

Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation

Terrence Fong

CMU-RI-TR-01-34

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213

November 2001

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

Thesis Committee
Charles Thorpe (chair)
Charles Baur
Eric Krotkov
Chris Atkeson

© 2001 by Terrence Fong.
All rights reserved.

This research was partially funded by grants from the DARPA ITO “Mobile Autonomous Robot Software” and TTO “Tactical Mobile Robots” (NASA Jet Propulsion Laboratory 1200008) programs, the National Science Foundation (Dissertation Enhancement Award 1120030), and SAIC. The Institut de Systèmes Robotiques (DMT-ISR) of the Ecole Polytechnique Fédérale de Lausanne (EPFL) served as host institution (institution d’accueil) for this thesis and provided research facilities and infrastructure.

Abstract

Telerobotic systems have traditionally been designed and solely operated from a human point of view. Though this approach suffices for some domains, it is sub-optimal for tasks such as operating multiple vehicles or controlling planetary rovers. Thus, I believe it is worthwhile to examine a new system model for teleoperation: *collaborative control*.

In collaborative control, a human and a robot *collaborate* to perform tasks and to achieve common goals. Instead of a supervisor dictating to a subordinate, the human and the robot engage in *dialogue* to exchange information, to ask questions, and to resolve differences. Instead of serving the human as a mere tool, the robot can operate more like a partner. With this approach, the robot has more freedom in execution and is more likely to find good solutions when there are problems.

With collaborative control, the human is able to function as a resource for the robot, providing information and processing just like other system modules. In particular, the robot can ask the human questions as it works, to obtain assistance with cognition and perception during task execution. This enables the human to compensate for inadequacies of autonomy, but does not require time-critical nor situation-critical response.

Collaborative control is both a novel and a useful paradigm for teleoperation. Collaborative control is novel because it uses dialogue as a framework for coordination, to direct joint task performance and to focus attention where it is needed. Collaborative control is useful because it provides an efficient mechanism for adaptation, to adjust autonomy and human-robot interaction to fit situational needs and user capabilities.

This thesis investigates the numerous human-robot interaction and system design issues raised by collaborative control, describes the implementation of a dialogue-based user interface and control architecture, and examines the use of this new approach in several vehicle teleoperation applications and a user study.

Keywords: collaborative control, human-robot collaboration, human-robot communication, human-robot interaction, man-machine systems, mobile robots, PDA interface, remote driving, robot control architecture, safeguarded teleoperation, vehicle teleoperation.

For Mom and Dad

Acknowledgements

When I started the Robotics Ph.D. program in August 1994, I had no idea the road to this thesis would be so long and winding. In fact, when Jessica and I arrived in Pittsburgh, we both thought, “We’ll be back in California before the year 2000.” Even two years later, I was sure that I would finish my research before Navlab II was retired. But then one day, Jessica said, “I think I want to join the High-energy (Physics) group. Do you want to move to Switzerland?”

So, we packed our bags and moved across the Atlantic. A few months later, after having settled in at EPFL, I caught myself saying, “It’s really not so hard to understand. I’m a CMU student, but I’m doing my research in Switzerland. I have two advisors named Charles and I’m being paid by a start-up company in California.” At that point, I realized that this was probably not the usual (or easy) route to a Ph.D. But having now reached the end, I must say that the journey has been memorable and rich, and I am truly glad for having traveled it.

There are many people to whom I would like to express my appreciation and gratitude. First and foremost, I would like to thank Dr. Charles (Chuck) Thorpe and Dr. Charles Baur for their guidance, encouragement, patience, and wisdom during the past few years. It has been a great privilege to work with such far thinking and inspirational individuals. I am particularly indebted to Chuck for allowing me to follow such an unconventional path and to Charles for his unwavering support along the way. All that I have learned and all that I have accomplished during the course of this thesis would not have been possible without their dedication.

To the community that is the Robotics Institute, my thanks go to: Marie Elm, Jim “Fraz” Frazier, Ruth Gaus, Martial Hebert, Suzanne Lyons Muth, Dorothy “Dot” Marsh, Matt Mason, Stephanie Riso, Sandy Rocco, Red Whittaker, and Marce Zaragoza. To the Navlab Group (Al, Barry, Dirk, Jennie, Julio, Kate, Jim), thanks for making the Slag Heap “enjoyable”. To Robograds ‘94 (Andy, Daniel, Dong Mei, Henry, Lisa, Murali, Patrick, Sundar, Zack), thanks for making Pittsburgh fun.

To my fellow Planetes (Butler, Erik, Laurent, Mike, and Phil), thanks for being so flexible and supportive even when I was on the other side of the world. Fourth Planet was a great experience, a joy, a family. I will forever remember our telecons, meetings, and lunches. I hope that we will soon have the opportunity to all work together again (and that I will once more have the pleasure of amusing you all with my choice of “breakfast”).

Un grand merci à tous ceux à l'EPFL qui m'ont aidé. Je suis particulièrement reconnaissant: au Docteur Reymond Clavel qui m'a accueilli au sein de l'ISR; à Mme Heidi Banz, à Mlle Karine Genoud, à Mme Marie-Jose Pellaud et à Mme Katrin Schleiss pour leur assistance avec tous les tâches administratives grandes et petites; et à Mme Nadia Nibbio (CFRC) qui a gracieusement financé quelques voyages aux Etats-Unis.

Je remercie tous les étudiants qui ont travaillé avec moi (Gilbert Bouzeid, Mathias Deschler, Roger Meier, et Grégoire Terrien). Bon travail! Je remercie chaleureusement le Groupe VRAI (Didier, François, Gaëtan, Nicolas, Olivier, Patrice, Philippe et Sébastien) grâce à qui ce doctorat a put être réalisé dans une ambiance amicale et motivante. A Lorenzo et à Cécile, merci pour votre amitié, votre générosité, et pour tout le reste.

I wish to thank my family for their love and for encouraging me to follow my dreams. To Tammy and Tim, thanks for always being there. To Mom and Dad, thanks for caring, for believing, and for supporting me all these years, no matter how far away I have been. To Karina, un très gros bisou de ton papa.

Finally, to Jessica, thanks for your hugs, your kisses, your laughter, your smile and your love. You make it all worthwhile.

Lausanne, Switzerland
November 2001

Contents

1	Introduction	1
1.1	Robot as Tool	1
1.2	Human as Controller	2
1.3	Collaborative Control	2
1.4	Why Collaborative Control?	5
1.5	Related Research	7
1.5.1	Supervisory Control	7
1.5.2	Cooperative Teleoperation	8
1.5.3	Human Robot Control Architectures	9
1.5.4	Adjustable Autonomy and Mixed Initiative	9
1.5.5	KTH Collaborative Human-Robot Systems	10
1.6	Thesis Organization	11
2	Developing Collaborative Control	13
2.1	Vehicle Teleoperation	13
2.2	Conventional System Models	14
2.2.1	Direct Control	15
2.2.2	Supervisory Control	16
2.2.3	Fully Autonomous Control	17
2.2.4	Limitations	18
2.3	Collaborative Control System Model	19
2.4	Key Design Issues	20
2.4.1	Dialogue	20
2.4.2	Awareness	20
2.4.3	Self-reliance	21
2.4.4	Adaptiveness	21
2.4.5	Human-Robot Interaction	22
2.4.6	User Interface Design	22
2.4.7	Control and Data Flow	23

2.5	Evaluation.	23
2.6	Summary	24
3	System Design	25
3.1	Design.	25
3.1.1	Principles.	25
3.1.2	Approach.	26
3.1.3	Architecture.	26
3.2	UserModeller	29
3.2.1	Attributes.	29
3.2.2	Definition.	29
3.2.3	Use.	30
3.3	QueryManager	31
3.4	EventLogger	32
3.5	Human-Robot Dialogue	33
3.6	Background	36
3.6.1	Human-Robot Interaction	36
3.6.2	Collaboration Systems.	37
3.6.3	Human-Computer Dialogue	38
3.7	Summary	40
4	PdaDriver: A User Interface for Collaborative Control	41
4.1	Requirements	41
4.1.1	Collaborative Control Needs.	42
4.1.2	Vehicle Teleoperation Needs.	42
4.1.3	Application Needs.	43
4.1.4	Requirements Summary	43
4.2	Approach	44
4.3	Architecture	45
4.4	Dialogue Displays	46
4.4.1	Robot to User	46
4.4.2	User to Robot	46
4.4.3	Event Replay.	47
4.5	Command Modes	48
4.5.1	Direct Mode.	49
4.5.2	Image Mode.	49
4.5.3	Map Mode	52
4.5.4	Sensor Mode	52
4.6	A Brief Review of Interface Design.	53
4.6.1	Design Factors.	54

4.6.2	Design Techniques	54
4.6.3	Interface Control Mechanisms	55
4.6.4	Common Design Problems	56
4.6.5	HCI Design and Robotics.	57
4.7	Summary	58
5	Safeguarded Robot Controller	59
5.1	Requirements.	59
5.2	Robot Hardware	61
5.3	Controller Design	63
5.4	Modules.	66
5.4.1	AudioServer	66
5.4.2	CameraManager	66
5.4.3	ImageServer	66
5.4.4	Localizer.	67
5.4.5	MapMaker	67
5.4.6	MapServer	71
5.4.7	MotionController	72
5.4.8	Safeguarder.	72
5.4.9	Sensor Modules	73
5.4.10	Task Modules	74
5.4.11	UIGateway	74
5.5	Related Work.	75
5.5.1	Safeguarded Teleoperation.	75
5.5.2	Control Systems for Teleoperation.	75
5.6	Summary	75
6	Applications	77
6.1	“A to B”.	77
6.2	Collaborative Exploration	79
6.2.1	Motivation	80
6.2.2	RockFinder.	80
6.2.3	Human-Robot Interaction.	81
6.2.4	Related Work	82
6.3	Multi-robot remote driving	83
6.3.1	Motivation	83
6.3.2	MotionDetector	84
6.3.3	Remote Driving Tests.	85
6.4	Summary	89

7	User Study	91
7.1	Evaluation.	91
7.2	Contextual Inquiry	92
7.3	Study Objectives.	93
7.4	Test Summary.	93
7.4.1	Test Procedure	94
7.4.2	Participants	95
7.5	Results	97
7.5.1	Using the System.	97
7.5.2	Answering Questions	99
7.5.3	Dialogue and Human-Robot Interaction	103
7.5.4	Design Suggestions	105
7.6	Summary	107
8	Conclusion	109
8.1	Discussion	109
8.1.1	Human-Robot Collaboration	109
8.1.2	Benefits of Collaborative Control	111
8.1.3	Limitations	112
8.2	Future Work	112
8.2.1	Improvements	112
8.2.2	Scaleability	113
8.2.3	Additional Evaluation	114
8.2.4	Unanswered Questions	117
8.3	Summary	118
8.3.1	System Review	118
8.3.2	Main results	119
8.3.3	Contributions.	120
8.3.4	Final Words	120
	Appendix A: Vehicle Teleoperation	121
A.1	Vehicle Teleoperation.	121
A.2	Operator Interfaces	122
A.3	History of Vehicle Teleoperation	123
A.3.1	Air Vehicles.	123
A.3.2	Ground Vehicles	125
A.3.3	Underwater Vehicles	127
A.4	Vehicle Teleoperation Interfaces	129
A.4.1	Direct Interfaces	129
A.4.2	Multimodal and Multisensor Interfaces	130

A.4.3	Supervisory Control Interfaces	131
A.4.4	Novel Interfaces	134
A.5	Summary	135
Appendix B: Toolkits		137
B.1	FPC	137
B.1.1	Overview	137
B.1.2	The FPC Communications Model	138
B.1.3	Clients	140
B.1.4	Message Data	141
B.1.5	Programming with FPC	142
B.1.6	Performance	144
B.1.7	FPC Open Source License	146
B.2	Vislib	147
B.2.1	Description	147
B.2.2	Requirements	147
B.2.3	Example: Motion Detection	148
B.2.4	Example: Object Detection	149
B.2.5	VisLib Open Source License	150
Appendix C: Contextual Inquiry Study		151
C.1	CMU Human Subjects Clearance Request	151
C.1.1	Cover Page	152
C.1.2	Proposal	153
C.1.3	How Subjects Will Be Used	154
C.1.4	Consent Form	156
C.1.5	Confidentiality	157
C.1.6	Authorization for Access to Subjects	157
C.1.7	Risk/Benefit Analysis	159
C.1.8	NIH Training Certificate	160
C.2	Contextual Inquiry Study Plan	161
C.2.1	Focus	161
C.2.2	Objectives	161
C.2.3	Approach	161
C.2.4	Study Procedure	162
C.3	User Questionnaire	164
C.4	Quick Reference Sheets	165
References		167

List of Figures

Figure 1. Vehicle teleoperation using collaborative control: Left, robot encounters a potential obstacle; center, query to the human: “Can I drive through?”; right, collaboration result.	4
Figure 2. Vehicle teleoperation framework	13
Figure 3. Direct teleoperation interface (International Submarine Engineering, Ltd.)	15
Figure 4. Direct control system model	15
Figure 5. The Web Interface for TeleScience (WITS) provides tools for supervisory control of planetary rovers (NASA Jet Propulsion Laboratory).	16
Figure 6. Supervisory control system model.	17
Figure 7. Fully autonomous control system model	18
Figure 8. Collaborative control system model.	19
Figure 9. Collaborative control system architecture	27
Figure 10. The QueryManager: left, attribute-based message filtering; right, message dispatch.	31
Figure 11. The EventLogger: left, event recording; right, event retrieval.	32
Figure 12. Left, PdaDriver; right, field-work with PdaDriver.	41
Figure 13. PdaDriver architecture	45
Figure 14. Queries to the user	46
Figure 15. Responses from the robot.	47
Figure 16. Display of system events captured by the EventLogger	47
Figure 17. PdaDriver control modes	48
Figure 18. PdaDriver button bars.	48
Figure 19. Direct mode provides position and rate control of robot pose.	49
Figure 20. Image mode is designed for image-based, waypoint driving.	50
Figure 21. Projection uncertainty (per pixel) for zero camera tilt	51
Figure 22. Map mode is designed for map-based, waypoint driving.	52
Figure 23. Sensor mode is used to control sensors and robot perception.	53
Figure 24. Left, Pioneer-AT; right, Pioneer2-AT.	61
Figure 25. Safeguarded robot controller architecture	63
Figure 26. Each range sensor reading is used to update the map grid.	67

Figure 27. The HIMM method produces accurate maps even if the range sensor (e.g., ultrasonic sonar) has low angular resolution.	68
Figure 28. The MapMaker uses a fixed-size grid which is shifted as the vehicle moves.	69
Figure 29. Localization error update moves all certainty values towards zero.	69
Figure 30. MapServer maps: (a) unexplored environment; (b) corridor; (c) corridor and room (initial entry); (d) corridor and room (after exploration).	71
Figure 31. Query to the human: “Can I drive through?”	78
Figure 32. Some interesting “rocks”.	80
Figure 33. RockFinder state machine	81
Figure 34. Collaborative exploration	82
Figure 35. MotionDetector state machine	84
Figure 36. Indoor surveillance with two robots. Left, PioneerAT and Pioneer2-AT; right, PdaDriver displays.	85
Figure 37. Human-robot interaction during surveillance	86
Figure 38. Outdoor reconnaissance with two robots	87
Figure 39. Human-robot interaction during outdoor reconnaissance	88
Figure 40. Test subject and Pioneer2-AT in the CI study environment	93
Figure 41. Solitaire was used as to practice contextual inquiry.	95
Figure 42. Vehicle teleoperation framework.	121
Figure 43. Radio-controlled target drones: left, Radioplane RP-5A (Western Museum of Flight); right, Culver PQ-14 Cadet (USAF Museum).	124
Figure 44. Left, Predator (USAF Air Combat Command); right, Predator control station (USAF Air Combat Command).	124
Figure 45. Left, Lunokhod 1 (Lavochkin); right, Lunokhod control station (NPO Lavochkina museum).	125
Figure 46. Unmanned Ground Vehicles: left, Advanced Teleoperator Technology TeleOperated Dune Buggy (SPAWAR); right, TeleOperated Vehicle (SPAWAR).	126
Figure 47. Left, the Pioneer robot; right, Pioneer control console. (Carnegie Mellon University and RedZone Robotics, Inc.).	127
Figure 48. Cable-controlled Underwater Recovery Vehicle I (SPAWAR)	127
Figure 49. Left, an “eyeball” ROV; right, control console. (Deep Ocean Engineering, Inc.)	128
Figure 50. A work-class ROV designed for drilling support and oceanographic research (International Submarine Engineering, Ltd.).	128
Figure 51. Direct teleoperation interface (International Submarine Engineering, Ltd.)	129
Figure 52. Telepresence interface for heavy work vehicles (Helsinki University of Technology).	130
Figure 53. Left, UI2D was used to operate Dante II; right, Dante II in the Mt. Spurr, Alaska volcano. (Carnegie Mellon University).	131
Figure 54. The VIZ virtual reality interface displays multiple data sets including stereo-vision mapped terrain and digital elevation maps (NASA Ames Research Center).	132

Figure 55. The Web Interface for TeleScience (WITS) provides tools for supervisory control of planetary rovers (NASA Jet Propulsion Laboratory).....	133
Figure 56. The Rover Control Workstation (RCW) was used to operate the Sojourner rover on Mars (NASA Jet Propulsion Laboratory).	134
Figure 57. GestureDriver (Swiss Federal Institute of Technology Lausanne)	134
Figure 58. Web interfaces (Swiss Federal Institute of Technology Lausanne)	135

List of Tables

Table 1. User attributes	29
Table 2. Stereotype attributes.	30
Table 3. Vehicle teleoperation dialogue (user to robot).	33
Table 4. Query-to-user (questions from the robot): attributes	34
Table 5. Query-to-user (questions from the robot): timeout and action	35
Table 6. Design and evaluation heuristics	44
Table 7. Motion commands	60
Table 8. Pioneer-AT and Pioneer2-AT specifications.	61
Table 9. Pioneer-AT sensor suite	62
Table 10. Pioneer2-AT sensor suite	62
Table 11. Controller modules	65
Table 12. Test session schedule	94
Table 13. Test subject demographic data.	96
Table 14. Test subject background data.	96
Table 15. Learning style and final performance	97
Table 16. Additional query-to-user (questions from the robot).	113
Table 17. FPC local throughput (messages per second)	145
Table 18. FPC remote throughput (messages per second).	145

1

Introduction

There's an old saying that collaborations succeed only if each partner does 60% of the work. It's funny and a little true, but even if it's entirely true, it's still a lot better than doing 100% of the work.

— Larry Bond

1.1 Robot as Tool

In telerobotics, a robot is commonly viewed as a tool: a device capable of performing tasks on command. As such, a robot has limited freedom to act and will always perform poorly whenever its capabilities are ill-suited for the task at hand. Moreover, if a robot has a problem, it often has no way to ask for assistance. Yet, frequently, the only thing the robot needs to get out of difficulty, and to perform better, is some advice (even a small amount) from a human.

Consider the situation in which a mobile robot is driving outdoors and encounters tall grass directly in its path. Depending on its on-board sensors, the robot's perception system may have difficulty deciding if the grass is an obstacle and whether it poses a danger. Thus, the robot may be unable to proceed or may decide to take a long, resource consuming detour. If, however, the robot is able to discuss the situation with a human, a better solution can be found. For example, if the robot asks "Is there an obstacle ahead?" and displays a camera image, the human can help the robot decide that it is safe to move forward (i.e., to drive through the grass).

Generally speaking, robots are more adept at making some decisions by themselves than others. For example, structured planning (for which algorithms or well-defined solutions exist) has proven to be quite amenable to automation. Unstructured decision making, however, remains the domain of humans, especially whenever common sense is required (Clarke 1994). In particular, robots continue to perform poorly at high-level perceptual functions, including object recognition and situation assessment (Milgram, Zhai and Drascic 1993).

In order for robots to perform better, therefore, they need to be able to take advantage of human skills (perception, cognition, etc.) and to benefit from human advice and expertise. To do this, robots need to function not as passive tools, but rather as active partners. They need to have more freedom of action, to be able to drive the interaction with humans, instead of merely waiting for (or blindly executing) human commands.

1.2 Human as Controller

Another way to appreciate the limitations of “robot as tool” is to look at its dual: “human as controller”. In this system model, the human receives information, processes it, and generates a control action for the system (e.g., a telerobot) to execute. If “robot as a tool” is troublesome for robots, “human as controller” is often problematic for humans, especially in teleoperation.

The first problem with “human as controller” is that it consumes valuable human resources and may awkwardly bind the system’s capability to the operator’s skill. Even if a robot is very capable (in terms of accuracy, repeatability, etc.), the human operating it may not be. These difficulties are particularly acute with direct teleoperation (manual control). Some common problems are: operator handicap (skill, knowledge, etc.), sensorimotor limits (reaction time, decision speed), cognitive and perceptual errors (e.g., misclassification), and physical difficulties (nausea, fatigue) (Ferrel 1967; Sanders 1993; Sheridan 1992).

The second problem with “human as controller” is that the quality of the human-machine connection significantly impacts performance. In particular, an effective operator interface¹ is critical for conveying information and feedback to the operator. Inadequate displays, inappropriate modeling, and inefficient control inputs contribute to operator error (Murphy 1996). Moreover, poor interfaces prevent the operator from becoming immersed in the task, thus limiting his ability to directly (i.e. efficiently) act. Finally, if the operator and robot are widely separated, communications may be affected by noise or signal transmission delays. Delay is particularly insidious because it can make direct teleoperation impractical (Sheridan 1993).

The third manner in which “human as controller” causes problems is the imbalance in roles (human as supervisor/master, robot as subordinate/slave). Whenever the human is “in the loop”, he has reduced capacity for performing other tasks. Additionally, since the robot is under human control, the system halts whenever the robot waits for direction. This is particularly problematic when the operator must simultaneously control multiple robots because he is forced to divide and switch his attention among the robots.

1.3 Collaborative Control

In short, there are numerous problems arising from “robot as tool” / “human as controller”. Thus, if we want to construct teleoperation systems that are not constrained by these factors, we need to find a better model. What we need is an approach that is more flexible, that allows both the robot and the human to participate in task performance, and that facilitates human-robot interaction.

To address this need, I propose a new system model for teleoperation called *collaborative control* (Fong, Thorpe, and Baur 1999). In this model, a human and a robot *collaborate* to perform tasks and to achieve common goals. Instead of a supervisor dictating to a subordinate, the human and the robot engage in *dialogue* to exchange ideas, to ask questions, and to resolve differences. Instead of the human always being completely “in control”, the robot is more equal and can treat the human as a limited source of planning and information, just like other system modules.

1. also referred to as “user interface” or “control station”.

My claim, which I will prove in this thesis, is that:

Collaboration is an effective model for building and operating teleoperation systems. Collaboration helps coordinate robot action, facilitates adjustable autonomy and human-robot interaction, and enables the human to compensate for inadequate autonomy.

I use the term collaborative control because it is analogous to the interaction between human collaborators. Specifically, when humans engage in collaboration, we encourage each collaborator to work with others towards a common goal. We also allow each collaborator to take self-initiative and to contribute as best he can. At the same time, however, we leave room for discussion and negotiation, so that potential solutions are not missed.

An important consequence of collaborative control is that the robot can decide how to use human advice: to follow it when available and relevant; to modify (or even ignore) it when inappropriate or unsafe. Thus, if the robot is operating autonomously and has problems, it can ask the operator “What should I do?” If the human is capable of responding (and can do so in a timely fashion), then the advice will be used. However, if the advice is late (e.g., due to communication delay) or unsafe (e.g., “Drive off the cliff.”), then the robot may view the advice with skepticism and seek clarification. For example, the robot could ask for confirmation (“Do you really want me to drive off the cliff?”), reformulate the question (“Other than driving off the cliff, what should I do?”) or negotiate (“Can I drive around the cliff instead of off it?”).

This is not to say that the robot becomes “master”: it still follows higher-level strategy (goals and tasks) set by the human. However, with collaborative control, the robot has more freedom in execution and is able to better function if the operator is distracted, inattentive, or unavailable. As a result, teleoperation becomes more adaptable, more flexible and better able to accommodate varying levels of autonomy, interaction and human capabilities.

Collaborative control is a radical departure from traditional teleoperation, from the conventional “robot as tool” system model. Collaborative control encourages human-robot interaction to be more natural, more balanced, and more direct. Collaborative control also allows robots to benefit from human assistance during perception and cognition, and not just planning and command generation.

To understand how this works in practice, let us take a closer look at the situation I described earlier: a mobile robot is driving along a path when it has difficulty deciding if there is an obstacle in its way. Figure 1 shows an example of this occurring in an indoor experiment I performed (see Section 6.1). During this test, the robot unexpectedly encounters a cardboard cat directly ahead (Figure 1, left). The problem, for the robot, is that range sensing (sonar, stereo vision, etc.) may indicate that the path is blocked or may return conflicting information. For example, sonar readings may indicate an obstacle, but stereo may not detect the cat due to lack of surface texture.

At this point, the robot must make a decision. With conventional system design, there are three choices: it can wait for the path to become clear (i.e., until all sensors return consistent “path is clear” readings), it can look for a way around, or it can ignore some or all of the sensor data and continue driving forward. All of these strategies have significant problems: “wait until clear” may delay the robot indefinitely, “make a detour” may not be pos-



Figure 1. Vehicle teleoperation using collaborative control:
Left, robot encounters a potential obstacle;
center, query to the human: “Can I drive through?”;
right, collaboration result.

sible or may consume excessive resources, and “charge ahead” may result in excessive damage to the robot.

With a collaborative control system, however, the robot is able to converse with the human and ask his opinion. For example, the robot can present sensor data (e.g., a camera image) to the human and ask whether or not it is safe to drive forward (Figure 1, center). Once the human sees the image, he can decide (based on his experience or interpretation of the data) that the blocking obstacle is a cardboard cat, that it does not pose a danger to the robot, and that driving over it is acceptable (Figure 1, right). This allows the robot to avoid needless delay or having to make an unnecessary detour. In other words, by collaborating with the human, the robot can take advantage of the most efficient solution (i.e., drive through).

I should point out that if the human is unable to answer “Can I drive through?”, either because he cannot decide that it is acceptable or because he is not available to respond, merely asking the question is not detrimental (except perhaps in time) to task performance. The only manner in which human-robot dialogue might cause a problem is when the human makes a significant error (e.g., he mistakes a real cat for a cardboard one). There are, however, numerous ways to address this. One way would be to weigh the human’s response against the votes of other decision making modules (Rosenblatt 1995).

1.4 Why Collaborative Control?

Collaborative control is worth studying for the following reasons. First, I contend that collaborative control is both a new and novel approach to teleoperation. The idea that human-robot collaboration can serve as the basis for system design is new. The use of dialogue for coordinating robot action and autonomy is novel. Second, I assert that collaborative control can enable humans and robots to work better together. This is important for applications (e.g., human-robot exploration) in which success is strongly dependent on joint task performance. Finally, I claim that collaborative control enables, for the first time, robots to use humans as a resource for non-command functions such as perception. Given that automation continues to be poor in many areas, it seems only logical to take advantage of superior human capabilities.

How is collaborative control better than traditional (conventional) teleoperation methods? To begin, I believe that it helps humans and robots to better interact. Unlike existing teleoperation models, collaborative control strives to make human-robot interaction more like human-human interaction, thus more natural and effective. Additionally, I would argue that collaborative control makes teleoperation more flexible. By giving robots greater freedom to act and to function as partners (instead of as subordinates), it allows progress to be made regardless of how “in the loop” the human is. Lastly, I am convinced that using dialogue for coordination enables collaborative control to better accommodate varied users and adjustable autonomy.

In short, I believe that the most important attributes of collaborative control are:

- It enables the human to function as a robot resource.
- It offers an effective framework for coordination.
- It provides an efficient mechanism for adaptation.

Let us examine each of these points in more detail.

Human as a robot resource

Collaborative control enables the human to serve as a resource for the robot. Other researchers (e.g., Rosenblatt 1995) have examined the use of humans as a system element or module. In these approaches, however, the assumption is that the human will only provide control (command) input to the system. Collaborative control, however, is not limited, or restricted, by this narrow assumption. Instead, the robot is free to ask for human assistance when it is performing functions such as perception or planning. This has several important consequences.

First, it gives the robot the opportunity to seek alternate solutions to problems (i.e., by asking for human advice), rather than being forced to pursue a poor strategy or merely to stop. In other words, collaborative control allows the human to help compensate for limited autonomy. Moreover, this is true regardless whether the human is a novice or an expert. For many teleoperation situations, especially when the robot is stuck or uncertain how to proceed, even a novice can help.

Second, it enables the robot to have a more “equal” role in teleoperation, in the sense that it can ask the human to do things for it (perceive, plan, solve problems, give advice). As I have already stated, this is quite different from conventional teleoperation, which limits the robot to being a subordinate or tool of the human. Yet, having the robot be more

“equal” is not only beneficial for the robot (since it can use the human to compensate for its inadequacies), but also for the human because he does not have to continuously be “in the loop” monitoring or supervising the robot (i.e., the robot notifies the human when it needs help).

Finally, although the human may serve as a robot resource, there is no fundamental requirement that the human immediately (or ever) answer every robot question. Thus, collaborative control encourages use of human perception and cognition without requiring time-critical or situation-critical response from the human. If the human is available, he can provide direction or problem solving assistance. But, if the human is not, the system can still function.

A framework for coordination

One of the problems in teleoperation, especially when supervisory control is used, is that the operator may, from time to time, find himself being a “passive monitor”. This is troublesome because whenever the operator is “out of the loop” (i.e., only monitoring/supervising), there is significant danger that he will become bored, complacent, and inattentive (Sheridan 1992). Collaborative control addresses this problem by having the robot notify the human whenever he is needed and directing his attention where it is needed. In other words, dialogue is used to coordinate joint task performance and problem solving.

Although this may seem similar to a prioritized alarm, as found in many automated monitoring or warning systems, there is a significant difference. With prioritized alarms, the operator is notified only for pre-defined error or warning situations. For example, computer anti-virus software typically alerts an operator whenever certain trigger conditions (e.g., unauthorized file modification) occur. With collaborative control, however, the human is contacted not only to deal with errors or contingency situations, but also when his assistance is required for “normal” operation (e.g., to help with perception).

Furthermore, since the human can only focus his attention on one task or problem at a time, collaborative control uses *query arbitration* (see Section 3.3) to select which robot request is presented. This allows the dialogue to focus where it is most needed, in terms of safety, priority, etc. This approach makes collaborative control scalable and helps improve the human’s effectiveness at performing simultaneous, parallel control.

For example, multi-robot teleoperation has traditionally been a difficult problem. Whenever an operator is often forced to divide his limited resources (attention, decision making, etc.) among multiple robots, control is complicated because the operator must continually switch contexts to locate and address problems. With collaborative control, however, context switching is driven by the needs of the robots. In particular, query arbitration guarantees that the operator’s attention will always be directed to the robot most urgently in need of help.

A mechanism for adaptation

Unlike traditional teleoperation, collaborative control can adjust its method of operation based on situational needs. Because the robot makes requests of the human, but is also aware that the human may not respond, collaborative control enables fine-grained trading and sharing of control. Specifically, in a situation where the robot does not know what to do or is performing poorly, it has the opportunity to give control (e.g., decision making) to the human for that specific situation. In other words, collaborative control enables work to

be dynamically allocated to the robot or the human throughout task performance. With other forms of teleoperation, the division of labor is fixed *a priori* or else is coarsely switched on a per task or time basis.

In a sense, collaborative control is similar to adjustable autonomy, which dynamically adjusts the level of autonomy of an agent depending on the situation. The difference lies in how adjustments are made. Collaborative control uses human-robot dialogue (i.e., queries from the robot and responses, or lack thereof, from the human), as the mechanism for adaptation. Adjustable autonomy systems, however, operate with event triggers or by having the human switch interface modes.

Perhaps the most unique feature of collaborative control, however, is that it can naturally accommodate users having varied backgrounds and capabilities. This is because collaborative control incorporates a user model (Section 3.2) for dialogue management. With this model, human-robot interaction is adapted to a user's expertise, preferences, and other characteristics. This means that the robot is aware of with whom it is interacting and is able to modify dialogue accordingly. Thus, the robot can ask questions and can interpret responses, based on whether the user is a novice or an expert.

For example, the robot can dynamically decide whether or not to ask a question based on how accurate the response must be (i.e., how important it is to have a good answer). This is similar to what happens when humans have a medical problem. If the problem is minor, and the required diagnosis "accuracy" is low, we are willing to talk to a general practitioner. But, as a correct diagnosis becomes more difficult to obtain, or critical to our continued well-being, we insist on consulting specialists.

1.5 Related Research

Collaborative control is related to work in several areas. In both supervisory control and cooperative teleoperation, the primary objective is to make tasks easier for humans to perform by reducing operator workload. Unlike collaborative control, these approaches focus solely on the human, without consideration for the robot. In human-robot control architectures, adjustable autonomy, and mixed initiative, the common theme is to integrate the human into an autonomous control system. None of these approaches, however, use dialogue as a coordination or information transfer mechanism.

1.5.1 Supervisory Control

Supervisory control emerged from research on earth-based teleoperation of lunar vehicles (Ferrel 1967). The term supervisory control is from the analogy between a supervisor's interaction with subordinate staff and an operator's interaction with a robot (Sheridan 1992). Supervisory control is used most often for telemanipulation (e.g., Blackmon and Stark 1996), especially in known or structured environments. During the past decade, it has also been used for vehicle teleoperation (Adams 1996; Backes, Tharp, and Tso 1997; Lin 1995; Simsarian 2000; Stone 1996; Wettergreen et al. 1995).

In a sense, supervisory control models military structure: hierarchical, rigid control flow, supervisor "in charge" and subordinates restricted in what they can do. Collaborative control more closely resembles a research group. Although collaborative control has hierarchy, its control is more flexible and dynamic. Furthermore, each collaborator has greater freedom to take the initiative and to lead. Finally, collaborative control preserves the best

aspects of supervisory control (use of human perception and cognition) without requiring time-critical or situation-critical response from the human.

With supervisory control, an operator divides a problem into a sequence of tasks which the robot can perform on its own. Once control is given to the robot, it is ideally expected to complete the tasks without human intervention. During automatic action, the operator watches displays to monitor progress and to understand how the robot is performing. If a problem occurs, the operator is responsible for finding a solution and for devising a new task plan.

One of the fundamental weaknesses of supervisory control, therefore, is that the operator may become complacent or may even be negligent whenever the robot is operating autonomously. In particular, when the operator is a passive monitor, he may not perceive critical factors in the environment, he may not understand what those factors mean (especially with respect to task objectives), and he may fail to predict what will happen in the near future. There is, in fact, a long history of cases in which operators were reported to be unaware of automation failures and did not detect critical state changes when acting as monitors of automated systems (Endsley 1996).

Collaborative control directly addresses this problem by keeping the human engaged. Because the robot can use the human (e.g., to aid in “autonomous” perception or decision making) during task execution, the operator is more likely to remain actively involved and able to maintain situational awareness. While it is true that this increases the operator's cognitive workload (relative to supervisory control), it is still far better than having to do everything manually.

1.5.2 Cooperative Teleoperation

In cooperative teleoperation, multiple operators share or trade control. Cannon (1997) describes the use of “virtual tools” for telemanipulation. In his system, operators use these tools to define key actions at a supervisory level. A networked interface allows multiple operators to share control. Cannon refers to this interaction as “collaborative control” since multiple humans collaborate to effect control. This differs from the “collaborative control” of this thesis, which focuses on collaboration between humans and robots.

A more general multiple operator approach is described in (Habib 2000). In this position paper, a team of humans and machines, each having some level of autonomy, communicate and cooperate to achieve shared goals independent of time and geographical location. Habib's “collaborative control” concept shares some philosophy (e.g., recognition of the need to engage humans and machines in constructive dialogue) with this thesis. However, at this time, Habib's concept has not yet been implemented.

Goldberg and Chen (2001) propose a formal model for analyzing the performance of “collaborative control”, which is defined to be multiple sources (sensors, control processes, or human operators) simultaneously controlling a single robot resource. In this work, sources are modeled as deterministic finite automata that jointly control the motion of a simulated robot. Performance is then measured in terms of path error. Although this model provides a method for assessing collaborative performance, it is limited to systems which share control of individual plant variables (e.g., pose). Thus, this approach would not be appropriate for the collaborative control system presented in this thesis, which treats “collaboration” as the exchange and sharing of information.

Teleassistance tries to improve teleoperation by supplying aid (data filtering, decision-making tools, etc.) to the operator in the same manner that an expert renders assistance. For example, Murphy (1996) describes a teleassistance system which combines a limited autonomy robot architecture with a knowledge-based operator assistant. During teleoperation, this system provides “strategic assistance” so that the operator and robot can cooperate in cognitively demanding tasks. Collaborative control is somewhat the dual of Murphy’s approach: it provides the robot with human assistance, instead of supplying computer-based aid to the operator.

Rogers, Murphy, and Ericson (1996) describe a system called the “Attention Director”, which attempts to automatically select, display and enhance sensor data to assist the human in identifying and responding to robot failures. The characterization of the human is based on an information processing model of human visual interaction (how humans focus on relevant information in an image), originally developed for diagnostic radiology assistance. The Attention Director is similar to the QueryManager (described in Section 3.3). Both systems are designed to direct human attention to urgent problems and both employ a model of the human. The difference is that the Attention Director focuses purely on visual information, whereas the QueryManager works with human-robot dialogue.

1.5.3 Human Robot Control Architectures

Although most robot control architectures are designed for autonomy, some have addressed the problem of mixing humans with robots. One approach is to directly incorporate humans into the design, i.e., as a system element. DAMN, for example, is a behavior-based architecture in which individual modules vote on possible actions (Rosenblatt 1995). Command arbitration allows modules as disparate as autonomous safety behaviors and teleoperation to coexist.

Another approach is the use of prioritized control, in which operator commands may be overridden by autonomous modules. The best-known example of this is NASREM, which explicitly incorporated an operator interface into a layered, hierarchical control system (Albus 1987). More recently, the concept of safeguarded teleoperation has been used to enable novices to teleoperate a planetary rover (Krotkov et al. 1996a,b).

In all of these approaches, the assumption is that the human will only provide control (command) input to the system. Collaborative control, however, is not limited by this narrow assumption. Instead, it also allows the human to contribute high-level planning or perception input to robot modules.

1.5.4 Adjustable Autonomy and Mixed Initiative

Both adjustable autonomy and mixed initiative systems have recently received considerable research attention. Adjustable autonomy¹ means automatically and appropriately adjusting the level of autonomy based on the situation. Adjustable autonomy recognizes that over the course of time, control of tasks may need to pass back and forth between an operator and an automated system in response to changing demands. Adaptive autonomy, therefore, attempts to optimize this dynamic allocation of tasks and to allow humans to interact at whatever level of control is most appropriate for any situation at any time

1. sometimes called “adaptive automation”.

(Bonasso 1999; Dorais et al. 1999; Endsley 1996; Muscettola et al. 1998; Musliner and Krebsbach 1999).

Adjustable autonomy is most often described in terms of autonomous agents. In this context, it includes the ability for humans to adjust the autonomy of an agent, for the agent to adjust its own autonomy, and for a group of agents to adjust the autonomy relationships within the group. The most critical challenge in adjustable autonomy concerns how changes among modes or levels will be accomplished (Scerbo 1996). Some possible criteria include operator performance (monitored or predicted), operator workload (using biopsychometrics), and occurrence of critical events (mission, environment, etc).

Bonasso (1999) addresses integrating humans and adjustable autonomy in robot control. He contends that even in robot systems which are supposed to be fully autonomous, there is very often a need for human involvement. One way to achieve this is to design a control architecture for full autonomy and then relax the autonomy restriction at each level to accommodate direct human involvement.

Whereas adjustable autonomy emerged from the aerospace and space robotics community, mixed-initiative was developed primarily by artificial intelligence researchers. The objective of mixed-initiative is for humans and autonomous agents to jointly participate in planning and problem solving. In successful mixed-initiative systems, this cooperation is believed to produce better results than either the human or machine can achieve alone (Ferguson, Allen, and Miller 1996).

For example, Myers and Morley (2001) describe a framework called “Taskable Reactive Agent Communities” (TRAC), which supports the directability of a team of agents by a human supervisor. With TRAC, the human manages agent activity and specifies the level of involvement (interaction) by modifying task guidance at execution time. A formal language is used to represent guidance and agents are assumed to always be capable of fully autonomous operation.

Although both adjustable autonomy and mixed-initiative share some aspects of collaborative control, neither of these approaches completely addresses the idea of peer interaction between humans and robots. The distinctive feature of collaborative control is that it uses human-robot dialogue as a mechanism for adaptation and a framework for coordination.

1.5.5 KTH Collaborative Human-Robot Systems

Collaborative control is perhaps most closely related to two collaborative human-robot systems developed at the Swedish Royal Institute of Technology (KTH). In both systems, humans and a mobile robot engage in “natural” (i.e., close and direct) interaction to perform tasks (Simsarian 2000; Green and Severinson-Eklundh 2001). To understand how collaborative control differs from these systems, let us briefly examine each in turn.

Supervisory collaborative control

Simsarian (2000) discusses “supervisory collaborative control” of a remote semi-autonomous mobile robot. In this system, the human interacts with the robot via a Collaborative Virtual Environment (CVE). The CVE enables the human to visualize the remote environment and to directly specify task commands (e.g., 3D point-to-point robot motion).

The primary weakness of Simsarian’s approach is the adherence to the “robot as tool” paradigm. In particular, he argues that a supervisor-assistant relationship is sufficient for

effective human-robot collaboration. Although Simsarian recognizes that some situations (dynamic environments, complex tasks in unstructured environments, etc.) require human-robot work to be dynamically allocated, his system is limited to providing multiple command options to the user. Thus, unlike collaborative control, in which the level of interaction and autonomy can adjust automatically, the burden is placed on the user to manually choose which control mode is appropriate at any given time.

Moreover, even though Simsarian observes that the operator “should have some method to evaluate (and possibly negotiate solutions for) robot task successes and faults”, his system only provides a single mechanism (i.e., a virtual environment user interface) for doing so. Collaborative control is more flexible because dialogue enables the robot to guide human attention to problems and allows the human to help compensate for inadequate autonomy.

The CERO Dialogue System

The Interaction and Presentation Laboratory (IPLab) is currently developing an intelligent service robot, CERO, for performing common office tasks such as “fetch and carry” (Green & Severinson-Eklundh 2001). Human-robot interaction occurs via task-oriented dialogue using a spoken natural language interface.

In the CERO system, speech input is transformed into a logical expression via a simple context-free grammar and semantic analysis. This expression is then processed using a set of rules to determine which robot actions (synthetic speech generation or command execution) are appropriate given the current system state. Consequently, speech may cause the robot to ask questions to acquire new commands (“How may I help?”), to obtain command confirmation (“Go to the kitchen?”), or to clarify a pending command (“Where is the coffee?”).

Although both CERO and collaborative control use human-robot dialogue to improve system operation, there are several key differences. With CERO, the robot’s questions are all command related, are triggered by human speech, and are always asked prior to task execution. With collaborative control, however, the robot may ask a broad range of questions (to support cognition, perception, etc.) at any time, especially to solve problems it encounters while executing. Additionally, collaborative control uses query arbitration to manage simultaneous questions (from multiple modules or robots) and to direct human attention. Finally, collaborative control incorporates an explicit user model to adjust robot behavior and dialogue to the user (static adaptation) and to the situation (dynamic adaptation).

1.6 Thesis Organization

This thesis is an investigation of collaborative control. In the following chapters, I establish what is needed to design, build, and operate a collaborative control system. In particular, I identify the fundamental issues raised by this new approach and the techniques required to implement such a system. I then provide evidence, through demonstrations and a user study, to support my claim that “collaboration is an effective model for building and operating teleoperation systems”. In doing so, I prove that “collaboration helps coordinate robot action, facilitates adjustable autonomy and human-robot interaction, and enables the human to compensate for inadequate autonomy.” The thesis is organized as follows.

In Chapter 2, I present the problem context and the development of collaborative control. I begin by describing vehicle teleoperation: its characteristics and the system models which

are conventionally employed. I then discuss how collaborative control can address the limitations of these existing methods. I conclude by examining the key design issues that must be addressed to create a collaborative control system.

The next three chapters are dedicated to system design and construction. In Chapter 3, I develop general design principles, review my design approach, then describe the system architecture and modules that support human-robot dialogue. In Chapter 4, I discuss how these general principles translate into a user interface that enables the human to interact with and assist the robot. In Chapter 5, I describe how both system and interface requirements, in turn, affect the design of a robot controller that supports varying degrees of human-robot cooperation.

In the last section of this thesis, I study and test collaborative control and present my conclusions. In Chapter 6, I analyze the impact of collaborative control in three vehicle teleoperation applications. In Chapter 7, I document the results of a user study that tested my system design and evaluated how collaborative control works in practice. In Chapter 8, I discuss the strengths and weaknesses of collaborative control, identify directions for future research, and summarize the contributions of my work.

Following the main text are three appendices. In Appendix A, I review the history of vehicle teleoperation and survey the wide range of operator interfaces currently in use. In Appendix B, I describe the two toolkits (FPC and VisLib) developed during the course of this thesis, both of which are freely available as open-source. In Appendix C, I give details of the user study (clearance request, study plan, and subject questionnaire).

Developing Collaborative Control

The context for this thesis is vehicle teleoperation: the remote control of a vehicle. Despite continuing advances in autonomy, there will always be a need for human involvement in vehicle teleoperation. In particular, tasks such as exploration, reconnaissance and surveillance will continue to require human supervision, if not guidance and direct control. For these applications, therefore, system performance will be governed by the efficiency and effectiveness of human-robot interaction.

This chapter begins with a brief presentation of vehicle teleoperation (greater details, including a history of vehicle teleoperation and a survey of current operator interfaces, is given in Appendix A). I then discuss the existing system models used in vehicle teleoperation and compare them to collaborative control. The chapter concludes with a review of the key design issues for collaborative control.

2.1 Vehicle Teleoperation

Vehicle teleoperation means simply: *operating a vehicle at a distance*. As with all teleoperation, vehicle teleoperation enables the capability to sense and to act in a remote location. Figure 2 illustrates the basic vehicle teleoperation framework. A human operator works at a control station, generating commands via input/control devices and receiving feedback from displays. The remote vehicle, which executes the operator's commands, is equipped with sensors, actuators and often some level of autonomy. The operator and the vehicle are separated by a barrier (environment, distance, time, etc.) and exchange information via a communication link.

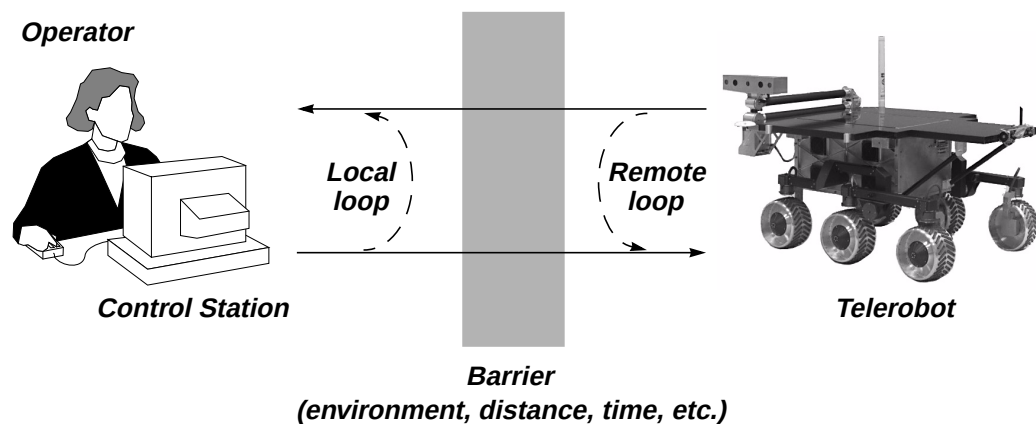


Figure 2. Vehicle teleoperation framework

Although some restrict the term *teleoperation* to denote only direct control (i.e., no autonomous functions), I consider teleoperation to encompass the broader spectrum from manual to supervisory control (Sheridan 1992). Thus, a vehicle teleoperation system may employ continuous/direct control, intermittent/supervisory control or any point between. Furthermore, the type of control can vary over time or as a function of the situation and may be shared/traded between the human operator and the remote vehicle.

One might argue that, by this definition, any robot that accepts and carries out human commands is teleoperated. I do not make this claim. Just as the term “artificial intelligence” does not encompass all of robotics (in spite of what some researchers and pundits believe), neither does the term “teleoperation”. I would, however, contend that any robot in which a significant fraction (in terms of time, function, etc.) of control or use is human-directed, human-guided, or human-supervised can be considered to be “teleoperated”.

Vehicle teleoperation has several characteristics which distinguish it from simple *remote control* (i.e. line-of-sight radio control) and other types of teleoperation (e.g., telemanipulation). First, vehicle teleoperation demands reliable navigation. Since vehicles are often deployed in unfamiliar or unstructured environments, navigation problems may lead to system failure or loss. Second, vehicle teleoperation requires efficient motion command generation. In many cases, task performance is directly correlated to how well a vehicle moves. Finally, vehicle teleoperation calls for accurate interpretation of sensor data. Considering that most applications focus on exploration, reconnaissance or observation, it is clear that misinterpreting or misjudging the remote environment will cause difficulties.

Vehicle teleoperation is used when it is necessary to operate in a hazardous or difficult to reach environment and when the primary task is exploration, observation or reconnaissance. It is also often used to reduce mission cost and to avoid loss of life. During the past century, vehicle teleoperation has been successfully applied in the air, on the ground and underwater. Current applications include reconnaissance, surveillance, and target acquisition (RSTA), exploration, remote mining, inspection, facility security and entertainment.

There are three basic problems in vehicle teleoperation: figuring out where the vehicle is and what state it is in, determining where it should go, and moving it there while avoiding obstacles and collision (Fong, Thorpe, and Baur 2001a). In addition, there are a wide variety of task-specific problems such as observation, environment mapping, and deploying payload. These problems can be difficult to solve, particularly if the vehicle operates in a hazardous environment with a limited (low bandwidth and/or high-delay) communication link. The difficulty increases when we add additional factors such as operator variation (training, skill, etc.), multiple vehicles, and moving (or malicious) hazards.

2.2 Conventional System Models

There are three system models conventionally used to operate vehicles: direct control, supervisory control, and fully autonomous control. The *direct control* model has been in wide use since the early 1900's. *Supervisory control*, originally called meta-control, was developed in the 1960's as a way to characterize operators functioning in discontinuous control loops (Sheridan 1992). *Fully autonomous control* describes robot systems that rely on humans to set high-level goals or to specify high-level strategy, but which then execute independently of the human.

2.2.1 Direct Control



Figure 3. Direct teleoperation interface (International Submarine Engineering, Ltd.)

The most common method for performing vehicle teleoperation has traditionally been *direct control*: the operator directly operates the remote vehicle using hand-controllers (e.g., 3-axis joysticks) while monitoring video (from vehicle or worksite cameras) on one or more displays (see Figure 3). A great many vehicles, including aircraft (RPVs, UAVs), ground transport (UGVs), and submersibles (ROVs) have all been operated using direct control. In fact, for many of these vehicles, this approach continues to be state-of-the-art.

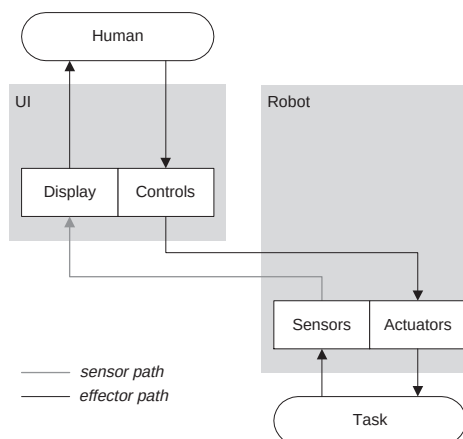


Figure 4. Direct control system model

Direct control is used because it is cheap and easy to implement, at least in comparison with the other system models. With direct control (Figure 4), the human closes the control loop. Although the robot (or interface) may assist in either (or both) the sensing or effector paths, the primary responsibility for perceiving the remote environment, making decisions, and executing the plan rests with the human.

It is well known that direct control can be problematic. Because all control decisions depend on the human, system performance is directly linked to human capabilities. Many factors including sensorimotor limits, knowledge, skill, training, etc., all play a role in how the system functions. Other factors, such as control station design and communication bandwidth, may also significantly influence the efficacy of a direct control system.

2.2.2 Supervisory Control

The *supervisory control* concept appeared as part of research on how earth-based operators might teleoperate lunar vehicles. The term supervisory control is derived from the analogy between a supervisor's interaction with subordinate staff (Sheridan 1992). To effect supervisory control, the operator divides a problem into a sequence of sub-tasks which the robot then executes on its own. To date, the majority of research in supervisory control has focused on process control and telemanipulation.

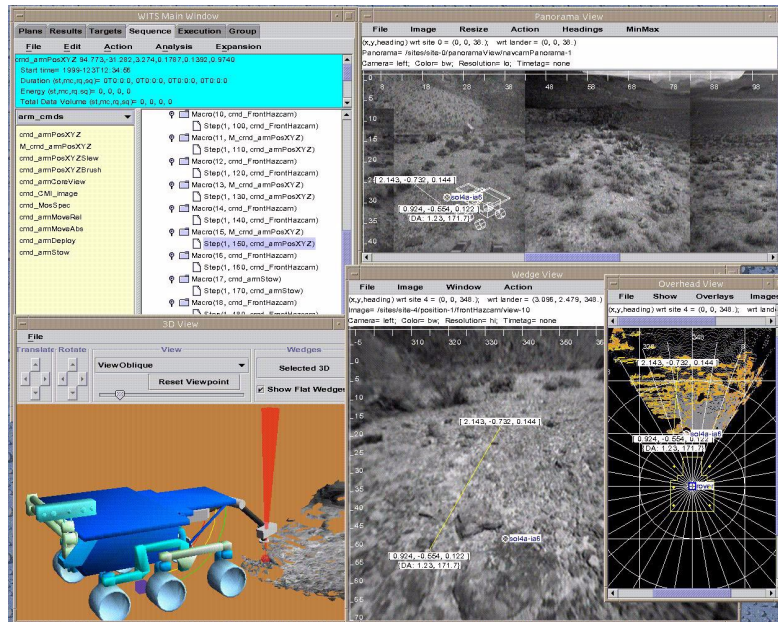


Figure 5. The Web Interface for TeleScience (WITS) provides tools for supervisory control of planetary rovers (NASA Jet Propulsion Laboratory).

With supervisory control, the human interacts with the robot through the user interface. Given a goal to achieve, the robot performs either a minor or major fraction of control, depending on its level of autonomy. The more competent a robot is, the longer it will operate autonomously (i.e., exclusive of the human). Once the human has given control to the robot, he supervises task execution by monitoring interface displays. These displays generally show sensor data, processed by one or more robot perception modules (Figure 5).

The supervisory control system model is shown in Figure 6. Sheridan (1992) notes that the human is not restricted to only to a supervisory role. Instead, he may intermittently control the robot by closing a command loop (*sensor-perception-display-control-cognition-actuator*), or he may control some variables while leaving the others to the robot. The former approach is known as “traded control” and the latter as “shared control”. With either approach, the human may choose to interact with the robot at different levels. For example, if the robot operates with a hierarchical controller, the human may close the control loop at a high, symbolic level (e.g., the top layer of robot *perception-cognition* shown Figure 6) or at a lower level (i.e., closer to the actuators).

Although supervisory control has been studied since the 1960's, there continue to be numerous open issues. First, various investigators have questioned whether removing the operator from active participation in the control loop makes detecting and diagnosing abnormalities more difficult. Additionally, it is unclear how to model a robot's compe-

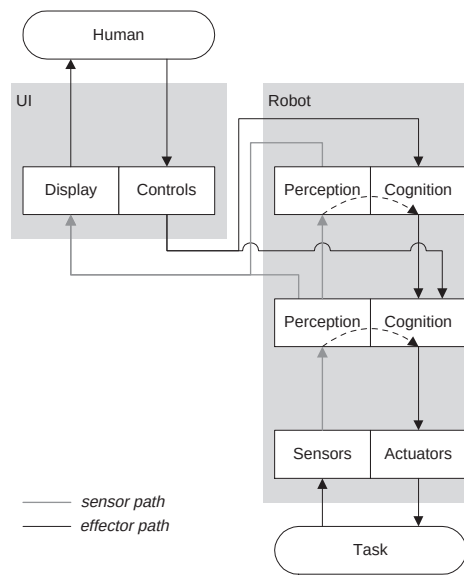


Figure 6. Supervisory control system model

tency (its ability to act “correctly” in a range of situations) to facilitate operator training and task partitioning. Finally, there are uncertainties related to the sharing/trading of control and to context switching, especially when an operator must control multiple vehicles.

2.2.3 Fully Autonomous Control

*Fully autonomous control*¹ is somewhat of a misnomer because it rarely, if ever, is fully automatic. With this system model, the human gives high-level goals or tasks to the robot, which independently achieves them. Planning may be performed prior to execution, interleaved with execution, or continuously during execution. Since the days of the Stanford cart and SRI’s Shakey, fully autonomous control has been what research in “autonomous mobile robots” has aspired to achieve.

In a sense, fully autonomous control is just supervisory control taken to its extreme: the human gives a high-level, abstract goal which the robot then achieves by itself. The difference between fully autonomous control and supervisory control is the nature of the goal. In almost all supervisory control systems, goals are limited (e.g., “drive to point X without hitting anything”) and the majority of task planning is performed by the human. With fully autonomous control, goals tend to be more abstract (e.g., “cut the hay in this field”) and the robot has more responsibility for deciding how the task will be performed.

Fully autonomous control is generally implemented using a robot control architecture (Hasemann 1995). At present, the most widely used approach is a three-layer architecture (Gat 1997). In this type of architecture, the bottom controller layer provides behavior-based reactive control, the middle layer sequences behaviors, and the top layer performs deliberative planning. Keeping the middle and top layers clearly separated is often more a philosophical issue than practical concern (i.e., it rarely impacts system operation). Figure 7 shows a robot controller with only two layers: the lower layer is close to the sen-

1. also called “automatic control”, “high-level control”, or “task-level control”.

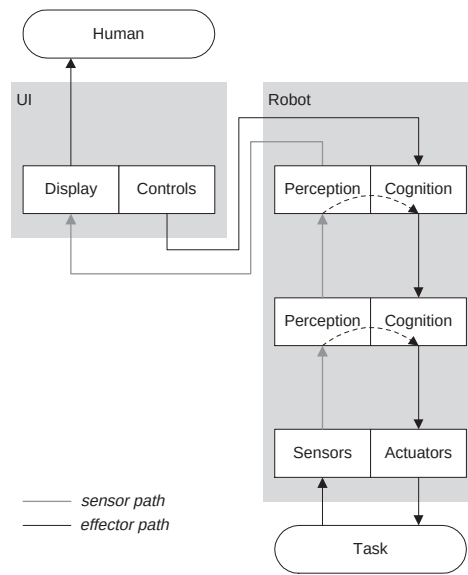


Figure 7. Fully autonomous control system model

sors/actuators and is “reactive” (i.e., *perception-cognition* is fast and minimizes use of state); the top layer is slower, more deliberative and involves more abstract reasoning.

With fully autonomous control, as with supervisory control, the human interacts with the robot through the user interface (Figure 7). Once the human has specified a goal to be achieved, the robot operates independently (i.e., it closes all control loops). System performance is thus constrained by how well suited the robot is for the situation. As the robot operates, the primary function of the human is to monitor execution via interface displays.

2.2.4 Limitations

Although all three of these system models have been used successfully, they all suffer from limitations of their common “robot as tool” / “human as controller” approach. In the case of direct control, tasks can only be performed while the human remains in the loop. Additionally, performance is limited by the operator’s capabilities and workload.

With supervisory and fully autonomous control, failure will occur whenever the human does not, or is unable to, recognize that the robot is ill-suited for the situation. In particular, whenever the human commands the robot to achieve a goal, the robot is forced to make decisions by itself. Consequently, the robot will always fail whenever any part of its autonomy (e.g., planning) is inadequate or inappropriate for performing the task.

Another limitation is that none of the models is able to accommodate a wide range of users. Direct control, for example, generally demands expert users because the difficulty and risk is high. The exceptions are when the vehicle is operating in a benign environment, or if substantial safeguards exist. With the other models, operator variation has less impact because the robot is able to execute autonomously. However, adjusting the level of autonomy to reflect the user’s needs, or to compensate for the user’s shortcomings, is difficult.

An additional problem is that these models all rely on the human’s ability to understand the remote environment and process. That is, in order for the system to function, the human must accurately maintain situational awareness, must ascertain what the robot is doing, and must recognize when the robot has problems. Thus, if the user interface is poor,

or if the human builds an incorrect mental model of what is happening, system damage or loss is likely to result. For example, the human may command the robot to drive through unsafe terrain because he believes the robot to be elsewhere.

Finally, given these limitations, it is clear that none of the models will work well for applications involving: a wide range of users (novices and experts), humans working in close proximity with robots, multiple robot control by a single user, and tasks which are difficult to automate in advance.

2.3 Collaborative Control System Model

As I have discussed, collaborative control addresses the limitations of existing system models through collaboration and dialogue. In supervisory or fully autonomous control, if the robot has difficulties, the only choices it has are to continue performing poorly or to stop. With collaborative control, however, the robot also has the option of asking the human to assist where needed: performing perception or cognition, providing advice or information, etc. Thus, more than other system models, collaborative control allows the human to supplement automation and to compensate for inadequacies.

Another way in which collaborative control differs is that it inherently provides fine-grained trading/sharing of control and adjustable autonomy. That is, because work is dynamically allocated through dialogue, the human is automatically included in the control loop as needed. This is significantly different from other models, which require the user to decide how, when, and where control should be allocated.

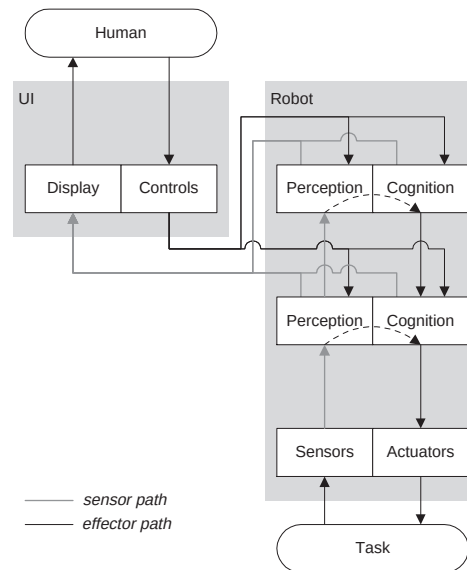


Figure 8. Collaborative control system model

With collaborative control, therefore, the human may be involved in multiple control loops (Figure 7). As with supervisory control, he may close a command loop (*sensor-perception-display-control-cognition-actuator*) or monitor task execution through interface displays. As a resource for the robot, however, the human may also close a perception loop (*perception-display-control-perception*) a cognition loop (*cognition-display-control-cognition*), or a combination of the two.

2.4 Key Design Issues

Collaborative control forces us to reexamine how we design, build and use teleoperation systems. Instead of developing only from a human viewpoint, we find that we must take a broader view. In particular, we must consider the robot perspective and design our system to satisfy not only human needs, but robotic ones as well. Although there are many design questions which we could ask, let us examine the key issues which we must address in order to construct a useful system.

2.4.1 Dialogue

First and foremost, a collaborative control system must have the capacity for *dialogue*. That is, the human and the robot should be able to communicate effectively, to convey information, to ask questions and to judge the quality of responses received. With collaborative control, each one should be able to say “Here is what I think”, to ask “What do you think?” and to interpret the response (i.e., to understand what the other has said or is trying to say). For example:

Robot. Based on map data, I think I should go to point ‘B’ directly.

Human. No, it is better to go to point ‘A’ first and then to point ‘B’.

Robot. I think the path to ‘A’ is blocked. Look at this image and tell me if you agree.

Human. Sorry, I’m too busy to look at it.

Robot. In that case, I’m going to wander until I find an unobstructed path to ‘B’.
When you have time, I’ll tell you what happens.

Though simple, there is a strong dynamic at work here. Initially, the robot seeks advice and the human provides a response. Since the response is unhelpful, the robot offers additional detail and asks for clarification. By this point, the human has become unavailable. So, the robot takes the initiative and suspends the dialogue until a later point in time. In this scenario, we see the human and the robot working together as partners, not supervisor and subordinate.

In short, good dialogue is two-way: it requires each party to understand to what the other is saying and to speak so the other can understand. To an extent, traditional teleoperation has dialogue (i.e., the feedback loop), but the conversation is limited. Dialogue offers the potential for richer, more flexible teleoperation. However, it raises questions such as: When should the robot “speak”? How does the robot format its queries? and What language features and vocabulary are required to guarantee effective communication?

2.4.2 Awareness

Under collaborative control, the robot is free to use the human to help satisfy its needs. But, as a consequence, the robot must be *aware*. This does not imply that the robot needs be fully sentient, merely that it be capable of detecting limitations (in what it can do and what the human can do), determining when and if it should ask the human for help, and recognizing when it has to solve problems on its own.

In addition, the robot must be able to decide, in any given moment and situation, whether involving the human has utility. Consider, for example, a robot operating a significant distance and time away from the human (e.g., a rover on Mars with an Earth-based operator).

In such a scenario, the robot has to be aware that it cannot ask the human for physical assistance and that obtaining cognitive/perceptual help may take considerable time. Thus, if the robot is about to collide with an obstacle, it makes no sense for it to ask the human whether it should stop (i.e., damage may already have occurred by the time the human replies).

To make a robot aware, we need to answer questions such as: At what level (and in what manner) does the robot need to model the human? How does the robot decide that its planned actions or chosen solution are sub-optimal? and What are the appropriate methods for detecting faults, both intrinsic and extrinsic, during planning and execution?

2.4.3 Self-reliance

A robot operating under collaborative control must be *self-reliant*. Since the robot cannot rely on the human to always be available or to always provide accurate information, it must have the capability to maintain its own safety. Specifically, the robot should be capable of avoiding hazards (collisions, rollover, etc.), monitoring its health (power level, temperature, etc.) and taking action to safeguard itself whenever necessary.

Under collaborative control, multiple robot modules may have multiple questions to ask at the same time. In the best case, the human will be available to provide a timely response to each one (i.e., he will answer each question while it is still pertinent). What is more likely, however, is that the human will not be able to answer every question because of his limited capacity (throughput) for answering questions. Thus, the robot must be able to cope with unanswered questions, even if that means proceeding on its own.

A related issue is what to do with invalid advice. Consider the situation in which the human answers an outdated question (i.e., the robot has already made a decision by the time the human responds). Should the robot ignore the answer or should it reconsider its action? The problem is that outdated advice may be hard to distinguish from unsafe advice. Thus, if we allow a range of users, how do we cope with the varying speed and quality of information?

2.4.4 Adaptiveness

By design, collaborative control provides a framework for integrating users with varied experience, skills, and training. As a consequence, however, the robot has to be able to adapt to different operators. In terms of dialogue, this means that the robot should ask its questions based on the operator's capacity to answer. For example, a scientist and a child usually (though not always!) have different knowledge and expertise for any given task. In other words, there is no benefit in asking a user for help if he lacks the skills to provide it.

Similarly, the robot needs to handle information received from an expert differently than that received from a novice. Specifically, the robot should evaluate a response based on the extent to which a user can be expected (or trusted) to provide an accurate response. Moreover, because a user's performance (accuracy, reliability, etc.) may vary, the robot may also need to adapt the weight it gives to a user's responses over time.

Finally, the robot needs to adapt its own behavior as needed. Because collaborative control allows for variable human response (in terms of accuracy, speed, etc.), the robot must be able to dynamically adjust its autonomy. In particular, the level of autonomy should adapt to fit situational needs and operator input (e.g., human decision making). If the robot is

unable to cope with a particular situation and the human is able to help, the level of autonomy should be reduced. But, if the human is unable to respond, the level of autonomy needs to be increased.

2.4.5 Human-Robot Interaction

When we use a collaborative control system, the traditional roles of operator and robot change. Instead of a subordinate awaiting direction, the robot is a co-worker seeking dialogue. Though the robot may ask for advice or assistance, there is no requirement for the human to immediately provide it. This frees the human from performing continuous control or supervision. If the human is available, he can provide direction. But, if he is not, the system can still function.

The key point is that with collaborative control, human-robot interaction can be very fluid and dynamic. This is both useful and problematic. It is useful because it enables the use of human perception and cognition without requiring time-critical response. It is problematic because it means that the relationship (expectations, roles, etc.) between human and robot is not static. Because the robot may not always ask for approval before acting (e.g., to guarantee its safety), it may be difficult for the human to build a mental model of the robot or to predict how a given task will be performed.

Fundamentally, the problem is that collaborative control changes the way humans perceive and interact with a telerobot. In conventional systems, there is always the underlying notion of “robot as tool”. Human-robot interaction, therefore, follows a fixed pattern: the human generates a command and the robot performs the action. With collaborative control, however, the robot is more equal and the model changes to “robot as partner”. In this case, the interaction pattern is less rigid, less defined, and less predictable.

As a consequence, if we are going to build a collaborative control, we need to enable humans and robots to understand each other through interaction, rather than by design or through inspection. We need to develop mechanisms which allow humans and robots to communicate, to discern and interpret intent, and to convey meaning. In short, we need to make human-robot interaction more “natural” and human-like.

2.4.6 User Interface Design

In conventional teleoperation, the user interface serves only the operator: displays provide information for human decision making, mode changes are user triggered, etc. In a collaborative control system, however, the user interface also has to support dialogue and to serve the robot. For example, the interface may need to provide mechanisms so that the robot can attract the user’s attention.

Most modern user interfaces are designed with user-centered methods. In user-centered design, the basic goal is to support human activity: to enable humans to do things faster, with fewer errors, and with greater quality (Newman 1995). A variety of human performance or usability metrics (speed of performance, error rate, etc.) are typically used to guide the design process.

We can use this approach to develop a collaborative control interface. For example, we can design dialogue support to maximize usability (e.g., allow the user to respond by drawing on maps or images). It is clear, however, that a strictly user-centered approach has limits. If we focus on the user, the interface will not support the robot. Thus, collaborative control

raises the questions: How useful is user-centered design? How do we consider the robot's needs? and Should the robot control the user interface?

2.4.7 Control and Data Flow

Collaborative control adds new constraints to system design. In conventional teleoperation, the flow of control is clear: the operator controls the robot's actions. Though he may share or trade control, the operator always retains ultimate authority. Collaborative control, however, allows control to be negotiated. It also allows the robot to consider commands as approximate or noisy.

Thus, a collaborative control system must have *command arbitration*: a means for deciding which actions to take over the short and long term. Command arbitration is particularly important when the robot and operator disagree. In such a situation, the arbiter needs to choose a course of action which will displease, and disrupt, both as least as possible.

As I discussed previously, collaborative control allows the robot to ask multiple questions of the human at the same time. These questions may have different forms (text, image, etc.), priority, validity (temporal, spatial) and difficulty. Thus, a collaborative control system must also have *query arbitration*: a mechanism for choosing which questions to ask based on both immediate (local) needs and overall (global) strategy. Because human-robot dialogue is central to collaborative control, an effective query arbiter is crucial to success.

Lastly, collaborative control requires flexible data handling. Since humans and robots operate differently, a collaborative control system must provide data in a variety of ways. For example, the human may decide that the terrain is flat by looking at an image; the robot may decide it is rough using proprioception. To have meaningful dialogue ("Why do you say it's rough when it's flat?"), both need to be able to exchange and present their data in a coherent manner.

2.5 Evaluation

To assess the value of any design, we need to evaluate it. Although evaluation can serve many functions, it is most often used to analyze strengths and weaknesses, to determine limitations and to ascertain (or estimate) how well a system works. The purpose of evaluation, however, should not be merely to compute a single measure of a system. Rather, evaluation should also provide an informed critique, so that the system can be improved. This is particularly important when we are trying to analyze a new concept or system that we have constructed for the first time.

Thus, to evaluate collaborative control, we need to adopt a formative approach. That is, we need to identify and apply evaluation methods that enable us to understand and improve the design. In particular, we want our evaluation to answer two questions: Are there problems with the current design? and What modification(s) should we make?

Formative evaluation can be done in many ways. We can collect anecdotal evidence to qualitatively assess how "easy-to-use" or "well-suited" a system is for a particular domain. We can measure specific features to quantitatively assess performance or error rates. Or, we can conduct a variety of examinations (structured walkthroughs, detailed review, etc.) to locate failure points, to prove correctness or to judge robustness.

For collaborative control, therefore, the challenge is to identify evaluation techniques that will provide the best insight into the concept and that will enable us to improve the system. Our objective should not be to produce simple performance metrics for a given task, but instead to ascertain if collaborative control is a useful paradigm for teleoperation.

2.6 Summary

In this chapter, I have described vehicle teleoperation, which is the underlying context for this thesis. Vehicle teleoperation means simply: operating a vehicle at a distance. Although some restrict the term teleoperation to denote only manual control, I consider teleoperation to encompass a broader spectrum which includes systems with autonomous functions (e.g., supervisory control).

There are three basic system models used in conventional vehicle teleoperation: direct (manual) control, supervisory control, and fully autonomous control. All these models share a common limitation: they do not allow the human and robot to effectively engage in dialogue. Thus, systems based on these models will fail or perform poorly whenever the human fails to recognize that the robot is ill-suited for the situation.

The collaborative control system model directly addresses these limitations. By allowing better human-robot communication and by enabling joint task performance, collaborative control has the potential to make teleoperation more flexible and effective. This is particularly true for applications involving changing or unknown environments, tasks which are difficult to automate, and users with varied capabilities.

In order to develop a collaborative control system, we must consider the robot perspective and design our system to satisfy not only human needs, but robotic ones as well. Among the key design issues we must address are dialogue, awareness, self-reliance, adaptiveness, human-robot interaction, user interface design, and control/data flow.

3

System Design

In this and the following two chapters, I present the design and implementation of a collaborative control system. The architecture contains three types of system modules, which operate independently and which are interconnected using an interprocess communication toolkit.

Dialogue modules control human-robot communication, manage the interaction between human and robot, and are described below. *Interface* modules mediate human-robot dialogue, enable the human to converse with and to assist the robot, and are presented in Chapter 4. *Robot* modules provide the control system for mobile robot operation, allow the robot to collaborate with and adapt to the human, and are discussed in Chapter 5.

This chapter starts with a review of the overall design, focusing on design principles, design approach, and system architecture. I then describe the three dialogue modules that I developed and the human-robot messages that support vehicle teleoperation. This chapter concludes with an overview of the research areas that guided the system design.

3.1 Design

3.1.1 Principles

To create a collaborative control system, I began by developing a set of design principles based on ideas in human-robot interaction, collaboration systems, and human-computer dialogue. Research in human-robot interaction provided three guidelines:

- Try to make robot more human, in the sense of engaging the user in task-relevant dialogue (e.g., asking human-like questions).
- Provide the robot with a model of the human to help guide communication (what is said/asked) and how responses are used.
- Mediate interaction through a user interface (no proximal interaction).

Collaboration systems contributed the idea that the human and robot should be able to function smoothly and productively as an integrated team and also the following ideas:

- Human-robot communication must be effective: focus only on task-relevant details
- Consider humans and robots to have fundamentally asymmetric abilities (i.e., design with a human complementary approach)
- Recognize that different users have different abilities
- Adjust autonomy as a function of situation and dialogue: the human's response should drive autonomy

Work on human-computer dialogue inspired the idea that the robot should be able to ask the human questions. From this basic concept, three other design principals emerged:

- Dialogue management should focus on choosing questions are asked: selecting which questions to ask based on the user and coordinating when the questions are to be presented
- Employ stereotype profile for modeling users, for assisting dialogue management, and for controlling the types of commands available
- Both the human and the robot should be able to control the conversation (though not necessarily in the same manner or with equal priority)

3.1.2 Approach

After defining design principles, I then considered how dialogue structure and flow should guide human-robot interaction. First, for human-to-robot dialogue, I chose to concentrate on vehicle mobility issues (navigation, collision avoidance, obstacle detection, etc). Thus, the design incorporates command tools for remote driving and for controlling automated perception tasks. The design also provides the human with tools for reviewing what has occurred (e.g., system events) and to ask for basic robot status (health, task progress, etc.).

Second, for robot-to-human dialogue, I decided to emphasize support of autonomous functions. In other words, to enable the robot to obtain human assistance during task execution. In particular, I identified two types of queries, safeguard and task, that the system would need to support. Safeguard queries are those that concern configuration parameters (safety levels, driving rates, etc.) and urgent, health-related problems such as rollover danger. Task queries could describe any number of issues relevant to task performance. However, to limit the scope of my investigation, the design restricts task queries to supporting automated perception, e.g., asking the human to analyze an image.

Finally, I wanted human-robot dialogue to directly affect system operation and autonomy. Specifically, I was interested in creating a system that would modify its behavior based on the answers (or lack thereof) given by the human in response to questions from the robot. The design, therefore, includes a robot controller that emphasizes awareness and adjustable autonomy. That is, when the robot is operating, it is always aware of to whom it is speaking, is able to determine whether a response has been given, and is capable of modifying its behavior (e.g., automatically increasing its level of autonomy) in case of non-response.

3.1.3 Architecture

I implemented collaborative control as a distributed set of processing modules, each of which operates independently and performs distinct functions. The modules are interconnected with an interprocess communication toolkit, FPC (described below), and communicate via a centralized message cache (the “FPC Server”).

The system architecture is shown in Figure 9. The current system has twenty-one modules in six principal groups, five of which connect directly to the FPC Server. The Safeguarded Robot Controller group contains sixteen robot modules, which are described in Section 5.4. Due to processing and data requirements (e.g., synchronous operation and communication), the UserInterface (Section 4.3) and MobileRobot (Section 5.2) modules have direct connections to the Safeguarded Robot Controller.

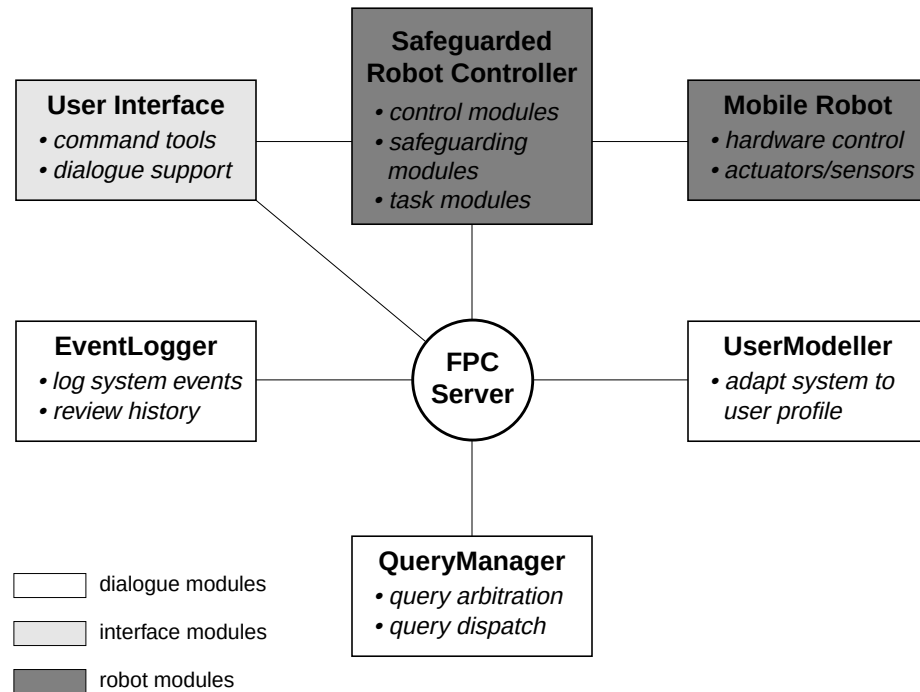


Figure 9. Collaborative control system architecture

Modules

As Figure 9 shows, there are three types of modules in the architecture: dialogue, interface, and robot. Dialogue modules are “system” level modules and serve as the “glue” between human and robot. Interface modules integrate the human into the system and allow him to work with the robot. Robot modules control how the robot operates and collaborates with the human.

Dialogue modules control human-robot communication and are responsible for user modeling, query arbitration, and message management. The *UserModeller* uses pre-defined, stereotype user profiles to adapt human-robot dialogue to a specific user. The *QueryManager* performs query arbitration and dispatch using an attribute-filtering scheme. The *EventLogger* records system events as they occur and provides historical playback. These three modules and the messages used for human-robot dialogue are described in greater detail in the following sections.

Interface modules (i.e., a user interface) mediate human-robot conversation and enable the human to converse with and to assist the robot. Interface modules provide feedback displays that allow the human to monitor the robot and to receive queries asked by the robot. Interface modules also process human input (commands, responses to robot queries), transforming and communicating the information to the robot. During the course of this thesis, I developed a variety of user interfaces for vehicle teleoperation (Fong, Thorpe, and Baur 2001a). One of these interfaces, the *PdaDriver*, is described in Chapter 4.

Robot modules are used for robot control, providing mechanisms for localization, planning, perception, and hardware operation. The design of robot controllers has long been an area of active research and debate in the robotics community. My approach has been to develop a safeguarded robot controller that supports various levels of robot autonomy and

that includes modules for safeguarding and specific task functions (e.g. motion detection for surveillance). Chapter 5 presents the design of this controller as well as the robot hardware (actuators, sensors, etc.) used in this thesis.

FPC interprocess communications

Interprocess communication toolkits have long been used to support distributed and parallel computing. Although there are a large number of general purpose communication libraries, very few are appropriate for robotic applications. This is because the suitability of a toolkit is determined not merely by how efficiently it can move data, but rather by how well its communication paradigm (messaging model) and functions match the data-flow of the robot architecture. Thus, numerous interprocess communication toolkits have been developed for robotics including IPT, NDDS, NML, TCA/TCX/IPC, and RTC (Gowdy 2000).

To support my collaborative control system, I have developed an interprocess communication toolkit called FPC (Fourth Planet 1998). FPC's design was inspired by both message-based (e.g., IPT) systems and information-based (e.g., NDDS) systems. FPC uses a publish and subscribe messaging model and a centralized message cache (the "FPC Server") for efficient, dynamically reconfigurable, and scalable data distribution. A detailed description of FPC is given in Appendix B.

FPC is well suited for collaborative control for several reasons. First, it provides both reliable (for message sequences) and unreliable (for fast idempotent data) delivery. Second, its performance (message rate and latency) meets or exceeds the requirements of the system modules. Finally, it facilitates integration of diverse modules with multiple language interfaces (C, Java, Perl, TCL) and support for multiple operating systems (Linux, WinNT, IRIX, Solaris, HP-UX).

3.2 UserModeller

The UserModeller module adapts human-robot dialogue to a specific user. It does this through the use of pre-defined, stereotype user profiles. Each profile describes a class of users through a set of attributes. To create the user profiles, I addressed the following design issues:

- *attributes*: What information about a user is required and how should it be represented?
- *definition*: What subgroups of the users (the stereotypes) are we going to define? How do we obtain information about each type of user?
- *use*: How will the system use information about different types of users (i.e., What useful behavioral adaptation can be made?)

3.2.1 Attributes

Although there are many attributes which can be used to describe teleoperation users, I have chosen to use the three listed in Table 1. I selected these attributes because they are well-suited for vehicle teleoperation in unknown environments and provide a sufficiently rich basis for experimentation.

Table 1. User attributes

Attribute	Description
response accuracy	estimate of how accurately the user can answer a question
expertise	estimate of task skill / domain knowledge the user possesses
query interval	indicates how often the user is available (or prefers) to answer questions

An estimate of the user's response accuracy is important for when the robot needs to ask a safety-critical question. If a user is highly accurate, then the robot can place greater confidence in the response.

Similarly, an estimate of the user's expertise (whether general or domain-specific) is valuable for adapting dialogue and autonomy. If a user does not have expertise, then it is not useful to ask questions requiring such.

Finally, query interval can indicate several things about a user: availability (how much time he can dedicate to responding), efficiency (how quickly he can answer), and personal preference (how often he prefers to be interrupted with questions).

3.2.2 Definition

To support evaluation of collaborative control, I defined three user stereotypes: novice, scientist, and expert. A novice is an inexperienced and untrained user: he does not know anything about teleoperation and has no special domain knowledge or task skill. Novices

do not answer questions well (i.e., their responses are inaccurate). They also have difficulty handling interruptions and resuming work once interrupted.

A scientist is also an inexperienced and untrained user. As with the novice, a scientist does not know anything about teleoperation. A scientist, however, does have domain expertise. Thus, a scientist can be expected to have difficulty answering teleoperation questions (e.g., “Is this an obstacle?”), but will be able to answer domain specific questions.

An expert is defined as a user who knows everything. Thus, an expert is experienced, has training, and has both teleoperation and task expertise. Furthermore, an expert understands how the system is designed, how it is implemented, and how it functions. An expert can answer all questions quickly and is skilled at simultaneously performing multiple tasks.

For each user stereotype, I assigned attribute values for response accuracy, expertise, and query interval. Response accuracy was taken to vary from 0 (inaccurate) to 100 (perfect). Expertise ranges from 0 (unskilled) to 100 (highly skilled). Query interval describes the minimum time required between subsequent queries.

Table 2. Stereotype attributes

Stereotype	Response accuracy	Expertise	Query interval (sec)
novice	30	30	15
scientist	50	100	15
expert	100	100	0

The assigned attribute values are shown in Table 2. The values were chosen based on the stereotype definitions and are somewhat arbitrary. Only the relative magnitudes between user stereotypes are significant. For example, novices are less accurate and have less expertise than either scientists or experts, thus the respective attributes are lower. In the current system, attribute values always remain constant, i.e., no adaptation or learning occurs during use.

3.2.3 Use

The UserModeller uses each profile to configure human-robot interaction in three ways. First, it modifies the user interface to fit the needs of each type of user. Each user profile defines which control modes are shown, what types of user input are allowed, and what types of feedback (displays) are presented. Second, the UserModeller controls how the QueryManager (Section 3.3) filters dialogue messages: only messages which are appropriate for each type of user are selected. Finally, the UserModeller modifies how the robot acts by configuring robot modules. Thus, the questions that the robot generates and how much independence (autonomy) the robot exhibits are user-dependent.

3.3 QueryManager

Under collaborative control, multiple robot modules may ask questions of the human at the same time. Thus, a collaborative control system needs query arbitration: a mechanism for choosing which questions to ask based on both immediate (local) needs and overall (global) strategy. In this collaborative control system, the QueryManager module performs this task with an attribute-based filtering scheme (Fong, Thorpe, and Baur 1999).

Whenever a robot has a question to ask the human, it sends a message to the QueryManager. A message is defined by user attributes (see Table 1), query attributes (type, priority level, expiration time), and question-specific data (image, text, etc.) Our collaborative control system currently supports two query types: *y/n* (user must answer *y* or *n*) and *value* (user must provide a decimal value).

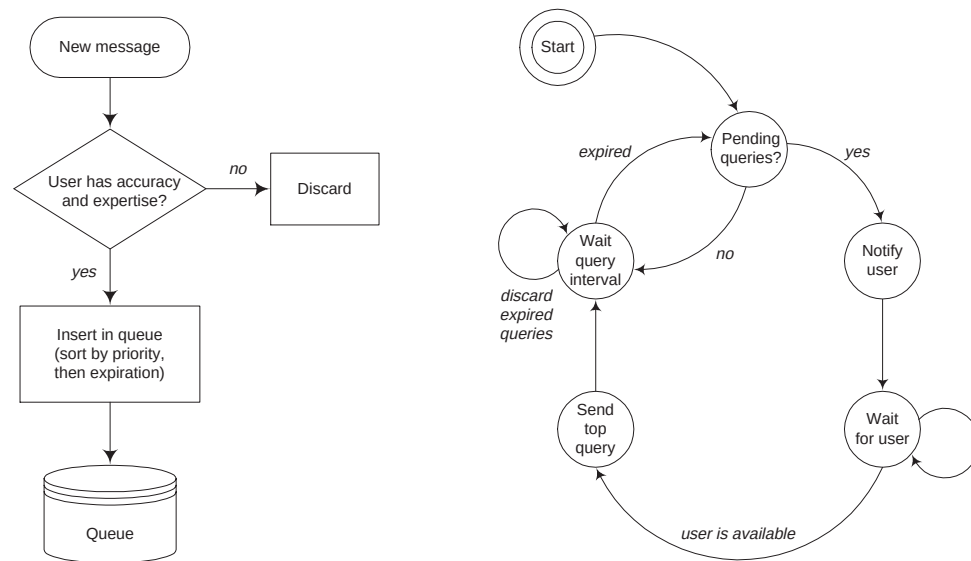


Figure 10. The QueryManager: *left*, attribute-based message filtering; *right*, message dispatch.

The QueryManager stores incoming messages in a queue, filtered by accuracy and expertise, and sorted by priority and expiration (Figure 10, left). Whenever the cache contains unexpired queries, the QueryManager notifies the human (via the User Interface) that there are pending queries. Then, when the human indicates that he is available to answer a question, the QueryManager sends the first message in the queue (Figure 10, right).

Because the queue is priority-sorted, urgent questions have preference. Expired questions are discarded (i.e., the user is never asked a question which is no longer valid). Once a question is asked, the QueryManager waits until the query interval has expired before repeating the process.

3.4 EventLogger

The EventLogger module is a historical data recording system. It performs two tasks: (1) records events in a structured fashion; and (2) retrieves stored events upon user request using criteria such as time, source, or description.

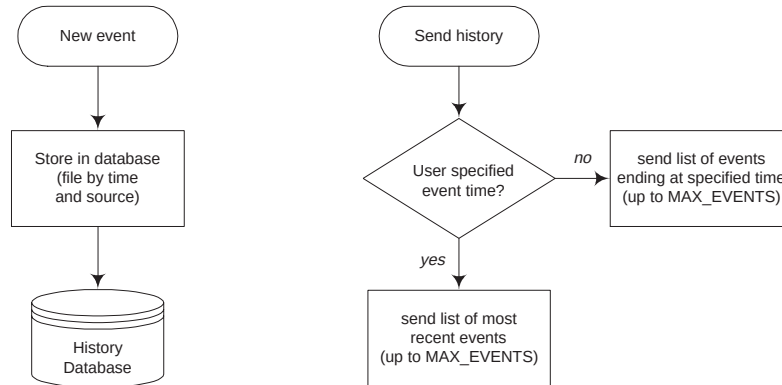


Figure 11. The EventLogger: *left*, event recording; *right*, event retrieval.

The EventLogger continuously monitors message traffic, captures key system events (as signaled by each module) and stores the event in a database, indexed by time and source (Figure 11, left). Whenever the user requests a review (e.g., based on event time), the EventLogger outputs a text-based history list (Figure 11, right).

Maintaining historical data is useful for a variety of reasons. Historical data provides an account of system operation for understanding what has occurred. This is particularly important for situations in which the operator is intermittently controlling or supervising the robot. Historical data can also be used as a memory and decision aid: to remember what has occurred as a basis for current decision making. Finally, historical data can be used for off-line analysis by designers, users, and organizations (regulatory, management, etc.). Having objective data is critical for producing unbiased assessments of performance, task success or failure, and user or system errors.

Van der Loos (1993) notes that historical information can greatly help limit the negative effect of interface breakdown in teleoperation. Breakdown, or loss of interaction context, occurs whenever the operator is interrupted and has to shift attention away from the interface during task performance. This may occur, for example, if a task requires a major operational change (equipment change-out) or if the user must perform multiple tasks in parallel. Historical information enables the operator to review previous actions, and thus to better “re-register” when resuming an interrupted task.

3.5 Human-Robot Dialogue

With collaborative control, dialogue arises from the exchange of messages between human and robot. Effective dialogue, however, does not require a full language, merely one which is pertinent to the task at hand and which efficiently conveys information. Thus, for this thesis, I do not use natural language and limit message content to vehicle mobility (navigation, obstacle avoidance, etc.) and task specific issues (e.g., confirmation of perception module discoveries).

In the current system, approximately thirty messages are used for vehicle teleoperation, as shown in Table 3. *Robot commands* and *information statements* are unidirectional (i.e., no acknowledgement is expected or required). A *query* (to either the human or robot) is expected to elicit a *response*, though the response is not guaranteed and may be delayed. It should be noted that Table 3 only describes the content of messages (what is said), and not the expression (how it is conveyed). Message expression is performed by the UserInterface (see Chapter 4).

Table 3. Vehicle teleoperation dialogue (user to robot)

	Category	Messages
user → robot	robot command (<i>command for the robot</i>)	position (<i>pose, path</i>) rate (<i>translate, rotate</i>) stop camera pose (<i>pan, tilt, zoom</i>) camera config (<i>exposure, iris</i>) sonar config (<i>polling sequence</i>)
	query-to-robot (<i>question from the user</i>)	How are you? Command progress?
	response-from-user (<i>query-to-user response</i>)	y/n value
robot → user	information statement (<i>information for the user</i>)	pose (<i>x, y, z, roll, pitch, yaw</i>) rates (<i>translate, rotate</i>) message (<i>event, status, query</i>) camera state (<i>pan, tilt, zoom</i>) get new image
	query-to-user (<i>question from the robot</i>)	see Table 4
	response-from-robot (<i>query-to-robot response</i>)	How are you? → bargraphs (<i>health, rollover, collision</i>) Command progress? → stripchart (<i>progress over time</i>)

Table 4 lists the queries that a robot can ask. Several of these queries have variable response accuracy levels, which shows that the importance of a question can change with time or situation. For example, the accuracy of the “Stopped due to high temperature. What should the safety level be?” query varies from 0 to 100 percent. An accuracy value of zero means that the robot is willing to accept any response (i.e., any safety level). High

accuracy values, however, indicate that the setting is critical to the robot's continued health. In other words, when the accuracy value is low, the robot is willing to ask any user (even a novice) for advice. But, when the accuracy is high (i.e., the response is critical), the robot will only ask the question to an expert.

Table 4. Query-to-user (questions from the robot): attributes

Query	Type	Priority	Response accuracy	Expertise
Can I drive through (<i>image</i>)? If you answer no, then I will stop right where I am.	y/n	1	50	50
Is this a rock (<i>image</i>)? If you answer 'y', I will stay here.	y/n	1	0	50
Motion detected. Is this an intruder (<i>image</i>)? If you answer 'y', I will follow him.	y/n	1	0	50
The environment is very cluttered (<i>map</i>). What is the fastest I should translate? The current setting is XX cm/s. I would recommend less than XX cm/s.	value	2	0-100	50
The environment is very open (<i>map</i>). What is the fastest I should translate? The current setting is XX cm/s. I would recommend less than XX cm/s.	value	2	0-100	50
My motors are stalled. Can you come over and help? If you answer no, then I will turn off motion control and will not be able to move.	y/n	1	0	0
Motion control is currently turned off. Shall I enable it? If you answer no, then I will not be able to move.	y/n	2	50	0
Safeguards are currently turned off. Shall I enable it?	y/n	1	50	50
Stopped due to high temperature. What should the safety level be (currently XX degrees C)?	value	1	0-100	0
Stopped due to low power. What should the safety level be (currently XX volts)?	value	1	0-100	0
Stopped due to rollover danger. Can you come over and help?	y/n	1	0	0

Several of the queries in Table 4 have non-zero expertise values. To answer these queries, the human must have a certain level of expertise. In the current system, I am only interested in distinguishing between trained (experienced) and novice users, i.e., I do not distinguish between different types of experts (e.g., skilled pilot vs. geologist). In general, however, we would use additional attributes to select queries for users who have specific task or domain expertise.

Some of the queries in Table 4 will expire after a period of time. All of the other queries remain valid indefinitely (i.e., until the user responds). Table 5 lists the queries that may timeout and the action that is subsequently taken on timeout.

Table 5. Query-to-user (questions from the robot): timeout and action

Query	Timeout (sec)	Action on timeout
Is this a rock (<i>image</i>)? If you answer ‘y’, I will stay here.	60	automatically resume motion
Motion detected. Is this an intruder (<i>image</i>)? If you answer ‘y’, I will follow him.	60	automatically resume motion
The environment is very cluttered (<i>map</i>). What is the fastest I should translate? The current setting is XX cm/s. I would recommend less than XX cm/s.	30	automatically adjust rate to recommended setting
The environment is very open (<i>map</i>). What is the slowest I should translate? The current setting is XX cm/s. I would recommend less than XX cm/s.	30	automatically adjust rate to recommended setting
Stopped due to high temperature. What should the safety level be (currently XX degrees C)?	30	if (<i>temp</i> < <i>max. safe temp</i>), automatically adjust safety to current (<i>temp</i> + 1.0 deg)
Stopped due to low power. What should the safety level be (currently XX volts)?	30	if (<i>power</i> > <i>min safe power</i>), automatically adjust safety to current (<i>power</i> - 0.2 V)

Timeouts are important because they allow the level of autonomy to dynamically vary. Specifically, if the user is unable to answer because he is occupied, or if the robot knows the user cannot answer (e.g., an expert is needed but only a novice is available), the robot can take the initiative to act. For example, if the human does not answer the “Stopped due to high temperature. What should the safety level be?” query within 30 seconds of it being asked, the robot will attempt to adjust the rate by itself.

3.6 Background

The design presented in this chapter was inspired and guided by ideas from three research areas: human-robot interaction, collaboration systems, and human-computer dialogue. The following section provides an overview of these areas.

3.6.1 Human-Robot Interaction

Human-Robot Interaction (HRI) can be defined as “the study of the humans, robots, and the ways they influence each other”. As a discipline, HRI regards the analysis, design, modeling, implementation, and evaluation of robots for human use. Although humans and robots have interacted since the 1940’s, the term “human-robot interaction” is very recent. In fact, prior to the 1990’s, the term did not appear in robotics research papers. Much of the current work in HRI centers on developing robots that perform human tasks and that can interact via natural language and gestures.

HRI is strongly related to human-computer interaction (HCI) and human-machine interaction (HMI). In both HCI and HMI, the emphasis is placed on understanding how humans model, perceive, and use interactive systems. HRI, however, differs from both HCI and HMI because it concerns systems (i.e., robots) which have complex, dynamic control systems, which exhibit autonomy and cognition, and which operate in changing, real-world environments. Moreover, unlike both HCI and HMI, HRI is not yet considered as a formal discipline.

Types of interaction

HRI may occur through direct, proximal interaction (e.g., physical contact) or may be remotely mediated by a user interface (Takeda 1997). In the latter case, the interface acts as a translator: it transforms human input (from hand-controllers or other control devices) to robot commands and provides feedback via displays. When the human and robot are separated by a barrier (distance, time, etc.) and information is exchanged via a communication link, then the interaction is usually referred to as teleoperation.

Milgram, Zhai and Drascic (1993) claim that human-robot communication can be classified into two languages: continuous and discrete. Continuous language represents continuously distributed spatial or temporal information, as exemplified by analogue displays and input devices such as mice and joysticks. Discrete language consists of independent elements such as programming languages, spoken commands and interface tools.

Sheridan (1997) notes that one of the challenges for human-robot communication is to provide humans and robots with models of each other. In particular, he claims that the ideal would be analogous to two people who know each other well and can pick up subtle cues from one another in order to collaborate (e.g., musicians playing a duet).

Human-like interaction

In recent years, much of the work in HRI has focused on making robots more “human”. Specifically, many researchers have been developing robots which perform tasks normally done by humans and which can interact via natural language and gestures. Social robots, for example, exhibit human traits (emotion, adaptation, etc.) and are increasingly capable of discerning and interpreting human intent. Because they are designed to provide natural, human-like interaction, social robots typically use direct, proximal modes of communication including gesturing (head and hand motion) and natural language.

Breazeal (1998) describes a motivational system for human-robot interaction. In this system, a human acts as a caretaker, assisting the robot in acquiring communication and social interaction skills. The organization of the framework borrows heavily from concepts in psychology, ethology and developmental psychology. More recently, Rogers and Wilkes (2000) describe a framework for human-humanoid robot interaction. The overriding goal for this system is to encourage “natural” interactions with people. Socialization involves having the robot communicate with a person concerning tasks and measuring the human’s satisfaction or frustration with the robot’s behavior.

3.6.2 Collaboration Systems

Collaboration systems is the term I use to describe research in human-machine collaboration. Unlike “Computer Supported Cooperative Work” (CSCW), which emphasizes software for human workgroups, collaboration systems focuses on enabling computers and robots to assist and collaborate with humans.

Collaboration is a process in which two or more parties work together to achieve shared goals. For collaboration to be effective, there must be: agreement of the goals to be achieved, allocation and coordination of tasks to be performed, shared context (keeping track of what has been done and what remains), and communication to exchange information and solutions. Numerous researchers have explored ways to enable humans to collaborate with computers and robots.

Human-computer collaboration

Terveen (1994) writes that there are two major approaches to human-computer collaboration. Human Emulation (HE) is closely tied to the artificial intelligence domain and assumes that the way to get computers to collaborate with humans is to endow them with human-like abilities, to enable them to act like humans. Human Complementary (HC) was developed by HCI researchers and assumes that computers and humans have fundamentally asymmetric abilities. The focus of HC, therefore, is to improve human-computer interaction by making the computer a more intelligent partner.

HE emerged from attempts to understand and model human communication. Most HE work, such as expert systems for trip planning or medical diagnosis, involve natural language processing (text and spoken dialogue). Consequently, much emphasis has been placed on designing representation formalisms (beliefs, goals, plans, etc.) and on developing techniques for systems to model and adapt behavior to different users.

HC is more recent than HE and is also more direct and pragmatic. Much of the work in HC has focused on developing interface technology to make human-computer communication more efficient and natural. This requires understanding human cognitive and perceptual abilities, how people work as individuals and in groups, and how to represent and present information. The HC approach has proven to be effective for student tutoring and for user critiquing.

Human-robot collaboration

Humans and robots have been working together since the 1940’s. At first, this interaction was primarily uni-directional: simple on-off switches or controls for operating manipulator joints and remotely piloting vehicles. However, as robots have become more intelligent over time, this relationship has changed to be more like the relationship between two

human beings. Consequently, humans and robots now communicate and collaborate in a multitude of ways (Sheridan 1997).

As with human-computer systems, the two fundamental approaches to human-robot collaboration are HE and HC. This is not surprising, given that all modern, “intelligent” robots are computer based. To date, both approaches have been used successfully and there no clear consensus has emerged about which is better suited for robotics. In fact, many systems have combined elements of both approaches and thus exhibit both human-like and “robot partner” qualities.

Personal service robots, for example, are designed to assist people in daily living. These robots perform many tasks in the same manner as humans, but also provide complementary services (e.g., computer-based functions). In recent years, service robots have been used as caretakers for the elderly (Baltus et al. 2000), assistants for the physically disabled (Green et al. 2000), and for indoor tasks such as tour guiding (Nourbakhsh et al. 1999). In these systems, HRI is mediated in a variety of ways: graphical user interfaces, natural language, and actuated displays (e.g., a mechanical “face” used to express emotion).

Recently, some researchers have begun studying how humans and robots can function as an integrated team (e.g., Laengle, Hoeniger, and Zhu 1997). When humans and robots work as a single unit, jointly participating in planning and problem solving, close interaction and effective communication are essential. Thus, much effort is currently focused on developing techniques to disambiguate human-robot communication (Perzanowski et al. 1999; Perzanowski, Adams, Schultz, and Marsh 2000), to facilitate collaboration and task specification (Simsarian 2000), and to adjust the level of robot autonomy as a function of situational or interaction need (Bonasso 1999).

3.6.3 Human-Computer Dialogue

Human-computer dialogue concerns the nature of communication between humans and computers. Dialogue is a joint process of communication between two or more parties: it requires sharing of information (data, symbols, context) and of control. Depending on the situation (task, environment, etc.), the form or style of dialogue will vary. However, studies of human conversation have revealed that many properties of dialogue, such as initiative taking and error recovery, are always present.

Communication and conversation

Lansdale and Ormerod (1994) describe five factors that control dialogue:

- *Linguistic competence* is the ability to construct intelligible sentences and to understand the other’s speech. Human-computer dialogue requires a vocabulary that associates labels with concepts and sequences of actions (i.e. grammar).
- *Conversational competence* is the pragmatic skill necessary for successful conversation. Human-human dialogue tends to proceed with greater ease than human-computer dialogue due to mechanisms such as inference. A user interface, therefore, must be designed such that users can unambiguously impart intentions and receive feedback.
- *Nonverbal skills* such as gestures are used for turn-taking, to differentiate between statements and questions, etc. These skills add coherence to a dialogue, serve as triggers for adaptation, and provide redundant information.

- *Medium constraints* such as communication links force a number of modifications on user behavior (slowing down, confirmation, etc.) With user interfaces, input devices and displays mediate conversation, thus technical limitations directly influence the nature of the dialogue.
- *Task constraints* can determine the structure of dialogue. For example, operative languages such as military communication and air traffic control use restricted vocabulary and economical grammar (e.g., acronyms) to avoid misunderstandings and to efficiently communicate complex information.

Perhaps the strongest factor which influences the structure and flow of dialogue is expertise. Studies of experts have consistently shown that the ability to assimilate and process information depends largely upon how much is already known about the structure of that information. This is because prior knowledge allows experts to categorize and compress incoming information (which may appear meaningless and uncorrelated to novices) into known patterns (Lansdale and Ormerod 1994).

As a consequence, when experts and non-experts (e.g., doctor and patient) engage in dialogue, the expert (the doctor) usually takes initial control of the conversation. The ensuing exchange tends to follow a question-answer sequence. But, when both parties are experts, control is shared and the dialogue is more focused. This dichotomy appears in user interfaces as well. In particular, many human-machine dialogues are designed: (1) to give experts more control than novices and (2) to ensure that a specific procedure is followed when users cannot be trusted to do so by themselves.

Dialogue management

Unless the human-computer dialogue is very simple, some form of dialogue management is required. The basic function of dialogue management is to translate user requests into a language the computer understands and the system's output into a language that the user understands (Goren-Bar 2001). In addition, dialogue management must be capable of performing a variety of tasks including adaptation, disambiguation, error handling, and role switching (Abella and Gorin 1999).

Role switching occurs because at any stage in a dialogue, one participant has the initiative (control) of the conversation. In a sense, initiative is a function of the participants' roles. Dialogue systems may allow the user or the computer to take the initiative, or may allow both to switch roles as required. The hardest dialogues to model, by far, are those in which the initiative can be taken at any point in the dialogue (Churcher et al. 1997).

The two most common methods for dialogue management are frames and graphs. With frame-based management, database queries drive the dialogue. Each frame represents a "chunk" of dialogue as a set of data slots and methods. Methods may be invoked automatically when data is added or becomes needed. This allows a dialogue system to be built in which the filling of required data slots can be achieved without having to specify a precise order of dialogue events (Young and Proctor 1989).

With graph-based management, dialogue consists of a series of linked nodes, each containing a limited number of choices the user can make (e.g., "For reservations, press 2"). Since the dialogue is structured and because the system always has the initiative, errors are limited. Graphs are less flexible than frames (i.e., they do not allow as much role switching), but are also more robust to errors as a result.

User model

Dialogue cannot make sense unless the user and the system have a reasonable understanding of each other. Given a user model, dialogue adaptation can be performed by referring to a user's expertise, knowledge, and preferences (Goren-Bar 2001). For example, the way in which information is collected (filtering, classification, etc.) and presented (text, graphics, speech) can be adapted to the user.

A user model (or profile) contains a set of attributes which describe a user or a group of users. Because dialogue is highly situation and task dependent, the attributes contained in a user model vary from system to system. Some common attributes are: skill in using the system, knowledge or expertise level, personal preferences, type of information needed or wanted, and responsiveness.

The *stereotype* approach is the most popular user modeling method. With the stereotype approach, a designer defines appropriate subgroups of the user population (the stereotypes), identifies user behaviors that enable a system to categorize users into a subgroup, and represents the set of features that characterizes each stereotype (Terveen 1994).

Effective construction of a user model is an open research area. Information about users may be acquired explicitly (by questioning the user) or implicitly (inferring information based on user actions). The former approach can be time consuming and users may not accurately characterize themselves. The latter approach is often impractical because it is difficult to determine when a user is starting a new task (Goren-Bar 2001; Terveen 1994).

3.7 Summary

In this chapter, I have described the design of a collaborative control system, which was inspired by work in human-robot interaction, collaboration systems, and human-computer dialogue. There are three types of modules (dialogue, interface, and robot) in the system, which are interconnected by an interprocess communication toolkit.

The system contains three dialogue modules that govern human-robot communication. The UserModeller uses pre-defined, stereotype user profiles to adapt the system to specific users. For this thesis, I have defined and use three stereotypes (novice, scientist, and expert). The EventLogger gathers and stores system events for future playback. It provides a structured mechanism for recording historical data of system operation. The QueryManager manages dialogue. It performs query arbitration to select which questions from the robot are presented to the human.

PdaDriver: A User Interface for Collaborative Control

In order for a human to participate in collaborative control, we need a user interface that enables him to communicate with, and to assist, the robot. In the last chapter, I presented the design principles for building a collaborative control system. In this chapter, I describe how these principles translate into user interface design.

I begin with a discussion of the interface requirements imposed by collaborative control and vehicle teleoperation. Next, I describe the design and implementation of an interface, PdaDriver, which meets these requirements (see Figure 12). At the end of the chapter, I conclude with some remarks on modern user interface design methods, focusing on HCI techniques and their use in robotics.



Figure 12. Left, PdaDriver; right, field-work with PdaDriver.

4.1 Requirements

In the system design described in Chapter 3, a user interface mediates human-robot conversation. It does this by providing feedback displays that allow the human to monitor the robot and to receive queries asked by the robot. A user interface also processes human input (commands, responses to robot queries), transforming and communicating the information to the robot.

For the user interface to be effective, it needs to be designed in a structured manner. The design principles outlined in the previous chapter provide a good starting point. However, because they served as general guidelines for the overall system, these principles are not, by themselves, sufficiently detailed not adequate for interface design.

My approach, therefore, was to identify and define additional detailed requirements by considering three areas of need. First, and foremost, the interface needs to satisfy the demands of collaborative control. Second, the interface must be appropriate for the problem context: vehicle teleoperation in unstructured, unknown environments. Finally, the interface has to support specific tasks within this context. Let us examine each of these areas in turn.

4.1.1 Collaborative Control Needs

For collaborative control, the interface needs to facilitate human-robot communication. That is, the interface should enable the human to express intent, to provide information, and to understand and interpret what the robot has accomplished. At the same time, the interface must permit the robot to make requests, to ask for help, and to directly interact with the human.

Specifically, the interface must be able to transform messages from the human/robot into a format that can be readily understood by the robot/human. The proper expression of a message is extremely important: if a message is not presented appropriately, communication will be inefficient. As a consequence, since vehicle teleoperation is often visual, and because situational awareness demands spatially data (e.g., maps), the interface should be capable of displaying message content ranging from text to images.

Additionally, as described in Section 3.5, there are numerous dialogue messages: commands, information statements, queries and responses. These messages are communicated at varying rates and directions. Queries, for example, are asynchronous (i.e., they are generated by the user or robot as needed). Thus, for queries, the interface has to provide mechanisms for capturing and relaying the response. On the other hand, information statements (e.g., robot status) are uni-directional, periodic and idempotent. For these messages, the interface must be able to handle continuous, possibly real-time, display updates.

Finally, to support the stereotype users described in Section 3.2.2, the interface needs to provide appropriate tools. Since these users have different backgrounds (knowledge, training, etc.) and available resources (control time, expertise, etc.), it would be impractical, or perhaps impossible, to provide tools which are optimal for all. A more pragmatic approach is to provide a set of tools which can be used by the most skilled user class (i.e. expert) and to choose subsets of these tools for use by the other classes. For example, if the interface supports multiple modes, experts would be able to use all the modes, but novices would only have access to a subset.

4.1.2 Vehicle Teleoperation Needs

A vehicle teleoperation interface enables remote driving tasks to be performed. To be effective, the interface must be well-suited both for the intended operators (users) as well as for the work to be performed. In particular, the interface must provide tools and displays that enable the operators to efficiently and easily perform tasks.

To perform vehicle teleoperation in unstructured, unknown environments, the interface must support flexible control. This is because control needs will vary as a function of location, situation, and task. For example, cross-country navigation and precision maneuvering have considerably different characteristics. Since no single command-generation method is optimal for all conditions, the interface must have a variety of control methods.

In addition, the interface needs to support human perception and decision making. To do this, the interface should provide displays that help the user to understand the remote environment and to maintain situational awareness. Moreover, regardless the level of user expertise and training, the interface should emphasize usability. An easy-to-use interface enables all operators to perform tasks more efficiently and with less effort.

4.1.3 Application Needs

To evaluate collaborative control, I have examined several vehicle teleoperation applications (see Chapter 6) and conducted a user study (Chapter 7). In these studies, the key tasks were remote driving and perception (human and autonomy-aided). Furthermore, no manipulation or other remote action (e.g., payload deployment) was required. The interface, therefore, needed to support navigation, positioning, and motion control. It also had to allow the user to interact with and to control autonomous perception modules.

Additionally, two of the applications focused on human-robot collaboration in the field. The first, collaborative exploration (see Section 6.2) concerned a human and robot performing planetary surface exploration. Thus, to support human-robot interaction in a field setting, the interface needed to be field-portable and to require minimal infrastructure.

The second field application, multi-robot remote driving (see Section 6.3), investigated how a single operator can use multiple mobile robots to perform reconnaissance and surveillance tasks. For this work, therefore, an important feature was that the interface aid the operator in switching contexts and in generating commands.

4.1.4 Requirements Summary

To summarize, the requirements for a user interface are:

Collaborative Control

- high usability (to support a wide range of users)
- support varied message types and rates
- support varied message content (text, image, etc.)
- provide tools which can be subsetting for various user classes

Vehicle Teleoperation

- multiple control methods for remote driving
- facilitate situational awareness
- efficient command generation
- emphasize usability

Applications

- ability to interact with and control autonomous perception modules
- facilitate navigation
- lightweight, field-portable
- support context switching

I should note that these requirements are intentionally *qualitative* in nature. Although it would have been possible to define quantitative specifications (display size, update rate, etc.), I do not believe that such requirements would have greatly aided interface develop-

ment. This is because the underlying purpose of the interface is to explore a new concept (i.e., collaborative control), rather than to achieve specific performance goals. However, given a specific task, it would be appropriate to define quantitative objectives in order to better guide interface development and evaluation.

4.2 Approach

Based on the above requirements, I created a user interface for collaborative control, which I call the PdaDriver. The interface was developed using a combination of heuristic design, heuristic evaluation, and cognitive walkthrough (Fong, Cabrol, Thorpe, and Baur 2001; Fong, Conti, Grange, and Baur 2000). I chose these methods because they are efficient (low-cost, rapid, easy to perform), can be used throughout the design process, and have been proven to produce high quality interfaces in a variety of domains.

Heuristic evaluation, or “usability inspection”, is an informal review method (Newman and Lamming 1995). With this method, a team of reviewers use a set of heuristics to critique an interface. The result is a list of problems (design flaws, implementation errors, etc.) which could reduce usability. The primary weakness of heuristic evaluation is that problem identification (types, coverage, etc.) is strongly correlated to the reviewers’ experience and fastidiousness.

Cognitive walkthrough is an evaluation method which simulates the way users explore and gain familiarity with an interface. During cognitive walkthrough, evaluators perform a structured examination by working step-by-step through the performance of a task, trying to predict the actions a user will take. This produces considerable insight into the design, especially regarding ease of use and likelihood of user errors.

Table 6 lists the heuristics I used during interface design and evaluation. I found that the PDA-related heuristics (“single-click interaction” and “design for small screen”) were the hardest to follow. This is not surprising, given the difficulty of creating effective PDA interfaces. In addition, I debated over the use of multiple modes. Although I would have preferred a modeless interface to speed learning and habituation, I do not believe that there is a single, optimal method for all vehicle teleoperation tasks. Thus, I chose to create task-specific modes, each with distinctive appearance and function to minimize modal errors.

Table 6. Design and evaluation heuristics

heuristic	meaning / effect
single-click interaction	facilitate PDA input avoid text entry where possible
design for small screen	present only task-relevant information
effectively use color (Tufte 1990)	avoid excessive hue use familiar color mapping
consistency	handle input and display feedback consistently
simplicity	make tasks easy to perform make displays easy to understand
task-specific modes	create separate modes for each distinct task

4.3 Architecture

PdaDriver provides a variety of command modes, enables human-to-robot dialogue, and supports human-to-human interaction (audio and video). The current version supports simultaneous (independent) control of multiple mobile robots and runs on WindowsCE¹-based PDA's such as the Casio Cassiopeia². I chose WindowsCE-based PDA's because they provide high quality color display (12-bit or 16-bit) and support high-bandwidth communication (wired and wireless ethernet). The PdaDriver architecture is shown in Figure 13.

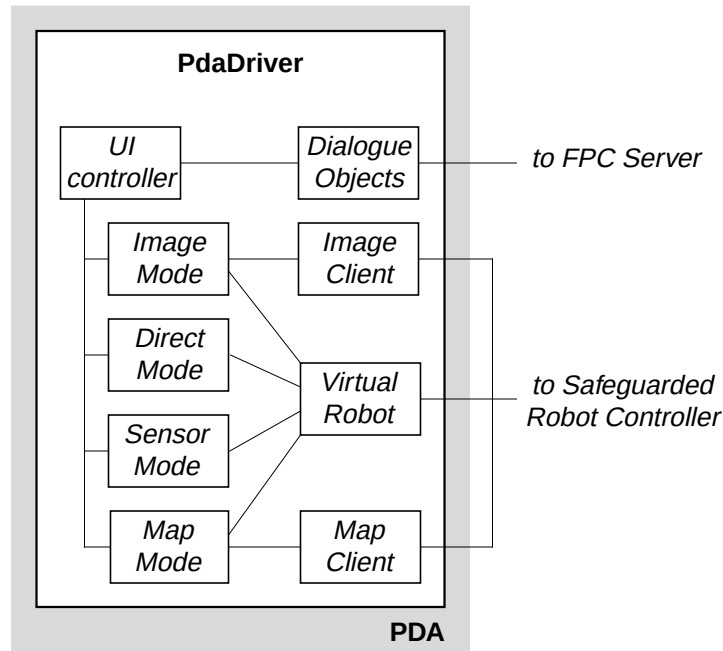


Figure 13. PdaDriver architecture

PdaDriver is implemented using Personal Java³, a Java application environment designed for personal consumer devices. The top-level Java object is the UI controller, which operates the user interface. The UI controller performs a variety of tasks including interface initialization/shutdown, object activation, and communication link monitoring. The other primary interface objects are the command and dialogue modes (described below) and the Virtual Robot.

The VirtualRobot encapsulates robot-specific parameters and maintains a local copy of robot state (pose, sensor readings, etc.). This enables interface objects to have continuous access to robot data. The VirtualRobot also provides coordinate frame (sensor, world, local) conversion and standard vehicle command methods. When the user issues a command, the object receiving the input invokes a method in the Virtual Robot, which then outputs a command to the robot controller. This level of indirection facilitates integration and use of different mobile robots.

1. WindowsCE is a trademark of Microsoft, Inc.

2. Cassiopeia is a trademark of Casio, Inc.

3. PersonalJava and Java are trademarks of Sun Microsystems, Inc.

4.4 Dialogue Displays

With collaborative control, dialogue results from messages exchanged between the human and the robot. PdaDriver transforms these messages to/from a set of *dialogue displays*. In particular, it enables both the human and robot to ask questions of the other and to receive the responses.

4.4.1 Robot to User

The current collaborative control system supports a variety of *query-to-user* messages related to robot health, configuration, and task performance (see Table 4 in Section 3.5). These questions are categorized into two types based on the expected response: *y/n* (requires y or n response) and *value* (requires a decimal value response).

Whenever the robot has a question to ask the human, it sends a message to the QueryManager (see Figure 13). If there are already one or more pending questions, the QueryManager performs query arbitration to select which question should be asked first. The QueryManager then notifies the PdaDriver, which flashes the query indicator (see Figure 18) to indicate that a question from the robot is waiting to be answered.

Figure 14 shows two queries to the user as displayed by PdaDriver. In the *y/n* query on the left, the robot is asking if the human can provide assistance. If the human answers ‘n’, the robot must then try to resolve the problem by itself. In the *value* query on the right, the robot is asking for guidance in setting a safeguard threshold.

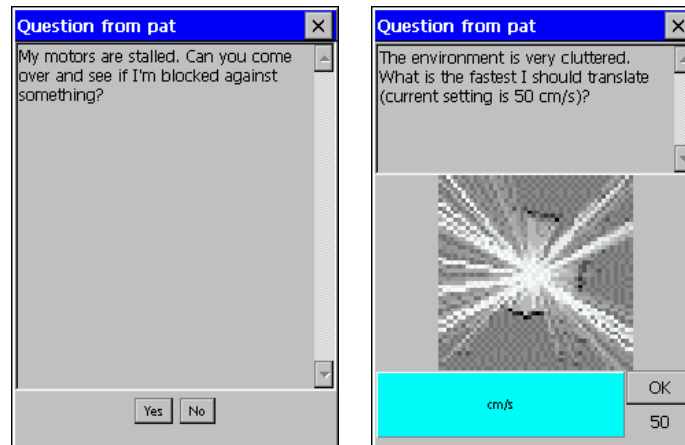


Figure 14. Queries to the user

4.4.2 User to Robot

In PdaDriver, most human to robot dialogue is expressed via the command modes (Section 4.5). This is because, under normal conditions, the human gives commands to the robot and receives feedback through the command displays. There are times, however, when the human may need to directly query the robot. In particular, if the robot has been operating autonomously without monitoring, or supervision, for a long period of time, the human may want to quickly learn how the robot is doing.

Consequently, PdaDriver provides a button for the user to ask questions to the robot (Figure 18). At present, three questions (*query-to-robot*) can be asked: “How are you?”, “What is the progress with the current command?”, and “What is the system status?”. Figure 15

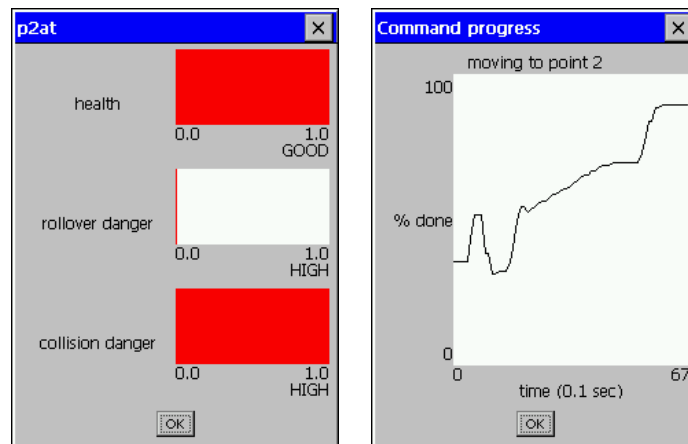


Figure 15. Responses from the robot

shows the robot's response to two of these questions. The left image shows the response to "How are you?": the robot's current health and safeguard metrics. The right image presents the response to "What is the progress with the current command?": a graph showing percent completion of the current task.

4.4.3 Event Replay

As I discussed in Section 3.4, the EventLogger module maintains a log of system events. PdaDriver provides a button (Figure 18) for reviewing this log, so that the user can review what human-robot dialogue has occurred and can understand what autonomous actions the robot has taken. This type of review helps the user maintain situational and contextual awareness, especially when he is only intermittently interacting with a robot or when he is controlling multiple robots.

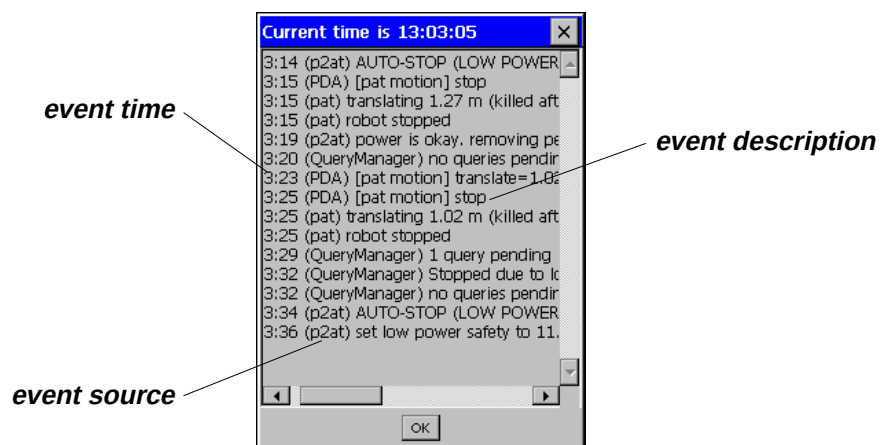


Figure 16. Display of system events captured by the EventLogger

Figure 16 shows a portion of the system log captured during multi-robot operation. Each line of the log displays a single event: the time it occurred, the source of the event, and a brief description. At present, the log displays events in the order they occurred (i.e., time-indexed). An obvious enhancement would be to add support to display events sorted by other criteria such as source or event type.

4.5 Command Modes

PdaDriver provides a variety of command modes (Figure 17). Three of these modes (direct, image, and map) enable the user to generate robot motion commands and to navigate. The fourth mode (sensor) allows the user to control sensing and perception.

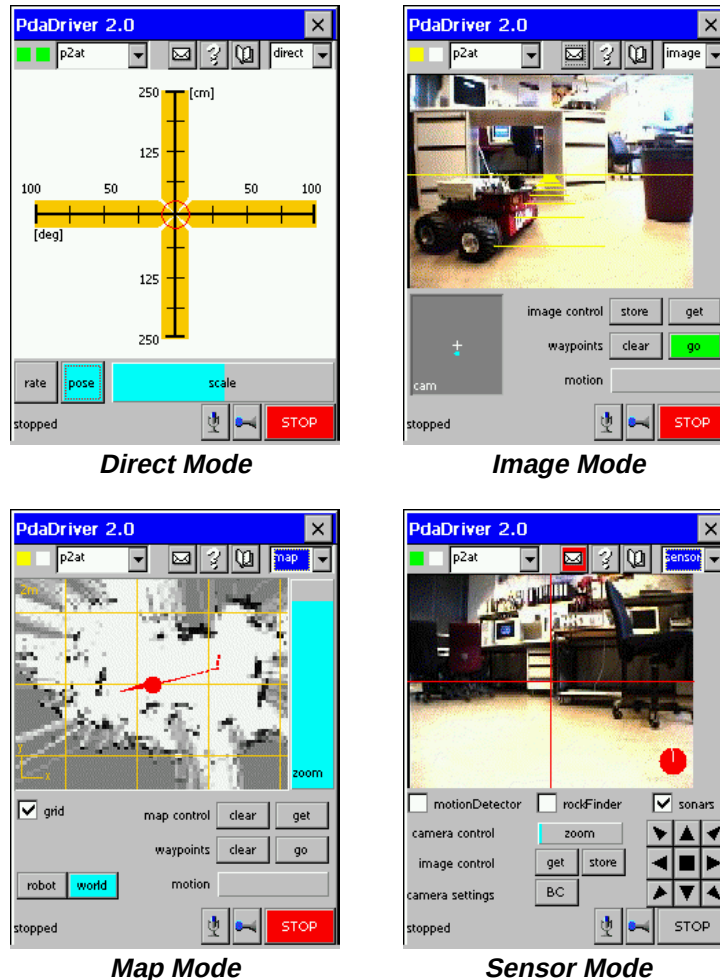


Figure 17. PdaDriver control modes

All modes share two common “button bars” (Figure 18). The top bar contains a communication link status display, buttons for managing human-robot dialogue, and a drop-down menu for switching modes. The bottom bar contains a robot status display, buttons for remote human communication, and a button for stopping the robot at any time.

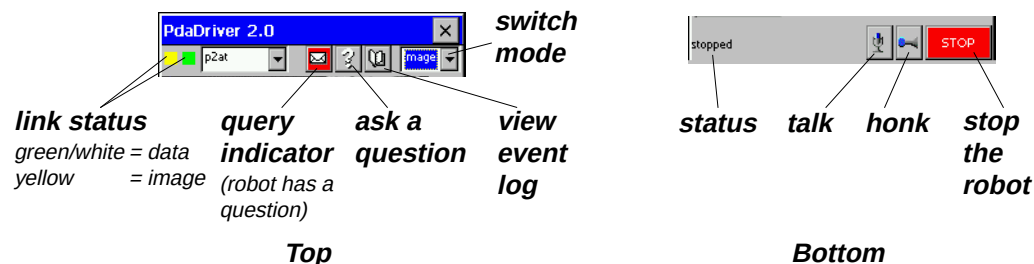


Figure 18. PdaDriver button bars

4.5.1 Direct Mode

Direct mode provides pose and rate control. This mode is most appropriate for precision driving tasks such as fine worksite positioning. Direct mode is also useful for specifying constant rates (e.g., forward translation) when long-distance or long-duration motion must be performed.

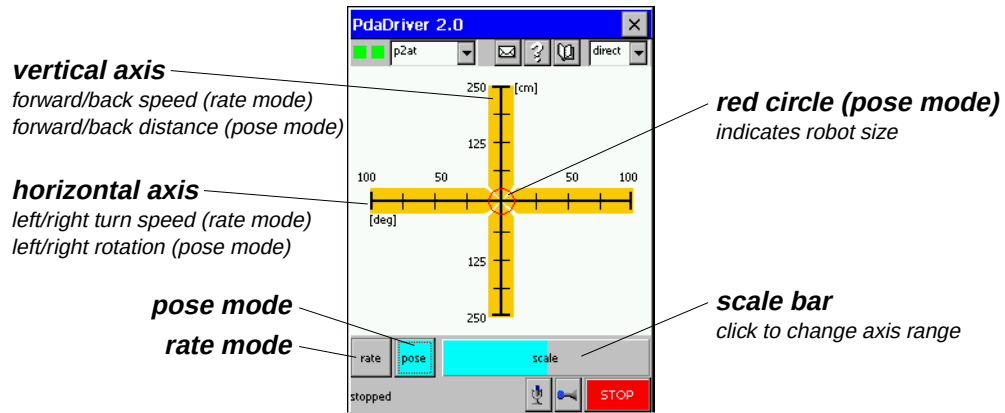


Figure 19. Direct mode provides position and rate control of robot pose.

In direct mode, a graduated cross is shown (Figure 19). Clicking on the vertical axis commands translation and on the horizontal axis rotation (all motions are performed in the robot's coordinate frame). The *rate* and *pose* buttons allow the user to switch between rate and relative position control. A scale bar is used to change command magnitude. The center circle (only shown in pose mode) indicates the size of the robot.

4.5.2 Image Mode

Remote driving is an inherently visual task, especially for unstructured or unknown terrain. Thus, image mode enables image-based, waypoint driving. Image mode was inspired by Kay (1997), but has two significant differences. First, instead of continuous ground-plane reprojection, image mode uses a flat-earth (planar world) model. This greatly simplifies computation, yet works well over short distances. Second, rather than Kay's ad-hoc approach, image mode uses Tsai's (1986) method for camera calibration. This method accurately corrects first-order radial distortion and allows use of wide-angle lenses (70° HFOV).

Image mode (Figure 20) displays images from a robot-mounted camera. Images are retrieved using an event-driven model to minimize bandwidth usage (Fong, Thorpe, and Baur 2001b). Horizontal lines overlaid on the image indicate the projected horizon line and the robot width at different depths. The user is able to position (pan and tilt) the camera by clicking in the lower-left control area. The user drives the robot by clicking a series of waypoints on the image and then pressing the *go* button. As the robot moves, a status bar displays the robot's progress.

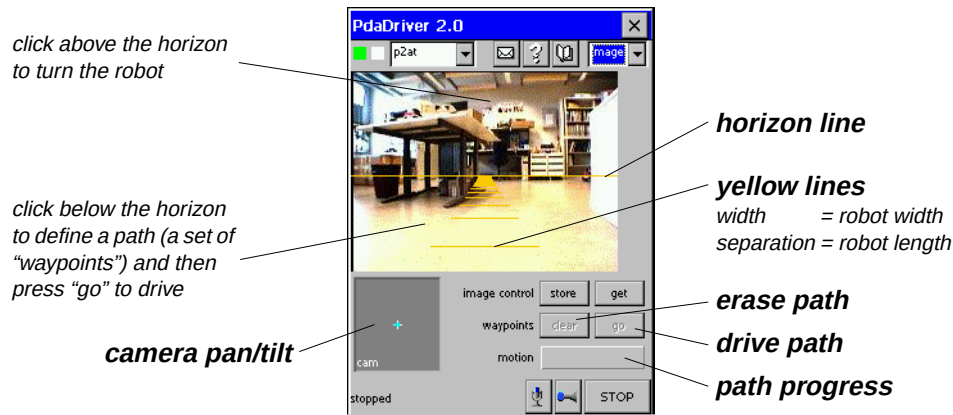


Figure 20. Image mode is designed for image-based, waypoint driving.

Camera model

To aid the operator's perception of the remote environment, image mode is designed to work with color CCD cameras and wide-angle lenses. Tsai's (1986) camera model and calibration technique is used to correct for the optical distortion inherent with these lenses and to obtain a precise estimate of focal length. Tsai's model is based on pinhole perspective projection and incorporates five intrinsic and six extrinsic camera parameters.

For this thesis, two cameras were used (see Section 5.2). Since both cameras have the same size CCD and because the video signal (Square NTSC, 640x480 pixels) is digitized using identical framegrabbers, the only calibration parameters that differ between the two systems are focal length and first-order radial distortion coefficient.

Flat-earth projection model

To transform image points to world points (and vice versa), image mode uses perspective projection based on a pinhole camera model. It assumes that the ground plane is locally flat and that it is parallel to the camera central axis (for zero camera tilt). Forward projection is performed as follows:

1. compute undistorted coordinates (Tsai dewarp)
2. transform from image to CCD sensor plane
3. project from sensor plane to camera frame
4. transform from camera frame to world frame

Although this procedure computes 3D world points, only 2D coordinates (i.e., ground points) are used for driving.

Designation error

There are many factors that affect the precision of waypoint designation, and consequently, driving accuracy. Sources of systemic error include camera calibration (imaging errors and camera mount pose uncertainty), vehicle localization (accuracy, repeatability, etc.), and projection uncertainty (perspective projection assuming planar world). Each of these factors may produce sizeable uncertainty (e.g., downrange and crossrange error), especially for waypoints far from vehicle. See Kay (1997) for a detailed discussion.

For PdaDriver, stylus input has a considerable impact. Unlike mouse-based interfaces, PDA's do not show a cursor to provide position feedback to the operator. Point selection (screen tap) is, therefore, essentially open loop. Additionally, stylus calibration can only partially compensate for touchscreen misalignment and irregularities. Thus, pointing precision may vary considerably across the display.

The most significant factor, however, is the projection uncertainty caused by limited image resolution and forward-mounted (i.e., zero tilt) cameras. Because PdaDriver is constrained by display hardware to low-resolution (208H x 156V) images, each pixel projects to a large ground-plane area. Moreover, with perspective projection and low-mounted cameras with low tilt, image points may be transformed to 3D points with high uncertainty.

Figure 21 shows how projection uncertainty (downrange error) varies with vertical pixel position for zero camera tilt. We can see from this graph that pixels near the image center may cause large driving errors. For example, at 70 pixels from the image bottom (i.e., near the vertical center of image), the downrange projection uncertainty is almost 1 m.

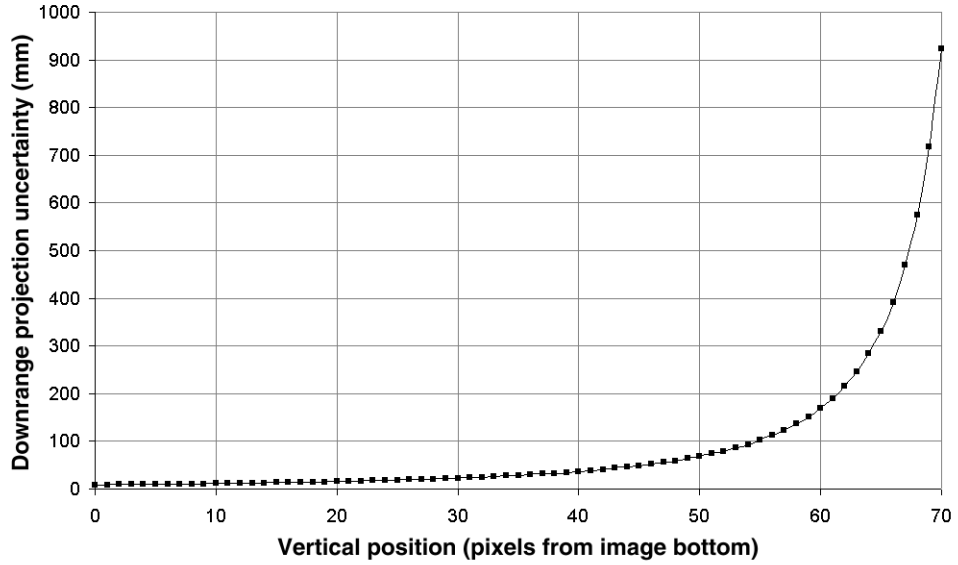


Figure 21. Projection uncertainty (per pixel) for zero camera tilt

Of course, the impact of this uncertainty depends on the task the robot is required to perform. For cross-country navigation (remote driving through an unknown and/or unstructured environment), such errors may be acceptable. For precision driving tasks (maneuvering in cluttered spaces, sample collection, etc.), however, designation errors that exceed the robot's dimensions are likely to cause problems¹.

1. The mobile robots used in this thesis have a central radius of 360 mm. Thus, waypoints designated near the vertical center of image are problematic for precision driving.

4.5.3 Map Mode

Although image-based driving is an efficient command mechanism, it may not provide sufficient contextual cues for good situational awareness. Maps can remedy this by providing reference to environmental features, explored regions and traversed path (Fong, Thorpe, Baur 2001b).

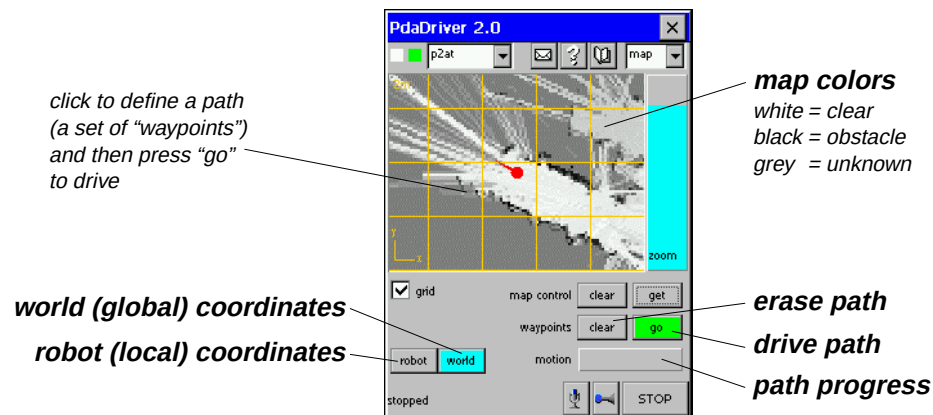


Figure 22. Map mode is designed for map-based, waypoint driving.

Map mode (Figure 22) displays a map in either robot (local) or world (global) coordinates. The map is built by the robot controller using a 2D histogram-based occupancy grid and sensor range data (see Section 5.4.5). As with image mode, the user drives the robot by clicking a series of waypoints and then pressing the *go* button. As the robot moves, the traversed path is shown in the display.

Map mode complements image mode: waypoints entered in map mode appear on the image mode display and vice versa. In addition, map mode is useful for entering waypoints which are problematic in image mode (outside the field-of-view, near the image horizon, etc.).

4.5.4 Sensor Mode

One of the primary difficulties with vehicle teleoperation in unknown or unstructured environments is that it is difficult to specify, *a priori*, the parameters required for perception. This is because environmental characteristics may significantly vary with time, location, and situation. For instance, outdoor scene illumination can rapidly and dramatically change due to sun position, the presence of shadows and scintillating objects, as well as other factors such as fog, mist, and dust. In addition, sensor characteristics may change with time and use. Long exposure to intense illumination, for example, may degrade the performance of CCD arrays.

As a consequence, *Sensor Mode* (Figure 23) was designed to enable direct control of sensing and perception. Sensor Mode complements the other control modes, i.e., it is not used to generate motion commands. Instead, Sensor Mode allows the user to configure, enable, and disable specific robot sensors and robot perception modules.

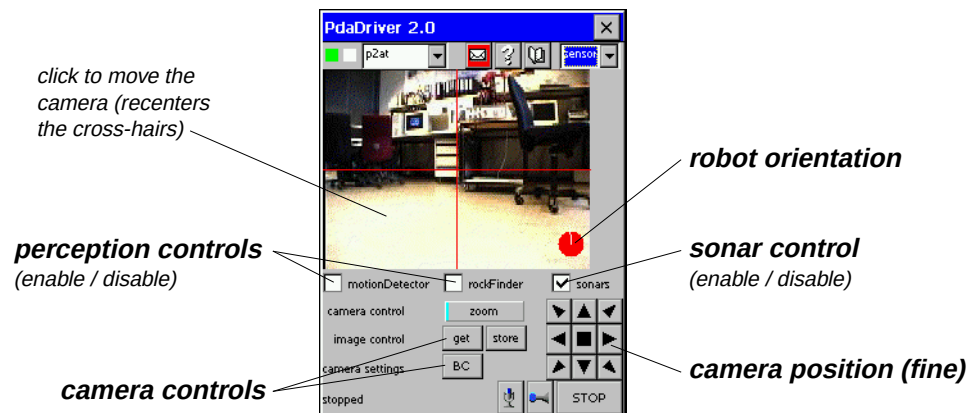


Figure 23. Sensor mode is used to control sensors and robot perception.

At present, Sensor Mode enables the user to control the robot's camera (position and imaging settings) and sonar array. The ability to directly modify camera gain settings, such as backlight compensation, is particularly important when the robot is operating in brightly lit, high-contrast scenes. Sensor Mode also allows the user to enable (or to disable) two perception modules, MotionDetector (see Section 6.3.2) and RockFinder (see Section 6.2.2), as needed for task performance.

4.6 A Brief Review of Interface Design

From a HCI perspective, the approach I used to develop the PdaDriver is hardly unique. Heuristic design, heuristic evaluation, and cognitive walkthrough are all well-known techniques that have long and widely been used to create user interfaces, especially for desktop computing applications.

From a robotics standpoint, however, the use of HCI techniques for human-robot interface design is somewhat unusual. Given that these methods have proven successful at increasing performance and reducing error in other domains, the natural question to ask is: Why is this the case? Why has there been so little use of HCI methods in robotics?

One hypothesis is that mainstream HCI design and evaluation techniques are ill-suited for human-robot interfaces (Graves 1998). Cognitive walkthrough, for example, is generally performed for multi-dialogue interfaces and from the viewpoint of novice users, both of which are fairly uncommon in teleoperation systems. Furthermore, analytical techniques (e.g., GOMS) may be difficult, or impractical, to apply due to the dynamic, control-based nature of human-robot interaction.

In my opinion, however, there are two reasons why robot developers continue to eschew these proven methods, preferring instead to create interfaces in an *ad-hoc* fashion. First, the emphasis in robotics, both in education and research, is strongly placed on developing fully autonomous systems. Consequently, there has been little incentive or need for designers to create effective human-robot interfaces.

The second reason is that HCI researchers, themselves, have done little to apply their techniques to robotics. Part of this derives from a widely held belief that human-robot interfaces are easy to build, i.e., that such interfaces do not require formal design methods. Another common assumption is that all robot "users" are trained experts, and thus will benefit less from well-defined interfaces than novices in other fields (e.g., consumer soft-

ware). As a consequence, there are few HCI researchers who work in robotics and fewer still who advocate the use of HCI methods.

As I have discussed earlier, I believe that there is a need for human-robot systems and that such systems offer significant benefits in terms of capability, flexibility, and performance. The following, therefore, is a brief review of interface design, given so that other robotics researchers will be aware of the issues involved in creating useful human-robot interfaces.

4.6.1 Design Factors

When humans and machines (computers, robots, etc.) communicate, the dialogue is mediated by an interface. Some interfaces, such as computer command languages, offer great power and flexibility but have an associated high learning cost. Other interfaces, such as option menu or direct manipulation (point-and-click) displays, are easier for novices because they make few assumptions about what the user knows. Regardless the type, however, an interface will only be effective if it is well-designed.

There are many factors which shape the way in which interfaces are created: psychological, economic, organizational, institutional, and even political. Interfaces may be designed to meet detailed requirements, specified standards, or to fulfill a perceived need. Thus, interface design is context sensitive: designers must understand the task they are designing for. Moreover, an interface designed for one task may be inappropriate for another task that seems superficially similar.

When an interface is well-designed and easy-to-use, it becomes transparent: we take it for granted and can use it with minimal effort. On the other hand, if an interface is poorly crafted and difficult to understand, it becomes burdensome, limiting performance and making work tiresome. Hence, the value of an interface is largely determined by how effectively it allows the user to perform tasks.

However, interface design is not simply about ease of use. Although making user interfaces easy-to-use is important, a different emphasis is sometimes needed to support the skilled user. Moreover, breakdowns in human performance can be caused by a variety of problems, not all of which are attributable to interface design. Some breakdowns are caused by physical or technological issues (e.g., display glare). Others breakdowns are psychological in nature and can be linked to cognitive limitations including memory capacity, learning, and attention failures (Lansdale and Ormerod 1994).

Interface design has traditionally followed a sequential approach. Task analysis is performed to identify design objectives, which lead to detailed requirements. These specifications are used to select a design method, to guide implementation and as a basis for evaluation. Among the more widely used design methods are heuristic design, guidelines (standards, style-guides, etc.) and iterative design (rapid prototyping).

4.6.2 Design Techniques

Most modern computer interfaces are developed using HCI design techniques, the basis of which is user centered design (Norman and Draper 1986). In user centered design, the basic goal is to support human activity: to enable humans to perform tasks faster, with fewer errors, and with greater quality. Thus, user studies are conducted (often prior to interface development) in order to understand the user's activities, how they perform them and what they need in support. The three primary methods for studying users are interviews, observation, and questionnaires.

Once a design is completed, some form of evaluation is generally needed to assess the usability of the interface. Qualitatively, usability can mean that an interface is “easy-to-use”, “easy-to-learn”, or that it is “well-suited” for a task. Quantitatively, usability may be specified as in terms of learning (training time, skill retention), performance (speed, accuracy, throughput), error (incidence, ability to recover), or even psychological experience (satisfaction, aggravation).

The purpose of evaluation, however, is not merely to compute a single measure of the interface (although this is sometimes necessary to satisfy project requirements). Rather, the true value of evaluation is to provide an informed critique of an interface, identifying its strengths and its weaknesses so that it can be improved (Newman and Lamming 1995).

There are two ways to assess usability: analytically and empirically (Lansdale and Ormerod 1994). Widely used analytical methods include formal modeling (e.g., GOMS), cognitive walkthrough, heuristic evaluation, and standardized checklists. Analytical evaluation, however, generally only provides a partial evaluation of a design. In order to obtain full quantitative measures, or to capture “real-world” failures, empirical evaluation (usability testing) must be performed. To do so, a prototype must be built, instrumented, and experimentally tested with users. This process can be both resource and time consuming, which is why analytical methods are more frequently used.

4.6.3 Interface Control Mechanisms

When humans communicate, they use a variety of techniques to regulate smooth conversation. Prosodic cues (e.g., intonation) indicate when it is time for a speaker to give control of the dialogue to a listener. Additionally, body language and facial expression provide confirmation whether or not information exchange is successful. In a similar fashion, user interfaces need to support control mechanisms which facilitate human-machine dialogue. The three most common control mechanisms are prediction, feedback, and repair.

Prediction

As a user operates an interface, he builds a mental model of how it works. Initially, the model may be based on the user’s prior experience with similar interfaces. Or, it may be constructed from theoretical knowledge gleaned from training or documentation. In either case, as he gains greater familiarity and experience, the user continues to refine the model until it accurately reflects (in his opinion) the main operational concepts of the interface.

Once the user has a working model, he uses it to make predictions of how the interface will respond to given actions (i.e., it allows him to predict the system status). Being able to predict the system status is useful because: it enables the user to recognize which actions are needed to perform a task; it allows him to identify which areas of the interface are relevant to that task; and it reduces the user’s workload (because he only need monitor the task relevant feature or features).

Most importantly, though, prediction helps the user identify errors. Specifically, if the user performs an action and the interface does not respond in the predicted manner, this alerts him that some problem has probably occurred. Similarly, if the pace or latency of the dialogue does not match the user’s expectations, he will conclude that there may be something wrong with the system.

Feedback

All interfaces provide feedback, though what is given may have little or no useful content. Feedback can take many forms. It may appear as properties of interface components, such as the color or shape of a display object. It may also be mediated directly through dialogue (e.g., a text error message). Or, feedback may be given indirectly, perhaps by the way the interface responds or ceases to function.

Feedback enables users to confirm that their mental model is correct. If the model is incorrect, feedback often can be used to correct the model or to help construct a new one. Feedback also allows users to confirm that their predictions are valid. Lastly, feedback provides an indication of what the interface is doing and signals when problems occur.

Designing an interface to provide “good” feedback is difficult. This is because very often the designer and the user of an interface are very different (in experience, training, etc.). Thus, feedback which is unambiguous to the designer may be vague or unhelpful to the user. In fact, many breakdowns in human-machine dialogue occur because of incorrect or inappropriate interpretation of feedback (Lansdale and Ormerod 1994).

Repair

Whenever human-machine dialogue fails or breaks down, some repair process is needed. Repair enables the user (or the interface) to regain control of the dialogue and to correct errors. In many situations, repair may be as simple as backtracking to a known “good” state. The “undo” feature in many computer interfaces is an example of this. For other situations, however, the error or failure is not reversible and repair is more complicated. Recovering a deleted a computer file, for example, can be a difficult and involved process.

In general, there are two types of interface errors: mistakes and slips. Mistakes occur when a user chooses the wrong action. In other words, mistakes are *incorrect intentions*: the user believes he has selected the correct action, but in fact, the action is incorrect. Slips occur when a user makes the correct decision, but then performs an erroneous action. Thus, slips are *incorrect execution*: the choice was correct, but the execution was not.

Of the two types of errors, slips are the most common source of error, occurring as a result of fatigue, lack of concentration, etc. Many interfaces, therefore, are intentionally designed to prevent slips. One often used approach is to limit the set of actions available to the user based on the situation (i.e., limit the opportunity for the user to inadvertently issue an incorrect command).

4.6.4 Common Design Problems

In research and engineering, it is very often the case that the designer of a system is also the designer of the user interface to that system. As a result, the designer typically creates an interface that reflects the underlying system and that affords direct access to the control points of that system. However, unless the designer is the sole user of an interface, which is rarely the case even in a research setting, this is a problem. The reason is that the designer usually has different goals than the user: whereas a researcher or engineer wants to control the system, the user simply wants to complete a task. Thus, an interface based on the task is often more appropriate than one based on the system mechanism (Gentner and Grudin 1990).

A user interface based on the task performance will, by its very nature, distance users from the underlying mechanism. This is often advantageous because it hides system complexity

and helps shield users from making mistakes. However, a task-based interface may also have problems. In particular, such an interface may constrain the use of the system, i.e., limiting the system to specific functions even if it is capable of performing other tasks. Additionally, concealing the system may cause the user to form incorrect models, or to make poor assumptions, about how the system actually works. This can lead to serious consequences, the least of which is user error.

Perhaps the most common design problem, however, is that many interfaces do not adequately support habituation (Raskin 2000). Whenever we, as humans, perform the same task repeatedly, it generally becomes easier to do. That is, with repetition, the task becomes habitual and we are able to perform it automatically, without conscious thought or attention. This is highly beneficial because the more predictable, automatic, and unconscious a task becomes, the less it will degrade or compete with other tasks.

Thus, user interfaces which facilitate habituation are both easier to use and better performing than interfaces which do not. For example, Raskin (2000) observes that one of the reasons Microsoft Windows 2000 is difficult to use is that it features adaptive menus, which are enabled by default. Because menu items are automatically reordered with use, habituation is severely impeded (or prevented) and the user must constantly dedicate attention to locating where a particular entry has been relocated.

4.6.5 HCI Design and Robotics

To date, relatively few researchers have used HCI techniques to design and to evaluate human-robot interfaces. The most commonly used method has been heuristic evaluation, primarily because it facilitates rapid detection of basic usability problems.

Graves (1998) proposes a set of heuristics for designing teleoperator user interfaces. The primary heuristics are: be consistent, make it simple, support decision making, keep the user in control, and assist error recovery. Graves also stresses the importance of contextual design, i.e., interface input/output should be appropriate for the working environment.

Green, Huttenrauch, and Norman (2000) describe the development of a personal service robot using methods from user centered system design. The authors performed a questionnaire study to assess attitudes towards personal service robots and then performed task analysis to ascertain user needs and work tasks. The interface design was performed via a “thinking aloud” walkthrough, followed by heuristic evaluation and guidelines checking.

Keates, Clarkson and Robinson (1999) discuss the use of heuristic design and evaluation for robotic interfaces. They propose a five step process for incremental interface design and testing. Emphasis is placed on verifying user understanding, interaction ease, and system behavior at each step of the design.

Several researchers have conducted user studies of supervisory control systems. These studies combine techniques from HCI and traditional human factors. Adams (1996) describes a study of the “Multiple Agent Supervisory Control (MASC) system in which a single user simultaneously operated four robots. Kay (1997) examines novice users operating an image-based, remote driving system. Simsarian (2000) discusses a study of system design and human-robot interaction issues in a collaborative virtual environment.

4.7 Summary

In this chapter, I described the requirements and design of a collaborative control interface called the PdaDriver. I designed PdaDriver using a combination of three HCI techniques: heuristic design, heuristic evaluation, and cognitive walkthrough. PdaDriver runs on WindowsCE-based PDA's and enables human-to-robot dialogue. It provides a variety of command modes and supports simultaneous (independent) control of multiple mobile robots.

Safeguarded Robot Controller

*A robot may not injure a human being or, through inaction,
allow a human being to come to harm.
A robot must obey the orders given it by human beings, except
where such orders would conflict with the First Law.
A robot must protect its own existence, except where such protection
would conflict with the First or Second Law.
— Asimov’s “Three Laws of Robotics”*

To function in collaborative control, the robot needs to be aware, self-reliant, and adaptive. Thus, the robot must have a controller that enables it to perceive the remote environment, to maintain its own safety, and to automatically adjust its level of autonomy whenever the human cannot, or is unable, to answer its questions. In the previous two chapters, I presented the system-level design principles for collaborative control and the specific requirements for building a user interface. In this chapter, we will see how both of these, in turn, influence robot controller design.

I begin by discussing the requirements for a robot controller, based on the needs of collaborative control, vehicle teleoperation, and the user interface. I then describe the two mobile robots used in this thesis, focusing on physical and sensor characteristics. Next, I present a controller design, with emphasis on its layered architecture and the key features of each module. An important attribute of the controller is that it provides continuous safeguarding to ensure that the robot is kept safe regardless of control mode, operator input, and environmental hazards. The chapter concludes with a brief review of related work in controller design.

5.1 Requirements

In Chapter 3, I listed the principles which guided the overall system design. As was the case with the user interface, however, these principles are not sufficient for developing a robot controller. Thus, at a minimum, we need to identify additional requirements before we can proceed to controller design. Let us start by considering what is needed to support the user interface.

All teleoperation user interfaces include tools to help the operator perceive the remote environment, to make decisions, and to generate commands (Fong and Thorpe 2001). An effective robot controller, therefore, provides resources to support these tools. In particu-

lar, the controller must supply command facilities and sensory feedback that make the interface effective and that make remote tasks easy to perform.

As I have discussed, the context of this thesis is vehicle teleoperation, and specifically, remote driving in unknown, unstructured environments. Thus, the controller needs to facilitate navigation and motion control. Furthermore, to support collaborative control, the controller should be able to cope with varying human-robot interaction and user expertise.

To support navigation, the controller must provide visual feedback (still images and/or video), spatial feedback (sensor-based maps), and situational feedback (robot health, position, command status). Additionally, because navigation is strongly dependent on perception, the controller must also provide facilities for sensor management and processing. For example, sensor-based map building requires range data processing.

To support varying human-robot interaction, the controller must provide a number of motion commands (Table 7). Position and rate control of robot translation and rotation is needed for direct control. Pose (single-point) and path (multiple-points) commands are needed for waypoint control.

Table 7. Motion commands

command	control type	control variable
translate	direct (position, rate)	distance, translation speed
rotate	direct (position, rate)	heading (relative/absolute), rotation rate
vector	direct (position, rate)	heading (absolute) + translate rate
pose	waypoint	2D (x, y), 3D (x, y, heading), path

To support untrained users, as well as operation in dangerous environments, the controller must be able to perform real-time, reactive safeguarding. Specifically, the controller should be capable of maintaining robot safety at all times without user intervention. This entails detecting obstacles, avoiding collisions, preventing rollover, monitoring health (power level, temperature, etc.) and safeing the vehicle when necessary.

Finally, to enable operation with a broad variety of user interface hardware (PDA, desktop computer, etc.) and in various operational settings (in an office, in the field, etc.), the controller must be able to function with a range of communication links. In particular, the controller, should still perform competently when bandwidth is limited, when there is transmission delay, and when the link is intermittent, spurious or unreliable.

5.2 Robot Hardware

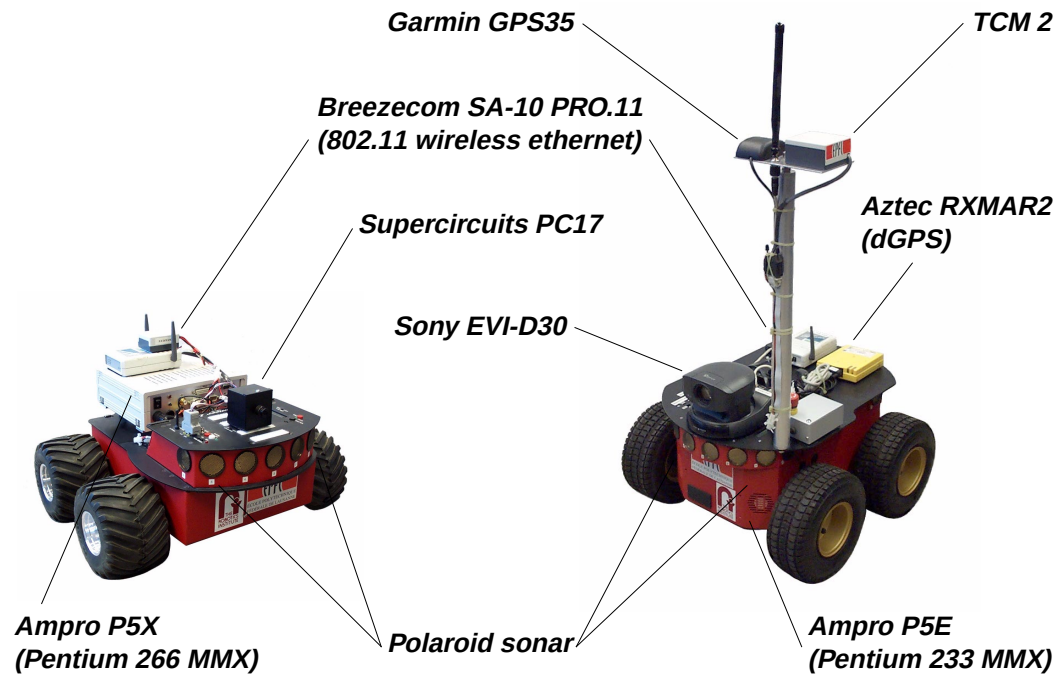


Figure 24. Left, Pioneer-AT; right, Pioneer2-AT.

I designed the controller to operate Pioneer¹ mobile robots. For this thesis, I used a Pioneer-AT and a Pioneer2-AT. Both robots (shown in Figure 24) are 4-wheel skid-steered and capable of traversing moderately difficult terrain. Both have a microcontroller (for motor servo and hardware control), Pentium-based computing, and 802.11 wireless ethernet. Table 8 provides physical and computing specifications for the two robots.

Table 8. Pioneer-AT and Pioneer2-AT specifications.

Feature	Pioneer-AT	Pioneer2-AT
Size (<i>l</i> x <i>w</i> x <i>h</i>)	45 x 50 x 24 cm	50 x 49 x 26 cm
Weight	11.3 kg	12 kg
Payload capacity	4.5 kg	30 kg
Power	144 W-hr	252 W-hr
Max speed	1.5 m/s	0.7 m/s
Max slope	80% grade	40% grade
Microcontroller	Motorola MC68HC11	Siemens C166
Computing (PC/104)	Ampro P5X (P266-MMX) Imagenation PXC200 framegrabber	Ampro P5E (P233-MMX) Imagenation PXC200 framegrabber

1. Pioneer is a trademark of ActivMedia, Inc.

In their standard factory configuration, the Pioneer-AT and Pioneer2-AT both are equipped with drive wheel encoders and an array of ultrasonic sonar. To this base configuration, I added additional sensing to make the robots suitable for remote driving. Sensor selection was guided by the intended driving environment for each robot: indoors for the Pioneer-AT and outdoors for the Pioneer2-AT.

The Pioneer-AT sensor suite is given in Table 9. Wheel encoders are used for odometry-based localization. Time-of-flight ultrasonic sonar is used for environment mapping. A fixed, forward-pointing color camera provides video feedback.

Table 9. Pioneer-AT sensor suite

Sensor	Description	Key characteristics
Hewlett Packard HEDS-9100-C00	incremental optical encoder (mounted on a 65.5:1 gear motor)	100 counts/rev
Polaroid 600 series sonar	time-of-flight ultrasonic ranging (electrostatic transducer)	15 cm to 10 m range 1% accuracy
Supercircuits PC17	1/3" color CCD camera fixed mount	60° HFOV

The Pioneer2-AT sensor suite is given in Table 10. Differential GPS, a triaxial magnetometer, a biaxial inclinometer, and wheel encoders are used for localization. As with the Pioneer-AT, time-of-flight ultrasonic sonar is used for environment mapping. A steerable (pan/tilt), forward-pointing color camera provides video feedback.

Table 10. Pioneer2-AT sensor suite

Sensor	Description	Key characteristics
Aztec RXMAR2	dGPS (RTCM) RDS receiver	up to 1m SEP
Garmin GPS35-HVS	12-channel C/A GPS receiver	20 cm resolution (Cartesian user grid)
Hewlett Packard HEDS-9100-C00	incremental optical encoder (mounted on a 65.5:1 gear motor)	100 counts/rev
Polaroid 600 series sonar	time-of-flight ultrasonic ranging (electrostatic transducer)	15 cm to 10 m range 1% accuracy
Precision Navigation TCM2-2-20	triaxial magnetometer, biaxial inclinometer, and temperature	$\pm 2^\circ$ heading accuracy 0.1° tilt accuracy, $\pm 20^\circ$ range
Sony EVI-D30	1/3" color CCD camera with 12x zoom and pan/tilt	4.4° to 48.8° HFOV $\pm 100^\circ$ pan, $\pm 25^\circ$ tilt

I should note that the TCM2 (which is widely used in mobile robotics) outputs roll, pitch, compass heading, magnetic field, and temperature. Although the static tilt data is reliable, dynamic performance and heading accuracy are marginal at best (Deschler 1998).

5.3 Controller Design

Three-layer architecture

The controller is designed with a three-layer architecture and is shown schematically in Figure 25. In the bottom layer, the controller provides reactive feedback-control: it uses behaviors for safeguarding and robot motion. In contrast to other three-layer architectures, these behaviors maintain internal state (activation time, number of attempts, etc.) to facilitate human-robot interaction and to assist in failure detection.

In the middle layer, the controller performs sequencing: it selects which primitive actions should be in use, then handles behavior activation and deactivation. For example, low-level motion control and safeguard behaviors are sequenced during execution of waypoint commands. Similarly, task-specific modules (e.g., autonomous survey) generate control sequences to exercise control over the robot.

In the top layer, the controller integrates human input via the user interface. In autonomous robot systems, the top layer contains high-level planning and decision making modules. In teleoperation, the human typically performs these tasks. With collaborative control, however, both the robot and the human share responsibility for producing plans and for responding to queries from the other controller layers.

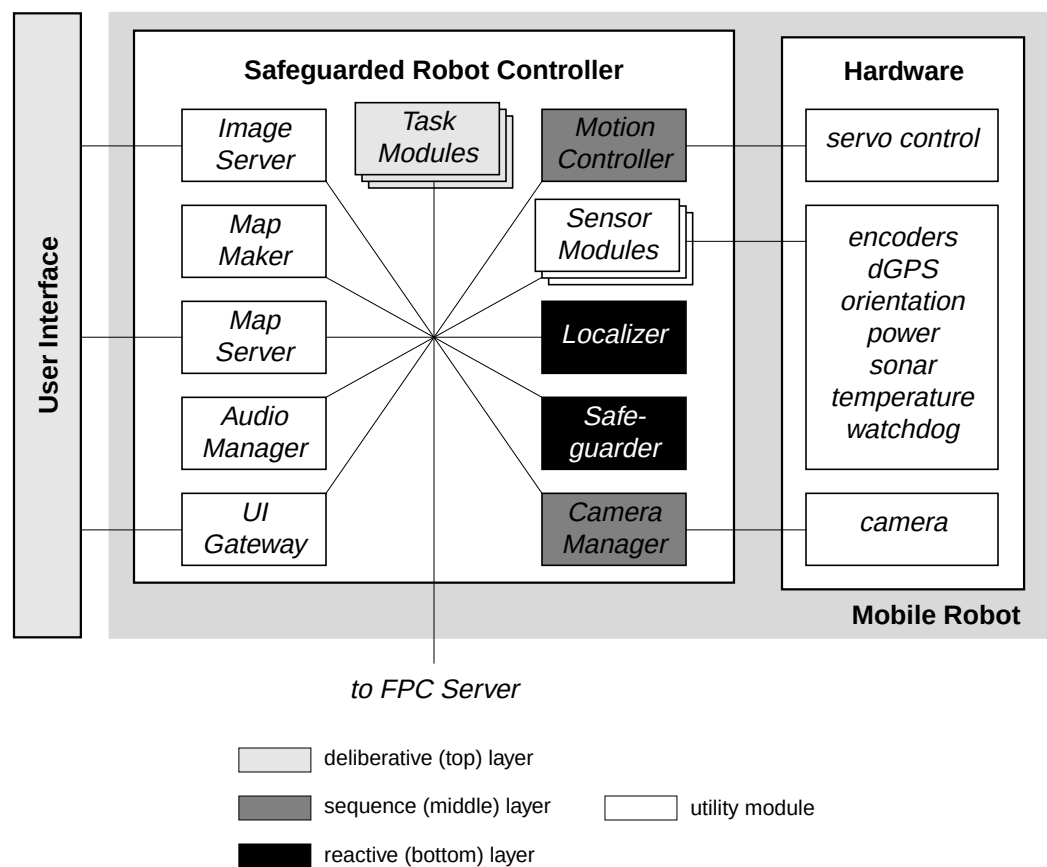


Figure 25. Safeguarded robot controller architecture

Modules

The controller is implemented as a distributed set of modules, connected by interprocess communications (see Section 3.1.3). These modules were designed to meet the requirements described in Section 5.1. In particular, the modules provide visual and spatial feedback, sensor management, motion control, and reactive safeguarding (Fong, Thorpe, and Baur 2001b). Figure 25 shows how the controller modules are connected to the user interface and to the mobile robot.

Some of the modules run standalone and operate asynchronously. Other modules, particularly those which process sensor data or operate robot hardware, have precise timing or data requirements and are implemented with Saphira (Konolige and Myers 1997). Saphira was chosen for several reasons: it is a mature system and works well with Pioneer robots; it provides efficient command fusion through fuzzy behaviors; it is extensible, modular and portable; and the micro-tasking operating system is synchronous and interrupt-driven, thus making it easy to implement modules with precise timing.

Table 11 lists the modules in currently used by controller. Many of these modules, such as the Localizer or the Safeguarder, perform tasks commonly required by autonomous robots. Several modules, however, are designed specifically to support vehicle teleoperation. The MapServer, for example, converts maps (constructed as 2D histogram grids using range sensor data) into gray-scale images for user interface display. Section 5.4 describes each controller module in detail.

Variable autonomy

Each module that performs an autonomous function is designed to operate with a variable level of autonomy. At the lowest level (i.e., minimal autonomy), each module is dependent on the human. That is, before a module executes a control action, it is required to interact with the human. For example, one of the Safeguarder's tasks is to prevent robot motion whenever the power level drops below a safety cutoff. When this happens, the Safeguarder must ask the human if it is safe to continue operating, and if so, at what level to set the safety cutoff.

At higher levels of autonomy, each module exercises independent control authority and operates with little (or no) human interaction. That is, whenever the human is unable, or is unqualified, to respond to robot questions, each module is designed to operate without human assistance. The Safeguarder, for example, can automatically adjust the power level safety cutoff, as long as it remains in a "non-critical" range, i.e., within known, acceptable margins.

Table 11. Controller modules

name	function	execution style	implementation (language)
AudioManager	sound playback speech synthesis	asynchronous	standalone (C)
CameraManager	camera control	asynchronous	standalone (C)
Hardware Controller	servo control vehicle electronics	real-time	Pioneer microcontrol
ImageServer	image capture	asynchronous	standalone (C)
Localizer	position estimation	synchronous (10 Hz)	Saphira (C)
MapMaker / MapServer	map building map generation	asynchronous	standalone (C)
MotionController	high-level motion	synchronous (10 Hz)	Saphira (C, fuzzy behavior)
Safeguarder	health monitoring motion safeguards	synchronous (10 Hz)	Saphira (C, fuzzy behavior)
Sensor Modules	sensor processing	synchronous (10 Hz)	Saphira (C)
Task Modules	task-specific functions	asynchronous	standalone (C)
UIGateway	proxy server for user interfaces	synchronous (varies)	standalone (C)

Whether a module operates with a low or high level autonomy is determined by numerous factors. These include situational demands (whether a module is sufficiently competent to operate autonomously), the user model (whether the current operator is able to assist in performing a specific task or making a specific judgement), and other factors such as urgency (e.g., if action must be taken in real-time or before the user can be consulted).

5.4 Modules

5.4.1 AudioServer

For some applications, particularly when the robot must operate around or with humans, audio plays an important role in human-robot interaction. Specifically, audio is a highly effective mechanism for conveying the robot's intent and for communicating information to humans. Thus, the AudioServer is designed to perform two functions: sound playback and speech synthesis.

Sound playback is used to produce informative signals. For example, the controller uses a train whistle sound to warn that the robot is approaching and to request that people move out of the way. It is interesting to note that a train whistle produces a significantly better response (i.e., people pay more heed and react more positively) than a horn or klaxon.

Speech synthesis is used for information which cannot be conveyed by simple sound playback, such as status messages ("turning right 90 degrees", "moving to point 3", etc.), health warnings ("low battery"), and alerts ("motor stall"). The AudioServer produces speech with *FreePhone* (an OpenSource text-to-phonetic transcription package) and the *MBROLA* speech synthesizer (Dutoit et al. 1996).

5.4.2 CameraManager

The CameraManager operates the robot's camera systems, arbitrating and sequencing camera commands from other modules. Its primary function is to control steerable CCD cameras, i.e., cameras mounted on or incorporating a positioning mechanism. The CameraManager is also used to configure imaging parameters (gain, exposure compensation, magnification, etc.). Whenever it changes a camera's configuration, the CameraManager outputs a message describing the camera's state (position, magnification, field-of-view, etc.), so that modules making use of the camera can act appropriately.

5.4.3 ImageServer

In many vehicle teleoperation systems, video is the primary source of visual feedback. High-quality video, however, uses significant communication bandwidth. Moreover, for applications with poor communications (low bandwidth and/or high transmission delay), video may not be practical.

As an alternative to video, the controller provides an event-driven ImageServer. This server is designed to minimize bandwidth consumption by outputting images only when certain events occur. Specifically, the ImageServer captures (or loads) a frame, compresses it into a JPEG image, and sends it whenever: (a) the operator issues a request; (b) the robot stops; (c) an obstacle (static or moving) is detected; or (d) an interframe timer expires.

Event-driven imagery is a flexible mechanism for visual feedback. For example, if an application allows use of a high-bandwidth, low-latency communication link, we can set the interframe timer to a low value (e.g., 0.2 sec). This results in an image stream which approximates video. Alternatively, if the link is low-bandwidth or has high delay, we set the timer to a high value. In this case, images are transmitted only when important teleoperation events occur. Since we are most likely to be using intermittent control (e.g., way-point-based driving) in this situation, event-driven imagery works well.

5.4.4 Localizer

The Localizer estimates vehicle position and orientation. On the Pioneer-AT, pose is dead reckoned with odometry. On the Pioneer2-AT, position is estimated using odometry and dGPS, and orientation using the TCM2 and odometry. When the Localizer is running, it continually outputs its pose estimate and localization uncertainty. The Localizer provides estimates in two coordinate frames.

The navigation (world) frame is inertially-fixed and locally-level: $\hat{x}$ is east, $\hat{y}$ is true north, and $\hat{z}$ is up (i.e., gravity aligned). When valid dGPS fixes are available, the world frame coincides with the regional Cartesian user grid (e.g., UTM). The body (local) frame is vehicle-fixed with the origin set at the center of rotation. On both robots, this is the mid-point of the longitudinal center-line at axle height. The body-frame axes are: $\hat{x}$ forward, $\hat{y}$ left, and $\hat{z}$ up.

5.4.5 MapMaker

Although image-based driving is an efficient command mechanism, it may fail to provide sufficient contextual cues for good situational awareness. Maps can remedy this by providing reference to environmental features, explored regions and traversed path. In addition, maps can be efficiently used for collision avoidance. The MapMaker builds maps using range sensors (e.g., ultrasonic sonar) and a method based on “Histogrammic In-Motion Mapping (HIMM)” (Borenstein and Koren 1991), but with several key modifications.

As in HIMM, the MapMaker uses a 2D Cartesian grid to map the environment. Each grid cell contains a certainty value cv that indicates the confidence of an obstacle (or free space) in the cell. Unlike HIMM, a signed 8-bit integer is used for each confidence value ($-127=clear$, $0=unknown$, $127=obstacle$). The more positive the cv , the higher the confidence that the cell contains an obstacle. The more negative the cv , the higher the confidence that the cell is clear.

The primary method of updating certainty values is to project range sensor readings onto the grid. For each range reading, one cell is incremented (i.e., the cell corresponding to the measured range) and all cells on the sensor axis (i.e., from the sensor mount point to the range reading) are decremented (see Figure 26). The values used for cell increment, $+I$, and cell decrement, $-D$, are constant and are experimentally determined for each sensor.

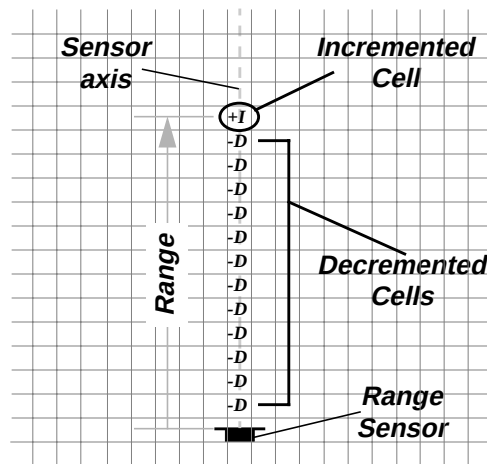


Figure 26. Each range sensor reading is used to update the map grid.

By continuously and rapidly sample range sensors while the vehicle is moving, a histogrammic probability¹ distribution is produced in which cells near an obstacle contain high certainty values. Rapid in-motion sampling helps reduce the effect of sensor noise and inaccurate range readings (i.e., spurious readings are quickly “washed out”). Furthermore, because cells containing an obstacle are repeatedly updated, this approach accurately localizes obstacles even when range sensors with low angular resolution (e.g., ultrasonic sonar) are used (see Figure 27).

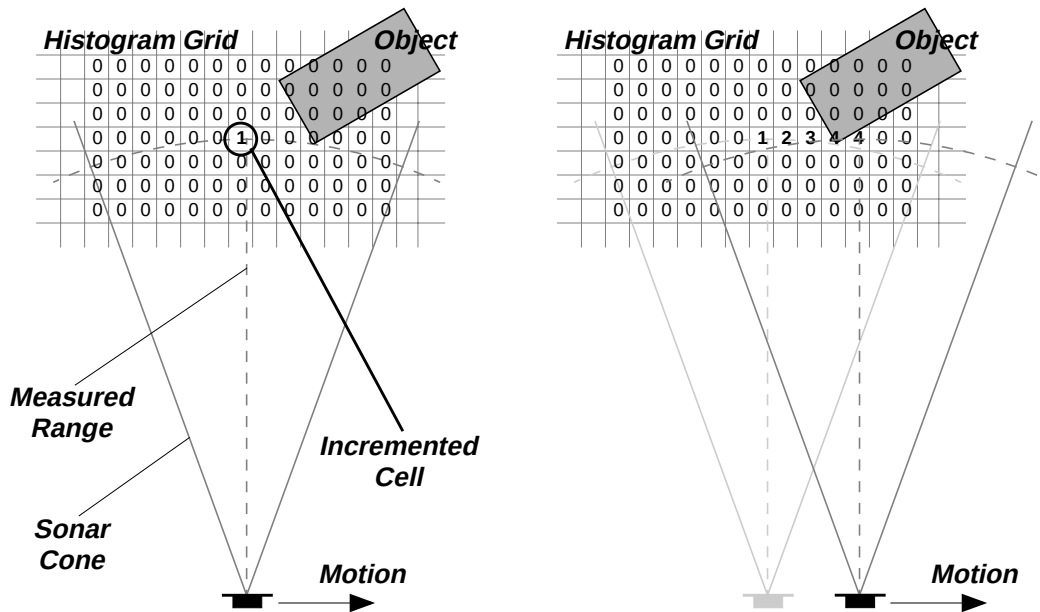


Figure 27. The HIMM method produces accurate maps even if the range sensor (e.g., ultrasonic sonar) has low angular resolution.

Instead of HIMM’s large, world-fixed grid, the MapMaker uses a small grid (200x200 with 10 cm x 10 cm cells) which is periodically relocated in response to robot motion. Specifically, whenever the robot approaches a border, the MapMaker performs a grid shift operation (discarding cells which are pushed over the grid boundary) to keep the robot on the map (see Figure 28). In this way, we can construct useful “global” maps (i.e., up to 20x20 m) while bounding computation and memory usage.

The most significant difference between HIMM and the MapMaker is how the histogram grid is updated. In HIMM, and its successors (*VFH* and *VFH+*), sonar ranges are used only while the robot is moving. This reduces the impact of spurious readings due to multiple reflections and sensor noise. However, this also makes HIMM perform poorly when dynamic obstacles are present: if the robot is stopped, the map does not reflect moving objects. To address this shortcoming, the MapMaker updates the grid whenever a new reading is available. However, if the robot is stopped, the MapMaker only uses the reading if it indicates clear (i.e., no return or large range).

In addition to range processing, the MapMaker updates the grid to account for localization error. It does this so that the map will reflect robot pose certainty, especially with respect to mapped features of the environment. Thus, whenever localization error increases, the

1. in the sense of *likelihood*

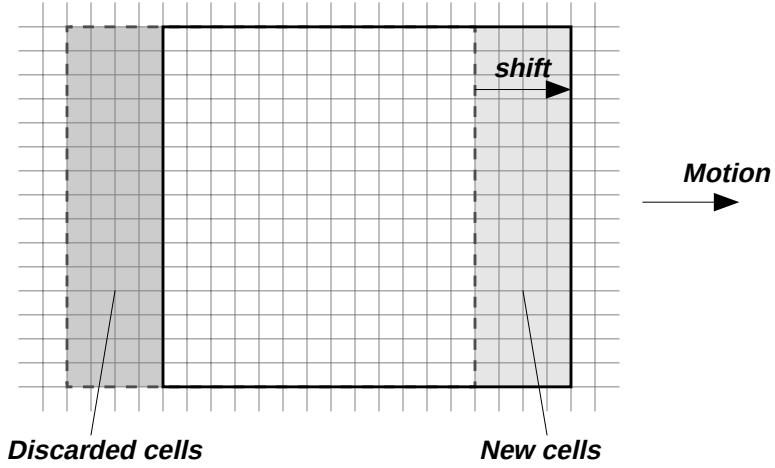


Figure 28. The MapMaker uses a fixed-size grid which is shifted as the vehicle moves.

MapMaker globally moves all certainty values towards zero. That is, given a confidence value change, Δcv , the MapMaker decrements all positive confidence values and increments all negative confidence values¹ (see Figure 29). As a consequence, local or close (i.e., recently mapped) areas appear “crisp” and distant (long-ago mapped) regions become fuzzy.

0	0	0	0	-30	-20	-20
0	0	0	0	0	-30	-20
0	0	40	40	0	-40	-40
0	0	60	0	0	0	0
0	20	20	0	0	0	0

$\Delta cv = 25$
 $\longrightarrow$

0	0	0	0	-5	0	0
0	0	0	0	0	-5	-0
0	0	15	15	0	-15	-15
0	0	35	0	0	0	0
0	0	0	0	0	0	0

Figure 29. Localization error update moves all certainty values towards zero.

As an example, consider the localization of a skid-steered vehicle using only odometry. With skid-steering, vehicle rotation (e.g., in-place turning) produces larger dead-reckoning errors than translation. We can use this fact to compute the Δcv due to vehicle motion:

$$\Delta cv = (K_t \Delta t) + (K_r \Delta r) \quad (1)$$

where Δt and Δr are position and orientation changes since the last grid update. The two constants, K_t and K_r , provide an estimate of error growth.

1. Both increment and decrement operations are thresholded at zero.

The grid update algorithm is then:

```
for each cell in grid do  
  if  $cv > 0$  then  
     $cv = cv - \Delta cv$   
    if  $cv < 0$  then  
       $cv = 0$   
    endif  
  else if  $cv < 0$  then  
     $cv = cv + \Delta cv$   
    if  $cv > 0$  then  
       $cv = 0$   
    endif  
  endif  
end
```

Because this update is computationally expensive (i.e., it requires modifying every cell), it is only performed when there is substantial change in localization error.

5.4.6 MapServer

The MapServer provides maps as images. Whenever it receives a request, the MapServer queries the MapMaker for the relevant grid region and converts certainty values to gray-levels¹. Clear areas appear as white, obstacles as black, and unknown as light-gray. The MapServer can generate maps in either the world or local frame, with arbitrary resolution (sub/super sampling) and of any extent (regions outside the grid are marked “unknown”).

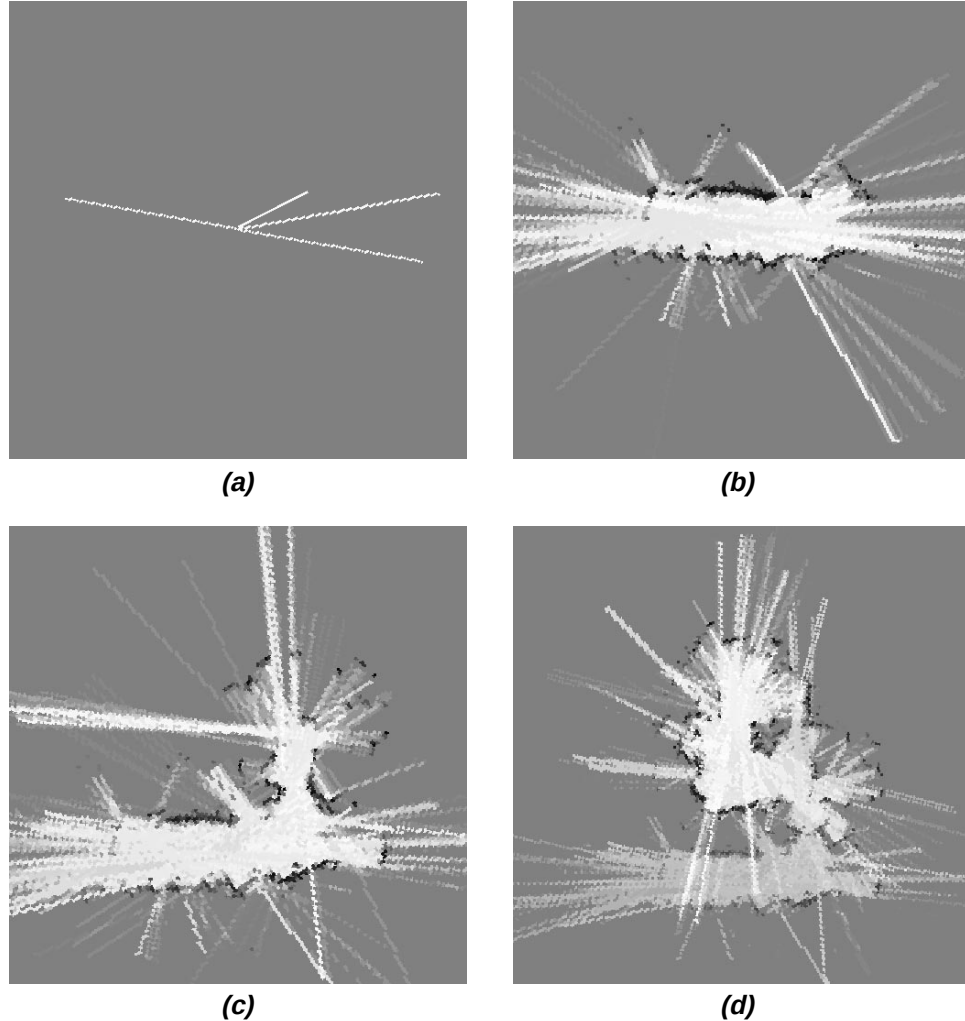


Figure 30. MapServer maps: (a) unexplored environment; (b) corridor; (c) corridor and room (initial entry); (d) corridor and room (after exploration).

Figure 30 shows some typical maps generated by the MapServer. Initially, the environment is unknown. Map (a) only shows four obstacle free rays (e.g., based on clear sonar returns). As the robot moves, it becomes clear that it is in a corridor (b). In (c), the robot has just left the corridor and entered a room. Map (d) shows the result of exploring the room using only odometry for localization: after multiple turns, the corridor has begun to disappear from the map (i.e., its location has become less certain due to localization error).

1. $\text{gray-level} = cv + 127$

5.4.7 MotionController

The MotionController executes and monitors motion commands. It generates position and rate setpoints (translation and rotation) for the robot's low-level servo controller. The MotionController sequences and executes motion commands using Saphira fuzzy behaviors. This allows all vehicle motion to be safeguarded (i.e., via command fusion with Safeguarder behaviors). Because Pioneer robots can turn in place, the MotionController uses a "turn then move" motion strategy.

The MotionController continuously monitors the progress and status of each executing motion behavior. Whenever it detects lack of progress (e.g., due to safeguarding) or failure, the MotionController contacts the operator. For example, if the MotionController determines that an object is blocking forward progress, it may send the operator a camera image (taken from a forward pointing camera) and ask "Can I drive through?". If the human answers "yes", the MotionController will temporarily disable collision avoidance and move forward.

5.4.8 Safeguarder

The Safeguarder is responsible for maintaining the robot's safety. Each safeguard is implemented as a Saphira fuzzy behavior and may be individually activated or disabled by other controller modules. At present, the Safeguarder provides collision avoidance, rollover avoidance, health monitoring, and clutter monitoring. All safeguards are designed to execute quickly¹ so that safety can be maintained even in dynamic environments.

Collision avoidance

To avoid collisions, the Safeguarder scans the MapMaker's occupancy grid for obstacles. The approach is inspired by VFH+ (Ulrich and Borenstein 1998), but computes distance to obstacles rather than obstacle-free steering direction. Using pre-computed lookup tables, collision detection can be performed quickly and with minimal computational overhead. The procedure executes as follows.

First, we define the active region, C^* , as a rectangular window of the MapMaker's histogram grid, centered at the current robot position. The Safeguarder uses a 25x25 cell (2.5x2.5 m) grid and can detect obstacles within a radius of 1.25 m from the robot center.

At each cycle, we construct a polar histogram containing distance to obstacles. This is done by scanning C^* for cells containing non-negative confidence values ($cv > 0$, indicates a possible obstacle) and then using lookup tables to find the obstacle distance and histogram bin. The Safeguarder currently uses a histogram with 5 deg angular resolution. As with the first stage of VFH+, obstacle cells are enlarged by the robot radius to account for the width of the robot.

The Safeguarder uses the histogram to scan for obstacles in the direction of motion. Whenever the robot approaches an obstacle, the Safeguarder limits translation speed, reducing the limit as a function of distance. If an obstacle reaches a pre-defined standoff distance, the Safeguarder forces the robot to stop.

1. at 10 Hz, the default Saphira clock rate.

Rollover avoidance

The Safeguarder prevents rollovers by monitoring vehicle attitude. Whenever roll or pitch exceeds a threshold for more than a short period, the Safeguarder forces the robot to stop and contacts the human for assistance.

At present, the Safeguarder uses pre-defined thresholds for both roll and pitch. It does not monitor terrain characteristics (e.g., slope), nor does it explicitly consider vehicle dynamics. An experimentally tuned exponential filter is used to reduce high-frequency effects (e.g., high-speed driving over bumpy terrain). Overall, the current rollover avoidance behavior is very conservative and limits motion performance.

Health monitoring

The Safeguarder continually monitors system health and takes action if it detects problems. In particular, the Safeguarder prevents vehicle motion if it detects low power, high temperature, excessive motor stall (indicative of drive obstruction), or hardware controller failure.

Whenever one of these conditions occurs, the Safeguarder may interact with the human (e.g., to ask if a safety trigger can be relaxed or modified) or may act autonomously. In the latter case, the Safeguarder will only modify a trigger level if it can do so without violating absolute, pre-defined safety margins.

Clutter monitoring

The Safeguarder monitors the density of obstacles in the environment by computing a clutter metric, ρ . It does this by counting the number of cells, n , which have non-negative confidence values in C^* and then applying:

$$\rho = \frac{Kn}{N} \quad (2)$$

Where K is an experimentally determined constant and N is the total number of cells in the active region.

The Safeguarder uses ρ as a guideline for setting maximum translation and rotation rates. If ρ is small, the environment is uncluttered (few obstacles), and the Safeguarder will recommend fast movement. If ρ is large, the environment contains many nearby obstacles, and the Safeguarder will recommend that the robot move slowly. In either case, the Safeguarder will ask the human for his opinion before changing the rates. If the human suggests different rates, the Safeguarder will use those. If the human does not respond, the Safeguarder will autonomously adjust the rates.

5.4.9 Sensor Modules

A Sensor Module interacts with a single sensor or a group of related sensors. Each module works like an operating system driver: it communicates with the device using sensor-specific protocols, processes the sensor data, then publishes the results for other controller modules to use. There are currently five Sensor Modules in the controller: GPS, Health, Odometer, Sonar, and TCM2.

The GPS module acquires GPS position fixes and transforms the data from geodesic coordinates to a regional Cartesian user grid, such as “Swiss grid” or “Universal Transverse

Mercator” (UTM). If data is unavailable, or of poor quality (e.g., high DOP), the GPS module outputs a warning.

The Health module monitors vehicle health sensors. At this time, these are power (battery voltage), temperature (environment), and watchdog (hardware controller).

The Odometer module processes wheel encoder data. It computes differential position and velocity based on encoder changes.

The Sonar module controls a ring of ultrasonic sonar. It is used to enable/disable transducers and to configure polling order (to minimize crosstalk). The Sonar module processes range data by applying a cut-off filter (ranges greater than a cut-off are discarded) and then transforming the range to a position (world frame).

The TCM2 module acquires orientation (roll, pitch, compass heading) and external temperature data from the TCM2. To reduce noise, the module smooths the data using an exponential filter.

5.4.10 Task Modules

Each Task Module performs a specific perception or decision making function to meet the needs of a particular teleoperation application. A Task Module may add an autonomous capability to the robot or may offer assistance to the human. In this thesis, two Task Modules have been developed: MotionDetector and RockFinder.

The MotionDetector (Section 6.3.2) is used for autonomous visual surveillance. It detects motion in the environment by acquiring camera image sequences and computing inter-frame differences. The MotionDetector notifies the human whenever significant motion occurs.

The RockFinder (Section 6.2.2) is a mock science module. It is designed to demonstrate the type of autonomous survey work a planetary rover could perform in collaboration with an EVA scientist. The RockFinder segments color camera images into contiguous regions based on normalized color. Each region which exceeds certain extents and which matches a pre-defined color signature is labeled as a potential “rock” for the human to examine in greater detail.

5.4.11 UIGateway

The UIGateway is a proxy server for user interfaces. It provides access to controller services (e.g., motion control) while hiding the controller’s complexity. The UIGateway uses a simple message protocol which works well even over low-bandwidth connections. The protocol is text-based (which speeds integration of diverse interfaces) and asynchronous (to reduce latency).

Whenever an interface is connected, the UIGateway continually monitors the connection. If it detects a communication problem (e.g., network outage) or that the interface is no longer responding, the UIGateway immediately stops the robot and closes the connection. Thus, the UIGateway ensures that operator commands are only executed while the interface is active and functioning.

5.5 Related Work

5.5.1 Safeguarded Teleoperation

The safeguarded teleoperation concept was developed to enable remote driving of a lunar rover (Krotkov et al. 1996a,b). Command fusion enables operators to share control with a safeguarding system (the “safeguarder”) on-board the robot. In benign situations, the operator has full control of vehicle motion. In hazardous situations, however, the safeguarder modifies or overrides operator commands to maintain safety. The safeguarder, therefore, exhibits many characteristics of autonomous systems such as perception, command generation, etc.

Unlike the system described in Krotkov et al. (1996), which was designed exclusively for untrained operators and continuous control, my controller supports a range of users (novices to experts) and intermittent as well as continuous control. Moreover, in addition to safeguarding vehicle motion (preventing collision and rollover), the controller monitors system health (vehicle power, motor stall, etc.) and “safes” the vehicle when necessary.

5.5.2 Control Systems for Teleoperation

Numerous researchers have addressed the problem of designing control systems for teleoperation. Because teleoperation applications are so varied, controllers encompass a wide range of designs and techniques. The majority, however, can be described within the framework of one or more *robot control architectures* (Hasemann 1995).

Maslowski et al. (1998) describe a parallel, three-layered control architecture for teleoperation of mobile robots. This controller provides reflexes for obstacle avoidance, plan learning, and compressed communications. Graves and Czarnecki (1999) discuss a “generic” telerobotic controller. The design uses a network of low-level control behaviors switched on and off by a high-level symbolic layer. Lin et al. (1995) present a mobile robot control system with multisensor feedback. This system supports four teleoperation modes (direct, traded, shared, supervisory) and allows operators to assist in environment modelling.

My controller shares features with each of these systems (three-layer architecture, reflexes for obstacle avoidance, etc). There are, however, two significant ways in which my controller differs. First, my controller supports a broad range of user interface hardware and functions even with poor communication links. All the other controllers are designed for a restricted set of system hardware. Second, and more importantly, my controller incorporates modules with variable autonomy, the levels of which are dynamically adjusted based on situational need and human-robot interaction. The other systems, even (Lin et al. 1995), use pre-defined robot autonomy, which cannot be disabled or must be manually activated.

5.6 Summary

The robot controller presented in this chapter supports remote driving in unknown, unstructured environments. The controller differs from other teleoperation control systems because it satisfies the needs of a broad range of operator interfaces and control modes. In addition, the controller provides continuous safeguarding to maintain vehicle safety regardless of control mode, operator input, and environmental hazards.

6

Applications

To gain insight into collaborative control, I examined its use in three vehicle teleoperation applications (Fong, Thorpe, and Baur 2001c). In all three, the key tasks were remote driving and perception, both autonomous and human-aided. No manipulation or other remote action (e.g., payload deployment) was required.

The first application, “A to B”, concerns the fundamental task in vehicle teleoperation: moving the remote vehicle from one location to another, i.e., from A to B. Though this may seem straightforward, it often is not, particularly when high delay (communication, system, etc.), low-bandwidth, or dynamic obstacles (moving or malignant) are present.

The second application, collaborative exploration, examines a human and a robot working together in the field. Human-robot teams have recently been the focus of much research effort, especially within the space robotics community. The motivation for this is that such teams have the potential to significantly enhance the capability and productivity of planetary surface missions.

The third application, multi-robot remote driving, considers how a single operator can use multiple mobile robots to perform tasks such as reconnaissance and surveillance. The American military is particularly interested in this type of human-robot interaction for supporting future combat systems.

6.1 “A to B”

Perhaps the most basic task in vehicle teleoperation is “A to B”. That is, if the robot is at point A, and if we know where point B is located, the problem is to control the robot’s actions so that it moves from A to B. As basic as this task may seem, executing it successfully is critical for many applications. In reconnaissance, for example, task performance is directly related to being able to move accurately from point to point. Thus, it is important to make “A to B” as efficient as possible.

In the past, most vehicle teleoperation systems used only direct (manual) control. As I have discussed, however, this mode of operation is subject to many performance limiting factors. Moreover, direct control is not practical (or possible) for applications involving low-bandwidth or high-delay communications such as planetary exploration. Thus, many vehicle teleoperation systems now use some form of waypoint driving (see Section A.4.3).

One problem with waypoint driving is that whenever the robot is moving autonomously, it may incorrectly identify an obstacle. Based on this false information, the robot would then be forced to choose a very poor solution. Yet, if the robot is able to ask the human “Can I drive through?” and the human decides that it is possible (based on his experience or his interpretation of the sensor data), then the robot take advantage of the most efficient solution (i.e., drive through).

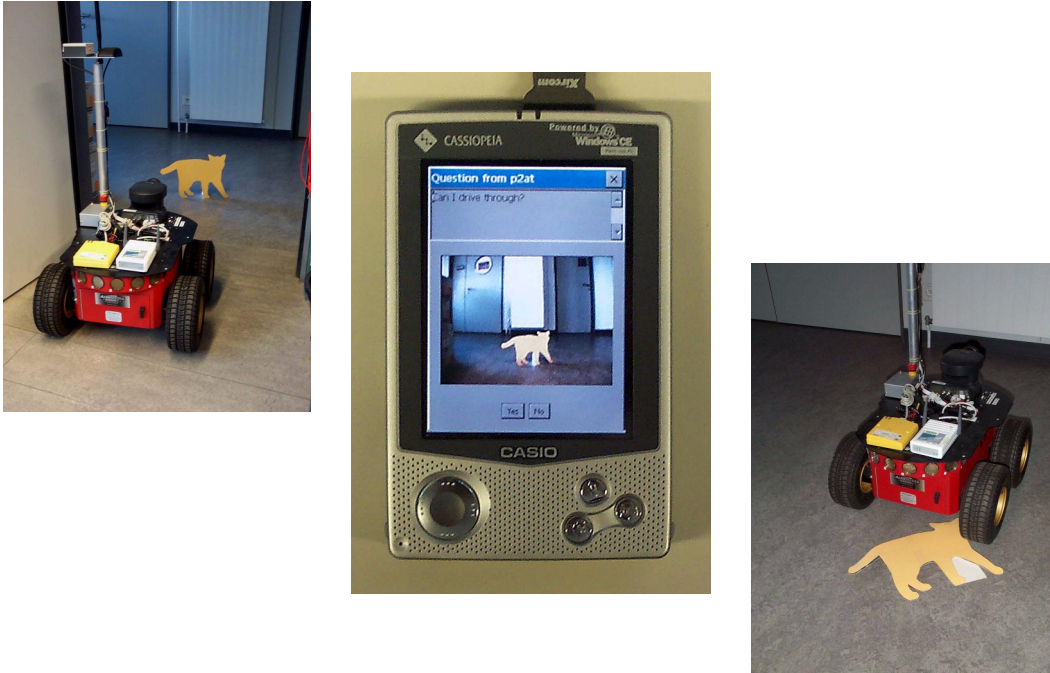


Figure 31. Query to the human: “Can I drive through?”

Figure 31 shows an example of this interaction occurring in an experiment performed in an office environment. During this test, the robot unexpectedly encounters a cardboard cat directly in its path (Figure 31, left). At this point, the robot sends the human a camera image and asks whether or not it is safe to drive forward (Figure 31, center). Based on image interpretation, the human answers “yes” and the robot rolls over the cardboard cat (Figure 31, right). At first glance, this situation may seem contrived, or even laughable. Yet, on closer examination, we find that it is not.

Sonar, ladar, and stereo vision are the range sensors commonly used on mobile robots. With all of these sensors, however, a cardboard cat will generally appear as an obstacle. Thus, unless the robot has specific knowledge of how to explicitly detect this object, and to recognize that it does not pose a danger, it will be forced to stop and make a decision. There are four choices: it can wait for the path to become clear; it can look for a way around the obstacle; it can attempt to drive through (or push aside) the obstacle; or it can ask the human for assistance.

The first option is very conservative. Yet, in an unknown environment, it is entirely possible that a robot might encounter an obstacle about which it knows nothing. In this case, the best the robot can do may simply be to wait or signal that a failure (unable to reach the goal) has occurred. Alternatively, if the robot does expect to encounter dynamic obstacles (people, cats, etc.), which move and which do not remain permanently in one place, a

“wait until clear” strategy might be appropriate. In either case, however, it is obvious that the solution could take a long time to succeed.

The second option is less passive than the first. If the robot has sufficient on-board autonomy (path planning, perception, map building, etc.), it can attempt to plan or look for an unobstructed path around the obstacle. There are two problems, however, with this approach. First, such a path may not exist, or may not be evident, based on the robot’s world model. Second, even if an unobstructed path exists, it may take a long time to find or may consume significant resources to traverse.

The third option is an aggressive one. In order to drive through, or push aside, the obstacle, the robot must assume that doing so will not produce undesirable consequences. Unless we know, or decide, in advance that collisions are both acceptable and tolerable, this approach may incur significant risk (e.g., damage to the robot or obstacle). Consequently, for most applications, this strategy can only be used when we are sure about the results.

The final option turns out to be the best approach. By asking the human for his opinion, the robot is able to find the optimal solution for this situation. Once the human sees the image, he is able to decide that the blocking obstacle is a cardboard cat, that it does not pose a danger to the robot, and that driving over it is acceptable. This allows the robot to avoid needless delay and having to make an unnecessary detour. Thus, if the human and robot collaborate, we can improve in at least one aspect, how “A to B” is performed.

I should point out that if the human is unable to answer “Can I drive through?”, either because he cannot decide it is acceptable or because he is not available to respond, merely asking the question is not detrimental (except perhaps in time) to task performance. The only case in which human-robot dialogue might cause a problem is when the human makes an error (e.g., he mistakes a real cat for a cardboard one). There are, however, numerous ways to address this, e.g., weigh the human’s response against the votes of other decision making modules.

6.2 Collaborative Exploration

Collaborative control has characteristics which make it attractive for humans and robots working in the field. Because the human is able to serve as a resource for the robot, he is able to use his skills and expertise to supplement autonomy. Thus, a geologist could help compensate for inadequacies in robot perception by assisting in sample identification, or for limitations in robot planning by designating regions to search.

Another useful characteristic of collaborative control is that it provides a framework for coordination. Specifically, dialogue can be used to coordinate task partitioning and execution. This would allow, for example, the human to give commands and be notified whenever his assistance (physical or otherwise) is needed, rather than him having to continuously monitor the robot’s execution.

To understand how collaborative control can support a human-robot team in the field, I chose to examine collaborative exploration.

6.2.1 Motivation

Within the next thirty years, a human crew is expected to explore the surface of Mars. Although much preparatory work has already been performed, it would be a serious misjudgment to believe that we will be able to design a productive Mars mission simply by assembling current technology and by employing the same exploration strategy as used for the Moon (Cabrol 1999).

Considerable focus has already been given to human and to robot systems for planetary surfaces. Scant attention, however, has been paid to the development of joint human-robot surface systems. Yet, such systems are attractive for numerous reasons. Rovers could be used to carry equipment, materials, and samples. An autonomous rover could follow a scientist in geological sampling sorties. Rovers could also provide contingency life support or emergency transport. More advanced rovers could be used to assist sampling, survey, and *in-situ* site characterization (Duke 1998; Parrish, Akin, and Gefke 2000; Weisbin and Rodriguez 2000).

6.2.2 RockFinder

To assist my study of collaborative exploration, I developed a robot module, Rock Finder, that locates “interesting rocks” (see Figure 32). RockFinder is not intended for field use (i.e., it does not examine geologic properties), but merely serves as an example of the type of work a rover could perform when assisting an EVA scientist.

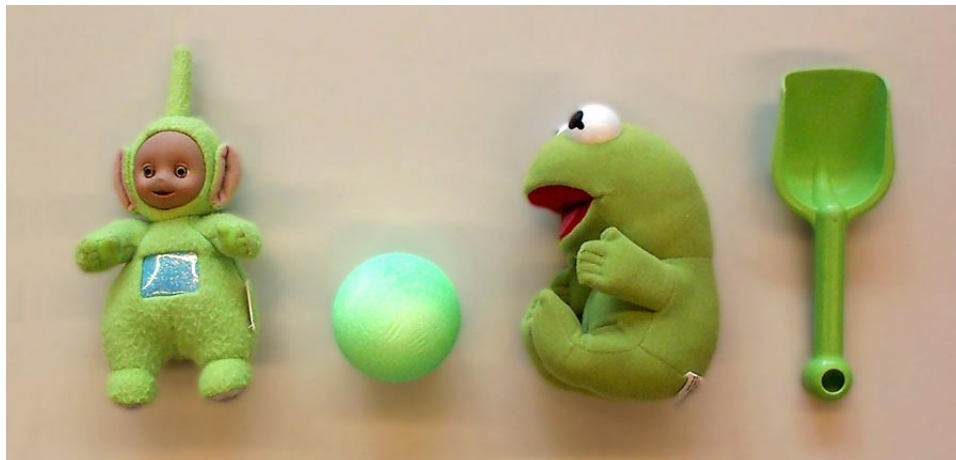


Figure 32. Some interesting “rocks”

RockFinder is a single-camera color vision system, which is implemented using VisLib (see Appendix B.2). It works by searching camera images for contiguous regions having a certain color signature. Color segmentation is an efficient method for object detection because it can be performed via simple pixel filtering. To reduce the effect of varying illumination, shadows, and scene geometry, RockFinder transform image pixels into a normalized color space before filtering (Gevers and Smeulders 1997). RockFinder uses morphological filters to reduce image noise and to merge scattered pixels.

RockFinder operates via a simple state machine (Figure 33). Once an image has been segmented, RockFinder labels each region exceeding certain extents as containing a potential “rock”. It then pauses the robot and asks the human for confirmation. If the human says

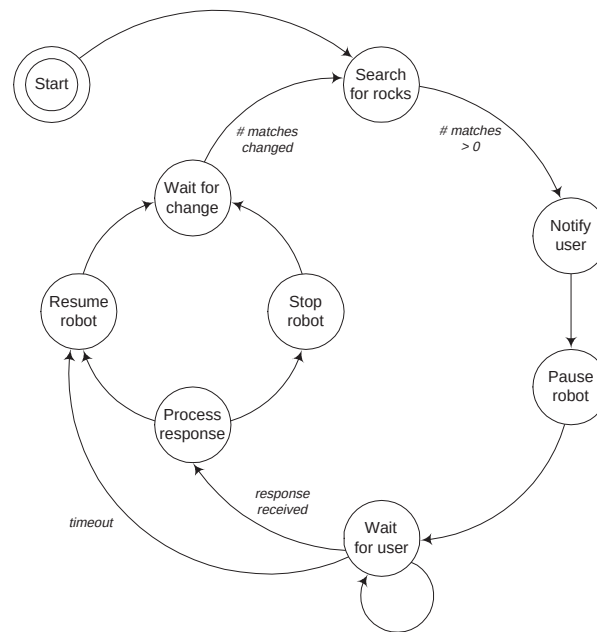


Figure 33. RockFinder state machine

that the object is not a rock, or if he does not respond, RockFinder releases the robot and restarts its search.

6.2.3 Human-Robot Interaction

I conducted a test using RockFinder in a cluttered, indoor environment. For this test, the central task was to search for interesting “rocks”. I selected this task as representative of the work normally performed by a geologist during site characterization.

Since there were many objects in the environment, it would have been fatiguing and tedious for a human to manually search for the rocks. A better approach was to have the human and robot work jointly, collaborating to perform the task. To do this, the human assigned a region to examine and the robot performed the search autonomously. Then, whenever it found a potential match, the robot asked the human to judge if an interesting “rock” had been discovered.

The key difference between the collaborative control approach and other methods (e.g., supervisory control), is that the human and robot work together as a team. Not only does the human set high-level strategy (where to search), but also provides low-level assistance (rock identification).

Figure 34 shows an example of this interaction occurring during the test. RockFinder has identified three possible “rocks” located immediately in front of the robot and has generated a question for the human:

“Are these rocks? If you answer ‘y’, I will stay here.”

This question is presented to the user, accompanied by an image of the matching objects (each marked with a bounding box). At this point, the human has the opportunity to decide if the robot has detected something of value and whether he should examine them in more detail.

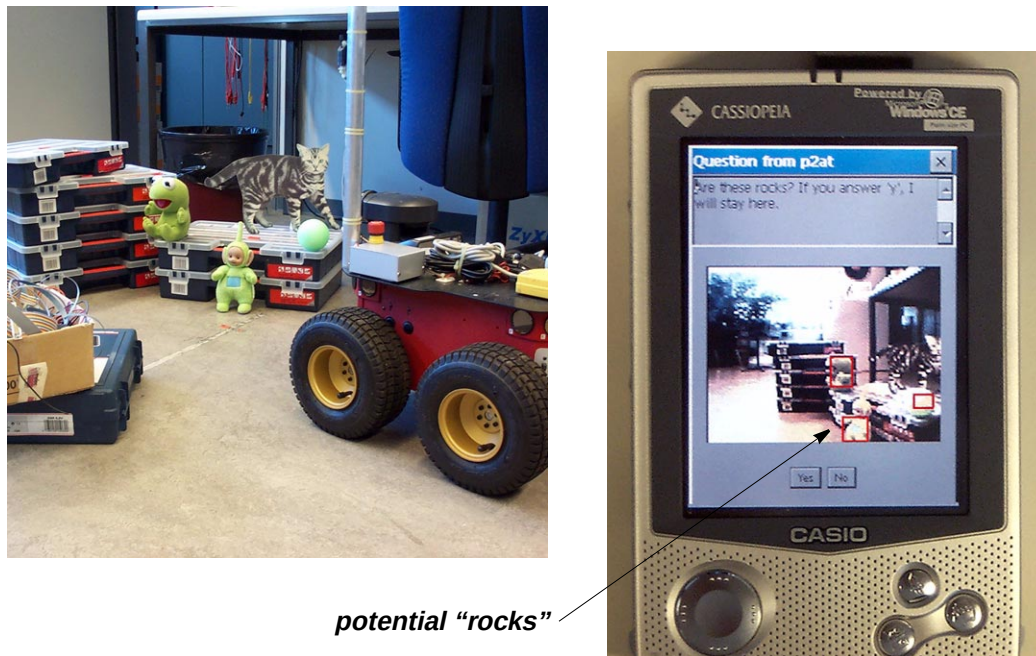


Figure 34. Collaborative exploration

In this particular case, human-robot interaction strongly resembles the relationship between a research assistant and a scientist. Specifically, we have partitioned the task based on expertise: the research assistant (the robot) knows enough to identify potential samples and the scientist (the human) has the skill to decide if the samples are worth keeping. In this way, with human and robot collaborating, exploration is efficient: the human is freed from performing a tedious task and the robot is able to search even though its on-board autonomy (e.g. RockFinder) may have limited capabilities.

It is important to note that this is only one example of how human-robot interaction could occur in collaborative exploration. Obviously, the more competent the robot, the higher the level of dialogue will be. For example, if the robot was capable of visually servoing towards and grasping a rock, then instead of saying “I will stay here”, the robot could offer to autonomously collect a sample.

6.2.4 Related Work

The February 1999 “Astronaut-Rover Interaction Field Test” (ASRO) was the first time that an astronaut and a rover explored a planetary surface together (Cabrol 1999; Trevino et al. 2000). During the test, four human-robot interaction scenarios were studied. However, only a single suited subject was tested, operator command and communication protocols were not rigorously defined, and test time was limited. Nevertheless, ASRO did provide valuable insight into human-robot exploration. Most significantly, it underscored the importance of grounding rover technology development in real-world tests.

Vandapel et al. (1999) conducted a series of experiments during the 1999 Houghton-Mars Project expedition (Devon Island, Canada) to evaluate robotic needs for supporting the field activities of biologists and geologists. This study examined four robot functions: scout, equipment transport, caretaker (monitor scientist during EVA), and co-worker (deploy sensor, register data, explore region). Vandapel et al. (1999) claim that the major

challenges for robotic EVA support are the integration of geological knowledge, autonomous exploration, and human-robot interaction.

There are two primary differences between this prior work and the collaborative control approach. First, neither ASRO nor Haughton-Mars allowed the EVA crew member to directly interact with the rover. During these field tests, the rovers were either teleoperated from a command post or executed pre-planned scripts. In my testing, the PdaDriver enabled the operator to directly control and communicate with the robot. Second, both ASRO and Haughton-Mars assumed a supervisory control model for human-robot interaction. Neither study considered the possibility of using dialogue, especially to enable robot autonomy modules to ask questions of the human.

6.3 Multi-robot remote driving

One of the attractive features of collaborative control is that it allows autonomy and human-robot interaction to vary as needed. This is particularly useful for environments that are unknown, dynamic, or difficult to plan for in advance. In these environments, system configuration (e.g., module parameters) may need continuous or repeated modification as the situation varies. Although such changes could be performed autonomously (e.g., Muscettola et al. 1998), in many cases human decision or approval is needed. Thus, providing dialogue between the human and robot is advantageous.

Being able to dynamically modify autonomy and interaction is also useful when an operator (or operators) must work with multiple robots. In such situations, especially when simultaneous control or supervision is required, the level of autonomy for each robot may need to be adjusted. For example, an event may occur which forces the operator to limit interaction to a single robot for an extended period of time. As a consequence, to prevent deadlock or blocking (i.e., robots waiting for human input), the level of autonomy for the other robots will have to be increased.

To examine how variations in autonomy and human-robot interaction might occur in practice, I chose to study a task involving multi-robot remote driving for reconnaissance and surveillance (Fong, Grange, Thorpe, and Baur 2001).

6.3.1 Motivation

The American military is currently developing mobile robots to support future combat systems. These robots will be used to perform a variety of reconnaissance, surveillance and target acquisition (RSTA) tasks. Because these tasks have traditionally required significant human resources (manpower, time, etc.) and risk taking, a primary area of interest is determining how to mitigate, or minimize, these factors by having a single operator control multiple robots.

As I discussed in Chapter 2, even when only a single robot is involved, conventional vehicle teleoperation is often problematic. With multiple robots, the difficulty is increased because the human is forced to divide his limited cognitive and sensorimotor resources among the robots. This is true whether the human controls the robots individually or as a group (e.g., in formation). Moreover, even if one or more robots work together (i.e., robot-robot collaboration), we must still find ways to direct the human's attention, so that he can help robots individually when necessary.

In general, whenever a single operator controls multiple robots, his workload becomes higher and task performance (completion time, error rate, etc.) deteriorates as a result. Moreover, if the robots are deployed in an unfamiliar setting or in the presence of deadly hazards, the operator has to dedicate significant attention to assess the remote environment each time he is forced to switch context.

One way to mitigate these problems is to use collaborative control. Since the human can only focus his attention on one robot at a time, we arbitrate among the queries so that the human is always presented with the one that is most urgent in terms of safety, timeliness, etc. This helps to reduce the level of attention and control the operator must dedicate to each robot. Consequently, the human can more effectively perform simultaneous, parallel control. In addition, because each robot is aware that the human may not be able to respond, it can still try to resolve problem on its own.

6.3.2 MotionDetector

To support the study of multi-robot remote driving for RSTA tasks, I developed a robot module, MotionDetector, to detect motion from a stream of camera images. MotionDetector is not intended as a robust video surveillance and monitoring tool. For example, it does not discriminate between moving humans and pixel variation due to varying scene illumination. Rather, MotionDetector merely serves as an example of the type of work a robot could perform when performing surveillance.

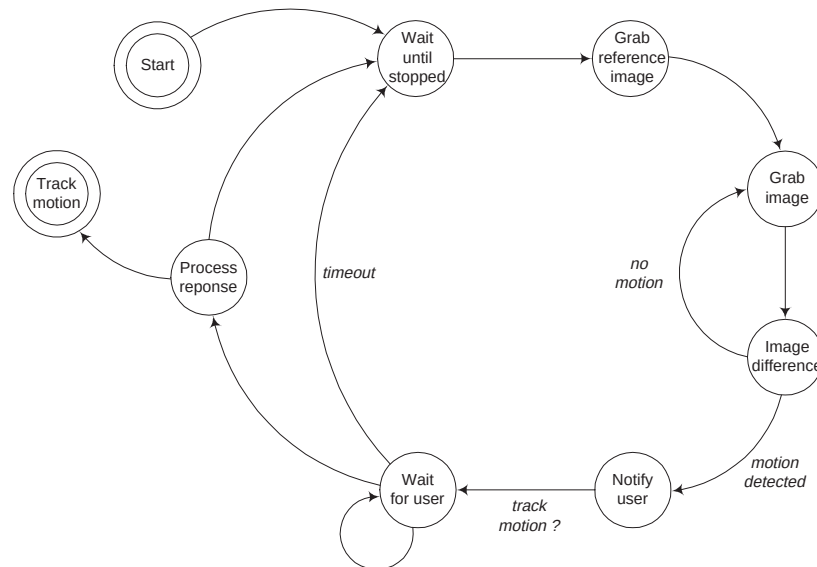


Figure 35. MotionDetector state machine

MotionDetector is implemented using Vislib (see Appendix B.2) and operates via a simple state machine (Figure 35). Whenever the robot is stationary, MotionDetector begins acquiring camera images and computing inter-frame differences. If it finds significant pixel changes or motion, it notifies the human.

6.3.3 Remote Driving Tests

I conducted two tests in which a single operator was responsible for controlling two mobile robots. These tests were designed to study basic RSTA tasks that would normally be performed by humans, but which could fairly easily be performed by mobile robots. The first test examined an indoor surveillance mission and the second test looked at an outdoor reconnaissance mission.

The two robots used in these tests are described in Section 5.2. Although the robots are not physically identical, they have similar characteristics and sensor suites. In addition, both robots were operated with identical controller software. Thus, the operator was able to assume that the robots would perform in a comparable manner.

Indoor surveillance

In the first test, the operator used the two robots to scout an unknown indoor environment. This mission required two primary tasks to be completed. Because the environment was unknown, the operator was instructed to begin by exploring the area. Once the operator has become sufficiently familiar with the area, he was asked to use the robots to watch for intruders entering the area.

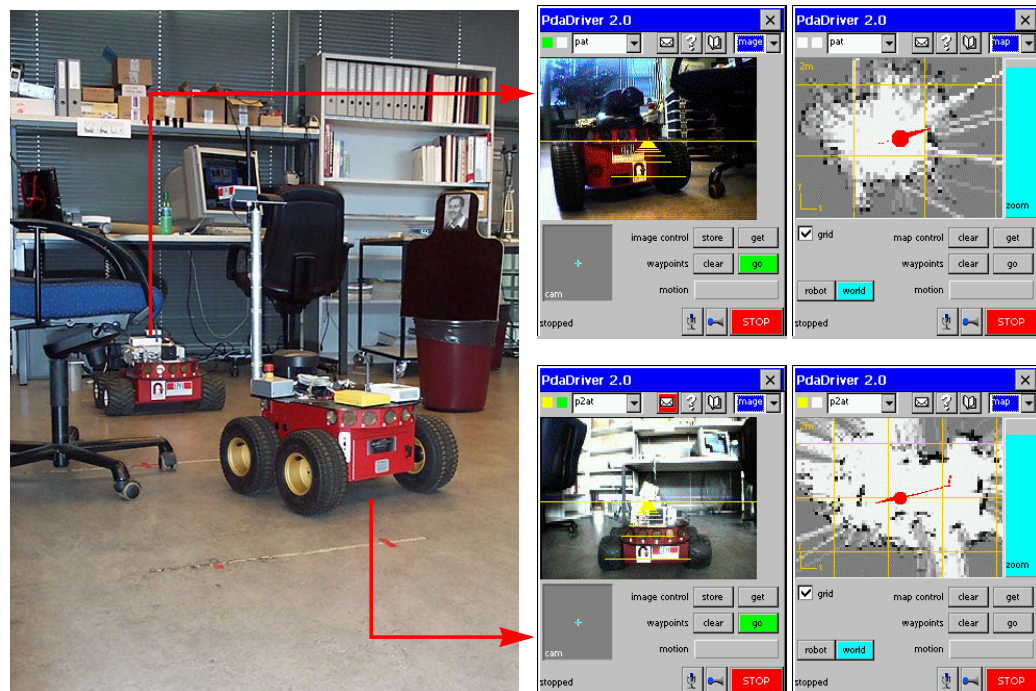


Figure 36. Indoor surveillance with two robots.

Left, PioneerAT and Pioneer2-AT; right, PdaDriver displays.

To perform the first task, the operator used PdaDriver to simultaneously control the two robots (Figure 36). Since the environment was relatively benign (i.e., no dangerous hazards or difficult terrain), little human-robot dialogue was observed. The most significant question asked by a robot was:

The environment is very cluttered. What is the fastest I should translate (current setting is 50 cm/s)?

This question was triggered by a safeguarding behavior when one of the robots detected that there were numerous obstacles nearby. Thus, the robot decided to ask the operator for guidance in setting its translation speed.

The primary problem encountered in this test was that each robot mapped the environment using only its on-board sensors. Each robot's map, therefore, only displayed the portion of the environment explored by that robot (see Figure 36, right). As a result, the operator was forced to mentally merge the individual maps in order to obtain a complete model of the environment. This shortcoming is due entirely to the current map building implementation, which was not designed for multiple robots.

For the second task, the operator received greater benefit from collaborative control. Once he had positioned the robots in appropriate surveillance locations, the operator was able to say "Notify me when you think you have detected an intruder".

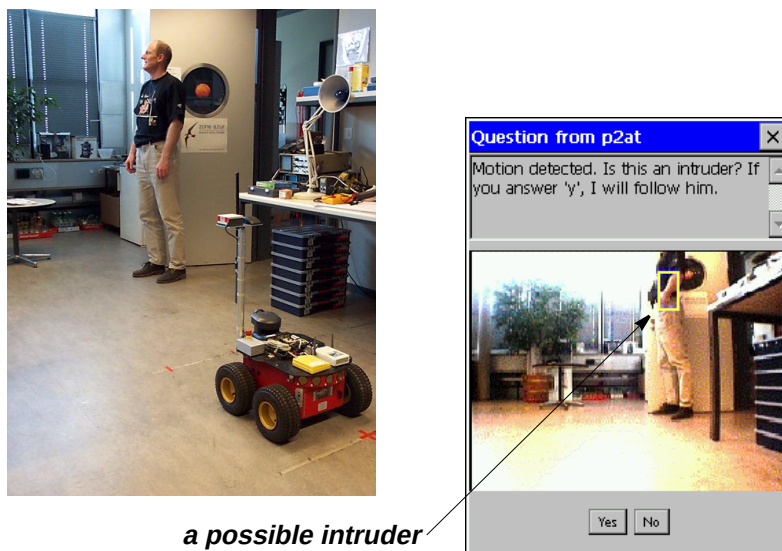


Figure 37. Human-robot interaction during surveillance

Figure 37 (left) shows an intruder walking into the room. A robot, which had been positioned near the door, detected this motion and generated the following question for the human:

Motion detected. Is this an intruder? If you answer 'y', I will follow him.

This question was presented to the user accompanied by an image showing the motion area (marked by a bounding box), as shown in Figure 37, right. At this point, the human became responsible for deciding if there really was an intruder present in the scene, and if so whether the robot should proceed to follow (track) him.

Although, in this case, the robot asks the human for confirmation before proceeding, it does not necessarily have to. With collaborative control, the robot has both the freedom and the opportunity to take the initiative. In fact, if the human does not respond within some period of time, the intelligent decision for the robot might be to take the initiative and track the potential intruder until the human says to stop.

As was the case in collaborative exploration, this is only one example of the interaction that could occur between human and robot. If the robot had other capabilities, such as recognition (e.g., people identification) or intruder countermeasures, then further dialogue could be used to select the appropriate action to take.

Outdoor reconnaissance

In the second test, the operator remotely drove the two robots through an unfamiliar outdoor environment (Figure 38). The objective for this test was to perform cross-country navigation and reconnaissance in the presence of moving and static hazards. Because the environment had not been previously explored, the operator was forced to rely on way-point driving and on-robot safeguarding to conduct the task.



Figure 38. Outdoor reconnaissance with two robots

During the test, the operator commanded reconnaissance motions for both robots. Because the robots operated with safeguarding behaviors, each one was able to ask safety-related questions as it traversed unknown terrain.

Figure 39 shows four of the questions generated by the two robots. During the test, the Pioneer-AT (“pat”) asked (Figure 39, top left):

Stopped due to low power (12.1 V). What should the safety level be (currently 12.2 V)

It is normal engineering practice to specify a safety level outside of which the system should not be operated. Yet, there are some occasions when a system must be used beyond its design specifications. This is particularly true for situations in which system loss is considered acceptable if a goal is achieved (e.g., military combat missions). Thus, this type of question may be entirely reasonable to ask. What is important is that because it is self-aware, the robot is able to notify the human when a limit is reached.

The Pioneer-AT also asked (Figure 39, top right):

Can I drive through?

This situation is identical to the “A to B” application (Section 6.1). As I discussed previously, by having the human and robot collaborate, the robot may be able to take advantage

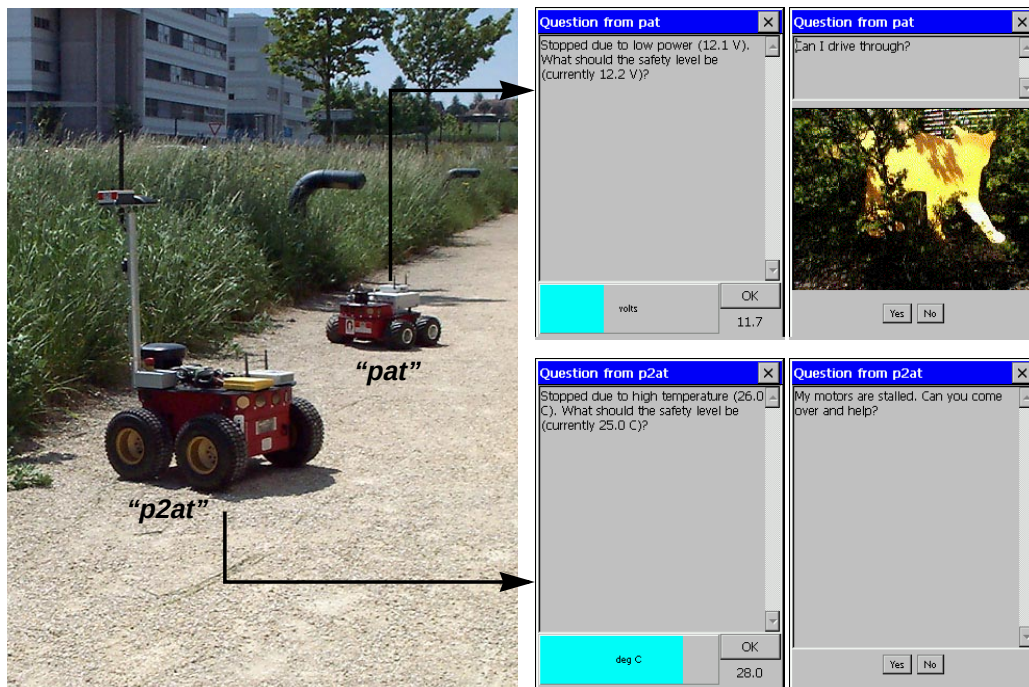


Figure 39. Human-robot interaction during outdoor reconnaissance

of human perception and decision making (i.e., it is acceptable to drive through) rather than wasting time or resources to find an alternate solution.

During the test, the Pioneer2-AT (“p2at”) asked (Figure 39, bottom left):

Stopped due to high temperature (26.0 C). What should the safety level be (currently 25.0 C)

As with the Pioneer-AT, this question demonstrates how the human and robot can collaborate to set a safety level appropriate for mission objectives.

Finally, at one time, the Pioneer2-AT found itself unable to move and detected a motor stall. Thus, it said to the operator (Figure 39, bottom right):

My motors are stalled. Can you come over and help?

When faced with this question, the operator responded “yes” and went to assist the robot. He discovered that two of the robot’s wheels had become wedged against an obstacle, which had not been detected by sensors used for collision avoidance. If the operator had answered “no”, the robot’s safeguarding system would have deactivated the drive system to prevent damage and rejected further motion commands.

In this task, a key benefit of collaborative control was that questions from multiple behaviors (on both robots) were arbitrated, so that the most critical ones were always presented first to the operator. Thus, the operator was always guided to the robot which most urgently needed his attention.

6.4 Summary

In each of the applications presented in this chapter, collaborative control enabled the robot to perform better than if left on its own. Specifically, in situations where the robot did not know what to do, or in which it was working poorly, having the human provide assistance (i.e., giving a simple answer) was all that was required to get the robot out of trouble and to improve its operation.

User Study

To understand how users interact with a collaborative control system, and to obtain feedback on my interface design, I performed a Contextual Inquiry (CI) based user study. CI is a structured interviewing method, adapted from ethnographic research, for grounding the design of interactive systems in the context of the work being performed. I conducted a CI study of eight users with a wide range of background and experience. In the study, I divided the users into two stereotype user classes (novice and expert) and asked each to explore an office environment using my collaborative control system.

7.1 Evaluation

Evaluating how well a vehicle teleoperation system works is difficult. Since the range of applications is so broad, and because the types of tasks which must be performed are so diverse, there is no single, universal standard by which we can determine the absolute “goodness” or capability of a system. Consequently, numerous methods have been developed for evaluating different aspects of operation and performance (McGovern 1988, 1990; Glumm, Kilduff, and Masley 1992; Adams 1996; Krotkov et al. 1996b; Kay 1997).

A common strategy for evaluating manually controlled vehicles (e.g., remotely piloted aircraft) is to measure the vehicle’s “handling qualities” (i.e., the characteristics which govern how well a pilot is able to perform tasks) through the opinion of skilled pilots. Various rating scales have been developed to assess pilot opinion, the most widely used of which is the Cooper-Harper Rating¹ (CHR) scale (Cooper and Harper 1969).

It is evident, however, that pilots having different backgrounds, experience, skill, and training may differ in their assessment of a vehicle. Moreover, pilots are not immune to time-varying physiological factors (e.g., fatigue) and may be influenced by psychological biases². Thus, it is often difficult to obtain repeatable assessments. As a result, the standard practice is to collect a statistically significant number of pilot opinions before assigning a definitive rating.

In terms of measuring human performance, a great number of methods have been developed. The American National Standard ANSI/AIAA G-035-1992 (ANSI/AIAA 1993) describes non-physiological guidelines appropriate for human factors engineering, training, and test and evaluation. These guidelines can be applied to any technology that is controlled and operated by humans (“system personnel”), or that affects humans, whether they are system personnel or not.

1. also called the “Handling Qualities Rating” (HQR) or “Pilot Rating” (PR).

2. e.g., it has been observed that test pilots frequently hesitate to give a perfect rating.

For example, workload, which can be defined as “the effort expended by the human operator in accomplishing the imposed task requirements” may be measured in terms of task performance. One of the most widely used techniques is the *secondary task*. This technique requires the operator to perform a primary task (within specified requirements) while simultaneously using any spare capacity to perform a secondary task. The reduction in secondary task performance is then taken as a measure of workload. Well-known secondary tasks include card sorting, choice-reaction time, mental mathematics, etc. (ANSI/AIAA 1993).

For this thesis, the question I needed to answer was: what is an appropriate method for evaluating collaborative control? Although it would have been possible to collect metrics such as CHR, or measurements of operator workload, I was not sure about the value of such an approach. For one thing, studies which produce quantitative metrics rarely provide deep insight into how operators learn to use a system (e.g., what strategies they adopt in order to achieve their goals). For another, such metrics do not generally provide the feedback required, nor guidance necessary, to improve an existing design.

My approach, therefore, was to avoid quantitative measures and to focus on qualitative assessment. The primary objective was to understand how collaborative control influences human-robot interaction. In particular, I wanted to observe how various types of users react to robot dialogue, how their actions and strategies evolve over time, and what are the strengths and weaknesses of the existing system. To do so, I conducted a user study based on Contextual Inquiry.

7.2 Contextual Inquiry

Contextual Inquiry (CI) is a structured interviewing method for grounding the design of interactive systems in the context of the work being performed (Holtzblatt and Jones 1993). CI is an adaptation of field research techniques taken from psychology, anthropology, sociology, and hermeneutics. Because the data it provides is primarily subjective in nature, CI is most appropriate for qualitative system assessment, rather than for performance measurement.

An interviewer performs CI by observing users while they work, asking questions as they perform tasks in order to understand their motivation and strategy. Typical questions are: What are you doing now? Is that what you expected to happen? What do you especially like or dislike about this tool? Through conversation and discussion, the interviewer and the user develop a shared understanding of the work. Thus, CI supports system development by providing a mechanism to help people identify and articulate their work experience (Beyer and Holtzblatt 1999).

CI is based on three principles: context, partnership, and focus. *Context* means having users describe their work as they perform it in their work environment. In other words, the best way to understand work is when users are fully immersed in what they are doing. *Partnership* is the concept that the user should share in guiding the design process. The key to partnership is maintaining a conversation that allows the user and the interviewer to create shared meaning about the work experience. *Focus* describes the objectives we are trying to achieve with the study. Focusing on specific goals helps guides what is attended to or ignored, what questions are asked, and what is probed further.

A fundamental problem in CI is: how does the interviewer encourage the user to “open up” and provide key information? One approach is the *apprenticeship model* (Beyer and Holtzblatt 1995). With this model, the interviewer takes on the role of an apprentice and asks the user to play a master (expert), who’s job is to teach how to use the system and to do the work. The apprenticeship model encourages users to shape and guide the conversation. It also helps ground the conversation on concrete details: what procedures are needed for work, where the problems are located, etc.

7.3 Study Objectives

The focus of my study was to *understand how collaborative control influences human-robot interaction* in the context of remote driving. Thus, users were required to perceive the environment, to generate commands, and to assist safeguarding autonomy. The three primary study objectives were:

Observe system use. An important part of understanding an interactive system is to observe how users learn how to use it. The key questions were: How do users explore the interface and the system’s capabilities? What control strategies do they use? How do they switch from giving commands to answering questions?

Gain insight into dialogue and HRI. Perhaps the most distinguishing feature of collaborative control is that the robot treats the human as a resource. Thus, I wanted to learn: How do users relate to a robot that asks them questions? Does this influence how users perceive the robot? How does dialogue affect the user experience?

Obtain design feedback. In order to improve a system, we must ascertain which design elements are key to its success. In particular: what works well, which functions need to be improved/discarded, and what features must be added.

7.4 Test Summary



Figure 40. Test subject and Pioneer2-AT in the CI study environment

I conducted the study in a cluttered, indoor environment (a research laboratory) at the Swiss Federal Institute of Technology in Lausanne (Figure 40). Because collaborative

control is designed to allow all users (novices and experts alike) to perform vehicle teleoperation, I tested a group of users with a broad range of background, skills, and experience.

The study was performed during normal office hours. No physical work, other than holding a PDA and stylus, was required. Psychological risk associated with this study was determined to be negligible or zero. A detailed description of the study, including test session support documents, is given in Appendix C.

7.4.1 Test Procedure

During each test session, qualitative data was collected in the form of a written questionnaire, written note taking, and user interview. No audio or video recording device was used. No direct quantitative measures (e.g., error rates) were made of either human or system performance. However, a log of all system activity, including robot action and human-robot dialogue, was recorded.

A schedule of a typical test session is presented in Table 12. Approximately one hour of time was required per subject. In the initial phase of the session, the subject was welcomed and then given an overview of the study objectives, methodology (contextual inquiry), and test procedures. Next, the confidentiality procedures were described and the subject was asked to sign an informed consent form (Appendix C.1.4). To establish demographic and background data (e.g., experience with vehicle teleoperation), the subject was asked to complete a questionnaire (Appendix C.3). Based on data from the questionnaire, the user was classified as matching either the “novice” or an “expert” stereotype. In total, there were five “novice” and three “expert” users.

Table 12. Test session schedule

	Phase	Time (min)	Description
preparation	welcome	3 - 5	overview of study goals and test session
	informed consent	2	confidentiality procedure and form
	questionnaire	3 - 5	gathering of background information
work	practice task	5	CI practice (computer solitaire)
	task session	30 - 45	remote driving with collaborative control
debrief	discussion	5	clarification and unanswered questions
	wrap-up	2 - 3	summary of what was observed

After the preparatory part of the session had been completed, the work portion began. To gain familiarity with CI, the user was first asked to perform a practice task using the PDA. Solitaire (Figure 41) was chosen for this task because it is a game that users were likely to have played previously, either with cards or on a computer. As he played, the user was encouraged to act out the role of a “master gamesman” teaching an apprentice how to play the game.

Once the user was comfortable with CI, the collaborative control system was started and configured to match the user stereotype. The user was then requested to perform a remote

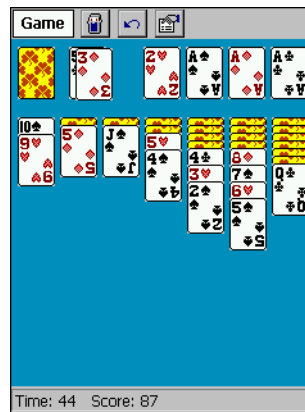


Figure 41. Solitaire was used as to practice contextual inquiry.

driving task, using the PdaDriver to interact with the robot. If the user was a “novice”, he was asked to use the robot to navigate the laboratory. If the user was an “expert”, he was asked to drive the robot out of the room and to explore the surrounding area.

While the user worked, extensive written notes were taken of what the user was doing and saying. To elicit information from the user, as well as to confirm his observations, questions such as “What are you doing?” and “Why are you doing that?” were repeatedly asked. From time to time, robot questions were triggered by injecting artificial sensor data into the system (high temperature, low power, etc.) or by simulating emergency conditions. For example, a radio controlled “emergency stop” was surreptitiously used to cause motor stalls.

After the user had worked for a period of approximately 45 minutes (practice task and remote driving combined), a short debrief was performed. During this time, the user was asked to describe and summarize his experiences. Any points of confusion, especially relating to observed user behavior, were also clarified. A summary of what was observed and learned during the session was then presented for user approval. Finally, the user was thanked for having participated in the study.

7.4.2 Participants

Eight volunteers (six male, two female) participated in the study. None of the test subjects had prior knowledge of collaborative control, nor were familiar with the system design or implementation¹. To preserve anonymity and confidentiality, each subject was assigned a user ID.

Demographic data is presented in Table 13. The test subjects range in age from 23 to 38 years old. All subjects are well-educated, having received one or more university degrees. A majority of the subjects have technical backgrounds, primarily in engineering. Three of the subjects are currently students, two are researchers, and two work in the business field.

1. one subject had previously used an early version of the PdaDriver, which did not support collaborative control.

Table 13. Test subject demographic data

User ID	Age	Sex	Occupation	Education (highest degree)
Chakotay	23	M	student	engineering (B.S.)
Janeway	38	F	executive recruiter	business (B.S.)
Neelix	26	M	student	engineering (M.S.)
Odo	25	M	student	engineering (M.S.)
Picard	32	M	business consultant	engineering (M.S.)
Spock	34	M	professor	computer science (Ph.D)
Uhura	27	F	none	art (B.S.)
Worf	30	M	scientist	engineering (M.S.)

Background information for each of the test subjects is summarized in Table 14. All of the subjects are experienced drivers. The majority consider themselves to be experienced computer users (only one subject reported being a novice). All subjects use computers on a daily basis. Only three subjects indicated prior experience with remote driving.

Table 14. Test subject background data

User ID	Driving experience (frequency)	Computer experience (usage)	Video game experience (types)	Remote driving experience (times)	User stereotype (assigned)
Chakotay	7 yr (daily)	nerd/hacker (daily)	weekly (all types)	toy cars (many)	novice
Janeway	20 yr (daily)	experienced (daily)	rarely (action/card)	none	novice
Neelix	5 yr (daily)	power user (daily)	rarely (action/strategy)	telerobots, toy cars (many)	expert
Odo	7 yr (daily)	nerd/hacker (daily)	rarely (action)	telerobots, toy cars (several)	expert
Picard	13 yr (daily)	nerd/hacker (daily)	daily (action)	telerobots (many)	expert
Spock	18 yr (daily)	nerd/hacker (daily)	monthly (action/strategy)	none	novice
Uhura	12 yr (weekly)	novice (daily)	rarely (card/strategy)	none	novice
Worf	11 yr (daily)	power (daily)	rarely (other)	none	novice

7.5 Results

7.5.1 Using the System

Initial learning

Because none of the users had prior experience with collaborative control, they each spent several minutes learning how to use the system. Two basic exploratory learning styles were employed: “incremental” and “experimental”. Five users employed the incremental style. These users were very cautious and methodical in learning how the system works. They worked slowly, making sure that they fully understood the effect of each interface control before incrementally exploring others. They were also very careful when answering questions, preferring to give conservative responses whenever possible.

The other three users were “experimental” learners. From the start, they were very aggressive as they worked. For example, the first command that Chakotay gave was an eight-point path. These users rushed to find the system’s limits, trying to determine what was possible, and what was not, as quickly as possible. Moreover, whenever the robot asked a question, they responded rapidly, with very little reflection. As a result of this aggressive style, these users initially made frequent errors and were sometimes confused by the robot’s questions.

Table 15. Learning style and final performance

User ID	Stereotype	Learning style	Final performance (subjective assessment)
Chakotay	novice	experimental	after initial “see what it can do” (8 point path), settled down to being very conservative (1 point paths)
Janeway	novice	incremental	very good control (especially for maneuvering in fairly cluttered areas), excellent use of image and map
Neelix	expert	incremental	excellent control (able to drive through doorways), very good at assessing system limitations
Odo	expert	experimental	excellent control (able to drive through doorways), very rapid at answering questions
Picard	expert	incremental	very good control, extremely perceptive at finding design faults and system limitations
Spock	novice	experimental	good control (especially for parking), somewhat hesitant about answering questions
Uhura	novice	incremental	very at ease with system, very good control
Worf	novice	incremental	excellent control (performed several impressive long-distance paths and precision maneuvers)

The final (subjective) performance of all users is summarized in Table 15. By the end of the session, all users reported that they were confident in answering questions. It is interesting to note that, regardless of learning style, all users achieved a high level of control

skill. Moreover, there was little observable, qualitative difference in the ability of novices and experts at controlling the robot's movements.

Learning style appears to have mostly affected how users initially handled the robot's questions. Incremental learners, because they worked slowly and methodically, were better able to integrate answering questions into their work practice. Experimental learners, on the other hand, were sometimes confused by the robot's questions. One possible explanation is that these users were exploring "globally" (i.e., trying to find the systems limits) whereas the questions were focused on "local" issues. This suggests, therefore, that dialogue management should consider how users learn, to adjust the rate and nature of questions asked during the learning phase.

Use of control modes

For remote driving, all users were instructed in the use of PdaDriver's image and map modes. Experts were also given the option of using direct mode, which did not provide any motion feedback. All users stated that image mode was more "precise" and easier to use. Users had mixed opinions about map mode: three reported that they had difficulty understanding what the map showed, the others thought that it clearly indicated where the robot could drive.

Chakotay. The map is good because you can tell where you can go. But, it's also bad because it does not tell you what is there.

Odo. The map is very useful because all the white areas are safe.

Spock. I assume that the map reflects better what the robot sees since it is built from sensors.

Once they were familiar with the system, many users adopted a strategy of switching between image and map mode before giving a command. This was due to a strong command bias (image mode for "precision", map mode for long range motion) and the different views of the remote environment.

Neelix. This (switching modes) helps me see where I am going.

Chakotay. The map is better for global context and for avoiding collisions . . . I like image mode for local context and for moving.

The expert users decided that direct mode was not useful. In particular, the lack of motion feedback was a significant shortcoming and all the experts stopped using direct mode after just two or three attempts.

Switching between controlling and answering

With the current system, an indicator flashes whenever the robot has a question to ask. To answer a question, the user must perform three steps: (1) notice that the indicator is blinking; (2) press the indicator to retrieve the question; and (3) input and submit his response.

This style of interaction allows the user to retain control of the dialogue. However, since steps (2) and (3) are blocking operations (i.e., the interface prevents the user from doing anything else), it also means that user must stop controlling the robot in order to respond to its questions. As a result, some users avoided answering questions for long periods of time, particularly when they were engaged in a complex task (e.g., precision driving).

Other users immediately stopped what they were doing whenever a question appeared. For these users, satisfying the robot's needs always had priority over achieving task objectives:

Janeway. In general, I would try to answer as soon as possible. That's my personality. Especially coming from the robot since it probably means it's urgent and would help make it work better.

Worf. I always tried to take care of (respond to) the messages (as soon as they appeared).

The most interesting consequence of having to switch between controlling and answering was that Neelix and Odo developed a technique for precision driving. This occurred because both users observed that the robot often asked "Can I drive through?" when it was operating in a tight spot. Thus, their approach was to command the robot to drive to a far waypoint (e.g., through an open doorway) and then wait for the question to be asked (to which they would always answer 'y'). As a result, these two users were able to drive the robot through very narrow openings. All the other users, who did not discover this strategy, were unable to do so because the robot's safeguarding was very conservative and its autonomy was inadequate for it to perform these maneuvers unassisted.

7.5.2 Answering Questions

Same question, different users

The study revealed that different users may respond quite differently when asked the same question. Consider, for example, the way in which users responded to "Stopped due to high temperature. What should the safety level be (currently 21 degrees C)?". This question was triggered by a simulated ambient temperature increase, which had been scripted to occur at approximately the same time in each test session. When the question appeared, each user was informed that "the simulated temperature may continue to increase" and that "the robot will only suggest values which it thinks are safe".

Here is how half the users responded:

Chakotay. I'm trying to set the temperature high. I'll assume that I can set it high.

Janeway. Okay, I'll set it really high so I won't be bothered again.

Uhura. I'll make the temperature higher.

Worf. Okay I'll just set it a little higher. It will probably ask me again later.

Chakotay and Janeway chose to set the safety level as high as possible, so that they would not be bothered or interrupted, even if the temperature dramatically increased. Uhura and Worf preferred to only slightly increase the safety trigger, because they wanted to be informed again (i.e., asked another question) if the temperature continued to rise. The explanation for this difference is personal preference: some users preferred to be kept informed, others were willing to let the robot take care of future problems.

Many questions in a short time

Occasionally in each test session, the robot would ask many questions in a short period of time. This usually occurred when the robot was operating in a very cluttered part of the room, or when multiple safeguarding behaviors were triggered in response to a difficult situation. Picard, for example, attempted to drive through a very narrow passage at the same time the simulated temperature increase occurred:

Robot. Stopped due to high temperature. What should the safety level be (currently 23 degrees C)?

Picard. 25 degrees C.

Robot. Can I drive through?

Picard. No.

Robot. The environment is very cluttered. What is the fastest I should translate? The current setting is 50 cm/s. I would recommend less than 37 cm/s.

Picard. 40 cm/s.

Robot. The environment is very cluttered. What is the fastest I should translate? The current setting is 40 cm/s. I would recommend less than 37 cm/s.

Picard. 38 cm/s.

Robot. The environment is very cluttered. What is the fastest I should translate? The current setting is 38 cm/s. I would recommend less than 37 cm/s.

Picard. These messages are annoying. How come it keeps asking?

At this point, Picard said that he was tired of answering questions and began ignoring the robot. He continued to do so until a safeguard behaviors forced him to answer a question (i.e., the robot refused to move until he responded). This indicates that one potential problem of collaborative control is that the user may grow weary of interacting with the robot. We will return to this point later.

In contrast to Picard, Janeway was highly receptive to questions throughout her session. As noted earlier, she always quickly answered questions because she believed that doing so would make the robot work better.

Robot. My motors are stalled. Can you come over and help? If you answer no, then I will turn off motion control and will not be able to move.

Janeway. Okay I'll go help it. Yes. (*Janeway walks over and helps the robot*)

Janeway. How come it's not moving? Oh, there's an error message.

Robot. Motion control is currently turned off. Shall I enable it? If you answer no, then I will not be able to move.

Janeway. Okay, yes.

Robot. Stopped due to high temperature. What should the safety level be (currently 21 C)?

Janeway. Okay, I'll set it really high so I won't be bothered again. 25 C.

As this exchange shows, answering multiple questions in a short period can be helpful, especially when the robot is stuck and in need of external assistance.

Disappearing (unanswered) questions

On several occasions, users were so engaged in their work that they did not notice that a question was waiting to be answered (i.e., that the query indicator was flashing), or they chose not to handle it right away. If the pending question had a timeout (see Table 5), however, then it was possible that the user never realized he had missed the question or that he never had the opportunity to answer it. This was because as soon as the timeout occurred, the question was deleted, the query indicator stopped flashing, and the robot took autonomous action if it could.

When I pointed out to Janeway that she had missed several questions, she said, “If I had noticed the ‘error’ flashing, I would have answered it right away.” This indicates that a flashing indicator may not always, by itself, be sufficient to attract the user’s *locus of attention* (Raskin 2000). Possible remedies include adding other indicators, using sound, etc. However, if the user is sufficiently absorbed by what he is doing, there is no guarantee that supplemental measures will be effective.

Spock. Would be interesting to see what it would have asked me.

Uhura. Why did it flash and stop? Just because I’m slow?

Worf. Oh message . . . Well, there was something blinking here.

Three users (Spock, Uhura and Worf) recognized that questions sometimes “disappear”, which disturbed them. When reminded that the event log contains a record of all human-robot dialogue, all three immediately looked at it to see what question the robot had tried to ask. From then on, however, these users referred sparingly to the event log and did not seem bothered when they missed a question. Possible explanations include: they trusted the robot to act correctly without their response, they found that reading the log was too difficult or too time-consuming, or they did not care if they missed a question. Another possibility is that a log is really only useful for reviewing lengthy conversations. Further study is needed to understand why the event log was rarely used.

Repeated questions

Some users remarked that, at times, the robot asked the same question over and over. This occurred because the robot’s power and environment clutter safeguards sometimes had difficulty assessing the current state. The power safeguard, which monitors battery voltage, halts the robot whenever the power falls below a safety level and asks, “Stopped due to low power. What should the safety level be (currently XX Volts)?”

The problem is that battery voltage may fluctuate significantly (i.e., large excursions) during driving. This occurs, for example, when the battery is low and the robot executes high-torque motions, such as skid-steered turning in place. As a consequence, the robot may ask the power question several times, until the level stabilizes.

The environment clutter safeguard, which monitors the local area obstacle density, is designed to adjust the robot’s translation speed. When it perceives that the obstacle density changes, it asks, “The environment is very <cluttered | open>. What is the <fastest | slowest> I should translate? The current setting is XX cm/s. I would recommend less than XX cm/s.”

When the amount of environment clutter varies widely, or if the user does not follow the robot's recommendation, the robot may ask this question several times, until a suitable setting is found.

Chakotay. The power questions seemed annoying. They kept appearing over and over. I felt like 'All right I answered it once, so go away'.

Neelix. Often I saw the blinking and I thought, it's probably the same question again, so I'll ignore it. It's kind of like (e-mail) spam.

Picard. These messages are annoying. How come it keeps asking?

Spock. I'm going to assume that the message is about low power again, so I'll ignore it.

From these reactions, we can see that repeatedly asking the same question may affect human-robot interaction in two ways. First, as evidenced by Chakotay's statement, the user may become annoyed with the robot, and thus may be less disposed to giving a good (thoughtful, accurate, etc.) response to questions. Second, the user may grow weary and avoid answering further questions. This is significant because, as I discussed earlier, it points out a limitation of collaborative control. In particular, it shows that human assistance is a limited resource: ask a question too often, or in too short a time, and the human will stop collaborating. This can have negative consequences because the robot may not receive help when it urgently needs it.

One way to address this problem would be to implement more sophisticated dialogue management. Rather than stateless attribute filtering, query arbitration could consider temporal factors such as how many messages the user has answered in a period of time, or how long it has been since the "same" question was last asked. Another approach would be to let the human provide feedback to the robot. For example, if the user feels that the questions are getting to be repetitive and tedious, he should have the ability to tell the robot to "deal with these questions yourself" and "only contact me when something urgent or unusual occurs".

Are the questions useful?

At the end of each test session, users were asked to summarize their experiences and their opinion about the robot's questions.

Janeway. I thought the questions were good. Do robots usually ask questions?

Janeway, a novice user, was surprised to learn that robots normally do not ask questions. She said that in movies ("Star Wars", "AI", etc.), robots are portrayed as being capable of intelligent interaction. Thus, she expected that actual robots would exhibit aspects of this behavior in the real-world.

Neelix. I thought the questions were cool. They seemed to be useful, but often they need to give more detail. Especially 'Drive Through' which needs more information about what/where it (the robot) is driving through.

Neelix's statement highlights the importance of context. A question without sufficient detail may be difficult for the human to answer correctly. Moreover, simply supplementing a question with more information may not be sufficient. What is essential is that the information be appropriate, relevant, and precise.

In the current system, for example, the “Can I drive through?” question is always accompanied by an image from a forward-looking robot camera. The design assumption was that the image would always unambiguously show what the robot wants to “drive through”. However, in Neelix’s experience, this was not true, i.e., it was not always evident what the image showed. Thus, to improve the “Can I drive through” question, the image should be annotated, perhaps with a graphic overlay highlighting the sensed obstacle or indicating the intended driving route.

Odo. The questions seem to be very technical, but they seem to be appropriate for when they are needed . . . You need to know something about robotics to answer . . . For novices, questions perhaps should not be about parameters, but rather more general, like ‘How safe shall I be?’

Worf. The messages seemed funny. The information about temperature would be better visually as a graph.

Odo raises an interesting point: should questions be asked, phrased, presented, etc. differently to different users? At first glance, the answer would seem to be obvious: experts should be given technical, quantitative information; novices should receive non-technical, qualitative questions.

On reflection, however, the matter is not so clear cut. The reason is that what we are really concerned about is obtaining a good answer from the human. If a question can be asked in multiple ways, our approach should be to use the phrasing most likely to produce a useful, but not necessarily precise, response. Thus, if giving an expert more “technical” details than a novice allows him to answer better, then yes, we should ask questions differently to different users. However, if asking a question differently (e.g., presenting a table of numbers instead of a chart) makes it harder for the human to interpret the information, then the answer would be no.

7.5.3 Dialogue and Human-Robot Interaction

How dialogue affects perception of robot

A significant side-effect of dialogue is that it directly affects how users perceive and treat the robot. In particular, it may cause users to personify the robot and to attribute qualities to it, which do not have a basis in reality.

For example, consider once more this comment made by Janeway:

In general, I would try to answer as soon as possible. That’s my personality. Especially coming from the robot since it probably means its urgent and would help make it work better.

This statement reflects a belief that the robot only asks questions when it desperately needs help. To an extent, this is true. Under collaborative control, the robot is supposed to have sufficient awareness to recognize when human assistance would be beneficial. However, as I previously discussed, sometimes the robot has difficulty assessing the situation (e.g., its power level) and so may tend to ask questions which do not significantly improve its operation or which may excessively consume the human resource (i.e., willingness to answer questions).

Another example is this reaction to repeated questions:

Neelix. It seems if it's not happy with what you tell it, it will ask you again and again until you give in.

In other words, Neelix felt that the robot was inflexible, unwilling to accept his answers. As in the prior case, this assessment is only partially accurate. While the robot is not explicitly designed to be obstinate, its safeguards are intended to keep the robot away from harm. Thus, in situations where its safety is at risk, such as driving fast in a cluttered area, the robot may appear to be uncompromising. The question is: to what extent does the robot's "stubbornness" (perceived or real) color, or harm, the relationship between human and robot? At this point, I do not have an answer and believe that additional study is needed to find out.

In a similar vein, dialogue may also have repercussions on the perception of authority. After receiving the "My motors are stalled..." question, Spock immediately reacted as follows:

Apparently, I don't get any priority over questions. The robot seems to decide how to use me.

Although it may seem to Spock that he never has control of the relationship, in fact, the robot retains control only when it is unable to function. Of course, "unable to function" can be subjective. In the case of motor stall, it is evident that the robot has to have priority. In other situations, human and robot may disagree over who should have authority and a mechanism such as command arbitration is needed to decide.

Incorrect assumptions

One factor that I did not consider when designing the study was how much to tell users (in advance) about how the robot works. In general, users were told only that "the robot may ask questions from time to time". As a consequence, some of them made completely incorrect assumptions, which caused significant frustration and which proved to be detrimental to their performance. This was particularly the case with Spock, who said at the end of his session:

I assumed that the next command I gave would result in different questions. That is, by giving a new command, old questions would go away since they were no longer relevant and all the new questions would then be based on the new command.

In truth, questions arise not only in response to user commands, but also from situational needs and contingencies. Because Spock assumed that pending questions would be removed as soon as he issued a new command, he was confused about why the query indicator did not stop flashing and why the robot often refused to obey him. This underscores the importance of accurately stating the ground rules for human-robot dialogue and for making sure that the user understands them. Otherwise, the human and robot are likely to end up talking at each other instead of to each other.

Asking the robot a question

Through out this discussion, I have focused on robot-initiated dialogue: questions asked by the robot and answers given by the human. This is because this type of dialogue and the concept of “human as resource for the robot” are central to collaborative control. However, since dialogue is two-way, let us briefly examine the human-initiated side.

Of course, a significant fraction of human-initiated dialogue relates to commands, tasks and directives the human wants the robot to carry out. This type of human-robot dialogue has been studied by others, particularly in terms of interpreting and sequencing commands based on the robot’s capabilities and the dialogue grammar. An excellent example of this is (Green and Severinson-Eklundh 2001).

The other part of human-initiated dialogue, which remains relatively unexplored in robotics, are questions asked by the human and answers given by the robot. Some research, such as (Breazeal (Ferrel) 1998), has focused on endowing the robot with the capacity to give emotional responses (facial expression, voice stress, etc.) to questions such as “How are you?”.

In this study, users were told that they could ask the robot three questions (Section 4.4.2). None of the subjects, however, dedicated much time to doing this. In fact, only Janeway used the facility more than once. When questioned, many of the users said that the questions did not seem very pertinent, or helpful, to the tasks at hand. Several users also pointed out that there was no need to ask the robot “What is the progress with the current command?”, since it was obvious from the displays when the robot had completed (or failed) executing a command.

The most significant comment was made by Odo, who said:

I never asked the robot questions because I expected that anything important would be asked by the robot.

In terms of collaborative control system design, therefore, it may not be necessary to have full dialogue, but only to provide support for the human to issue commands and the robot to ask questions.

7.5.4 Design Suggestions

How can dialogue be improved?

I have already described how better dialogue management could help alleviate problems associated with repeated and with multiple questions. In addition, the study yielded several other suggestions for improving human-robot dialogue.

Enable the user to specify a level of effort. Both Odo and Picard expressed interest in having the robot do more by itself, rather than asking them questions:

Odo. Some of the questions . . . it seems to know what’s best, so I would prefer to just let it decide rather than have it ask me.

Picard. I would like to have a way to ignore the question. I mean, to tell the robot to just do what it thinks is best.

One way to do this would be provide a mechanism, either through the user profile or a configuration option, to indicate how much effort the robot should exert by itself before

involving the user. In essence, this parameter would specify the minimum level of autonomy the robot is expected to exhibit.

Indicate the type and urgency of the pending question. The most serious limitation of the current “query indicator” is that it only tells the user that the robot has a question to ask. It does not tell the user what type of question is pending, what the question is about, nor how urgent it is.

Neelix. It would be useful to know when messages are urgent versus just details.

Picard. There seem to be two types of messages: suggestions/warning and critical. Should be some way to tell the difference between the two.

Worf. It would be useful to separate the really urgent messages from just the information / tuning messages.

As Picard observed, the existing set of query-to-user messages can be categorized into two categories. The first, which he refers to as “suggestions/warning”, contains questions related primarily to parameter settings (e.g., safety trigger levels). The second category, “critical”, involves questions which generally require urgent attention. Indicating which type of question is pending would allow the human to better judge when he needs to attend to it right away. Similarly, indicating the question’s priority level would also enable the human to better decide to take action.

Allow the user to override. One problem expert users had with collaborative control was that they were uncomfortable when the robot disagreed with, or did not accept, their opinion. This conflict occurred most often with the environment clutter safeguard, which is very opinionated about what it believes the proper translation speed should be. Consequently, whenever the user gave an answer that the safeguard found to be inadequate, it waited a moment, then asked the question again.

Picard. When I answer a question, I expect it to accept my response. But if it really has an issue with my answer, it should ask me in a different way or say ‘Hey, I can’t deal with this answer!’

To address this problem, it might be useful to allow the user override the robot when he deems it necessary. That is, to indicate that he has given his final answer and that further discussion is undesirable.

Provide detailed explanation when needed. A common complaint from all users was that they occasionally had difficulty understanding what the robot was asking.

Odo. Sometimes, from the way questions are phrased, it’s not clear that the robot is asking for help.

Spock. I’m not sure what this question means. Translate means motion?

To correct this failing, we would need to provide supplemental information when the user requests it. For example, pre-defined “help” text could be used to describe concepts, or to clarify terms, with which the user is unfamiliar. Another approach would be for the robot to ask questions with multiple levels of detail. This would enable the user to choose the amount of verbosity he needs in order to respond.

How can the interface be improved?

Although the study's primary focus was to provide insight into collaboration and dialogue, it also produced a number of suggestions for improving the user interface.

Waypoint editing. In both the image and map modes, users were unsatisfied with waypoint designation. More than half the users, including all the experts, expressed a desire to be able to clear or to reposition a waypoint after it had been placed. In the current PdaDriver, this is not possible: the only way to move a waypoint is to erase the entire path and start over.

Image mode. There were two specific recommendations for image mode. The first is that there should be a way to move the robot backwards without switching modes. This capability is not currently available because a design decision specified that image mode should only allow motions within the camera's field of view. Thus, reverse motions were not considered. However, for precision driving, being able to back up would clearly be useful. Assuming, of course, that the robot is capable of preventing rear collisions during such motions.

The second recommendation is that image mode should provide a tool for measuring depth. Having good depth information is essential for judging the relative position of objects in the remote world. One approach would be to provide a sense of depth by overlaying color cues on the image (Meier, Fong, Thorpe, and Baur 1999).

Direct mode. As I have discussed, all the expert users found direct mode to have serious shortcomings, primarily because it does not provide visual or spatial feedback. Picard suggested, "Maybe a combination of direct mode and map or image to see what's around would be good." One possibility would be to provide direct mode as a supplement (e.g., a pop-up or floating window) to either, or both, of the waypoint modes.

7.6 Summary

The contextual inquiry study revealed that dialogue, especially questions generated by the robot, is valuable for teleoperation. In particular, novices reported that dialogue significantly helped them understand what problems the robot encountered during task execution. Although experts were less satisfied than novices, primarily because they grew tired of answering the robot's questions, they also stated that dialogue was a useful aid which kept them involved and engaged in system operation. Finally, the study showed that human assistance is a limited resource and must be carefully managed.

Conclusion

As we have seen, collaborative control is both a novel and a useful system model for teleoperation. It is novel because it uses dialogue as a mechanism for unifying and coordinating robot action. It is useful because it enables humans and robots to benefit from one another, thus alleviating some of the common problems in teleoperation.

In this final chapter, I begin with some remarks on human-robot collaboration. This leads into a discussion of the benefits and limitations of collaborative control. I then present some areas for future work, with a particular emphasis on improving the current system. Finally, I conclude by summarizing the main results and key contributions of this thesis.

8.1 Discussion

8.1.1 Human-Robot Collaboration

When we enter into collaboration, there is no guarantee that it will be successful, that it will be effective and produce good results. Sometimes, in spite of our best intentions and dedication, collaboration breaks down or even fails. Making a collaboration succeed requires concerted effort and there is no easy or universal formula for success. The question, therefore, is: What can we do to help make collaboration work well?

In the introduction to his book, “Day of Wrath”, Larry Bond discusses the process of collaboration among writers and remarks:

When creative ideas are blended, there are always points where they don’t mesh smoothly. The conflicts can be the result of different viewpoints, different visualization of the story or characters, or even basic philosophies. Resolving these conflicts is a day-to-day part of collaboration, and the vital thing to understand is that nobody has a monopoly on good ideas and that almost any idea can be improved.

Thus, one part of making collaboration work is being able to recognize that conflicts will occur, that they are a normal component of the process, and that conflict resolution is beneficial because it encourages new ideas and approaches to be contributed.

I believe that this is an important concept whether we talking about collaboration among humans, or collaboration between humans and robots. What is essential is that when “we” (humans or robots or both) work together, that we find ways of interacting and communicating. This allows us to pool our resources and to share competency, as a way of compensating for individual limitations or deficiencies.

If we accept that resolving conflicts is significant, are there other key factors for achieving effective human-robot collaboration? In my research, I have identified three. First, roles and responsibilities must be assigned according to the capabilities of both the human and

the robot. For any given task, the work needs to be given to whomever is best equipped to handle it. Although this might seem easy to do, in practice it is not. In particular, the difficulty of many tasks, such as identifying obstacles in an unfamiliar environment, tends to be highly correlated to the situation. Since many factors can influence or affect real-world situations, we should not be surprised that the difficulty may vary with time, with location, etc. Thus, even if the robot has previously accomplished a task by itself, it may not be able to the next time without some amount of human assistance.

One way to approach this problem is for the human to focus on global strategy and allow the robot to handle the low-level details. If the robot is sufficiently aware (this is key), then it can interact with the human when it needs assistance or is unable to perform a task. With this method, we optimize collaboration based on what the robot is able to achieve on its own. By always having the robot first attempt tasks by itself, we are guaranteed to obtain maximum effort from the robot. And, by allowing the robot to engage the human as needed, we ensure that tasks will be achieved with minimum combined effort.

Given this approach, the second factor is clear: we must make it easy for the human to effect control, to assess the situation, and to assist the robot. In other words, we need to make the human-robot interface as efficient and as capable as possible. In my system, therefore, I designed PdaDriver to facilitate quick (single-touch) command generation, situational awareness, and dialogue between the human and robot.

Dialogue is particularly important because it allows the operator to review what has happened, to understand problems each robot has encountered, and to be notified when his assistance is needed. Dialogue also improves context switching: enabling the human to quickly change his attention from task to task, or if he is operating multiple robots, from robot to robot, as needed.

This takes us to the third factor, coordination. As Woods (1996) observes,

It seems paradoxical, but studies of the impact of automation reveal that design of automated systems is really the design of a new human-machine cooperative system. The design of automated systems is really the design of a team and requires provisions for the coordination between machine agents and practitioners.

In other words, humans and robots must be able to coordinate their actions so that they function as a team, not just two workers who happen to be working together. It is not sufficient to make the robot as capable, or competent, as possible. Rather, it is also necessary to make sure that the human and robot can coordinate what they do and how they do it. Otherwise, task efficiency and performance will be degraded due to interference, sequencing delays, etc.

In the case of multi-robot remote driving by a single operator, the human constrains performance because he has limited sensorimotor and cognitive resources to share. Hence, we need to reduce the level of attention the operator must dedicate to each robot. With collaborative control, a natural solution is to allow each robot to converse with the operator when it completes a task or when it encounters a problem. If multiple robots encounter problems at the same time, we arbitrate among the requests to identify the most urgent one. In this way, the human is able to focus on solving problems, rather than on monitoring the system and trying to detect anomalies.

8.1.2 Benefits of Collaborative Control

By enabling humans and robots to work as partners, collaborative control provides significant benefits to teleoperation. To begin, it allows task allocation to adapt to the situation. Unlike other forms of teleoperation, in which the division of labor is defined *a priori*, collaborative control allows human-robot interaction and autonomy to vary as needed. If the robot is capable of handling a task autonomously, it can do so. But, if it cannot, the human can provide assistance.

Collaborative control helps reduce the impact of operator limitations and variation on system performance. Because it treats the operator as a limited source of planning and information, collaborative control allows use of human perception and cognition without requiring continuous or time-critical response. If the human cannot respond because he is unavailable or busy performing other tasks, the system will still function.

The use of dialogue makes human-robot interaction adaptable. Since the robot is “aware” of to whom it is speaking, it is always able to decide if asking a question is useful or not. Because it has knowledge of the human’s expertise, accuracy, etc., the robot can consider whether to accept a response at face value or if it should weigh it against other factors. In this way, the negative effects of operator variation (e.g., novice vs. expert) are reduced and the system function can easily be adapted for different operators.

Dialogue helps the human to be effective. By focusing attention where it is most needed, dialogue helps to coordinate and direct problem solving. In particular, in situations in which the robot does not know what to do, or in which it is working poorly, a simple human answer (a single bit of information) is often all that is required to get the robot out of trouble or to perform better (i.e., than if left to its own devices). Moreover, this is true regardless of whether the human is a novice or an expert. What is important is to understand is that even a novice can help compensate for inadequate sensing or autonomy.

It also seems evident, though more research is needed to confirm it, that specific combinations of collaboration and dialogue are appropriate for multiple situations. In other words, it is possible that the interaction used for a specific user may be appropriate for all users when the system is constrained by factors such as bandwidth and delay.

For example, if communications are unconstrained, an expert may choose to use an interface that allows him to simultaneously generate multiple commands. In the same situation, the novice may prefer to use a more basic interface (e.g., control with live video feedback). However, if bandwidth is limited, both types of users may have to use a more sophisticated interface and to trust the robot to perform more steps. This is only one way the interface could vary.

In short, collaborative control is appropriate:

- for tasks in which a mix of human and robot skills are required
- when automation cannot be fixed in advanced and must be dynamically adapted to the situation
- when robot capabilities are known to be limited or inadequate

8.1.3 Limitations

Although collaborative control is clearly beneficial to teleoperation, there are limits to what it can provide. First, it can be difficult to identify which parameters are well-suited for a given task and to assign appropriate values to each robot query. If there are many tasks to perform or if task execution creates many questions, then dialogue may add considerable complexity to system design.

Second, if human-robot interaction is adaptive, then the flow of control and information through the system will vary with time and situation. This may make debugging, validation, and verification difficult because it becomes harder to precisely identify an error condition or to duplicate a failure situation. Obviously, this could be quite problematic if the system must be qualified for use in a hazardous or regulated environment (e.g., inside a nuclear facility).

Third, because humans and robots interact to achieve common goals, they are subject to team related issues. In particular, teamwork requires team members to coordinate and synchronize their activities, to exchange information and communicate effectively, and to minimize the potential for interference (Scerbo 1996). Moreover, there are numerous factors which can impact and limit group performance including resource distribution, timing, sequencing, progress monitoring, and procedure maintenance.

Finally, working in collaboration requires that each partner trust and understand the other. To do this, each collaborator needs to have an accurate model of the what the other is capable of and how he will carry out a given assignment. If the model is inaccurate or if the partner cannot be expected to perform correctly (e.g., a novice answering a safety critical question), then care must be taken. For the robot, this means that it may need to weigh human responses instead of taking them as is. For the human, this means the robot may not always behave as expected.

8.2 Future Work

8.2.1 Improvements

There are several ways in which the current collaborative control system could be improved. At present, only queries from the robot are arbitrated. A natural extension, therefore, would be to arbitrate commands from the human and from robot modules¹. It might also be useful to arbitrate among human responses, i.e., the robot would weigh responses based on the user (or users) providing them. Both command arbitration and response arbitration could be performed with a voting scheme such as (Rosenblatt 1995).

Another improvement would be to customize the way in which a question is asked, such as using icons or graphics instead of text, for different users. This would enable a wider range of users (children, handicapped, etc.) to interact with the system. Expressing questions in a more qualitative fashion would also make it easier for non-experts to use the system. For example, if the environment is cluttered, rather than asking “What is the fastest I should translate?” and expecting a numerical answer, the system could instead ask “Should I drive: slow, normally, fast?”.

1. Saphira does provide limited command arbitration for robot modules using fuzzy behaviors.

Instead of using statically defined user profiles, it might be useful to allow individual customization via a questionnaire or preference settings. A further refinement would be compute user metrics (accuracy, availability, response time, etc.) while the system is running and dynamically adapt (or learn) user profiles. Both these changes would enable finer grained control of human-robot interaction.

In terms of the user interface, a significant improvement would be to add sensor fusion (multisensor) displays (Wizse 1987). Sensor fusion displays combine information from multiple sensors or data sources to present a single, integrated view. Such displays can greatly enhance perception of remote environments by taking advantage of redundant and complementary information. For example, range data from stereo vision, ultrasonic sonar, and ladar can be fused to facilitate depth judgement and obstacle/hazard detection (Meier, Fong, Thorpe, and Baur 1999; Terrien, Fong, Thorpe, and Baur 2000).

Table 16. Additional query-to-user (questions from the robot)

Query	Purpose
The terrain seems rough (shows charts of roll/pitch vs. distance, or vertical acceleration vs. time). Shall I slow down?	configure driving (translation and rotation) speed
How dangerous is this (<i>image</i>)?	sets avoidance (stand-off) distance
Where is the object (<i>image</i>)?	designates an object for visual servoing
Is this sensor working properly (shows charts of unprocessed sensor data)?	decide whether or not to disable a sensor.
Where am I (<i>map</i>)?	assist robot localization (e.g., provide initial fix) based on recognized features/landmarks
Am I bothering you too much?	avoid generating queries too frequently
What area should I avoid (<i>image, map</i>)?	sets region to be avoided

Finally, it would be interesting to add additional autonomy to the robot to increase its self-awareness (to recognize when it needs help or when it is stuck) and capability for abstract tasks (e.g., “follow that object”). In particular, if the robot is able to ask the queries shown in Table 16, human-robot dialogue would be able to occur at higher level than in the current system. This would make the system easier to use (i.e., operators would need to have less detailed design knowledge to answer questions) and enable additional teleoperation scenarios to be studied.

8.2.2 Scaleability

As I have discussed, collaborative control provides a framework for coordinating and adapting both autonomy and human-robot interaction. Although I have shown that this framework is effective for numerous tasks, especially those characterized by limited

autonomy, scalability remains an open issue. In particular, if we wish to use collaborative control for complex applications, there are a number of challenges that must be faced.

For example, if the human must interact with a large number of robots (e.g., a fleet), it might not be possible for him to assist each one independently. Instead, it would be more practical to focus on group interaction and have the human work with squads or formations of robots. Similarly, if a robot has a large number of modules, with varied needs and autonomy, it may be difficult for the human to converse at different levels of abstraction. In this case, it might be more efficient for the human to only assist certain types of modules (e.g., high-level perception).

In more general terms, human assistance is clearly a limited resource. As the user study revealed, it is critical that we find ways of keeping the user involved and willing to answer the robot's questions. This is particularly important when the robot must operate over long distances, or for long periods of time. One approach would be to modify the existing dialogue management system to be more "user-friendly".

It seems evident, though evaluation should be performed to confirm this, that humans are more disposed to answering a question (and providing a good response) when they are interested in the topic, or have a stake in the result, or both. Thus, one modification would be to give the user some level of control over query arbitration, e.g., to allow the user to choose which "type" of questions (health, configuration, etc.) he will be asked.

Another alternative would be to develop an algorithm for choosing questions that are significant to the user. In this context, "significant" could have several meanings: having information content, matching user interest, being task/goal related. One problem, of course, is that "interest" may be difficult to quantify, because it may be strongly linked to personality, psychology, and time-varying factors (e.g. fatigue).

Finally, for long-duration or complicated operations, it will be necessary to determine how to make dialogue history more useful: easier to review, easier to understand what has occurred, etc. Some possibilities would be to provide a display that shows cause and effect, to summarize events in terms of type or frequency, and to allow the user to ask questions (e.g., "When did you ask me about X and what was the result?").

8.2.3 Additional Evaluation

Although I have obtained qualitative evidence that dialogue and user adaptation is useful, this evidence by itself does not provide a sufficient basis to guarantee that such techniques are always beneficial. Thus, I believe it is important to develop quantitative or analytical methods to evaluate collaborative control. In particular, future work should attempt to determine:

- How much does dialogue improve (or reduce) efficiency? Does dialogue create task interference?
- What if the human degrades the robot performance? What level of risk does an incorrect answer pose to the system?
- In what way is workload correlated to dialogue? Is it possible to estimate, or to assess, the amount of work incurred by answering questions?

The problem we are likely to encounter is that it may not be possible to answer these questions in general, but only for specific applications or tasks. This is not to say that addi-

tional evaluation should not be performed, but rather that we may have to devise a variety of tools and techniques to satisfy our needs.

Performance Studies

One way to quantitatively evaluate collaborative control would be to conduct controlled experiments to measure specific task performance. As I discussed in Section 7.1, there are numerous quantitative metrics that could be used to evaluate human-robot systems, including operator workload, completion time, and error rate. Of course, to obtain accurate results with these metrics, we would need to evaluate a statistically significant number of test subjects.

A useful study would be to compare collaborative control against a conventional teleoperation method, such as direct control, in terms of workload. For example, we might ask a user to perform “A to B” in a cluttered environment (containing both real and apparent hazards). With collaborative control, I would expect there to be higher secondary task performance because the robot would notify the user when his attention is needed (e.g., whenever a “drive through” decision is required). With conventional control, the user would have to dedicate more resources to monitoring the robot’s progress, and thus should have higher workload.

Some interesting variations to study would be to increase the “expense” of answering the robot’s questions by increasing the delay required to retrieve a question, to make the robot’s questions harder to answer (e.g., making it difficult to distinguish real hazards), and to discourage context switching (by increasing the cognitive state required for the secondary task).

Information Analysis

Another possible evaluation method would be to apply information theory to collaborative control. Information theory has long been used to describe and measure the performance of systems that communicate, process, or store information. For human-robot dialogue, we could define information as quantifying the amount of data conveyed by each human-robot communication such as a question from the robot, the click of a button, etc.

From this basic definition, we could then define *information* as being measured in bits: a single bit would represent a binary choice, i.e., enough information to choose (or distinguish) between two equally likely alternatives. *Information gain* would then be the number of bits acquired, received, or transferred as the result of a dialogue message. We might also define *information efficiency* as in (Raskin 2000): the minimum amount of information necessary to do a task, divided by the amount of information that has to be supplied. Thus, efficiency would vary from 0 (information is totally unnecessary or irrelevant to the task) to 1 (task requires all the given information).

With these measures, we might be able to ascertain the amount of information exchanged between human and robot during collaborative control. This would allow us to quantify the communication channel capacity needed to perform a given task. Moreover, if we were to apply the same measures to conventional teleoperation, we would then be able to directly compare the efficiency of the different approaches.

Information measures, however, might also be valuable for designing and managing human-robot dialogue. The most obvious use would be to consider the information con-

tent associated with each message. For the robot, this means that it should only ask questions when the expected information gain is high. If the robot already has a good guess about the answer, then the expected gain would be low, and it should refrain from asking the question to the human. The difficult part would be enabling the robot to quantitatively estimate the “goodness” of its answers.

For the human, the ideal would be to receive only those messages which provide high information gain, or which have high efficiency. High gain is valuable because it allows the human to maintain contextual, situational, and spatial awareness with minimal effort (i.e., few messages are needed). High efficiency is important because the human should not have to deal with unnecessary message content (e.g., questions which contain excessive details). The problem, however, is that it may not be possible to calculate the gain or the efficiency without a detailed (and continuously updated) model of the human.

Direct Comparison

Perhaps the most concrete way to evaluate collaborative control would be to perform a direct comparison. In such an experiment, the current system and another (existing) system would both be used to perform a specific task, subject to a clearly defined constraint.

For example, it might be useful to investigate the relative effectiveness of collaborative and manual control (rate-based joystick and video feedback) given a poor communication link. In particular, to confirm (or deny) the following hypothesis:

Given the same bandwidth or delay, the collaborative control system performs better because makes more efficient use of the limited link: sending sensor data (e.g., images) only when needed, directing attention to problems, etc.

To do this, we could define the following scenario and then test a statistically significant number of subjects:

Task. Remotely drive a mobile robot through multiple, outdoor test courses. Each course will have a variety of difficult terrain, including non-traversable regions and fixed hazards. Multiple sensors will have to be used to detect obstacles in foliage as well as through smoke, dust, and precipitation.

Metrics. Course completion time, number of collisions with obstacles

Test subjects. Expert users with significant experience in vehicle teleoperation.

Constraint. Limited communication bandwidth (total number of bits or fixed bits/sec) or high communication delay (fixed constant delay).

Procedure. Allow each user to gain familiarity on a training course with both systems, until they can operate both with high confidence and comfort. With collaborative control, allow the user to adjust the system parameters (e.g., query arbitration) to his preferences. With manual control, allow the user to adjust feedback displays (e.g., video size and rate within link constraints). Have each user perform the task while recording metrics.

8.2.4 Unanswered Questions

In this thesis, I have used a fairly basic dialogue model. In particular, the stereotype user model is limited, especially in terms of the number of user classes and attributes it possesses. Additionally, because the current query arbitration scheme is stateless, my system does not handle dialogue that requires multiple interaction steps or hierarchy, such as is normally found in speech-based dialogue systems. Thus, there many other aspects of dialogue that remain to be addressed. For example:

- Is multiple-state, multi-step dialogue needed for vehicle teleoperation?
- Is negotiation important? Should the robot and human be able to argue over who does what?
- How important is “shared understanding”, especially if the dialogue occurs at a high or abstract level?

Closely related to dialogue is the issue of user modeling. Thus, to implement an enhanced dialogue model, it will be necessary to answer:

- To what extent does the robot need to understand (or model) the human’s intent, objectives, and goals?
- Should the system adapt to the human’s behavior over time? Should previous human responses (or lack) be used to guide query selection?
- How can the system determine the human’s capacity and willingness to answer questions?

However, dialogue is only a part of human-robot interaction. If we wish to build capable, effective human-robot systems, we also have to understand:

- How is task complexity and degree of difficulty best characterized for humans and robots?
- How can we quantitatively measure human-robot performance? Should different performance criteria be used for humans and robots?
- How can we identify the optimum roles for humans and robots?

8.3 Summary

8.3.1 System Review

In Chapter 2, I identified the key issues involved in building a collaborative control system. Here is a review of how my system addresses each of these issues:

Dialogue. The applications examined in Chapter 6 and the user study clearly show that the human and robot can communicate with each other. The QueryManager efficiently controls when the robot “speaks”, and allows it to ask a variety of questions, in a variety of formats and about a variety of topics. The UserModeller gives the robot the ability to judge when to communicate (i.e., how often to ask questions).

Awareness. The system is capable of detecting when it needs help (to adjust operating parameters, to assist perception, etc.) With the UserModeller, the robot is able to decide whether involving the human has utility, i.e., whether the human has the required skill or expertise. This was demonstrated in the user study, because the questions the robot asked reflected the type of user involved.

Self-reliance. The safeguarded controller gives the robot the capability to maintain its own safety: avoiding hazards, monitoring its health, etc. The controller also enables the robot to automatically adjust its level of autonomy and to act independently whenever the human cannot, or is unable, to answer its questions. This was also demonstrated in the user study, especially when users failed to notice, or chose to ignore, pending questions.

Adaptiveness. The UserModeller provides a flexible, attribute-based framework for integrating users with various experience, skill, and training. This enables the system to adapt to different users, so that: questions are asked based on the operator’s capacity to answer, the level of autonomy is adjusted to the operator, and the user interface provides appropriate commands modes.

Human-Robot Interaction. The user study revealed that users attribute human characteristics (e.g., stubbornness) to the robot because it asks questions. This anthropomorphism is both helpful and problematic. It is helpful because it can encourage users to assist the robot. It is problematic when it causes users to make incorrect assumptions about how the robot operates or what it is capable of doing. Either way, it provides clear evidence that dialogue can significantly affect how humans and robots interact and relate.

User Interface Design. The PdaDriver supports not only the operator, but also the robot. In particular, it allows the robot to signal the user when it needs help, to ask questions, and to receive the human’s responses. It does this even though it was developed with user-centered design techniques. This demonstrates that an effective way to incorporate the robot’s needs into interface design is simply to express them in terms of their effect on the user (e.g., question asking mechanisms = question answering mechanisms).

Control and Data Flow. The QueryManager shows that it is possible, and practical, for the robot to ask multiple questions of the human at the same time. Moreover, even though the query arbitration method (attribute-based message filtering) developed for this thesis is fairly simplistic, my testing shows that it is effectively manages human-robot dialogue.

8.3.2 Main results

This thesis has been both an exploration and an investigation of collaborative control. By way of system design and implementation, I have shown that collaboration is an effective model for building teleoperation systems. Collaboration can serve as the basis for design, providing clear principles upon which to construct user interfaces and robot controllers. Through evaluation, I have provided evidence that dialogue helps coordinate robot action, facilitates adjustable autonomy, and enables the human to compensate for inadequate autonomy. Thus, dialogue allows us to create and operate teleoperation systems that are flexible, that have natural human-robot interaction, and that accommodate varied users.

In short, I have proved that collaborative control:

Enables joint problem solving. By treating the operator as a limited source of planning and information, collaborative control preserves the best aspects of supervisory control (use of human perception and cognition) but does not require situation-critical or time-critical response. In other words, inverting the conventional approach, so that the human works for the robot, can help improve teleoperation.

Collaboration enables the robot and the human to benefit from one another and to work towards common goals. With collaboration, the robot can ask the human for assistance, to help compensate for limitations or inadequacies in perception, planning, and cognition. Since the robot informs the human when he is needed, the human does not continually need to monitor the robot, nor look for problems. This produces better results than either the human or robot can achieve alone.

Provides coordination and control. Through the use of dialogue, collaborative control improves human-robot interaction. Since robot queries are prioritized (via arbitration), the operator's attention is always directed to where it is most needed. Thus, dialogue enables situation-based problem solving and provides an efficient framework for tasks such as multi-robot control.

Dialogue also enables control flow to be dynamic, flexible, and adaptable. In particular, by allowing the robot to consider variable human response (in terms of accuracy, speed, etc.) as well as situational needs, dialogue facilitates adjustable autonomy.

Reduces the impact of operator on system operation. By reducing the need for continuous human involvement, collaborative control helps balance the roles of operator and robot. This gives the robot more freedom in execution and allows it to better function if the operator is inattentive, unavailable or making errors.

Because we can adapt human-robot dialogue (based on the human's availability, knowledge, and expertise), collaborative control allows us to support a range of operators. Specifically, it allows us to build systems which are user adaptive: to configure system parameters and to customize the interface to appropriately match the user's capabilities and resources. Although this does not guarantee that all users will be able to perform a specific task equally well, it does provide a mechanism for improving how well each user is able to use the system.

8.3.3 Contributions

In this thesis, I have:

- developed a new system model for vehicle teleoperation
- developed a vehicle teleoperation system for dynamic, uncertain, hazardous environments that is flexible and easy to use
- developed a portable user interface that emphasizes rapid command generation, dialogue, and easy of use
- developed a safeguarded mobile robot controller that supports varying levels of autonomy and teleoperation
- developed two open-source software toolkits (FPC and VisLib)
- demonstrated that collaboration and dialogue are useful mechanisms for teleoperation
- demonstrated how dialogue allows us to build user adaptive systems that can support a range of users from novices to experts
- shown how collaborative control enables a single operator to control and manage multiple vehicles
- shown how collaborative control permits an operator to help compensate for inadequate autonomy
- surveyed the history and current state of vehicle teleoperation interfaces

8.3.4 Final Words

As we look ahead, it is clear that robots will continue to play an ever larger role in our world, working for and in cooperation with humans. Far beyond the repetitive, industrial tasks they perform today, robots will be increasingly called on to work in dynamic and unstructured environments, to function in the face of uncertainty, and to deal with situations that regularly exceed their capabilities.

In the future, robots will help improve the quality of everyday human life. Robots will assist in welfare, health care, and rehabilitation. Robots will serve as tour guides, office workers, and household staff. In the future, robots will magnify the capabilities of humans outside the factory. Robots will work in close proximity to humans, improving field science, enhancing military operations, and facilitating hazard remediation.

Central to the success of these applications will be close and effective interaction between humans and robots. Thus, although it is important to continue enhancing autonomous capabilities, we must not neglect improving the human-robot relationship. In short, to move beyond today's "robot as tool", we must continue to develop techniques that allow robots to become competent partners, with whom we communicate and collaborate to get meaningful work accomplished.

APPENDIX A

Vehicle Teleoperation

In this appendix, I provide an overview of vehicle teleoperation (its characteristics, its uses, and its history) and then present a summary of operator interfaces currently in use. A significant portion of this is based on (Fong and Thorpe 2001).

A.1 Vehicle Teleoperation

Vehicle teleoperation means simply: *operating a vehicle at a distance*. As with all teleoperation, vehicle teleoperation enables the capability to sense and to act in a remote location. Figure 42 illustrates the basic vehicle teleoperation framework. A human operator works at a control station, generating commands via input/control devices and receiving feedback from displays. The remote vehicle, which executes the operator's commands, is equipped with sensors, actuators and often some level of autonomy. The operator and the vehicle are separated by a barrier (environment, distance, time, etc.) and exchange information via a communication link.

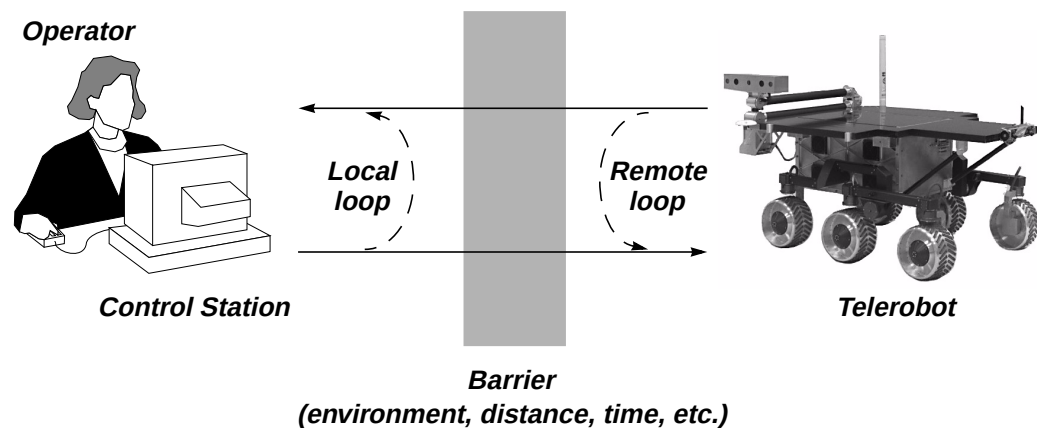


Figure 42. Vehicle teleoperation framework

Although some restrict the term *teleoperation* to denote only direct control (i.e., no autonomous functions), I consider teleoperation to encompass the broader spectrum from manual to supervisory control (Sheridan 1992). Thus, a vehicle teleoperation system may employ continuous/direct control, intermittent/supervisory control or any point between. Furthermore, the type of control can vary over time or as a function of the situation and may be shared/traded between the human operator and the remote vehicle.

Vehicle teleoperation has several characteristics which distinguish it from *remote control* (i.e., line-of-sight radio control) and other types of teleoperation (e.g., telemanipulation). First, vehicle teleoperation demands reliable navigation. Since vehicles often are deployed in unfamiliar, unknown, or unstructured environments, navigation problems may lead to system failure or loss. Second, vehicle teleoperation requires efficient motion command generation. In many cases, task performance is directly correlated to how well a vehicle moves. Finally, vehicle teleoperation calls for accurate interpretation of sensor data. Considering that most applications focus on exploration, reconnaissance or observation, it is clear that misinterpreting or misjudging the remote environment will cause difficulties.

Vehicle teleoperation is used when it is necessary to operate in a hazardous or difficult to reach environment and when the primary task is exploration, observation or reconnaissance. It is also often used to reduce mission cost and to avoid loss of life. During the past century, vehicle teleoperation has been successfully applied in the air, on the ground and underwater. Current applications include reconnaissance, surveillance, and target acquisition (RSTA), exploration, remote mining, inspection, facility security and entertainment.

There are three basic problems in vehicle teleoperation: figuring out where the vehicle is and what state it is in, determining where it should go, and moving it there (Fong, Thorpe, and Baur 2001a). In addition, there are a wide variety of task-specific problems such as observation, understanding the remote environment, and recognizing hazards. These problems can be difficult to solve, particularly if the vehicle operates in a hazardous environment with a limited communication link (low bandwidth and/or high delay). The difficulty increases when we add additional constraints such as operator variation (training, skill, etc.), multiple vehicles, and moving (or malicious) hazards.

A.2 Operator Interfaces

In order for vehicle teleoperation systems to perform well, it is critical that the operator interface be as efficient and as capable as possible. Since the 1920's, most vehicle teleoperation has been performed using rate-control joysticks and direct feedback (line-of-sight viewing or transmitted video). Even today, this is state-of-the-art for a majority of underwater ROV's and unmanned military vehicles. Recently, however, vehicle teleoperation systems have begun to employ multimodal or multisensor displays, supervisory control and novel interfaces (Web-based, PDA, etc.). These methods have proven useful for a number of exploration robots including Dante II (Fong et al. 1995), Sojourner (Cooper 1998), and Nomad (Nguyen et al. 2001).

The basic function of a vehicle teleoperation interface is to enable an operator to remotely perform tasks. Thus, all interfaces provide tools and displays to perceive the remote environment, to make decisions, and to generate commands. Most interfaces also attempt to maximize information transfer while avoiding information overload. In other words, providing high quality information is preferable to sheer quantity. Additionally, emphasis is often placed on minimizing cognitive and sensorimotor workload. A properly designed interface enables an operator to be productive longer and with less fatigue. Finally, an interface may be designed to minimize training or to be adaptive (to the operator or the situation or both). These characteristics are particularly important if the interface must support a diverse set of users having varied knowledge, skill or training.

Perhaps the most difficult aspect of vehicle teleoperation is that the operator cannot directly perceive the remote environment. Instead, he is forced to rely on the interface to provide him with sensory information. As a result, the operator often fails to understand the remote environment and makes errors. In particular, poor performance (e.g., imprecise control), poor judgement (inaccurate attitude/depth/size estimation), loss of situational awareness, and failure to detect or to avoid obstacles are common problems. A poorly designed or implemented interface exacerbates these problems. Thus, for vehicle teleoperation to be successful, we must build effective interfaces.

I must note that the importance of the operator interface does not diminish as level of autonomy increases. Even if a robot is capable of operating autonomously, it still needs to convey to the operator how and what it did during task execution. This is particularly important when the robot encounters problems or fails to complete a task. Thus, as the robot becomes more autonomous, the interface is used less for control and more for monitoring and diagnosis. Moreover, the use of autonomy creates additional interface design considerations. If an operator is only “in control” intermittently, how does the interface facilitate sharing and trading of control? If the operator’s only function is to supervise “out-of-the-loop”, how should the interface be designed to prevent the operator from becoming bored, complacent or inattentive?

A.3 History of Vehicle Teleoperation

Vehicle teleoperation first appeared in the early 1900’s, but it was not until the 1970’s that systems became prevalent. Vehicle teleoperation is not limited to a single application nor environment. Rather, a wide variety of air, ground, and underwater vehicles have been developed. Additionally, since development occurred over different time periods and in different domains, it is not surprising that vehicle teleoperation is described by a number of different terms (RPV, UAV, ROV, UGV, etc.). However, in reviewing the history of these vehicles, we discover that many common characteristics and features exist.

A.3.1 Air Vehicles

Pilotless aircraft have existed since the early twentieth century. Historically, most systems have been employed for military reconnaissance and have been small (relative to human-rated airframes). Today, *Unmanned Aerial Vehicles* (UAV)’s are used for a wide range of applications including reconnaissance, surveillance, and target acquisition (RSTA), remotely controlled strike missions, communications relay, electronic warfare/defense suppression, aerial photography, leaflet dropping, and battlefield damage assessments (BDA) (Jones 1997).

The first teleoperated air vehicles were radio-controlled target drones (also called *Remotely Piloted Vehicles*) used for anti-aircraft training. Drones such as the US Army’s RP-5 (1941) and the US Navy’s Culver PQ-14 Cadet (1943), both shown in Figure 43, primarily flew pre-programmed autopilot routes, although they could also be piloted (albeit crudely) using radio control (Bailey 1996). By 1964, the US Air Force was using the Ryan Aeronautical AQM-34 Lightning Bug drone for unmanned photo reconnaissance. In addition to film cameras and real-time video, Lightning Bugs often carried electronic intelligence and communication intelligence packages (Jones 1997).



Figure 43. Radio-controlled target drones:
left, Radioplane RP-5A (Western Museum of Flight);
right, Culver PQ-14 Cadet (USAF Museum).

During the late 1960's, NASA began developing the *Remotely Piloted Research Vehicle* (RPRV). In contrast to drones, RPRV's were full-size manned aircraft modified for remote controlled flight. In October 1971, NASA began test flights of a Piper PA-30 Twin Comanche (a light, twin-engine airplane). The PA-30 was equipped with a forward pointing video camera, video transmitter, and radio communication for telemetry and control signals. The PA-30 was flown from a ground cockpit by a pilot using joysticks, pedals and a video monitor. Electric motors connected to straps around the pilot's body provided physical cues by exerting forces during sideslips and stalls (Hallion 1984).

The impetus for modern UAV's can be attributed to Israel's use of remotely-piloted target drones for tactical reconnaissance during the Yom Kippur War (1973). Israel's success revived interest in UAV's, which had languished with the development of satellite surveillance. In particular, as conflicts began to demand increasingly regional response, military planners became attracted to systems which could be locally deployed, which cost less than manned craft, and which did not risk lives. As a result, there are now numerous UAV's which have been used in combat, including the US Navy Pioneer and the US Air Force Predator (see Figure 44).



Figure 44. Left, Predator (USAF Air Combat Command);
right, Predator control station (USAF Air Combat Command).

As with their predecessors, modern UAV's are used for tasks such as reconnaissance, surveillance, and target identification and flown from a ground cockpit. The Predator, for example, is a medium-altitude endurance UAV manufactured by General Atomics Aeronautical Systems. It carries EO, IR, and high-resolution SAR and is flown by two ground operators via a C-band (line of sight) radio or Ku/UHF satellite links. The Predator takes off and lands under manual control, but can autonomously execute flight plans once it is airborne (Canan 1999).

A.3.2 Ground Vehicles

Although powered ground vehicles pre-date aircraft, teleoperated ground vehicles have a more recent history than their aerial counterparts. Additionally, teleoperated ground vehicles have largely been limited to research organizations and have not yet had the commercial success of either air or underwater vehicles. At present, there are three primary categories of teleoperated ground vehicles: exploration rovers, *Unmanned Ground Vehicles* (UGV), and hazardous duty vehicles.

Exploration rovers are ground vehicles designed to remotely perform science tasks. These include sample collection, *in-situ* sensing (e.g., spectroscopy), and high-resolution imaging. Since the 1960's, there has been strong motivation to use exploration rovers in order to reduce the cost of exploring dangerous and extreme environments. Specifically, robots can be made smaller and lighter than humans, do not require safety-critical life support, and are expendable.

The first exploration rovers were the Soviet Lunokhods, which explored the moon in the early-1970's (Carrier 1992). Lunokhod 1 and Lunokhod 2 were both eight-wheeled, skid-steered vehicles driven via radio by a team of five earth-based operators (see Figure 45).

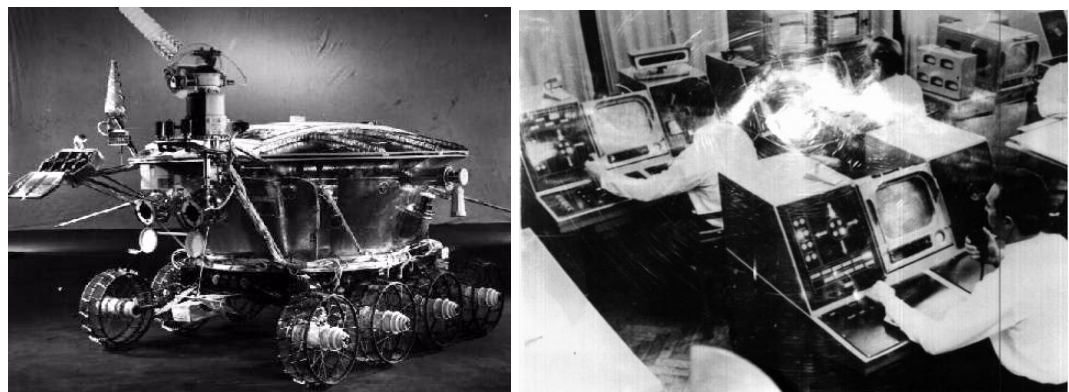


Figure 45. Left, Lunokhod 1 (Lavochkin); right, Lunokhod control station (NPO Lavochkina museum).

Lunokhod 1 explored the lunar Sea of Rains between November 1970 and October 1971, travelled over 10 km and conducted over 500 soil tests. Lunokhod 2 operated for the first four months of 1971, covered 37 km and transmitted some 80,000 television images (NSSDC 2000). Following the Lunokhods, NASA began sponsoring a series of research programs which produced numerous terrestrial vehicles including the Rocky series (Wilcox et al. 1992), Dante I/II (Bares and Wettergreen 1999) and Nomad (Wettergreen et al. 1997; Bapna et al. 2001). Then, in 1997, NASA landed the Sojourner rover on Mars (Cooper 1998).

The term *Unmanned Ground Vehicle* (UGV) encompasses a wide range of systems and technologies. In the broadest sense, a UGV is any piece of equipment which drives across the ground without carrying a human. However, I will limit my discussion to systems in which the principal focus is remote navigation and control. Within this context, a wide range of UGV applications have been developed: reconnaissance, surveillance and target acquisition (RSTA), route/area clearing, ordnance disposal, and landmine detection.

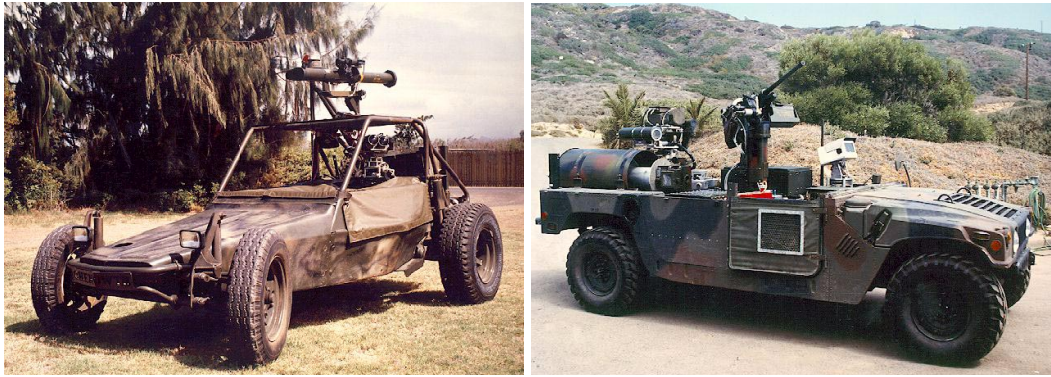


Figure 46. Unmanned Ground Vehicles:
left, Advanced Teleoperator Technology TeleOperated Dune Buggy (SPAWAR);
right, TeleOperated Vehicle (SPAWAR).

UGV development has been led primarily by the US military. In the early 1980's, the Naval Ocean Systems Center (NOSC) Hawaii developed the Advanced Teleoperator Technology (ATT) TeleOperated Dune Buggy (Figure 46-left). This vehicle was remotely driven using head-coupled stereo video, stereo audio, and replicated controls. Following the ATT vehicle, the US Marine Corps funded NOSC to develop the TeleOperated Vehicle: a HMMWV controlled via fiber-optic tether (Figure 46-right). During the same period of time, the US Army's Advanced Ground Vehicle Technology (AGVT) program contracted FMC to develop a teleoperated M113 and a General Dynamics led team to build a teleoperated Cadillac Gage scout vehicle (Siuru 1989).

By the mid 1990's, the Tactical Unmanned Ground Vehicle (TUGV) program produced several teleoperated vehicles including the Surrogate Teleoperated Vehicle (a Polaris all-terrain vehicle with a RSTA sensor suite) and the Surveillance and Reconnaissance Ground Equipment (SARGE, an upgraded version of the Sandia National Laboratory Dixie vehicle). As with earlier UGV's, the primary mechanisms for remote driving were stereo video and rate control. However, on-vehicle localization (e.g., GPS), attitude sensing, and supervisory control were also used to improve situational awareness and to enable higher speed operation (Shoemaker 1990; Gage 1996).

Hazardous duty vehicles are designed to work in conditions which pose extremely grave dangers to humans and which may cause rapid catastrophic system failure (e.g., the vehicle may be destroyed by explosion). In the early 1980's, the Remote Reconnaissance Vehicle and the Remote Core Borer explored and remediated the Three Mile Island reactor containment basement. Both of these vehicles were controlled via video camera and performed a variety of tasks including inspection, radiological mapping, sludge transport, etc. (Whittaker and Champeny, 1988).

Among the hazardous duty applications being studied today are: underground mine rescue and survey (Hainsworth 1993), bomb disposal (Graves 1997), and contaminated site assessment (Blackmon et al. 1999). Figure 47 shows the Pioneer robot, which is designed to evaluate conditions and structural damage inside the Chernobyl nuclear reactor in Ukraine (RedZone 1997).



Figure 47. *Left*, the Pioneer robot; *right*, Pioneer control console.
(Carnegie Mellon University and RedZone Robotics, Inc.)

A.3.3 Underwater Vehicles

Remotely Operated Vehicle (ROV)'s are the largest market for vehicle teleoperation. ROV's are unmanned submersibles which are generally connected to a surface vessel or platform via an electrical tether. Since their inception, ROV's have been employed for monitoring and manipulation tasks in all depths beneath the sea. Currently, ROV's are used for a wide range of applications including drilling support, site survey, debris clearance, structure cleaning, flowline and umbilical tie-in, inspection, mine countermeasures, exploration, marine archaeology, and oceanography (National Research Council 1996).

ROV's have increasingly taken over roles once performed by manned submersibles and divers. Among the reasons cited for the preferential use of ROV's are that they are less expensive to operate, do not risk lives, and can operate longer and deeper. At the same time, many marine professionals (especially deep sea oceanographers) argue that ROV's will never completely replace humans underwater. Their primary argument: electronic sensors will never have the resolution, range, and overall performance of human senses.

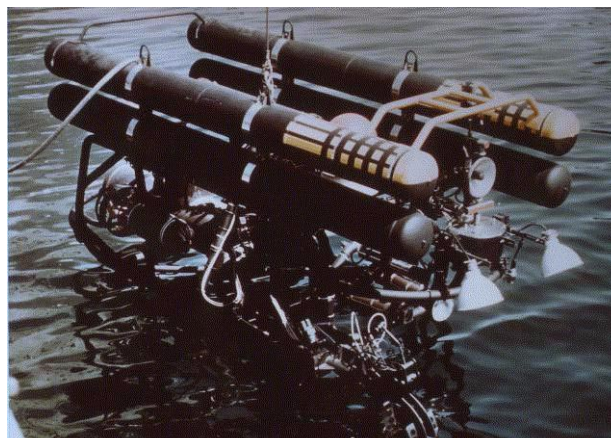


Figure 48. Cable-controlled Underwater Recovery Vehicle I (SPAWAR)

It is unclear who developed the first ROV, but one of the earliest was Poodle, a small tethered vehicle developed in 1953 by Dimitri Rebikoff and used to locate shipwrecks. The first significant ROV was the *Cable Controlled Underwater Recovery Vehicle I* (CURV I) which was built in the early 1960's at the Naval Ordnance Test Station (see Figure 48). In

1966, the US Navy used CURV I to recover an atomic bomb lost off Palomares, Spain in an aircraft accident. In the years that followed, spurred by the subsea oil boom, numerous ROV's were developed and commercialized for offshore operations (ROV 2000).

Today, many companies (Deep Ocean Engineering, International Submarine Engineering, Oceaneering, Perry Tritech, etc.) produce two basic types of ROV's: *eyeball* and *work class*. Eyeball ROV's carry video cameras (and occasionally sonar) and are used to monitor and support divers. Figure 49 shows an eyeball-class ROV. Work class ROV's are larger vehicles, constructed with an open frame for supporting propulsion and task-specific hardware (e.g., manipulators). Virtually all commercial ROV's are tethered and controlled using joysticks and video monitors. Figure 50 shows a work-class ROV designed for deep sea drilling and oceanography.

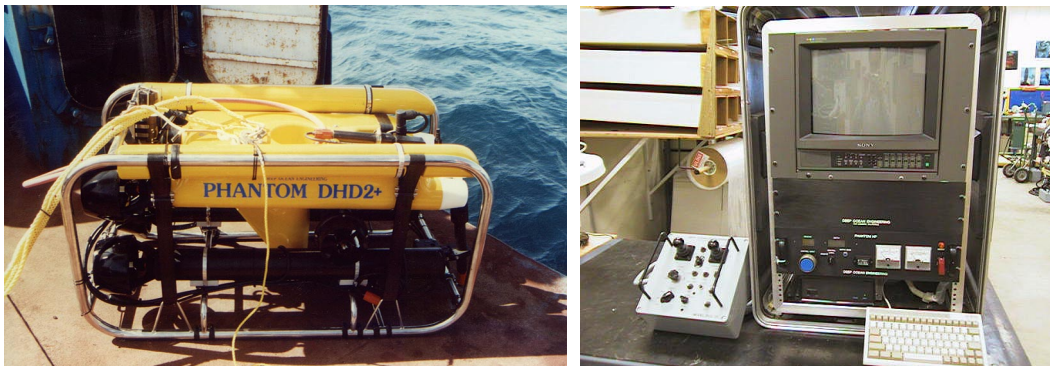


Figure 49. Left, an “eyeball” ROV; right, control console. (Deep Ocean Engineering, Inc.)

A number of ROV's have also been developed by the research community. In 1993, the Telepresence ROV (TROV) was used to conduct a benthic ecology survey beneath the sea ice in McMurdo Sound, Antarctica. The TROV was teleoperated via satellite links from the NASA Ames Research Center (Hine 1994). The Monterey Bay Aquarium Research Institute (MBARI) regularly conducts oceanographic studies using the Ventana and Tiburon work-class ROV's (Newman and Stakes 1994). Perhaps the best-known scientific ROV is the Jason/Medea dual vehicle system used by the Woods Hole Oceanographic Institute (WHOI). Jason was designed for supervisory control and provides autonomous functions such as station-keeping and track following.



Figure 50. A work-class ROV designed for drilling support and oceanographic research (International Submarine Engineering, Ltd.)

A.4 Vehicle Teleoperation Interfaces

There are four primary classes of vehicle teleoperation interfaces currently in use: *direct*, *multimodal/multisensor*, *supervisory control*, and *novel*. Direct interfaces contain all “traditional” vehicle teleoperation systems based on hand-controllers and video feedback. Multimodal/multisensor interfaces provide multiple control modes or use integrated display techniques such as augmented reality (Azuma 1997) or virtual reality (Barfield and Furness 1999). Supervisory control interfaces are designed for high-level command generation, monitoring & diagnosis. Novel interfaces use unconventional input methods (e.g., gesture) or are intended for unusual applications. Some interfaces, particularly recent hybrid systems, are difficult to categorize because they combine elements from more than one class.

A.4.1 Direct Interfaces

The most common method for performing vehicle teleoperation has traditionally been the direct interface: the operator directly operates the remote vehicle using hand-controllers (e.g., 3-axis joysticks to control vehicle rates) while monitoring video from vehicle mounted cameras on one or more displays (see Figure 51). This type of vehicle operation is often referred to as “inside-out”, because the operator feels as if he is inside the vehicle and looking out. As we have seen, air vehicles (RPVs, UAVs), ground vehicles (UGVs), and underwater vehicles (ROVs) have all been operated using direct interfaces. In fact, for many of these vehicles, the direct interface continues to be state-of-the-art.



Figure 51. Direct teleoperation interface (International Submarine Engineering, Ltd.)

Direct interfaces are appropriate when: (1) real-time human decision making or control is required, and (2) when the environment can support high-bandwidth, low-delay communications. Although direct interfaces can be (and often are) used outside these conditions, the resulting performance is sub-optimal. In particular, direct control in the presence of lengthy delay (transmission or otherwise) is tedious, fatiguing, and error prone (Sheridan 1992).

To make operation as familiar as possible (and to minimize training and testing time) some direct interfaces provide the same (i.e., spatially and functionally identical) controls as would be used if the operator was inside the vehicle. Most modern UAV's, for example, are remotely piloted using a ground cockpit which mimics the design and layout of aircraft

cockpits (Canan 1999). Other interfaces attempt to improve operator performance by providing a sense of telepresence via head-mounted, head-slaved video systems, binaural sound, and physical motion cues (Gage 1996, Ballou 2001).

It is well known that vehicle teleoperation using direct interfaces can be problematic. Sandia National Laboratory conducted a study of remotely driven ground vehicles and found that loss of situational awareness, inaccurate attitude and depth judgement, failure to detect obstacles, and poor performance are common occurrences (McGovern 1990). Other researchers have studied sensorimotor requirements for teleoperation (Kress and Almaula 1988) and the impact of video system field of view on remote driving (Glumm et al. 1992).

Moreover, since the operator is part of the vehicle's control loop and because he depends on continuous image data for perception, direct interface systems demand high-bandwidth (at least 7.5 MB/sec for a single uncompressed grayscale video stream) and low-delay (as little as 100 msec, depending on vehicle speed and environment) communication. This makes direct interfaces inappropriate or impractical for some applications.

Direct interfaces are currently used for applications requiring navigation through unknown environment (survey, inspection, reconnaissance, etc.) and vehicle positioning in support of remote tasks (loading, construction, etc.). Recent direct interfaces include an omnidirectional video system for target detection and identification (Power and Boulton 2001), a portable interface for tunnel and sewer reconnaissance (Laird 2001), remote mine survey and rescue (Hainsworth 2001), and video-based telepresence controls for submersibles (Ballou 2001) and heavy work vehicles (Halme and Suomela 2001) as shown in Figure 52.



Figure 52. Telepresence interface for heavy work vehicles (Helsinki University of Technology)

A.4.2 Multimodal and Multisensor Interfaces

When a vehicle operates in a changing, complex, or highly dynamic situation, it may be difficult for the operator to accurately perceive the remote environment or to make timely control decisions. Multimodal and multisensor interfaces can be used to cope with these problems by providing richer information feedback and command generation tools.

Multimodal interfaces provide the operator with a variety of control modes (individual actuator, coordinated motion, etc.) and displays (text, visual, haptic). Multimodal interfaces are useful for applications which demand context specific actions, i.e., when it is necessary to select control modes and displays based on situational requirements or needs.

Recent multimodal interfaces have been used to operate vehicles in changing hazardous environments and for tasks requiring multiple control modes. Fong et al. (1995) describe the interfaces employed during the Dante II / Mt. Spurr volcano exploration (see Figure 53). Lane, Carignan, and Akin (2001) discuss a virtual reality interface which provides predictive display, anomaly identification, and task planning for telerobotic servicing. Perzanowski et al. (2000) present a natural language and gesture interface for mobile robots with adjustable autonomy.

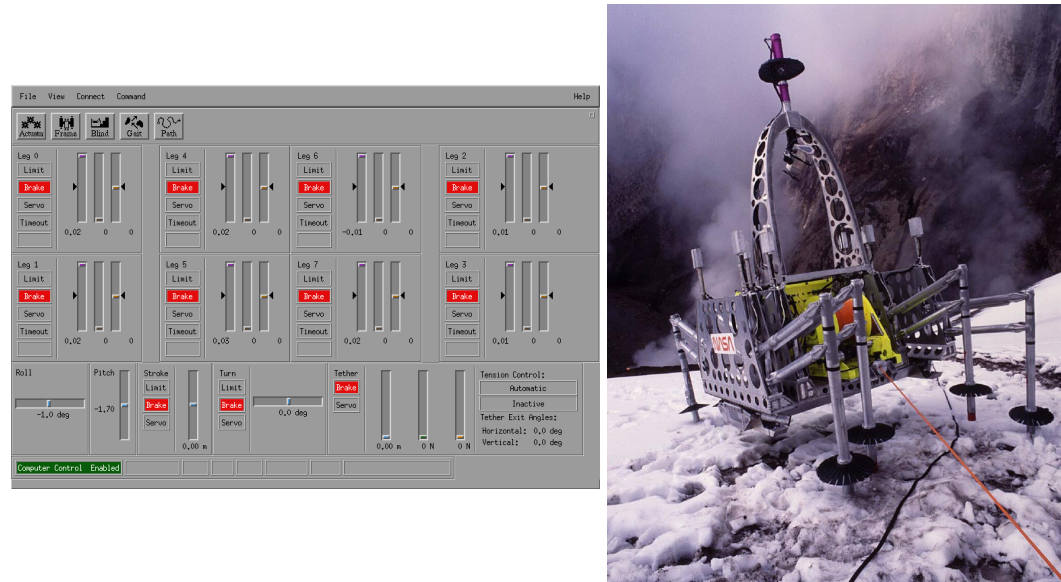


Figure 53. *Left, UI2D was used to operate Dante II; right, Dante II in the Mt. Spurr, Alaska volcano. (Carnegie Mellon University)*

Multisensor (or “sensor fusion”) displays combine information from multiple sensors or data sources to present a single integrated view. Multisensor displays are important for tasks in which the operator must rapidly process large amounts of multi-spectral or dynamically changing data. Multisensor displays have long been used in military aircraft to improve cockpit efficiency by fusing information from imaging sensors and databases (Terrien, Fong, Thorpe, and Baur 2000).

In vehicle teleoperation, multisensor displays can improve situational awareness, facilitate depth and attitude judgement, and speed decision making. Draper and Ruff (2001) discuss the use of multisensor displays for improving Predator UAV operator performance. Nguyen et al. (2001) describe a number of virtual reality based interfaces for exploration (used with the TROV, Dante II, Ames Marsokhod, Nomad and Mars Pathfinder missions) and environment modeling. Figure 54 shows one of these interfaces, VIZ, which has been used to operate a variety of planetary rovers. In contrast to direct interfaces, virtual reality provides an external perspective which allows the operator to drive/pilot the vehicle from the “outside”.

A.4.3 Supervisory Control Interfaces

The supervisory control concept appeared as part of research on how earth-based operators might teleoperate lunar vehicles. The term supervisory control is derived from the analogy between a supervisor’s interaction with subordinate staff (Sheridan 1992). To

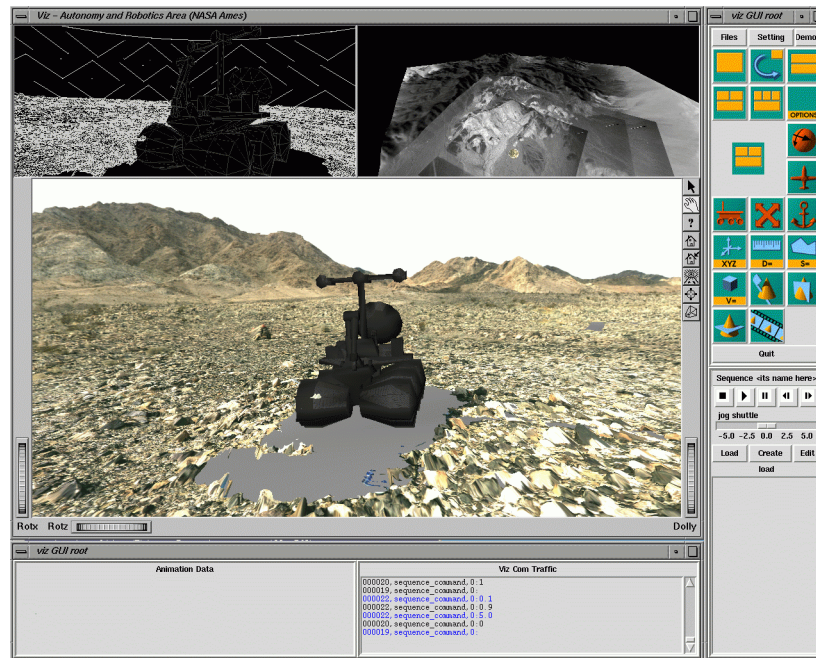


Figure 54. The VIZ virtual reality interface displays multiple data sets including stereo-vision mapped terrain and digital elevation maps (NASA Ames Research Center).

effect supervisory control, the operator divides a problem into a sequence of sub-tasks which the robot then executes on its own. To date, the majority of research in supervisory control has been focused on process control and telemanipulation.

Although supervisory control has been studied since the 1960's, there continue to be numerous open issues. First, various investigators have questioned whether removing the operator from active participation in the control loop makes detecting and diagnosing abnormalities more difficult. Additionally, it is unclear how to model a robot's *competency* (its ability to act "correctly" in a range of situations) in order to facilitate operator training and task partitioning. Finally, there are uncertainties related to the sharing/trading of control and to context switching, especially when an operator must control multiple vehicles.

Supervisory control interfaces are designed for high-level command generation, monitoring & diagnosis. These interfaces are well suited for applications which are constrained to operate with low-bandwidth communication links or in the presence of high delay. Supervisory control interfaces require that the remote vehicle have some level of autonomy. Most significantly, that it be capable of achieving goals (even limited ones) while keeping itself safe. Although there are few at present, the number of supervisory control interfaces for vehicle teleoperation will certainly increase as autonomy becomes more common in fielded systems.

In vehicle teleoperation, the operator's work is focused primarily on navigation and motion command generation. Thus, supervisory control interfaces provide tools to make these tasks easier (see Figure 55). Facilities for task planning and sequence generation (often supported with simulation and visualization) are common. Additionally, some interfaces provide methods for reviewing and analyzing results, so that the operator can monitor task performance and identify anomalies.

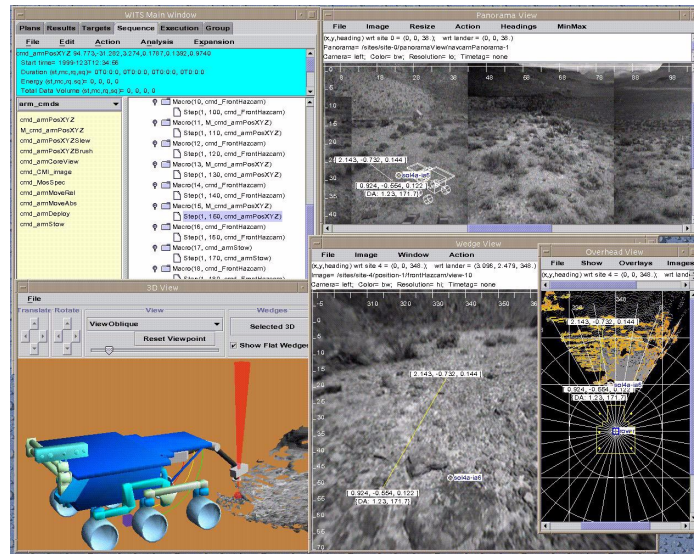


Figure 55. The Web Interface for TeleScience (WITS) provides tools for supervisory control of planetary rovers (NASA Jet Propulsion Laboratory).

There are many design challenges for supervisory control interfaces including display layout, human-robot interaction, and adaptability (to the situation, to the user, etc.). In addition, supervisory control interfaces must provide a means for the operator and the robot to exchange information at different levels of detail/abstraction. This is particularly important when the robot has problems performing a task and the operator needs to understand what happened. Also, since humans and robots perceive and make decisions differently, supervisory control interfaces must provide flexible displays. For example, the operator may decide that terrain is flat based on an image, but the robot may discover from proprioception (e.g., accelerometers) that the terrain is rough. In this situation, the interface needs to display data in such a way that the operator can detect and resolve the conflict.

Supervisory control interfaces have been developed for numerous ground vehicles. The most popular command mechanism is *waypoint driving*. In waypoint driving, the operator specifies a series of intermediate points which must be passed *en route* to a target position. A waypoint may be chosen for a variety of reasons: it may refer to a well-known or easily identified location, it may designate a safe area or place of interest, or it may provide a position fix for bounding localization error. Waypoint driving has numerous advantages over direct (rate or position) control. In particular, it requires less motor skill, uses less bandwidth, and can tolerate significant delay. Waypoint driving can be performed using either maps or images. Map-based driving, however, requires accurate localization and maps. Thus, for unexplored environments, most remote-driving systems are image-based.

Wilcox, Cooper, and Sato (1986) describe “Computer Aided Remote Driving (CARD)”, a stereo image based method for interplanetary teleoperation of planetary rovers. Rahim (1993) discusses the “Feedback Limited Control System (FELICS)”, a video system for real-time remote driving. Kay (1997) describes STRIPE, which uses still images and continuous groundplane reprojection for low-bandwidth driving over uneven terrain. Cooper (1998) and Matijevic (1998) describes how earth-based operators used the *Rover Control Workstation* (shown in Figure 56) to control the Sojourner rover via command sequences uploaded once per day to Mars.

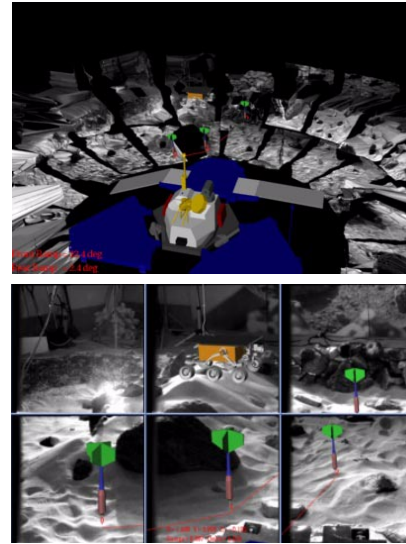
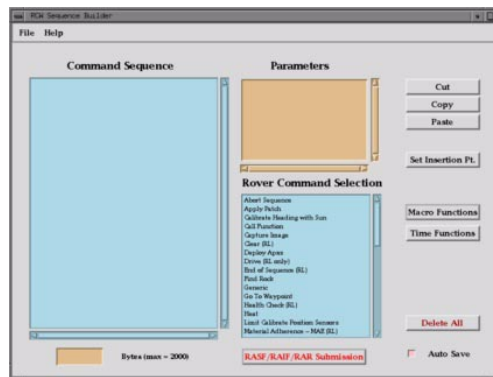


Figure 56. The Rover Control Workstation (RCW) was used to operate the Sojourner rover on Mars (NASA Jet Propulsion Laboratory).

A.4.4 Novel Interfaces

The last category of vehicle teleoperation interfaces are the novel interfaces. Of course, the term “novel” is relative: many present-day interfaces (e.g., virtual reality systems) were once called “novel”, but are now widely used and considered commonplace. Thus, it is possible, or perhaps even likely, that the interfaces described below will cease being novel at some point in the future.

Some interfaces are novel because they use unconventional input methods. Amai (2001) describes a hands-free remote driving interface based on brainwave (electroencephalogram) and muscle movement (electromyogram) monitoring. In Amai’s system, brain beta-wave amplitude controls a robot’s speed and gaze direction (determined from eye muscle signals) sets the robot’s heading. Fong et al. (2000) describe the *HapticDriver* (a haptic interface which enables an operator to “drive-by-feel”) and the *GestureDriver* (a vision-based system which maps hand movements to motion commands) as shown in Figure 57.



Figure 57. GestureDriver (Swiss Federal Institute of Technology Lausanne)

Personal interfaces are also currently considered to be novel. Since the first telerobot appeared on the World Wide Web in 1994, numerous Web-based vehicle teleoperation interfaces have been developed (Simmons 1996; Michel, Saucy and Mondada 1997; Siegwart and Saucy 1998; Grange, Fong and Baur 2000). Figure 58 shows two Web-based interfaces for rate controlled remote driving. A Web interface is attractive because it can be accessed world-wide, is highly cost-effective, and requires little (or no) operator training. Web-based teleoperation, however, is susceptible to problems that more traditional interfaces do not have to deal with. For example, data transmission through the Internet is often irregular and unreliable. Thus, a Web-based interface must be designed to handle potentially unbounded delay or loss of data.

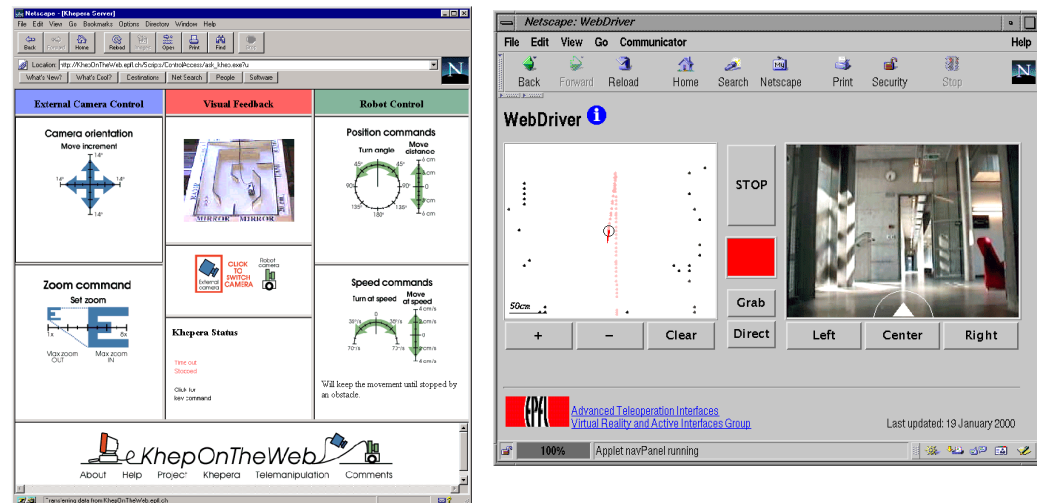


Figure 58. Web interfaces (Swiss Federal Institute of Technology Lausanne)

In some remote driving applications, installing operator stations with multiple displays and bulky control devices is infeasible (or even impossible) due to monetary, technical or environmental constraints. An alternative is to use a Personal Digital Assistant (PDA). PDA's are attractive interface devices because they are lightweight, portable, and feature touch-sensitive displays. PDA-based interfaces for remote driving are described by Fong et al. (2000) and Perzanowski et al. (2000).

Lastly, novel interfaces are not just characterized by unconventional input methods or displays. Interfaces are also novel if they are used in unusual ways. One such interface is described in (Paulos and Canny 2001). In this system, a vehicle teleoperation interface enables operators to “project their presence into a real remote space”. In other words, the teleoperated vehicle is a mobile, physical proxy or “real-world avatar” for the operator.

A.5 Summary

Vehicle teleoperation has become increasingly important for a wide range of applications in the air, on the ground, and underwater. Regardless the task, however, system performance is driven by the quality of the operator interface. Specifically, vehicle teleoperation efficiency and effectiveness is closely tied to the extent that an interface is appropriate for the operator, the remote vehicle, and the task to be performed.

Although there are many types of vehicle teleoperation interfaces, all interfaces provide tools and displays to perceive the remote environment, to make decisions, and to generate commands. Rate control interfaces continue to be widely used, principally in domains with real-time requirements and which can support high-bandwidth, low-delay communications. Multimodal/multisensor displays are increasingly in use, especially for controlling vehicles operating in highly dynamic or complex environments. Supervisory control interfaces, though currently few in number, will become more common as the use of autonomy increases, particularly in low-bandwidth, high-delay domains.

APPENDIX B

Toolkits

Two software toolkits, Fourth Planet Communicator (FPC) and VisLib, were developed during the course of this thesis. I made extensive use of both during my study of collaborative control.

I originally designed FPC in 1997 as a general-purpose, multi-platform interprocess communication toolkit for Fourth Planet, Inc. FPC came to be used in all of Fourth Planet's products, including applications as diverse as 3D visualization and computer network monitoring. FPC was released as Open Source on 30 July 2000.

VisLib was a joint effort by Sébastien Grange and myself to create an fast, flexible color vision library. We completed the initial release of VisLib in June 1999 and transferred it to ActivMedia Robotics, LLC. in exchange for a mobile robot. At the end of 1999, ActivMedia Robotics released VisLib as Open Source. In addition to this thesis, VisLib has been used as part of the "Human Oriented Tracking" system (Grange 2000).

B.1 FPC

The following text is extracted from Fourth Planet 1998.

B.1.1 Overview

Fourth Planet Communicator (FPC) is a system for distributing data across computer networks. FPC uses a *publish and subscribe* framework which provides efficient, dynamically reconfigurable, and scalable data distribution. With FPC, it is easy to distribute information and to communicate between multiple processes, whether they are running on the same computer or not.

FPC provides distributed, dynamic network communications using a *publish and subscribe* framework. In this model, a process which sends data is called a *producer* and a process which receives data is called a *consumer*. Producers *publish* data: they announce that they are generating some type of information. Consumers *subscribe* to data: they request that they receive certain types of information when it becomes available. Thus, when a producer produces data, it is delivered only to those consumers who want to consume it.

To avoid some of the problems associated with publish and subscribe (network congestion, complex consumer coding, etc.), FPC uses a centralized data cache, the FPC server, for managing data delivery to consumers. The FPC server operates like an efficient assistant: receiving and storing data whenever it becomes available, discarding data when it becomes outdated, and delivering data only when it is requested.

Here is a summary of the features provided by FPC:

- supports multiple producers and multiple consumers
- supports multiple “virtual networks” for isolating sets of producers and consumers from other producers and consumers
- simple, easy-to-use programming interface
- reliable and efficient messaging via centralized cache
- supports multiple operating systems: Linux, Microsoft Windows, Silicon Graphics IRIX, Sun Solaris
- multiple-language support: C, Java, TCL, Perl

B.1.2 The FPC Communications Model

Publish and subscribe

FPC provides distributed, dynamic network communications using a *publish and subscribe* framework. In this model, a process which produces data (sends messages) is called a *producer* and a process which consumes data (receives messages) is called a *consumer*. Producers *publish* data: they announce that they are generating (sending) some type of information. Consumers *subscribe* to data: they announce that they want to consume (receive) certain types of information when it becomes available. Then, when a producer produces data, it is delivered only to those consumers who want to consume it.

The *publish and subscribe* framework is extremely powerful. It enables flexible applications: there is no need for explicitly or rigidly specifying communication routing (e.g., process A on machine B is sending data of type C to process D on machine E). It allows dynamically expandable and scalable applications: any number of consumers and producers can exist at any time. Most importantly, it provides an intuitive, easy-to-understand model for designing interprocess communications.

At the same time, however, *publish and subscribe* can cause problems. It can very easily result in unnecessary network congestion, especially if data is sent¹ to every subscribed consumer whenever a producer produces new data. *Publish and subscribe* may also require complex programming, particularly for data consumers. For example, many consumers may subscribe to the same publication but may want to receive the data differently (e.g., some may only want the most recent data, others all the data).

The FPC server

To address these problems, FPC uses a centralized data cache (the FPC server) for managing data delivery. Whenever a producer produces new data, the FPC server stores the data in a *first-in first-out* queue for each consumer which has subscribed to this data. Consumers can query the FPC server for the contents of their queue at any time (i.e., whenever they wish to receive new instances of subscribed data).

1. “distributed”, “delivered”, “pushed”, etc.

Additionally, consumers can control the following by sending a request to the FPC server:

- the queue depth (the *cache limit*)
- the amount of data sent by the FPC server in response to a query (the *read limit*)

The *cache limit* parameter controls the amount of published data that the FPC server stores for a consumer. By setting a small cache limit, consumers can guarantee that they always receive recent data without having to process a potentially large backlog of “old” data. This is useful for applications in which consumers have to work with producers that produce data at high rates. Conversely, by using a large cache limit, consumers can guarantee that they do not miss too much (or any data) that is produced.

The *read limit* parameter provides consumers control over how they receive published data. By setting a small *read limit*, consumers can guarantee that they will only have to process a limited amount of data whenever they query the FPC server. This is important for time-critical applications (e.g., real-time graphical interfaces) which cannot afford to be overloaded with too much data at once. Conversely, by using a large (or unconstrained¹) *read limit*, consumers can guarantee that they always process a significant amount (or all) of the published data stored by the FPC server.

An efficient postal system

FPC allows producers and consumers to communicate as if they were using an efficient postal system. Producers send mail (publish data), consumers receive it (subscribe and read). Producers can send mail to the post office (the FPC server) at any time. Consumers can get their mail at any time by “going to the post office” (querying the FPC server). Any number of producers can send the same mail (publish the same type of data) to all consumers who have subscribed to it.

The post office (FPC server) stores mail from producers into post office boxes (FIFO queues) of subscribed consumers. The post office boxes can be different sizes (*cache limit*) to suit the needs of each consumer. If a box becomes “full” (i.e., the *cache limit* is reached), the FPC server makes room for new mail by throwing out old mail. When consumers “check” their post office boxes, they can take as little or as much out of it as they desire (*read limit*).

Thus, with FPC producers can send mail (produce data) at any time to any number of consumers, but each consumer controls how and when they receive it.

1. unconstrained means “all messages currently queued by the server”

B.1.3 Clients

An application built with FPC can be a producer, a consumer, or both. Regardless of type, these applications are all *clients* because they all communicate via the FPC server. In fact, since the FPC server delivers data from producers to consumers, an FPC server must be in operation before any client-to-client communication can occur.

Channels

Unlike other communication systems, FPC does not multiplex different types of data on a single client-server communications link. Instead, FPC clients use one or more *channels* to communicate with the FPC server¹. Producers use a different channel for each type of data they are producing: each channel carries one type of data from producer to server. Consumers generally also use different channels for each type of data they are consuming: each channel carries one type of data from server to consumer².

Channels provide tremendous flexibility for application design. For example, if we are building an application which distributes both time-critical (though low-rate) “control” information as well as non-timely (though high-rate) “data”, we can create a *control channel* and a separate *data channel* in all the producers and consumers. Then when the application is running, consumers will be able to process the two channels in parallel. In other words, consumers will be able to receive urgent messages from the *control channel* without having to first wade through a stream of routine data on the *data channel*.

Producers

An FPC producer does the following for each type of data it produces:

- connects (opens a channel) to the FPC server
- specifies the publication (what type of data will be sent on the channel)
- sends (produces) data on the channel

Connections to the FPC server are performed using the FPC library function `fpcConnect`. The producer and FPC server may both be run on the same computer or on different computers. The only requirement is that they share a TCP/IP network connection.

A publication is specified in FPC as a simple text string using the function `fpcPublish`. So, “data”, “Temperature : value”, and “Captain Janeway” are all valid FPC publications.

More than one producer can publish the same publication. However, since all FPC clients are considered equal peers, there is no arbitration among multiple producers of the same data. Consequently, if a consumer subscribes to a publication produced by multiple producers, it will receive data from all producers in no specific order.

FPC producers use the `fpcSend` function to send (produce) data to the FPC server. Once connected, producers operate completely independently and asynchronously of the server. Thus, producers can produce data at any time and at any rate.

1. FPC clients are not restricted to a single server, but may use multiple channels to communicate with multiple servers.
2. Consumers can use a single channel to communicate with the FPC server, but doing so will mix different types of data together (i.e., the data will be multiplexed). Since FPC does not provide a mechanism for distinguishing data, the consumer itself must be able to identify different data arriving on the same channel.

Consumers

A FPC consumer does the following for each type of data it consumes:

- connects (opens a channel) to the FPC server
- specifies the subscription (what type of data will be received on the channel)
- specifies the channel's *handler function* (to process new data)
- optionally set the channel's *cache limit* and *read limit* parameters
- reads (consumes) data on the channel

As with producers, FPC consumers connect to the FPC server using the function `fpcConnect`. The consumer and FPC server may both be run on the same computer or on different computers. The only requirement is that they share a TCP/IP network connection.

A subscription is specified in FPC as a simple text *pattern* with the function `fpcSubscribe`. Patterns are matched according to the “wildcard” rules used by UNIX command shells (i.e., globbing)¹ to match filenames. Thus, “data”, “d*a”, and “[d]a?a” are all valid subscription patterns which will match a publication called “data”. We can use the “*” pattern to receive all publications (i.e., “*” matches everything, including the NULL publication).

A consumer may have multiple subscriptions on the same channel. However, since FPC does not provide a mechanism for distinguishing data, the consumer, specifically the handler function, must itself be able to identify different data (subscriptions) arriving on the same channel.

If a consumer subscribes to a publication produced by multiple producers, it will receive data from all producers in no specific order². Thus, if message order or message priority is important to the application, we will have to design the message data appropriately (e.g., encode a priority level in the data and have the handler function arbitrate among messages based on this level).

FPC consumers use the `fpcRead` function to poll the FPC server whenever they want to receive data. The FPC server then sends any cached messages (subject to each consumer's *read limit* parameter), which are processed by the *handler function*. The handler will be called one time for each data message that is sent from the FPC server to the consumer. The `fpcRead` function reads (consumes) data synchronously with the server. To reading asynchronously, the `fpcReadBackground` function can be used.

B.1.4 Message Data

All communication in FPC uses a single data type: a NULL terminated string (also known as a *C string*). Thus, producers must produce data as strings and consumers must consume data as strings. Because the strings are NULL terminated, it is not possible to send or receive a NULL (ASCII value 0) character. However, all other ASCII characters (values 1-255) can be used in FPC. Additionally, FPC restricts the string length to a maximum of `FPC_DATALEN_MAX`³ bytes (including the NULL terminator).

1. The pattern matching function used by FPC is BSD's `fnmatch`.

2. Actually, the data will be arranged in the order it arrives at the FPC server.

3. Currently 32,768 bytes.

B.1.5 Programming with FPC

Communication with FPC works as follows:

- The FPC server delivers messages between FPC clients (producers and consumers) using channels.
- Each FPC client may have multiple channels connected to multiple FPC servers.
- Producers send messages to the FPC server for deferred (cached) delivery to all subscribing consumers.
- The FPC server caches all messages in a FIFO for each consumer (subject to each consumer's *cache limit* parameter).
- Consumers poll the FPC server for cached messages whenever they want to receive data. The FPC server then sends any cached messages (subject to each consumer's *read limit* parameter), which are processed by a *handler function*. There is one *handler function* per channel.

A simple FPC producer

A FPC producer does the following for each type of data it produces:

1. connect (open a channel) to the FPC server
2. specify the channel's publication (i.e., what type of data will be sent on the channel)
3. send (produce) data on the channel

To create a simple FPC producer, we just need to do these steps for a single channel. Let's look at each step in detail.

First, we need to connect the producer to the FPC server. We do this using `fpcConnect` which creates a communication channel. For example, we can connect to a FPC server running on host machine `foobar` and port 5000 as follows:

```
int channel;

/* connect to the comm server, failing after 5 attempts */
if (0 > fpcConnect (foobar, 5000, NULL, 5, &channel)) {
    fprintf (stderr, "error: could not connect to foobar:5000\n");
    fpcPerror ("producer");
    return (-1); /* failure */
}
```

Note that we allow `fpcConnect` to try the connection up to 5 times before giving up. This is useful if the network is unreliable (i.e., the connection between the producer and the server) or if the server is not yet running.

Next, we need to specify what type of data will be published on the channel. We can use the `fpcPublish` function to specify that data of type “blah” will be produced:

```
/* publish messages */
if (0 > fpcPublish (channel, "blah")) {
    fprintf (stderr, "error: could not publish <blah>\n");
    return (-1); /* failure */
}
```

Now we are ready to start producing “blah” data. Whenever we want to send data to the server, we use the `fpcSend` function. For example, to send the data “the cow jumped over the moon”, we would do this:

```
if (0 > fpcSend (channel, "the cow jumped over the moon")) {
    fpcPerror ("producer");
    return (-1); /* failure */
}
```

A simple FPC consumer

A FPC consumer does the following for each type of data it consumes:

1. connect (open a channel) to the FPC server
2. specify the channel’s subscription (i.e., what type of data will be received on the channel)
3. specify the channel’s *handler function* (to process new data)
4. optionally set the channel’s *cache limit* and *read limit* parameters
5. read (consume) data on the channel

So, to create a simple FPC consumer, we just need to do these steps for a single channel. As we did for the simple FPC producer, let’s look at each of these steps.

First, we need to connect the consumer to the FPC server. As for the simple FPC producer, we can use `fpcConnect` to connect to a FPC server running on host machine `foobar` and port 5000 as follows:

```
int channel;

/* connect to the comm server, failing after 5 attempts */
if (0 > fpcConnect (foobar, 5000, NULL, 5, &channel)) {
    fprintf (stderr, "error: could not connect to foobar:5000\n");
    fpcPerror ("producer");
    return (-1); /* failure */
}
```

Next, we need to specify what type of data will be consumed on the channel. We can use the `fpcSubscribe` function to specify that data of type “blah” will be consumed:

```
/* subscribe to messages of type "blah" */
if (0 > fpcSubscribe (channel, "blah")) {
    fprintf (stderr, "error: could not subscribe to <blah>\n");
    return (-1); /* failure */
}
```

Now we need to install a handler function for processing the data messages the consumer receives on the channel. Here's the handler function:

```
int messageHandler (char *pData, void* pPrivateHandlerData)
{
    if (!pData) {
        fprintf (stderr, "messageHandler: error: NULL data\n");
        return (-1); /* failure */
    }

    printf ("messageHandler: received: %s\n", pData);
    return (0); /* success */
}
```

FPC does not guarantee that data referenced by this pointer will persist once the handler returns. Thus, if the consumer needs to use this data at a later time, the handler should save a copy.

Here's how we install the handler using `fpcHandlerRegister`:

```
/* register message handler (no private data) */
if (0 > fpcHandlerRegister(numChannel, messageHandler, NULL)) {
    fprintf (stderr, "error: could not register handler\n");
    return (-1); /* failure */
}
```

Note that if we wanted to pass a private parameter to the handler (e.g., pointer to a storage area, a function, etc.) we would pass it as the last parameter to `fpcHandlerRegister` (instead of `NULL`).

If needed, we could set the channel's cache limit and read limit parameters using the functions `fpcLimit` and `fpcReadLimit` respectively. But, if we do not call these functions, the FPC server will use default parameters which should work well for most applications.

Now we are ready to start consuming data. Whenever we want to consume any "blah" data that has been sent to the server, we poll the FPC server:

```
if (0 > fpcRead (channel)) {
    fpcPerror ("consumer");
    return (-1); /* failure */
}
```

If the FPC server has cached any "blah" messages for us to consume, it will send them. Each message that is received will be processed by our handler function.

B.1.6 Performance

Benchmarking the performance of network communication applications is an extremely difficult task. To some extent, performance is affected by application characteristics (application structure, code tuning, etc.). These factors, however, are largely overshadowed by network characteristics (interface efficiency, bandwidth, congestion, physical structure, etc.). For example, transport layer (UDP, TCP) throughput benchmarks can be strongly biased by the underlying network hardware (e.g., the presence of bridging hubs).

Thus, it is often not possible to produce consistent and accurate benchmarks. More troubling is that benchmarks tend to represent ideal performance and do not accurately reflect actual application performance. The bottom line is that, in reality, the only criteria which *really* matters is whether an application performs sufficiently well to accomplish its tasks.

However, to give some insight into FPC, here are results from a simple consumer and producer benchmark. These figures are not intended as a guarantee of performance, but rather to provide some guidelines for designing FPC applications.

Local performance

In this test, the consumer, producer, and the FPC server were all run simultaneously on the same computer.

Table 17. FPC local throughput (messages per second)

Computer System	message size (bytes)			
	4	64	256	1024
IBM PC compatible RedHat Linux 4.2 (kernel 2.0.3) Pentium-166, 32 MB RAM	400	400	400	900
Silicon Graphics O2 IRIX 6.3 R5000-180, 256 MB RAM	1800	1500	1200	1100
Silicon Graphics Octane IRIX 6.4 2xR10000-195, 256 MB RAM	4700	4700	3900	3000
Sun UltraSparc 1 Solaris 2.5.1 Ultra1-140, 128 MB RAM	250	500	1000	600

Remote performance

In this test, the consumer, producer, and the FPC server were run on three identical Silicon Graphics O2 workstations, connected on the same 10-Mbit ethernet segment (IEEE 802.3). However, the location of each program was varied.

Table 18. FPC remote throughput (messages per second)

Configuration (program machine)			message size (bytes)			
FPC Server	bench Produce	bench Consume	4	64	256	1024
A	A	B	1900	1700	1300	600
A	B	A	3100	2600	1900	700
A	B	C	3100	2000	800	400

B.1.7 FPC Open Source License

The Fourth Planet Communication (FPC) Library is a simple TCP/IP based messaging system. Clients wishing to communicate connect to a communications server and exchange simple strings. FPC uses the Simple Communications Library (SCL) for TCP/IP transport.

FPC should work with little or no modification under any POSIX compliant UNIX system. Support is provided for Win32 (note: compilation requires installation of Cygnus' CYGWIN and Microsoft Visual C++). FPC has been tested on the following systems:

HP/UX 10.20 IRIX 5.3, 6.2, 6.3 RedHat Linux 5.2, 6.0, 6.1 Solaris 2.5.1, Windows 95, Windows 98 Windows NT 3.5, 4.0

FPC was developed by Fourth Planet, Inc. and was released as Open Source on 30 July 2000 under terms of the following notice.

Copyright (c) 1997-2000. Fourth Planet, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

B.2 Vislib

B.2.1 Description

VisLib is a fast and flexible library for color machine vision. It was designed specifically for object tracking using a single color camera. Although it is possible to use the library for monochrome (black & white) vision, many of the library functions will function sub-optimally. VisLib contains functions for image acquisition (capture) and display, image format conversion, 2D image processing, and template-based object tracking.

The image acquisition and display functions are hardware specific. VisLib is optimized for RedHat Linux, framegrabbers supported by the Linux Bt848 driver and any X11 server which provides a TrueColor visual (with 8, 16 or 24 bit depths). VisLib provides a variety of functions for 2D drawing and for reading/writing images.

VisLib supports numerous image formats including bitmap, RGB (24-bit), normalized *rg*, gray-scale, HLS (double hexcone), HSI (Gonzalez and Woods), and *l1-l2-l3* (Gevers and Smeulders). VisLib provides functions to convert between the various formats.

The 2D image processing functions are generic convolution (2D kernel), filtering and morphological tools (dilate and erode). These tools have been extensively optimized for speed and extensively tested.

The object tracking functions are designed for single-camera, color-based tracking. VisLib allows a variety of parameters (features) to be used for object definition including normalized *rg* blob, HLS/HSI blob, and shape (edge map). Objects are initially defined by pixel region growing with simple boundary conditions (i.e., until a maximum change in the pixel value). Other routines deal with object shape update, color update, and object redefinition. The localization of the object in each frame is done using correlation (template matching) of the binary shape of the object within a tracking window that has been filtered with the object's normalized colors.

B.2.2 Requirements

VisLib was originally released as Open Source by ActivMedia Robotics, LLC in 1999.

The current version (v1.12) of VisLib may be obtained from the Virtual Reality and Active Interfaces (VRAI) Group, Institut de Systèmes Robotiques (DMT-ISR), Ecole Polytechnique Fédérale de Lausanne (EPFL).

VisLib should work with any version of Linux with kernel 2.0.35 (or later). In addition, VisLib requires the following:

- SVR4 IPC: System V shared memory interprocess communication extensions
- X11 shared memory extension
- X11 TrueColor visual
- Bt848 driver (“bt848-1.0” or later, with the Matrox Meteor API). Available from ActivMedia, Inc.

Although VisLib will operate with an 8-bit TrueColor visual, image display will be poor. Thus, it is highly recommend that either a 16 bit or 24 bit visual be used. Most XFree86 servers support 16 or 24 bits (though perhaps at lower screen resolution depending on the amount of video RAM in your system).

B.2.3 Example: Motion Detection

Let us look at how VisLib can be used to detect motion in a scene. An easy method for doing this is to continuously capture camera images and compute differences between consecutive frames. We will assume that pixel variations between frames indicate motion, not changes due to varying scene illumination.

First, we initialize VisLib and some work variables;

```
vlInit(); /* initialize VisLib */
vlImage *binImage = VL_IMAGE_CREATE();
vlImage *displayImage = VL_IMAGE_CREATE();
vlImage *lastImage = VL_IMAGE_CREATE();
vlImage *srcImage = VL_IMAGE_CREATE();
vlWindow *window = VL_WINDOW_CREATE();
int threshold = 75; /* 0-255 */
```

Next, we grab the initial reference frame:

```
if (0 > vlGrabImage (lastImage)) {
    return (-1); /* failure */
}
```

Now, we start the continuous capture. On each pass, the first thing we do is to capture a new image:

```
if (0 > vlGrabImage (srcImage)) { /* capture a new image */
    return (-1); /* failure */
}
```

Then, we subtract this image from the reference image and threshold (binarize) the result. The window variable specifies a rectangular portion of the image. By default, it covers the full image. If we wished, we could use it to limit our search to a smaller region:

```
vlDifference (lastImage, srcImage, window, displayImage);
vlImageCopy (srcImage, lastImage);
vlRgb2Binary (displayImage, threshold, window, binImage);
```

At this point, any set (i.e., non-zero) pixels in the binary image may indicate motion. We can draw bounding-boxes around large, connected regions using the library function `vlDrawBinaryRects()`:

```
if (vlDrawBinaryRects (srcImage, binImage,
                      widthMin, heightMin, VL_COLOR_BLUE) > 0) {
    printf ("motion detected\n");
}
```

This function returns the number of boxes drawn. So, if the result is non-zero, then we have found motion in the scene.

B.2.4 Example: Object Detection

VisLib can also be used for color-based object detection. A simple, but surprisingly effective approach, is to segment an image via pixel color filtering and then searching for large connected regions.

First, we initialize VisLib and some work variables:

```
vlInit(); /* initialize VisLib */
vlImage *binImage = VL_IMAGE_CREATE();
vlImage *srcImage = VL_IMAGE_CREATE();
vlImage *tempImage = VL_IMAGE_CREATE();
vlObject *object = VL_OBJECT_CREATE();
vlWindow *window = VL_WINDOW_CREATE();
int threshold = 75; /* 0-255 */
```

Next, we load a VisLib object from a file (“blah.obj”). In VisLib, objects may be defined in a variety of ways including shape (binary mask), position, and color range.

```
if (0 > vlObjectLoad (object, objectFilename)) {
    fprintf (stderr, "error: object load failed\n");
    return (-1); /* failure */
}
```

Now we are ready to look for the object. We begin by capturing an image:

```
if (0 > vlGrabImage (srcImage)) {
    return (-1); /* failure */
}
```

To reduce the effect of varying illumination, shadows, and scene geometry, we transform image pixels into a normalized color space before pixel filtering. VisLib supports several normalized color spaces, such as normalized *rg*. As in the motion detector, the window variable specifies the full image by default.

```
vlRgb2Nrg (srcImage, window, tempImage);
```

Now we filter the image based on the object’s color. The function `vlNrgFilter` will select image pixels that are within the color range of the object. The result will be output as a binary image:

```
vlNrgFilter (tempImage, object, window, binaryImage);
```

We can use morphological filters (dilate and erode) to reduce image noise and to merge scattered pixels.

```
vlBinaryDilate (binaryImage, 3, window, tempImage);
vlBinaryErode (tempImage, 6, window, binaryImage);
```

At this point, any set (i.e., non-zero) pixels in the binary image are potential object matches. As with motion detection, we can look for large connected regions using the function `vlDrawBinaryRects()`:

```
numMatches = vlDrawBinaryRects (srcImage, binaryImage,
                                widthMin, heightMin, VL_COLOR_RED);
if (numMatches > 0) {
    printf ("found %d possible object(s)\n", numMatches);
}
```

If `numMatches` is non-zero, then we have found one (or more) matching objects in the image.

B.2.5 VisLib Open Source License

VisLib, a high-performance vision procession library. ActivMedia Robotics, LLC.

LICENSE TERMS

VisLib is protected by the GNU General Public License.

VisLib is owned and distributed by ActivMedia Robotics, LLC

ActivMedia Robotics, LLC

44 Concord St

Peterborough NH 03458

(603) 924-9100

<http://www.activrobots.com>

<http://robots.activmedia.com>

support@activmedia.com

ActivMedia Robotics does not make any representation on the suitability of this software for any purpose.

VisLib is released “as-is”, without implied or express warranty.

This package is VisLib, a vision processing library, developed by Terry Fong and S. Grange from the Virtual Reality and Active Interfaces Group, of the Swiss Federal Institute of Technology (<http://imts7.epfl.ch>).

The VisLib package is owned and distributed by ActivMedia Robotics, LLC (<http://www.activrobots.com> and <http://robots.activmedia.com>). The included source and executables are covered by the GNU General Public License. Please read and understand this license, which is included with the package. If you did not receive the GNU COPYING licence with this product, please contact ActivMedia Robotics immediately at support@activmedia.com.

APPENDIX C

Contextual Inquiry Study

This appendix contains documents related to the contextual inquiry study.

The first section contains an excerpt from the clearance request that was submitted to CMU's Institutional Review Board (IRB). All research conducted at CMU involving human subjects is required to obtain IRB approval.

The second section contains the test plan used to conduct the contextual inquiry study.

The last section contains the questionnaire and the "quick reference" sheets used in the study. All test subjects were asked to complete the questionnaire before participating in the study. Each test subject then received one of the "quick reference" sheets, based on their responses.

C.1 CMU Human Subjects Clearance Request

The following is excerpted from CMU Protocol No. HS01-168 (CMU 2001). All questions regarding the approval process and certification of this research protocol should be addressed to:

Institutional Review Board
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213-3890

C.1.1 Cover Page

CARNEGIE MELLON UNIVERSITY HUMAN SUBJECTS CLEARANCE REQUEST

DATE: 13 August 2001

CMU Protocol No. _____
(for office use only)

New Request X Renewal _____

Principal Investigator(s): Chuck Thorpe (supervisor) / Terry Fong

P.I. Title/Degree: Dr. / Ph.D. student Department Robotics Institute

Phone: x3612 (Dr. Chuck Thorpe) E-mail: {cet, terry} @ri.cmu.edu

Project Dates: From 28 August 2001 To 31 December 2001

Project Title: Vehicle teleoperation system study using contextual inquiry

Name of Experimenter(s): Terry Fong

Source of Funding:

Internal Sponsor: _____ No. _____

External Sponsor: _____ No. _____

Brief Description of Research:

Qualitative assessment of a vehicle teleoperation (remote driving of a mobile robot) system using contextual inquiry.

1. How many subjects will be used in this experiment? 8

2. From what source do you plan to obtain subjects? volunteers (students & community)

3. Is there any benefit gained by the subject for participating? no monetary benefit

4. Will the subjects include any of the following: **NO** X **YES (please check below)** _____

_____ Fetuses	_____ Mentally Retarded
_____ Hospitalized Patients	_____ Minors
_____ Institutionalized Patients	_____ Pregnant Women
_____ Mentally Disabled	_____ Prisoners

5. Degree of Physical Risk: X Negligible Mild Moderate High

6. Degree of Psychological Risk: X Negligible Mild Moderate High

Please submit each of the following with this Clearance Request form:

1. A draft of the proposal or abstract
2. A clear definition of how the subjects will be utilized or how the experimental treatment will be administered
3. A copy of the "informed" consent form(s) which the subjects will be required to sign
4. An indication of how confidentiality/anonymity will be protected.
5. The name(s) and address(es) of official(s) authorizing access to any subjects in cooperating institutions not under the direct control of Carnegie Mellon
6. Risk/Benefit analysis.
7. A copy of your on-line training certificate (<http://cme.nci.nih.gov/>)

C.1.2 Proposal

Background

Vehicle teleoperation means simply: operating a vehicle at a distance. It is used when it is necessary to operate in a hazardous or difficult to reach environment and when the primary task is exploration, observation or reconnaissance. It is also often used to reduce mission cost and to avoid loss of life. Current applications include reconnaissance, surveillance, and target acquisition (RSTA), exploration, remote mining, inspection, facility security and entertainment.

To make vehicle teleoperation more accessible to all users, novices and experts alike, I have developed a new system model called collaborative control (Fong, Thorpe, and Baur 1999). In this model, a human and a robot collaborate to perform tasks and to achieve common goals. Instead of a supervisor dictating to a subordinate, the human and the robot engage in dialogue to exchange ideas, to ask questions, and to resolve differences. Instead of the human always being completely “in control”, the robot is more equal and can treat the human as a limited source of planning and information, just like other system modules.

For example, with collaborative control, a geologist should be able to say, though not necessarily using natural language, to a robot: “Drive along this path and explore the region. If you find an interesting rock, tell me.” Similarly, as the robot is executing the task, it should be able to query the human or to convey information: “How should I get around this obstacle?”, “Is this an interesting rock?”, or “Come help me! I’m stuck!”

Collaborative control is both a novel and a useful paradigm for teleoperation. It is novel because it focuses on dialogue as a mechanism for unifying and coordinating robot action. It is useful because it enables both humans and robots to benefit from each other, thus alleviating some of the common problems in teleoperation.

Purpose of study

In order to gain an understanding of how collaborative control influences human-robot interaction, I am planning to conduct a Contextual Inquiry study of my system. Contextual Inquiry (CI) is a structured interviewing method for grounding the design of interactive systems (e.g., user interfaces) in the context of the work being performed (Holtzblatt and Jones 1993). In essence, CI is an adaptation of ethnographic research methods to fit the time and resource constraints of engineering. As such, it is most appropriate for qualitative system assessment, rather than performance measurement, because the data it provides is primarily subjective in nature.

CI is based on three principles: (1) that understanding the context in which work being performed is essential for design, (2) that the user should be a partner in the design process, and (3) that assessment must be focused on specific system attributes. An interviewer performs CI by observing users while they work, asking questions as they perform tasks in order to understand their motivation and strategy. Typical questions are: What are you doing now? Is that what you expected to happen? What do you especially like or dislike about this tool? Through conversation and discussion, the interviewer and the user develop a shared understanding of the work (Beyer and Holtzblatt 1999).

Collaborative control is designed to allow a variety of users (novices and experts alike) to perform vehicle teleoperation. Thus, I plan to observe 6-8 people having a broad range of backgrounds, skills, and experiences. For this study, I will use the master-apprentice model proposed by Beyer and Holtzblatt (1995). In this model, the user (i.e., the test subject) performs a set of tasks while the interviewer watches. As he works, the user assumes the role of “master”, describing what he is doing and why he is doing it. The interviewer, in the role of “apprentice”, takes notes and asks questions to learn how the master works.

Objective

In this study, I intend to do the following:

- Observe how humans and robots interact in a collaborative control system
- Acquire qualitative understanding of how users react to questions asked by the robot (e.g., do they attribute higher level of intelligence and trust to the robot as a result?)
- Gain insight into how people learn to use the system: what strategies they adapt (switching from active control to answering questions), how human-robot dialogue fosters/hinders learning, etc.
- Obtain feedback about concept and system design: good/bad features, what would be useful to add/remove, etc.

The study will not acquire or obtain metric-based performance data (i.e., no quantitative measurement of task performance or error rates). Only volunteers will be used in this study.

References

- Beyer, H., and Holtzblatt, K. 1995. “Apprenticing with the customer”. *Communications of the ACM* 38(5).
- Beyer, H., and Holtzblatt, K. 1999. “Contextual design”. *Interactions* 6(2).
- Fong, T., Thorpe, C., and Baur, C. 1999. “Collaborative control: a robot-centric model for vehicle teleoperation”. In *Proceedings of the AAAI 1999 Spring Symposium: Agents with Adjustable Autonomy*, Stanford, CA.
- Holtzblatt, K. and Jones, S. 1993. “Contextual inquiry: a participatory technique for system design”. In Schuler, D., and Namioka, A. (eds.) *Participatory design: principles and practice*.

C.1.3 How Subjects Will Be Used

The study will be conducted in an office environment at the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland. I will administer all tests with the authorization of Dr. Charles Thorpe (CMU RI) and Dr. Charles Baur (Swiss Federal Institute of Technology).

All subjects will be given a short, one-page questionnaire to establish background information. The questionnaire will be used solely to ascertain education and experience (in vehicle teleoperation and computer-based interfaces). No names or other biographical data will be collected.

Approximately one hour of time will be required per subject.

No physical work (other than using a computer mouse and touchscreen to remotely drive a mobile robot) will be required. Psychological risk is expected to be negligible or zero.

Data collection method will be primarily performed via written note taking. No audio or video recording device will be used.

Each subject will participate in the study as follows:

1. Introduction

- presentation of study goals, contextual inquiry, and study procedure (duration of test, what will be recorded and how, etc.)
- description of confidentiality procedures
- subject will be asked to read and sign the informed consent form

2. Contextual inquiry

- subject will perform a practice task for 5 minutes (e.g., playing computer solitaire) to gain familiarity with the method
- presentation of vehicle teleoperation system
- subject will be asked to perform a remote driving task (e.g., “drive the mobile robot to the end of the hallway”)

As the subject works, the test administrator will make written notes of what the subject does and says. The administrator will ask questions such as: What are you doing? Why are you doing that? What you're saying leads me to conclude...

3. Wrap-up

- summary of what was observed and learned
- clarification of uncertainties and unanswered questions
- conclusion (ask if can contact if needed, thank for participating)

C.1.4 Consent Form

Carnegie Mellon University Consent Form

Project Title: **Vehicle teleoperation system study using contextual inquiry**
Conducted By: **Terry Fong**

I agree to participate in the observational research conducted by Dr. Charles Thorpe and Dr. Charles Baur (EPFL), or by students under the supervision of Drs. Thorpe and Baur. I understand that the University's Institutional Review Board has reviewed the proposed research and that to the best of their ability they have determined that the observations involve no invasion of my rights of privacy, nor do they incorporate any procedure or requirements that may be found morally or ethically objectionable. If, however, at any time I wish to terminate my participation in this study I have the right to do so without penalty. I have the right to request and keep a copy of this form.

If you have any questions about this study, you should feel free to ask them now or anytime throughout the study by contacting:

Dr. Charles Thorpe
Robotics Institute
Carnegie Mellon University
+1 (412) 268-3612
cet@ri.cmu.edu

Dr. Charles Baur
Robotic Systems Institute (DMT-ISR)
Swiss Federal Institute of Technology (EPFL)
+41 (21) 693 25 69
Charles.Baur@epfl.ch

You may report any objections to the study, either orally or in writing to:

Dr. Ann Baldwin Taylor, IRB Chair
at0j@andrew.cmu.edu
Carnegie Mellon University
(412) 268-4727

Purpose of the Study: I understand I will be learning about a system for remotely controlling a mobile robot. I know that the researchers are studying human-robot interaction and different interfaces for the system. I realize that in the experiment I will learn how to control the system and then use the system for about an hour in an office environment at the Swiss Federal Institute of Technology (EPFL).

Study Procedure: I understand that, in this study, I will first complete a short questionnaire. After a briefing of the study goals and method (contextual inquiry), I will be asked to perform some tasks. While I am working, Terry Fong will observe what I am doing and will ask questions to understand how I am using the system.

I understand that the following procedure will be used to maintain my anonymity in analysis and publication/presentation of any results. Each participant will be assigned a number, names will not be recorded. The researchers will save and reference the data by participant number, not by name. All records will be stored in locked files by Terry Fong at EPFL until December 2001. After this date, the files will be transferred to Dr. Charles Thorpe for permanent storage at CMU. No other researchers will have access to these files.

I understand that in signing this consent form, I give Dr. Thorpe and his associates permission to present this work in written and oral form, without further permission from me.

Name (please print)

Signature

Telephone

Date

C.1.5 Confidentiality

To safeguard the anonymity and confidentiality of study subjects, each subject will be assigned a number. All collected data (written notes) will be referenced by this number.

No names or other biographical information will be recorded.

C.1.6 Authorization for Access to Subjects

List of officials authorizing access to subjects in cooperating institutions not under direct control of CMU:

Dr. Charles Baur Virtual Reality and Active Interfaces Group DMT-ISR Swiss Federal Institute of Technology (EPFL) CH-1015 Lausanne Switzerland

Tel. +41 (21) 693 25 69 Fax. +41 (21) 693 65 40

See attached letter from Dr. Baur.

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY

DEPARTEMENT DE MICROTECHNIQUE (DMT-ISR)
CH-1015 LAUSANNE, SWITZERLAND
Téléphone: +41 (21) 693 58 50 Fax: +41 (21) 693 65 40
E-mail: Charles.Baur@epfl.ch



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, 10 august 2001

Office of the associate provost
Carnegie Mellon University
Warner Hall 405
Pittsburgh, PA 15213 USA

To Whom It May Concern,

Please allow me to introduce myself. I am the leader of the VRAI (Virtual Reality and Active Interface) Group at the Swiss Federal Institute of Technology (EPFL) in Lausanne. My group develops human-machine systems in the areas of medical training, micromanipulation, and remote environments.

Since 1997, the VRAI Group has collaborated with Dr. Charles Thorpe of the Robotics Institute to realise “advanced teleoperation interfaces”. In the framework of this project, we have exchanged personnel (graduate students and assistants), developed human-robot systems and published numerous peer-reviewed papers. I am actually co-supervising Terry Fong, who is one of Dr. Thorpe’s Ph.D. students in Robotics and who is completing his thesis research in my laboratory.

I am writing to inform you that I will be authorizing access to test subjects at the VRAI Group for the purposes of the “Vehicle teleoperation system study using contextual inquiry”. All of these test subjects will be obliged to sign the “consent form” required by CMU before being allowed to participate in the study.

If you have any questions or require additional information, please do not hesitate to contact me at the above address or telephone.

Sincerely yours,

Dr. Charles Baur
Adjoint Scientifique
Microengineering Department
Swiss Federal Institute of Technology
Lausanne, Switzerland

C.1.7 Risk/Benefit Analysis

The level of physical and psychological risk is negligible. The subjects will perform no physical labor and will have an active, guiding role in how contextual inquiry proceeds.

This study will provide valuable insight into a new approach, collaborative control, to vehicle teleoperation. As such, it will benefit researchers and engineers working to construct more flexible and effective human-robot systems. In addition, collaborative control is the central subject of my Robotics Ph.D. thesis. Thus, this study will provide evidence to help support my claim that human-robot collaboration and dialogue are useful tools, which facilitate joint task performance.

In this study, the subjects will learn about vehicle teleoperation and what problems are associated with it. By using my system, the subjects will have the opportunity to remotely drive a mobile robot and will gain first-hand experience with a modern interface design technique (contextual inquiry). Finally, through their feedback, the subjects will be able to directly influence the developing field of human-robot interaction.

C.1.8 NIH Training Certificate



Human Participant Protections Education for Research Completion Certificate

This is to certify that

Terry Fong

has completed the **Human Participants Protection Education for Research Teams** online course, sponsored by the National Institutes of Health (NIH), on 08/10/2001.

This course included the following:

- key historical events and current issues that impact guidelines and legislation on human participant protection in research.
- ethical principles and guidelines that should assist in resolving the ethical issues inherent in the conduct of research with human participants.
- the use of key ethical principles and federal regulations to protect human participants at various stages in the research process.
- a description of guidelines for the protection of special populations in research.
- a definition of informed consent and components necessary for a valid consent.
- a description of the role of the IRB in the research process.
- the roles, responsibilities, and interactions of federal agencies, institutions, and researchers in conducting research with human participants.

National Institutes of Health
<http://www.nih.gov>

<http://cme.nci.nih.gov/cgi-bin/hsp/cts-cert4.pl> 10.08.2001

C.2 Contextual Inquiry Study Plan

C.2.1 Focus

Study human-robot interaction in a collaborative control system:

- user assisting safeguarding autonomy
- coordination: multiple queries from multiple safeguarding behaviors (handling many simultaneous questions)
- dynamic adaptability: novice/expert filtering
- user switching to/from navigation to answering questions
- query rate (novice is asked fewer questions)

C.2.2 Objectives

- acquire qualitative understanding of how users react to questions asked by the robot (e.g., do they attribute higher level of intelligence and trust to the robot as a result?)
- observe how people learn to use the system: what strategies they adapt (switching from active control to answering questions), how human-robot dialogue fosters/hinders learning, etc.
- observe human-robot interaction, particularly how people handle switching roles from controller (giving commands) to sage (giving advice).
- ascertain what a range of users (both novices and experts) think is good/bad, should be added/removed: (1) the concept of collaborative control (2) the existing implementation (UI, controller, etc.)
- do *not* measure user or system performance (i.e., no task completion metrics or error rates).

C.2.3 Approach

Test subjects

- small sample of novice and expert users, none of whom has a priori familiarity with the assumptions, rationale, development or implementation of collaborative control
- novice = no experience with robotics or remote driving, may have experience with computers/PDA, matches “novice” stereotype
- expert = has experience and knowledge of robotics and remote driving, very skilled with computers/PDA, matches “expert” stereotype
- approximately one hour of time will be required per subject
- data collection will be performed via: (1) pre-test questionnaire (background information); (2) written note taking (during contextual inquiry) (3) post-test interview (experience summary)

CI questions

- open-ended, designed to elicit information from the user
 - What are you doing now?
 - Why are you doing that?
 - Is that what you expected to happen?
 - What do you especially like or dislike about this tool?
 - What you're saying leads me to conclude...
 - In other words, you are doing X?

C.2.4 Study Procedure

Introduction

- informed consent
 - description of confidentiality procedures
 - ask subject to read and sign the informed consent form
- pre-test questionnaire
- describe hardware (show robot) and PDA
- describe focus
 - to understand how people interact with an intelligent robot which can operate by itself and can ask questions
 - emphasize that I am testing my system, not you
 - emphasize that I am studying interaction, not performance
 - emphasize there are no right or wrong answers. I'm just trying to determine what is good/bad with the system, what is easy/hard to do.
- describe contextual inquiry
 - what it is, how it works
 - master-apprentice
 - emphasize that I want them to teach me what is happening
 - emphasize importance of explaining (thinking aloud)
 - say what you are trying to do
 - try to give a running commentary of what you do, what surprises you
- describe study procedure
 - duration of test
 - what will be recorded
 - what will do (first a practice, then...)
 - okay to quit at any time

Contextual inquiry

- demonstrate solitaire (how to play it on PDA)
- give PDA to user and let him play solitaire a minute
- practice CI: user plays solitaire, ask CI questions

-
- presentation of vehicle teleoperation interface
 - various modes (note: different for novice/experts)
 - give cheat sheets
 - vehicle teleoperation
 - configure PDA as “novice” or “expert”
 - reassure that cannot hurt robot (show e-stop)
 - ask user to drive robot to explore environment (drive around room)
 - if skilled, ask to drive out the door (place cat) and down the hall

Wrap-up

- summarize what was observed
- clarify uncertainties and unanswered questions (both mine and user’s)
- post-questionnaire
- conclusion (ask if can contact if needed, thank for participating)

C.3 User Questionnaire

General

Age: _____ Gender: ☐ Male ☐ Female Ref. Num: _____

Education (highest degree obtained)

☐ High School / Maturité ☐ Bachelor's ☐ Master's / Diplôme ☐ Ph. D.

Area of interest or study (major): _____

Driving experience

Do you have a driver's license? ☐ Yes ☐ No

If yes, how long have you been licensed: _____

If yes, how often do you currently drive:

☐ daily ☐ weekly ☐ monthly ☐ rarely ☐ never

Computer experience

How often do you use a computer:

☐ daily ☐ weekly ☐ monthly ☐ rarely ☐ never

What type of user do you consider yourself:

☐ novice ☐ experienced ☐ "power" ☐ nerd / hacker

If you own a PDA, how often do you use it:

☐ daily ☐ weekly ☐ monthly ☐ rarely ☐ never

Video game experience

How often do you play video games:

☐ daily ☐ weekly ☐ monthly ☐ rarely ☐ never

What type of games do you play (check all that apply):

☐ action ☐ card ☐ strategy ☐ adventure ☐ other

Remote driving experience

Have you ever remotely controlled or driven a vehicle (toy, robot, etc.)? ☐ Yes ☐ No

If yes, please describe:

If yes, how many times: ☐ once ☐ several ☐ many

C.4 Quick Reference Sheets

Novice Configuration

TOP BUTTONS

TX / RX
green/white packet
yellow image

robot has a question

ask a question

view event log

switch mode

BOTTOM BUTTONS

status

talk

honk

stop the robot

IMAGE MODE

click above the horizon to turn the robot

click below the horizon to define a path (a set of "waypoints") and then press "go" to drive

camera pan/tilt

horizon

yellow lines
width robot width
separation robot length

erase path

drive path

path progress

MAP MODE

click to define a path (a set of "waypoints") and then press "go" to drive

world (global) coordinates

robot (local) coordinates

map colors
white clear
black obstacle
grey unknown

erase path

drive path

path progress

Expert Configuration

TOP BUTTONS

TX / RX
green/white packet
yellow image

robot has a question

ask a question

view event log

switch mode

BOTTOM BUTTONS

status

talk

honk

stop the robot

IMAGE MODE

click above the horizon to turn the robot

click below the horizon to define a path (a set of "waypoints") and then press "go" to drive

camera pan/tilt

horizon

yellow lines
width robot width
separation robot length

erase path

drive path

path progress

image control store get

waypoints clear **go**

motion

stopped

DIRECT MODE

vertical axis
rate mode forward/back speed
pose mode forward/back distance

horizontal axis
rate mode left/right turn speed
pose mode left/right rotation

pose mode

rate mode

red circle (pose mode)
indicates robot size

scale bar
click to change axis range

pose mode

rate mode

map control clear get

waypoints clear **go**

motion

stopped

MAP MODE

click to define a path (a set of "waypoints") and then press "go" to drive

world (global) coordinates

robot (local) coordinates

map colors
white clear
black obstacle
grey unknown

erase path

drive path

path progress

☒ grid

map control clear get

waypoints clear **go**

motion

stopped

References

- Abella, A. and Gorin, A. 1999. "Construct Algebra: analytical dialog management". In *Proceedings of the Annual Meeting of the ACL* (Washington, DC). San Francisco: Morgan-Kaufmann.
- Adams, J. 1996. Human factors experimental analysis of the MASC system. Technical Report MS-CIS-96-11, University of Pennsylvania, Philadelphia, Pennsylvania.
- Albus, J., et al. 1987. NASA/NBS Standard reference model for telerobot control system architecture (NASREM). Technical Note 1235, NIST, Gaithersburg, Maryland.
- Amai, W., Fahrenholtz, J., and Leger, C. 2001. Hands-free operation of a small mobile robot. *Autonomous Robots* 11(1).
- ANSI/AIAA 1993. *Guide to Human Performance Measurements*, ANSI/AIAA G-035-1992. Washington, DC: AIAA.
- Azuma, R. 1997. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments* 6(4).
- Backes, P., Tharp, G., and Tso, K. 1997. "The Web Interface for Telescience (WITS)". In *Proceedings of the IEEE International Conference on Robotics and Automation* (Albuquerque, NM). Piscataway: IEEE.
- Bailey, B. 1996. Drones in southeast Asia. Navarre, Florida: 55th Strategic Reconnaissance Wing Association, Inc.
- Ballou, P. 2000. "Improving pilot dexterity with a telepresent ROV". In *Proceedings of the Vehicle Teleoperation Interfaces Workshop, International Conference on Robotics and Automation* (San Francisco, California). Piscataway: IEEE.
- Baltus, G. et al. 2000. "Towards personal service robots for the elderly". In *Proceedings of the Workshop on Interactive Robotics and Entertainment* (Carnegie Mellon University). Menlo Park: AAIL.
- Bapna, D., et al. 1998. "The Atacama Desert Trek: outcomes". In *Proceedings of the International Conference on Robotics and Automation* (Leuven, Belgium). Piscataway: IEEE.
- Bares, J., and Wettergreen, D. 1999. Dante II: technical description, results, and lessons learned. *International Journal of Robotics Research* 18(7).

-
- Barfield, W and Furness, T. 1995. *Virtual environments and advanced interface design*. Oxford: Oxford University Press.
- Beyer, H., and Holtzblatt, K. 1995. Apprenticing with the customer. *Communications of the ACM* 38(5).
- Beyer, H., and Holtzblatt, K. 1999. Contextual design. *Interactions* 6(2).
- Blackmon, T., and Stark, L. 1996. Model-based supervisory control in telerobotics. *Presence: Teleoperators and Virtual Environments* 5(2).
- Blackmon, T., et al. 1999. Virtual reality mapping system for Chernobyl accident site assessment. In *Proceedings of SPIE Vol 3644* (San Jose, California). Bellingham: SPIE.
- Bonasso, P. 1999. "Issues in providing adjustable autonomy in the 3T architecture". In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy* (Stanford, California). AAAI Technical Report SS-99-06. Menlo Park: AAAI.
- Borenstein, J., and Koren, Y. 1991. Histogramic In-Motion Mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation* 7(4).
- Bouzeid, G. 1998. Acquisition and visualization of ultrasound data and sensor fusion with a 2D camera. Diplôme report, Microengineering Department, Swiss Federal Institute of Technology, Lausanne.
- Breazeal (Ferrel), C. 1998. "Regulating human-robot interaction using emotions, drives, and facial expressions". *Agents in Interaction Workshop, Autonomous Agents*.
- Cabrol N. (ed.). 1999. Proceedings of the astronaut-rover interaction post Silver Lake Field Test debriefing: results, lessons learned, and directions for future human exploration of planetary surfaces. NASA Ames Research Center, Moffett Field, CA.
- Canan, J. 1999. Seeing more, and risking less, with UAV's. *Aerospace America* 37(10).
- Cannon, D., and Thomas, G. 1997. Virtual tools for supervisory and collaborative control of robots. *Presence: Teleoperators and Virtual Environments* 6(1).
- Carrier, D. 1992. "Soviet rover systems". In *Proceedings of the Space Programs and Technologies Conference* (Huntsville, Alabama). Reston: AIAA.
- Churcher, G., et al. 1997. Dialogue management systems: a survey and overview. Report 97.6, School of Computer Studies, University of Leeds.
- Clarke, R. 1994. Asimov's Laws of Robotics: implications for information technology. *IEEE Computer* 26(12) and 27(1). Piscataway: IEEE.
- Clavel, R. 1991. Conception d'un robot parallel rapide à quatre degrés de liberté. Ph.D. Thesis 925. Microengineering Department. Swiss Federal Institute of Technology, Lausanne.

-
- CMU. 2001. Vehicle teleoperation system study using contextual inquiry. CMU Protocol No. HS01-168, Institutional Review Board, Carnegie Mellon University, Pittsburgh, Pennsylvania.
- Cooper, B. 1998. "Driving on the surface of Mars using the Rover Control Workstation". In *Proceedings of SpaceOps '98: the 5th International Symposium on Space Mission Operations and Ground Data Systems* (Tokyo, Japan). Tokyo: NASDA.
- Cooper, G. and Harper, R. 1969. The use of pilot ratings in the evaluation of aircraft handling qualities. NASA TN-D-5153. Washington, DC: NASA.
- Deschler, M. 1998. TCM2 sensor development, Technical Report, Virtual Reality and Active Interfaces (VRAI) Group, Microengineering Development, Swiss Federal Institute of Technology, Lausanne.
- Dorais, G., et al. 1999. "Adjustable autonomy for human-centered autonomous systems". In *International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems* (Stockholm, Sweden). San Francisco: Morgan-Kaufmann.
- Draper, M., and Ruff, H. 2001. "Multi-sensory displays and visualization techniques supporting the control of unmanned air vehicles". In *Proceedings of the Vehicle Teleoperation Interfaces Workshop, International Conference on Robotics and Automation* (San Francisco, California). Piscataway: IEEE.
- Duke, M. 1998. Mars Surface Mission Workshop. LPI Contribution 934. Houston: Lunar Planetary Institute.
- Dutoit, T., et al. 1996. "The MBROLA Project: Towards a set of high-quality speech synthesizers". In *Proceedings of International Conference on Spoken Language Processing* (Philadelphia, Pennsylvania). Piscataway: IEEE.
- Endsley, M. 1996. "Automation and situation awareness". In *Automation and human performance: theory and applications*, edited by R. Parasuraman and M. Mouloua. Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.
- Ferrel, W., and Sheridan, T. 1967. Supervisory control of remote manipulation. *IEEE Spectrum* 4(10).
- Fong, T., et al. 1995. "Operator interfaces and network-based participation for Dante II". In *Proceedings of the 25th International Conference on Environmental Systems* (San Diego, California). Warrendale: SAE.
- Fong, T., Cabrol, N., Thorpe, C., and Baur, C. 2001. "A personal user interface for collaborative human-robot exploration". In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space* (Montréal, Canada). Canadian Space Agency.
- Fong, T., Conti, F., Grange, S., and Baur, C. 2000. "Novel interfaces for remote driving: gesture, haptic, and PDA". In *Proceedings of SPIE Telemanipulator and Telepresence Technologies VII* (Boston, Massachusetts). Bellingham: SPIE.

-
- Fong, T., Grange, S., Thorpe, C. and Baur, C. 2001. "Multi-robot remote driving with collaborative control". In *Proceedings of the 10th IEEE International Workshop on Robot-Human Interactive Communication* (Bordeaux and Paris, France). Piscataway: IEEE.
- Fong, T., and Thorpe, C. 2001. Vehicle teleoperation interfaces. *Autonomous Robots* 11(1).
- Fong T., Thorpe, C., and Baur, C. 1999. "Collaborative Control: A robot-centered model for vehicle teleoperation". In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy* (Stanford, California). AAAI Technical Report SS-99-06. Menlo Park: AAAI.
- Fong, T., Thorpe, C., and Baur, C. 2001a. Advanced interfaces for vehicle teleoperation: collaborative control, sensor fusion displays, and remote driving tools. *Autonomous Robots* 11(1).
- Fong, T., Thorpe, C., and Baur, C. 2001b. "A safeguarded teleoperation controller". In *Proceedings of the International Conference on Advanced Robotics* (Budapest, Hungary), August 2001. Piscataway: IEEE.
- Fong, T., Thorpe, C., and Baur, C. 2001c. "Collaboration, dialogue and human-robot interaction". In *Proceedings of the 10th International Symposium on Robotics Research* (Lorne, Victoria, Australia). London: Springer-Verlag.
- Fourth Planet. 1998. FPC: Fourth Planet Communicator. Version 2. Los Altos, California: Fourth Planet, Inc.
- Gage, D. 1995. UGV History 101: A brief history of Unmanned Ground Vehicle (UGV) development efforts. *Unmanned Systems Magazine* 13(3).
- Gat, E. 1997. On three-layer architectures. In *Artificial Intelligence and Mobile Robots*, edited by D. Kortenkamp, R. P. Bonasso, and R. Murphy. Cambridge: MIT Press.
- Gentner, D. and Grudin, J. 1990. "Why good engineers (sometimes) create bad interfaces". In *Proceedings of CHI '90: Empowering People*. New York: ACM.
- Gevers, T. and Smeulders, W. 1997. "Object recognition based on photometric color invariants". In *Proceedings of Scandinavian Conference on Image Analysis* (Lappeenranta, Finland). Helsinki: Pattern Recognition Society of Finland.
- Glumm, M., Kilduff, P., and Masley, A. 1992. A study on the effects of lens focal length on remote driver performance. Technical Report ARL-TR-25, Army Research Laboratory, Adelphi, Maryland.
- Goldberg, K. and Chen, B. 2001. "Collaborative control of robot motion: robustness to error". In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems* (Maui, Hawaii). Piscataway: IEEE.
- Goren-Bar, D. 2001. "Designing model-based intelligent dialogue systems". In Rossi, M., and Siau, K. 2001. *Information Modeling in the New Millennium*. London: Idea Group.

-
- Gowdy, J. 2000. A qualitative comparison of interprocess communications toolkits for robotics. Technical Report CMU-RI-TR-00-16, Carnegie Mellon University, Pittsburgh, Pennsylvania.
- Grange, S. 2000. Vision-based sensor fusion for active interfaces. Diplôme report, Microengineering Department, Swiss Federal Institute of Technology, Lausanne.
- Grange, S., Fong, T., and Baur, C. 2000. "Effective vehicle teleoperation on the World Wide Web". In *Proceedings of the International Conference on Robotics and Automation* (San Francisco, California). Piscataway: IEEE.
- Graves, A. 1997. Enhancement of a direct control teleoperator system. Working Paper 1, Centre for Computational Intelligence, De Montfort University, Leicester, United Kingdom.
- Graves, A. 1998. User interface issues in teleoperation. Working Paper 3, Centre for Computational Intelligence, De Montfort University, Leicester, United Kingdom.
- Graves, A., and Czarnecki, C. 1999. "A generic control architecture for telerobotics". In *Technical Report UMCS-99-3-1*, University of Manchester.
- Green, A., Huttenrauch, H., and Norman, M. 2000. "User centered design for intelligent service robots". In *Proceedings of the 9th IEEE Robot-Human Interactive Communication* (Osaka, Japan). Piscataway: IEEE.
- Green, A. and Severinson-Eklundh, K. 2001. "Task-oriented dialogue for CERO: a user-centered approach". In *Proceedings of the 10th IEEE International Workshop on Robot-Human Interactive Communication* (Bordeaux and Paris, France). Piscataway: IEEE.
- Habib, M. 2000. "Teleoperation and collaborative control". In *Proceedings of the 6th International Conference on Intelligent and Autonomous Systems* (Venice, Italy). Amsterdam: IOS Press.
- Hainsworth, D. 1993. "Mine emergency survey vehicle: Numbat". In *Proceedings of the 25th International Conference of Safety in Mines Research Institutes* (Pretoria, Republic of South Africa). Pretoria: Department of Mineral and Energy Affairs.
- Hainsworth, D. 2001. Teleoperation user interfaces for mining robotics. *Autonomous Robots* 11(1).
- Hallion, R. 1984. On the frontier, flight research at Dryden, 1946-1981. NASA History Series SP-4303. Washington, DC: NASA.
- Halme, A., and Suomela, J. 2001. Tele-existence techniques of heavy work vehicles. *Autonomous Robots* 11(1).
- Hasemann, J-M. 1995. "Robot control architectures: Application, requirements, approaches, and technologies". In *Proceedings of the SPIE Intelligent Robots and Manufacturing Systems* (Philadelphia, Pennsylvania). Bellingham: SPIE.

-
- Hine, B, et al. 1994. "The application of telepresence and virtual reality to subsea exploration". In *Proceedings of ROV '94: The IARP 2nd Workshop on Mobile Robots for Subsea Environments* (Monterey, California).
- Holtzblatt, K. and Jones, S. 1993. "Contextual inquiry: a participatory technique for system design". In *Participatory design: principles and practice*, edited by D. Schuler and A. Namioka. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Jones, C. 1997. *Unmanned aerial vehicles (UAVS). An assessment of historical operations and future possibilities*. Maxell Air Force Base: Air Command and Staff College.
- Kay, J. 1997. STRIPE: Remote driving using limited image data. Ph.D. dissertation, Computer Science, Carnegie Mellon University.
- Keates, S., Clarkson, J., and Robinson, P. 1999. "Designing a usable interface for an interactive robot". In *Proceedings of the 6th International Conference on Rehabilitation Robotics* (Stanford, CA). Stanford: Stanford University.
- Konolige, K., and Myers, K. 1997. "The Saphira architecture for autonomous mobile robots". In *Artificial Intelligence and Mobile Robots*, edited by R. Bonasso and R. Murphy. Cambridge: MIT Press.
- Kress, G., and Almaula, H. 1988. Sensorimotor requirements for teleoperation. Report R-6279, FMC Corporation, San Diego, California.
- Krotkov, E., et al. 1996a. "Safeguarded teleoperation for lunar rovers". In *Proceedings of the 30th International Conference on Environmental Systems* (Monterey, California). Warrendale: SAE.
- Krotkov, E., et al. 1996b. "Safeguarded teleoperation for lunar rovers: from human factors to field trials". In *Proceedings of the IEEE Planetary Rover Technology and Systems Workshop*. Piscataway: IEEE.
- Laengle, T., Hoeniger, T., and Zhu, L. 1997. "Cooperation in Human-Robot-Teams". *International Conference on Informatics and Control* (St. Petersburg, Russia). Moscow: Russian Academy of Sciences.
- Laird, R., et al. 2001. "Issues in vehicle teleoperation for tunnel and sewer reconnaissance". *Proceedings of the Vehicle Teleoperation Interfaces Workshop, International Conference on Robotics and Automation* (San Francisco, California). Piscataway: IEEE.
- Lane, C., Carignan, C., and Akin, D. 2001. Advanced operator interface design for complex space telerobots. *Autonomous Robots* 11(1).
- Lansdale, M. and Ormerod, T. 1994. *Understanding Interfaces*. London: Academic Press.
- Lin, I., et al. 1995. "An advanced telerobotic control system for a mobile robot with multisensor feedback". In *Intelligent Autonomous System IAS-4*, edited by U. Rembold and R. Dillmann. Amsterdam: IOS Press.

-
- Maslowski, A., et al. 1998. "Autonomous mobile robot controller for teleoperation system". In *Proceedings of International Symposium on Measurement and Control in Robotics*. Prague, Czech Republic. Prague: IMEKO.
- Matijevic, J. 1998. Autonomous navigation and the Sojourner microrover. *Science* 276.
- McGovern, D. 1988. Human interfaces in remote driving. Technical Report SAND88-0562, Sandia National Laboratory, Albuquerque, New Mexico.
- McGovern, D. 1990. Experiences and results in teleoperation of land vehicles. Technical Report SAND 90-0299, Sandia National Laboratories, Albuquerque, New Mexico.
- Meier, R. 1999. Sensor fusion for teleoperation of a mobile robot. Diplôme report, Swiss Federal Institute of Technology, Lausanne.
- Meier, R., Fong, T., Thorpe, C., and Baur, C. 1999. "A sensor fusion based user interface for vehicle teleoperation". In *Proceedings of Field and Service Robots* (Pittsburgh, Pennsylvania). Pittsburgh: The Robotics Institute, Carnegie Mellon University.
- Michel, O. Saucy, P., and Mondada, F. 1997. "KhepOnTheWeb: An experimental demonstrator in telerobotics and virtual reality". In *Proceedings of the IEEE VSMM'97* (Geneva, Switzerland). Piscataway: IEEE.
- Milgram, P., Zhai, S., and Drascic, D. 1993. "Applications of Augmented Reality for Human-Robot Communication". In *Proceedings of the International Conference on Intelligent Robots and Systems* (Yokohama, Japan).
- Murphy, R., and Rogers, E. 1996. Cooperative assistance for remote robot supervision. *Presence: Teleoperators and Virtual Environments* 5(2).
- Muscettola, N., et al. 1998. Remote Agent: to boldly go where no AI system has gone before. *Artificial Intelligence* 103(1-2).
- Musliner, D. and Krebsbach, K. 1999. "Adjustable autonomy in procedural control for refineries". In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy* (Stanford, California). AAAI Technical Report SS-99-06. Menlo Park: AAAI.
- Myers, B, Stiel, H, and Gargiulo, R. 1998. "Collaboration using multiple PDA's connected to a PC". In *Proceedings of the Conference on Computer-Supported Cooperative Work* (Seattle, Washington). New York: ACM.
- Myers, K. and Morley, D. 2001. "Human directability of agents". In *Proceedings of K-CAP'01* (Victoria, British Columbia). New York: ACM.
- NSSDC. 2000. NSSDC Master Catalog. <http://nssdc.gsfc.nasa.gov>, National Space Science Data Center, NASA Goddard Space Flight Center, Goddard, Maryland.

-
- National Research Council. 1996. *Undersea vehicles and national needs*. Committee on Undersea Vehicles and National Needs, Marine Board, Commission on Engineering and Technical Systems, National Research Council. Washington, DC: National Academy Press.
- Newman, J., and Stakes, D. 1994. "Tiburon: Development of an ROV for ocean science research". In *Proceedings of Oceans/Osates '94 conference* (Brest, France). Piscataway: IEEE.
- Newman, W., and Lamming, M. 1995. *Interactive System Design*. Boston: Addison-Wesley.
- Nguyen, L., et al. 2001. Virtual reality interfaces for visualization and control of remote vehicles. *Autonomous Robots* 11(1).
- Norman, D. and Draper, S. (eds). 1986. *User Centered System Design*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Nourbakhsh, I., et al. 1999. An affective mobile robot educator with a full-time job. *Artificial Intelligence* 114(1-2).
- Parrish, J., Akin, D. and Gefke, G. 2000. "The Ranger Telerobotic Shuttle Experiment: implications for operational EVA/robotic cooperation". In *Proceedings of 30th International Conference on Environmental Systems* (Toulouse, France). Warrendale: SAE.
- Paulos, E., and Canny, J. 2001. Designing personal tele-embodiment. *Autonomous Robots* 11(1).
- Perzanowski, D., et al. 1999. "Goal tracking in a natural language interface: towards achieving adjustable autonomy". In *Proceedings of 1999 International Symposium on Computational Intelligence in Robotics and Automation* (Monterey, CA). Piscataway: IEEE.
- Perzanowski, D., Adams, W., Schultz, A., and Marsh, E. 2000. "Towards seamless integration in a multi-modal interface". In *Proceedings of the Workshop on Interactive Robotics and Entertainment* (Carnegie Mellon University). Menlo Park: AAAI.
- Power, C., and Boulton, T. 2000. "Evaluation of an omnidirectional vision sensor for teleoperated target detection and identification". In *Proceedings of the Vehicle Teleoperation Interfaces Workshop, International Conference on Robotics and Automation* (San Francisco, California). Piscataway: IEEE.
- Rahim, W. 1993. "Feedback limited control system on a skid-steer vehicle". In *Proceedings of the ANS Fifth Topical Meeting on Robotics and Remote Systems* (Knoxville, Tennessee). LaGrange Park: ANS.
- Raskin, J. 2000. *The Humane Interface*. Boston: Addison-Wesley.
- RedZone. 1997. Functions and requirements for the Pioneer system. Document number 96055-REPT-004.2, RedZone Robotics, Inc., Pittsburgh, Pennsylvania.

-
- Rogers, E., Murphy, R., and Ericson, B. 1997. "Agent-based expert assistance for visual problem solving". In *Proceedings of the First International Conference on Autonomous Agents*. New York: ACM Press.
- Rogers, T., and Wilkes, M. 2000. "The Human Agent: a work in progress toward human-humanoid interaction". In *Proceedings of the International Conference on Systems, Man, and Cybernetics* (Nashville, TN). Piscataway: IEEE.
- Rosenblatt, J. 1995. "DAMN: A distributed architecture for mobile navigation". In *Proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents* (Stanford, California). Menlo Park: AAAI.
- Sanders, M., and McCormick, E. 1993. *Human Factors in Engineering and Design*. New York: McGraw-Hill.
- Scerbo, M. 1996. "Theoretical perspectives on adaptive automation". In *Automation and human performance: theory and applications*, edited by R. Parasuraman and M. Mouloua. Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.
- Sheridan, T. 1992. *Telerobotics, automation, and human supervisory control*. Cambridge: MIT Press.
- Sheridan, T. 1993. Space teleoperation through time delay: review and prognosis. *IEEE Transactions on Robotics and Automation* 9(5).
- Sheridan, T. 1997. "Eight ultimate challenges of human-robot communication". In *Proceedings of the 6th IEEE International Workshop on Robot and Human Communication* (Sendai, Japan). Piscataway: IEEE.
- Shoemaker, C. 1990, "Low data rate control of unmanned ground systems". In *Proceedings of the Association for Unmanned Vehicle Systems* (Dayton, Ohio).
- Siegwart, R., and Saucy, P. 1998. "Interacting mobile robots on the Web", *Proceedings of the Internet Robotics Workshop, International Conference on Robotics and Automation* (Detroit, Michigan). Piscataway: IEEE.
- Simsarian, K. 2000. Toward human-robot collaboration. Ph. D. dissertation, Computational Vision and Active Perception Laboratory, Swedish Institute of Computer Science.
- Simmons, R. 1996. Where in the world is Xavier, the robot? *Robotics and Machine Perception, Special Issue: Networked Robotics* 5(1).
- Siuru, B. 1989. Robo Warriors. *Mechanical Engineering* 111(5).
- Stone, H. W. 1996. "Mars Pathfinder Microrover: A low-cost, low-power spacecraft". In *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics* (Madison, Wisconsin). Reston: AIAA.

-
- Takeda, H., et al. 1997. "Towards ubiquitous human-robot interaction". In *Working Notes for International Joint Conference on Artificial Intelligence Workshop on Intelligent Multimodal Systems*. San Francisco: Morgan-Kaufmann.
- Taylor, K., Trevelyan, J. 1995. "A telerobot on the World Wide Web". In *Proceedings of the National Conference of the Australian Robot Association* (Melbourne, Australia).
- Terrien, G. 2000. Sensor fusion interface for teleoperation. Diplôme report. Microengineering Department, Swiss Federal Institute of Technology, Lausanne.
- Terrien, G., Fong, T., Thorpe, C., and Baur, C. 2000. "Remote driving with a multisensor user interface". In *Proceedings of the 30th International Conference on Environmental Systems* (Toulouse, France). Warrendale: SAE.
- Terveen, L. 1994. An overview of human-computer collaboration. *Knowledge-Based Systems* 8(2-3).
- Trevino, R., et. al. 2000. "First astronaut-rover interaction field test". In *Proceedings of the 30th International Conference on Environmental Systems* (Toulouse, France). Warrendale: SAE.
- Tsai, R. 1986. "An efficient and accurate camera calibration technique for 3D machine vision". In *Proceedings of Computer Vision and Pattern Recognition* (Miami Beach, Florida). Piscataway: IEEE.
- Tufte, E. 1990. *The Visual Display of Quantitative Information*. Princeton: Graphics Press.
- Ulrich, I., and Borenstein, J. 1998. "VFH+: Reliable obstacle avoidance for fast mobile robots". In *Proceedings of the International Conference on Robotics and Automation* (Leuven, Belgium). Piscataway: IEEE.
- Van der Loos, H. F. M. 1993. A history list design methodology for interactive robotic systems. Ph.D. dissertation, Mechanical Engineering, Stanford University.
- Vandapel, N. et al. 1999. "The EVA Robotic Assistant Experiment on the Houghton-Mars 99 Expedition". In *Proceedings of the Mars Society Convention* (Boulder, Colorado). Boulder: Mars Society.
- Weisbin, C. and Rodriguez, G. 2000. NASA robotics research for planetary surface exploration. *IEEE Robotics and Automation Magazine* 7(4).
- Wettergreen, D., et al. 1995. "Gait execution for the Dante II walking robot". In *Proceedings of the 1995 IEEE Conference on Intelligent Robots and Systems*. Piscataway: IEEE.
- Wettergreen, D., et al. 1997. "Operating Nomad during the Atacama Desert Trek". In *Proceedings of the Field and Service Robotics Conference* (Canberra, Australia). Canberra: The Australian National University.

-
- Whittaker, W., and Champeny, L. 1988. "Conception and development of two mobile teleoperated systems for TMI-2". In *Proceedings of the ANS/ENS International Meeting and Topical Meeting TMI-2 Accident: Materials Behavior and Plant Recovery Technology*. LaGrange Park: American Nuclear Society.
- Wilcox, B., Cooper, B., and Salo, R. 1986. "Computer Aided Remote Driving". In *Proceedings of AUVS-86* (Boston, MA).
- Wilcox, B., et al. 1992. "Robotic vehicles for planetary exploration". In *Proceedings of the International Conference on Robotics and Automation* (Nice, France). Piscataway: IEEE.
- Woods, D. 1996. "Decomposing automation: apparent simplicity, real complexity". In *Automation and human performance: theory and applications*, edited by R. Parasuraman and M. Mouloua. Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.
- Wise, J. 1987. Design of sensor fusion displays: Human factors and display system guidelines. Research Report 87-1C60-SCVIS-R1. Westinghouse Research Center, Pittsburgh, Pennsylvania.
- Young, S. and Proctor, C. 1989. The design and implementation of dialogue control in voice operated database inquiry systems. *Computer Speech and Language* 3.

