An Experiment in Internet-Based, Human-Assisted Robotics

Lung Ngai, Wyatt S. Newman (wsn@po.cwru.edu), Vincenzo Liberatore (vxl11@po.cwru.edu)
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio 44106

Abstract: *This paper describes an experimental exploration in Internet-based control of robots. The motivation of this work is that Internet communications can be exploited to achieve greater productivity from machines with local intelligence. Local intelligent systems contact human experts to solicit advice when a problem facing the machine is beyond its cognitive capabilities. This topic is explored in the context of a robot performing a sample domestic task (sorting laundry). An experimental system was constructed that has limited autonomous competence, but which proved to be significantly more productive through the use of occasional Internet-based human supervision.*

1. Introduction: The Internet creates many technological opportunities, one of which is the ability to use a standard network infrastructure to control robots from a remote location [1-6]. Internet-controlled robots will have valuable applications in automation and, more futuristically, in space and terrestrial exploration and in home robotics. Much research has focused on using Internet connectivity to control robots [7-9], and Internet robotics can be regarded as the natural extension of research in remote control. Previous work in Internet robotics demonstrates that:

1. It is possible to control robots over the Internet. However, this technology is not yet mature due to *long and unpredictable delays*, especially on heavily loaded IP networks that lack any provisioning for Quality-of-Service [7].
2. There is some evidence that it is possible to implement remote *non-real-time* robot control [8].

This paper offers an exploration of the potential for such future applications of Internet-based robotics. In this approach, local intelligent systems contact human experts to solicit advice when a problem facing the machine is beyond its cognitive capabilities. Simple tasks may be performed autonomously. For example, it is easy to write a program that directs a robot to go to point A, pick up an item, go to point B, and drop the item. If the location of the item were always the same, the robot would continue to repeat the task indefinitely and would never get tired of doing the same procedure. A more complicated scenario would be more difficult to automate. For example, if the system is required to perform different actions depending on the item color, the robot will start making mistakes. The developer will also need to develop and trouble-shoot the vision analysis code. While intelligent systems are getting more sophisticated, achieving adequate competence for autonomous operation in complex environments is not yet feasible. However, partial solutions may still be usable if augmented with occasional human assistance. With human-assistant capability, a robot may ask for help when it is confused. The idea is to allow the robot to complete the system sub-procedures on its own and ask for human help to overcome the more difficult ones. The approach has clear applications to automation. For example, an automated manufacturing plant may be relatively predictable and robustly controlled, but occasional diagnostics, corrections, and resets may be required. These operations might be performed via the Internet, possibly collaboratively, by expert supervisors located anywhere in the world. Furthermore, the approach can enable innovative applications, such as lumbering, mining, space exploration, and domestic service operations through Internet-based human supervision.

In this paper, a specific experimental system is described, including local low-level controls, Web server front-ends, and a human/machine interface. An abstraction of a household task—sorting laundry—is used to illustrate and evaluate the prospective value of Internet-based robots, the effectiveness and needs of a browser-based interface, and the economic opportunities presented through the use of supervisory control.

The paper is organized as follows. In section 2, we review two early demonstrations of Internet robotics that are particularly relevant for this paper. In section 3, we describe the physical and software organization of the system. In section 4, we discuss the interaction between a robot and a remote supervisor. In section 5, we evaluate our approach on a simple household task, and section 6 summarizes conclusions.

2. Earlier Demonstrations: In 1997, Tarn demonstrated network capability with a joystick in front of a conference audience in Albuquerque controlling a robot located in St. Louis. The robot's movement was captured from a remote camera and projected on a screen in front of the audience in real-time. However, the robot's movements were noticeably delayed. The robot's remote control capability could not be activated during the Internet's peak hours because poor Internet response time causes delays on the robot's servo. Hence, the demonstration was performed during the morning, when the Internet was less busy. This experiment demonstrated that controlling robots over the Internet was possible. However, due communication delays and jitter, this technology was not practical.

Another early demonstration of network-based robot control was at the University of Southern California in the Mercury project [8]. Users could change a robot's position over a bed of sand by interfacing via a browser. An air jet was attached to the robot's gripper. The user directed the robot to puff air at the sand to discover what lay underneath. A picture of the robot was provided on the browser but it was not updated in real-time. Pictures were captured periodically by a camera, which was attached to the robot. The user would get an update of the robot's movement periodically. This demonstration showed that a robot could be controlled as a non-real-time system. The robot gathered all the information about the next movement from the user before it began to move. The Mercury project later developed into Telegarden [9]. In Telegarden, each registered user is responsible for his/her plant. He or she logs in every day to water the plant through the robot's water hose that is attached to the gripper. However, remote operation did not reduce human effort.

While these projects do not appear to have clear applications beyond their experimental and entertainment value, they nonetheless demonstrated the capability for network-based robot control and revealed some of the challenges.

3. System Overview: At CWRU, we constructed a robotic test facility for evaluating the prospect of internet-based supervisory control of semi-autonomous systems. Our system consisted of a robotic arm, two cameras, a PC controller, and a Web server.

3.1 Robot & controller: The robot used in this experiment was a low-cost educational robot that had been retrofit for open-architecture control. While this robot had limited workspace, payload, speed, and precision, this choice was attractive in terms of safety, which is a significant consideration in remote control. The robot had five degrees of freedom in a serial kinematic chain, similar to popular industrial designs (see Fig 1 and 2). The robot was interfaced at the torque level to an analog output board within a PC control computer. Incremental encoders on the joints were interfaced to encoder counters within the I/O card hosted by the PC. The PC had a Pentium 133 processor, which was relatively slow, but adequate for this investigation.

3.2 Actuators & Sensors: The actuators consisted of the joint motors and the robot's gripper. The sensors included motor encoders, an optical distance sensor, a gripper-mounted black-and-white camera, and a color Web camera.

3.3 Operating System: The control software on the PC ran within the environment of the QNX operating system, a real-time operating system derived from Unix and commonly employed on x86 machines [10]. QNX supports real-time multi-tasking; i.e., multiple processes can be prioritized and scheduled to run independently, emulating parallel execution. Multi-tasking was employed in our system to run concurrently several control and communication processes. Communications among the separate processes is coordinated through semaphores [14]. Semaphores were used to drive processes at fixed rates and as a mutual exclusion mechanism. We used an existing priority-driven real-time infrastructure [12,13] that divided processes into low and high levels. The low-level processes were simple but required execution at high frequency and at high priority. In contrast, the high-level processes required more computation but did not require frequent execution. The controller software ran on the QNX PC, was written in C, and was compiled with the Watcom compiler [11] that is standard in QNX 4.

3.4 Web Server: The experimental system used an Apache Web server [15] to run as the front-end for user interaction. In turn, the Web server communicated with the QNX controller and relayed user commands to the robot. There is no version of the Apache server for the QNX operating system, and so we installed the Web server on a separate PC running Windows NT. The NT machine connected to the robot controller through Ethernet. In general, the Web server and the robot controller could be installed on the same or on different computers. A single-machine installation is characterized by reduced hardware requirements and by fast communication between the Web server and the controller. The separation of front- and back-end hosts can result in legacy with existing platforms and can lead to higher system scalability and security.

3.5 CGI (Common Gateway Interface) scripts: The Web server initiated robot operation by invoking a CGI script that ran at the Web server side (see figure 3). In turn, the script embedded TCP connectivity to relay instructions to the robot. Furthermore, the scripts gathered feedback from the robot and passed it on to the user. As a result, CGI scripts provided a module to communicate and pass information back and forth between the server and the controller. CGI scripts were exposed to remote users by creating a form within an HTML document and inserting the script URI as the form action.

CGI is an acronym that stands for "Common Gateway Interface" and refers to the protocol used to pass arguments from the Web server to the script. Internally, the script could be written in any programming language but, in practice, CGI scripts are typically written in Perl. The Apache server had a handler for running CGI scripts as threads that are part of the Web server process: when the server invoked a CGI script, a new thread was created within the Web server process to execute the script. An earlier version of the system did not use such a module and was running CGI scripts as separate processes. The robot reacted very slowly to users' commands due to the overhead for forking additional processes to execute the scripts. When CGI scripts were wrapped in the same server process, the robot reaction time improved dramatically [16].

3.6 Vision Analysis: We used a Matrox card as the image frame grabber. The Matrox card has a driver for Windows NT but no driver for QNX. Therefore, the Matrox card was installed on the same NT machine that hosted the Web server. The NT machine also performed vision analysis. Thus, images from the frame grabber were sent to two processes: a CGI script within the server process to forward the images to the end user and a C process that analyzed the images (see figure 4).

The Matrox card was connected to a gray-scale camera, which was mounted on the robot's gripper. The camera and frame grabber provided 512 x 480 frame with pixel intensities in the 0 to 255 range. The driver was written using Matrox's Mil-Lite Version 6.1 C library [17]. The C program for image processing was developed using Microsoft Visual Studio.

3.7 User Interface (UI) presentation: The Web interface was written in HTML. The interface had feedback buttons for a user to transmit commands to the Web. The commands would be relayed to the robot controller for execution. Radio buttons, drop down menu, and normal buttons were used. The radio buttons forced the user to select from a list of valid choices so that the return string would not be an empty string. The drop down menu stored choices for the users (see figure 5).

Image-based servo control was accomplished through the actual image from the camera. The frame grabber took a snapshot of the robot's view and the Web server encoded the saved image as part of the HTML display. The image was a black and white JPEG picture and it projected the robot's point of view (see figure 6). Such pictures were used as an input means. The user could click on the actual picture on the Internet browser, and this action would invoke recording the actual pixel X and Y coordinates. These coordinates were then delivered to the Web server and a CGI script relayed the data from the web server to the robot controller. Image-space coordinates were translated into robot-space coordinates, transparent to the operator, to command the robot to move to the selected location.

3.8 Web Cam: The buttons and the servo control were the input elements; however the user also needed feedback from the system. There were two forms of feedback: images from the robot's viewpoint (with the gripper-mounted camera) and a wider-angle side view from a Web camera (a *WebCam*). The first camera was mounted on the robot's arm and its frame was refreshed only after the robot arm completed a movement. In practice, the frame was refreshed at the behest of a CGI script spawned by the server when the arm movement completed. Since the picture was delivered only at the end of the arm movement, the user would see a "busy" icon during the movement. A second camera was mounted near the robot with a side view of the robot and its workspace and provided a real-time view of the robot environment. The camera used the Webcam32 surveyor software [18] that acted as a real-time streaming video server to capture and deliver the images to a remote user in real time (see figure 7). Webcam32 was a separate server that worked independently from the Apache server, but ran on the same NT machine as the Web server. To display the video stream, the client used a Java applet [18], i.e., a Java bytecode executable that was dynamically downloaded into a browser over the Internet. The applet continuously downloaded video streams from Webcam32, displaying the video at a rate of about 1 frame per second.

3.9 Client site: The human supervisor interacted exclusively with the Apache Web server and with the Webcam32 video server. In practice, the operator could use any of the commercially available Web browsers to supervise the robot and download video streams. When the user clicked on a button or selected an option from a drop-down menu, the user choice was sent to the Web server in a standard HTTP request. The server would then trigger a CGI script that communicated with the robot controller through a TCP connection. Meanwhile, the Webcam32 server would continuously push video frames to the Java applet running on a Java Virtual Machine within the browser, presenting the video feed to the user. The client site used widespread off-the-shelf components, such as Java-enabled Web browsers, so that the human interface to the robot required no specialized software or hardware beyond what is already commonly available. Furthermore, the use of standard components shifted the software design process from the internals to the interface between humans and machines.

4. Human / Machine interaction: Internet-based control makes it possible for a remote user to direct robot operations from a remote location. However, network connectivity is inherently subject to time delays, which constrains an interface designer to limit the communication demands between the two hosts. Direct teleoperation, as described in [7], requires low latency to be safe and effective. In contrast, if most of the control communication is in the form of low-level commands, such as unidirectional movements, these can be interpreted and expanded into real-time execution with only low-level intelligence local to the robot. If robots can handle simple local tasks as well as basic survival skills, such as obstacle avoidance, the latency requirements would be greatly reduced. The principle of supervisory control motivates our adoption of coarse-grained interfaces such as drop down menus and point and click interaction.

As for control feedback, it should provide reliable real-time information about the robot's motion to a remote supervisor, and motivates our use of a streaming video server at the robot site and of the corresponding client applet at the user site.

5. Evaluation: The following experimental evaluation illustrates the potential for network-based supervisory control of semi-autonomous systems. The experiment simulated a laundry-sorting task, which is mundane for humans, but complicated for robots. Consequently, this task is an instance in which human assistance could make a robot productive. In the experiment, there were three baskets with three types of washcloths: bleachable, non-bleachable, and ambiguous. The robot was pre-programmed to pick-up the washcloths from a source basket and then use its cameras and visual analysis program to determine the color of the washcloth. The robot would then drop the washcloth into either the bleachable basket or the non-bleachable basket, depending on the analysis result. A pick-and-drop sequence was considered a "task". The entire sorting procedure was a group of tasks. If the robot had a problem determining the color of a washcloth, it paused and asked for help from the user, who communicated with the robot through an Internet browser. The user could assist the robot by using buttons from the Web page. The human advisor looked at the picture that was shown on the browser and clicked on the corresponding radio button on the panel to provide guidance. The robot then completed the task according to the human interaction then continued with the remaining tasks on its own.

The experiment employed a set of five washcloths, which included four bleachable washcloths and one non-bleachable washcloth, which were randomly placed in a basket. Every two sets of washcloths contained a yellow washcloth whose black-and-white camera representation could be mistaken for a bleachable cloth. There were total of 20 sets for the robot to sort. Autonomous operation resulted in 83% correctness. There were a total of 4% gripper mistakes, in which the robot gripped more than one washcloth at a time. The remaining 13% of mistaken sets was due to the incorrect vision analysis (see figure 8).

The system made some mistakes on the yellow washcloths, which were incorrectly classified as bleachable. The experiment was then modified to include the capability for network-based human assistance. When the robot had trouble recognizing the color of a cloth, it would pause and take a snapshot of the cloth. The picture of the cloth would be posted on the Web. After the remote supervisor advised the robot, operations continued with the remaining tasks. The robot needed help about 20% of the time (see figure 9) and the accuracy increased to 92%. Subsequently, we refined the gripper control and installed a color camera, which improved autonomous performance to 98% consistency. Combined with human assistance, the robot made no errors while asking for human assistance only 5% of the time.

In the experiment above, the "human assistance" system helped the robot to be more effective when performing difficult tasks. More generally, it is not practical to build a competent, autonomous system that performs flawlessly in complex situations. But if a system has some degree of competence, can recognize situations beyond its competence, and can contact a remote supervisor for assistance when necessary, then it can be substantially more valuable than in imperfect fully autonomous system. In our demonstration, we focused on building a robot with a high percentage of consistency and a good human assistance system.

6. Discussion and Conclusions: In this paper, a system was designed to experiment with the concept of human-assisted internet-controlled robotics. The laundry-sorting example yields encouraging results, motivating further work with this conceptual approach. Human assistance systems offer a new concept to automation. Building a fully automated, reliable system can be infeasible or cost-prohibitive for complex tasks using current technology. Human assistance can lower development cost significantly, and the resulting system can begin to be operational and productive sooner than a fully autonomous system. With human assistance, emerging technologies could be utilized more rapidly, since absolute dependability is not required. Thus, with human assistance, imperfect technology could be productively deployed in demanding scenarios. Immediate applications would include remote exploration, remote experimentation, hazardous environment operations and defense applications. More futuristically, this approach could help lower the cost of achieving adequate competence in domestic robots.

Besides lowering the cost of building a system, our demonstration has illustrated that human assistance can improve the consistency of a procedure. From the laundry-sorting task example, the system performed the operation error free with the help of human supervision.

Current and future developments are inspired by the need to add reliability and consistency to complex robotic systems. To this end, we are implementing a sophisticated software platform that is based on mobile intelligent agents for large-scale software development and on local controls and low-level intelligence for performing force-controlled contact operations.

Acknowledgments. We would like to thank Bob Barmish, Mike Branicky, Maria Dikshteyn, Ken Loparo, Steve Phillips, Deepak Rao, and David Rosas for helpful discussions. This work has been supported in part under NASA grant NAG3-2578.

References:

1. Sayers Craig, Richard Paul, Dana Yoerger, Louis Whitcomb, and Josko Catipovic (WHOI). "Subsea Teleprogramming", http://www.cis.upenn.edu/~sayers/tele/subsea.html May 1995
2. Sayers Craig, Augela Lai and Richard Paul. "Visual Imagery for Subsea Teleprogramming", *IEEE Robotics and Automation Conference*, May, 1995.
3. Sayers Craig and Richard Paul. "An operator interface for teleprogramming employing synthetic fixtures", *Presence*, Vol. 3, No. 4, 1994
4. *Scalable Coordination of Wireless Robots* ftp://deckard.usc.edu/pub/arena
5. Card S. K., Moran T. P., Newell A. "The psychology of human-computer interaction". Hillsdale, NJ, Lawrence Erlbaum Associates. (1983).
6. Taylor K. and Dalton B., "Internet Robots: A New Robotics Niche", *IEEE Robotics and Automation Magazine*, 2000, pages 27-34
7. Tarn T. J. "Live Remote Control of a Robot via the Internet", *IEEE Robotics & Automation Magazine*, pages 7-8, September 1999
8. Goldberg Ken, et. al., "Desktop Teleoperation via the World Wide web". *IEEE Int. Conf. On Robotics and Automation*, 1995, pages 654-659.
9. Goldberg Ken and Joseph Santarromana (UC Irvine). "The Telegarden" http://www.usc.edu/dept/garden/ , 1995
10. *Photon 1.1 Widget Reference*, QNX Software Systems, Ltd. (1998)
11. *Watcom C (v10.6) Library*, QNX Software Systems, Ltd. (1998)
12. Krten, Rob, "A Guide for Realtime Programmers", Parse, 1998.
13. Morris, Daniel M., "Experiments in Mechanical Assembly Using a Novel Parallel Manipulator", M.S. thesis, EECS Dept., on Robotics and Automation, 1999.
14. *Photon 1.1 Programmer's Guide*, QNX Software Systems, Ltd. (1998)
15. *Apache web Server* http://www.apache.org
16. *Mod_Perl*, http://perl.apache.org
17. Matrox Imaging Library, version 6.1 http://www.matrox.com/imaging/prod/mil-lite/home.htm
18. *Surveyor (web Cam software)* http://www.surveyor.com

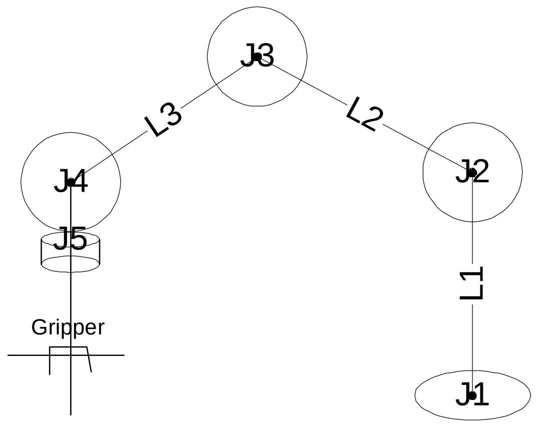


Figure 1. Robot Kinematics.

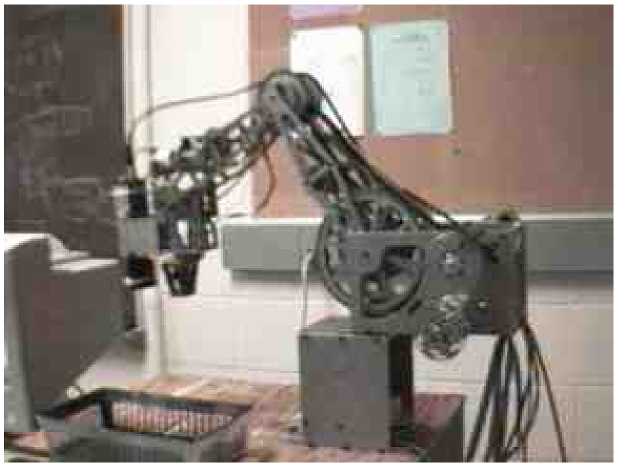


Figure 2. Robot's side view.

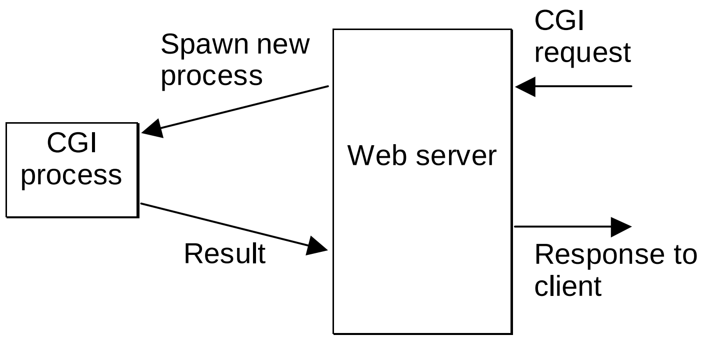


Figure 3. Perl CGI Process.

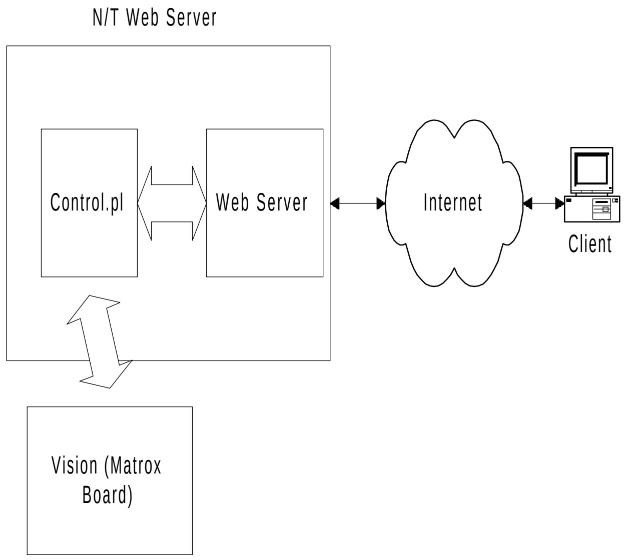
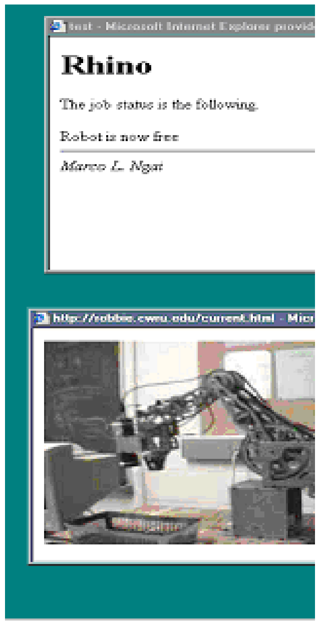


Figure 4. Client-server architecture.


test - Microsoft Internet Explorer provid
http://robbie.cwru.edu/current.html - Micr

Rhino

The job status is the following.

Robot is now free

Marco L. Ngai

Figure 7. Picture of the robot’s view control.

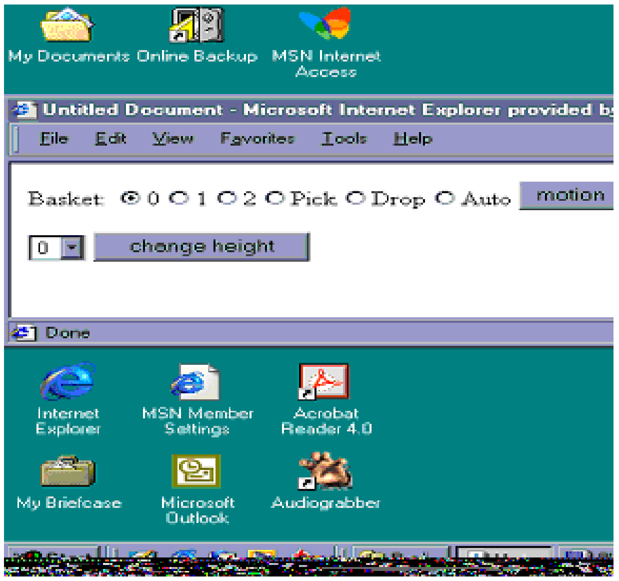


Figure 5. Picture of the control interface.

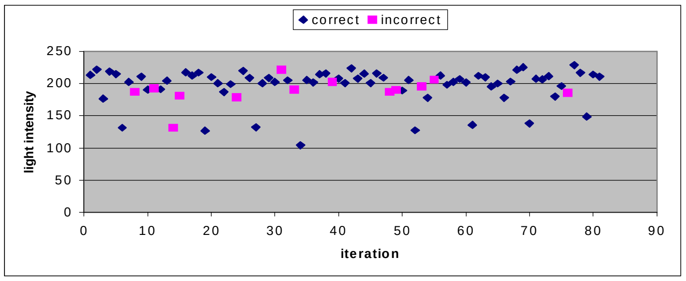


Figure 8. Robot’s performance result.

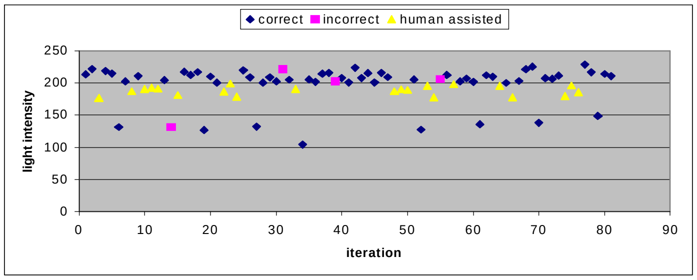


Figure 9. Effect of human assistance.

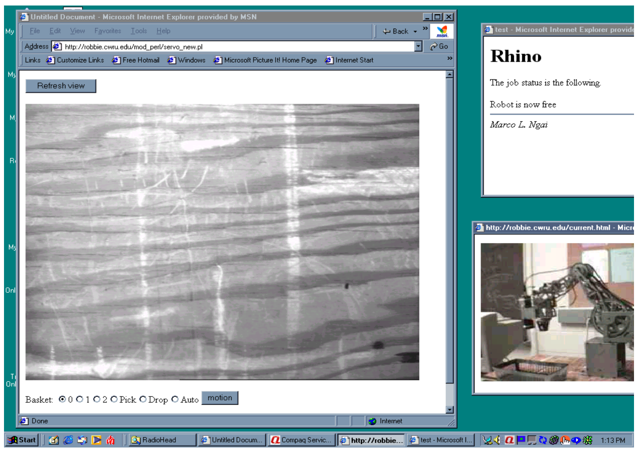

Untitled Document - Microsoft Internet Explorer provided by MSN
Address http://robbie.cwru.edu/mod_perl/servo_new.pl
Refresh view
Basket: ◉ 0 ○ 1 ○ 2 ○ Pick ○ Drop ○ Auto
motion
Done
test - Microsoft Internet Explorer provide
Rhino
The job status is the following.
Robot is now free
Marco L. Ngai
http://robbie.cwru.edu/current.html - Micr
Start
1:13 PM

Figure 6. Picture of the robot’s view control.