A Generic Control Architecture for Telerobotics

Alan R. Graves and Chris Czarnecki
Centre for Computational Intelligence,
De Montfort University,
Leicester, LE1 9BH.
arg|cc@dmu.ac.uk

26.3.99

Abstract

This paper describes a telerobotic system designed for the real world task of bomb disposal. The system is based around an existing remotely operated vehicle built for this purpose some years ago. The original vehicle has been enhanced in order to contend with a number of operational problems identified with its use. A sensor configuration and distributed control system are described which address these problems. The core of the control system is founded upon the principles of the Subsumption Architecture, although the design is grounded in a wider concept of the development of generic user interfaces which may be used to control a number of different types of telerobot. This is a desirable feature for application domains in which a number of different types of robot are used for performing distinct functions, such as the nuclear industry and military operations. Localising the differences between telerobots to within the remote control system can increase flexibility and can simplify the development of different versions of a system. This requirement presents a number of problems when considering the physical structure (sensors, chassis and actuators) and the control systems (functions and behaviors) of different telerobots. To this end, a system architecture, communication protocol and robot description language are described which address these issues.

Proc. TIMR 99 "Towards Intelligent Mobile Robots", Bristol 1999
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 - 6161.
Report number UMCS-99-3-1. http://www.cs.man.ac.uk/cstechrep/titles99.html

A Generic Control Architecture for Telerobotics

Alan R. Graves and Chris Czarnecki
arg|cc@dmu.ac.uk

Centre for Computational Intelligence,
De Montfort University,
Leicester, UK, LE1 9BH.

Abstract

This paper describes a telerobotic system designed for the real world task of bomb disposal. The system is based around an existing remotely operated vehicle built for this purpose some years ago. The original vehicle has been enhanced in order to contend with a number of operational problems identified with its use. A sensor configuration and distributed control system are described which address these problems. The core of the control system is founded upon the principles of the Subsumption Architecture, although the design is grounded in a wider concept of the development of generic user interfaces which may be used to control a number of different types of telerobot. This is a desirable feature for application domains in which a number of different types of robot are used for performing distinct functions, such as the nuclear industry and military operations. Localising the differences between telerobots to within the remote control system can increase flexibility and can simplify the development of different versions of a system. This requirement presents a number of problems when considering the physical structure (sensors, chassis and actuators) and the control systems (functions and behaviors) of different telerobots. To this end, a system architecture, communication protocol and robot description language are described which address these issues.

1 Introduction

Telerobots are commonly used for extending the ability of a human expert to perceive, reason with and perform actions in an environment which is distant, hazardous or otherwise accepted as being inaccessible to humans. Traditional, direct control teleoperator interfaces are giving way to modern computer workstations which allow rich representations of the remote environment to be presented to the user through a number of techniques ranging from 2D computer graphics through to virtual reality and augmented reality displays [9]. Additionally, Artificial Intelligence (AI) techniques enable robots to perform sensor-guided functions, either alleviating the need for continuous control by the operator, or providing higher levels of accuracy in positioning and manipulation than may be achieved by an unaided operator. Interfaces to such modern teleoperator control systems must therefore allow the user to interact with both the remote environment *and* with complex functions which control the mechanical hardware systems. These interfaces are, by necessity, much more sophisticated those which front traditional teleoperator systems.

Given a well-designed interface of this type, there are a number of application areas in which motivations exist for using it to control a number of different types of telerobot. Typically, telerobots and their user interfaces are designed as single cohesive packages. However, in application areas such as the nuclear industry and bomb disposal, a number of different types of robot are typically employed, each of which has a physical form and set of capabilities suited to a particular role, and a resultant unique user interface. Additionally, in situations where specific system variants are developed in order to meet mission requirements as they arise, it is advantageous to have a single interface which can adapt itself to the particular remote system.

The research presented in this paper is applied to a physical robot designed for the well-defined real world application of bomb disposal, as has been previously described by Graves and Czarnecki [8]. An existing teleoperated vehicle used for military bomb disposal operations has been upgraded into an advanced telerobotic system, however, this particular robot is only one of many designs which are used in this field of work; other robots range from small reconnaissance robots through to large teleoperated forklift trucks.

The addition of distributed computational elements such as microcontrollers and embedded PCs within the telerobotic control loop allows communication between the user interface and the telerobot to be performed at a higher level of abstraction than with direct teleoperation. The concept presented in this paper is that, given a communication protocol sufficiently powerful and abstract, a single user interface can be used to operate a number of different types of telerobot which adhere to the protocol. The protocol described in this paper is founded upon the principle of a telerobot describing itself to the user interface, taking into account not only aspects such as the physical configuration of the robot (actuators and sensors) but also more abstract components; i.e. the semi-autonomous functions which the robot control system provides.

The SMART architecture [1] shares many of the same motivations, although that approach is designed expressly for manufacturing workcells rather than mobile telerobots. The approach described here also differs quite markedly from other robot modelling techniques in both intent and implementation. In contrast to modelling techniques designed for simulation which provide physically accurate models of the robots and their environments [5], this approach describes the characteristics of the robot in a user interface-oriented way. For instance, presenting an ergonomically valid user interface to manipulation tasks requires the remote control system to associate actuators with appropriate gestures (e.g. 'forward', 'up', 'rotate left') which may be sensed by a joystick or other input device. However, the approach taken here in the representation of generic sensors has some similarities with the simulation work of Moore and Ericksson [10].

This paper describes a telerobotic system which has been designed as separate components built around the communication protocol. The format of the paper is as follows. Section 2 describes the original teleoperated vehicle and the problems identified with its use. Sections 3 and 4 describe how this has been enhanced through hardware improvements and a software control system respectively. Section 5 presents the distributed control system architecture and communication protocol which bind the user interface and the telerobot together, providing the potential for controlling a number of different types of robots from a single user interface. Section 6 provides a summary and presents some directions for future work.

2 The Original Teleoperated Vehicle

The bomb disposal robot is a fairly large, mechanically robust device. It consists of a mobile platform with a 3 degree of freedom hydraulically operated boom with extra degrees of freedom for the end of boom and cameras. To disable a bomb, a disruption device attached to the end of the boom is triggered when appropriately positioned by the operator. The vehicle also has lamps on-board and travels on two sets of tracks which are electrically driven. The tracked platform allows the robot to traverse difficult obstacles such as stairs. The vehicle is shown in Figure 1.

The human operator views the remote environment on a small monitor which displays the view sent back from the monochrome cameras mounted on the vehicle. This version of the system is controlled directly by means of a hand-held electro-mechanical control box which requires the operator to be in continuous control of the vehicle at the level of each individual motor and actuator. Also, many of the controls on the unit serve multiple functions.

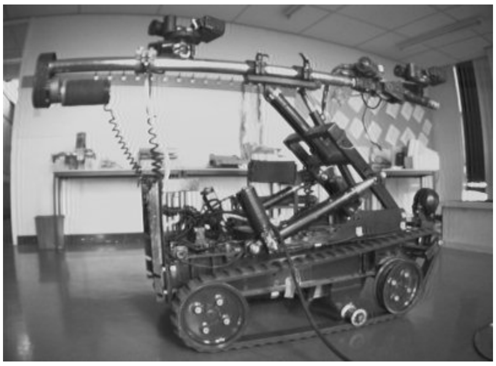

Fig. 1 – The bomb disposal robot.

This system configuration leads to a number of problems being experienced in the navigation of the vehicle en route to the target site. The video feedback can prove inadequate for successful navigation, especially in situations of impeded visibility. This leads to collisions with obstacles, or even more seriously, the vehicle becoming trapped in passageways which appear deceptively wide enough to pass through. Also, when the large, heavy boom is extended, the traversal of steep inclines can cause the robot to become unstable and topple over. There is also a perceived need to reduce the mental workload placed upon the operator by the current system, allowing them to concentrate more on mission goals rather than peripheral concerns of robot navigation.

An additional problem which has been identified concerns the fact that, although just one system has been described here, bomb disposal units make use of a number of different custom-designed robots, each having a mechanical design suited to a specific role. There is therefore a requirement to control a number of different types of robot from a single, well-designed user interface. It is envisaged that this approach will lead to benefits in terms of operator training, and a reduction in the amount of equipment which must be transported on operations.

3 Hardware Enhancements

The fundamental mechanical structure described above has been enhanced with a control system based around an embedded PC which controls the relays which were previously manually switched through the hand-held control unit. This control system bases its behavior both on control instructions sent from the user, and information provided by sensors mounted on the vehicle. These include an array of ultrasonic sensors, a laser range-finder and a navigational system consisting of a Global Positioning System (GPS) receiver and an electronic compass.

These sensors allow the control system to perform a number of navigational tasks semi-autonomously. The GPS receiver and electronic compass provide global navigation to a specified target point, and the laser range-finder and ultrasonic sensors permit it to steer around obstacles in this path. The control system also halts forward movement or retracts the boom when steep inclines are encountered which would otherwise cause the robot to become unstable. Due to the short nature of the missions and constant human supervision, navigation does not need to extend to the complexity of map building and planning. (e.g. as in [3].)

4 Man-Machine Control System

The core of the control system on-board the telerobot has responsibility for combining commands from the human operator with data from the numerous sensors, and providing a coherent control strategy to drive the robot. The control system which performs this is based upon Brooks' Subsumption Architecture (SA) [4]. This approach overcomes the limitations imposed by the demands of real-time autonomous robotics which have beleaguered traditional techniques for achieving robot control. This architecture is ideal for implementation on computers on-board mobile robots which are constrained in terms of computational power. For these reasons, this was the architecture chosen for the basis of the control system.

However, classical SA controllers tend to provide hard-coded behavior, lacking the programmability and versatility required in telerobotic applications. Connell's SSS architecture extends the basic model by sandwiching the subsumption network between a low-level servo control layer, and a high-level symbolic layer [6]. The symbolic layer exerts overall control by providing parameters for the subsumption modules, thereby making the whole system more versatile than a traditional SA controller. In the controller described in this paper, one of the main purposes of the symbolic layer is to receive commands from the operator and apply them to the subsumption network, including selectively enabling and disabling modules. The overall system architecture into which this controller fits is shown in Figure 2.

The subsumption network is composed of four distinct levels (also referred to here as Behaviors). The lowest level implements global navigation, taking input from the navigational subsystem and sending signals to the robot's motors. Above this layer is the local navigation Behavior which guides the robot around obstacles. Above this is a layer which controls the motion of the hydraulic boom, and the highest level is that which ensures that the vehicle does not topple over.

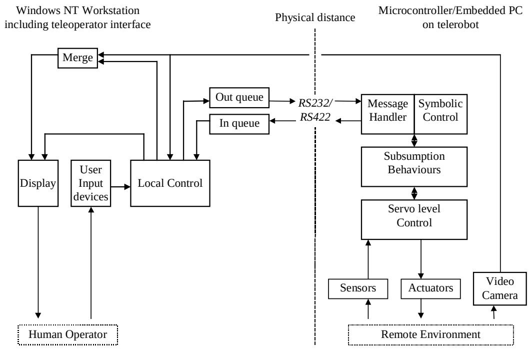

Fig. 2 – Overall system architecture.

In Figure 3, the supervision exerted by the symbolic layer is shown as a signal routed to each module in the subsumption network. At the coarsest level of control this allows the human operator to instruct the symbolic layer to switch Behaviors off to achieve full manual control. Other, more fine-grained, integration of human and machine behavior is achieved by the symbolic layer injecting operator control signals directly into modules of the network (shown as shaded boxes) leading to hybrid man-machine competences.

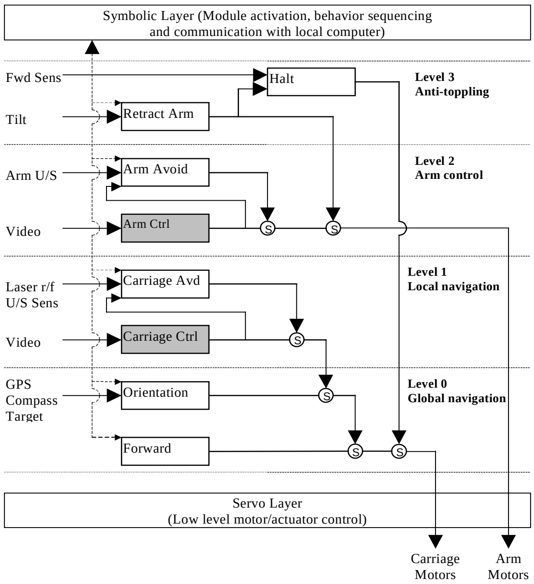

Fig. 3 – The subsumption network.

The two main responsibilities of the local control system are to interface with the symbolic portion of the remote control system, and to manage the user interface. Input from the remote video camera is digitised and displayed in an application window together with standard GUI controls for displaying and manipulating the status of the remote telerobot. The video image is merged with computer-generated graphics in order to overlay data onto the image, allowing the operator to view important status information whilst remaining focused on the environment. In this way, features of the remote environment perceived by the sensors pass up through the subsumption network and are displayed to the user via the local control system and user interface.

The SA represents only one approach for implementing robot control systems, and it is therefore not a prerequisite of the architecture presented in this paper that telerobot control systems must follow this design approach. However, the wider field of Behavior-based robotics includes many other controller designs which incorporate the notion of behavioral modules, although operating on radically different computational substrates [2, 7]. Additionally, it is conceivable that virtually any underlying control system could present itself via a wrapper in which its functions are encapsulated by conceptual behaviors. However, true Behavior-based controllers have an underlying structure which maps well to the concept of "services which the telerobot can perform for the user", and therefore allow an accurate model of the underlying behavioral system to be presented cleanly and intuitively to the operator via the user interface.

5 Distributed, Generic Telerobot Control

The previous two sections have dealt with how the operational problems highlighted in section 2 have been targeted. However, the most novel element of this research is the attempt to produce a generic local control system which allows any one of a number of different types of robot to be controlled.

Whilst the robot described in this paper has a known physical configuration of sensors, actuators, and control system Behaviors, these aspects will differ quite markedly from robot to robot. The solution to this problem requires the representation of these specific characteristics at a sufficiently abstract level, thereby allowing a particular robot to describe itself to the local control system in such a way that the operator can access all of its specific features. Unlike simulation models, this representational form must be geared towards the practical requirements of producing a well-organised and easy-to-use man-machine interface. A communication protocol and descriptive language have been developed which bind the two parts of the distributed control architecture together. These are:

- Protocol for Abstract Control of Telerobots (PACT); a messaging protocol which permits dynamic interaction between local and remote control systems
- TeleRobot Interface Language (TRIL); a definition language allowing a telerobot to provide a static description of its interface requirements to the local control system.

5.1 Protocol for Abstract Control of Telerobots

The protocol is grounded upon the notion of services which the remote telerobot can perform when requested by the local control system. The services are accessed in a number of different ways by the local computer. A number of basic motion and obstacle sensing services are assumed of the telerobot, and these are built into the messaging protocol. These are referred to as *core services*. A general level of access to the unique features of the telerobot is established through an initial handshaking phase involving an exchange of messages relating to the basic sensor, manipulator and Behavior structure of the remote robot. These are known as *generic services* and are mostly synonymous with control system Behaviors. The third class of services are the *specific services*; these allow the user interface to present aspects of the telerobot to the user in more detail, and allow access to internal features of the telerobot's Behaviors.

To clarify the distinction between generic and specific services, during the handshaking phase, the robot describes its Behaviors (e.g. obstacle avoidance) as *generic* 'black boxes', which the user interface may simply toggle on or off. When these Behaviors are described in *specific* terms, the user interface provides mechanisms for interacting with the inner structure of the Behaviors; e.g. supplying parameters, selecting modes or sending specific commands. Definition of specific services is performed through a user interface-oriented description of the robot written in TRIL. In effect, the second two levels of services act to customise the standard user interface to the generic and specific features of an individual type of telerobot.

Initial dialogue between the user interface and the telerobot control system involves establishing a basic description of sensor arrays, manipulator arms and Behaviors. This provides a unique identifier for each sensor within a sensor array, and degree of freedom within a manipulator arm. These ID's are used in all subsequent dialogue. Some of the message types used in the registration phase are shown below.

```
REGISTER_SENSOR_ARRAY(int arrayID,
string arrayDesc, int sensorRange)

REGISTER_SENSOR(int arrayID, int
sensorID, string sensorDesc, float
sensorOrientation, enum sensorType)

REGISTER_MANIPULATOR(int manID,
string manDesc)

REGISTER_ACTUATOR (int manID, int
actuatorID, string actuatorDesc, enum
interfaceGesture)

REGISTER_BEHAVIOR (int behaviorID,
string behaviorDesc)
```

The default user interface provides a basic layout of sensors on a diagram of a generic robot, and sensor information will be overlaid on the video image for basic collision warnings. For instance, a warning arising from a sensor which has registered itself as facing left will result in warnings placed on the left-hand side of the camera view. Alternatively, if the sensor reports itself capable of distance measurements rather than boolean obstacle detection then these distances will be overlaid instead.

The Behaviors which the telerobot registers are represented on screen by a series of labelled toggle buttons within an area of the screen designated for this purpose. The names which the robot controller registers the Behaviors with will provide the labels for these areas; i.e. these labels (such as "Arm control", or "Collision avoidance") will be derived from the *behaviorDesc* field in the *REGISTER_BEHAVIOR* message shown above.

The communication protocol provides high level messages for movement of manipulator arms, and it is the responsibility of the remote control system to translate these commands into low-level joint movements. In the first interface developed which adheres to this protocol a standard 2D joystick is used to generate these commands. However, because during the registration phase the remote system describes the manipulator arm down to the level of individual degrees of freedom, if required, the operator may exert control over an individual joint for fine manipulation. Additionally the control system may send textual messages to the user through the interface, informing the user of the robot's status and any specific warnings or alerts which require greater prominence.

Although core and generic services allow a high degree of control to be exerted over a remote telerobot through a common interface, the architecture definition allows a telerobot controller to include many more functions than may be controlled through the core or generic interface alone. To allow the local computer to represent these additional services to the user, a telerobot may register itself as having a robot type identifier which relates to the name of a file (containing a definition written in TRIL). If a TRIL file does not exist for that particular robot, or the file is not available to the local system, then the generic interface will be used to permit basic usability. However, one of the most powerful features of this concept is that during the registration phase, the telerobot may transmit a stored copy of a TRIL file to the interface computer if no file can be found locally. This leads to a situation whereby the local computer can have a previously unknown type of telerobot attached to it, which is capable of automatically adapting the user interface to suit its individual requirements.

5.2 TeleRobot Interface Language

A basic working language definition and accompanying parser for the syntax and semantics of TRIL currently exists, although there are certainly many more features which would be desirable in future versions. Two of the first features to be implemented relate to the graphical appearance and status of the robot and access to the inner features of specific Behaviors. TRIL is a descriptive language, rather than a procedural one, with a simple syntax and structure which is based on that of VRML (Virtual Reality Modelling Language) [11]. Like VRML, TRIL is composed of nodes which contain named fields. These may store primitive types (such as ints and strings), other nodes, or arrays of these. Nodes are organised hierarchically, with abstract parent nodes defining the type of a number of related sub-nodes. Top-level nodes are those which are not abstract and have no parent. These are the only nodes which may appear outside of a field and typically represent conceptually self-contained interface components. As the extract of the language definition below shows, the two top level nodes (shown underlined) are those which specialise a Behavior (i.e. allow access inside the 'black box') and provide a graphical description of the robot structure.

```
BehaviorSpecialisation { behaviorID int
                         name string
                         specialID int
                         function Widget }

Widget { abstract }
  ├── Message { messageName string }
  ├── Boolean { default bool }
  ├── Combo { items string[] }
  └── ...

RobotDescription { robotName string
                   view RobotView[] }

RobotView { abstract }
  └── NormalView { viewName string
                   image Shape[]
                   sensor SensorArray[]
                   manip Manipulator[] }

Shape { abstract }
  ├── Line { startPoint coord
  │               endPoint coord
  │               thickness int
  │               lineColor color }
  │
  ├── Text { startPoint coord
  │               caption string
  │               rotation float
  │               size int }
  │    ...
  └──
```

The *BehaviorSpecialisation* node allows the operator to access functions inside the registered Behaviors by sending a variety of messages to them. It defines which Behavior it is extending by the *behaviorID* field. This will be a unique number which the control system will have previously allocated to that particular Behavior during the registration phase (discussed in section 5.1). Such ID numbers are used to identify all elements of the remote robot, such as Behaviors, sensors and manipulator segments. The functionality field of the *BehaviorSpecialisation* node contains a *Widget* type node. This will define a screen element such as a button, checkbox, or combo box which can be used to send parameterised messages to that particular Behavior. For example, the TRIL definition may cause a combo box to be displayed in the area of the screen dedicated to that particular Behavior, providing a list of different modes which the user may select between. It is important to note that this definition provided by TRIL gives the user interface no details of the semantics behind the screen elements it displays. When the user selects one of the options the user interface simply dispatches a message to the robot's controller detailing which option has been selected. Effectively, the user interface is behaving as a client, with the robot controller acting as a server.

Because of the constraints on the storage and processor power of the remote computer, and tight bandwidth constraints and possible time delays between the two sites, it is desirable to perform as much processing as possible at the local computer, rather than localising all user interface code on the remote computer and having a local computer which amounts to little more than a dumb terminal. By using the TRIL language and PACT protocol to establish the user interface and communicate with the local computer, these demands are decreased and the robot controller is not concerned with low-level matters of user interface management and user interaction.

The second major node type in the current version of TRIL is the *RobotDescription* node which allows the interface to provide a visual depiction of the robot. In addition to purely descriptive lines and captions which make up the image, other elements such as sensors, manipulators and miscellaneous devices may be drawn. These differ from basic drawing primitives as the representation on the screen is bound to the actual robot components through their predefined ID numbers. This allows the status of the robot which is relayed back to the user interface to be displayed graphically. Currently the language only permits a number of visual appearances to be selected between (like the *switch* construct in VRML) for each of these components, although more sophistication is intended in future versions of the language.

As a simple example of interface customisation via TRIL code, consider a function of the arm-controlling Behavior which causes it to automatically return the arm to a 'parked' position. The remote robot controller needs to inform the user interface that a button should be placed on the screen allowing this function to be invoked by the user. The fragment of TRIL code shown below customises the generic interface to do this.

```
BehaviorSpecialisation {
  behaviorID   ARM_CONTROL
  specialID    RESET_ARM
  function       Message {
          messageName "Reset"  }
}
```

The *functionality* field of this *BehaviorSpecialisation* node contains an element of type *message* which describes a simple message to be sent (i.e. a message with no parameters). The *messageName* field will cause a button labelled "Reset" to be placed on the interface window, grouped with other controls relating to the arm Behavior. When the user presses this button a message with an identity of *RESET_ARM* (a constant value) will be sent to the Behavior identified by the constant *ARM_CONTROL*. Upon receiving this message, the symbolic layer of the robot control system will then send the corresponding signal into the arm control Behavior causing the arm to reset itself.

A full interface definition for a type of telerobot will consist of a number of such descriptive nodes. Together with the information contained in the hand-shaking phase of the messaging protocol, this description will serve to generate a user interface which is both suited to that particular type of robot, and presents a user friendly front-end.

Figure 4 shows a telerobotic interface currently under development which has adapted itself to a robot connected to it. The features which the remote control system have added to the interface include the sensor display (underneath the video feedback) and interfaces to two remote Behaviors (at the bottom left of the window).

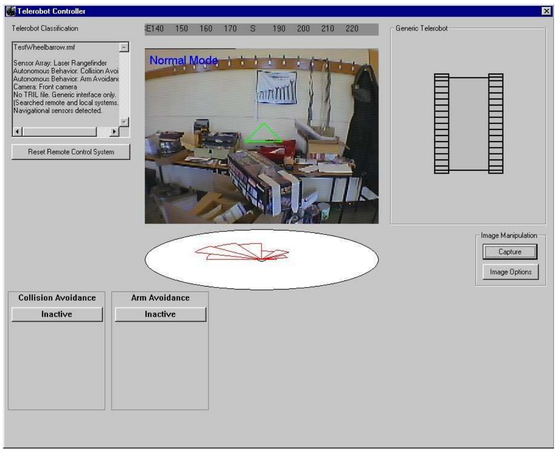

Fig. 4. - An interface currently under development

6 Summary and Further Work

This paper has described a specific telerobotic system, general system architecture and a protocol by which the local control system can be used to control different types of telerobot within a general class of mobile teleoperated robots. This architecture provides a 'plug-and-play' arrangement whereby telerobots customise the user interface to suit their own individual features and requirements.

The current versions of the PACT protocol and TRIL language require further development to allow them to cater for additional features of telerobots. It is the eventual intention of the protocol to be sufficiently abstract to contend with the interface requirements of components of telerobots which cannot currently be anticipated, possibly because they do not yet even exist. Additionally, there is a need for more sophisticated user interface elements; for instance animated displays of manipulator arm movement. One possible further development which it is envisaged might provide solutions to both of these problems is the inclusion of a scripting language into TRIL which may control screen elements. For instance, the current user interface assumes that sensor data will be in the form of obstacle distances, whereas a more sophisticated sensor might require quite a unique type of display. Using a scripting language, it would be possible for the remote system to send raw data to the local system for processing and presentation at the interface. Although the basis of TRIL would remain mainly descriptive, a high-level scripting language would allow small, highly specialised fragments of code to be executed on the local computer, thus providing much greater scope and versatility for man-machine co-operation via the telerobotic user interface.

References

[1] Anderson, R. J., "SMART: A Modular Control Architecture for Telerobotics," IEEE Robotics and Automation Society Magazine, Vol. 2, No. 3, Sept. 1995, pp. 10-18.

[2] R. C. Arkin, “Reactive Control as a Substrate for Telerobotic Systems”, IEEE AES Magazine, Vol. 6, No. 6, June, 1991, pp. 24-31.

[3] Betgé-Brezetz, S., Hebert, P., Chatila, R., & Devy, M., “Uncertain Map Making in Natural Environments”, in Proceedings of IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 22-28, 1996, Vol. 2, pp. 1048-1053.

[4] Brooks, R. A., “A Robust Layered Control System for a Mobile Robot”, IEEE Transactions on Robotics and Automation, Vol. 2, No. 1, 1986, pp. 14-23.

[5] Chen, C. X., Trivedi, M. M., & Bidlack, C. A., “Simulation and Graphical Interface for Programming and Visualisation of Sensor-based Robot Operation”, in Proceedings of IEEE International Conference on Robotics and Automation, Nice, France, May 12-14, 1992, Vol. 2, pp. 1095-1102.

[6] J. H. Connell, “SSS: A Hybrid Architecture Applied to Robot Navigation”, in Proceedings of IEEE International Conference on Robotics and Automation, Nice, France, May 12-14, Vol. 3, 1992, pp. 2719-2724.

[7] J. Connell, & P. Viola, “Cooperative Control of a Semi-Autonomous Mobile Robot”, in Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, OH, May 13-18, 1990, pp. 1118-1121.

[8] Graves, A. R., & Czarnecki, C. A., “A Novel Teleoperated Robot Control System”, to be published in Proceedings of Mechatronics ‘98, Skövde, Sweden, Sept 9-11, 1998.

[9] Ince, I., Bryant, K., & Brooks, T., “Virtuality and Reality: A Video/Graphics Environment for Teleoperation”, in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2, 1991, pp. 1083-1089.

[10] Moore, P. R., & Eriksson, P., “A Role for Sensor Simulation”, in Computer Aided Robotics, Proc. Of 26^{th} International Symposium on Industrial Robots, ISIR ’95, Singapore, 4-6 Oct, 1995.

[11] VRML97 Specification, VRML Consortium. URL: http://www.vrml.org/Specification/, Date: 16/06/98.