ONLINE ROBOT EDUCATIONAL ENVIRONMENTS

Gerard T McKee
Department of Computer Science, The University of Reading, Reading, UK
gerard.mckee@reading.ac.uk

Abstract

In this paper we review work at the University of Reading over a number of projects exploring the use of multimedia and networking technology to create online-robot laboratory facilities that can be shared across the Internet. The projects have progressed from an emphasis on the evaluation of technology to a closer integration and exploration of online robot laboratories within educational programmes. The paper emphasises recent work that highlights the role of multimedia technology in the assessment of student work, in contrast to its early role in the development of the online robot facilities. Students have found the online robot scenarios created for assessed project work both challenging and rewarding. The assessment of the student work, however, requires the implementation of alternative strategies that both allow the students to demonstrate the functionality of their implementations and reflect the considerable effort that many students put into their projects.

Keywords: Online robots, Internet robotics, education, WWW.

1 INTRODUCTION

Multimedia and networking technology has played an important role in the development and application of robotics technology. The impact has been most observable in providing the opportunity to create online robot demonstrations [1-4]. These systems have highlighted the scope for robotics education via the Internet by providing opportunities for students and general members of the public to control real robots in a range of motivational scenarios. Internet robot stores and educational robot manufacturers have also seen the benefits of providing similar services to illustrate their products. Some projects have also explored the use of these environments in student projects [5,6]. The PumaPaint project, for example, allowed students to develop skills in computer interface development for remote control [6]. However, considerable work is still required to establish a framework and a rich set of educational facilities that can support the persistent use of these environments in education.

In this paper we describe a set of projects developed at the University of Reading aimed a exploiting networking and multimedia technology to provide laboratory facilities that can be shared across the Internet. The early motivation for the first project, NETROLAB, was the then recent development of high bandwidth networks [5]. The project was motivated internally by the desire to give many more students access to a mobile robot system. Supervised access was costly of staff resources. The desire for a more efficient provision of limited laboratory resources also reflected the more widespread need otherwise to maintain industrial level laboratory resources to support robotics education, a cost that prohibited many universities providing such facilities to support their teaching programmes.

The more recent development of the goals of the NETROLAB projects is the smaller scale TORUS project [7]. The focus in TORUS has shifted to using simple toys to create small scale, but nevertheless challenging, robot scenarios that will motivate a range of educational topics in robotics and artificial intelligence. A stronger focus is also placed on integrating these scenarios into educational programmes as assessed student projects. One such project, described in this paper, develops the theme of a three-player game scenario based on construction-site toys, the TORUS Construction Site (TCS) scenario [9]. This project has in turn motivated further issues in the need for multimedia support for both conducting the project and assessing the resulting student work.

The MVIDEO software system is a more recent project aimed at providing image services for students who are creating simple stop-look-act control systems spanning a remote online robot system (the TCS arena) and the student's local workstation. The application of the MVIDEO system to a recent student project has highlighted its benefits both to the students when implementing their projects and for the teachers when monitoring the online environment and its use by the students. This same application has also highlighted the need to develop better strategies for assessing student work that involves online robot systems. A more empirical reporting of the results would help reduce the overheads of assessment and better reflect the considerable effort that many students put into the completion of their projects.

The remainder of the paper is organised as follows. The following section summarises the main components and conclusions of the NETROLAB project. Section 3 describes the TORUS project, focusing on the TCS scenario. Section 4 describes a student project developed from the TCS scenario that focused on providing some level of automated control for a toy digger. Section 5 describes the MVIDEO project, whose aim was to provide image support for both remote viewing and image processing functions. Its incorporation in the most recent, the second, run of the digger project is described. Section 6 discusses the assessment of student projects that use online robot systems. It proposes a set of deliverables that students should provide for the assessment of this work. This includes the developoment of a permanent record of experimental results. In the digger project this is best provided as an animated sequence of images that illustrate the most successful task runs. The provision of such a deliverable can go a long way to easing the process of assessing the student work and providing a better reflection of the effort the students put into the project. Finally, section 7 provides a summary and conclusions.

2. NETROLAB

The NETROLAB project was motivated by practical and economic problems in the delivery of robotics education to a large body of students. Specifically, creating a multiple-station robotics laboratory requires significant funding which cannot be justified in the vast majority of educational settings. If at least one such laboratory could be established, and access to it could be provided via the Internet, many more students could get access to robotics systems. NEROLAB, in short, aimed to investigate how the combined power of high bandwidth networks and multimedia workstation technology could be successfully harnessed to bring otherwise inaccessible resources to a wider audience than hitherto possible.

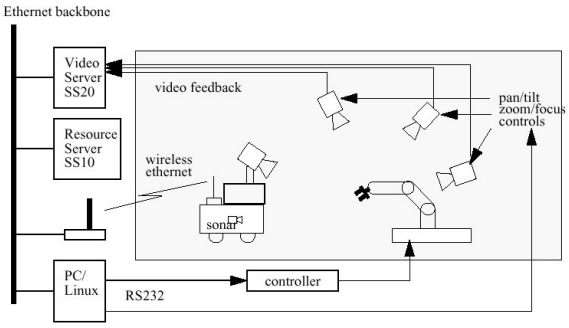

Figure 1. Netrolab established a set of robotics devices as networked resources

The practical goal of NETROLAB was the creation of a shared robotics laboratory on the Internet by "networking" a set of physical robotics resources, including vision and sonar sensing modules, a manipulator arm, and a mobile robot, housed within an environment located at the University of Reading. Five basic requirements were established for the provision of such a laboratory service, namely the coverage of the subject of robotics and the support for individual student experiments, for group-based experiments, for software development activities and finally for the concurrent use of the experiment facilities by multiple students. The conventional model of a laboratory environment comprises a set of resources which are configured as needed to support a diverse range of experiments (Figure 1). A similar model was adopted for NETROLAB. The notion was that experimental modules could be created by recruiting a subset of the NETROLAB resources to satisfy the practical needs of a particular experiment (Figure 2). In order to access these experiments across the Internet the following additional requirements were established:

1. The physical robotic devices require hardware and software interfaces which are customised to support network-based access.
2. The resources must provide a set of services that can be used to configure a wide range of experiments.
3. Primitive and complex resources, the latter composed from primitive resources, need to communicate with each other and with the user, across the network.

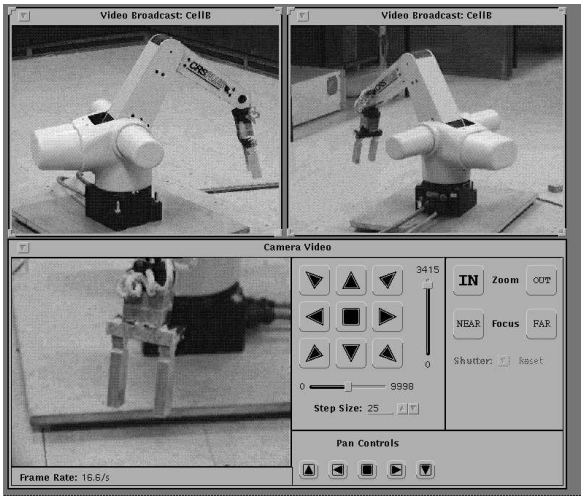

Figure 2. Multiple camera resources (video streams and pan-tilt & zoom-focus controls) could be configured to create a viewing palette for remote control of a manipulator.

In order to support these requirements a resource-based client-server model was implemented employing object-oriented techniques and implemented in C++. In this model each resource was implemented as a server providing a set of services to client applications. For example, each joint of a robot

manipulator would be treated as a separate resource. These could be recruited along with a video source and a simple client interface for remote control of a manipulator could be configured. A more complex configuration might employ multiple cameras. The controls for the cameras are also resources that could be configured to create a camera control application that could run on the same workstation or on a separate workstation, allowing students to perform advanced teleoperation experiments.

NETROLAB demonstrated the potential of technology, specifically advanced networking and multimedia technology, to create new types of educational resources. Remote viewing and control in various forms were demonstrated, reflecting early Web-based online robot demonstrations and inteface-based programming projects [2]. However, the full benefits of these types of facilities in extended educational programmes has still to be achieved. One of the important requirements is to develop technology and tools to reflect the specific requirements of projects that involve the creation of robot control architectures.

3. TORUS

TORUS (Toys Operated Remotely for Understanding Science) is a project aimed at exploiting toys to create interesting and challenging robotics demonstrations and problem scenarios for students. The main advantage of using toys rather than industrial-level robot systems is the low cost of purchase and the minimal maintenance and safety requirements. There are also many robot-like toys available, allowing students and other to replicate on-line scenarios, and to develop unique scenarios of their own. The TORUS Construction Site (TCS) scenario illustrates the general approach incorporates three toys remotely accessible via the Internet within the context of a three player game (Figures 3-5).

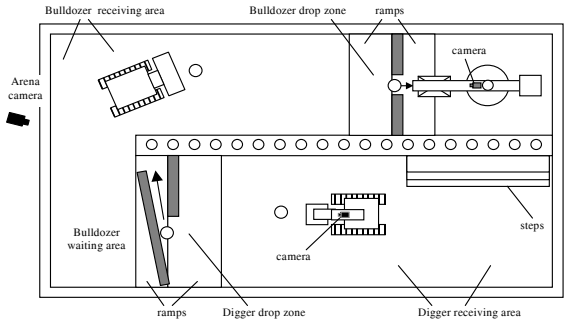

Figure 3. Diagrammatic overview of the TORUS Construction Site Arena

The TCS environment comprises an arena split up into three sections. Each section is associated with a different toy device, namely a tower-crane, a digger and a bulldozer respectively. The three sections of the arena form a circuit and the task is to move a ball around the circuit in the sequence bulldozer-tower_crane-digger. Each player in the game has control of one of the devices, which in turn implies that they are taking charge of the corresponding section of the circuit. The players have access to a number of camera views. An arena camera located off the arena provides a view of the whole arena. A small camera mounted on the digger provides a view ahead of the digger. A third camera view is provided by a small camera mounted on the tower-crane just below the cab so that it looks downwards and moves with the turret. Each player can switch between any one of these views during the game.

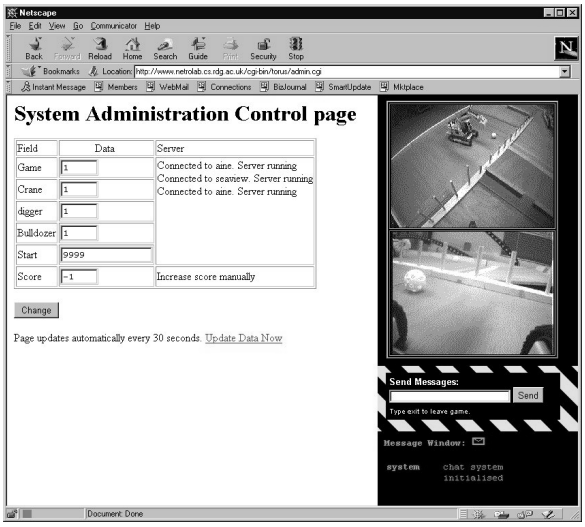

Figure 4. The TCS administration page. Used to control access, set game parameters and to monitor the arena through the tower_crane and digger cameras.

The software environment of the TCS comprises three basic components, namely the device servers, the video server and the game server. All three servers are implemented in the programming language C running on Linux boxes that support the corresponding device interface. Management of the TCS game is provided by a password protected web page based on a Perl cgi script. The administration web page (Figure 4) allows administration and observation of the game as it progresses. The game server, the heart of the implementation, comprises a number of web pages allowing users to enter and play the game. A control page provides users with a simulated joystick interface, a video window to display the remote arena with a choice of three camera views, a message window for users to communicate with each other, a command window that displays the commands sent to the device and the reply, and a countdown timer which is nominally set to 5 minutes. The tower-crane control page is illustrated in Figure 5.

Figure 5. The tower_crane control page

The educational value of the TCS is in demonstrating the concept of teleoperation through a practical remote control scenario where optimal task completion rate is the main performance objective. The game demonstrates, in the first instance, the concept of remote control. In the second instance it demonstrates the user interfaces required for control of the remote devices. In the third instance it demonstrates the importance of video feedback as a method of "seeing" the remote workspace. It illustrates the importance of multiple camera viewpoints and indeed the benefits of mobile camera views over fixed cameras. The scenario chosen and the placement of the cameras on the tower crane and digger, for example, help illustrate the viewing options available to human operators and the restrictions thereof. The latter help to motivate the need for additional support for these operators. The tower-crane illustrates this particularly well.

The TCS also illustrates various forms of robotic manipulation including grabbing, carrying and pushing. Indeed, some ingenuity is required to avoid the task (i.e. the ball) going astray; for example, if the ball gets lodged against the sides of the arena or in a corner. The TCS also calls on the players to think about the task that is to be performed and the limitations that are placed on them by remote operations because of the necessity to not only control devices but also to think about viewing.

In addition to demonstrating teleoperation and various forms of cooperation, the TCS environment also provides opportunities to motivate advanced issues in robotics. One area is the design of user interfaces, perhaps with joystick, spacemouse and other forms of input devices. Another area is the provision of enhanced visualisation of the task environment, including video overlays and task planning/previewing facilities, and ultimately the immersion of the operator within the environment through telepresence and augmented reality systems. Yet another area is the study and development of telerobotic control by adding local intelligence and seeking still greater degrees of autonomy. This in turn raises issues of shared control, multiple robot systems and human-robot systems.

The development of the TCS scenario raised a number of issues concerned with the user interfaces, the scaling of the robot devices to their environment, and also in fact to the difficulty of the task itself if the arena is poorly designed. However, it also motivated intermediate and advanced student projects that extend the existing hardware and software. One key area for development is providing some local autonomy for each of the toy devices. Specifically, it motivated a student project, the digger intelligence project, that has been the focus of recent work on TORUS. This student project has become more tightly integrated with undergraduate teaching programmes than any of the previous projects.

4. DIGGER INTELLIGENCE – AUTOMATED CONTROL

A natural extension of the TCS scenario is the addition of some local intelligence to the devices. A student project was developed, therefore, whose aim was for the students to develop a level of autonomy for the devices. To complete the project the students would need to understand the particular task the device had to perform, the way in which image feedback and the device controls can be employed to interrogate the environment for task information, and to implement a simple stop-look-act control strategy that realises a solution. The project focused on the addition of a level of autonomy to the toy digger.

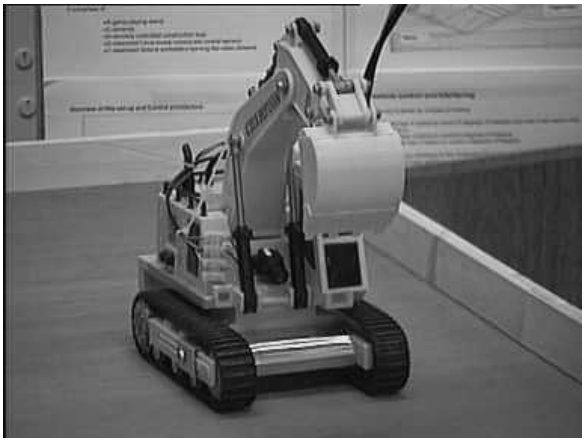
Figure 6. Profile of the Digger

The digger (Figure 6) is a tracked toy that has both forward, reverse and rotational motion controlled by the relative direction of drive to two motors controlling the two tracks, respectively. It has, as well, an arm and a bucket. Both can be moved up and down within certain limits. A miniature camera is mounted in the belly of the digger, providing a view of the area towards the front of the digger. The

digger's task can be broken out into the following six steps (refer to Figure 7):

1. Locate the ball. This can be achieved with a simple scanning operation.
2. Traverse towards the ball until it is within the grasp-zone.
3. Activate the arm and bucket of the digger to affect a scooping motion to pick up the ball.
4. Locate the drop-zone for the ball. This is another scanning operation, similar to locating the ball.
5. Traverse to the drop zone.
6. Drop the ball into the drop zone.

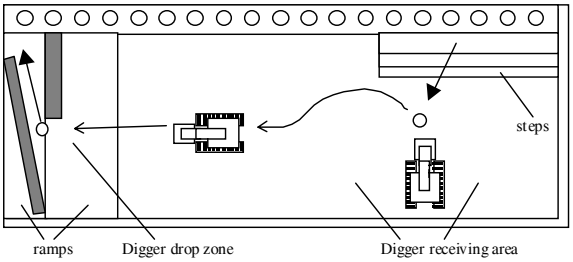

Figure 7. Illustration of the digger task components

The project focused on the first half of this task, namely picking up the ball. There are three major tests associated with the task, and all must be performed using the image data from the camera mounted on the digger. The first, during the localise stage, requires detecting the presence of the ball in the image. The second, during alignment, requires determining whether the centre of the ball is within some threshold distance from the centre of the image. The third involves determining whether the ball is in the grasp zone.

A suite of software was developed to support the project. Its main components were two servers, a control server for controlling the digger and a video server for grabbing images from the camera mounted on the digger. Both servers were implemented in the programming language C. Students provided a username and password to login to the servers. A small library and a simple client program were implemented in Delphi, the designated language (procedural) for the project. These could be downloaded from a web site specifically set up for the project.

The project was set in two parts to a body of just over 100 students attending a taught module on robotics and artificial intelligence. The goal of the first was to provide a simple interface for remote control of the digger. The goal of the second was to automate the control of the digger during the pickup task. This formed the major part of the overall project. Students were required to provide two levels of control. The first, manual control, as per the first part, but to include the ability to manually designate the centre and boundary of the ball in an image. The second level was automated control and was based on automatically locating the centre and the boundary of the ball from the image data. The students were given a short introduction to histogram-based image segmentation techniques as a suggested method for locating the ball. They were also required to define labeled buttons that initiated each element of the task separately, and one button that caused the sequence of steps to by executed automatically. The student were given 6 weeks in which to complete this second part of the project.

On completion of the project the students were asked to provide feedback on their experience. In general, they found the project challenging and they felt that they had learned a lot from it. The fact that students were interacting with a “real” environment which provided various challenges of its own, including poor repeatability of the digger commands, seemed to be an important factor in their engagement of the project. The students were also relatively clear about the code library provided on the project web site. They found it both not very useful and hard to use. This might reflect their limited experience with library use and management. The student were less conclusive in their assessment of the accessibility of the servers and the quality of the documentation. Some felt that the queue to gain control of the digger was sometimes very long and the time allotted to each user was too long also. Analysis of logs kept during the project, however, showed that there was often no more than three users on the queue. These are all issues that require further study. One of the major improvements that was deemed useful, by both the students and the teachers, was better provision for the delivery of image grabbing functions. This motivated a further look at video delivery through the MVIDEO project.

5. The MVIDEO Project

Online robot demonstrations have shown the benefits of the Internet technology for education. Many of these demonstrations incorporate vision feedback to allow the user to view a mobile robot or manipulator arm perform some action [1]. For remote manipulation or guided traversal vision data provides a rich source of information. For automated control, as in the digger project, the image data needs to be delivered to an application where it can be interrogated for task-relevant information as part of a stop-look-act control strategy. The MVIDEO project aims to develop a suite of software to satisfy these needs. It uses a multi-port model of service delivery – each service type is provided at a separate port, eliminating in many cases the need for the client application to negotiate a particular service. The

MVIDEO software provides video streaming via multicast and unicast connections and includes support for a variety of client bandwidths. The project builds on and evolves software developed in a number of previous projects.

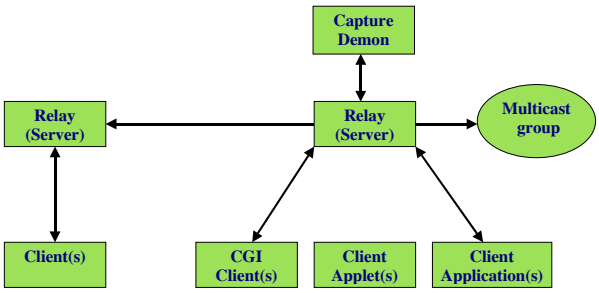

Figure 8. MVIDEO system concept

The main components of the MVIDEO system are a capture demon, a relay server and a number of utilities including a Java application, JAVA applets, and a CGI script. The basic system concept is illustrated in Figure 8. The function of the capture demon, written in C, is is to capture video from the target device and to forward it to a relay server via a unicast connection. The relay server, written in Java, is the core of the MVIDEO system. It is responsible for sending video data to clients and multicast groups. It supports RAW and JPEG image formats and can convert between video formats. Hence, it can receive a JPEG stream and forward it in a number of formats. The relay server delivers the video stream as a continuous sequence of images. Users do not need to login to receive the images, they simply connect to the appropriate port on the server, or the multicast group, and begin receiving images. An additional 'request-image' server is also provided aimed at synchronisation of control with image delivery. Users connect to this port and request an image of the required format by sending one of the strings 'jpeg' or 'raw'. Users login to the image server was a feature employed in previous image delivery services used for student projects. It was deemed not to be a requirement for the MVIDEO project – the multi-port service aims to accommodate any custom requirements of a user. More services can be added on additional ports as required. In addition to supporting transmission of video, the relay server incorporates functionality to display statistics on the usage of the services via a web browser. Figure 9, for example, shows the basic set of statistics for the DiggerCam to be described shortly.

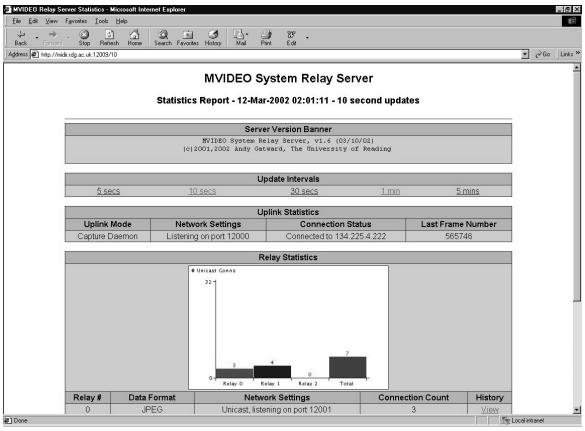

Figure 9. Usage statistics provided by relay server

The MVIDEO utilities are three in number at present. The first two are a Java application and a Java applet, and the third is a CGI script written in Perl. These provide basic stand-alone and in-browser viewing functions. The Java application shown in Figure 10 supports multicast and unicast connections. It is configurable for name server, group address and port numbers, and allows images to be saved as RAW or JPEG files. It is a lightweight application that makes use of a set of libraries that are shared by the relay server and the Java applet. The Java applet runs in the Java 1.3.1 or later runtime environment. It is configurable for a splash screen test card to be shown when not connected to a server. It does not have multicast support due to the security restrictions placed on applets. The Java application and the applet together provide very useful facilities for students to view the digger's environment and to download and save images for off-line processing and analysis. Finally, the CGI script is written in Perl 5. It performs a 1-shot-grab of a still image from a video stream. It is configurable for unicast connections using URL parameters and no multicast support is provided.

Figure 10-a. The Java client application

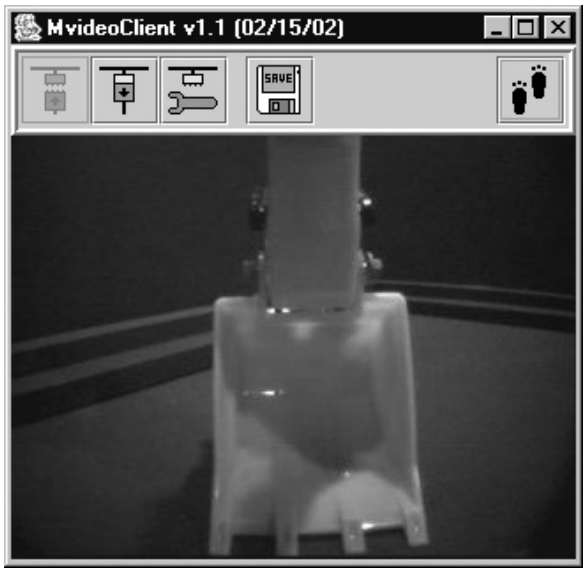

Figure 10-b. The Java client application

The MVIDEO system is replacing existing video services in a number of online robot scenarios we have developed for student projects, and is being employed in the development of a number of new online scenarios. We note first that it is currently being used to re-implement the TORUS TCS scenario [7], which depended in its initial implementation on server push technology for video delivery to clients. In this paper we focus on the application of the MVIDEO system to the "digger intelligence" project assignment. A new 'digger arena' has been developed to support the project, providing better control of lighting, camera viewing, and with scope to add more complex scenarios and viewing facilities (Figure 11).

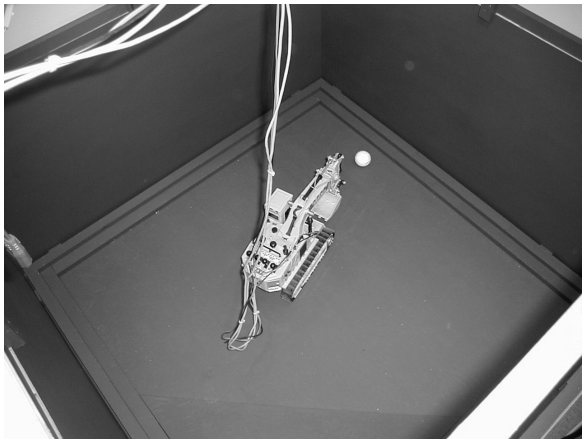

Figure 11. The new digger arena

For the more recent implementation of the project the students were provided with two views of the environment from, respectively, a camera mounted on the digger - the DiggerCam - and a camera mounted overhead - the ArenaCam. Only the DiggerCam had been provided previously. The two views provided by the cameras via the MVIDEO applets are illustrated in Figure 12. Streaming jpeg and raw image services were provided, but no multicast or image grab on request. WWW pages were created to support the assignment and to allow the students to view the applet-based streams from a web browser, and to download the Java application. The basic computing environment comprised two Linux workstations, the first providing the WWW server and hosting the MVIDEO relay server. The second supported the capture demons for the two cameras. The image capture devices were WinTV PCI cards to which were connected the two miniature CMOS colour cameras, the DiggerCam and the ArenaCam.

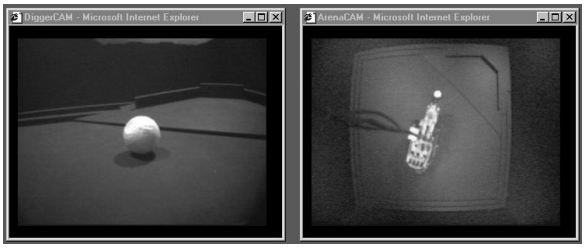

Figure 12. DiggerCam and ArenaCam views

This new environment was vastly superior to the previous. The easy access to the two video streams provided via the applets was extremely useful both for staff and students. For the former it provided a means of noting that the environment was being used, for recognising situations where the ball might get stuck in a corner, or identifying if the tether became too tightly wound up. Additional monitoring of the student logins to the control server provided an extremely powerful but lightweight environment for identifying students having problems running their programs. From the student's perspective there were a number of points to note. First, the students liked the easy access to the applets for viewing the image streams, and made use of the Java application both for viewing and for saving images to file prior to the introduction of these facilities in their own programs.

Second, many of the students opted to download raw images rather than jpeg images, and many also tended to read the images from the image stream a byte at a time. Both of these approaches slowed the applications significantly. This crude approach reflected the inexperience of the students with network programming and some of the lesser-known features of the Delphi programming environment. Some of these students were later able to switch to the jpeg image stream and take advantage of inbuilt facilities for displaying and manipulating jpeg images.

Third, instead of maintaining a permanent network connection to the image streams, and grabbing images as needed, many of the students opened the connection, grabbed the first image on the stream, and then closed the connection until they needed another image. This again appears to reflect

inexperience with more advanced programming features.

Fourth, many of the students addressed the lack of facilities to support synchronization between the digger controls and the image sequences by introducing a time delay into their program and experimenting with a sensible duration for the delay. This was required because of the time delay between the capture of an image and its arrival at the student's application. This problem is addressed through the 'request image' feature, which will be available in future projects. More advanced student projects, however, would also benefit from prior expore of the students to network programming, multi-threading and advanced GUI programming facilities for image display and manipulation.

Fifth, just under about 10% of the students were able to get their program to complete the task automatically. This was a noticeable improvement on the previous set of students. This success is due to a number of factors including the benign nature of the environment (it was relatively easy to locate the ball), good reliability and fine control of the digger movements, and a slightly more advanced set of students, by one academic term, compared to the previous set.

Finally, a small number of these students were encouraged to create a more permanent record of their success. This permanent recorded typically included an animated gif of the camera views during one of the successful sequences.

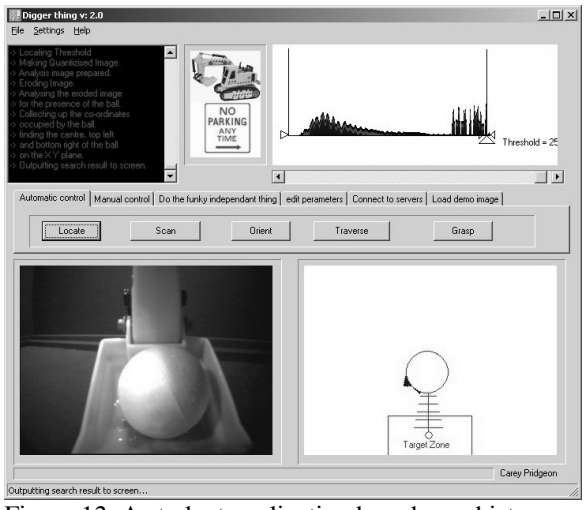

Figure 13. A student application based on a histogram technique for locating the ball

Figures 13 illustrates one examples of a completed student project. The figure shows a screen shot of the student's application interface. This student took the suggested route of locating the ball using a simple histogram technique. The implementation also incorporated morphological pre-processing functions, such as erosion, to clean up the image. The range of functionality was impressive for this level of student.

6. ASSESSMENT OF STUDENT WORK

The demonstration of program code is an important means of assessing student projects involving software development. For the digger projects, both previous and new, students are nominally allowed 5 minutes in which to demo their program. The demos are carried out in a signle PC laboratory and students wait their turn on a first-come first-served basis. Demo session for the previous project had employed one assessor (GTM) and a number of techniques were used to help them run smoothly. These involved observing the student's program for simple image download and processing to locate the ball, for recognising misalignment of the ball, for orientation of the digger relative to the ball, and finally a small section of the traverse towards the ball. Generally, the setup was such that the ball was always kept within the view of the camera. On completion of one demo the digger could be rolled back a couple of moves and rotated slightly so that the ball was off-centre in the camera view.

The demos for the most recent run of the project were organised similarly, except for two modifications: a second assessor was employed and a stronger effort was made to observe the full task sequence, from scanning the horizon for the ball through to eventual pickup. The overall result, however, was frustration and disappointment from the perspective of the assessors and for many of the students. One of the main problems was coordinating the use of the robot environment between the two assessors. This included situations where the ball went out of view of the DiggerCam and when one demo lost control of the digger to the other due to timeout of their control session. Verbal communication (requests, queries) between the asssessors was the means of addressing these, but this only added to the frustration of completing the demos. In some cases it was simpler to go into the laboratory and manally adjust the digger and ball so that they were set up for the next demo. The coordination problem was most noticeable when one assesor had to leave to attend to other duties. The remaining portion of the demo session progressed more smoothly.

The main conclusion one can draw is that although there are clear benefits of having multiple assessors available during the demo session, these can be outweighted in the case of online robot projects if the assessors do not have sensible coordination policy. It seems best to have one assessor. The question arises as to how to improve matters for both teachers and students. One option is to have multiple versions of the online robot system, but that defeats the purpose of the facility in the first place. Another option is to

provide more time for each demo, but that takes more staff resources and, even then, many of the students will not get their programs to complete the full task sequence successfully in the time alotted for their demo. A further option is to be minimalist in what is assessed, namely to assess the separate functions and exclude any attempt at completion of the full task sequence, but this approach doesn't do justice to the considerable effort that students put into the assignment. Many students did indeed complain that they were not being given sufficient time to demo work for which they had contributed a considerable amount of effort. It is important, however, to view these problems in the context of the wider Internet and the opportunities it offers for distance learning. In this wider context demos may not need to be carried out in focused demo sessions, allowing participant more time to demonstrate the program's achievements. However, there may be a much larger numbers of individual demos to be assessed, leading again to frustration and a significant assessment bottleneck. Another very important aspect of the wider Internet setting is that the assessor may be remote from the user performing the demo.

It is possible to draw a tentative proposal for the assessment of student work for online robot environments. The proposal builds on practice that underpins the reporting of research at conferences and workshops. Namely, student should include a "record of experimental results" with their submission for assessment of the project. In general, the students should submit a set of four deliverables to be assessed:

- An executable version of the program, for assessing the basic functionality of the application.
- A permanent record, typically in the form of an animated gif or a movie sequence of successful or nearly successful task sequences.
- A report (paper), presenting the key features and design components of the application.
- Source code, primarily for the purpose of assessing good programming style, code layout and code documentation.

For local assessment the demos sessions can focus on the basic functionality of the application. The students can play the permanent record, the movies sequences or their equivalent, as part of the demo. The minimal requirements placed on the online robot system and the lower expectations placed on the live portion of the demo, would mean that less coordination is required between the assessors and much of the frustration can be eliminated.

Remote assessment of student work becomes a modification of this procedure. In this case the student can deposit the deliverables with the assessor as a zip file via email or a browser-based upload facility. The assessor can then run the executable to assess tbe basic functionality of the system and review the other materials for overall assessment of the work. Existing video conferencing tools can be employed at this stage to support direct interaction with the student and between the assessors. Some form of queueing mechanism may be required for focused demo sessions. The students can also publish their results, programs, report and demos on their home web pages, providing a visible, public record, of their work. If this approach is to be successful, however, it is important that the students are given guidance on how to assemble and present animated movies of task runs. It is important, in turn, to identify and develop tools that will support this process. These issues, and the evaluation of this proposal, require further study.

7. SUMMARY AND CONCLUSIONS

In summary, we have described a number of projects undertaken with the goals of both exploring networking and multimedia technology to support online robot educational environment and integrating these environments within educational programmes. The NETROLAB project demonstrated that it is feasible to create online robot facilities that can be shared across the Internet. The more recent TORUS project has focused on the tigher integration of these environments with educational programmes. A number of student projects have been developed. As these are refined the assessment of student project work involving online robot systems is also coming under scrutiny. Multimedia technology has a role to play here as well. The smooth and efficient assessment of student work in local and remote settings will benefit both the students and the teachers. The TORUS project will continue to develop this theme.

References

[1] K. Goldberg, S. Gentner, C. Sutter and J.Wiegley, The Mercury Project: A Feasibility Study for Internet Robots, IEEE Robotics and Automation Magazine, Special Issue on Internet Robotics, December 1999.

[2] J. A. Fryer, Remote-control experiment using a networked robot, Special issue of SPIE Robotics and Machine Perception Technical Group Newsletter on "Networked Robotics", Vol. 5, No. 1, p. 12, 1996.

[3] R. G. Simmons, R. Goodwin, K. Z. Haigh, S Koenig, J. O'Sullivan and M. M. Veloso, Xavier, Experience with a Layered Control Architecture, Sigart Bulletin, Vol. 8, 1997.

[4] R. Siegwart and P. Saucy, Interacting Mobile Robots on the Web, Workshop on Current Challenges in Internet Robotics, IEEE International Conference on Robotics and Automation, Detroit, MI, USA, 1999.

[5] G. McKee and R. Barson, Using the Internet to Share a Robotics Laboratory, International Journal of Engineering Education, Vol. 12, No. 2, 1996.

[6] M. R. Stein and K. Sutherland, Project Update: Sharing resources over the Internet for robotics education, SPIE Proceedings Vol. 3524, Telemanipulation and Telepresence V, pp. 180-188, 1998.

[7] G. T. McKee and K. Phillips, TORUS: Toys operated remotely for understanding science, SPIE Proceeding Vol. 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, 9pages, Nov. 2000.

[8] G T McKee & R M Maunders, Exploiting Toys and the Internet for Robotics Education, WSES International Conference on Robotics, Distance Learning and Intelligent Communication Systems (RODLICS 2001), Malta, October 2001. Published in Advances in Signal Processing and Communicationis, V.V.Kluev & N.E. Mastorakis (Eds), 2001, pp. 281-286.

[9] G T McKee, The Development of Internet-Based Laboratory Environments For Teaching Robotics and Artificial Intelligence, Proceedings of ICRA 2002, Washington, USA, 2002.