An Online Robot Educational Project

Gerard T. McKee, *Member, IEEE,* Andrew J. Gatward, and Duncan I. Baker

***Abstract*— In this paper we present the further development of an online environment to support student projects in robotics and artificial intelligence. We describe an arena that has been constructed to offer a realistic challenge for a vision based task that involves locating and picking up a simple object. A toy digger, the 'robot', is integrated into the environment and wired into a computer for online access. A new suite of video software, MVIDEO, provides better support for remote visualization of the 'digger arena' than provided in the previous version of the online environment, and for the vision processing required in the student project. The new environment has been used by a slightly more advanced set of students, creating a better fit with their background and leading to more successful projects. New issues are now arising concerning the overall assessment procedures for these projects. The paper describes the new arena, the overall performance of the online robot system in the most recent run of project. Recommendations are made for changes that should improve the assessment procedures.**

***Index Terms*— Online Robots, Educational applications, Video streaming.**

I. Introduction

Online robots have offered a new medium for teaching topics in robotics and artificial intelligence, and indeed for inspiring interest in a wide range of science, art and engineering subjects [1]-[3]. The research reported in this paper focuses on the tighter integration of these types of online robot environments, through practical student projects, with educational courses. Following early work on networking traditional robotics environments we are now exploring the use of toys to support online robot educational projects [4], [5]. Realistic and challenging student projects can be created by placing these toys in an interesting task setting.

In previous papers we have described the origin and development of an Internet-based student project that involves the students creating programs that embody a toy digger with the ability to pick up a simple object [6]. In this paper we describe the further development of this environment. A new 'digger arena' is described that offers better support for remote viewing and finer control of the digger. The toy digger is also better protected against wear and tear. The new arena makes the project task, namely getting the robot to pick up an object, more readily solvable while continuing to offer an interesting challenge to the students. The basis for the improved viewing support is the incorporation of a custom video delivery system, MVIDEO, that is more closely tailored to the project requirements. The project has also been set to students with a slightly more advanced level and background than before, resulting in improved performance. The further experience gained with this new environment has led to new proposals for both monitoring students during these projects and for assessing the completed work of students for projects of this form. The running of the experiment for the second time, for a body of over 100 students, also provided an opportunity to extend our study of the performance of the system through the analysis of the system logs.

The remainder of the paper is organized as follows. The following section presents the MVIDEO system, including the motivation behind its development. Section III presents the Digger II arena, including the integration of the new toy digger and the characteristics of the computing environment supporting control of the digger. Section IV describes the performance of the new environment for a new body of students. Section V provides an analysis of assessment procedures for the student work and recommendations for new procedures to reflect the particular requirements of assessing online robot projects of this form. Finally, section VI provides a summary and conclusions.

II. The MVIDEO System

Online robot systems often use video feedback to allow users to view the mobile robot or manipulator arm that they are controlling [1]. This video feedback is typically in the form of within-browser single image snapshots using server push or applet-based video streaming. The goal of these systems is *remote viewing*. They do not, therefore, provide ready access to the underlying image data, a requirement for online robot projects that involve the development of vision guided automated control of the online robot. Further, more advanced forms of video streaming technology often use data compression formats that are not readily interrogated by external programs, and typically do not have server-based access to single images on request. Finally, video conferencing systems, a possible alternative to these browser-based and video streaming systems, are aimed at group communications

G. T. McKee is currently with the Department of Computer Science, The University of Reading, Reading, Berkshire, RG6 6AY, UK. (e-mail: gerard.mckee@rdg.ac.uk).

A. J. Gatward is currently a postgraduate research student with the Department of Computer Science, The University of Reading, Reading, Berkshire, RG6 6AY, UK. (e-mail: a.j.gatward@rdg.ac.uk).

D. I. Baker is currently a postgraduate research student with the Department of Computer Science, The University of Reading, Reading, Berkshire, RG6 6AY, UK. (e-mail: d.i.baker@rdg.ac.uk).

and are, therefore, top heavy relative to the requirements of vision-guided robot control projects. In short, there is inadequate support across a wide range of video delivery systems and environments for online robot projects where the students aim to write programs that provide vision-based control of a remote robot system.

The essential requirements of these projects are video streams that support remote viewing, that can be readily intercepted by student programs, that provide support for a variety of image formats (including raw images), and offer support for synchronizing image capture with robot control. In order to address these limitations, and to allow us to build a better environment for our students to carry out these types of projects, we have developed a software environment, MVIDEO, for delivering these requirements [7]. The MVIDEO system uses a multi-port model of service delivery, whereby each service type is provided at a separate port. This eliminates in many cases the need for the client application to negotiate with the server for a particular service. The MVIDEO system provides video streaming via multicast and unicast connections and includes support for a variety of connection bandwidths. The MVIDEO system, and software, builds on experience we have developed over a number of previous online robot student projects we have undertaken.

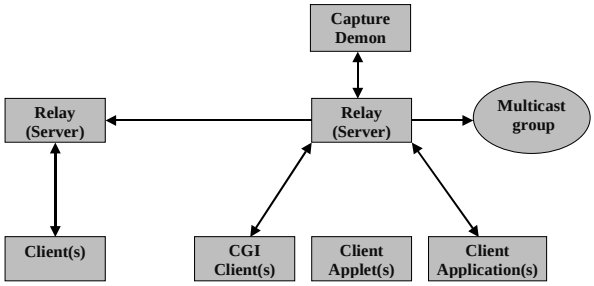

Fig. 1. The MVIDEO system concept.

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 Fig. 1. The function of the capture demon, written in C, 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 transmits video data to clients and multicast groups. It supports RAW and JPEG image formats and can convert between video formats. For example, it can receive a JPEG stream and forward it as a JPEG or a RAW stream. 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. A client-pull service is also provided, aimed at synchronization of control with image delivery. Users no longer need to explicitly login to the 'image server', as they did with our previous environment [6]. In addition to supporting transmission of video, the relay server incorporates functionality to display statistics, via a web browser, on the usage of the services. This facility provides a means of visually monitoring the current and recent level of unicast connections by service type.

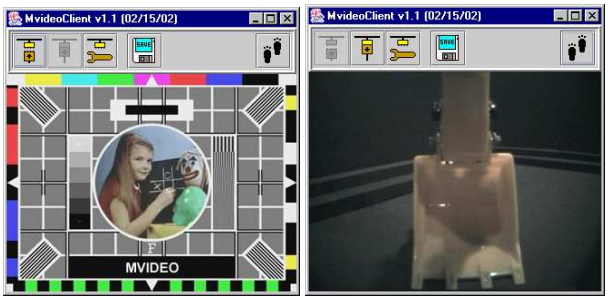

Fig. 2. The MVIDEO Java client application.

The MVIDEO utilities are three in number at present: a Java application, a Java applet, and a CGI script. These provide basic stand-alone and in-browser viewing functions. The Java application, shown in Fig. 2, 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 a very powerful set of facilities for students to view the digger's environment, and to download and save images for off-line processing and analysis. The applets, in particular, also provide a ready mechanism by which supervisors and teachers can monitor the project arena. 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. Ongoing work is continuing to improve and develop the MVIDEO system, including a web-based configuration environment and a wider range of compression formats.

III. The Digger Arena

The student project is to develop a control program that enables a digger to automatically pick up a ball. The project comprises elements of localisation (of the ball), traversal and manipulation. The control is to be based on using image data to locate the ball and align the digger so that the ball is directly ahead. The digger can then approach and grasp the ball using its arm and bucket. The latter is an open loop process, so it is important for the digger to approach close to the ball. The project is carried out using an online environment that comprises the following key elements:

- The physical arena housing the digger.

- The hardware interfaces to the digger and cameras.
- The MVIDEO software system.
- The digger command server.
- A WWW site with instructions, data formats, and access to the MVIDEO applet and application.

The digger arena, shown in Fig. 3 (left), comprises two main elements, the arena frame and the digger. The arena frame is made of wood with a basic box shape and an open framework roof. The fixed internal surfaces of the arena are painted matt black to reduce reflections and to provide a good contrast between the ball and its surrounding surfaces. The lower surface, the base, supports a simple two-tier step to prevent the ball getting caught in corners or sides, where it would be unreachable by the digger. Bases of different designs can be inserted to support different projects. The upper framework of the arena provides support for mounting lights and cameras. For the present project it mounts a miniature CCD color camera, referred to as the ArenaCam, providing an overhead view of the digger (Fig. 4, right).

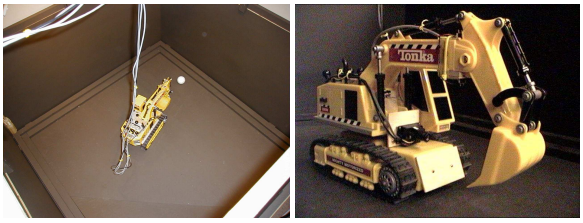

Fig. 3. The Digger Arena and the toy Digger (a Tonka toy).

The digger, shown in profile in Fig. 3 (right), is a Tonka toy that has been modified in three ways to support the project. First, a cable has been introduced to provide external access to control the four motors in the digger's base. These motors control respectively the two wheel tracks, the arm and the bucket of the digger. A 9-pin connector has been added to allow both computer control and a pendant-based joystick control. Second, limit switches were introduced to cut power to the arm and the bucket when these reach their mechanical limits. This prevents unnecessary wear and tear due to crunching of the gearboxes when the arm and bucket are driven to their mechanical limits. Finally, a miniature CCD camera, referred to as the DiggerCam, is mounted on the digger, just to the side of the cab. This camera provides the sole source of on-board sensor data for the student project. The DiggerCam view is illustrated in Fig. 4 (left).

Computer control is based on a simple 8-relay card that provides bi-directional control of each of the motors on the digger. (left and right tracks, arm and bucket) [6]. The DiggerCam and the ArenaCam are plugged into Hauppage WinTV PCI cards. Both cards are mounted on a PC running the Linux operating system. A tether carries cables for control of the digger, the power supply for the DiggerCam, and the return video feed from the latter.

The system has two main software components, namely a digger command server and a video server. The digger command server allows students to gain control of the digger. The students provide a username and password to login to the server. Their applications must include facilities for this login sequence. When a user logs into the digger command server they join a queue. Each user on the queue can have control of the digger for a fixed period of time. The current duration is approximately four minutes. When a user gains control, they can issue one of a set of possible commands, including clockwise or counterclockwise rotation of the digger, forward or backward movement, arm up and down motions, and bucket in and out movements. The commands can be given an optional parameter, in the range 1 to 4 (default 2), to control the duration of the motion. One unit in the case of translation or rotation motions of the digger corresponds to a quarter of a second, whereas for arm and bucket motions one unit corresponds to half a second. A single command is represented by a string of the form "<cmd> <duration>". For example, "cw 2" would rotate the digger clockwise for 2 time units (i.e. 1 second).

Fig. 4. Example DiggerCam and ArenaCam views.

The video server is the MVIDEO system described in the previous section. The views provided by the DiggerCam and the ArenaCam are illustrated in Fig. 4. Finally, the WWW site provides instructions for the assignment and allows the students to download the MVIDEO applet and application. The basic computing environment comprises two Linux workstations, the first supports this WWW server and the MVIDEO relay server. The second supports the capture demons, the interfaces for the two cameras, and the interface to the toy digger for the digger command server.

IV. Performance

The project, using the new arena and the MVIDEO server, was set in two parts to a body of just over 100 students attending a taught module on robotics and artificial intelligence. The majority of the students where studying for a single or a joint honours degree in Computer Science. All were on the second year of their courses. The students were given a short introduction to histogram-based image segmentation techniques as a suggested method for locating the ball. They were required to develop a user interface comprising labeled buttons that initiated each element of the task separately, and one button that caused the sequence of steps to by executed automatically [6]. The students were given 6 weeks in which to

complete the first part of the project and 4 week to complete the second. They took the first part of the project during the fall term of their second year and the second part during the immediately following spring term.

This schedule was different to the first run of the project, when the students took the first part during the summer term of their first year and the second part during the fall term of their second year. Hence the current group of students were slightly more advanced than the previous group. In addition, the previous group were allowed six weeks in which to complete the second part, whereas the current students were allowed only four weeks. A further restriction was also placed on the students during this second stage: the digger command server was available during only the final week of the second part of the project. Despite these limitations, however, the level of performance and success improved. In addition, the students were free to select the language for implementing their programs. Most selected the Delphi programming environment, since this was the main programming environment for their course. A few implemented their programs using C, under Linux.

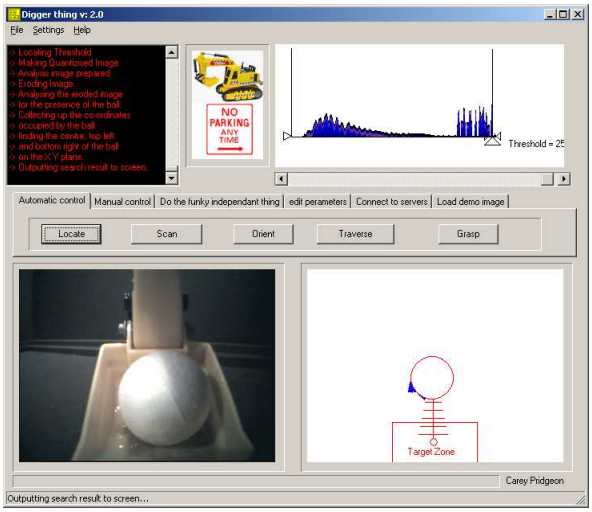

Fig. 5. A sample student application.

Fig. 5 illustrates one example of a completed student project. This student adopted the suggested method of locating the ball using a simple histogram-based technique. The implementation also incorporated morphological pre-processing functions, such as erosion, to clean up the image. These had not been explicitly taught to the student.

In general, the students were impressed with the visualisation environment provided by MVIDEO, both via the applets and the downloaded java application. Students working on the project were regularly seen with both camera views (DiggerCam and ArenaCam) visible on their monitor using the applets. The students did not have access to the client-pull image service for this run of the project. Most, however, identified the synchronization problem, namely selecting an image to process on the completion of a control command, given that the video lagged behind the event completion signal from the digger command server. The tended to solve the problem by introducing a time delay into their program before grabbing the next image from the stream. The duration of the time delay was determined through trial and error.

One area of the project that caused poor performance on the part of the students programs was the method that the students adopted to read an image from the video stream. Many attempted to do this a byte at a time. Those that used the raw image stream had even poorer performance, since the image size was much larger compared with the JPEG stream images. Further, when grabbing a succession of images, many students reconnected to the server each time, rather than maintain a permanent connection. Some students also found it difficult to cope with reading an image from the JPEG stream and additionally gaining access to the raw image data. When the procedure was explained to them, normally by other students, they were often quick to change over to the JPEG image stream in order to improve the performance of their program. These findings reflect the limited experience of these students with network-based programming and the less used multimedia features of the Delphi programming environment. The more skilful programmers among the students were able to find the better (more efficient) solutions for most problems.

One of the key goals of studying the use of the environment during the student project was to monitor the pattern of usage of the environment by the students, and to compare these with previous results. However, it was also possible to provide some more detailed analysis of system usage than before. Fig. 6, for example, shows the number of connections and logins per day over the period when the digger command server was available to the students during the second part of the project. The deadline for the assignment is marked by the vertical line in the figure. A login represents successful completion of the login procedure to enter the queue for controlling the digger. Connections, on the other hand, represent a successful connection to the server, but includes both successful and unsuccessful logins (including deliberate disconnection by the student's program). The graph shows the pattern we have seen before, and expected, namely the increasing use of the digger command server as the deadline approaches. The connections and logins following the deadline were for periods when the students were allowed further access in preparation for the demos of their code for assessment. In all cases the number of connections is larger than the number of logins, which is to be expected. The failed connections appear to increase as the deadline approaches. Further studies are required to determine the reasons underlying these differences. It cannot simply be the difficulty of programming the login sequence, since the students completed this during the first part of the assignment.

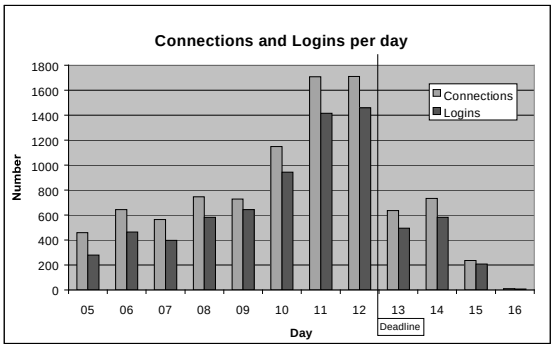

Fig. 6. User connections and logins per day

The number of concurrent connections on the digger command server per day, for the second part of the project, are shown in Fig. 7. This again follows the pattern seen previously. The numbers are generally higher, however, reflecting most probably the limited time when the command server was accessible. There are also occasions when the number of connections reaches the maximum of 20 allowed. On further inspection of the log files these appeared to be due to a student program making repeated connections to the server, as if it had entered an infinite loop. The maximum of 20 users acts as a safeguard against this type of situation. However, these events suggest that a further check should perhaps also be made to limit the connections from any one machine. It would also be helpful to the student if there was some way to inform them that their program was failing in this way. These will be explored in the further design of the environment.

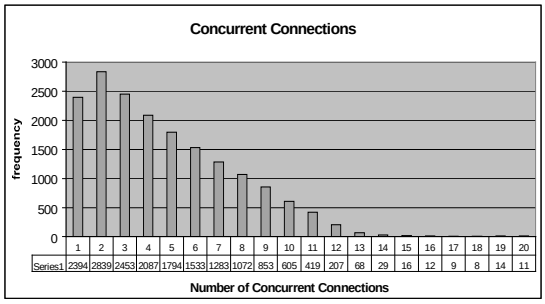

Fig. 7. Numbers of concurrent connections

Fig. 8 provides another visualisation of these data, giving a plot of changes in the level of concurrent connections over the same period. The time axis is nonlinear in this case, since it simply records changes in the numbers of connections. Three situations can be observed when the number of concurrent connections reaches the maximum. For the majority of the time the number of concurrent connections remains below 15, and during periods of intense use remains around 9 or 10. This confirms from the previous results that the maximum of 20 is realistic for this number of users.

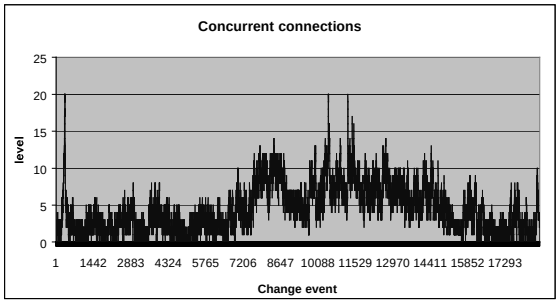

Fig. 8. Changing levels of concurrent connections

As a further level of analysis, Fig. 9 shows a histogram of the duration of successful login sessions. The figure shows that there are a large number of logins that are less than one minute in duration. This seems surprising, but could reflect basic program testing. A sizeable number of login sessions last for more than the maximum of five minutes allowed at the top of the queue. In some cases these longer durations reflect students waiting extended periods to reappear at the top of the queue – when a user is removed from the top of the queue they are automatically re-entered at the bottom of the queue.

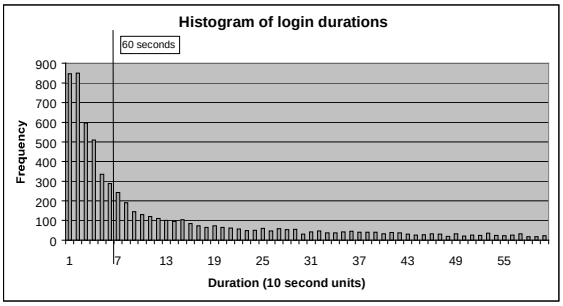

Fig. 9. Histogram of login session duration

Finally, there were occasions when the digger command server seemed to grind to a halt. On many of these occasions the problem arose because the student repeatedly issued a string command and either did not wait for a ‘command completed’ response to come from the server and/or concatenated the new command onto the end of the output command buffer. This meant that on occasions very long command sequences were received by the server, filling up the command buffer. The natural approach to cope with these events is to have the server intercept them. However, it is also important on these occasions to inform the student that something is wrong with the code and where the fault may lie. Thus it was found very useful to monitor the command messages received by the server on a per user basis. On a number of occasions this provided the basis for very helpful feedback to the students. In general, the most productive method of monitoring the students, and the environment, was to have the command server display incoming messages on one monitor and to have the two video streams (DiggerCam and ArenaCam) displayed on the same or another monitor.

This aspect of online monitoring is a prime area for further research and development, and for the creation of a new range of support tools for online robot projects.

V. Assessment Procedures

The work that the students complete for the Digger project is assessed largely through a laboratory demonstration of the program code. The students are allowed 5 minutes to demo their program. The demos are carried out in a PC laboratory on a first-come first-served basis. Two assessment sessions are normally held, each lasting typically three hours. Two assessors are typically present for at least one of these sessions. The digger command server queue is used to coordinate between the assessors, where were the only users allowed to login during the demo sessions. These procedures are largely based on practice that had been successful before. However, a number of problems arose during the demonstrations for this current group of students.

The major problem that arose was coordinating the online robot environment between the two assessors. This included situations where the ball was out of view of the DiggerCam and when the current user lost control of the digger before key functions could be demonstrated. In general, it was best to have the ball in the left or right visual half-field of the DiggerCam. This saved valuable time and allowed the relatively quick demonstration of orienting functions. If the ball was not visible, the ArenaCam would need to be used to quickly locate the ball in the arena. However, the MVIDEO applet or application needed to have been initialised when the student demo started, otherwise time was lost starting them up. This became frustrating, and on occasions it was simpler to manually reset the digger arena – the lab was just along the corridor. The second problem, loss of position at the head of the queue, meant that the assessor would either have to wait to return to the head of the queue, or ask the second assessor to relinquish control while the first demo was completed. This was again frustrating for the supervisors.

The simplest means found of addressing these problems was for the two assessors to coordinate their actions verbally. However, with many students milling around the noise levels often made this difficult. The difficulties created by the presence of two assessors was highlighted when one of the assessors left to attend other duties – the remainder of the session progressed smoothly.

The question now is how these sessions can be improved so that the do not take excessive amounts of time, yet give the students the opportunity to demonstrate their work. We assume in the general case the need for multiple assessors. One option, of having multiple versions of the online robot system, defeats the purpose of the online robot concept. Our current proposal to address these problems is for the students to include a "record of experimental results" with the work they submit. In general, the students should submit the following four items for assessment:

- An executable, for assessing basic functionality.
- An animated gif or movie of experimental sequences.
- A report presenting key design features.
- The source code, for assessing programming style.

The students are currently required to hand in only the last two of these. For local assessment the demo sessions can still be helpful, but can focus on basic functionality. For remote assessment the student could deposit the deliverables with the assessor as a zip file, via email or a browser-based upload facility. The assessor can run the executable to assess basic functionality of the system, and review the other materials for overall assessment of the work. We propose to explore these procedures with a new group of students since fluent and efficient assessment procedures are an important element of making online robot systems an integral component of educational environments.

VI. Summary and Conclusions

In this paper we have presented a new online robot environment for a vision-based student project. The environment builds on our previous experience with similar hardware and software environments, and student experience and feedback from previous projects. The new environment provides better control of a toy digger and better support for grabbing images and remote viewing. Student feedback has been positive, supporting the improvements in the environment. There is still considerable scope for further development as new issues arise, including mechanisms for monitoring and assessing student work. These are forming the basis for further development of the environment and its tighter integration with pedagogical goals.

References

[1] K. Goldberg, The Robot in the Garden, *The MIT Press*, 2001.

[2] K. Goldberg and R. Seigwart, Beyond Webcams: An Introduction to Online Robots, *The MIT Press*, 2002.

[3] 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.

[4] G. McKee and B. Brooks, "Interactive Robotics Using the Internet," *Journal of Computer Science Education*, Vol. 7, No. 2, pp. 279-290, 1996.

[5] 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.

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

[7] G T McKee, Multimedia Technology for Online Robot Projects. IEEE ICRA Workshop on Educational Applications of Online Robots, Washington, 2002.