Towards Universal Access to Robotic Resources*

Jian L. Zhen jlz@cs.ucla.edu
M. Anthony Lewis tlewis@cs.ucla.edu
Kar-Han Tan tankh@herd.cs.ucla.edu

The Commotion Lab
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095-1596

Abstract- *This paper describes steps toward making a collection of robots easily accessible to remote users. The current approach is to standardize the interface to an existing robot, provide connectivity to the Internet and provide a Universal Interface to remote users. We leverage existing system software and operating systems to create an easily programmable, flexible system. By providing the system with certain tools, the experimenter is able to automatically create a record of the experiment for future analysis.*

1.0 Introduction

In this paper we present a system for controlling groups of robots through a World Wide Web interface. By leveraging the existing technology infrastructure, we show that it is feasible to control robots at a remote location using commonly available user interfaces afforded through the WEB. This approach allows the widest possible access to robots at remote locations.

The key benefit is that it may be possible for researchers with no access to robotics resources to have simple and easy access to robotic resources at other labs. This would facilitate a key aspect of the scientific method which is currently lacking in much robotic work: *the systematic reproduction of the results of others.* Under our approach, it should become possible for researchers to open their labs to outside and independent verification of certain results. This paper describes steps toward this long term goal.

Since off-the-shelf robots are becoming more standardized, it should be possible for researchers to share robots as much as people now share workstations. It is our observation that this situation does not exist now. In robotics research labs, robots are typically used exclusively by one or two researchers. Yet even the most dedicated researchers cannot fully utilize a robot or systems of robots for 24 hours each day. In most locations, robots remain idle throughout most of the day. This situation has led to a large number of under-utilized robots.

This situation would be tolerable if robots were inexpensive and each researcher had access to as many types of robots as needed for research. Unfortunately, this is not the case. Robots are expensive and few roboticists have sufficient resources to maintain a schedule of experimentation which keeps pace with theoretical developments.

This ought to change. Laboratories with the fortune of owning valuable hardware should make it a point to share what they have; expensive robotics resources should be made available to other interested researchers.

We call this philosophy *Universal Access*. We advocate the sharing of robotics resources between experimenters.

In designing our proposal for a move towards universal access, we identified the following problems in the current situation:

- *Limited public availability of existing robots* - most existing robots are typically not available to researchers outside the laboratories. That is, robots are usually not wired to a world wide research network, and researchers have no means of accessing the robots unless they are physically in the laboratories.
- *A lack of standardized experimental facilities* - to perform scientific experiments good monitoring and metrology equipment are needed. For example, in a motion control experiment the trail followed by the robot needs to be recorded accurately and returned in a form easily usable by scientific visualization software packages. In addition, it would be valuable to have a coordinated video and data record.
- *Unfamiliar development environments* - researchers currently have to adapt to development environments provided by robot manufacturers, which are often unfamiliar and require significant amount of time to master.

Incorporating ideas for the removal of these problems, we developed a robotic testbed called the W3R3 system. We give a brief overview of the system here.

* This research was supported by NSF CDA-9303148 and matching funds from the UCLA School of Engineering and Applied Science.

Proc. IROS 96
0-7803-3213-X/96/ $ 5.00 © 1996 IEEE

The basic robot we started with was the R3 from IS Robotics. This robot has a wide range of sensing capability as well as the ability to grip objects. We felt that this robot was representative of the types of robots which other researchers might wish to have at their disposal.

The R3s were augmented with Linux PCs. This augmentation gave the robot the ability to be connected directly to the Internet. Next we developed a vision based metrology system to track the robots as they moved. This system provides the ability to monitor the progress of an experiment.

In addition, we built a Web interface which allows easy remote access to our site from virtually any platforms.

Finally, we have begun to augment the W3R3 with a status and safety monitoring system which assists in the maintenance of the robots.

In this paper we describe in detail the architecture of the system.

2.0 Previous Work

There is a growing number of Web accessible robots. One of the most popular sites has been the Mercury project and the Robotic Tele-garden project [7,8] at USC. In this environment the researchers where able to allow virtually unrestricted access to robotic manipulators capable of fixed tasks.

The manipulators where interfaced to the Internet and access to them was gained through a Web interface.

The Mercury project consists of a robot manipulator moving above a section of earth known to contain artifacts. There is a downward-pointing camera attached to the distal end of the robot arm. By clicking on an image or buttons on the Web page, users can make the robot move along the X-Y axes and the camera along the Z axis. In real-time operation, GIF images of the camera view are updated on the Web page, and a session is logged as an MPEG movie; the only sensory data is the energy level. This project has been decommissioned since March 31, 1995.

The Australia's Telerobot On The Web project at the University of Western Australia [9] challenges users to control the parallel-jaw grippers of a six-axis manipulator to pick up blocks on a table. Through a fill-out form, users can submit desired (X,Y,Z) coordinates, and roll, pitch, yaw actions. A fixed camera gives a side view of the arena, and the image is updated when the user command is submitted and executed.

The Bradford Robotic Telescope [10] allows users to work either in batch mode or in real-time mode. The user describes what he wishes to observe and the telescope moves accordingly. One of the strong features of this project is the good implementation of user access control: users must register, login (passwords can be changed via e-mail), and submit jobs. Furthermore, jobs are prioritized; only those with the highest priority are allowed real-time control of the telescope.

These projects are precursors to the work presented here. In these works, the researchers have begun to address the question of connectivity and reliability.

A key element missing is the ability to program the robots. Allowing programmability incurs certain costs however. The number of people with access to the Web is far greater than the number of people who can program a robot productively. To allow programmability, it is necessary to restrict access to trusted users. This is the approach we have taken.

3.0 Approach

Our approach is to increase connectivity, offer support for experimental development and execution.

3.1 Increasing Connectivity

As mentioned earlier the R3 is used as the robot platform. The processors on the R3 robots run the Venus operating system, which does not support network connectivity. In addition, programming the R3 requires downloading code from a host computer via a specialized serial interface. These two factors limit the connectivity and programmability for remote experimentation, since remote users cannot download control programs onto the robots using standard network applications.

A solution to this was found by adding a Linux notebook computer. This provides an interface between the robot and the rest of the world, and lays a solid development foundation.

For example, we were then able to add a Web interface to the system, which provides programmability with Unix shell scripts. Remote users can thus program the robots and get near real-time feedback from overhead video cameras mounted over the R3s' work area.

3.2 Experimental development and Execution Support.

Several tools were developed to assist in development. First a 3-D simulator was developed. At this point, the simulator runs on platforms supporting Open Inventor. Using this simulator, investigators can begin to debug their algorithms before moving onto real hardware.

Secondly, an Applications Programming Interface (API) was developed which allows researchers to write programs that control robot actuators and receive feedback from the robots sensors. Although programming

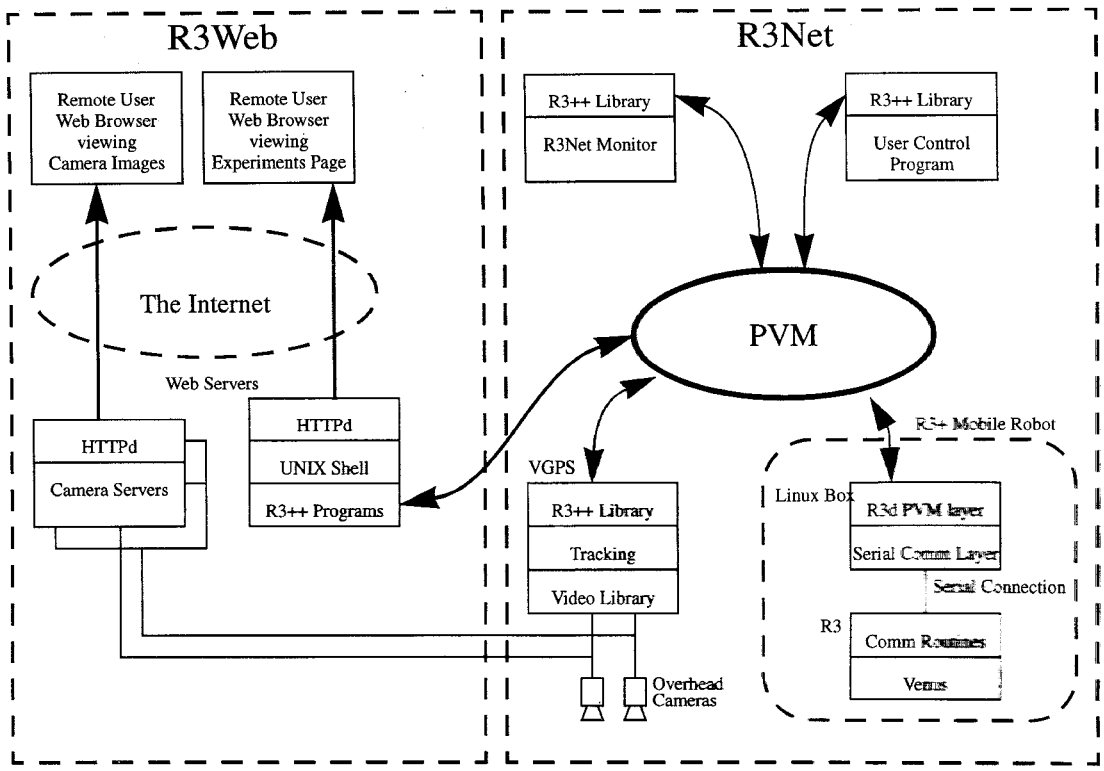

Figure 1. W3R3 software architecture overview.

typically involves some communication between different computers, the underlying networking is transparent to the user. This is achieved by building our system on top of PVM (Parallel Virtual Machine), which simplifies distributed systems programming with heterogeneous architectures.

Thirdly, a metrology system was constructed. This system tracks each robot with a frequency of 10 hz. Thus global positioning information is available to the robots.

3.3 Embodiment in W3R3

The elements described above were combined in the W3R3 system. W3R3 refers to the acronym for the World Wide Web and the name of the robots 'R3.' The two subsystems in W3R3 include R3Net and R3Web. R3Net consists of the software and hardware needed to implement lower-level connectivity and R3Web refers to the higher-level Web interface. This is shown in figure 1.

4.0 W3R3 Architecture

We now examine in more detail the design of W3R3.

4.1 The Original R3 Platform

The robots used in the W3R3 project are IS Robotics R3 robots. The R3 robots are small, autonomous mobile robot about 30 cm both in diameter and in height. Each R3 carries 68332 and 68HC11 microcontrollers which runs the Venus operating system. User programs are ran in 1 MB of non-volatile RAM.

The R3 robot carries a rich set of sensors including infrared proximity sensors, bump sensors, shaft encoders, and power status indicators. A user input panel is also provided for programs that requires user interaction. The robot moves using a differential drive and can manipulate objects using a force-sensing gripper subsystem. Power is supplied by rechargeable NiCad batteries. The robot has the capability to detect low power conditions and recharge under software control when attached to an external power source.

The R3 runs an operating system called Venus. The runtime software system consists of three pieces: the assembly language kernel, the C function libraries and the L runtime libraries. The assembly language kernel consists of low level routines which interface to interrupts and handle processor reset cycle. The system is currently designed with L as its main programming language. L supports multi-tasking and can call the C rou-

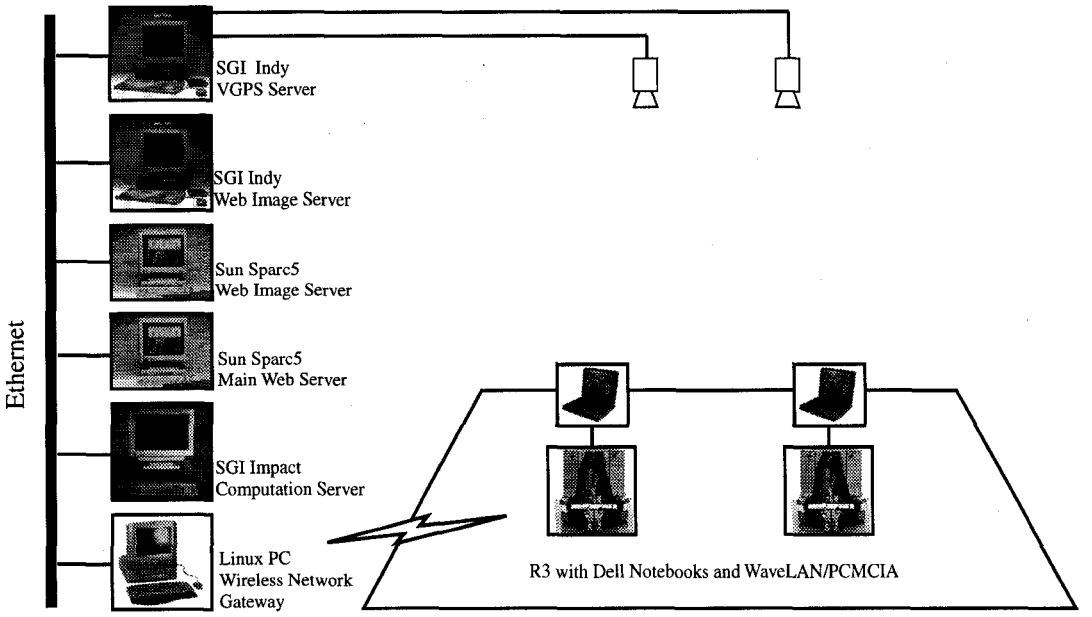

Figure 2. W3R3 Hardware Overview.

tines to manipulate the hardware. Simple processes are implemented outside of L via a round robin scheduler which interrupts 1024 times a second. Thirty two of such processes can be loaded and therefore each process gets executed thirty two times per second.

While the R3 robot provides many of the necessary capabilities for collaborative robotics, it needed several augmentations for our purpose. First of all, it needs an application development environment accessible remotely. With the default setup of the R3, the programmer must write their program on a Macintosh, then download it to the R3's battery-backed RAM using a special serial link. Whenever changes are made to the application, the programmer must repeat this process, which is tedious, time-consuming, and not possible remotely. Second and probably the most important shortcoming is that the original R3 platform needs a higher-bandwidth and more reliable communication mechanism between robots. The original R3 setup uses Proxim radio modems as a wireless serial link between each R3 and its basestation with a data rate of 19.2 Kbps. Not only does the throughput degrades when the number of robot increases, it was also shown to be unreliable in sending and receiving data.

4.2 The Upgraded R3 (R3+)

To overcome the two problems mentioned earlier, we examined several ways of modifying the R3. An alternative we chose was to add TCP/IP on the original R3 since TCP/IP has long been proven to be a fast and reliable communication mechanism. While it is possible to write a TCP/IP stack for almost any platform we took a more expedient path. Each R3 has been augmented with a notebook computer carrying a 486DX4-100 processor, 8 MB of RAM, and 400 MB of mass storage. The notebook computer runs Linux. Although Linux is public domain software, it has proven to be very reliable. In addition, the Linux source code is freely available, making kernel-level modifications possible.

The notebook computer communicates with the host R3 via a RS-232 serial link running at 19.2 Kbps. The host R3 sends out sensor information through the serial link at a rate of 32Hz in correspondence to the Venus scheduler execution rate. The notebook computer is responsible for receiving this information and sending them to user applications when requested. The notebook computer is also responsible for receiving robot commands from user applications and sending it to the host R3 via the serial link.

4.3 Communications

As shown in figure 2, the physical network of the R3Net testbed consists of a wireless network using AT&T/NCR's WaveLAN and an Ethernet LAN. The WaveLAN network runs in the 915MHz range and provides a bandwidth of up to two Mbit/s. This wireless network is actually an IP subnet consisting of 10 R3+ robots and a basestation. Each R3+ carries a WaveLAN/PCMCIA network adaptor, and the basestation, which is also connected to a wired network, carries an ISA bus

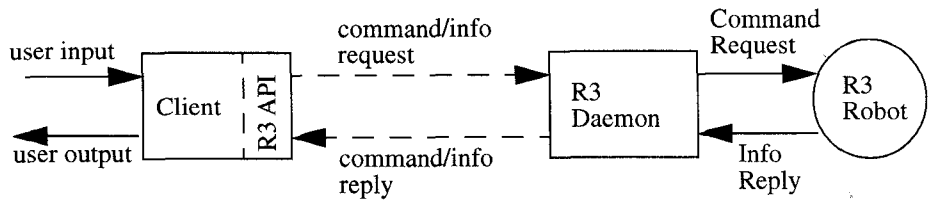

Figure 3: R3Net communications model. The dashed line represents the network communications between the R3 applications and R3 Daemon.

WavePoint adaptor and acts as a gateway to the Internet. The setup with a separate IP subnet not only gives a sense of autonomy, it also reduces unnecessary traffic, which makes communication between robots more efficient.

The logical network of the R3Net includes the R3 wireless network and computers that wishes to communicate with the R3+ robots. This setup is accomplished by another public domain software package called PVM (Parallel Virtual Machine). The PVM system is an integrated set of software tools and libraries that emulates a general-purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture. With PVM, any computer on the Internet can be part of the R3Net logical network. Applications that wishes to communicate with the R3+ robots can be running anywhere inside this logical network.

The communication model of the R3Net is based on the client/server paradigm (shown in figure 3), more specifically, a two-level Remote Procedure Call (RPC) model. RPC uses a request-and-reply communication model. The client sends a request message to the server which returns a reply message. In the R3Net, the "real" server is the Venus operating system running on the host R3. The "real" clients of the R3Net are applications that wishes to communicate with the robots. To facilitate reliable communications and easy application programming, a relay server, r3d, is put in between the "real" clients and servers. R3d acts as a relay station between the clients and the host R3, it receives commands from the clients, sends them to the host R3, then receives replies from the host R3, and return them to the clients. This setup hides all the network communications from the researchers and allow them to focus on their algorithms.

4.4 R3 Application Programming Interface (API)

With the R3s upgraded and the networks in place, a C++ API has been implemented in order to provide researchers a high-level programming interface for implementing their algorithms.

The C++ API implements a R3 object which emulates the real R3 robot. Methods such as LiftGripper and LowerGripper are provided so the researchers can easily control the robots within their application.

When a R3 method is called, the R3 object then collects the necessary information into one message, sends the message to the r3d running on the R3, and wait for a reply. When a reply is received from the r3d, it then returns the status to the user. All the background processing are transparent from the application's point of view. The application only knows that it called a R3 method and the method returned a value.

In order to trap errors such as lost messages, a timeout mechanism has been incorporated into the API. After 100 milliseconds, if no reply has been received, then the R3 methods will return an error.

Figure 4 shows an example R3 program which uses the R3++ API to control the forward movement of the robot.

4.5 R3 Daemon

As mentioned previously, the R3 daemon is a relay station between the "real" clients and servers. It basically relays client requests to the host R3 (the server) and relays server replies to the clients.

The R3 daemon is implemented as a multi-process server: a PVM process and a serial process. The PVM process is responsible for waiting for requests from clients, and the serial process is responsible for constantly updating a R3 object with sensor and actuator information from the host R3. The serial process is also responsible for sending commands to the host R3 when such requests arrive.

In addition to these functions, the R3 daemon is also responsible for controlling the access to the host R3. Since it is unreasonable for two application programs to control the same host R3 at the same time, the R3 daemon must provide a registration mechanism for the client applications. When a R3 object instance is created for a certain robot, it transparently sends the appro-

```
#include <iostream.h>
#include <R3++.h>
#include <stdlib.h>

main(int argc, char **argv)
{
  if (argc != 3) {
    cout << "Usage: " << argv[0]
         << " rid velocity"
         << endl;
    exit(-1);
  }

  int rid = atoi(argv[1]);
  int veloc = atoi(argv[2]);

  // Instantiates an R3 object
  R3 r3(rid, R3_ACCESS_COMMAND);

  // Exit if control to the robot
  // is denied (maybe another app
  // is controlling the robot.)
  if (r3.Errno() ==
      R3_REGISTRATION_DENIED)
      exit(-1);

  // Move the robot forward
  r3.MoveForward(veloc);
}
```

Figure 4: Example R3 program that uses the R3++ API to control the movement of the robots.

priate R3 daemon a registration message. The R3 daemon allows the registration only when no other R3 object instance is currently controlling the robot.

4.6 R3Net Monitor

In order to provide maintenance assistance and in general to monitor the system health, a R3Net Health and Safety Monitor is being developed.

This monitor will watch the R3Net system and update an R3 information database with status of battery, serial link, physical network, host R3, and tracking. When necessary, the monitor will take control of the robot and try to fix the problem. For example, if the battery of a certain robot is getting low, the monitor will guide the robot to a recharging station and turn on its recharging circuitry. When the robot finishes recharging, the monitor will return the robot to its home position.

4.7 Vision-based Global Positioning System (VGPS)

An essential element of the R3Net testbed is the Vision-based Global Positioning System (VGPS). It provides accurate information at approximately10Hz on the position and orientation of each R3 being tracked. Without VGPS it would be very difficulty to implement motion control algorithms such as a global "move to" function. VGPS is also needed for the R3s to find the recharging station. VGPS hardware include CCD wide-angle video cameras that are mounted over the R3 arena, and a VGPS server which digitizes images from the cameras and performs the tracking computation.

Briefly, the system works as follows: 1) Based on previous measurements, the predictive stage of a Linear Kalman Filter estimates a new position of the robots 2) Based on this prediction, regions of interests are established at the estimated position of the target features on the robots 3) Small regions of interest, encompassing an individual robot's targets are acquired, and segmented. The position of each target in the image plane is established. 4) A model of the robot and its target is matched to the acquired target positions. This establishes a measured position of the robots 5) The measured position of the robots and previous estimates of the robot's position are combined using the estimation stage of a Linear Kalman filter 6) The procedure is repeated again from step 1.

The use of this kind of active vision approach significantly reduces the amount of computation needed to track the robots. By scanning only certain small regions of interest, a substantial improvement in tracking performance is realized.

Since the robots are tracked in global coordinates, it is relatively easy to use multiple cameras to track the robots. When a robot vanishes from one cameras field of view, it will be picked up by another camera. By using a model of the camera, and the predicted position of the camera in world coordinates, it is easy to predict which camera is able to 'see' the robot at any particular time.

While VGPS is running it maintains a database of robot positions. At constant intervals it sends information using the R3 API to each individual R3 Daemon, which in turn updates their local database. When users make R3 API calls to query the position information, the corresponding R3 Daemon replies with the most current position. In this arrangement, VGPS is also transparent to the user applications.

4.8 R3Web

In order to provide a uniform interface to all users, we implemented a graphical user interface (GUI) based on the current available World-Wide-Web (WWW) facilities, e.g. WWW servers such as httpd, and browsers such as Netscape. The main reason to implement our GUI using the WWW is because WWW facilities are widely available and therefore maximize the accessibil-

```
#!/bin/sh

MoveTo 1 5 -0.5 -1.6
SetMotor 1 3 230
sleep 1s
MoveTo 1 5 -1.5 -1.6
StopWheels 1
```

Figure 5: Example shell script that moves the robot the recharging station.

ity to our systems. Also, with the recent introduction of the Java language, we will be able to provide all the necessary features that a standard GUI package, e.g. Motif, can provide.

The R3Web setup consists of a main Web server and two camera servers. The main server is responsible for providing information about our W3R3 project, QuickTime movies of our demos, and script submission. The two camera servers are responsible for serving images from the two overhead cameras to the users continuously.

Our WWW-based GUI currently provides the following capabilities: (i) QuickTime movies of various demos, (ii) submission of simple shell scripts with commands such as Forward, LiftGripper, (iii) near-real-time view of the R3 arena on the web page (figure 7).

Robot controlling is currently supported in three levels: level 0, level 1, and level 2. Level 0 support allows users to view QuickTime movies from our web page and control a single robot by selecting commands from our experiments web page. Level 1 and Level 2 supports are for subscribed users only. Level 1 support allows users to submit scripts via the web to control the robots. The user will also get instant feedback of the script they sent. Figure 5 shows an example shell script that could be sent via the web interface. With level 2 support, an account is created for the users on our system. They will be able to login and write R3 applications using the R3 API.

5.0 Discussion and Future Work

In the above sections we detailed how we have begun to facilitate universal access to a robotics lab. If this paradigm sees widespread usage, the robotics community should realize a greater efficiency in the use of its resources, and hopefully a more rapid advance toward its collective goal.

The next step in our implementation will involve making the R3Net monitor system operational. This will allow rapid debugging of hardware faults in the system.

The final tool which will be developed is automatic logging of experimental data. This would include creation of a compressed MPEG (or equivalent) movie from multiple cameras and synchronization of sensor data and out-going commands.

Figure 7. View of the arena from the web interface.

Acknowledgments

The authors wish to thank all of the individuals, past and present, who have contributed significantly to this project. These include: Agueda Simo (Art Director and Web Page design and construction), Martin Harris (original R3Net Design), Andre Stechert (Systems Issues and Programming), Dan Liebgold (VGPS implementation), Loren McQuade (Web page-UNIX interface), Alex Fukunaga and Forrest Shaaf (experimentation with early versions) Tsuwei Chen (Networking of wireless communication) and Aditya Damle and Jim Park for web and programming support.

In addition, we thank Andrew B. Kahng for many useful discussions of this project.

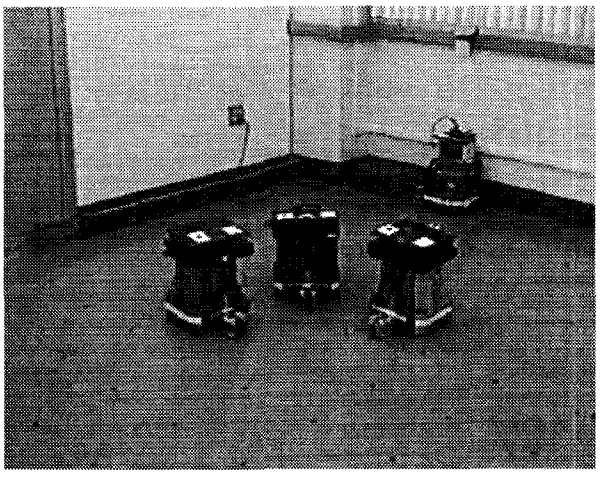

Figure 6. View of the R3 arena room the ground. The two dots on top of each R3 are tracking targets used by VGPS.

Finally, we thank Chuck Rosenberg of IS Robotics for his help in creating time-slicer software on the R3 Robots.

References

[1] T. J. Berners-Lee, R. T. Fielding, and H. F. Nielsen, "HyperText Transfer Protocol - http1.0", *Internet Draft*, March 1995, http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-v10-spec-00.txt.

[2] T. J. Berners-Lee, R. Cailliau, J. F. Groff, and B. Pollermann, "World-Wide-Web: The Information Universe", *Electronic Networking, Research, Applications and Policy*, No. 2, 1992, pp. 52-58.

[3] T. J. Berners-Lee and R. Cailliau, "World-Wide-Web: Proposal for a Hypertext Project", 1990, http://info.cern.ch/hypertext/WWW/Proposal.html.

[4] T. J. Berners-Lee and D. Connolly, "HyperText Markup Language Specification - 2.0", *Internet Draft*, Feb. 1995.

[5] J. Borenstein and Y. Koren, "Teleautonomous Guidance for Mobile Robots", *IEEE Trans. on Systems, Man and Cybernetics*, 20(6), Nov.-Dec. 1990, pp. 1437-1443.

[6] M. J. Cox and J. E. F. Baruch, "Robotic Telescopes: An Interactive Exhibit on the World-Wide-Web", *Proc. 2nd International World-Wide-Web Conference*, Oct. 1994.

[7] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop Teleoperation via the World-Wide-Web", *Proc, IEEE International Conference on Robotics and Automation*, 1995.

[8] http://cwis.usc.edu:80/dept/raiders.

[9] http://telerobot.mech.uwa.edu.au.

[10] http://wwwieia.brad.ac.uk/rti.

[11] IS Robotics, *Venus Reference Manual*, 1995.

[12] IS Robotics, The R-3(tm) Robot Manual, Nov. 1993.

[13] I. Tou, S. Berson, G. Estrin, Y. Eterovic, and E. Wu, "Strong Sharing and Prototyping Group Applications", *IEEE Computer*, 27(5), May 1994, pp. 48-56.

[14] Y. U. Cao, et al. "A Remote Robotics Laboratory on the Internet", *Proc, International Networking Conference*, 1995, pp. 65-72.

[15] AT&T, WaveLAN/PCMCIA User's Guide, 1994.

[16] Al Geist, et al. "PVM: Parallel Virtual Machine", The MIT Press, 1994.

[17] J. Bloomer, "Power Programming with RPC", O'Reilly & Associates, Inc., 1992.