RICE UNIVERSITY

A SEMI-AUTONOMOUS MOBILE ROBOT/TELEOPERATOR WITH APPLICATIONS AS AN AID FOR SEVERELY HANDICAPPED PEOPLE

by

MICHAEL ANTHONY REGALBUTO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE DOCTOR OF PHILOSOPHY

APPROVED, THESIS COMMITTEE

John B. Cheatham Jr.,
Professor of Mechanical Engineering,
Director

Thomas A. Krouskop,
Adjunct Professor, Mechanical Engineering,
Co-director

John E. Akin,
Professor of Mechanical Engineering,
Department Chairman

John K. Bennett
Assistant Professor, Electrical and Computer Engineering

Houston, Texas
January, 1990
A SEMI-AUTONOMOUS MOBILE ROBOT/TELEOPERATOR
WITH APPLICATIONS AS AN AID FOR
SEVERELY HANDICAPPED PEOPLE

MICHAEL ANTHONY REGALBUTO

Abstract

A framework for a practical, easy-to-use system which can allow severely handicapped people to control and interact with their home and work environments is presented. Software written for a personal computer allows the user to command a robot to move about a structured indoor environment as well as control household appliances. Underlying these functions is a user-generated computer model of the environment which maintains spatial and relational information about household items. The model represents a data base which is accessed by a variety of procedures in commanding the robot to perform specific tasks, such as localization and navigation. Developed incrementally, a model can take on as much detail as necessary to command the robot to perform a given task. Beyond its use as a domestic aid, the system can serve as an inexpensive testbed for research in intelligent mobile robotics and can ultimately be targeted for use by the general population.

The system is composed of relatively inexpensive equipment designed for in-home use. Software is written in object-oriented Allegro Common LISP and runs on a Macintosh SE having 4Mb of RAM. A HERO 2000 robot was assembled from a kit and modified to extend its workspace to cover desks and tabletops. Control of lamps and household appliances is via an X-10 Powerhouse computer interface module, which is connected to the SE. An Esteem 9600 baud wireless modem augments the 600 baud radio frequency communications link provided with the robot, allowing untethered operation. A custom Machine Control Interface has been developed for interfacing severely physically handicapped users to the system.
ACKNOWLEDGEMENTS

FORMAL

This research was sponsored in part by grants from the Clayton Foundation for Research, which owns the copyright to all software written for the control interface. Computer hardware and various other equipment were supplied by the Department of Mechanical Engineering and Materials Science at Rice University, and The Institute for Rehabilitation and Research in the Texas Medical Center.

PERSONAL

I would like to thank Dr. Cheatham for his encouragement in helping me in my decision to return to Rice, for giving me the freedom to follow my own ideas, the time to pursue outside interests, and for all the free long distance phone calls to California he doesn't know about. To Dr. Tom Krouskop, I give thanks first for his financial support, but mostly for the pleasure of having worked with him, regardless of the days he was cranky and I had to make fun of him behind his back. I appreciate greatly the friendship afforded me by the gentlemen in the lab and department, Misters Jack Chen, Sarmad Adnan, Paul Fisher, Chris Wu, John Norwood, Pete Weiland, Mike Welsh, Suri Suryanaryana, Aladin Boriek, and Sean O'Brien, most of whom I will probably have forgotten in five years. Lastly, I thank my fiance Debbie, who will beat me severely if I don't.
Table of Contents

Abstract
Acknowledgements
Chapter 1: Introduction
  Objectives
  Specifications and Requirements
  Design Modifications
  Background
  Comparison
  User Interfaces
World Modelling with an Object Oriented Approach
  The Three Laws
  Reasonableness Assumption
  Utility
  The Future
Chapter 2: World Modelling
  The Need for a Model
  The Frame Problem
  Model Accuracy
  World Modeller - Create Mode
  Region Dialog Item
  Zoom Region Dialog Item
  Specialized Region Dialog Items
    Lamps
  Robot Modelling
  Alternate Input Methods
  Text Input Dialog
  Object Information Dialog
  Sliders and Spinners
  View Menu
  Menubar
# Table of Contents cont.

| Chapter 3: Robot Command Generation          | 38 |
| Command Dialogs                            | 38 |
| Teleoperator Dialog                        | 39 |
| Teach Facilities                           | 41 |
| Robot Controller Dialog                    | 44 |
| World Modeller Dialog - Command Mode       | 44 |
| Chapter 4: Robot Localization              | 48 |
| Localization Accuracy                      | 57 |
| Chapter 5: Path Planning                   | 58 |
| Nodal Network                              | 58 |
| Moving from Room to Room                   | 63 |
| Point to Point Navigation                  | 66 |
| Chapter 6: Results of Experimental Testing and Demonstrations | 68 |
| Chapter 7: Directions for Further Research  | 71 |
| Manipulation                               | 71 |
| Communications                             | 72 |
| Onboard Programming                        | 72 |
| Hardware Upgrades                          | 72 |
| Modelling Improvements                     | 73 |
| Exploring/Mapping                          | 75 |
| Other Environments                         | 75 |
| Teaching                                   | 75 |
| Technology Transfer                        | 76 |
| Chapter 8: Summary and Conclusions         | 77 |
| References                                 | 80 |
# Table of Contents cont.

Appendix 1: System Configuration and Setup  84  
- Schematic  84  
- Assembly  85  
- Machine Control Interface  86  
- HERO 2000 Robot  89  
- Modifications  89  
- Sensors  91  
- BASIC and DOS  91  
- Communications  92  
- Remote Console  92  
- Wireless Modem  94  
- Host Computer  95  
- X-10 Powerhouse Computer Interface and Modules  95  

Appendix 2: Software Basics  94  
- LISP  96  
- Recursion  96  
- Object-Oriented Programming  97  
- Object Hierarchy and Inheritance  99  
- Interactive Graphics  102  
- System Software  103  
- Master/Slave Architecture  103  

Appendix 3: Biographical Note  106
List of Figures

1. Instantiation of an Object 16
2. Active Sites of a Region Dialog Item 19
3. Example Relational Hierarchy of Region Dialog Items 21
4. Combining Specialized Dialog Items 23
5. Kinematic Quantities of the Modified HERO 2000 26
6. Geometric Construction for Solving Inverse Kinematics 27
7. Robot Region Dialog Item 29
8. Text Input Dialog 31
9. Object Information Dialog 32
10. Sliders and Spinners 33
11. View Menu 34
12. Zoom Region Dialog Item 36
13. Schematic of Menubar 37
14. World Modeller Cursors 39
15. Teleoperator and Robot Controller Dialogs 41
16. Replay Dialog 43
17. A Typical Model Environment 45
18. Horizontal Workspace of Gripper 46
19. Sensor Sweep 49
20. Geometric Construction for Triangulation by Triangles 50
21. Arc Radius and Center 54
22. Geometric Construction for Triangulation by Arcs 55
23. Parent Node Generation 59
24. Child Node Assignment 60
25. Path Planning Flowchart 62
26. Path Expansion Toward Destination 63
27. Room to Room Path Planning 64
28. Point to Point Navigation 66
29. Wheel Rotation and Base Movement 67
30. Robot Control System 85
31. Schematic of Machine Control Interface System 87
32. Schematic of Machine Control Interface Encoder 88
33. Workspace of the Modified HERO 2000 Arm 90
List of Figures cont.

34. Recursive vs. Looping Iteration 98
35. Class vs. Instance 100
36. Allegro Common Lisp Class Hierarchy 101
37. Master/Slave Control Software Architecture 105

List of Tables

1. Minimum and Maximum Horizontal Workspace Radii 47
   as a Function of Gripper Height Conditions
Chapter 1: Introduction

This chapter presents an overall outline of the thesis, including the objectives, specifications, and motivation. The state of the art in robotic aids for the handicapped is reviewed, and comparisons are drawn with the system developed in this thesis.

Objectives

This thesis presents the results of an experimental project aimed at developing a mobile robotic system as an aid for persons who are severely physically handicapped. One hypothesis tested in this thesis is that a useful and inexpensive system can be created by integrating innovative software with a combination of off-the-shelf and custom hardware. In designing the system, a range of problems was addressed:

1) How can a severely physically handicapped user be interfaced to the system?
2) Can the system be made simple enough to be operated by a nontechnical user?
3) How can the robot be controlled both semi-autonomously and as a teleoperator?
4) Can the hardware be made to cost less than 1/4 of an attendant's $25,000 annual salary?
5) Can the robot reduce a disabled person's dependence on a human attendant by enabling the person to perform pick and place tasks?

The first of these questions is answered by using the custom designed Machine Control Interface (MCI), which transduces head motion and breath action into signals for controlling the cursor of a personal computer. A second hypothesis tested in this thesis is that a software interface based solely on cursor action can provide sufficient means for generating commands for the robot. Simplicity and ease of use are accounted for by using hardware designed for in-home use, and by making the software self-booting. Software for the system is written to emulate commonly available interactive graphics applications,
with special emphasis on making the interface completely cursor-driven. Direct, teleoperator control of the robot is accomplished using a custom designed control panel window, in which cursor action is translated into motion commands for individual joints. In solving the more difficult problem of semi-autonomous operation, the concept of world modelling is developed, and AI programming techniques are applied. Inverse kinematic commands, those that involve coordinated movement of several joints, are generated based on user-directed activity of a model robot, a concept referred to as "model-reflective command generation".

Costs are reduced by incorporating off-the-shelf technology wherever possible, with modifications being made where necessary in order meet with the project specifications. Aside from the Machine Control Interface, a framework for a useful system was developed using a modified HERO 2000 robot, a Macintosh SE host computer, and the X-10 Powerhouse household appliance controller. Such equipment should be able to be marketed for under $6500. Utility is enhanced by using equipment that is multi-functional. The MCI is designed to serve as an all-in-one control unit, capable of generating control signals for a variety of machines, including wheel chairs, a musical synthesizer, and personal computers. The host computer and the robot itself have intrinsic value as personal computers, and are members of two very popular brands for which a great deal of software exists. Also, even in the absence of the robot, the system can serve as an environmental control unit.

The remainder of this chapter presents the specifications and requirements for the system, and contrasts the approach taken in this thesis with that of other researchers. Several points concerning the motivation behind this thesis are also discussed. In Chapter 2, the concepts of object and world modelling are developed within the context of interactive computer graphics. Specific components of the graphics interface are
mentioned. Teleoperator and semi-autonomous command generation are described in Chapter 3, as well as the foundations of a facility for teaching the robot. Chapters 4 and 5 describe accomplishments in two essential aspects of robot autonomy, localization based on sensory feedback, and global path planning using the world model.

Appendix 1 gives a general description of the components of the system, and how they are physically connected. Particular attention is paid to the Machine Control Interface. Appendix 2 discusses the fundamental tools used in developing the robot control software, including object-oriented programming, interactive graphics, and the Master-Slave software architecture. A listing of the source code written for the thesis appears in Appendix 3.

Specifications and Requirements

The robotic system is designed to perform simple pick and place tasks within the user's home or office environment, either under the direct control of the user, or semi-autonomously. To establish particular system requirements, potential users were surveyed as to what tasks they would like to be able to do with the help of a robot. The following is a list of desired tasks that were deemed feasible given the currently available technology:

- pick up papers and other items from the floor
- load and unload computer disks
- turn pages
- reach for groceries and other small objects
- open and close doors, curtains, and drawers
- lock and unlock doors.
- push buttons on TV and stereo
- operate faucets
The above tasks involve the robot reaching and manipulating a variety of small objects that could be located on tabletops, the floor, or in between. As such, the robot is required to be able to manipulate objects on surfaces between ground level and a table top height of 33 inches, with a depth of reach of 24 inches at table top height. The robot’s gripper must be able to handle objects measuring up to 5 inches wide and weighing up to 3 pounds. No particular object shape is assumed, though cylindrical and boxy objects are most typical.

In a general sense, the robot must be able to maneuver autonomously about the user’s home or office environment, avoiding major obstacles such as furniture and walls. Also, the robot should be able to drive over smaller obstacles, such as raised thresholds and electrical cords, without excessive tipping or shaking. The robot should turn sharply enough to move about in hallways that are as narrow as 3 feet.

Design Modifications

In order to meet the above specifications, the HERO 2000 robot used in the project had to be modified. Modifications were implemented in the course of design projects involving groups of Mechanical Engineering seniors at Rice University. Part of the effort of this thesis was in recommending and supervising design improvements, and in integrating the results of the groups' efforts into a working system.

The original robot was able to reach no higher than 30 inches, with no depth of reach at that height. To meet the workspace requirements, the arm of the robot was lengthened and repositioned in a second torso section [Andersen et. al., 1987], (Appendix 1). A shock absorbing suspension has been designed for replacement of the fixed suspension of the original robot [Badders et. al., 1989], alleviating problems with jerkiness in stopping, starting, and traversing small obstacles. Following the designs proposed by Badders, a 3-
degree-of-freedom wrist and a gripper with improved payload, dexterity, and sensing are currently under development [Bensten et. al., 1989].

While the HERO is equipped with a number of sensors (Appendix 1), others have been added for the purpose of enhancing autonomy in grasping [Fisher, 1989]. A sonar range finder has been developed for attachment to the main body of the original gripper, allowing robot to home in on an object once the gripper is placed in the neighborhood of the object. An infrared LED pair has been placed in the fingertips for detecting the presence or absence of an object within the gripper.

Background

Several groups have been developing robotic systems as a partial alternative to paid human or trained animal assistants for severely physically disabled persons in domestic and vocational settings [Engelhardt, 1983, 1987, 1988; Holloway et. al., 1985; Fu, 1986; Leifer et. al. 1986; Seamone and Scheisser, 1986; Cheatham et. al., 1987a, 1988; Hammel et. al., 1988; Lees et. al., 1987, 1988; Nof et. al., 1988]. Projects at Stanford University, Carnegie Mellon University, and Boeing Aerospace Company have produced prototype aids in the form of voice activated, menu-driven robotic workstations. Such a system includes a small (having a working envelope of roughly 1 cubic meter) fixed-base robot controlled through a speaker dependent speech recognition system resident on a small personal computer or larger computer workstation. Within reach of the robot are an arrangement of appliances and custom designed fixtures, and storage spaces for holding books, papers, and personal items. Most recently, PRAB Robotics has commercialized such technology in the PRAB "Command 1" workstation [PRAB 1988].

In the realm of mobile robotic aids, a joint effort between Stanford and the Palo Alto Veterans Administration has produced a prototype based on experience gained from a
series of projects begun in the late 70's [Van der Loos et. al., 1988]. An industrial grade Puma 260 arm is mounted atop a custom designed omnidirectional base. Attached to the arm is a two-fingered gripper equipped with a 6-axis force sensor and an array of 12 optical proximity sensors. A CCD camera is mounted on the arm, providing visual feedback to the user. A laser scanner in the base is used for localization within a room, in combination with wall-mounted reflectors. The base also has an instrumented bumper system for sensing collisions with obstacles. A separate control console/worktable serves as the common ground between the robot and user. A Kurzweil speech recognition system is the main user interface for the system, and a command vocabulary of some 70 phrases has been developed.

An extensive system of computer hardware and software has been developed for the above project [Michalowski, 86]. Onboard the robot, an LSI11/73 processor communicates with a number of parallel and serial I/O modules via a 22-bit QBUS. Programming is done in MicroPower Pascal, which allows concurrent execution of the nine software modules used in controlling the various features of the robot. At the control console, an IBM PC/AT is used as a host computer for the speech recognition system, a separate speech synthesis unit, and color graphics equipment. Software is developed by a team of programmers working on a separate Vax 11/750 computer, and is downloaded to the robot.

Comparison

While elegant in its use of advanced technology, the Stanford/VA project suffers from the standpoints of cost, complexity, and shear bulk. The ultimate measuring stick of any such project is its utility to the end user, and this thesis demonstrates for the first time that utility can be derived at a relatively low cost using a system composed of an educational
grade robot, a small personal computer, custom software, and a multipurpose Machine Control Interface.

Development time and cost were substantially decreased by using the HERO 2000 robot and making minor modifications to it. The PUMA 260 has definite advantages in accuracy, precision, and payload over HERO 2000 arm. However, the workspace of the Puma arm is limited, since is fixed to the top of the omnidirectional base and cannot reach the ground. The developers of the Stanford/VA project do mention the future design of additional degrees of freedom for gross positioning of the robot arm. The omnidirectional ability offers marginal utility above that of the twin-opposed drive of the HERO, particularly in making gross movements about the environment. Very simple and robust routines exist for implementing paths as a combination of straight line segments connected by pure rotations (Chapter 5).

Owing to the large user control console and 350 pound weight of the mobile robot, the Stanford/VA system is difficult to transport. The bulkiest part of the system developed in this thesis is the modified HERO, which weighs less than 100 pounds, is easy to handle by two people, and can fit into the back seat of a car. Hence, the entire system is portable between remote locations. With hardware upgrades (Chapter 6), all components of the system besides the robot can carried on the user's wheelchair.

Software for the system boots automatically after the host is turned on. The most recent software version was developed by a single programmer (the author) in less than 2 years time, on the same machine that is used for controlling the robot. In designing the system architecture, special emphasis was placed on reliability (Appendix 2). This thesis is unique in its development of comprehensive robot control routines within a personal computer environment.
User Interfaces

Robotic technology is relatively new, and as with any new technology, questions arise as to how to present and interface the technology to nontechnical end users. Speech has been used by many of the researchers previously mentioned as well as others, [Cheatham et. al., 1897b; Crangle et. al., 1988; Mital & Leng, 1988] owing to its "naturalness" as a means of communication. However, voice input was judged unacceptable in the context of the current project for reasons of safety, reliability, vocabulary limitations, and unnecessary cost and complexity.

Since the current head motion controlled interface is used not only for commanding the robot, but for controlling the user's wheelchair, a comparable speech controlled interface would need to do the same. In order to be inherently safe for controlling a wheelchair, a speech system would need to have a 100% recognition rate. In laboratory conditions, recognition rates are significantly less than perfect [Wicke, 1983]. Recognition rates are affected by ambient noise and changes in a user's voice quality due to factors such as fatigue, stress, or panic.

Current speech recognition systems are speaker-dependent, requiring the user to train and periodically refresh a vocabulary of phrases. In any application, special attention must be paid in developing a set of meaningful, distinctly pronounced phrases, which the user must memorize. In developing the robotic system at Boeing, Fu reported how the need for an expanded command vocabulary was met by using two speech systems together [Fu, 1986].

The personal computer industry has answered the interface question in part by introducing the interactive user interface, complete with mouse, menus, icons, windows, and graphics, which serves to remove the user a step or two away from the details of the machine's operation, establishing the concept of "user friendliness". This thesis presents
an extension of the interactive interface for robot control, in which the Machine Control Interface acts as a mouse for cursor control. It is hoped that interaction with a robot can be made into child's play for adults and children alike.

World Modelling with an Object Oriented Approach

The foregoing assertion is realized by presenting the user the means to construct a model of his own living or working environment, using everyday objects as primitives. Various researchers have pointed out the need for world modelling as a basis for robot control [Palma-Villon & Dauchez, 1986; Sato & Hirai, 1987; Hirai & Sato, 1988; Vuscovic, 1988; Even & Marce, 1988]. While Vuscovic approached the problem of data representation and manipulation in a world model using a variation of the UNIX shell, Hirai and Sato demonstrated the usefulness of an Object Oriented System (OOS) as a natural framework for representing both geometric data and manipulation procedures relevant to particular objects in the environment. The later approach was taken in the current work, in which an object oriented dialect of LISP known as Allegro Common [Coral Software, 1988], was used. Allegro Common is written for the Macintosh line of computers, and has full access to its ROM-based event handling, file handling, and graphics routines. By including an interactive graphics software package that emulates those commonly available for personal computers (MacDraw, MacDraft) as a front end to an object oriented data base, a fully 3-dimensional model of the environment can be generated quickly and easily [Regalbuto et. al., 1989].

The Three Laws

As this thesis promotes a system for domestic or occupational use, the philosophical underpinnings of the work must deal to some extent with society's relationship with
Overcoming fear of subservience on one hand, and overexpectations of ability on the other, (and economics, on a third), are issues that must be addressed as robots find applications in society at large. A starting point for such a discussion might be the often quoted Three Laws, which promote robots as watchful servants to mankind [Asimov, 1950].

THE THREE LAWS OF ROBOTICS

1- A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3- A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

To these laws the author proposes a fourth:
4- Until robots have the capability of understanding the First, Second, and Third Laws, they should not be expected nor trusted to perform unsupervised activities that may harm a human.

Reasonableness Assumption

This fourth law points to a guiding assumption that clarifies what can be reasonably expected from a robotic system. A "reasonableness assumption" is made such that a robot should not be expected to do anything more than a human possessing comparable sensory,
motor, and logic abilities. Although great strides are being made in sensor guided autonomous robots [Pin et. al., 1988], today's inexpensive, non-industrial robots, typified by the model used in this thesis, are hardly beyond the level of a deaf, dumb, blind, one-armed moron, and hence can be expected to perform only the most basic of tasks, such as pick and place manipulations, and general plane navigation. Some provision must always be made for supervisory or manual operation modes. The reasonableness assumption must also be extended to the world model, which should not be expected to handle all possible real world phenomena, but merely have the structure and detail required for its handicapped inhabitants to perform useful tasks. Lastly, the reasonableness assumption must be applied to the real world environment. Until low cost sensory systems with functionality equivalent to the human senses are available, the robot's workplace must be kept relatively structured, i.e., with objects in known locations.

Utility

Given the limited abilities of affordable robots, the question arises as to what serious use they can have, beyond their entertainment and educational value. As was pointed out by one of the founders of industrial robotics, [Engleberger, 1965], economics drives robotic development, and will be a key factor as robots evolve into more familiar environments. Robots are finding their way into popular culture in many ways, to say the least through the entertainment industry in the form of toys (e.g. Transformers), cartoons (i.e. weekly half-hour commercials for the former), and films (e.g. *The Forbidden Planet*, *The Day the Earth Stood Still*, *Lost In Space*, *Star Trek*). In developing the concept of world modelling and applying Artificial Intelligence techniques, this thesis attempts to make many of the technicalities of commanding a robot transparent to the user, laying a
framework of a system that can help a significant portion of the population lead more fruitful, independent lives. Although anyone can use the system, the severely handicapped user was targeted as one who would receive the greatest marginal benefits, since the system can eventually provide the means to perform simple tasks, albeit slowly, that might not otherwise be possible at all. As clinical studies have shown [Engelhardt 88], the handicapped represent an untapped pool of well motivated workers, and society's burden of health care costs can be reduced by increasing their personal autonomy [Leifer, 1986].

The Future

Beyond the need for a system to help the handicapped is the need for a system having popular appeal and acceptance, somewhat analogous to the Model T automobile, that can serve as a economic basis for the further innovation. As stated by researchers at M.I.T., [Flynn & Brooks, 1988], an important goal of current robotics research is to make mass produced, low cost robotic systems available to the public, regardless of limitations in their intelligence or utility. It is hoped that the work presented here is a step in such a direction.
Chapter 2: World Modelling

In this chapter the concept of world modelling is developed. Real world objects, including the robot, are modelled within an object hierarchy. Individual objects are combined to form a coherent picture of a real, though simplified, world. A variety of interactive windows and other software tools are developed to facilitate model creation.

The Need for a Model

Without digressing into human psychology, it can be said that people in some sense carry around a representation or model of the real world, gained through sensing, reasoning, and experience, that allows them to function in the real world; to plan, and to predict, and to act [Fischler & Firchein, 1987]. Physical laws impose constraints on the real world: where gravity exists, objects tend to fall; two objects cannot occupy the same point in space at the same time. Human's understanding of such physical laws allow them to predict fairly surely the outcome of some actions, such as flipping a light switch, while other actions, such as rolling a pair of dice, cannot be predicted so surely.

In a similar light, a computer model of the environment is created to facilitate the user's interaction with the robot, enabling the robot to solve problems and plan actions. The object-oriented model provides a common conceptual groundwork for the user and robot. The model is a data base for describing relevant characteristics (size, location) of individual objects, as well as the relationships between objects. Certainly, although physical laws such as gravity can be incorporated into the model, the existence of unpredictable actions and actors in the real world will always preclude the creation of a perfect model. A question opened by this thesis is whether a model can be made "good enough" for commanding a robot to do useful work in particular environments.
The Frame Problem

As described by numerous researchers in the field of Artificial Intelligence, a major hurdle in developing a realistic world model is the so-called "frame problem" [Pylyshyn, 1987]. Succinctly, the frame problem is the problem of modelling change, of finding a representational form permitting a complex, changing world to be efficiently and adequately represented [Janlert, 1987]. Incorporating the concept of change into a model requires some form of updating of the information in the model. Owing to interactions between objects in a complex world, one particular change can produce a series of consequential changes. Serious questions arise as to how a representation can effectively keep the model in correspondence with the real world, given unexpected or unpredictable consequences. "Efficiently" implies that the model must be tractable, that changes to one small part of the model need not result in a complete updating of entire model. Any practical model must run on reasonably sized computer. "Adequately" implies the model must have sufficient scope and detail to describe a situation or task in terms the robot and user can understand and act upon. Once a model containing a rich description of the world exists, problems arise for an autonomous robot in deciding what portion of the available information is relevant to solving a particular problem.

The frame problem and related issues have some bearing on the modelling undertaken in this project, but do not present an overwhelming hurdle. First, the nature of the robot's proposed work environment is simple, structured, and not subject to rapid change. In a handicapped user's home, the robot would be one of only a few active agents, and a reasonable expectation is that other active agents do not move furniture or other objects of interest to the user. Secondly, the robot is not completely autonomous; the system will be operated in supervisory or teleoperator modes, with the human operator available for help in making decisions. Much like Heisenberg's Uncertainty Principle in physics places
limits on the information available from atomic particles, the frame problem in AI indicates that there will always be uncertainties and imperfections in a world model. Hence, a major concern in the ongoing design of the robot control system is coping with uncertainty in performing tasks. This thesis places emphasis on using feedback available from human and machine senses. The user and robot interact in maintaining an accurate world model. Notably, the object-oriented approach does provide an efficient representational form, one that can be refined incrementally without requiring the entire underlying data base to be adjusted.

Model Accuracy

As the model serves as a data base from which all parameters involved in commanding the robot are extracted, accuracy of the data has a large effect on the success of robot operations. The larger the uncertainty in size and location of modeled objects, the greater the need for sensory feedback in performing an operation. A rule of thumb for specifying dimensions is to use "tape measure" (versus "micrometer") accuracy, which means that size and position dimensions of a modeled object should be accurate to within 1/4 inch, and orientation to within five degrees.

World Modeller - Create Mode

Two main dialogs, the World Modeller and the Primitive Object Dictionary, are used in creating a working model of the robot and its environment. Operating in a "create" mode, the World Modeller initially presents the user with a blank window, representing a top-down view of the environment. The Dictionary presents a menu of primitive, predefined household objects. The user selects an item from the menu, giving the item its own name. The item is then instantiated as a new object within the World Modeller, appearing in as a
graphical outline in the World Modeller window (Fig. 1). The object can then be resized, rotated, and translated to any point in the window under mouse action, much as is done in creating a MacDraft or MacDraw document. One by one, items are created (instantiated) and positioned in the model, reflecting the real-world geometry of the robot's environment. While at present the user is required to input the model directly, model creation is ideally a shared effort between the human and the robot.

Figure 1: Instantiation of an object;
1) selection from dictionary of primitives
2) assignment of object name
3) graphic representation as dialog item.
Region Dialog Item

A dialog item is an object within the Lisp hierarchy (Fig. 35, Appendix 2), which in addition to having an internally represented structure, has a graphical form and exhibits a particular behavior in response to manipulation via the mouse and other user inputs. In order to provide a representational form that not only describes the physical characteristics of real world objects, but also their general behavior, a special archetypical object known as a region dialog item was developed.

A region dialog item is given various real continuous characteristics (e.g., size, position, orientation) by virtue of its being a representation of a real world object, and analogous discrete characteristics (e.g., scaled size, screen coordinates) by virtue of its being a graphical entity displayed on the computer's screen. The region dialog item is supplied with many functions for converting between real world and screen quantities. As its name implies, a region dialog item has the graphic form of a region, an arbitrarily shaped area of the computer's graphics plane. In the top-down view provided by the World Modeller, the region corresponds to the top-down projection of the modelled object.

A region is generated by issuing a sequence of point-to-point "draw" commands within a window. Such commands draw lines on the computer's screen, and together form the outline of the region. Although in principle any collection of simple closed curves defines a region, current software for the World Modeller generates rectangular or regular dodecahedral shaped regions by default. Coordinates representing the corner points of a region are stored in polar form to facilitate rotational transformations.

As a region dialog item is to be a scale model of a real (although generic) object, the user must be able manipulate the item so as to specify its size (width, depth, and height dimensions), position (relative x, y, and z coordinates) and orientation (relative pitch,
yaw, and roll angles). Every region dialog item has a fixed internal coordinate system, whose origin corresponds to the geometric center or centroid of the object (Fig. 2). The system's x- and y-axes are aligned along the width and depth dimensions, respectively. The distance between coordinate system origins defines the relative position of one object with respect to another. Relative orientation of objects is defined by the relative orientation of their coordinate systems. "Standard" position and orientation are defined such that the object appears in the upper left corner of the World Modeller window, with the object's x-axis aligned with the window's x-axis, and the object's y-axis aligned opposite to the window's y-axis.

Five active sites are defined within the region dialog item: four control points and the remaining interior area. The user "selects" the region dialog item by clicking anywhere within its interior. Selecting the region dialog item activates it, causing its control points to be displayed. Any previously selected dialog item is de-selected. By clicking on the rotation control point and dragging the mouse to the left or right on the screen, the user rotates the region clockwise or counterclockwise about its z-axis. Clicking and right/left dragging (with respect to the screen) of the x-stretch control point increases/decreases the width (x-dimension) of the region. Clicking and up/down dragging of the y-stretch control point increases/decreases the depth (y-dimension) of the region. Clicking and diagonal dragging of the x-y-stretch control point proportionally changes the width and depth dimensions of the region. The position of the object is changed by clicking and dragging the interior area. The directionality of drag operations with respect to the screen remains the same, regardless of the orientation of the object. Associated with each region dialog item is a set of sliders and an Object Information Dialog (explained later in this chapter), for specifying the height dimension, pitch angle, and roll angle. As a safety
feature, a region dialog item can be "locked", meaning it can be prevented from responding to mouse activity.

A region dialog item also exhibits relational behavior, i.e., its location in space can be influenced by other region dialog items. In the real world, when an object $a$ is placed on top of an object $b$, $a$ moves with $b$. Disregarding the intricacies of friction, $b$ does not usually move with $a$, however. This type of asymmetric relational behavior is instilled in region dialog items using relational hierarchy characteristics, also referred to as "ownership" (Fig 3.). Every object represented by a region dialog item has an owner or "parent". When the parent object moves, the child (in fact, the entire hierarchy of children) moves with it in an absolute sense. In a relative sense, the parent and child act as if they were rigidly fixed to each other. When the child alone is moved, the parent remains fixed.
in an absolute sense. A child's position and orientation are normally expressed with respect to the axis system of its parent, a fact used extensively in path planning (Chapter 5).

Although an object cannot have more than one owner, it can own any number of other objects, as long as the ownership is not circular (i.e., an object cannot own itself). Ownership is assigned implicitly by virtue of the user placing the child on top of or inside of the parent, such that the origin of the child lies anywhere within the region of the parent. The World Modeller enforces an ordering of dialog items to prevent ambiguities in ownership assignment (i.e., when an object could be the child of more than one parent) as more and more objects are added to the model. Using a characteristic of "openness", the region dialog item has itself a mechanism for overriding the implicit assignment of children. The region dialog item must be "open" for children to be assigned to it.

Recursive updating procedures are used to maintain the region dialog item's state when changes occur. Recursion allows selective updating of branches of the World Modeller's data base, by propagating changes down into the relational hierarchy of objects. For instance, in order for the World Modeller to display a particular object as the user moves it in the window, the screen coordinates of the object are continuously recalculated and the object continuously redisplayed. Once the user has finished moving the object, the screen coordinates of all children of the object are recalculated, and the children are redisplayed in their proper relative positions.
Zoom Region Dialog Item

The zoom region dialog item is a derivative of the region dialog item, adapted to serve as an absolute base of reference for all other region dialog items in the model. When a new object (and hence a new region dialog item) is instantiated, it is owned by the zoom region. Although its ownership may be reassigned, the object will be either directly or indirectly owned by the zoom region. The zoom region inherits the region dialog item's recursive updating procedures, meaning that all owned objects (i.e., the entire modelled world) will move when the zoom region moves. This feature of the zoom region provides a convenient mechanism for displaying different views of the modelled world.
By scaling and spatially transforming the zoom region, operations such as zooming in and out, zooming in on individual objects, and scrolling vertically and horizontally can be performed. The zoom region cannot be directly manipulated by the user with the mouse; all operations are activated by the user's selection from within the View Menu (explained later in this chapter). The zoom region dialog item also has special functional forms that reflect its position as the the end of all backward recursion up the hierarchy of region dialog items.

Specialized Region Dialog Items

The region dialog item provides a basic structure and functionality that can be augmented to create specialized region dialog items that more closely model the form and behavior of real objects. For instance, a door can be modelled as a long, slender rectangular region that pivots about a corner point when manipulated by the mouse. In addition to having all the characteristics of a region dialog item, the door has a "handedness" and a measure of being "open". Further, specialized region dialog items can be combined together to form more complicated objects, similar to what is done in Constructive Solid Geometry. Objects such as refrigerators and microwaves can be constructed from a simple box and a door (Fig. 4).
Modelling lamps and similar household light sources is of particular interest, since light sources are used to guide the robot through the environment (Chapter 4). The region dialog item representing the lamp is given a variable indicating the lamp's state, which can be either on or off. It is also given two identification variables, corresponding to the "unit" and "house" codes used by the X-10 computer interface to address individual modules. In its unlocked state, the dialog item acts no differently than the generic region dialog item. When locked and clicked on, the dialog item asks the X-10 Dialog to toggle the state of the real lamp, i.e., turn it on if it is off, or turn it off if it is on.
Robot Modelling

Robots are in general complicated objects, composed of many connected, moving parts. In order to completely describe a robot, one would have to represent its kinematic design, its resulting dynamic properties and control requirements, and its sensing abilities. Kinematics is the study of the geometry of motion, the interrelation between positions, velocities, and accelerations of components of a mechanism. In robotics, kinematics describes the relationship between the position, velocity, and acceleration of the base and arm joints to the position, velocity, and acceleration of the gripper. Forward kinematics describes the unique transformation from joint coordinates (the set of values of the arm joints) to world coordinates (the Cartesian coordinates and orientation of the gripper). In more familiar terms, forward kinematics answers the question: "If the size of all the robot's structural components and the values of its joint angles are known, where is the gripper in space?". Inverse kinematics describes the non-unique transformation from world coordinates to joint coordinates, answering the question: "If the gripper is to be positioned at a particular point in the robot's work space, how must the joints of the robot be moved?". Dynamics is the study of the relation between the forces and torques applied to a body, and the resulting motion. In robotics, dynamics describes the relation between forces generated by actuators and the resulting motion of the arm and gripper. Control is the study of the relation between the inputs applied to a physical system, and the resulting outputs. Given a kinematic and dynamic description of a robot, a robotic controller adjusts the inputs to actuators (voltages, currents, pressures) in order to move the arm and gripper in a prescribed manner.

From the point of view of modelling, a complete description of the robot is impractical and unnecessary. Dynamics and control deal with time dependent quantities. Since the main focus of the World Modeller is not its real-time representation of the world, these
two aspects of the robot are not important. As long as procedures exist for polling sensors, detailed knowledge of the operation of the sensors is likewise not needed. Hence, the model can be simplified by including only a kinematic description of the robot. Forward kinematics of a robot can be concisely written in terms of Denavit-Hartenberg (D-H) parameters and homogeneous transformations [Denavit & Hartenberg, 1955]. The HERO 2000 arm is basically planar, so that forward kinematics can be as easily derived trigonometrically (Figs. 5-6, Equs. 1-6). Inverse kinematics relations are typically derived by manipulation of the forward kinematics, both in the case of D-H and trigonometric representations (Equs 5-10). Forward and inverse kinematic equations are incorporated into the Robot Controller object, as described in Chapter 3.
Figure 5: Kinematic quantities of the modified HERO 2000;
a) link lengths, b) joint angles.
Forward Kinematic Equations:

\[
gripper_x = x_{\text{offset}} + \text{arm\_length} \sin(\text{arm\_angle}) + \text{elbow\_length} \sin(\text{elbow\_angle}) + \text{gripper\_length} \sin(\text{pitch})
\]

\[
gripper_z = z_{\text{offset}} + \text{arm\_length} \cos(\text{arm\_angle}) + \text{elbow\_length} \cos(\text{elbow\_angle}) + \text{gripper\_length} \cos(\text{pitch})
\]

\[
gripper_y = \text{gripper\_x} \sin(\text{torso\_angle})
\]

Inverse Kinematic Equations (assuming prescribed gripper orientation):

\[
wrist_x = \text{gripper\_x} - \text{gripper\_length} \sin(\text{pitch})
\]

\[
wrist_z = \text{gripper\_z} - \text{gripper\_length} \cos(\text{pitch})
\]

\[
delta_x = \text{wrist\_x} - x_{\text{offset}}
\]
\[ \text{delta}_z = z_{\text{offset}} - \text{wrist}_z \]

\[ a = \text{arm}_\text{length} \]

\[ b = \text{elbow}_\text{length} \]

\[ c = \sqrt{\text{delta}_x \times \text{delta}_x + \text{delta}_z \times \text{delta}_z} \]

\[ g1 = a + b + c \]

\[ g2 = \sqrt{(g1 - a) \times (g1 - b) \times (g1 - c) / g1} \]

\[ \text{angle}_1 = 2 \times \arctan\left(\frac{g2}{(g1 - b)}\right) \]

\[ \text{angle}_2 = 2 \times \arctan\left(\frac{g2}{(g2 - c)}\right) \]

\[ \text{angle}_3 = 2 \times \arctan\left(\frac{\text{delta}_z}{\text{delta}_x}\right) \]

\[ \text{arm}_\text{angle} = \frac{\pi}{2} - \text{angle}_1 - \text{angle}_3 \]

\[ \text{elbow}_\text{angle} = \frac{\pi}{2} - \text{angle}_1 + \text{angle}_2 - \text{angle}_3 \]

\[ \text{torso}_\text{angle} = -\arctan\left(\frac{\text{gripper}_y}{\text{gripper}_x}\right) \]

From the user's point of view, the most important features of the robot are its gripper and its base, since task execution ultimately involves the robot moving about and gripping things. A specialized region dialog item incorporating three region dialog items, corresponding to the base, torso, and gripper, was therefore created (Fig. 7). Relationships between the three items were arranged to reflect the kinematic constraints of the robot. Aside from its ability to rotate about its geometric center, the torso is fixed with respect to the base and moves with it. Since the gripper has no yaw ability, meaning it is constrained to have the same yaw orientation as the torso, the torso and gripper are made to rotate together as the gripper is moved. The gripper is free to move in and out with respect to the torso, within the constraints of its workspace. Given the kinematic configuration of the robot, the workspace of the robot's arm can be calculated and displayed relative to the robot. Dialog activity with the gripper is constrained to lie within
the workspace, avoiding the problem of trying to make the arm reach outside of its workspace.

![Diagram](image)

Figure 7: Specialized region dialog item for robot; a) components, b) horizontal workspace (shaded).

**Alternate Input Methods**

While direct manipulation of a region dialog item using the mouse is a quick and easy way of adjusting the object's size, position, and relation to other objects, real dimensions generated by scaling up from the computer's screen are subject to round-off errors. Manipulation becomes more difficult as the scale of the model increases, as the screen size of modelled objects decreases. Accuracy is limited by the resolution of the computer's screen (i.e. size of the pixels) combined with the overall scale of the model. Accuracy and redundancy are added to the system by providing alternate means of entering data into the model.
Text Input Dialog

The most common way of entering data into the computer is using the keyboard. The computer continuously monitors the keyboard, which transmits character codes corresponding to the keys a user has manually pressed. In the case of a severely physically handicapped person, manually pressing keys may not be possible. A software keyboard called the text input dialog was therefore created, which responds to both mouse and hardware keyboard inputs (Fig. 8). Within the dialog, button dialog items duplicate the function of hardware keys, generating characters when clicked on by the mouse. Input characters are displayed above the top row of buttons. The entire string of input characters is returned when the user clicks on the return button or hits the return key. The text input dialog is used to prompt the user for input, such as when a new object is to be added to the world model, or when an old object is to be changed. It should be noted that the arrangement of buttons within the text input dialog is arbitrary, and could be reprogrammed for better input efficiency.
In combination with the text input dialog, the object information dialog allows the user to change information about an object, including its name, size, position, orientation, shape, and owner. The dialog contains static text dialog items corresponding to each property (Fig. 9). When clicked on by the user, the static text dialog item acts to cause the text input dialog to pop up, prompting the user to enter a new value for the selected property. The text input dialog passes the new value to the object, which performs appropriate update procedures.
Sliders and Spinners

The fact that region dialog items can be quickly instantiated, resized, and repositioned demonstrates an underlying property of the interactive graphics interface: a user can very easily generate numbers with the interface. Individual graphics windows as well as the entire computer screen can be considered 2-dimensional digitizing tablets. The mouse thus acts as a digitizing pointer, returning its x-y screen coordinates when polled within a
procedure. In addition to the region dialog item, customized **slider** and the **spinner** dialog items take advantage of this digitizing property. The slider and spinner act as linear and rotary digital potentiometers, respectively (Fig. 10).

![Figure 10: a) slider dialog item, b) spinner dialog item.](image)

For both dialog items, three active regions are defined: two for setting the output range, and the third for specifying the output value. The dialog items are divided into upper numeric and a lower graphical display areas. The number at the right of the numeric display indicates the output range (base value), while the number on the left is the actual output value. By positioning the mouse in the right half of the numeric display area and holding down the button, the user increases the output range of the slider or spinner up to a predefined upper limit. Similar activity in the left half of the display area decreases the output range down to a lower limit. At the time of the object's instantiation, values are set for the upper and lower limits of output range, as well as the rate of incrementing/decrementing. When the mouse is dragged across the graphical display area, a darkened area "follows" the mouse, filling a fraction of the graphical display area. This fraction is multiplied by the base value to produce the numeric value of the slider output. Sliders are used to a small extent in creating objects within in the World Modeller;
whenever an object is selected, sliders for adjusting the object's height and pitch/roll orientation appear in the upper right corner of the window. The primary use of sliders is in the Teleoperator Dialog (Chapter 3).

View Menu

As a model becomes more complicated, the need may arise to view particular portions. For instance, if a large building is being modelled and appears in its entirety on the screen, the scale may be such that smaller objects such as desks and chairs will be barely visible. Manipulation of smaller objects using the mouse becomes impossible. Accordingly, a selection of zooming and scrolling options is made available using a view menu (Fig. 11).

<table>
<thead>
<tr>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom In 4x</td>
</tr>
<tr>
<td>Zoom In 2x</td>
</tr>
<tr>
<td>Zoom Out 4x</td>
</tr>
<tr>
<td>Zoom Out 2x</td>
</tr>
<tr>
<td>Zoom Object</td>
</tr>
<tr>
<td>Scroll Up</td>
</tr>
<tr>
<td>Scroll Down</td>
</tr>
<tr>
<td>Scroll Left</td>
</tr>
<tr>
<td>Scroll Right</td>
</tr>
<tr>
<td>Set Scale</td>
</tr>
<tr>
<td>Home View</td>
</tr>
<tr>
<td>Refresh</td>
</tr>
</tbody>
</table>

Figure 11: View Menu

As described earlier, the zoom region dialog item provides the base reference frame for all other region dialog items. The zoom region is something of a "ghost" dialog item, having no interactive behavior and no graphic form. Somewhat arbitrarily, the zoom
region is made twice as wide and twice as high as the World Modeller window, and in its "Home View" is placed such that its geometric center (and hence origin) is at the lower left corner of the window (Fig. 12). The region provides a right handed Cartesian system, versus the left handed systems of the screen and window.

Zooming out by factors of 2 or 4 is accomplished by increasing the overall scale of the World Modeller window by the same factors, effectively shrinking the zoom region and its contents symmetrically about the zoom region's origin. When zooming in by factors of 2 or 4, the cursor takes the form of a rectangle representing the area that will be enlarged to fill the window. The overall scale is decreased by a factor of 2 or 4, and the zoom region is translated such that the center of the zoomed area will be in the center of the window. When zooming an object, the overall scale is set so that the object fill roughly sixty percent of the window, and the zoom region is translated such that the center of the zoomed object will be in the center of the window. To scroll the display, the zoom region is simply translated by a fixed amount (10 pixels by default) in the desired direction. The "Set Scale" option allows the user to set the overall scale of the window to any value. The scale is the ratio of real world units to pixels, e.g., 3 inches per pixel. The "Refresh" option redraws the entire display, since parts of the display will be erased when the used moves objects around.
A variety of miscellaneous features make the world modelling software easier to use. The menubar presents the user with menus of related features (Fig. 13). Under the File menu, the user may save a new model to disk, retrieve an old model from disk, or delete the current model within the World Modeller Dialog. The Mode menu allows the user to switch the World Modeller between "create" or "command" modes. In create mode, the model is detached from reality, so that the user can move objects at will. In command mode, the model is taken to be an accurate representation of the world, with objects being moved only through the actions of the robot. The Object menu is used to address individual objects within the model, to retrieve and change object variables. If no object is currently selected, the Select menu item activates the Select window, which presents the user an alphabetized list of all objects in the model. Selection of an object in the list causes
the object to be selected within the World Modeller window. A selected object may be deleted from the model, taking with it all objects that it directly or indirectly owns. The selected object can be locked/unlocked and opened/unopened from the Object menu as well. Selection of the Info Dialog menu item activates the Object Information Dialog, previously described. Within the Show Me menu, the Object Coords option can be made active, causing the World Modeller to display a selected object's relative position and orientation as it is being manipulated. The Path Planner option graphically displays the progress of path planning routines (Chapter 5).

Figure 13: Schematic of Menubar with File, Mode, Object, and Show Me menus
Chapter 3: Robot Command Generation

In this chapter, forward and inverse kinematic command modes are described. Robot commands are generated by cursor actions within dialogs, which serve as a medium of communication between the user and the robot control software.

Command Dialogs

Three main dialogs are responsible for generating commands for the robot. The Teleoperator Dialog generates forward kinematic commands, i.e., commands that move individual joints of the robot directly. The World Modeller Dialog operates in two modes: Create and Command, which are distinguished to the user by the style of cursor (Fig. 14). As previously described, Create mode is used to develop a working model. New objects, including an object representing the robot, are created (instantiated) and positioned in the model, reflecting the real-world geometry of the robot's environment. Once a detailed enough model exists (and has been saved on disk), the Command mode is entered, and dialog activity with the robot object is translated into inverse kinematic commands for the robot, i.e., commands that position the gripper and base at desired points in space. This feature of the World Modeller Dialog is termed model-reflective command generation, since activity of the model robot is reflected in activity of the real robot. A third dialog, the Robot Controller, maintains information pertinent the the operation of a particular type of robot, such as kinematic configuration, joint limits, and command syntax. Supplied with a desired incremental amount of joint movement, the Robot Controller constructs and returns a valid BASIC command.
Teleoperator Dialog

Figure 15 shows the layout the Teleoperator Dialog, in which rows of buttons and sliders are arranged as in a control panel. A combination of start button, stop button, and slider is assigned to each joint of the robot. The start button displays the name and direction of the joint it controls, while the stop button simply displays "stop". The slider indicates the desired amount of joint movement. For base forward/backward movement the amount is in inches. For base spin ccw/ spin cw and all arm joints, the amount is in degrees. Speed, grip, and force sliders represent relative values.

Two command modes are available: incremental (Incr.) and continuous (Cont.). In incremental mode, each time the start button is clicked, a command is issued to the robot for moving the joint incrementally by the amount indicated by the slider. In continuous mode, a click on the start button results in the joint moving until either the stop button is clicked or the joint reaches the end of its range. The continuous mode can be seen as a derivative of the incremental mode, in which the increment of movement is as large as kinematically possible. Hence, the difference between incremental and continuous mode is a matter of degree. In fact, the upper range limit of a slider is set to the kinematic limit of its joint, such that the incremental mode can be made to operate exactly like the continuous mode. The distinction exists as a matter of facilitating manipulation;
incremental mode is useful in making slow, fine motions or in training an operator, while continuous mode allows faster actions. Continuous mode suffers from overshoot due to transmission delay, which is roughly 2 seconds when using the Remote Console, and a fraction of a second with the wireless modem.

Buttons for miscellaneous functions are also provided. The "home" button issues the BASIC "home" command, causing the robot to run its built in homing procedure. A "teach" facility is implemented using the combination of "record" and "replay" buttons, as described in the next section. Buttons for controlling other specialized functions can be added as space on the control panel allows. Positioned below the Teleoperator Dialog is the Robot Controller Dialog, which displays the state of the robot's joints as well as a confirmation of a successful command transmission.
Teach Facilities

When operated with the Remote Console, the HERO can be placed in a teach mode, in which positions of the robot are recorded within a BASIC program onboard the robot. When the program is subsequently run, the robot moves through the recorded positions. Similar teach facilities were implemented in earlier versions of the control software, when all programming was in BASIC (Appendix 2). In normal operation, the Teleoperator Dialog or Master is responsible for generating one- and two-digit codes that are interpreted
as commands by the Slave routine on the robot. The Slave continuously polls its serial port for input from the Master, waiting idle much of the time. The Master keeps a record of the state of the robot, i.e., the values of all its joint angles.

A "teach" operation is triggered with each click of the "record" button, during which the Master augments the Slave program. By sending a control code (i.e., ^C), the Master halts regular program execution of the Slave, placing the robot's computer in the immediate mode. The Master then transmits two program lines, which the BASIC interpreter on the robot dutifully adds to the Slave program. The first line is a command to move the arm to the recorded position and is in the form of a subroutine. The second line is a conditional inserted into the main body of the Slave, which interprets a code from the Master as a command to execute the subroutine. Following these transmissions, normal execution of the Slave is resumed.

Within such a programming scheme, a system of "primitives" and "macros" is used to build up a library of recorded robot actions. A primitive is defined as the configuration or state of the robot at a particular instant. A low level macro consists of a sequence of primitives, i.e., a sequence of robot motions between recorded configurations. A high level macro consists of a combination of primitives and lower level macros. Primitives and macros are presented to the user in the form of buttons within a Replay Dialog (Fig. 16). At the time the user clicks the "record" button within the Teleoperator Dialog, he is prompted to enter a name for the recorded position. A primitive button labelled with this user-supplied name is added to the Replay Dialog.

The user accesses the Replay Dialog by clicking on the "replay" button. Within the Replay Dialog, primitive buttons, macro buttons, and macro creation buttons are displayed. A new macro is generated by the user's first clicking the "start macro" button, then a meaningful sequence of primitive and pre-existing macro buttons, and finally the
"end macro" button. The user is prompted for a name for the new macro, at which point a macro button labelled with the name appears in the window. To implement a macro, the Master again halts execution of the Slave and inserts a program line. The line is a conditional that interprets a code from the Master as a command to execute the sequence of subroutines corresponding to the sequence of primitives composing the macro.

Figure 16: Replay Dialog.
Robot Controller Dialog

The Robot Controller Dialog is an artifact of the evolution of the control software, implemented while the Teleoperator Dialog was being ported from BASIC to object-oriented LISP, and before the task of world modelling was begun. Given that BASIC onboard the robot cannot handle error conditions, an object was needed to "filter" robot joint commands issued by the Teleoperator Dialog such that no kinematic constraints of the robot were violated. In addition, it was desirable to present the user with a display indicating the state of the robot arm. Hence was born the Robot Controller Dialog. With the advent of world modelling, the Robot Controller Dialog object is largely redundant with and can be subsumed into the class of robot dialog item objects.

World Modeller Dialog - Command Mode

As has been described previously, higher level command of the robot begins with a sufficiently detailed model of the robot and its environment (Fig. 17). With the World Modeller in Command mode, the robot can be commanded in two ways: explicitly by choice of a command from within a menu of commands, or implicitly by means of manipulating the body or gripper of the model robot. Commands for robot localization (Chapter 4) and path planning (Chapter 5) have been implemented currently. Selection of the "FIND-PATH" item allows the user to specify a destination point for the robot. The World Modeller constrains the user's choice to prevent the robot from ending up inside of another object, or outside of the modelled portion of the world. If the path planning routines can find a collision-free path to the destination, the path is displayed and the user queried as to whether appropriate motion commands should be transmitted to the real robot. The "FIND-ROOM" selection initiates a procedure for detecting the robot's presence within a room, while "TRIANGULATE -COARSE" and "TRIANGULATE -
FINE" have the robot triangulate its position and orientation within a room using coarse and fine sensor sweeps, respectively.

Figure 17: A typical model environment.

Implicit command generation for base movement follows the same procedure as the explicit "FIND-PATH" command, with the user moving the model robot directly to a prospective desired end point. If a clear straight-line path exists between the current robot position and its destination, commands for lining up and traversing the robot are transmitted. Otherwise, the path planner is invoked.
In commanding the gripper position, manipulation of the model gripper is constrained to lie within the horizontal workspace of the gripper. This horizontal workspace is a function of the vertical position of the gripper, and is displayed as a shaded area relative to the model robot (Fig. 18). Desired vertical position is specified by means of a slider that appears when the gripper is selected. Inequality conditions (7) - (11) combined with Table 1 describe the functional dependence between gripper height and horizontal workspace. Once the user has specified a new desired gripper end point, required joint angles are calculated and transmitted to the robot.

![Figure 18: Horizontal workspace of the robot gripper for a particular gripper height.](image_url)

**Inequality Conditions:**

\[
\begin{align*}
  z_{\text{offset}} + \text{arm}_{\text{length}} + \text{elbow}_{\text{length}} + \text{gripper}_{\text{length}} &\geq \text{gripper}\_z, \text{ and} \\
  z_{\text{offset}} + \text{arm}_{\text{length}} + (\text{elbow}_{\text{length}} + \text{gripper}_{\text{length}}) \times \cos(154.4) &\leq \text{gripper}\_z \\
\end{align*}
\] (7)
\[ \text{z\_offset + arm\_length + (elbow\_length + gripper\_length) } \times \cos(154.4) \geq \text{gripper\_z}, \text{ and} \]

\[ \text{z\_offset + arm\_length } \times \cos(25.6) - \text{elbow\_length - gripper\_length} \leq \text{gripper\_z} \quad (8) \]

\[ \text{z\_offset + arm\_length } \times \cos(25.6) - \text{elbow\_length - gripper\_length} \geq \text{gripper\_z}, \text{ and} \]

\[ \text{z\_offset + arm\_length } \times \cos(120) - \text{elbow\_length - gripper\_length} \leq \text{gripper\_z} \quad (9) \]

\[ \text{z\_offset + arm\_length + elbow\_length + gripper\_length} \geq \text{gripper\_z}, \text{ and} \]

\[ \text{z\_offset + (arm\_length + elbow\_length + gripper\_length) } \times \cos(120) \leq \text{gripper\_z} \quad (10) \]

\[ \text{z\_offset + (arm\_length + elbow\_length + gripper\_length) } \times \cos(120) \geq \text{gripper\_z}, \text{ and} \]

\[ \text{z\_offset - arm\_length - elbow\_length - gripper\_length} \leq \text{gripper\_z} \quad (11) \]

\[ \begin{array}{|c|c|}
\hline
\text{condition} & \text{r\_min} - \text{x\_offset} \\
\hline
(7) & \text{elbow\_length + gripper\_length} \times \sin(\text{elbow\_angle}) \\
\hline
(8) & \text{arm\_length} \times \sin(\text{arm\_angle}) \\
& + \text{elbow\_length + gripper\_length} \times \sin(154.4 + \text{arm\_angle}) \\
\hline
(9) & \text{arm\_length} \times \sin(\text{arm\_angle}) \\
\hline
(10) & \text{arm\_length + elbow\_length + gripper\_length} \times \sin(\text{arm\_angle}) \\
\hline
(11) & \text{arm\_length} \times \sin(120) \\
& + \text{elbow\_length + gripper\_length} \times \sin(\text{elbow\_angle}) \\
\hline
\end{array} \]
Chapter 4: Robot Localization

This chapter discusses methods of determining the robot's position and orientation within a room, using feedback available from onboard sensors. Localization via triangulation on light sources is discussed in depth.

Localization refers to the ability of a robot to determine its location within the environment. On the scale of a house or office, localization involves the robot's establishing its mere presence within a particular room, and further calculating its position and orientation with respect to room coordinates. While the user can manually update the robot's location when visual inspection is possible, a variety of alternative methods are available using the robot's sonar and light sensors. The robot can detect walls and light sources, meaning it can use either room corners, a combination of a wall and a light source, or group of three or more lights to establish its position and orientation. The built in auto-docking routine for charging the robot can serve for localization as well.

Triangulation on three controllable light sources is the simplest of the above methods, since the robot remains stationary. In the most general case, the robot exists in an environment with both natural (windows, skylights) and artificial (floor lamps, ceiling lights) light sources that combine to produce the ambient lighting. A particular controllable light source is not necessarily the brightest source in the robot's field of view. Hence, in making the robot detect a controllable light source with its head-mounted light sensor, the background ambient light levels must be filtered out of the readings. This is accomplished by having the robot take a series of three reading sweeps. A standard reading sweep consists of an ordered set of 24 values gained by polling the light sensor at 15 degree increments as the reflector travels in a 360 degree arc (Fig. 19a). The first and third sweeps are with the light source turned on (or off) and the second with the light source
turned off (or on). The three sets of values are reduced to one set by adding corresponding values of the first and third sweeps together, and subtracting twice the values of the second sweep. The position of the largest value in the reduced set marks the bearing of the controlled light source with respect to the robot. Once bearings corresponding to three independent light sources have been taken, the difference between any two robot-referenced bearings gives an absolute angle between sources (Fig. 19b).

Given the availability of a minimum of three known reference points (light sources) and observed bearings of these references, the robot's position is calculated from trigonometry. Two approaches have been taken, the first involving triangular constructions [Reynolds, 1928], and the second a circular construction [Jarvis and Byrne, 1988]. In special cases in where one approach fails, the other can be used.

In the triangular construction, four possible solutions to the equations describing the robot's position exist. Depending on the geometry of the real world, some of the
solutions will not correspond to physical reality. Hence, four sets of computations are carried out (Equs. 12-23), and the solutions checked for validity against real world constraints. In performing calculations, the three reference points are first ordered into a right-handed axis system, generally non-orthogonal, with the origin assigned to the reference point whose corresponding angle is the greatest (Figs. 20a-20d). This approach fails due to the indeterminacy of the equations when the robot lies on the circumference of the circle that passes through the three reference points.

Figure 20: a) first quadrant geometry, b) second quadrant geometry.
Figure 20: c) third quadrant geometry, d) fourth quadrant geometry.
Triangulation equations, triangular construction:

Auxiliary angle:

\[
\text{aux} = \text{atan2}((y_2-y_1), (x_2-x_1))
\]

First Quadrant:

\[
a_2 = \text{atan}\left(\frac{-\sin(a_1) + (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 + a_9)}{\cos(a_1) + (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 + a_9)}\right)
\]

\[
x_4 = x_2 + 12\cos(a_2 + \text{aux})
\]

\[
y_4 = x_2 + 12\sin(a_2 + \text{aux})
\]

Second Quadrant:

\[
a_2 = \text{atan}\left(\frac{-\sin(a_1) + 05/14)(\sin(a_1)/\sin(a_4))\sin(a_4 - a_9)}{\cos(a_1) - (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 - a_9)}\right)
\]

\[
x_4 = x_2 + 12\cos(a_2 + \text{aux} - \pi/2)
\]

\[
y_4 = x_2 + 12\sin(a_2 + \text{aux} - \pi/2)
\]

Third Quadrant:

\[
a_2 = \text{atan}\left(\frac{-\sin(a_1) + (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 - a_9)}{\cos(a_1) + (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 - a_9)}\right)
\]

\[
x_4 = x_2 + 12\cos(a_2 + \text{aux} - \pi/2)
\]

\[
y_4 = x_2 + 12\sin(a_2 + \text{aux} - \pi/2)
\]

Fourth Quadrant:

\[
a_2 = \text{atan}\left(\frac{-\sin(a_1) + (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 + a_9)}{\cos(a_1) - (15/14)(\sin(a_1)/\sin(a_4))\sin(a_4 + a_9)}\right)
\]

\[
x_4 = x_2 + 12\cos(-a_2 + \text{aux} + \pi/2)
\]

\[
y_4 = x_2 + 12\sin(-a_2 + \text{aux} + \pi/2)
\]
In the circular construction, the bearing between two sources defines a locus of possible robot positions, lying on a circular arc (Fig. 21a). The radius and center of this arc are calculated from trigonometry (Fig. 21b). Ideally, the robot's location is found as the common point of intersection between three overlapping circular arcs (Fig. 22). As a result of inaccuracies in data, calculations will likely produce not a single point, but a set of points lying in a neighborhood about the robot. Also, since there are in general two points of intersection between any two overlapping arcs, calculations will produce extraneous intersection points. The robot's position is estimated by averaging those points that lie in a close neighborhood, and disregarding the extraneous points. This method fails when the center points of two or more arcs coincide, and when the radius of an arc is infinite.
Figure 21: a) Locus of possible robot locations,

b) Arc radius and center from bearing.

\[
d = \left( (xs1 - xs2)^2 + (ys1 - ys2)^2 \right)^{1/2}
\]
\[
r = d/2 / \sin(b)
\]
\[
a = \text{atan2}(ys2 - ys1), (xs2 - xs1)
\]
\[
xc = xs1 + r * \cos(la) - (90 - b)
\]
\[
yc = ys1 + r * \sin(la) - (90 - b)
\]
Figure 22: Geometric construction for triangulation.

Triangulation equations, circular construction:

Auxiliary angles and parameters:

\[
\begin{align*}
  s_{12} &= \frac{(r_1 + r_2 + d_{12})}{2}; \\
  s_{13} &= \frac{(r_1 + r_3 + d_{13})}{2}; \\
  s_{23} &= \frac{(r_2 + r_3 + d_{23})}{2}; \\
  r_{12} &= \sqrt{(s_{12} - r_1) \times (s_{12} - r_2) \times (s_{12} - d_{12}) / s_{12}}
\end{align*}
\]
\[ r_{13} = \sqrt{(s_{13} - r_1) * (s_{13} - r_3) * (s_{13} - d_{13}) / s_{13}} \]
\[ r_{23} = \sqrt{(s_{23} - r_2) * (s_{23} - r_3) * (s_{23} - d_{23}) / s_{23}} \]
\[ a_{12} = 2 * \tan(\frac{r_{12}}{s_{12} - r_2}) \]
\[ a_{13} = 2 * \tan(\frac{r_{13}}{s_{13} - r_3}) \]
\[ a_{23} = 2 * \tan(\frac{r_{23}}{s_{23} - r_3}) \]
\[ t_{12} = \tan(\frac{y_2 - y_1}{x_2 - x_1}) \]
\[ t_{13} = \tan(\frac{y_3 - y_1}{x_3 - x_1}) \]
\[ t_{23} = \tan(\frac{y_3 - y_2}{x_3 - x_2}) \]

Intersection points:

**Between 1st and 2nd arcs:**
\[ x_{12a} = x_1 + \cos(t_{12} + a_{12}) \]  \hspace{1cm} (24)
\[ y_{12a} = y_1 + \sin(t_{12} + a_{12}) \]  \hspace{1cm} (25)
\[ x_{12b} = x_1 + \cos(t_{12} - a_{12}) \]  \hspace{1cm} (26)
\[ y_{12b} = y_1 + \sin(t_{12} - a_{12}) \]  \hspace{1cm} (27)

**Between 1st and 3rd arcs:**
\[ x_{13a} = x_1 + \cos(t_{13} + a_{13}) \]  \hspace{1cm} (28)
\[ y_{13a} = y_1 + \sin(t_{13} + a_{13}) \]  \hspace{1cm} (29)
\[ x_{13b} = x_1 + \cos(t_{13} - a_{13}) \]  \hspace{1cm} (30)
\[ y_{13b} = y_1 + \sin(t_{13} - a_{13}) \]  \hspace{1cm} (31)

**Between 2nd and 3rd arcs:**
\[ x_{23a} = x_2 + \cos(t_{23} + a_{23}) \]  \hspace{1cm} (32)
\[ y_{23a} = y_2 + \sin(t_{23} + a_{23}) \]  \hspace{1cm} (33)
\[ x_{23b} = x_2 + \cos(t_{23} - a_{23}) \]  \hspace{1cm} (34)
\[ y_{23b} = y_2 + \sin(t_{23} - a_{23}) \]  \hspace{1cm} (35)
Localization Accuracy

Accuracy of the localization routines depends both on the accuracy with which the positions of the light sources are known, the separation of the sources, and the ability of the robot's light sensor to resolve the bearing of the sources. Results from routines simulating the output from the robot's light sensor corresponding to lights arranged within a 12' by 12' room indicate the need for sensor resolution greater than the 15 degrees currently available. Resolution of up to 1/10th of a degree is possible by programming the reflector and robot torso together. A series of standard sweeps can be first taken to determine the 15 degree sector in which the source lies. The reflector is then fixed at one end of the sector, and as the sensor is polled, the robot's torso is rotated through the sector in increments as small as 1/10th of a degree. Assuming no error in the positions of the light sources, a simulation using the triangular construction showed that a resolution of 2 degrees was sufficient to calculate position accurate to within a radius of 3 inches, outside the neighborhood of indeterminate points. Accuracy and reliability can be enhanced by using more than three sources.
Chapter 5: Path Planning

This chapter describes how path planning for the robot is accomplished. A network of nodes is generated from the world model and used by the A* search routine for inter- and intra-room path planning. Point-to-point navigation for a robot having independent twin-opposed drive wheels is also discussed.

An essential component of a semi-autonomous mobile robot is the ability to travel through a household or office environment. Toward this end, routines have been written that operate upon the data base of information available from the World Modeller and generate collision-free paths for the robot within the static, structured environment [Regalbuto et. al., 1988]. Path planning is accomplished by generating a search space (the set of all possible solutions) from the model geometry, and applying a search algorithm to find a solution to the problem of finding the shortest safe path from the robot's initial position to a desired final position. A typical path consists of a series of straight-line segments connecting the initial and final positions, passing through intermediate nodal points (Fig. 17).

Nodal Network

Path planning begins with a series of precalculations on the model environment. First, an array of "parent" nodes is generated from the geometric data in the model (Fig. 23). The index of the array represents the node number, while the value of the indicated array element is the Cartesian coordinate of the node. Numbering of nodes is sequential from 0. Nodes represent points where the robot can be safely moved, and are assigned to the corners of each obstacle in a room. Coordinates of a node are relative to the room's axis system. Obstacles are "expanded" outward to account for the size of the robot, such that
no collisions will occur as the robot passes through the nodes. Nodes that would lie inside the expanded region of another obstacle, or within the "contracted" walls of a room are eliminated.

Figure 23: Parent node generation.

Once all parent nodes are found, a second array is generated. The index of the array again represents the node number, while value of an indicated element of the array is a list of the node numbers of "children" compiled for each parent. Children are found such that the region corresponding to the swath the robot sweeps out in moving in a straight line from the child node to the parent node does not intersect the unexpanded
region of any obstacle (Fig. 24). This procedure takes advantage of the Macintosh's graphics facilities for determining the intersection of regions.

![Diagram](image)

**Figure 24: Child node assignment.**

The result of nodal precalculations is a massive interconnected inheritance tree. Added onto the tree at the time of a desired robot move are nodes corresponding to the robot and to the children of the robot. Path planning for the robot is a matter of linking together nodes in the inheritance tree, compiling a list of nodes that represent a point-to-point path between the beginning node and a child node from which the destination point can be
reached safely. The process becomes complicated in that generally there are numerous possible paths. Search is used to decide which of the possible paths to pursue.

Numerous search techniques are available [Winston, 1984], most of which involve formation and sorting of a queue, an ordered list of items. Search techniques are differentiated by the manner in which the queue is built and updated. In the path planning search, elements of the queue are lists of partial, incomplete paths. The search builds the partial paths into complete paths from the beginning node to the destination.

The A* search is used as a relatively fast means of generating an "distance optimal" (shortest) safe path. Two quality functions are relevant to the search, one being a measure of the known length of the current partial path, the other being the estimated remaining distance to the destination. Remaining distance is estimated as the straight-line distance from the current partial path end point to the destination point, regardless of obstacle interference.

The search operates in a cycle of path expansion, path evaluation, and queue sorting (Fig. 25). To reiterate, a path is an ordered list of nodes. The first node represents the starting point in the path, the robot's current position. The last node in the list represents the most recently explored avenue toward the destination. Path expansion involves first finding the children of the last node in the path at the front of the queue, then generating a set of new paths, one for each child, in which the child is added to the end of the path (Fig. 26). Evaluation involves determining whether the destination can be safely reached by any of the new paths, and calculating known and estimated remaining path lengths for each of the new paths. New paths are sorted against all paths in the queue according to total path length (known+estimated), the shortest path being placed at the front of the queue. If a complete path is found, it is compared against any other complete path and the
INPUT DESTINATION POINT

SAFE PATH FROM ROBOT TO DESTINATION?

Y

N

GENERATE CHILDREN NODES OF CURRENT ROBOT POSITION

INITIALIZE QUEUE WITH CURRENT ROBOT POSITION NODE

QUEUE EMPTY?

Y

N

EXPAND PARTIAL PATH AT HEAD OF QUEUE, CREATE NEW PARTIAL PATHS BY ADDING CHILDREN NODES

CAN DESTINATION BE REACHED BY ANY NEW PATHS?

Y

N

ANY PREVIOUSLY FOUND PATHS?

Y

N

CALCULATE QUALITY OF NEW PATHS, ADD TO QUEUE

THROW OUT ALL BUT SHORTEST PATH

SORT PATHS IN QUEUE

REMOVE FROM QUEUE ANY PARTIAL PATHS WHICH ARE LONGER THAN COMPLETE PATH

ANNOUNCE FAILURE

SUCCESS?

TRANSMIT MOVEMENT COMMANDS TO ROBOT

Figure 25: Path planning flowchart.
longer path discarded. Next, any partial paths having a larger known path length than the complete path are removed from the queue. The search continues until the queue is empty.

**Figure 26: Path expansion toward destination.**

**Moving from Room to Room**

Since indoor path planning can always be broken down into planning within individual rooms, path planning routines are part of the functionality of the class of room objects, i.e., a room can be asked to plan a path for a robot. By making the classes of houses and buildings inherit from the class of rooms, the path planning ability is
propagated to a more global, building-wide scale. Also, since the class of room objects inherits from the same generic region dialog item object as all other modelled classes, rooms themselves can be treated like any other obstacle on the scale of a building (Fig. 27). A room is owned by a building just as household items are owned by the room, and hence the room's position and orientation are described with respect to its owner. Treating rooms as obstacles avoids the necessity of defining a class of hallway items, since all area in a building not covered by a room is by default equivalent to a hallway.

Figure 27: Room-to-room path planning.
The same processes of node generation and queue formation are used in planning paths among rooms and other obstacles in a building as in planning paths among obstacles in a room. The room's corners are "expanded" and nodes assigned as with any obstacle. Of special importance in room-to-room path planning are doors, which define the interconnectivity and passability between rooms. Associated with each door are two "passage nodes", which are defined to lie two feet away from the middle of the door on either side, and represent a line of safe passage from one room to another. As the robot passes from one room to the next, its ownership is transferred to the room it is entering, and its position and orientation are reassigned to reflect the new ownership. As such, a passage node is assigned coordinates relative to the room in which it resides, and provides a convenient stopping point for the reassignment. A passage node also represents the place where the robot would be commanded to move in preparation for opening or closing a door.

On the scope of a building, a path is not a list of nodes, but a list of rooms, doors, and intra-room paths. If the robot's destination point lies in another room, the robot must find a path to a door connecting its current room to any intermediate rooms, through these rooms and to a door connected to the destination room, and from thence to the destination point. In a simplified notation, a path is represented as the following:

\[(\text{start\_room}, \text{interconnecting\_door}, \text{next\_room}, \text{path\_from\_start\_position\_to\_door})\]

\[\cdot\]

\[\cdot\]

\[\cdot\]

\[(\text{next\_room}, \text{interconnecting\_door}, \text{end\_room}, \text{path\_from\_door\_to\_door})\]

\[(\text{end\_room}, -, -, \text{path\_from\_door\_to\_end\_position}).\]
**Point to Point Navigation**

Whether on the scale of a room or a building, a path is navigated as of a sequence of straight line traverses connected by pure rotational realignments. Assuming a flat terrain, the traverse distance between two node points is calculated as the norm of the vector between the two points (Fig. 28). In preparation for making the traverse from node n to node n+1, the robot is oriented to face node n+1. The proper orientation for the traverse, relative to room coordinates, is calculated as the arctangent of a right triangle constructed between the two nodes. The robot is commanded to rotate an amount equal to the difference between its current orientation and the orientation for the traverse.

\[
d = \sqrt{(xd-x)^2 + (yd-ys)^2}
\]

\[
\theta = \text{atan2}((yd-ys), (xd-xs))
\]

---

![Diagram of point to point navigation](image)

**Figure 28: Point to point navigation.**

Pure rotation is produced by counter-rotation of the robot's two drive wheels, while pure translation is by means of co-rotation (Fig. 29). The BASIC commands for moving the wheels require an argument in linear inches, so that the desired degree amount of pure
rotation is converted to a linear amount of wheel movement by multiplying by a factor of 41.2/360 (i.e., it was found experimentally that wheel counter-rotation by 41.2 inches resulted in 360 degrees of base rotation).

Figure 29: Wheel rotation and base movement:
a) pure translation from co-rotation,
b) pure rotation from counter-rotation.
Chapter 6: Results of Experimental Testing and Demonstrations

This chapter discusses the results of laboratory and in-home testing. The hypotheses posed in this thesis are shown to be correct, within limits. Using the Machine Control Interface and cursor-activated software, handicapped users were able to command the robot to do useful pick and place tasks. Each of the users expressed an interest in purchasing the robotic system, assuming advanced features such as path planning and navigation were available.

Laboratory testing of the robotic system was ongoing as new software was written. Preliminary testing took place in the Mechanical Engineering Robotics Lab and the author's office at Rice University, and The Institute for Rehabilitation and Research in the Houston Medical Center. Once the author became confident of its reliability, the system was made available for public demonstrations and testing by other individuals. The Machine Control Interface and robot systems were transported from Houston to San Jose for the 1987 Rehabilitation Engineering Society of North America Conference (RESNA '87), where the MCI was successfully used in commanding the robot in teleoperator mode to move cups and other small objects between the floor and a table. Teleoperator operation using both mouse and voice inputs was also used at the 1987 Space Operations Automation and Robotics Conference (SOAR '87) at NASA Johnson Space Center in Houston. In laboratory demonstrations in December of 1988, an initial version of path planning and navigation routines were used to move the robot 20 feet back and forth across an obstacle filled room, with a positional error of under 6 inches. Automated grasping routines were demonstrated by Paul Fisher [Fisher, 1988]. In one trial, a three
inch diameter can weighing one pound was located and grasped from atop a 30 inch high table.

The World Modeller, path planner, model reflective command generation, and untethered operation via the Remote Console were demonstrated during the SOAR '89 conference at NASA/JSC. Testing of localization and path planning routines in combination with the wireless modem link has been carried out in the author's office and adjoining hallways in the Cox Building at Rice University. A skeleton model of the entire second floor of the building was created, with details of desks, chairs, shelves, and lamps included in the author's office. Using the model, the robot's position and orientation within the office could be determined to within 6 inches and 5 degrees, respectively, although major miscalculations occurred occasionally. Precalculations for path planning lasted roughly five minutes, and actual path planning required less than a minute. The robot successfully traversed paths extending from the office into the hallway. The range of the modem link was exceeded and communications lost if the robot traversed more than 100 feet away from the base unit.

Beginning in March of 1988, the robot and MCI were placed in the homes of severely physically handicapped volunteers. At that time, the robot was tethered and none of the autonomous features had been developed. Five subjects used the system in their homes for periods of time ranging from one day to six weeks. The equipment was transported to the subject's home, and set up by laboratory personnel. The user was supplied with a manual describing the bootup procedures for the system and given a brief explanation of the Macintosh operating system and MCI, as well as the operation of the robot and control software in Teleoperator mode. Some users were able to operate the system using the standard mouse.
Each of the subjects was able to make the robot perform useful tasks that increased their personal autonomy. Typically, users experimented with moving cups and other small objects about their living quarters. One subject mentioned using the robot to move papers around on a desk and retrieve materials that got dropped on the floor. The subject who used the MCI over the most extended period of time found the robot to be compatible with the MCI. Subjects found the control software easy to understand and use in conjunction with the MCI, but had problems with the encoder unit in that it tended to droop out of their range of motion.

Subjects noted several areas that needed improvement. The robot was found to be cumbersome to move about the room due to its tether and to the incremental nature of commands generated in Teleoperator mode. Those who were able to use the remote console to control the robot found the remote console too slow and unreliable to be really useful. The time required to perform a single pick and place task was rather long, often greater than 5 minutes. Hence, users felt that if the robot was easier to move and required fewer commands to operate, it would be a more useful tool. All users indicated they would be interested in purchasing a robot once simplified means of commanding the robot exist. Such means are available through the World Modeller, so that the system is ready for a second cycle of in-home evaluation.
Chapter 7: Directions for Further Research

While the work presented in the preceding chapters has laid a foundation for a practical robotic system, testing and user evaluation have demonstrated the need for further development, particularly towards solving the problem of commanding tasks that combine navigation and manipulation. Software by nature can be made better by revision. This chapter outlines areas that need improvement and possible avenues of future development.

Manipulation

The major area of development needed to produce a completely autonomous robotic aid is manipulation. The robot not only must be able to get to the vicinity of an object, but must know how to approach and grasp it. A separate thesis has addressed problems associated with sensor-guided manipulation of objects, and algorithms for automated object detection and grasping have been incorporated into early versions of control software [Fisher, 1989]. Further, an attempt should be made to integrate knowledge of the intrinsic (size, geometry, weight, texture, etc.) and extrinsic (location and orientation in space, proximity to other objects) characteristics of an object in developing a manipulation strategy.

Recent efforts aimed at increasing the manipulative abilities of the robot have brought to light several possible approaches. In addition to developing a more complicated, dextrous gripper, tools and fixtures can be designed for the robot. For instance, suction cups or gum rubber can simplify the task of picking up a piece of paper. Another very conventional approach is to modify the environment, adding levers and handles to objects. The tradeoffs between these approaches need to be considered further.
Communications

The communications link between the Macintosh and HERO, detailed in Appendix 1, is the most troublesome component of the system. Despite extensive efforts toward providing error checking routines, communications is not always reliable, and can be a bottleneck. The synchronous protocol used results in the host computer remaining idle much of the time a transmission in taking place. An asynchronous, interrupt driven protocol with enhanced error checking would be preferable.

Onboard Programming

Some of the problems associated with the communications link arise due to the uneven distribution of computational burden within the system software architecture, which necessitates a great deal of low level data transfer between the Macintosh and HERO. Appendix 2 gives an overview of the Master/Slave system architecture that is currently configured such that Master software onboard the SE does all computations involved in planning an action, and transmits commands in the form of BASIC statements to the HERO for direct execution by the onboard BASIC interpreter. Communications requirements could be reduced if the Slave was written to interpret and execute high level commands as well as to handle error conditions.

Hardware Upgrades

Various hardware upgrades exist or can be imagined that can enhance the versatility of the robot control system. The advent of lap-top computers and the availability of a remote controlled household appliance controller allow the robot control system to be made
completely portable, which is of particular benefit to wheelchair-bound users. Graphics and search routines can be made to run closer to real time on newer, faster machines.

A very active topic in the field of robotics is sensors, since "intelligent" action is intimately related to the ability to sense. As demonstrated in Fisher's thesis, sonar, LED switches, and other sensors can be easily added to the HERO using an experimenter card. A touch-sensitive bumper and a sonar/light detector with finer resolution would be highly desirable in making navigation more reliable and accurate. Although inaccuracies in estimated position will not prevent the robot from navigating autonomously, they will necessitate more frequent localization readings, or the use of supplemental procedures. While cost and complexity are major arguments against incorporating a computer vision system, a remote camera could be placed onboard the robot for the benefit of the user, and would be particularly necessary when the robot is out of sight of the user.

Manipulation and passage through doors presents a problem for the handicapped as well as for the robot. In lieu of being able to program a robot to open doors, a household appliance for actively opening doors, similar in design to commonly used passive dampers, could be developed. Such an appliance could be controlled by the X-10 interface.

Modelling Improvements

Taking the World Modeller software into consideration, two areas of immediate interest are the ability to display a model from different vantage points in three dimensions, and to model dynamically changing objects and environments. Written from scratch in the course of less than a year, the World Modeller software emulates the 2-dimensional graphics capabilities of the MacDraft application. Since the underlying data base of the World
Modeller is fully 3-dimensional, software emulating a 3-dimensional graphics package such as the Mac3D application could be written.

Currently, the World Modeller is able to model static, structured environments in which the robot is the only active agent. The ability to model dynamically changing objects and environments would greatly extend the possible applications of the system. If the temporal dependence of a dynamic object's state can be described analytically, the computer's real-time clock and event handling routines (e.g., window-null-event-handler) can be used to continuously update the object's state within the model. Otherwise, the World Modeller would necessarily rely on sensory information available from the robot, user, or other system in determining the object's state.

The particular search strategy used in planning paths is suitable for static, structured environments in which objects are modelled as polygons. The process of node generation and assignment of node children becomes hopelessly bogged down as objects change relative position in a dynamic environment. A more flexible strategy, such as that proposed in [Gilmore et. al., 1984] can take advantage of the system's ability to model objects as randomly shaped regions, and does not rely on a predefined nodal network. Navigation in dynamic environments will depend on a combination of global, high level path planning based on a static data base, and local, low level obstacle avoidance using sensor-based "reflexive" routines onboard the robot.

To extend the model into outdoor environments, the ability to model contoured surfaces would be needed. As a first pass, contours can be generated in a 2 1/2-dimensional sense within the World Modeller by laying down areas or "slabs" of floor having a particular uniform thickness, one on top of each other. Navigation would involve calculating traverse distances as an integral of a path segment over a contour.
Exploring/Mapping

Two areas where the World Modeller framework is particularly well-suited are in exploring and mapping. An unexplored area can be represented by a "clean slate" within the World Modeller, with objects being added to the model as they are detected and identified by the robot. The generic region dialog item provides a suitable representation for objects that have been detected but not identified as belonging to any particular class. An object can be re-instantiated within its rightful class as sensory data reveal further detail. In mapping a region, the robot can be made to wander or travel in a predefined pattern, establishing its location within an ever-expanding neighborhood by means of (yet to be developed) portable infrared beacons deployable and controllable by the robot, dropped like bread crumbs. In combination with the beacons, an onboard electronic compass would allow the robot to define an absolute world reference frame.

Other Environments

Much current research in robotics is focused on developing autonomous extra-terrestrial rovers, for Mars and perhaps the Moon. Such environments are basically static, with no active agents besides radiation, falling asteroids, or in the case of Mars, wind. The World Modeller described in this thesis, complete with suggested improvements, might be applied to guiding a robot in such environments.

Teaching

Beyond the simple record and playback strategy used so far, teaching can be developed as a query and response interaction between the robot and user, much like that between a parent and child. The user may help the robot in defining an algorithm for solving a particular problem or class of problems. Along these lines, the definition of "primitives"
found in Chapter 6 can be extended to include not only robot states, but entire patterns of activity such as performing a sensor sweep or planning a path between two known points. Ultimately, familiarity with concepts of teaching may lead to an understanding and implementation of machine learning.

Technology Transfer

Widespread application of robotics in areas proposed in this thesis will rely on the ability to train individuals who can disseminate the technology. To this end, laboratory courses have been developed that introduce basic concepts and stress the abilities as well as limitations of current robotic technology [Cheatham et. al., 1987b]. Work with undergraduate engineering students at Rice University and occupational therapists studying in a Master of Arts program at Texas Women's University in Houston has shown that people from different walks of life can become quickly familiarized with the technology. As an example, an informal study was made to determine how long it takes non-disabled users to perform a manipulation with the robot, and what sort of learning curve is involved. Groups of 4 to 5 students performed a timed pick-and-place task. Using the Remote Console, the subjects moved the robot 5 feet across the floor, picked up a block from the floor, then deposited the block on a tabletop. On the first try, an average time of 5 minutes was recorded, which decreased to under 3 minutes on the second try. As a further example, groups of students from both programs were able to program the robot to perform a variety of advanced operations, including navigational tasks using sensory feedback. Such instruction in robotic and computer technology is fruitful and should be continued.
Chapter 8: Summary and Conclusions

This thesis has presented a beginning framework for a practical, useful application of robotics and related technology within a domestic or office environment. A system of hardware and software has been developed that allows a user, disabled or not, to command a mobile robot both in a low-level, teleoperator mode, and a high-level, task-oriented mode. Also provided are facilities for controlling household appliances, regardless of whether the robot is needed or not. Viewed as an aid for severely physically handicapped people, the robot control system is merely a subsystem in a larger project aimed at providing for needs of mobility, entertainment, and independence in performing tasks of everyday living.

A great deal of emphasis has been placed on making the system easy to understand and operate, especially by nontechnical people. A minimum of equipment is used in an attempt to make installation of the system as simple as possible. Components can be set up and hooked together in less than 10 minutes. The World Modeller boots up automatically when the host computer is powered up, and the user can be commanding the robot shortly thereafter.

At the heart of the system is an interactive graphics interface. Software was designed to emulate commonly available software applications that users may already be familiar with. With the interface, the user participates in creating a scale model of his world, adding detail as the need arises. This graphic model not only provides a data base for the computer, but gives the user a window into how the robot is made to execute a task. A working model of a room can be developed in a matter of minutes or hours, depending on the user's skill and familiarity with the interface. Though blueprints are helpful for
modelling on a building-wide scale, a good tape measure is all that is needed in measuring dimensions of ordinary objects.

Modelling tools have been written in an Object-Oriented LISP language, which provides a modular framework for program development. At the root of an object hierarchy, the region dialog item was created as a bridge between a 3-dimensional internal representation of a generic real world object, and a 2-dimensional graphical entity that can be manipulated under mouse action. The property of inheritance allows detail to be added to a model, since code written to represent a general class of objects can be added upon incrementally to create ever more specialized classes. The fact that Object-Oriented programming combines data and procedures into one primal entity, the "object", allows both physical characteristics (size, shape, position, and orientation) and behavior (changes of position and orientation) of real objects to be modelled compactly, and in a sense "naturally".

The phrase "model reflective command generation" has been coined to describe the manner in which commands can be generated for a real robot based on user-directed actions of a model robot. Forward kinematic commands, those that move individual joints in discrete amounts, are generated using sliders and buttons within a control panel window. Inverse kinematic commands, those that require a concerted motion of several degrees of freedom, are generated indirectly as a result of the user specifying desired position of the robot base or gripper. Command generation relies heavily on the use of the computer screen as a digitizing pad and the mouse as a digitizing stylus for producing numerical parameters.

The stated hypotheses of this theses have been shown to be correct, with limitations. Using the Teleoperator command strategy, users were successful in commanding the robot to manipulate objects such as papers and cups in a domestic environment, and have
expressed a willingness to purchase such a system, pending the availability of the more powerful command features provided within the World Modeller software. Other necessary improvements involve overcoming limitations in the sensory capabilities and onboard programming power of the robot, and the reliability of its overall operation, particularly the operation of the arm and the wireless communications link.

Several conclusions can be drawn from this thesis:

1) Inexpensive, off-the-shelf robotic and computer technology is available that can be of use to researchers and to a significant segment of the general population, particularly the severely physically handicapped.

2) This thesis develops the framework of a system for commanding a robot in an indoor, static, structured environment. A severely handicapped user's home is an example of such an environment.

3) Object-Oriented LISP provides a powerful, compact structure for representing the real world and for implementing advanced problem solving routines related to robot autonomy.

4) A completely optimized, tested, and debugged version of the software developed may have market value. This conclusion was confirmed initially by response to a demonstration of the system at the 1987 RESNA conference in San Jose, and will be further tested by a demonstration at the 1990 RESNA conference in Washington DC.
References

Andersen; Donna, Lowe; Carlyle, Reeme; John, Struthers; Jim, and Webster; Will, "Enhanced Capabilities for a Personal Autonomy Robot", final report, Mech 404, Department of Mechanical Engineering and Materials Science, Rice University, 1987.


Badders; Dan, Chang; Noel, Go; Pam, Holguin; Sophie, Lawler; Ken, and Le; Phi, "Design of RHAMBOT: Rice Handicap Aid Manipulator BOT", final report, Mech 408, Department of Mechanical Engineering and Materials Science, Rice University, 1989.

Bentsen; Fred A., McCurdy; Mark, Finkelstein; Stuart, and Porter; Pat,"Design of a Dextrous Hand for a Mobile Robot for Physically Disabled Persons", final report, Mech 407, Department of Mechanical Engineering and Materials Science, Rice University, 1989.


Crangle; Colleen, Liang; Lin, Suppes; Patrick, and Barlow; Michael, "Using English to Instruct a Robotic Aid: An Experiment in an Office-Like Environment", Proceedings of the International Conference of the Association for the Advancement of Rehabilitation Technology (ICAART '88), Montreal, Canada, June 25-30, pp. 466-467.


Hammel; Joy, Hall; Karyl, Van der Loos; Machiel, Leifer; Larry, and Perkash; Inder, "Clinical Evaluation of a Desktop Robotic Aid for Severely Physically Disabled Individuals", *Proceedings of the International Conference of the Association for the Advancement of Rehabilitation Technology (ICAART '88)*, Montreal, Canada, June 25-30, pp. 448-449.

Holloway; Karen, Leifer; Larry, and Van der Loos; Machiel, "Factors in the Design and Development of an Interactive Human-Robot Workstation", Proceedings of the Eighth Annual Conference on Rehabilitation Technology (RESNA '85), Memphis TN, 1985, pp. 268-270.

Jackson; Gregory S., Giddings; Ross, Robert; Douglas B., and Coil.; Millicent, "An Improved End Effector for the HERO 2000 Robot", final report, Mech 404, Department of Mechanical Engineering and Materials Science, Rice University, 1988.


Lees; David S., Crigler; Robert, Van der Loos; Machiel, and Leifer; Larry, "A Second Generation Desk-Top Robotic Aid", Proceedings of the 10th Annual Conference on Rehabilitation Technology (RESNA '87), San Jose, CA, 1988, pp. 796-798.

Lees; David S., Crigler; Robert, Van der Loos; Machiel, and Leifer; Larry, "A Third Generation Desktop Robotic Assistant for the Severely Handicapped", Proceedings of the International Conference of the Association for the Advancement of Rehabilitation Technology (ICAART '88), Montreal, Canada, pp.450-451.


Van der Loos; H. M. F., Michalowski; S. J., and Leifer; L. J., "Development of an Omnidirectional Mobile Vocational Assistant Robot", Proceedings of the International Conference of the Association for the Advancement of Rehabilitation Technology (ICAART '88), Montreal, Canada, June 25-30, pp. 468-469.


Winston; P. H., Artificial Intelligence, Reading, MA, Addison-Wesley, 1984, pp. 89-114.

Appendix 1: System Configuration and Setup

This first appendix describes the individual pieces of equipment used in this thesis, and how they are integrated to form a working system. Some of the programming nuances required for this integration are mentioned. The appendix can serve as a practical guide for setting up the system.

Schematic

As shown in Figure 30, the system has 4 basic hardware components: a personal computer (Macintosh SE), a mobile robot (HERO 2000), communications link (remote console and/or wireless modem), and household appliance controller modules (X-10 Powerhouse). The Machine Control Interface is optional, and replaces the mouse. Custom software running on the SE ties these components together.
Assembly

Assembly of the system is straightforward. The Macintosh can be set up so that the robot control software is activated on power up. Although the two serial ports on the back of the computer are equivalent, the control software is set up with the modem port used for
communications with the X-10 computer interface, and the printer port for the radio frequency link. As shown, the X-10 power cord plugs into any standard 2-prong electrical outlet in the building, as do the individual lamp and appliance modules. The RS-232 cable from the X-10 is plugged into the modem port of the SE with the help of an adaptor cable.

Either of the radio frequency links can be connected to the printer port of the SE individually; the A-B switch is included for redundancy. If the Remote Console is to be used, a null modem cable (one with pins 2 and 3 swapped) will be needed. Alternatively, the Console can be altered by swapping connections on its internal circuitry. If the wireless modems are used, the second of the pair can be housed in the torso of the HERO 2000. The unit is nearly identical in size to the optional disk drive, and hence can fit in the bracket provided. Also, the 12 volt power cable supplied for the disk drive can be easily adapted for the modem, which otherwise has its own battery. The output of the modem is connected to the upper serial port on the back of the robot, again using a null modem cable. No pin swapping is needed in the cable between the base modem and the computer. As a last resort, the robot can be tethered to the SE by running a null modem cable directly between the serial ports of the two machines.

Machine Control Interface

The Incap Machine Control Interface (MCI) was developed by Tom Krouskop at The Institute for Rehabilitation and Research in Houston. The MCI is composed of a motion encoder unit, an LED menu display, and first and second stage control units (Fig. 31). The motion encoder, LED display, and first stage control unit constitute a portable subsystem that can be mounted on a user's wheelchair. The first stage unit has outputs for controlling a wheelchair and a musical synthesizer. Using an RS-232 protocol, the
first stage control unit communicates with the second stage over either cable or infrared LED links. The second stage unit produces mouse, paddle, and keyboard outputs.

Figure 31: Incap Machine Control Interface System Schematic.

The encoder unit uses Hall-effect transducers to translate 3-D head movement and breath sip/puff action into four independent analog outputs. Fitted with a mouthpiece at its extremity, the encoder is a three-segment, parallel-linkage arm that can be mounted to the user's wheelchair or on the user's shoulder (Fig. 32). The converted signals from the encoder are used by the control signal output unit to produce a variety of machine control
outputs, including computer keyboard, paddle, and mouse emulation as well as controls for powered wheelchairs and a musical synthesizer. One notable feature of the MCI is that its output response and sensitivity are programmed to fit an individual user's range of motion.

Taken in the context of its application as an aid for severely physically handicapped people, the robot control system can be seen as an optional peripheral device for the MCI. Use of the robot can be part of an entire rehabilitation program, where a patient uses the MCI in a progression of activities ranging from wheelchair control to interaction with a computer and finally control of the robot.

Figure 32: Schematic of Machine Control Interface Encoder.
HERO 2000 Robot

The HERO 2000 robot is available as a kit or in assembled form from the Heath/Zenith company. The robot is equipped with a four axis, revolute jointed arm; a rotatable torso section adds a fifth degree of freedom. The parallel jaw gripper can be commanded both in position and force. Twin opposed independent drive wheels in the base provide forward/backward movement as well as a zero turning radius. An onboard 8088 processor supports BASIC for programming all joint and sensor functions, while individual 8042 processors handle position and velocity control of optically encoded, 12 volt DC joint drive motors. The robot features a light sensor, two sonar sensors, a temperature sensor, and a low battery level indicator, as well as a speech synthesizer. For communications, a 600 baud radio frequency link and two programmable baud rate RS-232 serial ports are provided. The upper port is used for communications with a dumb terminal or host computer, while the lower will drive a printer.

Modifications

An ongoing series of projects has sought to improve the functionality of the HERO by making simple modifications to its mechanical design [Anderson et. al., 1987; Jackson et. al., 1988; Badders et. al., 1989; Bentsen et. al., 1989]. By doubling the length of the link between the shoulder and elbow, and placing the entire arm in a second, higher torso section, the workspace of the arm was extended to reach from floor level to table top heights of up to 40 inches, with a depth of reach of 24 inches (Fig. 33). Efforts have also been made to smooth out the overall ride by adding a shock absorbing suspension. Design of a more versatile, sensitive gripper is underway. The parallel gripping action will be augmented by a curling action of additional finger segments.
Figure 33: Vertical workspace of the modified HERO 2000 arm, roughly to scale.

Horizontal workspace is generated by sweeping the vertical +180/-165 degrees.
Sensors

As previously mentioned, the robot is equipped with a variety of sensors. The light sensor and one of the two sonar transducers are housed above a rotating mirror in the head. Readings can be taken every 15 degrees around the entire circumference. The signal from the light sensor is converted into a digital reading between 0 and 255, indicating the relative intensity of ambient light at a particular reading. The phototransistor used in the sensor is sensitive to light in the visible and infrared spectra. The sonar transducers have a range of 10 feet to 4 1/2 inches, with an accuracy of +/- 1/2 inch. The second sonar transducer is fixed in the base, pointing forward.

BASIC and DOS

BASIC on the HERO is contained in ROM, and is activated from a keypad on the head of the robot [Heath, 1986a]. This particular dialect supports commands for moving and polling the arm joints and base drive wheel, as well as for polling the sensors. A typical arm joint command has four parameters: 1) the joint to be moved, 2) the absolute degree amount, relative to vertical, to set the joint at, 3) the relative speed of movement, and 4) a "hold" code that specifies whether the movement must be completed before program execution continues. Base commands are similar to arm commands, except that the second parameter is an incremental amount of movement in linear inches. Drive wheels can be programmed both individually and in unison, allowing a variety of movements. The sonar and light sensors in the head are typically polled from within a loop, while a single statement polls the base sonar. Sensor and motor functions can also be accessed on a lower level by means of statements that address status and command ports directly. Such facilities are used in programming the head reflector to sweep over a particular region.
One serious shortcoming of the dialect is its lack of error handling statements. If an out of range parameter is entered or generated for a joint command, an error message is signaled and program execution is halted. This places serious limitations on autonomous operation of the robot.

With the optional disk drive installed, the onboard computer will run under MS DOS, from whence BASIC can be booted. Availability of DOS aids in program development for the HERO, although it does not alleviate the shortcoming of BASIC mentioned above. Also, operation of the computer cannot be switched between BASIC and DOS while using the Remote Console. However, if the wireless modem is used, command can be switched at will, although the problems arise in attempting to transmit control characters, which are interpreted as commands for the modem itself, and hence are not transmitted. For safe operation of the robot, control codes are needed, such as the "control C" code for halting program execution. For these reasons, the disk drive has not yet been used.

Communications

Error-free communications between the Mac SE and the robot are essential if the system is to work dependably and semi-autonomously. Two different combinations of hardware and software were used, each with its own drawbacks.

Remote Console

The Remote Console for the robot transceives at 600 baud and has a range of 50 feet. It operates in one of three main modes: remote, link, and terminal. In remote mode, the Console acts as a wireless keyboard, from which programs can be entered and executed. From the remote mode, a teach mode can be entered, in which case direct control of motor functions is controlled by auxiliary keys on the Console. The teach mode also allows
sequential positions of the robot to be recorded automatically in a BASIC program, by means of which the robot can be made to repeat a sequence of movements. In terminal mode, the Console has the same keyboard function as in the remote mode, but communicates via an RS-232 cable between its serial port and the top serial port on the robot. Lastly, in link mode, the Console acts as a dumb transceiver between the robot and a dumb terminal or host computer. ASCII characters received by the Console from the host are collected in a buffer until a carriage return character is entered, at which point the buffer is transmitted to the robot. The Console can be made to echo characters sent to it by the host.

It is this last mode that is used for communications between the Mac SE and HERO computers via the Console. Lack of an efficient handshake protocol for the Console requires scrupulous error checking routines to be written on the host SE (Appendix C). The Console is programmed to continuously check for contact with the robot, and it is the host's task to try to get a transmission in edgewise, so to speak. Hence, transmission of a sequence (string) of ASCII characters is a 3-step procedure: 1) determination of the Console state, 2) transmission and return echo of the first character of the sequence, and 3) transmission and return echo of the entire remaining sequence.

At any point in time, the Console can be in one of three states: "normal," "hung", or "transmitting". The normal or "idle" state occurs when neither the host nor the robot is attempting to transmit. The host recognizes this state by the availability of a continuous stream of escape sequences produced by the Console, alternating between \(<\text{esc}>21\) and \(<\text{esc}>2h\) [Heath, 1986b]. In order for transmission to begin, the Console must be placed in its normal state. The hung state, as well as the state when the Console's power is off, is characterized by a lack of escape sequence production by the Console. If the console is merely hung, it will respond to communication from the host with an echo of characters
sent. If off, the Console will not respond at all, naturally. The hung state occurs as a part of a normal transmission, when the first character of a string has been received, and the console is waiting for further characters. If this case is true, the host will have received an echo of the first character, and will complete the transmission. However, the hung state also arises when spurious characters enter the Console's buffer, as occurs when the Console is listening for the robot rather than to the host while the host is attempting to communicate. If this second case is true, the host will not have received an echo, and will clear the Console before attempting the entire transmission again. Clearing the Console is a matter of the host transmitting "delete" characters to the Console until the Console is detected to be in its normal state. By default, if the Console is not in a normal or hung state, it is in its transmitting state, when transmission between the Console and the robot is actually taking place.

**Wireless Modem**

Many of the intricacies of error-checked transmission using the Console can be avoided if a pair of wireless modems such as the Esteem is used. The modems transceive at up to 9600 baud, with a range of up to a mile, line of sight [Esteem, 1988]. Using the modems, error-checked communication is a matter of the host first transmitting the entire string, then waiting for the entire string to be echoed by the robot itself. Failures that occur are almost always due to the units being out of range of each other. For the them to work properly with the HERO and SE, the modems must be programmed to suppress all internally generated prompts, messages, and echoing. In preliminary indoor testing within the system, the modems were found to have a range of not more than 100 feet.
Host Computer

The Macintosh was selected from a variety of possible host computers on the basis of its user-friendly interface, two serial ports, portable size, and the availability of Object-Oriented LISP development software. A hard disk and extra memory were desired for more efficient software execution. One drawback is the machine's small screen. An ideal machine would be battery powered and truly portable, allowing the entire system to move with the user. Such machines are currently appearing on the market. The computing speed of the newer 32 bit microprocessors (Intel 80386, Motorola 68030) is also desirable, as it will allow interactive graphics and task planning facilities to operate closer to real-time.

X-10 Powerhouse Computer Interface and Modules

The household lamp/appliance controller is sold under a variety of names by electronics outlets such as Radio Shack. Two models of the base unit are available, one that must be plugged directly into a wall outlet, the other that is remotely linked to a separate wall unit by radio. Lamp modules can accommodate household fixtures rated at up to 200 watts, while appliance modules can deliver up to 15 amps. The modules incorporate two 16-position rotary switches, one for designating a "house" code, the other for a "unit" code, allowing the X-10 to address up to 256 modules individually. The controller is connected to the modem port of the SE, and communicates at 600 baud. The controller responds to coded hexadecimal commands generated by a host computer according to the manufacturer's specifications [X-10, 1989]. Inside a command, a module's house code, unit code, and desired activation state are specified. Having the X-10 interfaced to the SE provides the user with an environmental control system, regardless of whether the robot is used or not.
Appendix 2: Software Basics

This Appendix explains the particular features of the language in which the robot control software is written, and the reasoning behind the choice of developmental software.

Lisp

The LISP (LISt Processing) language predates most other formal high-level languages besides FORTRAN, and, as its name implies, involves operations on lists of things. Since "things" in many cases are symbolic rather than numeric entities, LISP has traditionally been associated with research in Artificial Intelligence (AI), a field that endeavors to make machines "smart" in the sense of having the logical and symbolic reasoning abilities of humans. Among recent authors on general topics in AI, Tanimoto gives an excellent treatment of the uses of LISP [Tanimoto, 1987]. This thesis emphasizes aspects of AI in robot control, and hence LISP was chosen over other available languages such as object-oriented versions of C (C++). Allegro Common was the first object-oriented dialect of LISP available for the Macintosh.

Recursion

One of the powerful features of various dialects LISP is the ability of a function to recurse, to call upon itself. Recursion can be seen as a generalized iterative process, where a function may be called again and again, each time with a different argument. Analogous to a looping iteration's use of and index variable that is updated (incremented or decremented) each iteration, a recursive iteration uses functional arguments that are updated on each recursive call. Analogous to the looping iteration's termination when the
index variable reaches a pre-specified value, recursive iteration termination occurs when an argument-dependent condition exists. Figure 34 shows a comparison between recursive and looping implementations of the integer factorial function. A good learning guide to recursion can be found in [Friedman & Felleisen, 1987]. Recursion becomes important in referencing objects within an object hierarchy, as occurs when developing and operating on a model of the environment (Chapter 2).

Object-Oriented Programming

Object-Oriented programming is a departure from conventional approaches in the sense of how data and procedures exist within the code [Stefik & Bobrow, 1988]. While traditional programming separates data from the procedures that operate on the data, object-oriented programming combines data and procedures in entities called "objects". As a result, objects can "stand alone", can perform an operation without communicating with other processes. Object-oriented programming emphasizes modularity and ease of development.

In conjunction with LISP, object-oriented programming eases the task of representing real world items on a computer. Non-numerical characteristics of a modelled item can be recorded directly into the data structure of an object, rather than perhaps being coded in numerical form. Behavior or changes of state of a real item can be modelled computationally by procedures included as a part of an object. Examples of objects and object behavior are given in Chapter 2.
a) (define loop-factorial (j)
    n=1
    loop i = 1 to j
        n = n * i
    next i
    return n
)

b) calling loop-factorial (5)
    loop-factorial (5) returned 120

(define recurse-factorial (j)
    if j = 0 then return 0
    else return (j * recurse-factorial(j-1))
)

calling recurse-factorial (5)
calling recurse-factorial (4)
calling recurse-factorial (3)
calling recurse-factorial (2)
calling recurse-factorial (1)
calling recurse-factorial (0)
    recurse-factorial returned 1
    recurse-factorial returned 2
    recurse-factorial returned 6
    recurse-factorial returned 24
    recurse-factorial returned 120

Figure 34: Recursive vs. Looping iteration;

a) pseudocode b) trace of function calls.
Object Hierarchy and Inheritance

A basic feature of object-oriented languages is the existence of an object hierarchy, by means of which newly created objects are made to inherit properties from previously existing objects. To say that a child object inherits from a parent object means the child has direct access to all internal variables and procedures of the parent, but not vice versa. A child may be assigned additional internal variables and procedures that serve to differentiate it from its parent. Objects can be divided into two categories: classes and instances. A class is a description of one or more similar objects, a template that imposes a basic data structure and supplies basic procedures for an object, but not necessarily any details. An instance is an object in which the class template has been filled in with all the details, i.e. for which all variables have assigned values. Figure 35 depicts a comparison between a class and an instance. Ever more specialized objects are created as new classes are created based on a more general classes. This specialization process is demonstrated in the class hierarchy of Allegro Common LISP objects used in development of the robot control software (Fig. 36)
Figure 35: Class vs. Instance;

a) template for class of "desk" object,

b) instantiation "desk 1" of the class "desk".
Figure 36: Allegro Common Lisp class hierarchy; instances of a class are outlined in bold.
Interactive Graphics

Interactive graphics lie at the heart of the robot control interface. Software resident in the Macintosh ROM controls basic graphic elements, including windows, dialogs, dialog items, menus, and menu items. As demonstrated in Figure 36, Allegro Common LISP has full access to these routines, incorporating them into the object hierarchy [Coral & Franz, 1987]. A window is a rectangular region on the computer’s screen to which screen-related input/output is directed. For instance, most Macintosh text editors operate within windows, displaying keystrokes as they are typed. Messages to users are displayed in windows that open or "pop up" when some condition occurs. Associated with each window is a graphics plane or portrect, the area in which graphic objects can exist. The portrect is in general much larger than the area of the screen, so that a window literally acts as a movable "window" for viewing a portion of the graphics plane.

A dialog is a special class of window that performs actions in response to mouse and keyboard activity. Text, graphics, and dialog items can be combined in a dialog. A dialog item is like a dialog in that it performs or initiates some action in response to mouse or keyboard activity, but is a smaller, special purpose graphical entity. Allegro Common Lisp specifically defines button, radio button, editable text, static text, checkbox, table, and checkbox dialog items, as well as providing a generic user dialog item.

Menus are interactive window-like entities residing in the menubar at the top of the Macintosh screen, which open up or "pull down" when activated by a mouse click, presenting the user with a selection of menu items. Like the dialog item, the menu item responds to the user's "selecting" it by performing some operation.

User actions, such as moving and clicking the mouse, hitting keys, and inserting disks, are referred to as events. Dialogs and dialog items are made to respond to specific events in a particular manner by means of event handler procedures. Potential users of the
robot control interface are assumed to be familiar with the operation of the basic elements in the Macintosh interface.

**System Software**

Software development and choice of hardware for the project was very much an evolutionary process. Throughout, a "max-min" approach was taken, meaning the maximum amount of usefulness was gotten from the minimum amount of existing hardware and software before adding more software and better equipment. For instance, the teleoperator mode (Chapter 3) was first implemented as a stand-alone program written in BASIC onboard the robot, activated by keyboard inputs from a dumb terminal. As better teleoperator performance (more operations per given time period) and a more adaptable user interface became issues, a 512K Macintosh was added to the system, and a mouse-activated control panel was designed. At this point, the Master/Slave software architecture was adopted (Fig. 37), giving rise to the issue of how computational tasks were to be distributed between machines. Initially, programming on both machines was in BASIC, with the thought that BASIC would be the language of choice should the eventual end user desire to develop software himself.

At the point where autonomous functions such as path planning and navigation were attempted, the larger memory capacity and faster file access of the Macintosh SE were desired. Use of the more powerful LISP development software on the SE allowed programming to enter the realm of Artificial Intelligence.

**Master-Slave Architecture**

Software for commanding the robot is arranged in a Master/Slave configuration, with the Master software residing on the Macintosh SE, and the Slave on the robot (Fig. 37).
At present, distribution of computational tasks is one-sided, with the bulk being handled by the SE. The HERO has few facilities for interfacing with the user, particularly one who is severely physically handicapped, so that software supporting the user interface is entirely within the domain of the Master. On the other hand, execution of low level commands, such as those used to move joints, poll sensors, and monitor overall robot status is entirely within the domain of the Slave, since such commands are written into BASIC onboard the robot.

A middle ground exists when intermediate and high levels commands are considered. High level commands describe a complete task, one that may involve a sequence of robot actions, such as navigating to an object, identifying it, picking it up, and presenting it to the user. Intermediate level commands describe one particular robot action, such as navigating between two points in a building, or calculating position within a room based on sensory inputs.

In selecting a distribution, a relation exists between the level of commands the Slave can interpret, and the communication requirements between the Master and Slave. For instance, if a base movement routine is resident on the Slave, the Master need only pass the coordinates of the desired end point in order to command the robot to move point to point. If no such routine exists on the Slave, the Master must transmit entire BASIC statements for turning and traversing the base. Given the current limited availability of alternate high level development languages for the HERO, and the lack of error handling statements in the HERO's ROM-based BASIC, the minimum share of the computational burden was given to the Slave, i.e., only the BASIC interpreter running in immediate mode is required. Hence, the Master's role is not simply to support the user interface, but to interpret user actions as high or intermediate level commands for the robot. This one-sided distribution has the advantages of simplicity and dependability.
Master Software
(Macintosh SE)

Mode Selection Menu

Command Parameters

World Modeller
Dialog

BASIC Statement

Transmission Echo,
Returned Data, and
BASIC prompt

BASIC Command Sequence

Command Parameters

Robot Controller
Dialog

BASIC Statement

Transmission Echo,
Returned Data, and
BASIC prompt

BASIC Command Statement

Command Parameters

Teleoperator Mode
Dialog

Error Checking

X-10 Command

Slave Software
(HERO 2000)

Basic Interpreter

Figure 37: Master/Slave Control Software Architecture
Appendix 3: Biographical Note

I began my life on December 13th, 1961 in Princeton, N.J., where my Pop became Dr. Regalbuto. I was soon transplanted to Fort Worth, where I grew into the grace and beauty which Texans are renowned for. When I was 13, I had grown my first mustache and had almost drowned after purposely hyperventilating. My first romance came when I was 17; I used to love how all the other guys stopped and stared as I went down the street with her topless, my '71 MGB convertible and I. By the time I was 22, I had graduated magna cum laude from Rice University, ready for a career in the movies, or at least cigarette commercials. By the time I reached 24, I had abandoned these earlier aspirations and merely wanted to get out of M.I.T. with a Master's Degree and the remnants of my self esteem. At 28, one of my mane pastimes was plucking grey hairs, or rather keeping my beloved fiance Debbie from doing so. At any rate, I was about to finish my college career. The following is a chronicle of an adventure lasting well over three years of my life, a time I feel was spent fulfilling my imagination as much as adding to my knowledge.

Michael Regalbuto
Houston, 1990