Supervised Autonomy: A Paradigm for Teleoperating Mobile Robots

Gordon Cheng and Alexander Zelinsky
Department of Systems Engineering
Research School of Information Sciences and Engineering
The Australian National University
Canberra, ACT 0200. Australia
email: gordon@syseng.anu.edu.au

Abstract

In this paper we propose a new paradigm for teleoperating a mobile robot. Our teleoperation paradigm is made up of five major components: "self-preservation"; "instructive feedback"; "qualitative instructions"; "qualitative explanations"; and a "user interface". Our aim is to provide a mobile robot with autonomy while being teleoperated. Our approach can be used to combat the continuous closed-loop problem. A qualitative approach has been taken in the design of each of the components. The usefulness of our approach is demonstrated with an implementation of a user interface to teleoperate an autonomous mobile robot equipped with a vision based navigation system. In this paper we we describe the implementation of our teleoperation system.

1 Introduction

Teleoperation systems have been a major part of robotics research. Much of this research was initiated during the nuclear reactor era of the 1960's [1]. The high level of interest was due to the large number of applications that were typically performed in hazardous environments that were too dangerous for humans. Examples of such work include the exploration of Mt. Spurr by Dante II [2]. Many other examples include toxic waste cleanup, military operations and mining etc. can be found in Sheridan's book [3].

Our interest in telerobotics is focused on providing a qualitative navigation system for mobile robots that can accept high level instructions. Our work is inspired by the problems of continuous close-loop control that are associated with most teleoperational systems. Typically these systems are of the direct control style that require constant monitoring by an operator of the robot's actions while sending a stream of commands to the robot by the operator. A side-effect of the constant monitoring and commanding of the robot by the operator is the burden of not just the overall mission but also the general safety of the robot. This produces an effect that Arkin [4] refers to as "cognitive overload". We share the same belief that by removing the burden of constant monitoring and commanding this will help reduce the cognitive load for the operator.

Some common strategies for relieving this load from an operator include providing a set of pre-programmed routines [1]. Other approaches use "planning by trying it out on a computer simulation first" [3]. Most of the reported work in simulation relies strongly on the use of an environmental model that must be sufficiently well understood before any pre-set routines can be derived and programmed. By taking the simulation first strategy, the solution to the problem relies completely on the accuracy of the modeling of the real world. Such strategies lack the ability to attend to unforeseen situations that will occur in real dynamic worlds. This may be possible for a robot manipulator working in a desktop environment. In mobile robot environments this isn't possible.

An interesting approach was reported in the Rocky 7 Mars Rover project [5]. Rocky 7 performs its navigation by traversing a path of way-points selected from a 2D image by a human operator. This approach removes the need for a complex path planner and allows the robot to use a simple algorithm to control movement between the way-points. However, while this approach is suitable for the Martian environment it can not handle dynamic worlds. After the operator selected a way-point for the robot to move towards, a dynamic obstacle could move to block the direct path or the way-point itself could move!

Our teleoperation paradigm consists of five major components; self-preservation, instructive feedback, qualitative instructions, qualitative explanation and of course a user interface. We will now explain each component.

One approach to teleoperation is to shift the general low level survival capabilities, such as collision avoidance onto the robot [4]. We believe this approach is a key characteristic that must be incorporated into any design of a teleoperational system. In this paper we outline the general architecture of our system. By providing the self-preservation attribute, the latency problem associated with direct control is eliminated.

An important issue in teleoperation is providing suitable feedback to the human operator. In our system we have chosen visual feedback. The same visual images are shared by the operator and the robot's vision processing system. Our choice is based on the fact that by sharing the same view with the robot more reasonable decisions by an operator can be taken. We refer to this as "instructive feedback". The images that are transmitted to the operator do not necessarily have to be live, as the communications bandwidth available to the robot-human interface could be low. Images at the end or the beginning of a action sequence could be sent to the operator.

A further attribute that must be considered is the nature of instructions used by the operator to command the robot. Shibata *et al.* [3] reported a system that used high level instructions to command a robot scooter to "Go toward" a given point in an image. In our system we have extended this concept further by providing more qualitative instructions. We have implemented "Go forward", "Go toward", "Go between" and "Keep to" instructions. The details concerning these qualitative instructions are presented in Section 4.

Since our system does not rely on direct feedback, a record of events that occur during mission execution should be recorded. We have chosen Purposive Maps (PM) reported by Zelinsky *et al.* [6] [7] to maintain a log of events. A PM is used to provide a qualitative explanation of the events that occur during the execution of an high level qualitative instructions. For example a person could be given the task of "making a cup of coffee". This might involve the following actions, pick up a cup, walk down the corridor, avoid collisions with people, open the door, approach the coffee machine, pour a cup of coffee etc. The implementation details of PMs are given in reference [7]. Using a PM the robot is able to communicate to the user, how well a task instruction was performed, and what behaviours within the robot were triggered during its execution e.g. collision avoidance. By maintaining a record of events within a PM, we have the ability to automatically re-execute a user task instructions.

In Section 2 we present the physical details of the robot system that we are teleoperating. Section 2.1 describes how command data and sensor data flows in our system.

It is well known that the design of the user interface in teleoperational systems is important to the overall efficient operation and functioning of the system. A bad design can lead to low productivity and poor operator acceptance of the system. Care must be taken in the overall design of the teleoperational system. As part of our investigation we have evaluated several different alternatives for user inputs and feedback. Our experiences have been incorporated into our overall design. We discuss this aspect in more detail in Section 5.

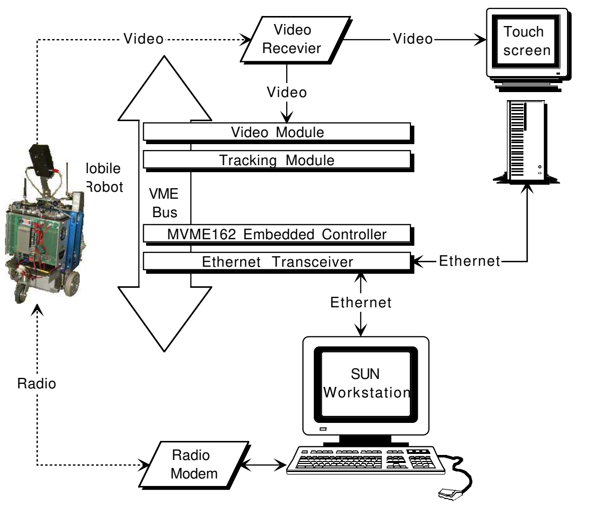


Figure 1: System Configuration

2 Our System

Our teleoperation system consists of four major hardware components; a vision processing system; a communication server; a teleoperational console; and a mobile robot. The logical relationship of these components is shown in Figure 1.

The robot we are teleoperating for our experiment is a Yamabico mobile robot built at the University of Tsukuba, Japan [8]. The Yamabico is a distributed and flexible mobile robot that has an extendible architecture. For our needs we have included a CCD camera with a video transmitter, and a radio modem. The video transmitter is used to transmit an NTSC video signal to both the vision system and the teleoperating console. The radio modem is used for all remote data transfers between all the machines. The video transmitter is rated at 1 km, and the data modem is rated at up to 700 metres. The teleoperational console we are using is a Apple Power Macintosh™ computer, equipped with an AV frame grabber capable of accepting NTSC video signals. Additional input devices have been incorporated onto the console, such as a unidirectional microphone for high quality sound input for speech processing, and also a touch screen. The touch screen replaces all the operations that are normally performed by a mouse.

A Fujitsu MEP template matching vision system is employed to perform all of the off-board vision processing that is required by all of the robot's visual behaviours. A similar vision system is used on the Hyper Scooter of Shibata *et al.* [9]. Further details of our vision processing are provided Section 3. For all our data communication needs we are using layered communication software developed by Jung *et al.* [10], named *Radnet*. *Radnet* provides all data packet routing to the robot from the vision system and the console via a communications server. This server is a SUN Sparc™ workstation equipped with a radio modem.

2.1 Flow of Commands

In the current implementation of our system there are four distributed server processes that form part of the infrastructure to support our work. Each process is responsible for all command flow and event notification between the processes. These can be divided into three categorises: direct robot commands, qualitative instructions and operator notification.

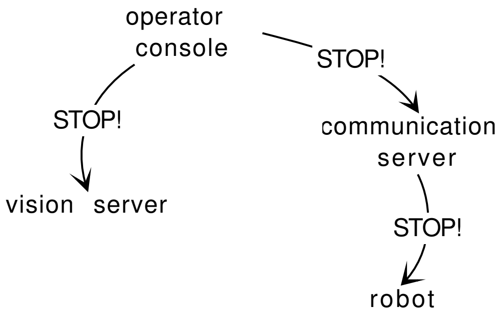


Figure 2: Direct robot command

The direct robot command is illustrated in Figure 2, here the operator has decided to abort the current mission. A "STOP!" command is issued, this command is sent to the robot and a notification is sent to the vision system.

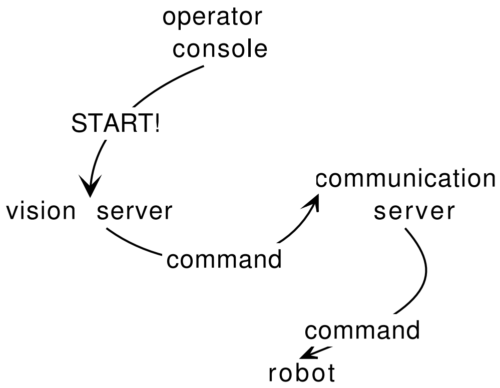


Figure 3: Qualitative Instructions

For qualitative instructions, as depicted in Figure 3, the operator performs all high level construction of a task, then instructions are sent to the vision server for execution. At this point the execution of commands to the robot are performed by the visual navigation behaviours. With operator notification, it is important to gain the operator's attention once the execution of a task can no longer be continued.

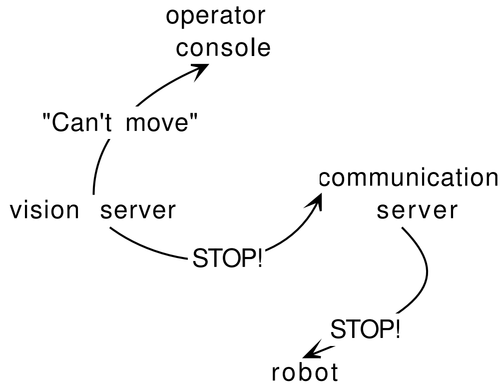


Figure 4: Operator Notification

The command flow of event is shown in Figure 4. An typical situation is depicted in Figure 5, here the dialogue between the system and the operator are as follow, at a) the robot was instructed to "Go To" an object(i.e. wheelchair), b) the robot is moving toward the object, c) shows the robot in "Stop!" mode. At this point the vision system commands the robot to stop, then notifies the operator that it can not move further and waits for the operator's next instruction.

a) b) c)

Figure 5: Operator Notification

2.2 Robot System Architecture

Behaviour-based systems have proven to be successful and effective for implementing autonomous systems that work in unstructured and dynamic environments [11] [12].

The general survival control architecture that we are using is shown in Figure 6. There are two low-level survival behaviours; Collision Avoidance and Free-Space-Move.

Collision Avoidance is provided to relieve the operator of the burden of worrying about the general safety of the robot. While performing an instructed task the Collision Avoidance behaviour acts as a safeguard against any harm that the instruction may cause to the robot. This is done through a visual search of free-space surrounding the robot. The vision process determines the available free space directly in front

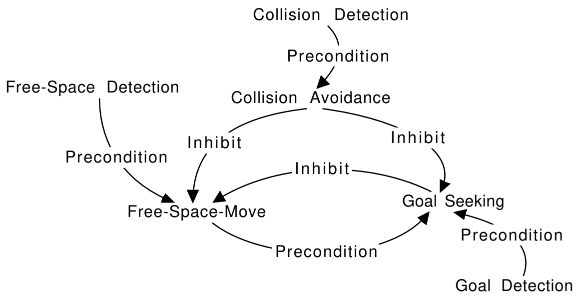


Figure 6: General Survival Architecture

of the robot. If insufficient free space is available, the robot suppresses its other navigational behaviours and activates the collision avoidance scheme, i.e. stops and notifies the operator that it can not move any further.

Free-space-move provides the robot with the ability to move freely within its environment. In effect this behaviour provides obstacle avoidance for both static and dynamic obstacles. This behaviour also relies on the vision process to search for free-space. The avoidance scheme determines the best available area of space for the robot to travel. By integrating these behaviours into our teleoperating system we have provided the robot with its own sense of self-preservation. Figure 7 shows an example of the robot avoiding a dynamic obstacle(a person walking). An important aspect of these behaviours is that all vision processing is performed in real-time (30 Hz). Currently our system uses a passive camera model of the floor space, and a planar world model is used in determining the position of objects. A brief explanation of the vision processings is provided in Section 3. Full details of the vision processing behaviours see references [11] [13].

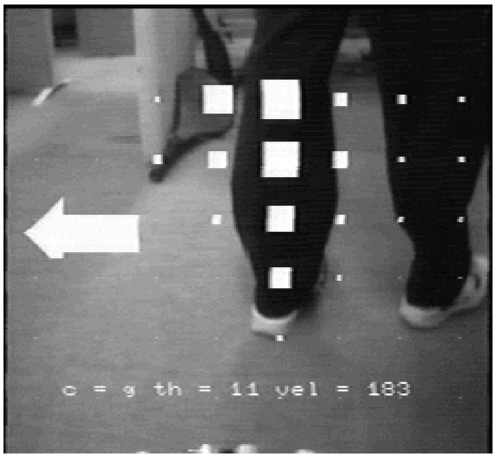


Figure 7: Obstacle Avoidance

The final component of our architecture is the Goal Seeking behaviour. This is an adaptable behaviour which assists the robot in performing all of its purposeful tasks. Section 4 will discuss each of the teleoperational behaviours that was derived from the Goal Seeking behaviour.

3 Vision processing

Here we briefly describe the general vision processing that are supports the behaviours in our system. As mentioned earlier the vision processing is based on template matching, the core of the matchings are performed through the computation of an mean absolute error of a stored image over an area of the live video. A distortion value D and a motion vector are returned from the vision processor. The template matching calculation is done by the correlation function shown in Equation (1). The motion vector is determined from a search of a given area for the lowest correlation value. Our vision system performs up to 100 reference template matches at video frame rate (30 Hz).

$$D = \sum_{x}^{m} \sum_{y}^{n} \mid g(x-m, y-n) - f(x,y) \mid \qquad (1)$$

where $g(x,y)$ is the greyvalue of the template at the specified coordinates
$f(x,y)$ is the greyvalue of the pixel in the last frame

3.1 Free space searching

The free space search process is as follow, correlation is performed on a 7×8 grid where each grid cell is 64×64 pixels in size. Each cell is correlated against a pre-stored set of template images of the floor. For each row of the grid one pre-stored image of the floor is used for correlation. From these correlated values a normalisation process is performed to eliminate the distortion from the lens. Normalisation is perform through the use of an error matrix. Then a threshold value is calculated from the variances between all of the normalised values, this is given by Equation (2). Figure 7 shows a resultant image of the free space search. The size of each white square represents the strength of the mismatch. The robot sees a moving obstacle on its right, causing the robot turn away to its left.

$$\sigma = \frac{\sum_n D(x,y) - \bar{D}}{n-1} \qquad (2)$$

3.2 Target tracking

Target tracking is an important part of our vision processing. To ensure smooth tracking of target images. A confidence value is calculated for each correlation match, and is shown in Equations (3). The confidence values are used to determine the ratio of positional changes for each of the templates currently being tracked. As shown in equations (4) and (5), the new tracking position is determined by this confidence value. The “Go To”, Go between” and “Keep to” behaviours all employs this target tracking scheme. Once the tracking position is determined it is then transformed into the robot coordinates. Details of the coordinate transformation are described in Cheng *et al.* [11].

$$\delta = -\frac{D}{\epsilon} + 1 \tag{3}$$

$$x_t = x_{t-1} + m_x \times \delta \tag{4}$$

$$y_t = y_{t-1} + m_y \times \delta \tag{5}$$

where x_t and y_t are both in *screen* coordinate
ϵ is the minimum confidence threshold
m_x is the changed in x direction
m_y is the changed in y direction

4 Qualitative instructions

To reduce the level of complexity in our teleoperational system we provide four qualitative instructions. They are: Go Forward, Go To, Go Between and Keep To. Each of these instructions are inspired by basic qualitative notions of “Keep going until you can’t...”, “Going to this place...”, “Going between these two landmarks..” and “Keeping to this wall...”. Each of these qualitative instructions and their associative visual behaviours are discussed individually.

4.1 Go Forward

This instructs the robot to start moving ahead in its current direction until the collision avoidance behaviour notifies the operator otherwise. Each notification is recorded in a PM. The usage of the PM will be discuss in Section 5.

4.2 Go To

This qualitative instruction is used to instruct the robot to move toward a given object, whose image was selected by the operator. Various effects can be achieved from this behaviour, such as visual servoing, following and posing. Visual servoing can be simply achieved through the operator selecting a static object and the robot moving toward it while keeping it in its view. A side effect of visually servoing to an object, is that if the object was to move the robot will start to follow it. Finally, posing can be accomplished by setting the minimum distance that a servoed object can be approached.

Figure 8 shows an example of the GoTo instruction. In this case the user selected the toy car as a landmark. The robot can servo on the landmark, and if the car moves as shown in the example, the robot follows it autonomously. While the robot is in the following behaviour mode it is also able to avoid obstacles. If the robot loses sight of the landmark, the robot stops and sends a notification to the operator.

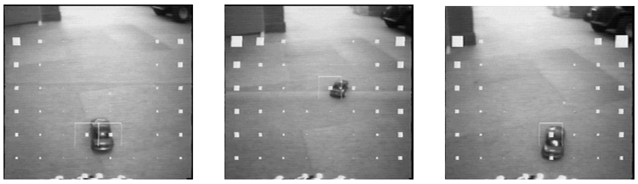
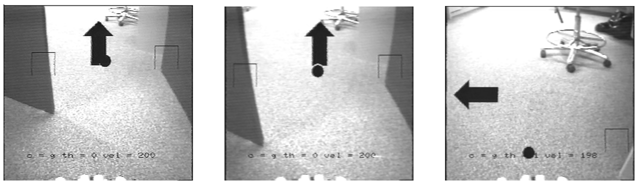


Figure 8: Go To Qualitative Behaviour

4.3 Go Between

In this instruction mode the operator deliberately selects two horizontal template images that are used as landmarks. Qualitatively the robot is instructed to travel between these two landmarks. The motion trajectory is simply calculated by determining the average floor positioning of these two landmarks, this can be seen in Figure 9. The figure shows the robot being instructed to pass through a doorway.



a) b) c)

Figure 9: Go between

While traveling between these two landmarks the vision system maintains the two image’s positions. If the system loses track of any one of these images, or if the positioning error of the images become significantly large, a notification is issued back to the operator and the robot stops. This behaviour is useful for commanding the robot to move through tight spaces such as doorways.

$$x_t = x_1 + \frac{x_2 - x_1}{2} \tag{6}$$

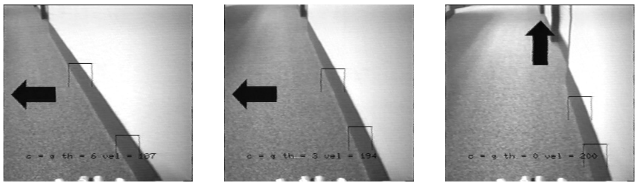
$$y_t = y_1 + \frac{y_2 - y_1}{2} \tag{7}$$

where x_t and y_t are both in *robot’s* coordinate

The robot’s motion trajectory is determined by the center of the two landmarks, and this is given by Equations (6) and (7). Once the robot has moved sufficiently close to the two landmarks and if the target point is sufficiently centered, then the robot begins to align the horizontal displacement of the two landmarks.

4.4 Keep To

For this qualitative instruction the operator needs to specify two diagonally adjacent template images. This determines the trajectory along which the robot will travel. For example if the skirting of a wall is used, the robot follows along the wall until it loses sight of the wall or tracking fails. The calculation of the robot's trajectory corresponds to the angular relationship of the two diagonal templates, shown in Figure 10. The vision processing for this behaviour is similar to the others, but the search area for template matching is altered according to the angular relationship. The position of the templates are adjusted accordingly to the velocity of the robot.



a) b) c)

Figure 10: Keep To

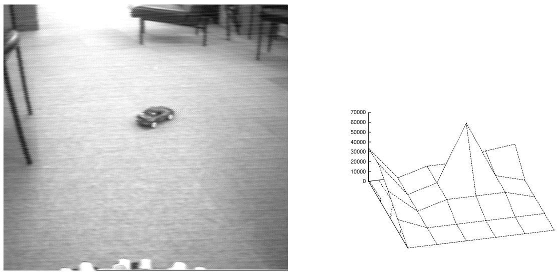
$$\theta = \tan^{-1}(\frac{x_2 - x_1}{y_2 - y_1}) \tag{8}$$

where x_t and y_t are both in *robot's* coordinate

Using the first template(closest template), the robot servos until it is sufficiently close to the it, e.g. 30cm to a wall. This stage of processing is similar to the "Go To" behaviour. Once the robot has reach the desired distance, then the robot's alignment is calculated using Equation (8). Thus, θ determines the zero alignment for the robot to track along.

4.5 Make a Suggestion

An additional feature that we have incorporated into our system is to allow the vision system to suggest to the operator a landmark or landmarks that are interesting. This command instructs the vision system to determine from its current view a distinctive feature that it perceives to be a good candidate as a landmark for tracking purposes. Such a command is important because the perception capabilities of the operator are far more sophisticated than those of the robot. Therefore by allowing the robot's own sensor to make such suggestions ensures that the most effective landmarks are selected. We have developed an interest operator that derives a tracking target from the correlated values used for the free-space search process. The Interest-operator is applied to the normalised correlated values described in the vision processing section above. For example in Figure 11 we can see the effect of the Interest-operator. The unknown toy car is shown as a single peak after the operator has been applied and thus can easily be identified as an interesting landmark feature.



a) b)

Figure 11: Finding unknown landmark targets

4.6 Purposive Map Making

Once a robot has been instructed to perform a particular task, the execution of the task needs to be recorded and presented back to the operator. This is to ensure that the operator is well informed of the current and past events that have occurred during the execution run. The visual presentation to the operator should include which behaviours were activated and also in what situation did the activation occur. In our implementation we have used Purposive Maps (PM) to provide a representation of this information. A key property of PMs is the ability to express events and locations qualitatively.

An example of the Purposive Map is shown in Figure 12. This illustration shows places and behaviours, no geometric representation of the world is used. The PM representation of the visual behaviours allows the robot to start from a Home location and safely move about the environment until it finds a target on which in visually servos. Other advantages of using PMs are that a general representation can be learnt via a simple run of an experiment, further execution using the same PM representation can enhance system performance. For further details refer to reference [6].

Presently our system a graphical presentation for displaying features of a PM, after the operator first constructs a task by specifying a qualitative instructions, such as "Go to that object". During execution the robot may activate various behaviours and send notifications to the console e.g. the robot could avoid obstacles while trying to reach the target. Each notification is recorded in the PM and then displayed on the user console.

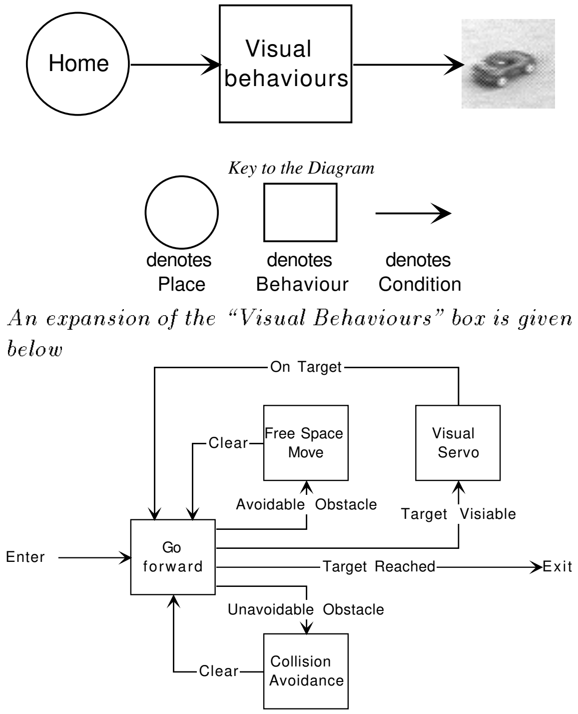


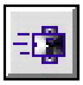
Figure 12: Visual Behaviours

5 Implementation

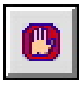
5.1 Teleoperation console

A minimalist approach has been taken in the design of our user interface. A screen dump is shown in Figure 13. A video display from the robot's video camera forms part of the interface, this is displayed within the main console window. A message log window is provided, that displays any errors and messages from any of the servers. The window below the main console is the Qualitative explanation window, which displays a PM as it is being constructed. The main controls within the main console window are on the bottom of the video display.

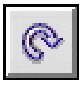
We will briefly explain the main controls:



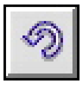
This icon instructs the robot to begin executing a set of qualitative instructions.



This icon commands the robot to abort the current actions it is executing.



This icon commands the robot to start spinning right.



This icon commands the robot to start spinning left.

The spin icons are used to give the operator a pan view of the scene while deciding which visual behaviour

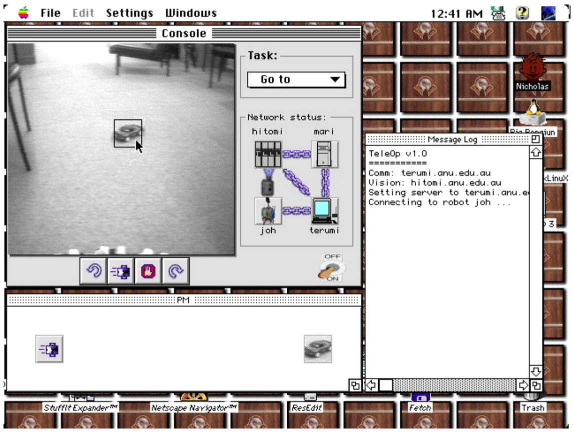


Figure 13: Console

to select for navigation from the Qualitative instructions menu.

The user interface takes advantage of the Macintosh look and feel. The touch screen provides all of the functionality that can be accessed by the mouse. Thus, an operator may instruct the robot to "STOP" simply by placing his/her finger over the stop button.

Qualitative instructions are constructed by selecting from a list of menu items that are located on the right side of the main console. This ensures only one instruction is executed at a time. Beneath the menu is a status indicator which gives the communication status between each server. The user selects a qualitative instruction from the touch-screen menu and then selects the visual landmarks that are to be associated with this instruction, by pointing at the relevant parts in the displayed image. For example after selecting the GoTo instruction, the user selects a landmark in the image to move towards. In the case of the Keep To and Go Between instructions, the user selects two landmarks by touch the screen at the appropriate places in the image.

Other researchers have also implemented similar user interfaces for various command systems. Sekimoto *et al.* [14], uses a touch panel to control a robot's motion to a target position. The robot has no autonomy and would drive into a wall if commanded to do so. The system that reported by Shibata *et al.* [9] also uses a touch screen for selecting an image of an object to construct a "Go toward" instruction. However, this system can also be commanded to collide with obstacles.

5.2 Mission

Figure 14 shows an example of a mission in an indoor office environment(our laboratory). At ① the robot was instructed to "Spin" right, at ② to "Go Forward", at ③ to "Go Between" and out to the corridor. Next at ④ to "Keep To" the skirting boards of the wall, then ⑤ "Spin" left, ⑥ to "Go Between" and through the door to another room. And finally at ⑦ to "Go To" an object on the floor.

The "Keep To" has been used to follow walls. The "Go Between" has been used to drive through doorways out into corridors and into offices.

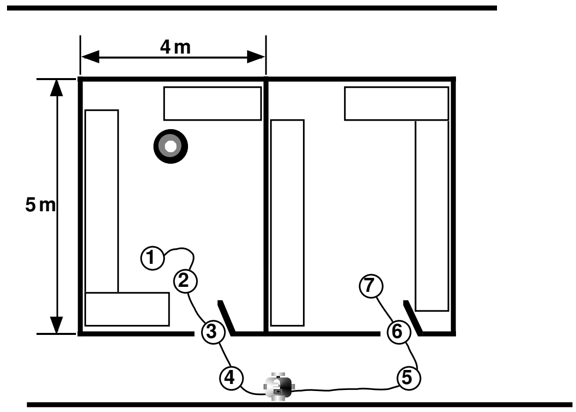


Figure 14: Mission

6 Conclusions

We have proposed "Supervised Autonomy", a teleoperational paradigm to combat some of the existing problems in mobile robot teleoperation. Specifically, the "cognitive overload" and the latency problems. The paradigm is aimed at relieving the stress loads of an operator by shifting many of the low and intermediate level controls to the robot. As part of the paradigm we have incorporated instructive behaviours, these behaviours form high level "Qualitative Instructions". This approach serve as an example in constructing systems without the need for full controls by an operator in accomplishing a task. An implementation was completed with a Yamabico Mobile Robot over a data radio network in an indoor office environment.

In this paper we have focused on the current state of our system, which was mainly geared toward the development of low to intermediate level behaviours, that are incorporated into high-level instructions. In further work, we will be focusing on evaluating and improving the User Interface, i.e. the operator's console.

Acknowledgments

This research is supported by the Australian Research Council. Special thanks are extended to David Jung for his implementation of *Radnet*, and to Setyani Sutedjo for her help in porting *Radnet* to the Power Macintosh.

References

[1] W. R. Corliss. *Teleoperators: Man's Machine Partners.* United States Atomic Energy Commission, Office of Information Services, 1972.

[2] D. Wettergreen, H. Pangels, and J. Bares. Behaviour-based Gait Execution for the Dante II Walking Robot. In *International Conference on Intelligent Robots and Systems*, 1995.

[3] Thomas B. Sheridan. *Teleorobotics, Automation, And Human Supervisory Control.* 0-262-19316-7. MIT Press, 1992.

[4] R. C. Arkin and K. S. Ali. Integration of Reactive and Telerobotic Control in Multi-agenmt Robotics Systems. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994.

[5] R. Volpe, J. Balaram, T. Ohm, and R. Ivlev. The Rocky 7 Mars Rover Prototype. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1558–1564, Osaka, Japan, November 1996. IROS'96.

[6] A. Zelinsky, Y. Yuniyoshi, T. Suehiro, and H. Tsukune. Using Argmentable Resource to Robustly and Purposefully Navigate a Robot. In *Proceedings of IEEE Robotics and Automation Conference*, pages 2586–2592, 1995.

[7] A. Zelinsky and Y. Yuniyoshi. Learning to Coordinate Behaviours in Mobile Robots. In *Journal of Advanced Robotics*, pages 143–159, 1996.

[8] S. Yuta, S. Suzuki, and S. IIda. Implementation of a Small Size Experimental Self-Contained Autonomous Robot. In *Proceedings of the 2nd Int. Symposium on Experimental Robotics*, pages 902–909, Toulouse, France, June 25–27 1991.

[9] T. Shibata, M. Inaba, and H. Inoue. Instruction Use by a Vision-based Mobile Robot. In *Proceedings of Workshop of Integration of natural Language Processing and Vision Processing*, 1994.

[10] D. Jung and B. Stanley. Yamabico Autonomous Mobile Robots—Reference Manual. Technical Report 1.5, Department of Computer Science, 1995.

[11] G. Cheng and A. Zelinsky. Real-Time Visual Behaviours for Navigating a Mobile Robot. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 973–980. IROS'96, 1996.

[12] I. Horswill. Visual Collision Avoidance by Segmentation. In *Intelligent Robots and Systems*, pages 902–909. IROS'94, 1994.

[13] G. Cheng and A. Zelinsky. Real-Time Vision Processing for a Soccer Playing Mobile Robot. In *Proceedings of the 1997 World Robot Cup*. IJCAI'97, 1997.

[14] T. Sekimoto, T. Tsubouchi, and S. Yuta. An Implementation of a Human Interface using a Touch Panel and Visual Images for Controlling a Vehicle. In *Proceedings IEEE International Workshop on Robot and Human Communication*, Tsukuba, Japan, November 1996. ROMAN'96.