Robotica

A comparison of rehabilitation robotics languages and software.

William S. Harwin[1], Ray G. Gosine[2], Zunaid Kazi[3],
David S. Lees[4], and John L. Dallaway[5]

1. SUMMARY

There is a wide diversity in the functioning and programming of robots designed and programmed to assist individuals with disabilities. The planning and structure of four rehabilitation robot implementations is presented. The first is the CURL language developed for human interface and the most widely used in this field. The second MUSIIC explores methods for direct manipulation of objects. RoboGlyph uses symbolic constructs to assist with the direction and programming of rehabilitation robots and finally a multi-tasking operating executive is discussed that controls a bilateral head operated telerobot. These four implementations reflect a wide range of interface concepts for the intended users.

2. INTRODUCTION

Over that last 20 years there have been a number of different approaches to the application of robotic technology that attempt to meet the needs of individuals with physical and learning disabilities. The areas considered for the application of robot technology include daily living tasks such as personal hygiene and assistance with eating1-3, educational tasks such as block play and laboratory work4-6, and vocational tasks such as manual assistance in office environments7-10 and robot-assisted assembly11.

In rehabilitation robotics, and in particular in educational and vocational applications, the aim is to replace the disabled person's manipulation deficiencies and to enable the person to partake in activities requiring manipulation. Thus rehabilitation robots in educational applications seek to put the student in charge of an environment. Educational science experiments are especially appropriate as a robot allows the student the full abilities to make a hypothesis, conduct experiments and draw conclusions, thereby gaining immediate information from the natural world4-6.

The high cost of developing purpose built robots has resulted in many projects utilising small industrial robots to evaluate rehabilitation applications. Whereas there is a high level of sophistication and reliability in these machines, they possess characteristics that are undesirable in a rehabilitation setting. These limitations include the inaccessibility of the low level control loops, the paucity in the programming language for interactive work and the difficulty with interfacing with human and environmental sensors. The languages and software discussed in this paper have been developed in an attempt to resolve these issues.

A small number of purpose built rehabilitation robots now exist9,11-13 but only the Neil Squire arm has the ability to be programmed. This is a desktop device whereas the other commercial devices are all intended to be mounted on a wheelchair. This dichotomy illustrates the combined needs expressed by envisaged users that robots should be easy and intuitive to use yet should be available to them on all occasions. Cost issues and technological sophistication have not yet allowed these two needs to be resolved.

The programming languages written for industrial robots are not well suited to rehabilitation applications. The major emphasis in industrial robot programming has been on attaining a high degree of precision during the motion and minimal consideration has been given to the role that the human operator may play during execution of the task. In a typical industrial application, the human does not interact with the device while the task is being carried out and moreover it is desirable

1. Department of Cybernetics, University of Reading, Reading, England
2. Medical Engineering Group, Faculty of Engineering and Applied Science, Memorial University, Newfoundland, Canada
3. Applied Science and Engineering Laboratories, University of Delaware and the Alfred I. duPont Institute, Wilmington, Delaware, USA
4. Center for Design Research, Stanford University, Stanford, and Migration Software Systems Ltd., San Jose, California, USA
5. Department of Engineering, Cambridge University, Cambridge, England

to isolate the robot workcell for safety reasons to ensure that there is no robot-human interaction. When a robot is employed in an industrial setting it works independently, or in conjunction with other robots or machines, on the repetitive task that it has been taught or programmed to perform in a highly structured environment. The role of the human operator is restricted to reprogramming robot trajectory to suit the task, and initiating a shutdown when the robot malfunctions.

Commercial robots may be programmed using text-based, robot-level languages such as VAL, AML or RAIL which specify positions and orientations to the robot as a string of numbers. This approach is not intuitive, particularly for orientations, since it is difficult for a human to envisage the position of the robot by looking at a string of numbers. Given that these numbers must also relate the robot to objects in its workspace, that the operator probably does not have a university education, and that he or she will have a slow rate of data entry when compared with an individual without motor impairments, the difficulty of the interface becomes apparent. Since the basic components of conventional robot programming languages are not intuitive, better manipulation programs and techniques must be developed. The human user typically will know what he or she intends for the robot, but does not have a language that allows these wishes to be readily expressed to the robot. Further when the robot fails to carry out the user's wishes, the reasons for this failure and the strategies necessary to re-specify the task become an important auxiliary function for the human-machine interface.

Providing a functional method for human-machine interaction is a primary concern in developing a programming language or approach for rehabilitation systems. This paper discusses some of the issues to be addressed in the development of interactive rehabilitation robots system and provides an overview of several complementary developments. The limitations of these approaches and the outstanding issues for further consideration are also discussed.

3. REQUIREMENTS OF AN INTERACTIVE ROBOT SYSTEM

Batavia and Hammer14 used panels of disabled consumers to rate attributes of assistive technology and the panel studying rehabilitation robotics ranked effectiveness with the highest priority followed by operability. Effectiveness was defined as the extent to which the functioning of the device improves the consumer's living situation, as perceived by the consumer, including whether it enhances functional capability and/or independence. Operability was defined as the extent to which the device is easy to operate and responds adequately to the consumer's operative commands. Discussion panels at the Applied Science and Engineering Laboratories in 1993 concurred with these priorities and also included simplicity and portability as concerns that demand a high priority.

These attributes cover many factors in engineering design but with regard to the human-machine interface, the system design must be to allow the person to exert ultimate control over the robot in a form which is appropriate to his or her disability and for activities that have value for that individual.

Four characteristics are desirable in a rehabilitation robotic system

- the system should support a flexible user interface.
- the robot should be able to operate in an environment which lacks structure.
- the system should be designed for the easy specification of tasks.
- the disabled person must have both high- and low-level control of the robot.

These requirements are consistent with the more general requirement that a rehabilitation robotic system operating in a real-world domain must be adaptable, flexible, and cost effective as well as fit with the cognitive, perceptual and motor skills of the human operator.

3.1 Need for a flexible user interface.

The requirement that the robot system must support a flexible user interface is shared with many other projects which are concerned with the use of robot and computer technology by individuals with severe motor disabilities. The interface can be considered as a bi-directional flow of information between the individual and the system. Information provided by the operator can be generated as any suitable body movement such as head movements, spoken utterances, gross or fine movements of the fingers, hand or feet. The measurement of these movements can be achieved and interpreted via a continuous flow of information or as discrete timed events. Examples of the former include pointing with head or hand movements, or operating a wheelchair joystick. The latter may be information derived from an augmentative communication device, an adapted keyboard, or specialised switches. Emulator software allows one form of input to be multiplexed, so that a single switch can be used to select from a full lexicon of symbols, or converted so that continuous information can select a single symbol and discrete events can be integrated to produce continuous information flow.

The flow of information from the system to the user may come from the direct action of the robot in the environment or via indirect indications of the robot state via sounds and spoken utterances, visual displays of icons, animations or overlayed video images, or kinesthetic and tactile feedback.

3.2 Need to operate in an environment which lacks structure.

A practical rehabilitation robot system must be capable of operating in an environment which need not be rigidly structured. Four levels of operation can be considered, 1) the system constructs and utilises an internal model of the environment based on intelligent sensors, 2) the human-machine interface can be arranged so that this model can be supplemented by information provided by the operator, 3) the robot sensor information can be relayed back to the user in a suitable form, or finally 4) the system can rely totally on the human for all environmental sensing.

When an internal model is necessary the robot must learn about its environment, and maintain and use a representation of the environment as an activity proceeds. An appropriate sensory capability is necessary to ensure the model is sufficiently accurate for the workspace. In addition, the robot must be sufficiently flexible to adapt and react to inherently uncertain and physically unpredictable situations15-16.

A practical system should utilise the human intelligence provided by the operator while maximising and utilising appropriate machine intelligence which may be developed for the system. For example, while educational activities with young children may be highly structured, the environment in which they are carried out may be unstructured. An unstructured environment may be necessary to preserve the educational value of an activity and to maintain the interest of the child. Vocational rehabilitation applications may also require the robot cope with variability in its environment. It is necessary for the robot, through its control software, to be able to cope with such changes with guidance from the operator. The cognitive and physical workload of the human operator can be reduced by introducing sufficiently advanced robot planning and reasoning techniques while providing necessary checks and diagnostic procedures. In remote time-delayed work sites, autonomous planning is necessary to perform a major part of the tasks. In the rehabilitation domain, user load can be reduced by autonomous planning of low level tasks such as path planning and basic object manipulation.

The advantage of using an internal representation of the environment is that it is then possible for the operator to issue high level commands and expect the robot to accomplish these. The interface is natural, and has a low physical cognitive burden for the operator. The disadvantage is that the system costs tend to be greater and the system reliability is reduced requiring operator intervention when a task is not accomplished.

The opposite is true when an internal model of the environment is not available to the operator. Although system costs tend to be lower and systems more reliable, the operator is continuously responding to information either directly conveyed by the interaction of the robot in the environment or via an appropriate representation of the data that is acquired by the robot sensors.

3.3 Need for the easy specification of tasks.

The third requirement of the robot system is that it support the easy specification of robot tasks. A simple, rapid and reliable method for specifying robot tasks is important to the success of robot systems, not only in educational and vocational applications, but in all areas of rehabilitation robotics. A high level approach requires the operator to specify robot applications in terms of the actions that the robot is to perform on objects in its environment, rather than in terms of the robot's position and orientation at various instances throughout the task. This task-level approach is favoured over more robot-oriented approaches to robot programming and control. Since the robot knows or learns about its environment, robot tasks may be specified using short programs comprising very high-level commands, and less information pertaining to the details of the robot's motion needs to be specified. The elimination of much of the technical detail from robot programming opens the way for an elementary approach to application development which is particularly beneficial in rehabilitation robotics settings since it is likely that the applications will be developed by non-technical support staff with little computer programming experience.

Even in systems that do not attempt to model or use information from the environment there is a need to specify tasks to reduce the physical and cognitive burden on the operator. Thus under particular conditions the robot might move to an area of interest, such as a work surface, a set of tools, or near to the person's face. The robot may switch modes to assist the person with different types of movement such as the trajectory to pour water from a cup, a velocity mode to allow large ranges of movement or a precisions mode to allow fine movements to be achieved.

3.4 Need for both high- and low-level control of the robot.

Independent of the level of complexity of the robot the operator should be able to interact with the system at a variety of levels. This is required to ensure that the operator can be successful in aided or augmentative manipulation strategies that might be subject to a high level of error or not be part of an original set of programmed tasks. Thus a system providing for low level commands (i.e. manipulator-like control of the robot), high level commands (i.e. task-level instructions) and a 3-D direct manipulation interface (i.e. instruction by gestures to real-world objects) would represent a flexible system which provided ultimate control to the operator.

4. THE USER INTERFACE FOR REHABILITATION ROBOTICS

A human-machine system (HMI) can be considered as the process by which a human operator indicates an intention to the machine and receives feedback regarding the machine's operation. In rehabilitation robotics, three categories of users, each having requirements for the HMI can be identified, a commissioning engineer, the non-technical care worker who would be expected to assist in the development of applications, and the individual with motor disabilities who will operate the system on a daily basis.

The complexity of many systems demands that an engineer be involved in the initial configuration of the robot and its peripherals. The engineer may require complete access to the robot controller at the lowest level to undertake calibration and alignment procedures. Low-level system access, however, may not required by other users and its provision may even be considered dangerous.

Those robotic systems which accommodate the programming of robot motion sequences or tasks require the generation of installation-specific task definitions prior to deployment. The creation and subsequent modification of these tasks may be undertaken by a carer or therapist who will not, in general, be familiar with robot technology. The provision of an appropriate interface for this class of user is critical to the viability of a robotic system as a genuine product and the obstacles to a successful implementation should not be underestimated.

The final class of user is the end user, the individual for whom the robot will provide assistance. The HMI requirements for a specific end user are influenced by many factors. Significant factors include the manipulative ability of the individual, their familiarity with particular interfacing technologies and the compatibility of these technologies with existing equipment. Large variations in end user HMI requirements dictate that robotic systems should accommodate multiple and disparate HMI options wherever practical.

Factors which must be addressed when developing a HMI include ergonomics, signal processing, information representation, and the human cognitive model. Ergonomics factors range from the choice of input device and the medium for information feedback to the seating arrangements for the individual. Signal processing problems concern the transformation of the input signal into a form acceptable for driving the underlying software system and processing internal system data for suitable feedback to the operator. A typical example might be the production of relevant information for a text-to-speech feedback system, or the graphical representation of the robot pose or its conception of the external environment.

A vital consideration in the HMI is the human cognitive model of the robot. This model determines the nature of commands that will be issued by the operator and the expected behaviour of the system based on the person's previous experience. The cognitive model that the person has prior to direct experience using the device may be very different from the model which is held after use. For example, based on exposure to robots in science fiction films, the term robot may conjure up images of an autonomous android capable of almost any task. Experience using current rehabilitation robot systems will undoubtedly cause conflicts with such a cognitive model.

An unresolvable conflict between the person's cognitive model and the actual operation of a device, in addition to flaws in the technical aspects of the interface, can lead to a rejection of the device. This is not to say that an initial conflict between the person's cognitive model and the operation of a system renders the system unacceptable. A successful interface is one which reforms the cognitive model in order to enable the person to exploit the function of the device.

The use of computers as an assistive technology for people with physical disability is widespread, and many potential users of robotic devices already have skills operating personal computers. The use of graphical user interfaces (GUIs) among disabled users is also widespread and it is natural to consider transferring this metaphor to implement the HMI for a robotic device. The consistency with which graphical environments access computer hardware has facilitated the development of software-based access utilities. These utilities enable individuals who cannot use a keyboard and/or pointing device to gain full control of all application software. It is not, therefore, necessary to make provision for every conceivable disability that might be expected of the potential user. Direct manipulation of graphical computer interfaces is now within the reach of many users with motor disabilities.

Direct manipulation GUIs have become the predominant paradigm for human computer interfacing in the 1990's. Direct manipulation systems exploit intuitive common sense skills that are developed throughout the life of the individual by transferring them to operate on semantic task representations in a natural and transparent way. This allows individuals to utilise frequently used and highly developed skills such as those involved in manipulating real physical objects (e.g., the desktop metaphor) to learn to manipulate semantic task representations easily and with few syntactic constraints. The user can concentrate on task solutions, rather than expend cognitive effort in maintaining representations of the computer's structures and processes and syntactic rules for forming commands. Direct manipulation GUIs have been shown to reduce learning time, error rates and to increase skill retention and subject satisfaction of users[17].

Early rehabilitation robotic systems used command line interfaces with voice-recognition to allow the user to instruct the robot[18,19]. This is a situation reminiscent of early computer interfaces which allowed dialogue through a syntax con-

strained command line interface. In the direct manipulation robot interface, the user concentrates on the task semantic representation and direct manipulation of physical objects, rather than the internal representation of the robot and computer controller. People without disabilities can easily identify, manipulate and transport objects while taking the environmental context into account. Employing a direct manipulation approach to the control of an assistive robot brings advantages similar to direct manipulation two-dimensional GUIs where manipulation acts can be linked to processes internal to the computer or system being controlled. Any system that by-passes a command line interface and permits a natural human-task dialogue would reduce fatigue and facilitate reliable and fast operation of the robot, an important consideration if the robot is in constant use by an operator with motor impairments.

5. SOFTWARE REQUIREMENTS FOR INTERACTIVE ROBOTICS

The breadth of the user interface requirements is one unique aspect that distinguishes rehabilitation robotics, the range of tasks is another. Because the field of interactive and rehabilitation robotics is relatively young, software must be structured to accommodate a range of interfaces and applications that have yet to be defined. Although the software structure may be equally applicable to developing autonomous robots, the inclusion of a non-technical user in the system means that software must be safe and robust yet still maintain a level of generalisation so that new tasks and user requirements can be accommodated.

Four requirements for software have been identified, programming flexibility, task representation, information flow, and an interactive interface.

The software must allow the system to be applied easily to a wide variety of activities, many of which have not yet been proposed. Given that rehabilitation robot research is also concerned with the identification and evaluation of potential tasks, it is logical to develop software that can be readily adapted to a large range of tasks by non-technical application developers. To achieve this goal the system interface components of the software must be easily extended and modified in order to cope with hardware changes, while the low-level components must provide the flexibility and programming power necessary to incorporate motion planning and control strategies based on information provided both by sensors and the disabled operator.

Where applications are developed the software needs to be able to facilitate the rapid specification of compact task specification or application programs. Care must be taken, however, in order to ensure that the system does not become so high-level that it loses generality. An efficient representation results in a readable application code that promotes desirable features such as rapid development, process visibility and easier code debugging thereby leading to higher levels of reliability and safety.

The flow of information and data between software modules facilitates rapid program and task development and helps to maintain programming flexibility. Two approaches have been used. CURL utilises the standard inter-process communication mechanism of the Microsoft Windows operating environment, (the Dynamic Data Exchange (DDE) facility) to receive user input from third-party software utilities. For example, GUIs running under Microsoft Windows can receive user input from third-party software utilities20. In generic terms, a rigid adherence to the standards laid down within a particular graphical operating environment serves to enhance the flexibility of robot application software and thus extends its scope.

Data flow in the MUSIIC environment is accomplished via remote procedure calls (RPCs). Although originally developed for unix systems, software is available that allows programs running under other operating systems to access this protocol. Thus MUSIIC is able to manage data from the image and sensor system running under unix, with a voice recognition system running in Microsoft Windows, the robot running in DOS, and a second unix system used to integrate the information.

Neither the DDE facilities nor the RPCs have well defined execution times so for strict real-time operation a third approach was demonstrated in the rehabilitation telerobot test-bed where generic C++ queue structures were used to allow tasks to exchange data with low latency.

Finally it is necessary to promote software interaction, not only for the commissioning engineers, the application developer but also to allow the operator to rapidly acquire experience from the system and evaluate its benefits and defects. All three types of users gain confidence and understanding of the system if the software can be designed to be interactive, as the understanding of the system is reinforced or can be reevaluated depending on the response it gives.

6. THE CAMBRIDGE UNIVERSITY ROBOT LANGUAGE (CURL)

The Cambridge University Robot Language (CURL) serves as an example of a flexible robot programming environment designed for workstation activities in rehabilitation. It is a commercial package running on PC-compatible hardware under the Microsoft Windows operating environment. CURL is structured to separate the device specific code from the

human-computer interface and command processor (figure 1). It is therefore possible to write device-independent CURL procedures for subsequent execution on a robot of unspecified geometry. Device independence is achieved by rigidly specifying the messages passed across a software interface between the command processor and the device-level control software. Microsoft Windows provides an effective platform for the implementation of such an interface. It facilitates the development of robot drivers as separately compiled Dynamic Link Libraries (DLLs). CURL Device Drivers (CDDs) are available for all RT-series robots manufactured by Oxford Intelligent Machines Ltd (Oxford, England) and the CDD specification has been placed in the public domain to facilitate third-party driver development.

The CURL command processor consists of a lexical analyser, a parser and an interpreter20 . Commands and procedures are entered directly by the user or retrieved from file as raw text. The lexical analyser converts the text into token form for parsing. The parser uses the tokens to generate an intermediate representation which is then interpreted. Each element of the command processor accesses a symbol table containing data element definitions. The use of an interpreter facilitates a level of user interaction not attainable with a compiled language. CURL procedures may be assembled, tested and modified 'on screen' before saving to file. Direct control (telemanipulation) operations may be interleaved with CURL commands to provide a particularly intuitive program development environment. The CURL syntax incorporates elements of natural language to enhance code readability and encourage program modification by non-technical personnel.

A key feature of CURL is the ability to program at the task level by making use of an integral world model. Specific CURL keywords facilitate the declaration of objects and locations of interest within the world model. The names of these data elements may be qualified and compared using pre-defined adjectives. For example:

object block is red.

the red block has location 100 200 300 45 0 0.

Robot motion commands can then be constructed with reference to defined objects and locations. The following task-level commands illustrate some of the possible motion statement constructions:

move the red block to the box.

move 100 to the left of the red block.

move to the box via the home_position.

The action of the robot in response to a task-level command is dependent on state information. In the case where the specified object constitutes the current payload, the robot will move to the specified destination. If the object is located elsewhere, a more complex path involving object acquisition will be executed. Collision avoidance is achieved using goal post motion as necessary.

The construction of CURL procedures is illustrated as follows:

procedure peg_to_hole #peg.

move the peg 30 above the hole.

move to the hole directly with speed 20.

release.

move up 30.

end procedure.

In this simple example, a peg object is specified as a procedure parameter. The robot grasps the peg and moves it at the default speed to a point 30mm above its destination. The peg is then inserted into a pre-specified hole at reduced speed and released. Finally, the end effector is withdrawn and the procedure terminates.

While the interpreted nature of the CURL command processor prohibits the programmatic processing of sensor information in real time, sensor integration may be achieved within a CDD. This approach has been used to provide sensor-based modification of robot trajectories as part of an experimental shared control strategy21. The CDD specification accommodates device-specific auxiliary axes which may be mapped to I/O ports within the CDD. It is therefore possible to command peripheral equipment from CURL. The ability to read input ports also facilitates program flow control using external transducers.

The standard CURL user interface provides user interaction via three child windows (figure 2). One window provides access to the direct control features of CURL. Another window allows direct entry of CURL commands and a third window lists pre-defined CURL procedures for immediate invocation. These child windows may be displayed concurrently to facilitate rapid switching between the various forms of control. Alternatively, those aspects of the interface which are not required for a specific installation may be reduced to iconic form. The novice user is therefore less likely to be distracted by irrelevant display elements. The menu bar provides access to lesser used configuration facilities.

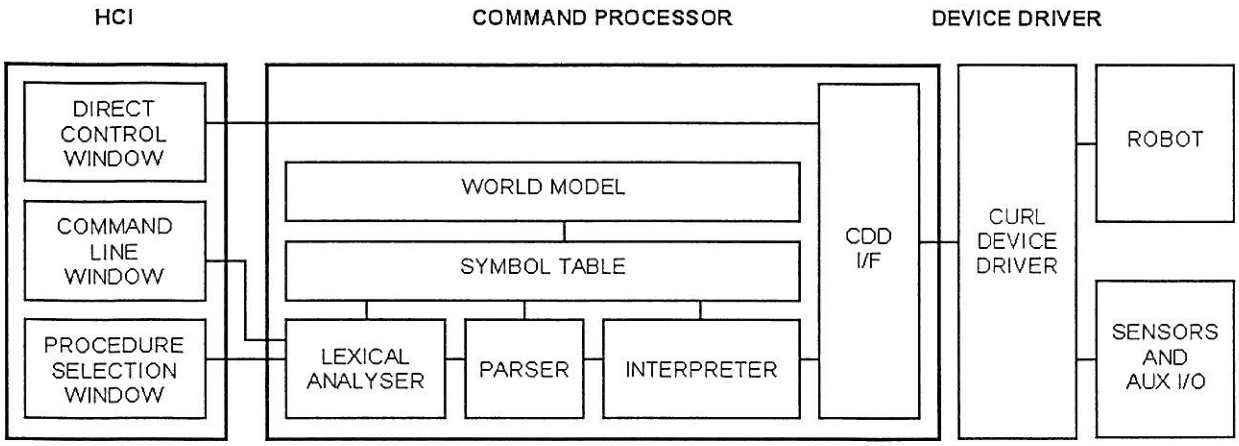

Fig 1. The CURL software structure

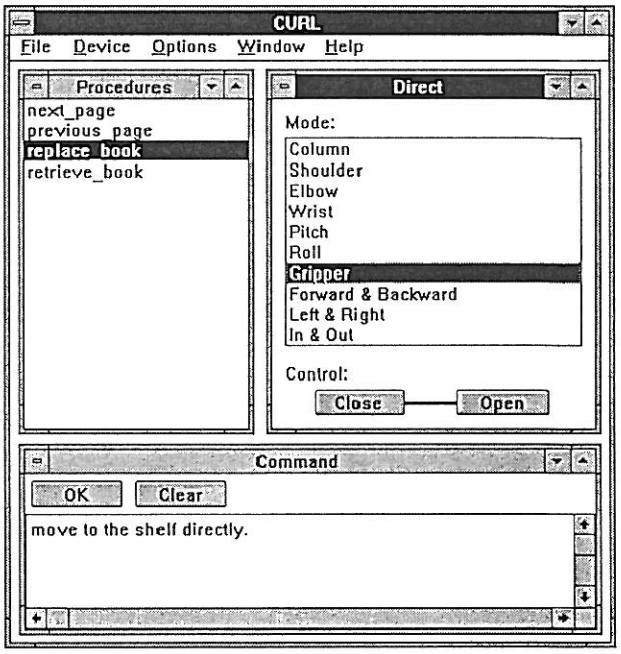

Fig 2. The standard CURL user interface

[Figure 1. The CURL software structure.]

[Figure 2. The standard CURL user interface.]

7. AN ENVIRONMENT FOR PROGRAMMING A ROBOT WITH GRAPHICAL SYMBOLS (ROBOGLYPH)

RoboGlyph has been developed in order to address the need to provide a robot programmer with a more comprehensible and efficient way of writing programs in comparison to conventional robot programming languages22. The software preserves the ability to generate printed (or on-screen) histories of the motions which are valuable for documenting and reusing code. One of the primary goals in developing RoboGlyph was to allow robot users (rather than robot experts) to modify existing robot programs or to create simple new programs themselves. For healthcare and rehabilitation applications, this is an important step toward making robots more cost effective.

With RoboGlyph, icons are used to represent the different configurations that the robot can assume and the motions that it can perform. The icons provide a visual indication of the behaviour of the robot and are directly manipulated by the

user to create programs. These characteristics help to make the computer interfaces more accessible to users.

Because of its iconic interface, a particular implementation of RoboGlyph must be associated with a particular robot geometry. However, because of the inherently modular nature of the iconic interface used in RoboGlyph, it would not be difficult to adapt RoboGlyph to a different robot geometry, though this is a task which would need to be performed by an installation engineer rather than the end-user. It is necessary only to create a new set of icons and associate them with appropriate robot behaviours. These behaviours are typically specified in a high level textual language (VAL in the current implementation) and are therefore fairly geometry independent.

Since the operator of the system will be present to create the manipulation program and watch it run, programming labour is divided between the robot and the user so that both may perform the tasks that are best suited to their capabilities. In particular, the user functions as the robot's vision system. Since he/she can quickly position the robot in the vicinity of an object, the RoboGlyph system does not use machine vision for the task. On the other hand, precise positioning of the robot is an excessive burden on the programmer. Because of this, force feedback is an integral part of the RoboGlyph environment. Once the user has positioned the robot near an object, the force sensor is used to determine the object's orientation and to facilitate its manipulation.

The test-bed for the development of RoboGlyph was the DeVAR8 vocational robot system employing a PUMA-260 robot. The standard VAL controller performs the low level control needed to move the robot from one location to another, including computing the kinematics for straight-line motions.

The hardware configuration used to test RoboGlyph is shown in figure 3. The NextStation runs the user interface and supervises the overall operation of the system. It is connected to a Unival robot controller which handles low level position control of the robot, including computing inverse kinematics and controlling straight-line trajectory planning. The PC is a real time controller for force-controlled motions. It processes data inputs from the force sensor and computes and controls the robot's trajectory.

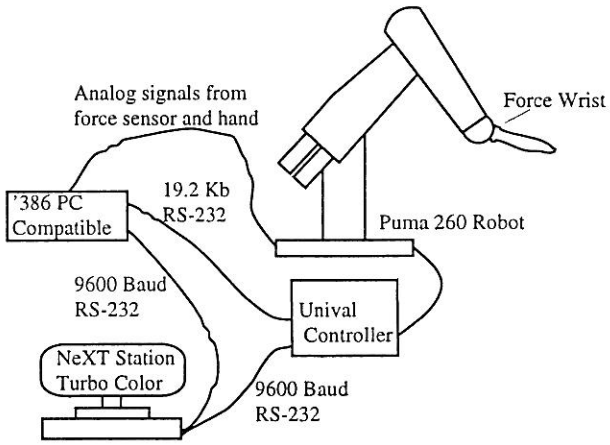

Fig 3. DeVAR hardware diagram

[Figure 3 - DeVAR Hardware Diagram]

RoboGlyph's software architecture is based on a "storyboard" or "comic strip" metaphor. A robot program is composed by placing icons representing robot positions and actions into successive cells of a storyboard. There are two main classes of icons in RoboGlyph, those associated with position control and those associated with force control. The position icons move the robot to a particular pose (which is shown dynamically by the icon). Each position icon is associated with a command which will move the robot to a particular location. The speed of the motion is also associated with the icon and can be set by the user. Force controlled icons initiate more autonomous motions, e.g. plane-finding, that are intended not only to free the robot's user from performing fine positioning tasks manually, but also to make robot programs more robust by allowing them to adapt automatically to variations in their environment. As with position icons, force icons have a set of parameters associated with them which can be changed by the end user (by selecting the icon and opening an "inspector" window). The selection of force controlled icons was based in part on the force-controlled object exploration work of Allen and on the force-driven manipulation architecture proposed by Craig. At present there is no explicit interaction between RoboGlyph icons, except that two position icons with a "straight-line" icon between them will cause the robot to move on a straight path between the two positions.

Choosing a position and orientation for the robot is at the core of the RoboGlyph system. Positions and orientations are chosen from "palettes" (figure 4) that display a collection of icons representing various poses of the arm in space.

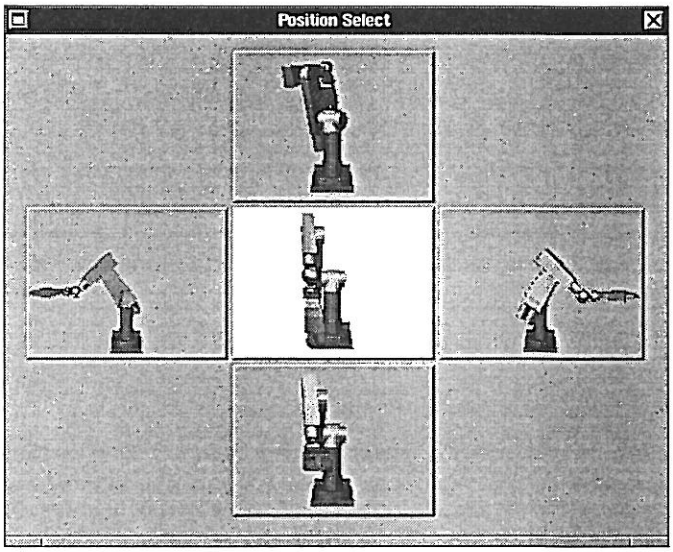

Fig 4. RoboGlyph pose selection pallet

[Figure 4 - RoboGlyph pose selection palette]

Double clicking of the mouse on any icon will cause the robot to move to the chosen pose. A single click anywhere in the palette will stop the arm, and fine tuning can be performed using the mouse. The pose palettes are used for coarse positioning of the robot which is needed before activating force driven exploration of the environment. RoboGlyph exploits the fact that many robots allow for the decoupling of position and orientation, and primitive sets of motions have been combined to create manipulation programs. The motion primitives are represented by icons (figure 5) that allow the various types of motions to be clearly understood by the user. These include primitives such as rotation about an axis (e.g. swinging a door), constrained motion (e.g. inserting an object into a slot, contour following) and straight-line motions. As described above, motions have the ability to be constrained interactively. For example, a rotation can be initiated and then stopped when the door has opened by a desired amount.

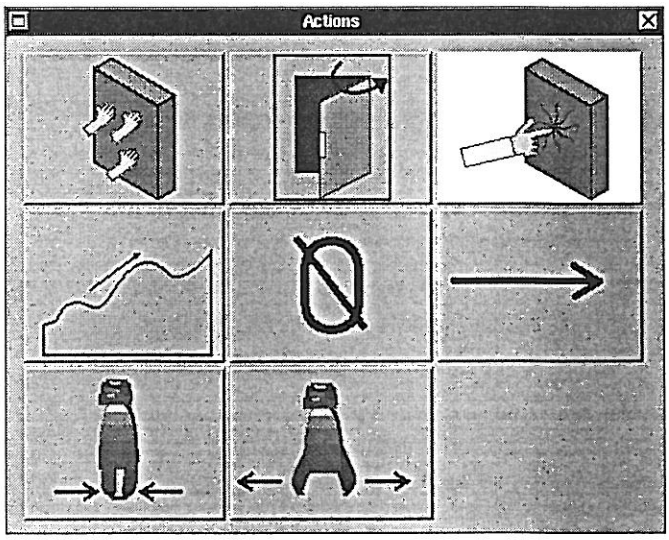

Fig 5. RoboGlyph motion and exploration primitives

[Figure 5 - RoboGlyph motion and exploration primitives]

A motion storyboard is a graphical equivalent of a program listing. It shows the nature and sequence of the operations

in a manipulation program by using graphical representations of the motion primitives. A program is composed by dragging the icons onto a storyboard from the palettes. For each frame of the storyboard the current configuration of the robot is shown giving the user a pictorial history of the actions taken to execute a task.

Figure 6 shows a motion storyboard with a fragment of a RoboGlyph program. The figure illustrates the use of position icons and a guarded (force limited) move to press a button which opens a door. The position icons were created by constraining the robot to the plane of the door using the plane-finding action and then using the mouse to guide the end-effector near the door button. Then the force guarded move was placed in the storyboard to complete the door opening operation. This design allows the storyboard to function successfully even if the position of the button changes slightly after the program is created.

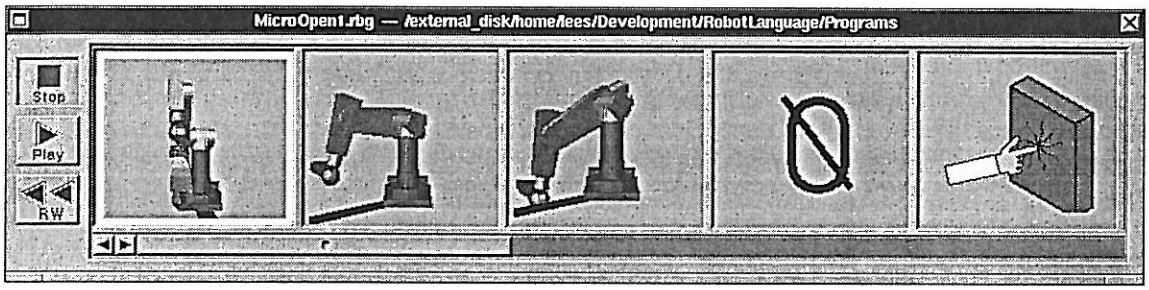

Fig 6. Extrct from a RoboGlyph motion storyboard

[Figure 6 - Extract from a RoboGlyph Motion storyboard]

8. A MULTIMODAL USER SUPERVISED INTERFACE AND INTELLIGENT CONTROL (MUSIIC)

Based on experience at ASEL regarding the objectives of CURL and its applications, the MUSIIC (Multimodal User Supervised Interface and Intelligent Control) system was developed as a novel framework for the control of an assistive interactive telerobotic system for operation in a real world domain by an end user. By combining current state of the art in natural language processing, robotics, computer vision, planning, machine learning, and human-computer interaction, it is possible to build a practical robotic assistant for people with disabilities without having to solve the major problems in each of these fields, i.e., full text understanding, autonomous robot arm control, real-time object recognition in an unconstrained environment, planning for all contingencies and levels of problem difficulty, speedy supervised and unsupervised learning, and intelligent human-computer interfaces. This is possible because current solutions to each of these problems, when combined with each other and with the intelligence of the user, can compensate for the inadequacies of each individually.

In a three-dimensional direct manipulation interface for unstructured environments, rather than having screen objects which are cursor addressable using a mouse, the user points to physical objects in the world via gesture, and specifies that certain manipulatory actions be performed on the objects. In order to operate, however, the system must have the ability to quickly and reliably identify the objects and their position and orientation in the environment.

Enabling the computer and robot to be aware of the identity and pose of a physical object so that direct manipulation may be carried out by the robot system introduces significant perceptual and motor bottlenecks. The computer's internal representation of the domain must be updated with respect to the physical object's identity, shape, pose and location, as well as any constraints that might be associated with the manipulation of that particular object. These attributes are not immediately accessible as they are in the case of a window manager with screen objects and associated data structures through which they can be rapidly indexed. In principle, a highly reliable and rapid machine vision system would provide the necessary recognition and pose determination for objects, but this is currently far beyond the state of the art and even if it were, might involve significant costs in terms of processing time and system complexity.

Furthermore, actions specified by the user, such as the equivalent of dragging a screen object, would require grasping and transportation of real objects. Each of these processes is influenced by the world state, while in the case of a screen representation, such processes tend to be context independent. A human invokes highly sophisticated path planning abilities when planning trajectories for objects. Much work has been done in trajectory planning and plan synthesis in the robotics and AI planning communities, but practical, rapid and highly autonomous systems are still a long way from practical reality. One approach towards mitigating the requirements for perceptual and planning systems to support a direct manipulation system is to utilise a multimodal interface to combine input evidence from a user dialogue. This permits

perceptual and planning requirements of the system to be relaxed to the point where existing semi- autonomous techniques are sufficient to carry out tasks and make the system practical. By engaging in dialogue with the user in such a way that natural deictic gestures and voice input can be used to carry out a task, the system gains many of the advantages present in direct manipulation interfaces. The user can directly designate objects and locations in the environment around him/her, and use natural language to describe the desired actions on those objects and locations. By combining different modalities, rather than attempting to constrain dialogue to one modality, great simplification of processing can be accomplished, as has been demonstrated by several multimodal systems that have been developed for graphical user interfaces23,24 This simplified processing allows for less delay in the processing of user interaction, which supports faster system response to user actions, that has been demonstrated to improve user task completion times and to result in less frustration17.

MUSIIC includes a knowledge driven planning subsystem in which objects are represented in an increasingly specialised sequence of object classes in an inheritance hierarchy 25. There is also a user extendable library of plans ranging from primitive robot handling tasks to more complex actions such as feeding. A multimodal human-machine interface and object-oriented representation allows the user to interact with the planning system at any level of the planning hierarchy, from low level motion and grasp planning to high-level task planning of complex tasks. Synthesised plans are supplemented by user intervention whenever incomplete information or uncertain situations prevent the development of correct plans. This is done by taking over control of the planning process or by providing necessary information to the knowledge bases to facilitate the development of a plan capable of handling a new or uncertain situation. Furthermore, incomplete sensory information may be supplemented by user input, enabling the planner to develop plans from its plan library without the need for extensive user intervention.

The basic architecture for MUSIIC is composed of three knowledgebases: a hierarchical knowledgebase of objects (WorldBase), a knowledgebase of objects in the actual domain of operation (DomainBase), and a knowledgebase of plans (PlanBase). The planner, which is based on a STRIPS-like26 planning mechanism, uses the three knowledgebases and user/sensor provided feedback, to construct robot plans.

Objects are represented in an increasingly specialised sequence of object classes in an inheritance hierarchy. At the top level, generic abstract objects give rise to generic plans in the absence of exact information. The second level includes classification by general shape (i.e. cylindrical, flat, spherical). The third level includes a general representation of commonly used everyday objects, such as a "cup" or a "can", and at the bottom level there are actual objects in the domain whose attributes are fully specified. This knowledgebase enables the planner to modify approach and grasp plans when more information as available for the object. Each object, depending on the degree of generalisation, has a set of attributes that will assist the planner in developing correct plans such as shape, size, dimensions, weight, approach point, grasp points, constraints and plan fragments. Additional information include the location and orientation, and attachment relationships to other objects and the workspace.

MUSIIC also includes a vision subsystem which facilitates the determination of the three-dimensional shape, pose and location of objects in the domain27. No object recognition is performed. The vision requirement is to provide the knowledge-based planning system with the parameterised shape, pose and location information of the objects in the immediate environment. A human-computer interface subsystem uses a multimodal input schema where users of the system use deictic gestures (pointing, achieved by a head mounted laser pointer) to indicate locations, and spoken commands to identify objects and specific actions. The combination of spoken language along with deictic gestures performs a critical disambiguation function. It binds the spoken words in terms of nouns and actions to a locus in the physical workspace. The spoken input is used to supplant the need for a general purpose object recognition module in the system. Instead, 3-D shape information is augmented by the user's spoken word which may also invoke the appropriate inheritance of object properties using the adopted hierarchical object-oriented representation scheme.

Prior to interaction with the user, the system sets up the DomainBase as a collection of entities with the shape and position with respect to the world origin identified by a vision system. Based on the premise that the user is in the planning loop, the user points to an entity and identifies it to the system. For example, the user may point to a specific blob and inform the system that this is a cup. The system then updates the attribute for the entity with attributes that it obtains from the WorldBase. The user may also identify the entity as a specific object, such as my-cup; in such a case, the system is aware of a specific object in the WorldBase which is known as my-cup and the entity in the DomainBase is replaced by the exact my-cup that the system knows, and the attributes of my-cup in the DomainBase are instantiated from the WorldBase and information obtained from the vision system. It is entirely possible that the user may not have identified any specific entity. In such a case, the system is only aware of the general shape, and the entity is identified at a certain degree of generalisation.

MUSIIC is illustrated using a simple annotated example of a task scenario that illustrates some of the capabilities of the system. As illustrated in figure 7, the system includes a vision subsystem (RoboEye) based on an SGI computer and Galileo graphics interface, a Zebra Zero robot and controller (RoboArm), a DragonDictate speech recognition subsystem (RoboEar), and the planning and knowledge base sub-system (RoboMind).

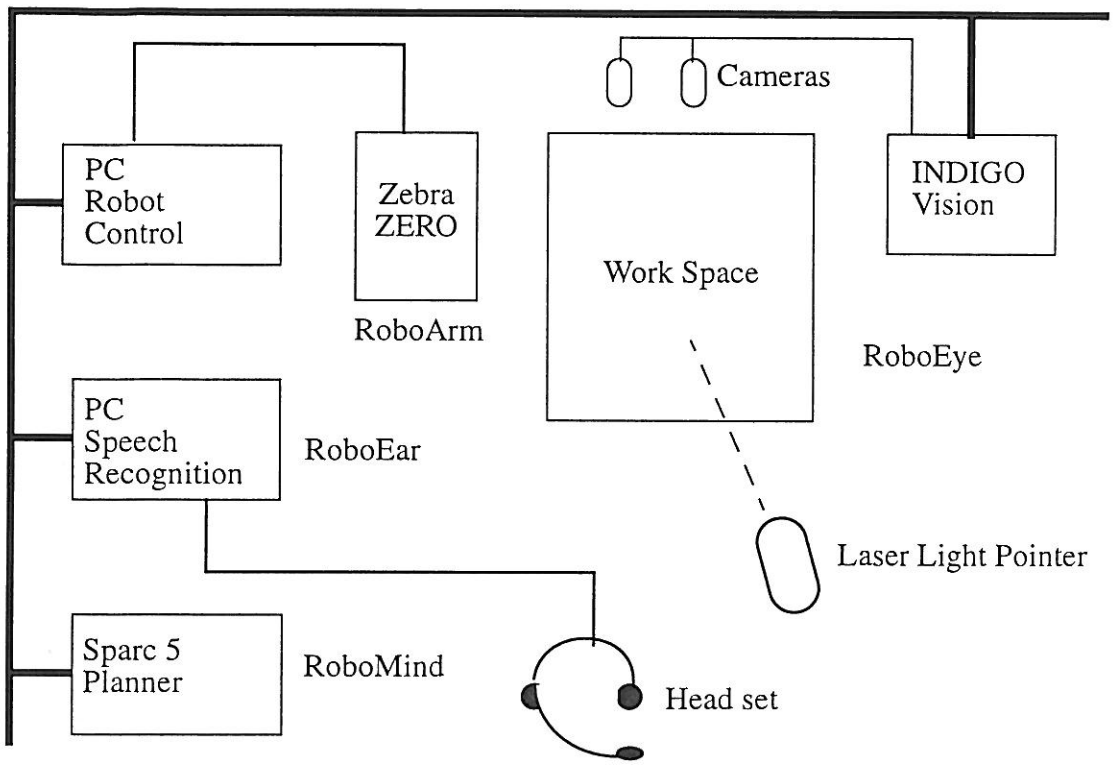

Fig 7. Musiic Experimental Set-Up

[Figure 7 - MUSIIC Experimental Set-Up]

The MUSIIC approach is illustrated for the task of inserting a straw into a cup and bringing the cup to the user. In this simple task scenario the workspace has a cup and a straw. The knowledgebase of objects contains information about "straws" and "cups". The following interaction between the user and the system illustrates the steps necessary to execute the desired task:

	Command/Response	**Comment**
User	"Load plans"	Instruct the system to load in the plan library.
MUSIIC	"Plan loading complete"	Following successful load of plan library.
User	"Synchronise"	Instruct the system to synchronise the various system components.
MUSIIC	"Synchronisation complete"	Following successful synchronisation.
User	"Home"	Instruct the robot to move to home configuration. The robot then moves to it's home configuration while the user observes the motion
MUSIIC	"Home successful"	Following successful homing procedure.
User	"Scan"	Instruct the system to scan the workspace. The vision system takes two images of the workspace using each of the two cameras and then generates object size, position, orientation and colour information, which is stored locally.
MUSIIC	"Scanning complete"	Following successful scanning of the work-space.
User	"Load domain"	Instruct the vision system to transfer the information to the planning subsystem. The system loads in the world knowledge base of objects and then transfers domain object data from the vision system to build up the knowledge base of domain objects.
MUSIIC	"Domain loading complete"	Following successful load of domain

	Command/Response	Comment
User	"That's a straw"	User points to the straw and identifies it to the system (see Figure 8). The user points to the straw while simultaneously saying the word "straw". This tells MUSIIC that the object in question is a straw. Information about straws contained in the world knowledge base is used to augment the information in the domain knowledge base.
MUSIIC	"Looking for the straw"	MUSIIC searches the world knowledge base for the "straw".
MUSIIC	"I found the straw"	After finding the straw in the world knowledge base and augmenting the domain knowledge base.
User	"That's a cup"	User points to the cup and identifies it to the system (see Figure 8). The user points to the straw while simultaneously saying the word "cup". This tells MUSIIC that the object in question is a cup. Information about cups contained in the world knowledge base is used to augment the information in the domain knowledge base.
MUSIIC	"Looking for the cup"	MUSIIC searches the world knowledge base for the "cup".
MUSIIC	"I found the cup"	After finding the cup in the world knowledge base and augmenting the domain knowledge base.
User	"Insert the straw into the cup"	Instruct the robot to move to execute the desired subtask. The system then looks up the plan library for the plan associated with "insert". If all conditions necessary for execution are met the system begins the execution of the task. The arm translates horizontally until it is above the straw. It then approaches the straw and closes the gripper on the straw's mid-section. The arm then moves up vertically and rotates the straw such that the straw is vertical. The arm then moves horizontally to a location on top of the cup. The arm moves down, releases the straw and then returns to the home position.
MUSIIC	"I am ready"	Indicates successful completion of the subtask. During the execution of the sub-task failures may have occurred. After each low level robot action, the system checks to see if the desired goal of the action has been achieved. On failure, MUSIIC tries to autonomously re-plan to correct the situation. For example, if the desired goal location for a motion task is not met, the system will generate another motion command to correct the situation. However, when "catastrophic" failure occurs, such as the gripper dropping the straw during translation, the system will engage in a dialogue with the user to re-plan. One of the preconditions for each task prior to gripper release is that the gripper is holding the object being manipulated. If the straw is dropped accidentally, MUSIIC informs the system that it is no longer holding an object. The user then must inform the system to rescan the workspace and resume the previously described sequence of actions. There are also situations when the system is unable to determine failures. One such case is when the cup may have tipped over during straw insertion. If this happens, the user may use low level arm manipulation commands to pick up and re-place the cup.
User	"Bring the cup"	Instruct the robot to move to execute the desired subtask. The system then looks up the plan library for the plan associated with "bring". If all conditions for executions are met the arm will then begin execution of this task. The robot will approach the cup and grasp it by the rim. It will then bring the cup to a predetermined position that is accessible to the user.
MUSIIC	"I am ready"	Indicates successful completion of the subtask
User	"Bring that"	Here we present an alternate scenario where the user did not explicitly identify the cup by saying "That's a cup". The user points at the spot while simultaneously saying the verbal deictic "that". The vision system continuously scans the work-space and keeps a record of any spot identified along with time stamp values that mark when the spot was present. The speech system also time stamps utterances whenever the parser recognizes verbal deictics such as "that", "there", and "it". During interpretation of "that", its time stamp value is used to determine the location of the spot that was generated when the word was uttered. The system then finds the object that is in that specific location and invokes the "bring" task on that object. Since the object has not been identified specifically, the planner would plan its task based on general principles. The entry for the "cup" object in the world knowledge base tells the system that the grasp position is the rim (computed from the diameter and centroid of the object). However, in this general case, no specialized grasp position is marked. Therefore, the arm will attempt to grasp the cup across its width, rather than at the rim.

Fig 8. Direct object manipulation in MUSIIC

[Figure 8 - Direct object manipulation in MUSIIC.]

9. A REHABILITATION TELEROBOT TEST-BED

Whereas the previous languages and software have concentrated on providing an individual with motor disabilities with a mechanism to command and direct a robot, there is evidence that direct correlation between the movement of the person and the movement of the robot provides a more robust, and intuitive mechanism to control a robot28. To achieve an apparently direct link requires close attention to the low level control structures of the software and although these may link with higher levels of command software identified in previous sections this discussion will cover the philosophy, control structure and software needs of a bilateral rehabilitation telerobot.

A bilateral telerobot attempts to achieve the perception of a physical and rigid link between the operator and the task. One common method to achieve this perception is to use the positions of the master control to drive the slave and apply the forces sensed by the slave to the master. This position forward/force reflection scheme has been studied in the context of haptic displays in virtual reality29, telemanipulators30,31 and human-machine interfaces32. A major problem of this type of interface is the understanding of the combined system response of the human, telerobot and environment - all three subsystems are essentially non linear. Because stability criteria of the combined system are poorly defined there has been a trend in recent research to look at impedance methods for the master-slave subsystem as this provides a better model for understanding the complex interaction33.

Despite the problems of bilateral telerobots they have considerable perceptual advantages to the operator allowing less dependence on visual channels of feedback, permitting the operators power to be scaled up or down and in the case of rehabilitation applications, providing a method of interfacing head-movements, limited hand movements or foot movements to an arbitrary set of slave responses.

Rehabilitation telerobotic device are an appropriate technology for individuals with spinal cord injuries where the traumatic spinal damage affects both fine motor skills and sensory channels. When a spinal cord injury level is between C2 and C5 there may be some limited hand function, although typically the individual retains a near to normal range of head movement34, thus head movements are an obvious candidate for telerobot operation. A telerobot may also be an appropriate technology for individuals with arthrogryposis where there is a need to map a limited range of hand input motion to a full range of motion at a remote site.

To explore the issues relating to a rehabilitation telerobot a test-bed was designed and built. This test-bed is a bilateral head operated telerobot intended to explore teleoperation in both position forward force feedback implementations, and impedance based implementations (figure 9). The test-bed consists of two 6 degree of freedom master and slave robots, each controlled by separate IBM-PC 486DX/66 p.c. compatible computers with a high-speed parallel data communications link. The master is the PerForce hand-controller (manufactured by Cybernet Systems of Ann Arbor, MI) that has been mechanically modified so that it is controlled by the user's head movements. The manipulation tasks in the environment are performed by the Zebra-ZERO (manufactured by IMI, Berkeley, CA). Both the master and slave are fitted with a 6 axis force/torque sensor. The slave force sensor is mounted in the robot wrist while the master sensor is installed in the user's helmet. The two robots have dissimilar kinematics thus ensuring that the algorithms developed on the test-bed achieve a high level of generality.

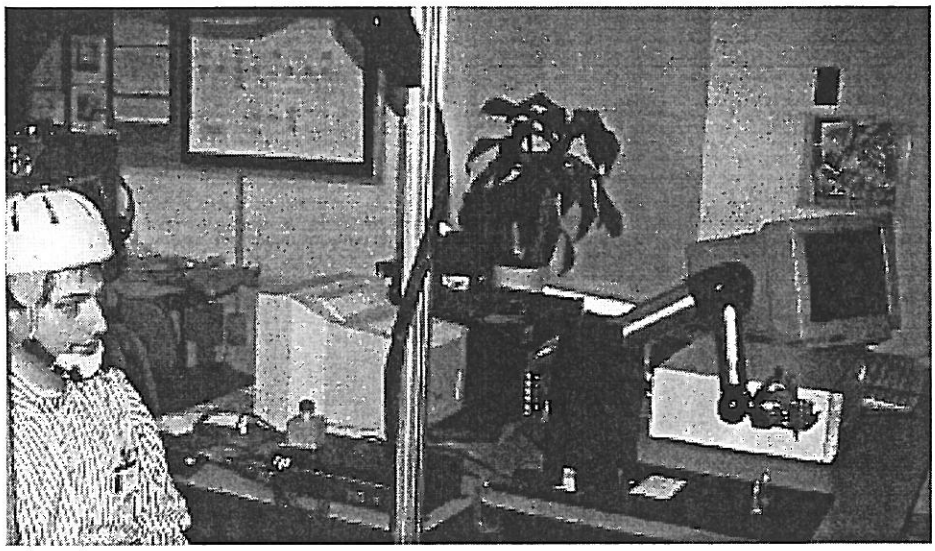

Fig 9. Configuration of telerobotic system.

[Figure 9 - Configuration of telerobotic system.]

9.1 Software structure

Software running on the master and slave attempts to enforce the control structure shown in figure 10. The software must run synchronously and the overall system timing is set by the master.

An important design and safety criteria is to maximise the visibility of the real-time software. This not only allows rapid debugging but also improves the flexibility of the system to allow several control architectures to be evaluated. The primary method of achieving controller visibility is establish a queue common to all processes that stores robot state information during operation and downloads it to the disk or screen once the real-time phase is completed.

Because the master and slave are kinematically dissimilar it is necessary to detect and avoid singularities. The design of the master is such that singularities are mechanically avoided. To achieve a reasonable working envelope the same approach would not be possible with the slave but instead the slave software warns the operator when the robot is within a preset "distance" of the singularity. Two warnings are provided, an audible bell and a phantom force that is added to the force sensor reading and has the effect of pushing the user away from the singularity.

The software has been designed as a number of self contained tasks that exchange information. Information is exchanged via fixed length queue structures that have a ring structure thereby allowing previous system states to be accessed. Two queues are special in that they queue information for exchange across the two control computers via the high speed parallel bus.

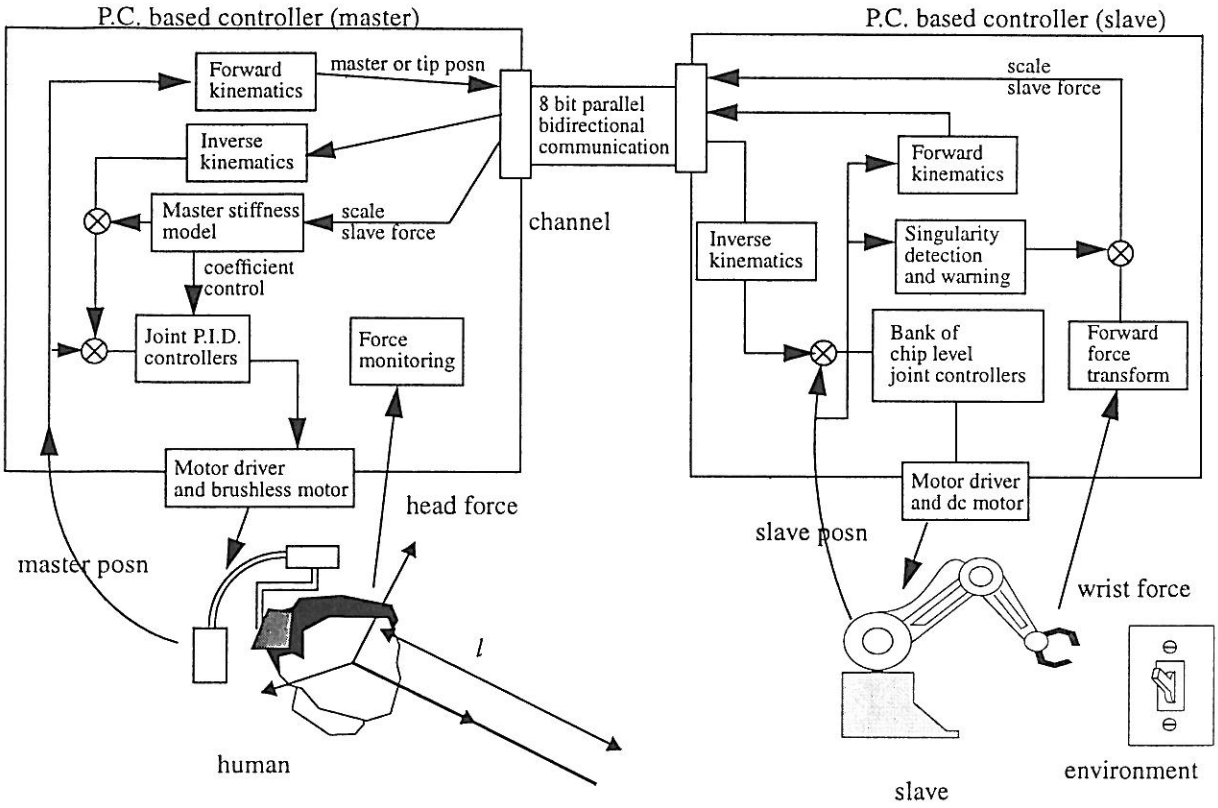

Fig 10. Telerobot master-slave control diagram.

[Figure 10 - Telerobot master-slave control diagram.]

9.2 Master servo controller

The master servo controller dictates the overall system timing, provides PID control for the master, and processes auxiliary inputs from the operator allowing the slave to change the mapping between master and slave. Two processes are run on the master, a main foreground process and a background servo loop process. The background servo loop process runs six PID control algorithms, one for each axis. The Cartesian nature of the master allows ready determination of both forward and inverse kinematics directly from and to these PID controllers. The servo rate is set by the master and usually chosen to be close to the computing capacity of the overall system. Servo rates of 200 Hz are typical.

9.3 Slave servo controller

The slave computer runs a real time operating executive (RTOE) that schedules and dispatches tasks. There are three phases in the RTOE, start up where the executive takes over control of the computer hardware, real time operation and system shutdown where the computer disk operating system (DOS) is restored. When running the RTOE manages tasks via wake/sleep exchange and tasks can be woken either via a hardware interrupt or by other tasks.

Six tasks are run by the RTOE, a debug task, a communication task, a motion task, a keyboard task, a direct movement task and a shutdown task. All tasks can send information to the universal queue for downloading once the RTOE closes down and restores the disk operating system. The structure of the tasks is shown in figure 11 and the core of the slave robot control is achieved in the motion task.

The motion task begins executing after an entire package of position and force information is received from the master robot. Requests for a new mapping between master and slave are received by this task and one of a number of preset maps are chosen. The task then calculates the current pose of the slave robot and uses this information along with the data received by the master to calculate the desired or goal position for the slave robot enforcing this new pose on the slave via the six HCTL1100 chips. The current forces being exerted on the end-effector are measured and transformed into base frame coordinates. Next the task checks for singular configurations and, if the slave is approaching one, the software generates an audio signal and modifies the force data to try to keep the master from reaching the singularity position. Finally, it creates a package with the slave's position and force data, and passes this to the queue for transmittal to the master.

The RTOE can be shutdown in two ways, the first is an ordered shutdown where any of the tasks wakes the shutdown task. If however the software "locks up" the computer, a preemptive shutdown is possible via a keyboard interrupt and as

much of the system state as possible is saved for subsequent debugging.

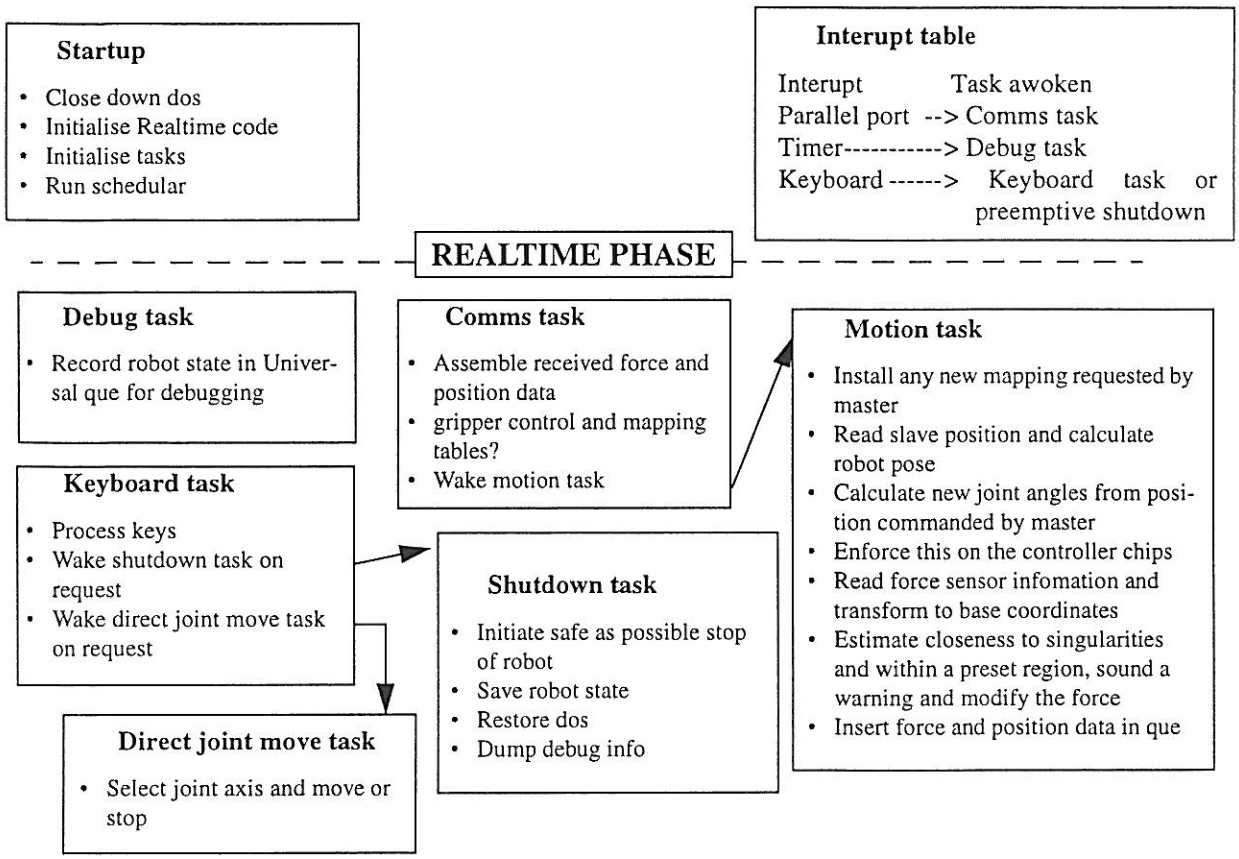

Fig 11. Telerobot slave software tasks.

[Figure 11 - Telerobot slave software tasks.]

9.4 Communication and timing

The servo loop rate is determined by the master controller and is able to detect when the software on the two computers is not able to complete the algorithm within the loop time. The servo loop time is propagated from the master to the slave via the data packets they exchange and two protocols are available to do this. The first is a data "ping-pong" protocol where the master sends data to the slave which the slave acts upon and returns the results once available. The second is a data exchange protocol where the master and slave alternately exchange data and then act upon that data. The data exchange protocol results in less wasted processor time and a potentially faster servo loop rate however software debugging proved simpler when using the "ping-pong" protocol.

9.5 Virtual head-stick control of the head operated telerobot

One natural method of operating the telerobot is to simulate the characteristics of a head-stick or mouth-stick using the telerobot test-bed. A head-stick, also descriptively known as a unicorn stick, is a combination of a stiff head-band and rigid stick. The stick ranges from 30-70cm in length and allows an individual to operate a keyboard, or do simple manipulation tasks with the tip of the stick by using appropriate movements of his or her head. A mouth-stick differs only in the fact the stick is held in the person's teeth and is more readily deposited in a convenient holder. Both head-sticks and mouth-sticks can be considered as a very simple example of a telemanipulator where there is no net power gain.

The virtual head-stick method of operation involves adding a rigid imaginary link, which can have a dynamically varying length, at the origin of the final coordinate frame on the master robot (the centre of the individual's head). Using the master robot's forward kinematic transform and a transform relating to the positioning of the virtual head-stick it is possible to determine the target position of the slave robot. This can then be implemented using the slave forward kinematic transform.

When the slave is not in contact with the environment it attempts to follow the trajectory of the tip of the virtual head-

stick. On contact with the environment forces are measured by a 6 axis force/torque wrist sensor and these must be relayed back to the user in an appropriate fashion.

Although other control schemes are possible the virtual head-stick approach provides a highly intuitive method for bilateral telemanipulation and operations that cannot be achieved using head-sticks or mouth-sticks can be done when using the test-bed in this virtual head-stick mode. An example of such a task is using head movements to control the test-bed to insert a key into a Yale lock and turn it. The same system will also provide the user with a level of strength enhancement well beyond that which could be achieved with a passive device35,36.

10. DISCUSSION

Each of the languages described in sections 6 to 9 have been developed with the goal of providing a useful and efficient method for human-robot programming and interaction for rehabilitation applications. These approaches are at various stages of development and some comments follow regarding the degree to which the various languages have been able to achieve their development goals.

10.1 CURL

CURL has the longest history of continuous use and has been evaluated in both educational and vocational environments within several research projects. Development of the language was undertaken in close cooperation with potential users of the system. Within a hybrid microcircuit fabrication facility, CURL was used to command a robotic visual inspection system 11. The six month trial involved a single user possessing insufficient dexterity to manipulate circuits manually. The inspection system enabled the user to scan the surface of hybrid microcircuits. When a potential circuit defect was noticed, the user was able to pause scanning and zoom in on a region of interest for closer examination. The investigators concluded that the language component of CURL was highly effective in this context. The interpreted nature of CURL enabled tasks to be rapidly modified to accommodate new circuit palettes. Certain inspection procedures were criticised by the operator as involving too many user-invoked sequential operations. These criticisms are being addressed within a successor project where a more specialised user interface is being developed. In another project, a finger painting activity was undertaken by five physically disabled children using an RTX robot and CURL. The trials were used to evaluate potential user input devices and interfaces for implementation within an educational robotic system5. Switch input devices, pointing devices and a voice recognition system were evaluated, demonstrating the input flexibility afforded to CURL through its implementation as a Microsoft Windows application.

Trials of an prototype iconic interface, developed for use with CURL in the context of an office-based robot installation, illustrated the benefits of HMI modularity (figure 12). The iconic interface facilitates the direct manipulation of objects within the robot's environment using a drag and drop metaphor. Pre-programmed robot tasks are associated with specific combinations of drag and drop operations. Substantial research in the area of direct manipulation systems has demonstrated the advantages of such interfaces for generic control purposes17. The prototype iconic interface was evaluated alongside several alternative interface strategies, each conveying user intentions to CURL37. Trial participants who were able to use a pointing device expressed a strong preference for the iconic interface.

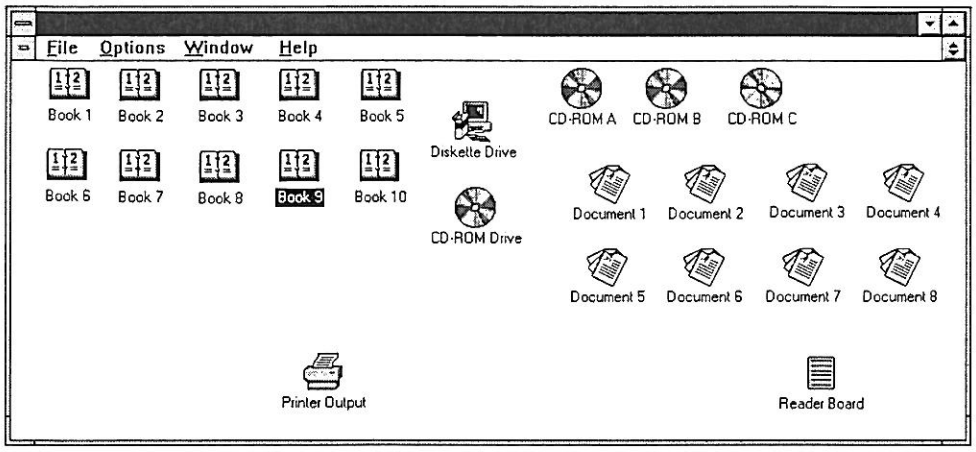

Fig 12. A prototype CURL iconic interface configured for an office-based workstation

[Figure 12 - A prototype CURL iconic interface configured for an office based workstation.]

The design, development and commercialisation of CURL has contributed to the understanding of the issues involved in rehabilitation robot command in several areas. In particular, there is an increased awareness of the requirement for generality in the design of software systems. Modularity and extensibility are vital attributes if the limited market for such software is not to be further restricted. Future design effort must concentrate on these issues.

Further development of CURL will look to object oriented software technologies such as OLE Automation to facilitate the development of increasingly modular software systems. In the future it should be possible to embed specific functionality into any application. Conversely, it should also be possible to encapsulate application software with a preferred interface. Investigations into the potential of these technologies within the field of rehabilitation robotics are in progress.

Through the development and application of CURL it has been possible to observe the substantial advantages of GUIs in the context of robot command. Certain features of such environments, however, can be problematic. The ability to run multiple applications concurrently and to switch between them in an arbitrary manner presents safety issues. A design philosophy was that the user should always be able to halt the robot using a single click of a mouse button or equivalent input device. Safety in CURL might be enhanced by providing an 'on screen' robot stop button within a window which floats on top of other active windows. There can also be latency issues associated with the multitasking nature of GUIs. A common method to surmount this difficulty is to run real-time control loops on a secondary processor located within the robot hardware.

10.2 MUSIIC

MUSIIC was developed within a framework of the supervised control of an assistive telerobot for use in a real-world domain. The ability to operate in an unstructured environment is the key feature of MUSIIC. The user is not constrained to work in a fixed work-cell or a predetermined structured environment. The hierarchical object oriented knowledge base allows the planner to synthesise plans for object manipulation tasks solely from shape, pose and location information obtained from the vision system. However, additional information contained in the knowledge base, as well as user supplied information can be used to supplement vision information for more accurate planning and control.

By engaging in dialogue with the user in such a way that natural deictic gestures and voice input can be used to carry out a task, MUSIIC provides many of the advantages present in direct manipulation interfaces17. The user can directly designate objects and locations in a chosen environment, and use natural language to describe the desired actions on those objects and locations. By combining different modalities, rather than attempting to constrain dialogue to one modality, great simplification of processing can be accomplished. The gesture control and the spoken input are used to make a general purpose object recognition module unnecessary. Furthermore, the user is not compelled to learn a specialised vocabulary to control the robot's behaviour nor is does the interface require sophisticated programming skills.

While MUSIIC is capable of generating autonomous plans, the architecture incorporates the user in the planning process. The human-machine interface and hierarchical object oriented representation of objects allow the user to interact with the planning system at any level of the planning hierarchy, from low-level motion and grasp planning to high-level task planning of complex tasks such as feeding. Machine synthesised plans can be supplemented, corrected and modified by user intervention whenever incomplete information prevents the planner from synthesising plans autonomously. This is done by either taking over control of the planning process or by providing information to the knowledge bases. Furthermore, interleaving planning with execution allows the user to maintain a supervisory role over plan execution.

Since MUSIIC is "instructable"38, the user is capable of giving verbal and gestural instructions to the robot and teaching it new tasks in a familiar own environment. The user is able to invoke a series of commands and name the sequence for later invocation by the robot.

MUSIIC is capable of adapting previously learned plans and tasks to new situations and objects. Previously synthesised or learned plans can be modified to operate on new tasks and objects. Furthermore, MUSIIC also has reactive capability. When things go wrong, the system tries to determine the cause of error and autonomously re-plans to rectify the problem in simple cases such as when object placement is inaccurate. If, however, it is not able to ascertain the cause of failure, the system then engages in a dialogue with the user who takes over the plan correction and re-planning procedures.

The long term goal of MUSIIC is to port the system to a wheelchair mounted system. Pending that goal, the system is going to undergo user testing on the experimental test-bed. A simulation mechanism is being developed that will be used to study the multimodal interface and the results will be used to further fine-tune the control language39. Following these user testing of the system will begin.

10.3 RoboGlyph

RoboGlyph has been evaluated on the DeVAR service robot, developed to assist high-level quadriplegic operators function in an office environment8. A total of nine people developed programs using RoboGlyph. The test subjects represented three groups including: robot programmers with at least a Master's degree and programming experience in VAL;

engineers with formal computer programming training and two years of practical software development experience but no robot programming experience; and non-engineers with no robot programming experience and little or no computer programming experience.

The assessment of RoboGlyph included a training activity in which the subjects were given a general orientation to RoboGlyph. This included a demonstration of some of the features and an opportunity to experiment with the tools. They were introduced to the storyboard concept and were expect to develop their own storybook for the simple application of locating a refrigerator door and positioning the gripper at the door handle. Once the subjects were comfortable with the position editing tools they were asked to create a more complicated storybook for opening the door of a microwave oven. The time required to develop the storybook was recorded for each subject as were the number of test runs required to get the program to work correctly. The final assessment task required the subjects to examine and repair a program written by someone else that had an error placed in its storyboard. The times required to identify the location of the error and to fix it were measured.

The results of the RoboGlyph assessment sessions were analysed using methods similar to those used in single subject experiments in special education and psychology40. The findings indicated that non-experts with less than one hour of training were able to program the robot just as fast as expert programmers. All of the subjects could understand and explain the individual steps in a RoboGlyph storyboard for a task and quickly identify the location of an error in the storyboard. Such a task is frequently difficult or impossible with a VAL program. The training times for all subjects were in the range of 20-50 minutes.

The subject performance for creating a storyboard was comparable across the subjects. This suggests that the interface design philosophy demonstrated by RoboGlyph does provide an intuitive approach to developing robot programs. It was interesting to note that the subjects used several different approaches to creating the storyboard although the time required was about the same. Also, seven of the nine subjects had the storyboard running correctly on the first attempt.

All nine subjects were able to correctly identify the location of an error to within one cell of the correct location. The time required to locate the error was comparable across the subjects (2-3 minutes) although the time to correct the error varied between 1 and 9 minutes. The non- engineers and robot programmers required significantly less time (factor of 3) than the engineers to correct the error. It is proposed that non-engineers tended to look for simple solutions to the problems whereas engineers tended to look for more complicated solutions.

Based on the assessment, it was concluded the RoboGlyph achieved its design goal of being accessible to users without extensive technical training.

10.4 A rehabilitation telerobotic test-bed

The rehabilitation telerobotic test-bed illustrates a novel and desirable means of providing an individual with a motor disability with information about the slave robot's interaction with the environment in a form that is intuitive and immediate. To do this it has been necessary to move away from the position control servo-mechanisms typical of most industrial robots and investigate control architectures where force information is integrated into the control loop. One promising approach is to treat the master, slave, operator and environment as a combination of a mechanical impedance coupled with either a force or position generator. The impedance of the master and slave robot can be determined from manufacturers data or experimentally41 and some estimate of stability and operation can be deduced. This approach is still in its infancy and there have been no obvious methods available for dealing with the nonlinearities inherent in the four components of a master-slave telerobot. To program successful controllers on the rehabilitation telerobot test-bed it has been necessary to develop multitasking software that conveys controller information to appropriate tasks in the structure. This need is especially relevant in rehabilitation robotics systems where higher system bandwidths are desirable so that the robot can track a surface, stop and react if the internal model conflicts with sensor information, or increase grip force to prevent slip. Although the telerobotic test-bed has demonstrated an operability similar to that of a mouth-stick and is able to extend the person's range of movement and force, there is one attribute that is lacking, that of tactile information in the 100-300 Hz range. It is information in this range that is passed up the body of the mouth-stick and to the individual, possibly via sensors on the person's lips, that permits precision manipulations such as lifting a single sheet of paper from a stack. It remains to be seen whether this information can be included in future rehabilitation devices and what value it has for individuals who would benefit from this technology.

11. Conclusions

In rehabilitation robotics the interface between the individual and the robotic system is of paramount importance and he need for the interface to be intuitive and accessible for non-technical individuals adds a further dimension to the problem. Other demands made on rehabilitation robotics are that they should, not only should it function reliably, intuitively and flexibly - demanding technical sophistication, but they should also be portable, and cost/effective.

None of the software and languages discussed meet all these needs, however each of the four projects described provides a window on these needs. Taken in combination the human machine interface concepts developed in rehabilitation robotics may have a part to play in the consideration of the wider needs of non-technical individuals working with intelligent machines.

12. Acknowledgements

Funding for the projects outlined has been provided in part by the Nemours Research Programs, the Leverhulme Trust (ref F/452/B), the Canadian Natural Sciences and Engineering Research Council, the U.S. Department of Veterans Affairs, and the National Institute on Disability and Rehabilitation Research of the U.S. Department of Education, (Grant # H133A20001 and Grant # H113E30013). Many individuals have contributed to this work and the authors acknowledge the considerable help of Larry Leifer, Fred Lakin, Jim Anderson, Richard Foulds, Daniel Chester, Matthew Beitler, Sean Stroud, Marcos Salganicoff, Daniela Pino, Shoupu Chen, Vijay Jayachandra, Richard Mahoney, Vijay Kumar, and Edwin Hereda.

13. REFERENCES

1. M. Hillman and J. Jepson, "Evaluation of a robotic workstation for the disabled" *Journal of Biomedical Engineering*, **14**(3), 187-192 (May 1992)

2. G. Bush, I. Al-Temen, J. Hancock, J. Bishop, E. M. Slack, and I. Kurtz, "Development of a myoelectrically controlled wheelchair mounted object manipulator for quadriplegics" In *proceedings of the 4th International Conference on Rehabilitation Robotics*(Applied Science and Engineering, PO Box 269, Wilmington, DE) 103-106 (June 1994)

3. M. Topping, "Early experience in the use of the Handy 1 robotic aid to eating" *Robotica,* **11**(6), 525-528 (November 1993)

4. A. M. Cooke, P. Hoseit, K.A. Man Liu, R. Y. Lee and C. M. Zenteno-Sanchez, "Using a robotic arm system to facilitate learning in very young disabled children", *IEEE Transactions on Biomedical Engineering*, **35**(2) 132-137 (February 1988)

5. M.T. Beitler, C.A. Stanger and R.D. Howell, "The design of an integrated interface to an educational robotic system" *Proceedings of the RESNA 94 Annual Conference*, Nashville, USA 448-450 (1994).

6. W.S. Harwin and A. Ginige and R.D. Jackson, "A Robot Workstation for use in Education of the Physically Handicapped" *IEEE Transactions on Biomedical Engineering*, **35**(2) 127-131 (February 1988)

7. J. L. Schuyler and R. M. Mahoney, "Vocational robotics: job identification and analysis" *Proceedings of the RESNA 95 Annual Conference*, Vancouver, 542-544 (1995)

8. J. Hammel, K. Hall, D. Lees, L. Leifer, H.F.M. Van der Loos, I. Perkash, R. Crigler, "Clinical Evaluation of a Desktop Robotic Assistant," *Journal of Rehabilitation Research and Development*, **26**(3), 1-16 (Summer 1989).

9. G. E. Birch, "Development and Methodology for the Formal Evaluation of the Neil Squire Foundation Robotic-Assistive Appliance" *Robotica* **11**(6), 529-534 (November 1993)

10. R. Cammoun, J.-M. Detriche, F. Lauture and B. Lesigne, "Clinical Evaluation of the MASTER Robotic System and Development of a New Version" *Robotica* **11**(6), 535-540 (November 1993)

11. J.L. Dallaway, R.M. Mahoney and R.D. Jackson, "The application of rehabilitation robotics within manufacturing industry" *Proceedings of the fourth International Conference on Rehabilitation Robotics*, Wilmington, USA, 145-149 (1994)

12. H. H. Kwee, M. M. M. Thonnissen, G. B. Cremers, J. J. Duimel and R. Westgeet, "Configuring the MANUS system" *Proceedings of the RESNA 92 Annual Conference*, Toronto, 584-587 (1992)

13. S. J. Sheredos, B. Taylor, C. B. Cobb, and E. E. Dann, "The Helping Hand electro-mechanical arm" *Proceedings of the RESNA 95 Annual Conference*, Vancouver, 493-495 (1995)

14. A. I. Batavia and G. S. Hammer, "Toward the development of consumer-based criteria for the evaluation of assistive devices" *Journal of Rehabilitation Research and Development* **27**(4), 425-36 (1990)

15. P. S. Schenker, "Intelligent Robotics for Space Applications", in *Intelligent robotic systems*, (Marcel Dekker Inc, 270 Madison Avenue, New York, 1992), 545-591.

16. "NSF Report of workshop on coordinated multiple robot manipulators: Planning control and applications" *IEEE Journal of Automation and Robotics*, **4**(1), 91-93 (1988)

17. B. Shneiderman, "Designing the user interface : strategies for effective human-computer interaction", (Addison-Wesley, 1992).

18. G. G. Gosine, W. S. Harwin and R. D. Jackson, "Development and application of a task-level robot control language" In *proceedings of the 1st International Conference on Rehabilitation Robotics*(Applied Science and Engineering, PO Box 269, Wilmington, DE) 97-118 (June 1990)

19. S. L. Minneman and Thanh Pham, "CALVIN: a robot control language for rehabilitation robotics" *Interactive Robotic Aids - One Option for Independent Living: an international perspective* Monograph 37, (World Rehabilitation Fund, New York, NY, 58-60 1986)

20. J. L. Dallaway, R. M. Mahoney, R. D. Jackson and R. G. Gosine, "An interactive robot control environment for rehabilitation applications" *Robotica* **11**(6), 541-552 (November 1993)

21. W.A. McEachern, J.L. Dallaway and R.D. Jackson, "Sensor-based modification of manipulator trajectories for applications in rehabilitation robotics" *Transactions of the Institute of Measurement and Control* **17,** 272-280 (1995).

22. D. Lees, L. Leifer, "A Graphical Programming Language for Robots Operating in Lightly Structured Environments," *Proceedings of the 1993 IEEE International Conference on Robotics and Automation,* 648-653, (Atlanta, GA, USA, May 1993).

23. D. P. Koons, "Capturing and interpreting multi-modal descriptions with multiple representations" In *Intelligent Multi-Media Multi-Modal Systems* (AAAAI press), 13-21 (1994).

24. R. A. Bolt, "Put that there: Voice and gesture at the graphics interface" *Computer Graphics* **14**(3), 262-270 (1980)

25. Z. Kazi, M. Salganicoff, M. Beitler, S. Chen, D. Chester, and R. Foulds, "Multimodal User Supervised Interface and Intelligent Control for a Rehabilitation Robot" In *proceedings of IJCAI-95 Workshop on Developing AI Applications for the Disabled,* Montreal, Canada, 46-58 (1995)

26. E. D. Sacerdoti, *A structure for plans and behaviour* (American Elsevier, New York, 1977)

27. Z. Kazi, M. Salganicoff, M. Beitler, S. Chen, D. Chester, and R. Foulds, "An Intelligent Telerobotic Assistant for People with Disabilities" In *proceedings of SPIE's International Symposium on Intelligent Systems and Advanced Manufacturing: Telemanipulator and Telepresence Technologies II,* Philadelphia, PA, USA, 120-130 (October 1995)

28. D C Simpson and J G Smith, "An externally powered controlled complete arm prosthesis" *Journal of Medical Engineering and Technology,* 275-277 (September 1977).

29. T. H. Massie and J. K. Salisbury, "The PHANToM haptic interface: a device for probing virtual objects" *International mechanical engineering congress*, Chicago, 295-302 (November 1994)

30. B. Hannaford, "A design framework for teleoperators with kinesthetic feedback" in *IEEE Journal of Robotics and Automation,* **5**(4) 426-434 (1989)

31. D. A. Lawrence, "Designing teleoperator architectures for transparency", *Proc. IEEE International Conference on Robotics and Automation,* 1406-1411 (1992)

32. H. Kazerooni and S. L. Mahoney, "Dynamics and Control of Robotic Systems Worn by Humans" *Journal of Dynamic Systems, Measurement, and Control*, **113**, 379-387 (September 1991)

33. J. E. Colgate, "On the design and control of bilateral manipulators for rehabilitation" In *proceedings of the 4th International Conference on Rehabilitation Robotics*(Applied Science and Engineering, PO Box 269, Wilmington, DE) 175-181 (June 1994)

34. C. A. Stanger, A. C. Phalangas, and M. F. Cawley "Range of head motion of high cervical spinal cord injured individuals for the design of a testbed robotic system" In *Fourth International Conference on Rehabilitation Robotics*, (Applied Science and Engineering, PO Box 269, Wilmington, DE) 37-42 (June 1994)

35. M Salganicoff, V Jayachandran, D Pino, T Rahman, R Mahoney, S Chen, V Kumar, W Harwin, J Gonzalez, "A virtual head-stick rehabilitation robot system" *Proc. IEEE International Conference on Systems, Man and Cybernetics* Vancouver (1995)

36. W S Harwin and T Rahman, "Analysis of force-reflecting telerobotic systems for rehabilitation applications" In *First European Conference on Disabilities, Virtual Reality and Associated Technologies* (University of Reading, ISBN 0 7049 1140 X), 171-187 (1996)

37. J. L. Dallaway, "Human-computer interaction within robotic workstations" *Proceedings of the RESNA 96 Annual Conference*, Salt Lake City, USA, 339-341 (1996)

38. C. Crangle and P. Suppes, *Language and Learning for Robots* (CSLI Publications, Stanford, CA, 1994).

39. M. Beitler, R. Foulds, Z. Kazi, D. Chester, S. Chen, and M. Salganicoff, "A Simulated Environment of a Multimodal User Interface for a Robot" In *proceedings of RESNA* Vancouver, Canada, 490-492 (1995)

40. D. S. Lees and L. J. Leifer "Experimental evaulation of a graphical programming environment for service robots" In *proceedings of the 4th International Conference on Rehabilitation Robotics*(Applied Science and Engineering, PO Box 269, Wilmington, DE) 19-23 (June 1994)

41. V. Jayachandran *A force reflecting assistive telerobot*, Masters Thesis, University of Delaware (1995)