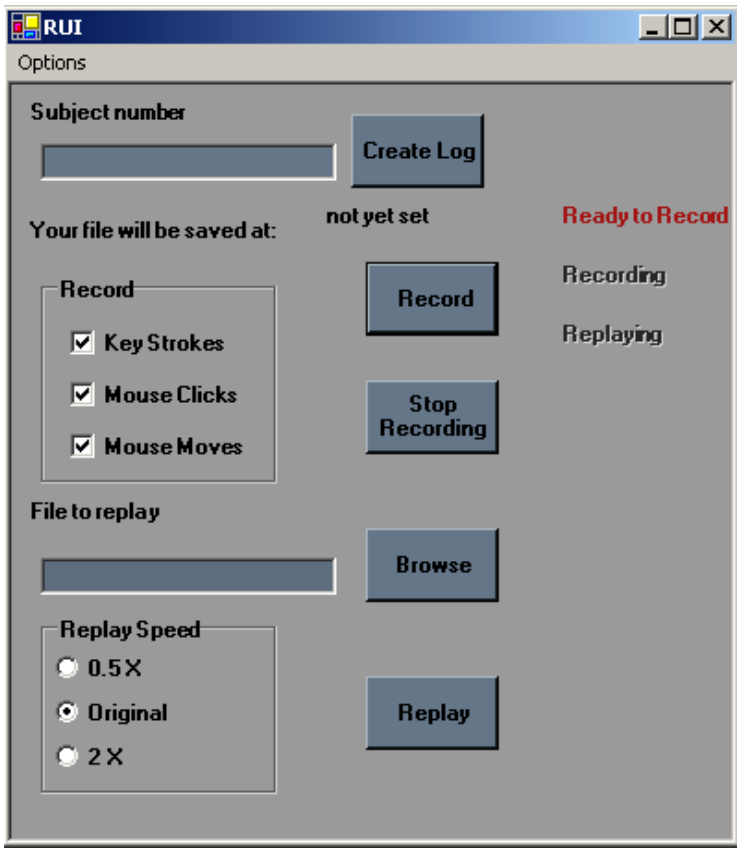

Figure 2.3 RUI's graphical interface.

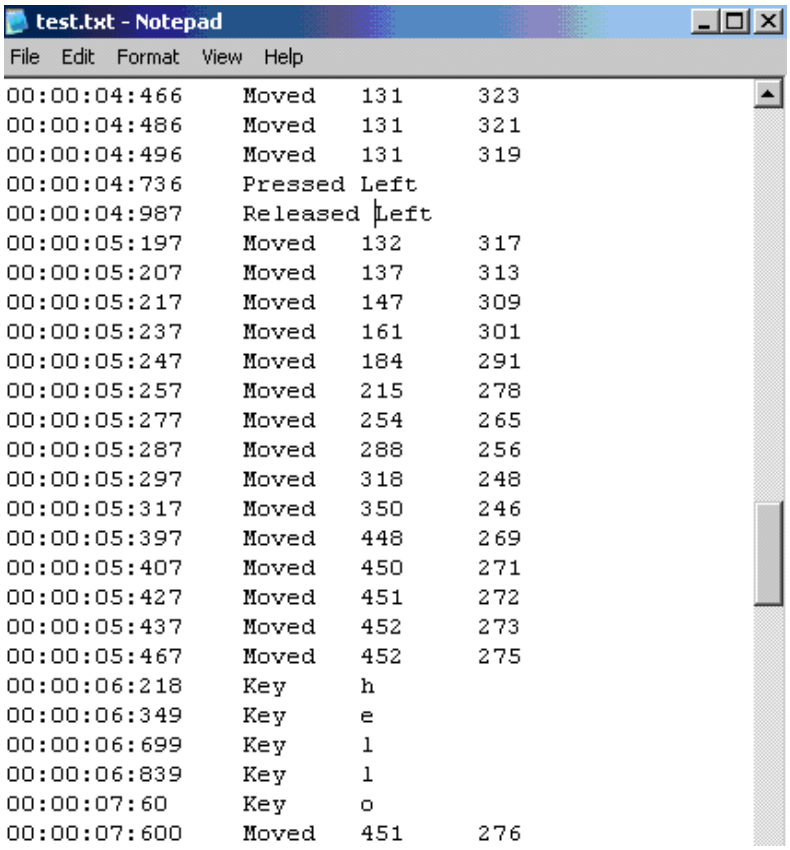

Figure 2.4 An example log of RUI showing the user moving the mouse, clicking, and then typing.

2.4 Method

Each subject was asked to do three tasks with the robot. The tasks were chosen so that users could both manually control the robot as well as program and run its behaviors. The tasks performed by human subjects were the same tasks that are performed by the cognitive model. Thus, while choosing the tasks, care had to be taken, so that they could be performed by a model developed in a currently existing architecture (Ritter, 2004). At the same time they should be interesting tasks.

Task 1

The subjects' first task was to manually control the robot. The subjects had to use the navigational arrows found in the bottom left corner of the interface to maneuver the robot to follow a red path, viewing the camera image on the screen, to find a plastic cup. They then had to position the robot so the cup was located inside the gripper arm, and press the gripper button to close it on the cup. Thus grasping the cup, they then returned to deliver the cup at the starting position, that was marked as "home". The subjects drove the robot along a "U" shaped path and is shown in Figure 2.5 .

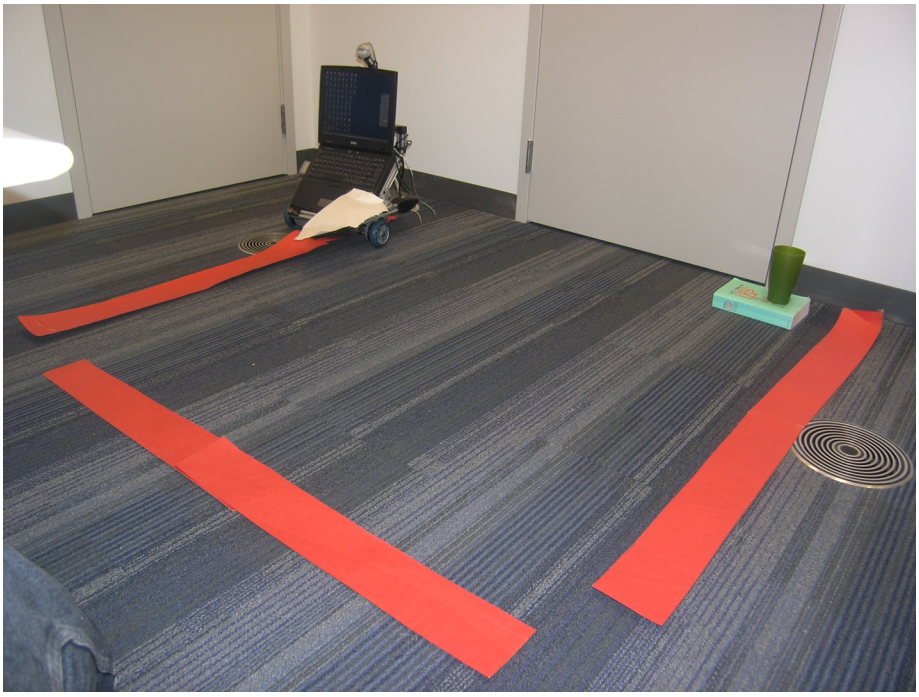

Figure 2.5 The path for the navigation task

To simulate different levels of experience and training, 10 users were allowed to watch the robot as the tasks were performed, 10 were not allowed to watch robot, but saw it beforehand, and 10 did not know what the robot looked like. These conditions are depicted in Figure 2.6.

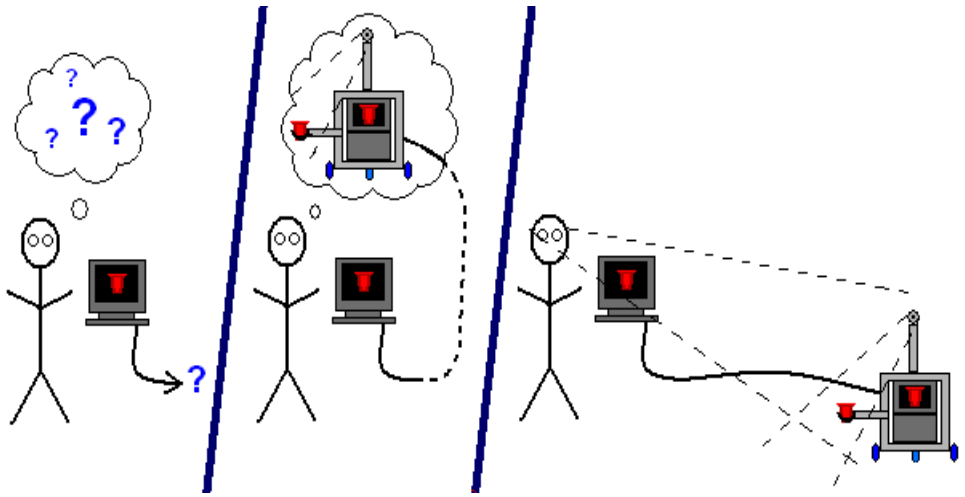

Figure 2.6 Some participants did not know what the robot looked like, others did know what it looked like, and still others could watch the robot while performing the tasks.

Five critical task types (Scholtz, 2003) have been identified in robot navigation tasks. These include local navigation, global navigation, victim identification, obstacle avoidance and vehicle state. The navigation task was designed in a way that simple versions of behavior across all these five could be studied. By having the robot follow a path, that was visible in the camera view, local navigation could be studied. Identifying and picking up the cup was to simulate victim detection and rescue. By dividing the

participants in the three groups described above, the influence of global navigation knowledge and vehicle state on task performance could be studied. To maintain simplicity of the task, obstacles were not placed in the path. However, if the subjects committed mistakes, and drove off the path, they encountered obstacles.

Task 2

The second task required participants to program the ER1 through its behaviors. The participants were asked to create a behavior so that when any object entered the robots gripper, the gripper would close. The program was then run to see if it was successful. The interface provides both the condition and action - 'object in gripper' and 'close gripper' as primitives.

Task 3

The third task required participants to program the ER1 interface to recognize a dollar bill. When the dollar bill is seen, the robot had to speak the phrase "I saw the dollar bill". The participants were asked to use an existing picture of the dollar. Again the program was tested to see if it was successful. The recognition could be programmed using mouse clicks. The action required mouse actions and typing.

2.5 Procedure

The participants were given instructions about each task they had to perform. Before the participant started the task, they were asked to fill out a pre-task questionnaire indicating how they felt about the task they were going to perform. For the navigation

task, the robot was placed in the adjacent room and depending on the group the subject belonged to, they were able to look into that room or not. After the navigation task, the robot was brought into the room where the subjects could see it and program it. After each task they were asked to fill out a Challenge Impressions questionnaire in which they stated how they assessed their own performance. At the end of the tasks they were made to fill out another post-task questionnaire stating how they felt after they had done the tasks. As the participants carried out their tasks, RUI was used to record their interactions with the interface. The subjects were paid, thanked, and debriefed.

Chapter 3

Experimental Results

Mouse activity and key stroke data was collected as the participants interacted with the robot using RUI. To analyze the data it was replayed and studied with analysis tools. The data was mainly studied to collect information on reaction times of the participants and the number of errors they committed while they performed the tasks. All thirty subjects completed the tasks.

3.1 Data from the navigation task

The analysis of the data helped to gain insight into how users drive and program a robot. Because the participants had different amounts of knowledge while they drove the robot, the effects of having a global view of the driving path and knowing the vehicle state were also studied.

The graph in Figure 3.1 shows the average driving times across the groups. Group1, “unseen”, was the group of people who had never seen the robot. Group 2, “previously seen”, was the group of people who knew what the robot looked like, and group 3, “visible” was the group that saw the robot while they performed the driving task. The mean time for the unseen group was 465.5 s, for the seen group was 370.5 s and for the visible group was 314.7 s.

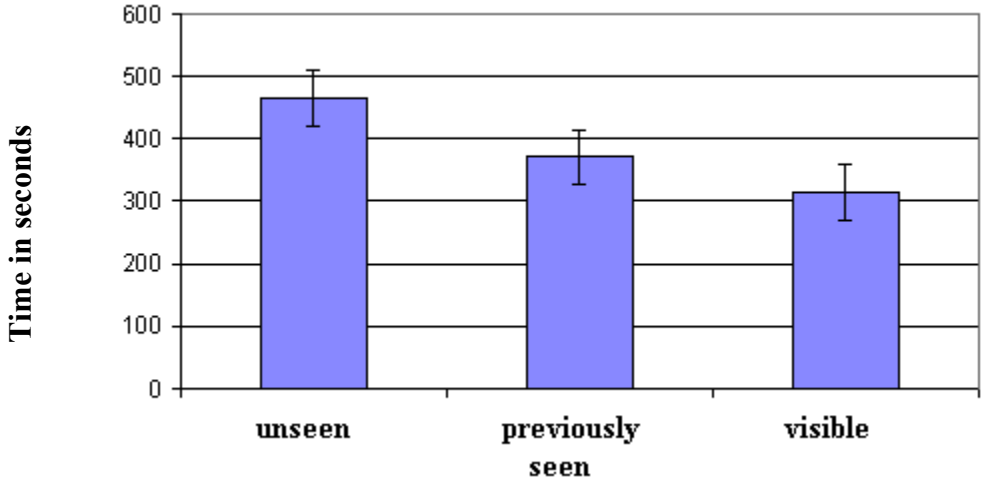

Figure 3.1 Mean driving times across groups with standard errors.

To test the statistical significance of the results the variance across the groups was analyzed (ANOVA) and it was found that the difference in the reaction times across the groups was in fact not significant ($F(1, 24) = 3.04$, $p < 0.05$). Further t-tests were conducted, two groups at a time and it was found that seeing the robot previously did not improve reaction times in a statistically significant way ($T(12, \text{two-tailed}) = 1.26$, $p<0.05$). However the reaction times across the visible and unseen group were significantly different ($T(12, \text{two-tailed}) = 2.54$, $p=0.1$).

It was seen that being co-located with the robot while navigating it along the marked path helped the participants by approximately 32 percent with the unseen group as a baseline. This was a little less than what was expected. There is a possible explanation of this.

While the study was conducted, an interesting observation was that the participants in “visible” group, who could see the robot and the path with the naked eye while performing the navigation, preferred observing through the camera view instead of

looking at the robot physically driving along the path. This means that the navigation task demanded a local view, which was available to all the three groups through the camera view, explaining their similar performance. The only time the subjects actually looked at the robot was when the victim (for the study, it was the cup) was detected and they steered the robot toward it. They also used their naked eye view, as against the camera view, when the robot was stuck due to an obstacle. The groups that did not see the robot, knew the robot was stuck due to an obstacle when they pushed the navigation buttons but the camera view remained the same. They also suspected an obstacle in the path when they heard scraping sounds of the robot motor and wheels. They then complained and needed assistance but could not themselves recover from these mishaps.

The actions of the participants performing each of the five task types previously discussed could be identified and studied from the dataset collected. While maneuvering to avoid obstacles in the path, the global view (visible group) and knowledge of the vehicle size and shape (previously seen group) was helpful and participants with access to this information performed better.

The graph in Figure 3.2 shows the average of the time spent by each group performing the operation of picking up the cup, once it has been identified.

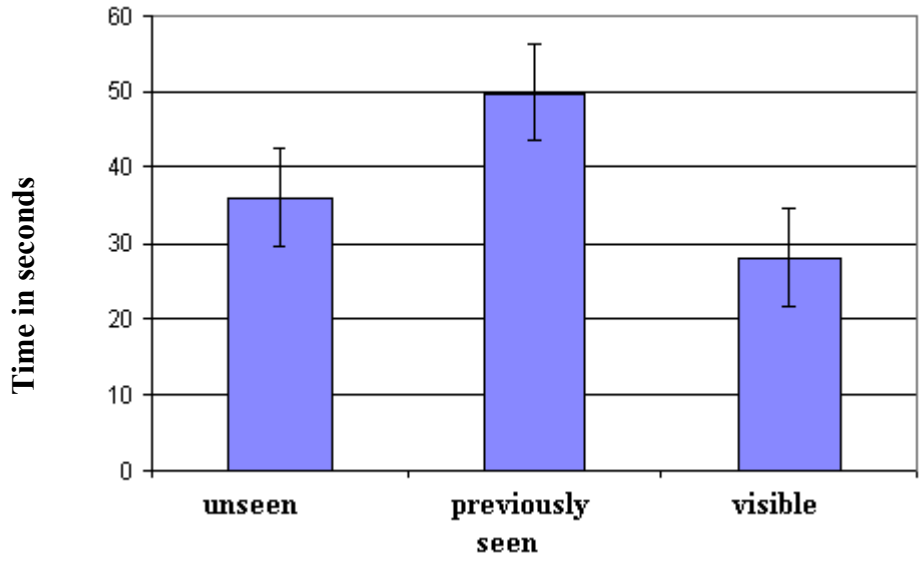

Figure 3.2 Mean time to pick up the cup once it had been identified, with standard errors bars.

The graph shows that having a global view helped the participants in the visible group who did much better than the participants in the unseen group. ANOVA tests show that the difference in the means is statistically significant ($F(1, 28) = 4.0$, $p<0.05$). The spike for the "previously seen" group appears as an anomaly. It may be a result of excessive knowledge or just a sampling error.

The graph in Figure 3.3 shows the division of the total navigation time, between the time to drive to the cup and the time to drive back from the cup. As seen from the graph, in each of the three groups, the participants took longer to drive to the cup than they took to drive back after picking the cup in the gripper. Also note, that the groups start out differently ($F(1,28) = 4.95$, $p < 0.01$), but on the way back, with practice, their performance appears more similar with time ($F(1, 28) = 0.56$, $p < 0.01$).

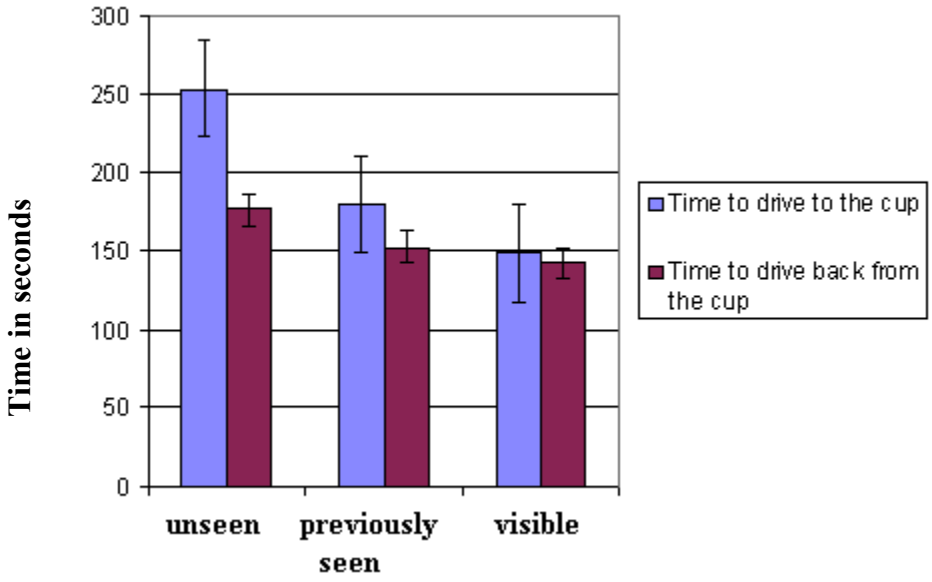

Figure 3.3 Mean time to drive to the cup and back with standard errors.

One reason for this improvement in time is that people create mental models (Johnson-Laird, 1998) of the task they execute and create a picture of the path in their mind while they are driving. For the participants who could see the robot as they drove the improvement in time is negligible, further confirming that the mental picture of the path created by the participants who could not see it while they drove, helped them to drive back sooner. The little improvement in the driving time for the visible group suggests that they were relatively well practiced in this type of task and were limited by the robot's speed (Ritter and Schooler, 2002) and there was in general, less to learn.

3.2 Data from programming tasks

Data collected from the programming tasks was analyzed to study mean times for each of the tasks and the number of errors committed. The subjects were grouped together because they all saw the same interface which these tasks used. It took the participants, on average 138 s to complete the gripper task and 171 s to complete the dollar task The errors the participants committed while executing these tasks were also studied. Any keystroke or mouse click that was not needed was counted as an error. To complete the tasks, the participants committed on average 7.6 errors on the gripper task and 8.7 errors while performing the dollar task.

The graphs in Figure 3.4 and Figure 3.5 show the time taken by each subject for each task and the errors committed in each task.

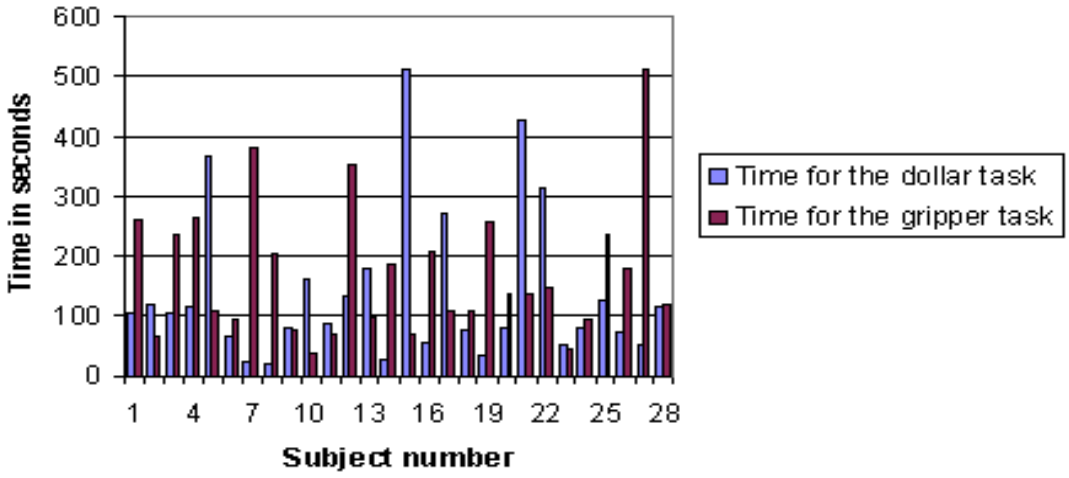

Figure 3.4 Time for the programming tasks per subject.

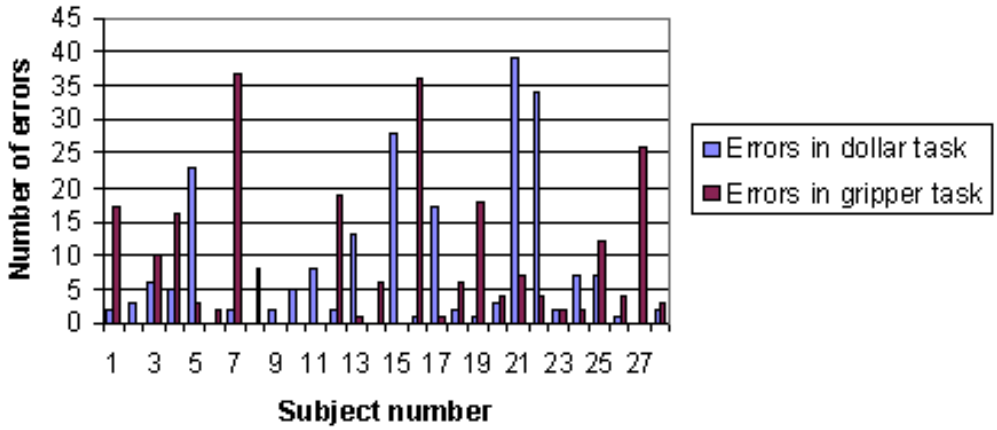

Figure 3.5 Errors committed during the programming tasks per subject.

As seen from the graph, the performance of the subjects in the dollar task appears to depend on their performance in the gripper task. The subjects who performed well (lesser time and fewer errors) in the gripper task, tended to do worse on the dollar task and vice versa. When the subjects committed errors on the gripper task, they examined primitives on the interface needed for the dollar task. They used this visual memory (Peterson, Kramer, Wang, Irwin and McCarley, 2001) to then quickly and correctly complete the dollar task. The participants who did well on the gripper task, spent time in visual search for the dollar task and hence took longer to complete the dollar task. Thus the effects of learning by exploration (Cox, 2001) were studied from the data collected. The graph in Figure 3.6 shows the number of errors committed in the dollar task versus the number of errors in the gripper task ($r = -0.3$).

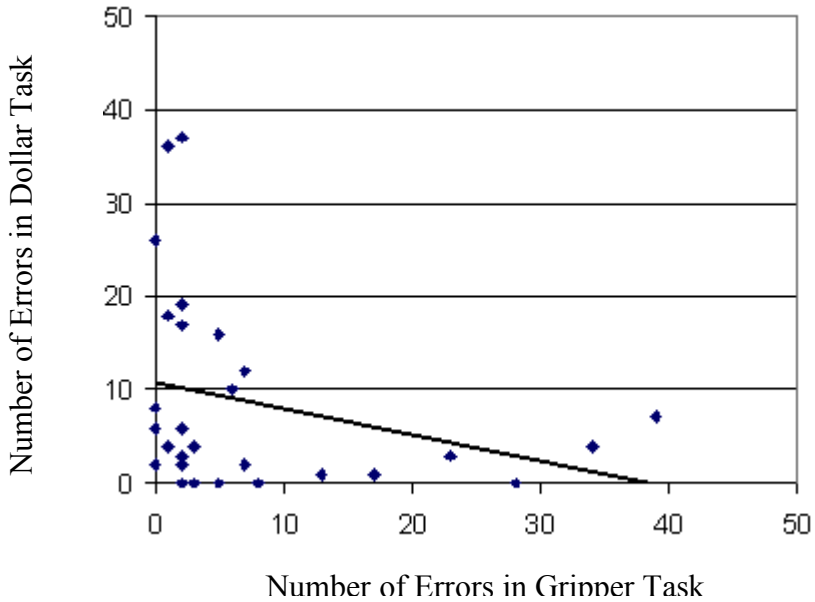

Figure 3.6 Dollar Task versus Gripper Task.

3.3 Performance on tasks – Revisited by Gender

While conducting the study, there were roughly equal number of males and females in each condition for the navigation task. The time taken to complete the navigation task, by gender is shown in the graph in Figure 3.7.

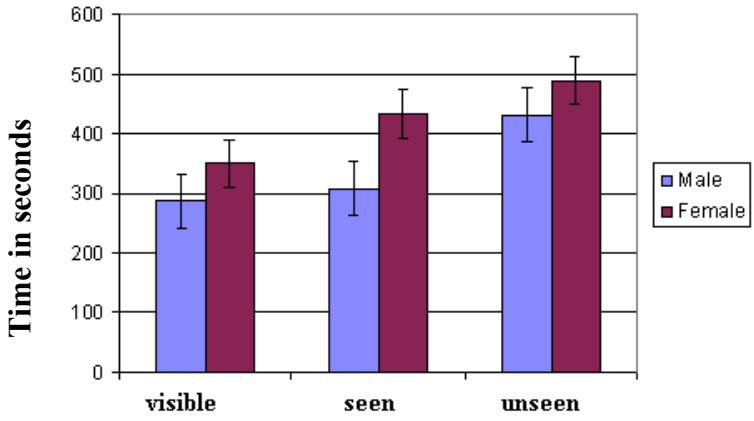

Figure 3.7 Navigation task across gender.

The graph in Figure 3.8 shows the performance by gender in the programming tasks.

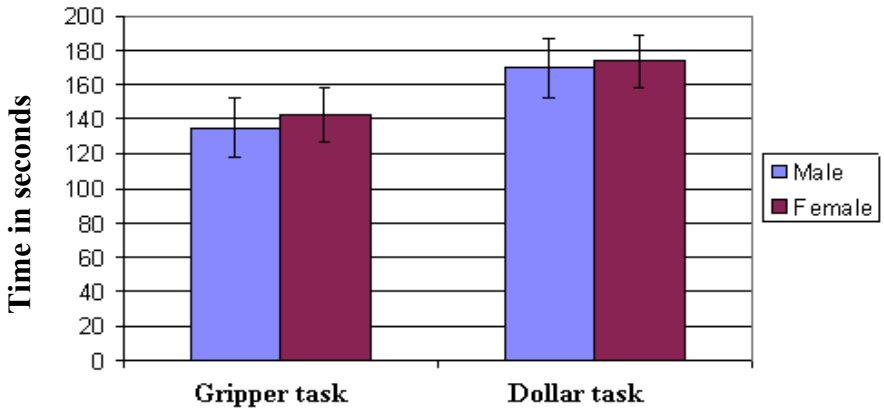

Figure 3.8 Programming tasks across gender.

In each graph it can be seen that the males took less time than the females. However, t-tests showed their performance is not reliably different.

3.4 Task Appraisal

Task appraisals were taken from 25 of the 30 subjects. The scores and standard deviations for the pre-task and post-task appraisals are shown in Table 3.1. The challenge impressions scores were taken after each task was performed. Table 3.2 shows the appraisal score and standard deviations for the navigation task. The scores are shown across each condition and by gender. Table 3.3 and Table 3.4 have the challenge impression data for the gripper and dollar tasks respectively. They show the means of the

scores and the standard deviations. The appraisal data was correlated with the subject performance. The correlation is shown in Table 3.5, Table 3.6, and Table 3.7.

Table 3.1 Pre-task and Post-task appraisal.

Appraisal Score	Pre-task (N=25)	Post-task (N=25)
Stressful (1-least, 5-most)	1.64 (0.8)	1.64 (0.7)
Cope (1-not well, 5-very well)	4.28 (0.9)	4.08 (0.7)
Demanding (1-least, 5-most)	2.16 (0.8)	2 (0.8)
Threatening (1-least, 5-most)	1.2 (0.4)	1.4 (0.4)
Perform (1-least well, 5-most well)	4.16 (0.7)	4.12 (0.6)

Table 3.2 Challenge Impressions for navigation task.

	Visible		Previously seen		Unseen	
Appraisal score	**M (5)**	**F (4)**	**M (4)**	**F (4)**	**M (3)**	**F (5)**
Difficult	1.6 (0.5)	1.5 (0.5)	1.75 (0.5)	1 (0)	1.3 (0.5)	1.4 (0.5)
Confident	4.6 (0.6)	4 (1)	4 (1.2)	4.25 (0.5)	5 (0)	4 (0.7)
Tried Hard	3.3 (1.7)	3.5(0.6)	3.5 (0.6)	4 (1.6)	4.3 (1.1)	2.6 (1.5)
Performed	1.3 (0.6)	2.5 (06)	2.5 (1.3)	2.5 (0.6)	1.3 (0.6)	2 (0.7)
Control	4.6 (0.6)	4 (1)	3.75 (1.5)	4.25 (1)	4.3 (0.6)	4.4 (0.5)
Experimenter's Evaluation	2(0.8)	2.5 (0.6)	2.75 (1.3)	1.5 (0.6)	1.3 (0.6)	2.2 (0.8)
Self-confident	4 (0.5)	3.25 (0.6)	3.25 (0.5)	3 (0)	4.3 (1.2)	4 (0.7)

Table 3.3 Challenge Impressions for gripper task.

Appraisal Score	Male (N=12)	Female (N=13)
Difficult	1.8 (1.1)	1.5 (1)
Confident	4.1 (0.7)	4.0 (0.8)
Tried Hard	3.6 (1.3)	3.3 (1.2)
Performed	2 (1.1)	1.9 (0.9)
Control	4.1 (1.1)	4.4 (0.7)
Experimenter's Evaluation	2.0 (0.9)	2.1 (1.2)

Table 3.4 Challenge Impressions for dollar task.

Appraisal Score	Male (N=12)	Female (N=13)
Difficult	2.16 (1.1)	2.08 (1.1)
Confident	4.25 (0.9)	3.6 (1.1)
Tried Hard	3.0 (0.9)	3.08 (1.5)
Performed	2.4 (1.2)	2.25 (1)
Control	3.8 (1.2)	3.8 (1.1)
Experimenter's Evaluation	2.3 (1.2)	2.25 (1.1)

Table 3.5 Correlation of the appraisal data to time the subject took to complete the navigation task

Appraisal Score	Time to do the task
Difficult	0.19
Confident	-0.32
Tried Hard	-0.35
Performed	0.45
Control	-0.22
Experimenter's Evaluation	0.31

Table 3.6 Correlation of the appraisal data to the subject performance in the gripper task.

Appraisal count	Number of Errors	Time
Difficult	0.55	0.69
Confident	-0.34	-0.38
Tried Hard	-0.46	-0.55
Performed	0.43	0.54
Control	-0.04	-0.11
Experimenter's Evaluation	0.3	0.39

Table 3.7 Correlation of the appraisal data to the subject performance in the dollar task.

Appraisal count	**Male**	**Female**
Difficult	2.16	2.08
Confident	4.25	3.6
Tried Hard	3.0	3.08
Performed	2.4	2.25
Control	3.8	3.8
Experimenters Evaluation	2.3	2.25

It was thus seen that the task was not a threatening or difficult task for this population.

There were no significant differences between the pre-task and post-task appraisal scores.

Also there were no significant differences across sex.

Chapter 4

The Model

A cognitive model is a theory of human cognition realized as a computer program. It provides a way of applying what is known from psychology to produce human-like behavior. The model deploys strategies to complete the task at hand, takes time to complete each task, commits errors, and learns.

Cognitive models have been applied in three main ways (John, 1998; Ritter, Baxter, Jones and Young, 2000) in human-computer interaction. Firstly, cognitive models can predict task performance times and errors in task execution. This helps to examine the efficiency of existing interfaces, and design better interfaces. The second way, models have been used are as embedded assistants to help users interact with their tasks such as cognitive tutors. The third way is as substitute users. These are used generally in synthetic environments like military simulations and video games.

4.1 Embodied Models

A major challenge in each of these uses of cognitive modeling in HCI has been the difficulty in connecting the cognitive model to the task environment. Recent developments in cognitive architectures and simulated “eyes” and “hands” have greatly improved the application of models in HCI.

In this work, user models are created for automatic interface evaluation for better design. Such tools already exist in certain design domains. For example, SPICE (Thorpe, 1992) contains a unifying theory of how electronic components work and uses this knowledge in automatic evaluation of circuits.

There exist some basic tools to analyze interfaces for simple properties like style guidelines. Some existing tools also use models to evaluate interfaces. These include GOMS and keystroke model to make predictions about user behavior. GLEAN (Kieras, Wood, Abotel and A.Hornof, 1995) has been used for applying models to interfaces. However the interface the model creates is not directly used by humans.

So far, to get a cognitive model to work with a task, the interface or task is modified to incorporate the model. For example, the models that were developed for the AMBR project (Gluck and Pew, 2001b) required that an API first be developed for data exchange between the simulation and the human performance model. Cognitive models that take inputs indirectly, rather than from the real interface, are limited by the accuracy of the simulation and/or specification. St. Amant and Reidel (2001) provide an example of stresses arising on users due to missing objects and object feature detection, the effects of which will be missing from the model. For example, multiple occurrences of an OK button on the screen may confuse users and they can distinguish between the right and wrong button based on visual context. If an interface description is provided to a model, the effects of these stresses will not be noticed.

Thus to truly evaluate interfaces, it is necessary for the model to interact with the interface in the same way that the human does.

Embodied cognitive models (Ritter and Young, 2001) are models that are built within a cognitive architecture and have simulated eyes and hands. A model created this way uses the same interface as the user and generates mouse clicks and key strokes. The figure below shows the relationship between the embodied model and the interface the user sees.

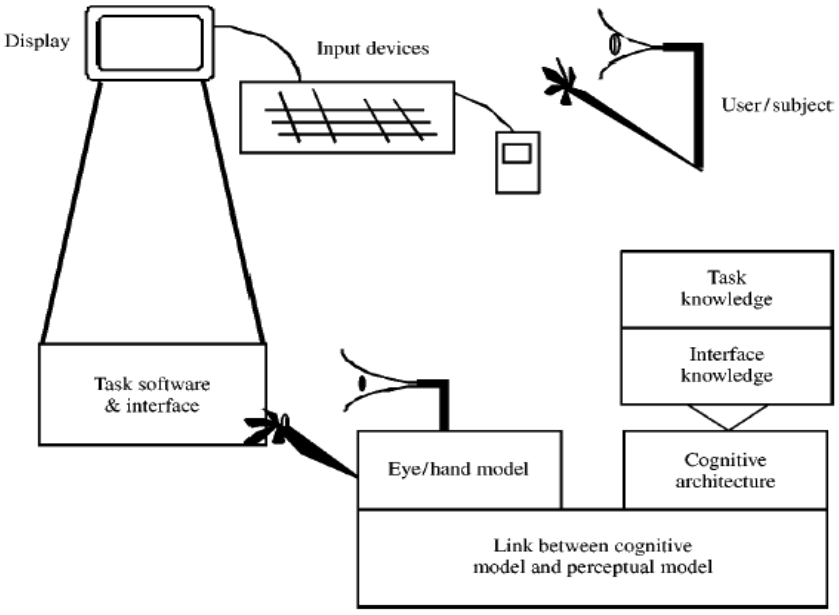

Figure 4.1 Components of a system for automatic evaluation (from Ritter, 2001).

4.2 The Architecture: ACT-R

Cognitive architectures (Newell, 1990; Ritter, 2004) are composed of cognitive mechanisms that are fixed across tasks and individuals. These generally include facts known from studies in psychology about perception and motor output, working memory and activation of declarative memory, and how to apply procedures. By adding knowledge to the architecture, various behaviors can be generated and the model can accomplish its task.

The ACT-R architecture (Anderson and Lebiere, 1998) is a production system theory of how human cognition works. A schematic of the current implementation of ACT-R is shown in Figure 4.2. ACT-R's main components are modules, buffers and a pattern matcher.

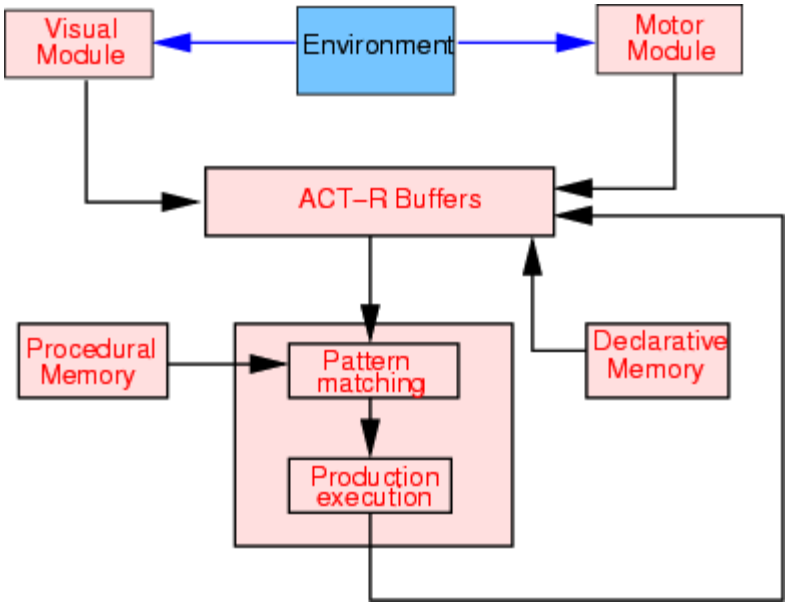

Figure 4.2 ACT-R 5.0 architecture (from act-r.psy.cmu.edu).

There are two memory modules in ACT-R: Declarative memory and Procedural memory. Declarative memory stores facts, like 2+3 =5, in the form of memory chunks. Procedural memory stores knowledge of things that we display in our behavior, like how to drive, as productions. ACT-R accesses its modules through its buffers. For each module, a dedicated buffer serves as an interface with that module.

The perceptual and motor modules provide “eyes” and “hands” to the model. With the perceptual module, the model can “look” at an interface and manipulate objects in that interface. Productions sent to the visual buffer can direct attention to an object on the screen and create a chunk in declarative memory that represents that object and its location on the screen. The production system can then send commands, initiated by a production rule, to the motor module that manipulates these objects.

The pattern matcher searches for a production that matches the current state of the buffers. Based on the production selected, an action is executed that may change the current goal and the state of the buffers. At one point, though several productions may match, ACT-R selects only one production to fire. This process of selecting which production to fire is called 'Conflict Resolution' and is decided based on the utility of the production v/s its cost measured over time.

An important aspect of the models created in the ACT-R system is that they predict human behavior including the timing of actions. Each step of cognition, like retrieving stored information or clicking the mouse, has a latency associated with it and these values are obtained from the psychological literature.

ACT-R seems to be a good choice for the architecture for the robot user project because it allows implementation of strategies and provides predictions for behavior. However, the Perceptual - Motor module of ACT-R can only work with interfaces developed in Macintosh Common Lisp. For the task at hand, the architecture hence needs to be enhanced with eyes and hands.

4.3 Eyes and Hands: SegMan

To enable the ACT-R architecture to interact with the robot interface, it was extended with SegMan (www.csc.ncsu.edu/faculty/stamant/cognitive-modeling.html).

Eyes

SegMan (Amant and O.Reidl, 2001) stands for Segmentation and Manipulation. It runs an image processing algorithm on a bitmap of the screen and creates a structured representation of the screen.

SegMan takes in pixel-level input from the screen and processes it to produce increasingly refined representations of the environment. At a higher level of processing, objects are recognized based on the information received at lower levels. For an object on the screen to be recognized, there must exist a template that describes how specific components should be present and their relative positions on the screen.

Certain patterns for features on windows interfaces for buttons and menus are saved, and if any are found, they are returned in the list. Patterns can be defined and if the pattern is found, it is returned upon request.

For example:
A pattern may be specified as :

```
define-pattern small-box ()
 (SEGMAN:DEFINE-PATTERN small-box ()
        (:AND (:COUNT 42) (:AREA 1) (:SIZE 42) (:HEIGHT 6) (:WIDTH 7)
        (:RED 212) (:GREEN 208) (:BLUE 200) (:COLOR 13947080)
        (:PROPORTION 5/6) 7 28 (31 5) 112 (124 4) 193 (199 4) (241 5)
        (255 20)))
```

A pattern is thus specified with respect to the *count* of pixels in the group; *size* which is the area of the group's bounding box; and the RGB values of color. When a request is made for all the patterns visible on the screen and if the “small-box” does exist on the screen, it will be present on the list of returned items.

Hands

SegMan can generate mouse and keyboard inputs to manipulate objects on the screen. With relevant information, the motor module can click buttons, select check boxes, pull down menus and so on. Functions such as (single-click()) and (double-click())

are implemented to simulate mouse-actions.(move-to-object) moves the mouse to the specified position and follows Fitts' law while doing so.

4.4 Cognitive Modeling Interface Management System (CMIMS)

With ACT-R and SegMan we thus have a Cognitive Modeling Interface Management System (CMIMS) for managing interactions of the model in a way analogous to the UIMS (User Interface Management System) (Ritter et al., 2000). SegMan uses its image processing algorithms to look at various parts of the screen and communicates this to ACT-R via the Lisp listener. ACT-R has the strategies and based on the knowledge provided to it, instructs SegMan to execute various actions on the screen. SegMan then uses its motor routines to perform the directed task.

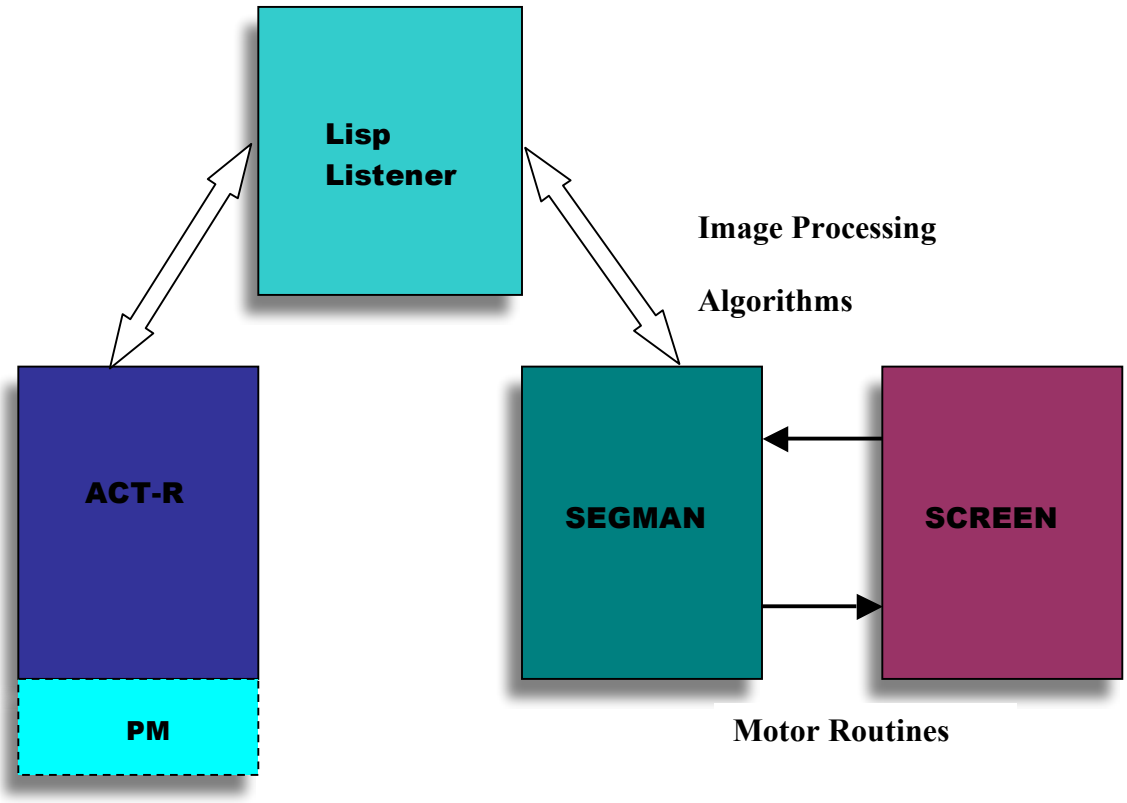

Figure 4.3 The CMIMS system used.

As seen in Figure 4.3, the perceptual and motor module of ACT-R are left out. Ideally it would have been nice if SegMan was used to assist PM instead of replacing it. However this, requires building visual features and modifying ACT-R's visual buffer and is a non-trivial task. However, the mouse and key press durations in SegMan are parameters adjustable by cognitive modelers. Here values that were used were obtained from the data of the user study. The models developed are explained next.

4.5 The Navigation Model

The model performed the same navigation task the human participants did. It steered the robot along the marked path that led it to the cup. It recognized the cup, closed the gripper on the cup, turned around and drove back along the same path to the spot marked "Home" where it dropped the cup.

Modifications to the Navigation Task

Perceptual processing with SegMan had limitations and the environment was slightly modified to accommodate them. The area the robot drove over was covered with white paper. The path was marked with black. Black and white are most apart on the RGB scale, and SegMan uses RGB values to identify patterns, so by modifying the environment in this way, the model was able to identify the path against the background using the ER1 camera. The black path was encoded as the pattern shown below:

```
(segman::define-pattern :black-path()
   (:AND(:RED 50 <) (:BLUE 50 < ) (:GREEN 50 <)))
```

The reason the path was not encoded with RGB value of zero (black) was because SegMan was extremely sensitive to lighting effects and distinguished across shades of black. Defining the pattern with this fuzziness (as against with a fixed count in the number of pixels, proportion, or area) makes it plausible with human vision where people do not distinguish between close gradients of color and intensity.

For perception, the model used a mixed vision system including SegMan as well as the vision system of the ER1 robot. The robot's vision system was able to correctly identify dollar bills from saved images. Taking advantage of this, a dollar bill

was placed at the target location. Once the robot's vision recognized the dollar, it showed a "recognition-cross" on it in the camera view. SegMan was able to identify the pattern for this recognition-cross, and when it was identified, it opened the robot gripper to pick up the cup that was placed just below the dollar bill. The robot's vision was also used to identify the "Home" location.

Thus, the robot used a mixed vision system, modeling how augmented vision systems can help users. The interface the model interacted with is shown in Figure 4.4.

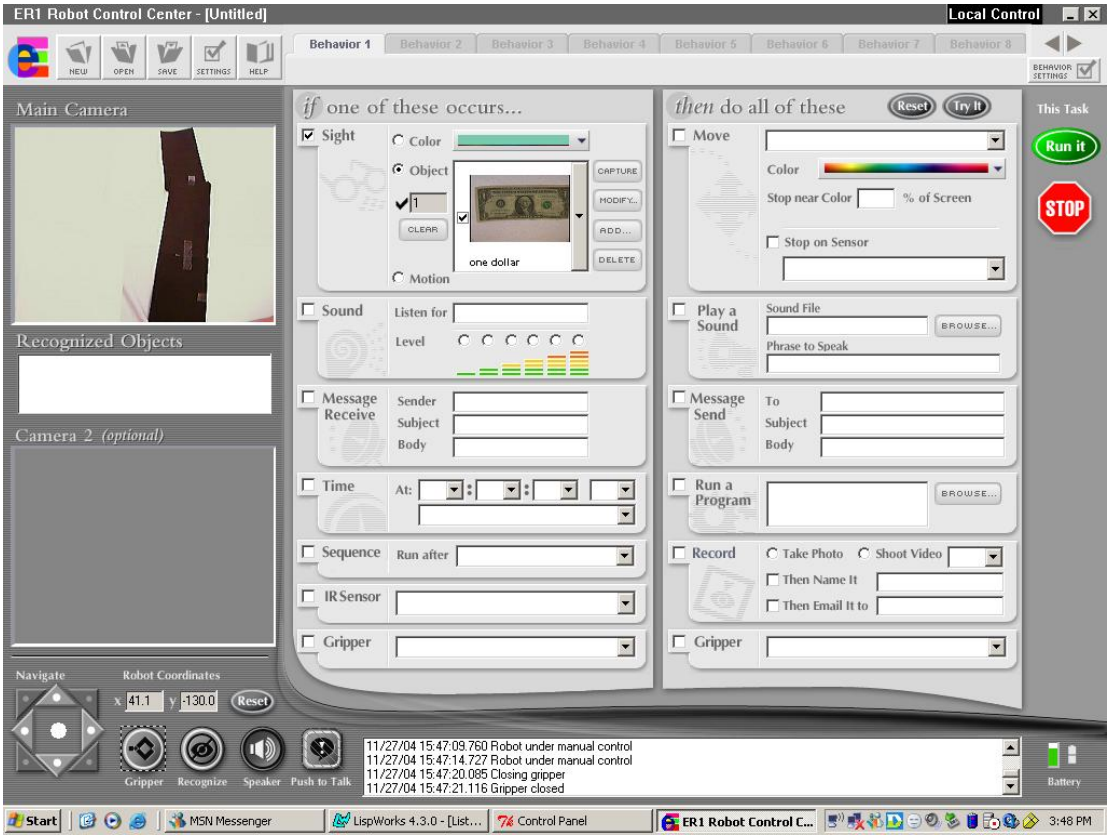

Figure 4.4 The ER1 robot interface, the camera view, of the black stripe to follow and navigation controls.

Navigation Strategy

The model looked iteratively at the camera view to see if the black path was present. When the pattern was found, its location with respect to the center of the camera view was noted. If the path was in (within an allowable threshold of) the center of the camera view, the mouse moved to the "ahead" navigation button. If the path was to the left or right of the camera view, the corresponding navigation arrow was selected.

If the black path was not found, the model performed visual search, randomly selecting one of the navigation buttons.

When the dollar was recognized, the mouse moved to the gripper control to open it. It then closed the gripper to grasp the cup and turned around. It then followed the same path until it recognized the place marked "Home" where it pushed the gripper button to drop the cup. A movie of the robot performing the task can be obtained at (www.acs.psu.edu/projects/HRI).

The model thus used a pseudo fovea looking alternately at the camera view and the navigation controls. Parsing the entire screen is a time consuming process (can take up to approximately a minute to parse a complex interface like the robot interface).By implementing this dual fovea, this time reduced to almost thirty-five percent of the time it took to parse the entire screen). A trace of the navigation task is shown in Table 4.1.

Table 4.1 A trace of the ACT-R model navigating the robot.

```
Time  0.000: Start Selected
Time  0.050: Start Fired
Time  0.100: Path-Found Retrieved
Time  0.100: Move Selected
Time  0.150: Move Fired
Time  0.200: Move-Straight Retrieved
Time  0.200: Move-Straight Selected
Time  0.250: Move-Straight Fired
Time  0.250: Start Selected
Time  0.300: Start Fired
Time  0.350: Path-Found Retrieved
Time  0.350: Move Selected
Time  0.400: Move Fired
Time  0.450: Move-Straight Retrieved
Time  0.450: Move-Straight Selected
Time  0.500: Move-Straight Fired
Time  0.500: Start Selected
Time  0.550: Start Fired
Time  0.600: Path-Found Retrieved
Time  0.600: Move Selected
Time  0.650: Move Fired
Time  0.700: Move-Straight Retrieved
Time  0.700: Move-Straight Selected
Time  0.750: Move-Straight Fired
Time  0.750: Start Selected
Time  0.800: Start Fired
Time  0.850: Path-Found Retrieved
Time  0.850: Move Selected
Time  0.900: Move Fired
Time  0.950: Move-Straight Retrieved
Time  0.950: Move-Straight Selected
Time  1.000: Move-Straight Fired
Time  1.000: Start Selected
Time  1.050: Start Fired
Time  1.100: Path-Found Retrieved
Time  1.100: Move Selected
Time  1.150: Move Fired
Time  1.200: Move-Right Retrieved
Time  1.200: Move-Right Selected
Time  1.250: Move-Right Fired
Time  1.250: Start Selected
```

4.6 Programming Models

The gripper and dollar task models were relatively simple and deterministic models. Patterns for the primitives on the ER1 interface, like gripper button, the run-it button, and others were saved as patterns in SegMan. To execute the task, SegMan looked-at the needed parts of the screen , and moved to the needed pattern and executed the task.

The trace for the model executing the gripper and dollar tasks are shown in Table 4.2 and Table 4.3 respectively.

Table 4.2 ACT- R model trace executing gripper task.

```
Time  0.000: Start Selected
SEGMENTED IF SIDE OF THE SCREEN LOOKING FOR THE WORD "GRIPPER">>>>>>
Time  0.050: Start Fired
 Time  0.050: Locate-Gripper Selected
 Time  0.100: Locate-Gripper Fired
 Time  0.100: Focus-If-Gripper Selected
 Time  0.150: Focus-If-Gripper Fired
 Time  0.150: Dropdown Selected
LOOKING IN THE DROP DOWN MENU FOR "OBJEC ENTERS">>>>
 Time  0.200: Dropdown Fired
 Time  0.200: Object Selected
 Time  0.250: Object Fired
 Time  0.250: Then side Selected
SEGMENTED THEN SIDE OF THE SCREEN LOOKING FOR THE WORD "GRIPPER">>>>
 Time  0.300: Then side Fired
 Time  0.300: Locate-Then-Gripper Selected
 Time  0.350: Locate-Then-Gripper Fired
 Time  0.350: Focus-Then-Gripper Selected
 Time  0.400: Focus-Then-Gripper Fired
 Time  0.400: Then-Dropdown Selected
LOOKING IN THE DROP DOWN MENU FOR "CLOSE GRIPPER">>>>>>>>>>>>>>>>
 Time  0.450: Then-Dropdown Fired
 Time  0.450: Close Selected
 Time  0.500: Close Fired
 Time  0.500: Run-It Selected
EXECUTING GRIPPER TASK>>>>>>>>>>>>
 Time  0.550: Run-It Fired
 Time  0.550: Checking for silent events.
```

The trace for the dollar task is shown below.

Table 4.3 Trace of the ACT- R model executing the dollar task.

```
Time  0.000: Start Selected

SEGMENTED IF SIDE OF THE SCREEN LOOKING FOR THE WORD "SIGHT">>>>>>
 Time  0.050: Start Fired
 Time  0.050: Sight Selected
LOOKING IN THE SIGHT MENU FOR THE WORD OBJECT
 Time  0.100: Sight Fired
 Time  0.100: Drop-Down Selected
LOOKING IN THE DROP-DOWN MENU FOR THE PICTURE OF ONE-DOLLAR BILL
 Time  0.150: Drop-Down Fired
 Time  0.150: Checkbox Selected
SEGMENTED THEN SIDE OF THE SCREEN LOOKING FOR THE WORD "SPEAK">>>>>>
 Time  0.200: Checkbox Fired
 Time  0.200: Then-Side Selected
TYPING>>>>>
 Time  0.250: Then-Side Fired
 Time  0.250: Type Selected
 Time  0.300: Type Fired
 Time  0.300: Execute Selected
EXECUTING DOLLAR TASK>>>>>>>
 Time  0.350: Execute Fired
 Time  0.350: Checking for silent events.
 Time  0.350: * Nothing to run:  No productions, no events.
```

Chapter 5

Fit of the Model to Human Data

The models were run and timing information collected. The model actions were compared to human data. The improvements to the model and its limitations are then discussed.

5.1 Navigation Model Data

The model executed the three tasks and data was collected using RUI, in exactly the same way that subject data was collected. The graph in Figure 5.1 shows a plot of times for four model runs and the average of the runs which was found to be 1029 s.

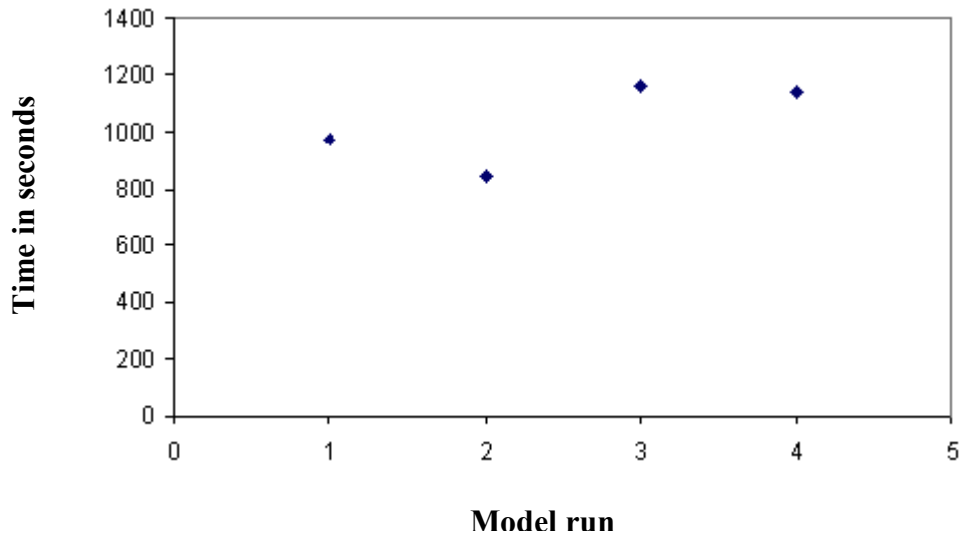

Figure 5.1 Time for the initial model to execute the navigation task.

5.2 Improvements to the navigation model

Following the task-based protocol analysis (Ritter and Larkin, 1994) , when human data was aligned with model data, it was found that the model took a much longer time in each run compared to humans. The reason for this was that the times between successive navigation actions in the model were greater than that in humans. Because of delayed actions, the camera view parsed to decide the course of navigation was also not the most updated and hence involved several course corrections. The reason for the delay was that while humans have some form of parallel processing in visual and motor processing (Gray et al., 1992; John and Gray, 1995), the model worked in a serial manner as described. It looked in the camera view, for the black path, calculated the steer direction based on the location of the path with respect to the center of the camera view. It then looked at the navigation controls and moved the mouse to the correct navigation arrow based on the direction calculated. Thus, it alternatively looked at the camera view and navigation control buttons, processing each visually in every decision cycle.

This is not exactly true with human behavior and humans tend to remember the location of the buttons once they have seen them.

The model was thus modified to correct this and the timing of the model improved by about 60 percent. Parsing the screen is a time consuming process in SegMan because of the vision processing algorithms used. Modifying the model in this way also helped the model to gain access to more current views of the path and hence better navigation. Thus the corrections to the model saved time of both, the repeated screen

parsing as well as repeated course corrections. The mean time for the revised model to complete the task was 401 s.

As seen in the graph in Figure 5.3 the model data compared very well with human data. The means across the three groups and the mean for the model data are plotted.

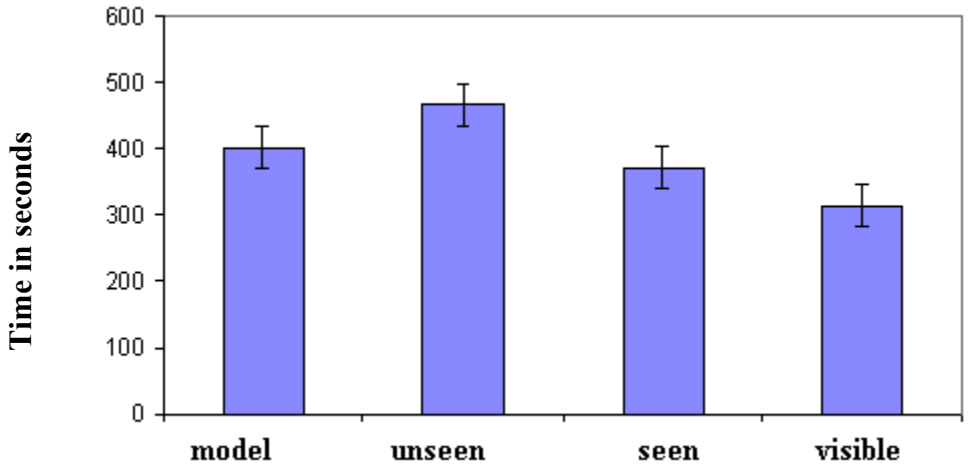

Figure 5.2 Human and revised model data for navigation task.

Statistical tests (t-tests) were conducted and it was found that the model data and human data were significantly different for the previously seen and the visible group. The model performance was in fact very similar to the performance of the group of people that had never seen the robot ($p<0.5$, 10 degrees of freedom, $t = 1.16$).

5.3 Programming Model Data

The gripper and dollar tasks were deterministic models and took the same amount of time in each run. The mean for the gripper task was 125 s and for the dollar task was

121 s. Figure 5.3 shows the model and human data. Here the model's performance is too fast, but of the same order of magnitude as the subjects. Act-r was used out of the box, and no changes were made to any of the parameters. By changing some of the settings, variance in the programming tasks could be seen.

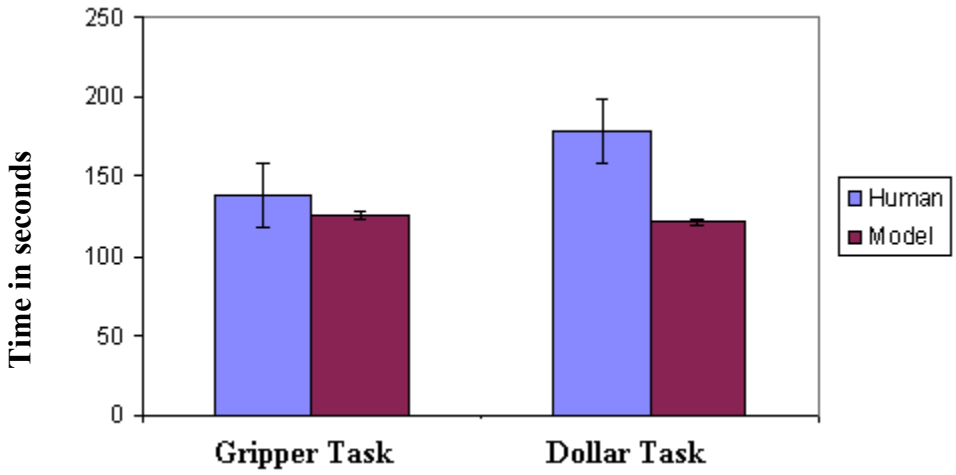

Figure 5.3 Human and model data for the programming tasks.

5.4 Limitations of the models

Though the model and human data compare well, there are some significant limitations with the model.

a) Controlled Environment

The navigation task where the robot, followed a marked path to the victim with almost no obstacles provided a well-controlled environment in which steering and lane-correction behavior of the model and humans could be studied. The model steered well across the path without unnecessary lane corrections which means that once the pattern is identified the model performed well. With improvements in SegMan with a robust way

to define patterns, the model could be extended to more complicated environments, but this remains an area of active research.

b) Perceptual Motor

Another limitation of the model is that it does not use the perceptual motor component of ACT-R and while SegMan was initially added to enhance the PM to interact with generic interfaces, it almost replaced PM. This means that several aspects of human vision like attention are not well accounted for in this model.

c) Learning

The model also differs with respect to humans in terms of learning**.** It was observed in human data collected that the participants took less time to drive back from the cup than they took to drive to the cup. This was because they created a mental picture of the path they drove through and could return quicker. There was no significant improvement in the time the model took to drive to and back from the cup.

d) Memory

As discussed earlier, visual search in humans has memory. If the participants steered too fast along turns and the path was no longer visible, they remembered the last seen location of the path and steered back till the path was visible again. The model just resorts to random search when the path is no longer visible. This can be fixed by making the model remember the location of each pattern it sees and use the last known location to find the driving direction when it cannot see the path.

e) Errors

In the programming tasks, the humans often commit errors in terms of wrong mouse clicks. The model makes no errors. By introducing noise in production activation

and imperfect matching of conditions (Lebiere, Anderson and Reder, 1994) the model can make errors and simulate human performance more closely.

Chapter 6

Implications and Conclusions

Though the ER-1 is an entry level robot and the model is limited in several ways, the study provides us with several useful lessons. The contributions of the study are discussed along three main domains: in the field of user modeling for interface design, robot interfaces, and robots in general.

6.1 Contributions to User Modeling for Interface Design

Cognitive modelers are continuously working to get simulated users work as closely in the environment that humans do. Here, we had a model interact directly with a robot interface. The contributions of this work for cognitive modelers are discussed next.

(a) Models can use robot interface directly

Enhancing ACT-R with SegMan has made it possible for models to interact directly with the interface, without modifying them in any way. These models of ACT-R and SegMan have previously been used to dial telephones (St. Amant, Horton and Ritter, 2004) and drive in a JAVA-based simulated game (Ritter et al., 2003). With this work we can see how it is becoming more and more possible for models to interact directly with fairly complex interfaces and across a range of task domains.

(b) Study human behavior

The model is developed with ACT-R as the cognitive architecture. By adjusting the amount of declarative and procedural knowledge, the model could simulate various

aspects of human behavior. The effects of anxiety and threat have been previously studied (Ritter, Reifers, Klein, Quigley and Schoelles, 2004). In USAR, the operators are often under stress, the effects of which can be studied with the model.

(c) Cognitive models as testers

The embodied model that was created, can see and interact with the interface in a way that humans do. The model can thus predict user reaction times and test the interface for usability and quality.

6.2 Contributions for Robots in USAR

Scholtz et. al.(2004) identified five task sub-types in the navigation task. The model was able to correctly predict that these sub-tasks are in fact different from each other and made predictions about human performance across each of the tasks.

(a) Difficulty in victim identification and rescue

It was difficult for the model to recognize the victim, and after it was identified, it was also difficult to maneuver to it. When the robot was not properly positioned, often the gripper would push the cup causing it to drop and become impossible to pick then. Both humans and the model faced this difficulty. It is known from literature in USAR (Murphy et al., 2000; Scholtz, Young, Drury and Yanco., 2004)that victim rescue is a hard task and the model was accurately able to predict that it is a hard task.

(b) Navigation and steering

The model helped in predicting various aspects of navigation. An earlier version of the model that was developed moved too fast and resulted in several unnecessary course corrections and often losing the path completely. Similar occurrences are also seen

in the data here as well as in USAR when if the operator turns too fast toward a location, the operator needs to perform a repetitive cycle of over-corrections with the robot (Murphy et al., 2000). The number of corrections varied with the parameter used to define the threshold for the center of the camera view simulating the difference in users who very strict about following the marked path and others who only roughly used it for guidance.

(c) Robot awareness

It was seen that the group that had previously seen the robot had an idea of vehicle state and this helped them when they performed the task. Also, an interesting observation was that the group of people who could not see the robot used the sound of the stuck robot motors and the not-changing camera view to realize the robot was stuck. They finally had to be assisted to get the robot away from the obstacle. If however there were multiple cameras that provided information about the vehicle state, they could have attempted to rescue themselves, and sound should be considered for future tele-operated robots.

6.3 Contributions for Robot Interfaces

HRI is different from HCI in many ways and special attention needs to be paid while designing robot interfaces. J.L.Drury, D.Hestand et. al (2004)(2004) laid down certain design guidelines for better human-robot interaction. The model predictions on how humans use robot interfaces provide ways in which robot interfaces can be improved for better interaction.

(a) Eye-Hand Co-ordination

A previous version of the model that alternatively looked at the camera view and the navigation controls and then moved to the appropriate navigation button took extremely long to complete the task because the camera view it processed did not reflect the most current position of the robot in the environment. The model thus helped to correctly predict that robot interaction tasks require high vision and motor co-ordination. A way to improve the interface would thus be to have the navigation controls and the camera view close to each other on the interface avoiding repeated shifts in attention.

(b) Global View

It was also seen that participants that had a global view of the area around which the robot drove were able to avoid obstacles and grasp the cup quickly and correctly. The model did not have access to this information and if it got stuck on an obstacle in the path, the robot needed assistance and the run was terminated. If however there were controls on the interface to adjust the camera view, the model could avoid obstacles if previous knowledge of obstacle patterns was loaded in the model.

References

Amant, R. S. and M. O.Reidel (2001). "A perception-action substrate for cognitive modeling in HCI." *International Journal of Human-Computer Studies* **55**: 15-39.

Anderson, J. R. and C. Lebiere (1998). *The atomic components of thought*, Mahwah, NJ: Lawrence Erlbaum.

Byrne, M. D. (2001). "ACT-R/PM and menu Selection: Applying a Cognitive Architecture to HCI." *International Journal Human-Computer Studies* **55**: 41-84.

Byrne, M. D. and W. D. Gray (2003). "Returning Human Factors to an Engineering Discipline:Expanding the Science Base through a New Generation of Quantitative Methods — Preface to the Special Section." *Human Factors* **45**(1): 1-4.

Casper, J. L. and R. R. Murphy (2001). Workflow Study on Human-Robot Interaction in USAR. *International Conference on Robotics and Automation*.

Cox, A. L. (2001). What people learn from exploratory device learning. Fourth International Conference on Cognitive Modeling.

Emond, B. and R. L. West (2003). "Cyberpsychology: A human-interaction perspective based on cognitive modeling." Cyberpsychology and Behavior **6**(5): 527-536.

Ericsson, K. A. and H. A. Simon (1984). Protocol Analysis: Verbal Reports as Data. Cambridge, MA, MIT Press.

Frank E. Ritter, D. V. R., Robert St. Amant (2002). "A user Modelling Design Tool Based On A Cognitive Architecture For Comparing Interfaces." *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CASUI'2002*: 111-118.

Freed, M. and R. Remington (2000). Making human machine system simulation a practical engineering tool: an APEX overview. 3rd International Conference on Cognitive Modelling,, Universal Press.

Gluck, K. A. and R. W. Pew (2001b). Overview of the agent-based modeling and behavior representation (AMBR) model comparison project. 10th Computer Generated Forces and Behavioral Representation Conference. 10TH-CGF-066. 3-6., Orlando, FL: Division of Continuing Education, University of Central Florida.

Gray, W. D., B. E. John, et al. (1992). The precis of Project Ernestine or an overview of a validation of GOMS. SIGCHI conference on Human factors in computing systems.

J.L. Drury , J. S., H. A. Yanco (2004) "Applying CSCW and HCI Techniques to Human-Robot Interaction. "

J.L. Drury, D. Hestand, et al. (2004). "Design Guidelines for improved human-robot interaction." CHI Extended Abstracts: 1540.

John, B. E. (1998). "Cognitive modeling in human computer interaction."Graphics Interface: 161-167.

John, B. E. and W. D. Gray (1995). CPM-GOMS: an analysis method for tasks with parallel activities. Conference on Human Factors in Computing Systems.

Johnson-Laird, P. N., Girotto, V., & Legrenzi, (1998). Mental models: A gentle guide for outsiders.

Kawamura, K., T. E. Rogers, et al. (2002). Development of a Cognitive Model of Humans in a Multi-Agent Framework for Human-Robot Interaction. International joint conference on Autonomous agents and multiagent systems.

Kazuhiko Kawamura, T. E. R., Kimberly A. Hambuchen (2002). "ISAC Humanoid: Test Bed in Human-Robot Teaming." *International Manufacturing Automation Leaders Forum*.

Kieras, D. E., S. D. Wood, et al. (1995). "GLEAN: a computer-based tool for rapid GOMS model usability evaluation of user interface designs." ACM Symposium on User Interface Software and Technology: 91-100.

Kortenkamp, D. and G. Dorais (2000). Tutorial : Designing Human Centred Autonomous Agents, PRICAI Pacific Rim International Conference on Artificial Intelligence.

Kukreja, U. and F. E. Ritter (submitted November, 2004). "RUI-Recording User Input from interfaces under Windows." *Behavior Research Methods, Instruments, and Computers.*

Lebiere, C., J. R. Anderson, et al. (1994). Error modeling in the ACT-R production system. Sixteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Erlbaum.

Murphy, R. R., J. L. Casper, et al. (2000). Mixed-Initiative Control of Multiple Heterogeneous Robots for Urban Search and Rescue. *Center for Robot Assisted Search and Rescue (CRASAR) Technical Reports*. **11**.

Newell, A. (1990). Unified Theories of Cognition, Cambridge, MA: Harvard University Press.

Peterson, M. S., A. F. Kramer, et al. (2001). "Visual search has memory." *Psychological Science* **12**.

Ritter, F. E. (2004). "Choosing and getting started with a cognitive architecture to test and use human-machine interfaces." *MMI-Interaktiv-Journal*: 17-37.

Ritter, F. E., G. Baxter, et al. (2000). "Supporting Cognitive Models as Users." *ACM Transactions on Human-Computer Interaction* **7**(2): 141-173.

Ritter, F. E., E. Churchill, et al. (2004). "Introduction to the User: The ABCS of User Interface Design."

Ritter, F. E. and J. H. Larkin (1994). "Developing Process Models as Summaries of HCI Action Sequences." *Human-Computer Interaction* **9**: 345-383.

Ritter, F. E., A. Reifers, et al. (2004). Using cognitive modeling to study behavior moderators: Pre-task appraisal and anxiety. Human Factors and Ergonomics Society.

Ritter, F. E., Dirk Van Rooy, et al. (2003). "Providing User Models With Direct Access To Computer Interfaces: An Exploratory Study Of A Simple Human-Robot Interface." *IEEE Systems, Man, and Cybernetics: Part A, Systems and Humans*.

Ritter, F. E. and L. J. Schooler (2002). The learning curve. In International encyclopedia of the social and behavioral sciences. Amsterdam: Pergamon.**:** 8602-8605.

Ritter, F. E. and R. M. Young (2001). "Embodied Models as Simulated Users: Introduction to this Special Issue on Using Cognitive Models to Improve Interface Design." *International Journal Human-Computer Studies* **55**: 1-14.

Robert St. Amant, T. E. Horton., Frank E. Ritter (2004). "Model-based Evaluation of Cell Phone Menu Interaction." *Proceedings of the ACM Conference on Human*

Factors in Computing Systems.

Scholtz, J. (2003). Human-robot Interactions: Creating Synergistic Cyberforces. Hawaii International Conference on System Science,.

Scholtz, J., J. Young, et al. (2004). Evaluation of Human-Robot Interaction Awareness in Search and Rescue. submitted ICRA.

St. Amant, R., T. E. Horton, et al. (2004). Model-based evaluation of cell phone menu interaction. In Proceedings of the. Conference on Human Factors in Computer Systems., New York, NY, ACM.

St. Amant, R. and F. E. Ritter (2005). "Specifying ACT-R models of user interaction with a GOMS language." Cognitive Systems Research **6**(1): 71-88.

Thorpe, T. W. (1992). Computerized Circuit Analysis with SPICE:A Complete Guide to SPICE with Applications, New York, NY: Wiley.

Trewin, S. (1998). "InputLogger: General -purpose logging of keyboard and mouse events on an Apple Macintosh." *Behavior Research Methods, Instruments and Computers* **30**(2): 327-331.

Westerman, S. J., S. Hambly, et al. (1996). "Investigating the Human-Computer Interface using the Datalogger." *Behavior Research Methods, Instruments and Computers* **28**(4): 603-606.