An Internet Accessible Telepresence

A.E. Kaplan, S. Keshav, N.L. Schryer, J. H. Venutolo

AT&T Bell Laboratories, Murray Hill, N.J.
{aek keshav nls jhv}@research.att.com

ABSTRACT

The US Vice President, Al Gore, in a speech on the information superhighway, suggested that it could be used to remotely control a nuclear reactor. We don't have enough confidence in computer software, hardware, or networks to try this experiment, but have instead build an Internet-accessible, remote-controlled model car† that provides a "race driver's" view via a video camera mounted on the model car. The remote user can see live video from the car, and using a mouse, control the speed and direction of the car.

The challenge was to build a car that could be controlled by novice users in narrow corridors, and that would work not only with the full motion video that the car natively provides, but also with the limited size and frame rate video available over the Internet multicast backbone [1]. We have built a car that has been driven from a site 50 miles away over a 56kbps IP link using *nv* format video at as little as one frame per second and at as low as 100 by 100 pixel resolution. We also built hardware to control the car using a slightly modified voice grade channel videophone. Our experience leads us to believe that it is now possible to put together readily available hardware and software components to build a cheap and effective telepresence.

1. Introduction

The original motivation for building a remote-controlled car was as an application for a locally developed infrared local area network, where the car would be controlled and transmit live compressed video over the IR network. The toy car came together more quickly that the network and other necessary parts (a digital video imager, in particular), and so we ended up building the Internet accessible telepresence that we describe here.

One key goal was to allow novice users to be able to walk up to a workstation and control the car, thus the user interface required careful thought. Second, since we wanted to control the car over the Internet, we wanted it to deal with slow frame rates and variable delay. To do so, we needed to highly compress the video in order to get even a few frames per second. Third, we wanted to multicast the car's video over the Internet Multicast Backbone (MBONE) [1] as well as over point to point logical links. Finally, to minimize effort, we wanted to reuse standard software modules as much as possible.

All of these goals were met in whole or in part in our design. The car does run over the MBONE, and since this video is compatible with the *nv* program [2, 3], this video can be received by a fairly large number of receivers. We were able to use much of the *nv* software and retail hardware to build the car. However, our desire to make the presence easy to pilot, even by a user unfamiliar with it or its environment were not fully realized. We discuss why this is so in the Section 5.

†

In the following sections, we describe the car in terms of its hardware, software and its human interface. Producing a plausible human interface was the most challenging part in some ways, considering that the base car used is meant to be a toy for someone with the reflexes of a video game trained teenager. We also discuss some thoughts on a two way telepresence and on the use of slow frame rate video, in general.

2. The Hardware

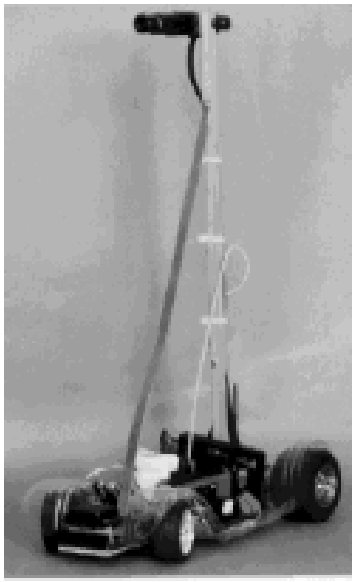

Figure 1: The remote controlled car

The hardware (Figure 1) consists of a radio controlled model car about eighteen inches long with a stock motor, 1 to 7 gearing (slower than normal), and a solid state speed controller. On the car we mounted a Sony 999 video camera and a Microtek 915 Mhz video transmitter. The Microtek transmitter is also used to transmit audio from the car. A separate transmitter can send audio to the car, but we don't as yet have a computer interface for the audio to the car.

The camera is mounted about eighteen inches above the ground rather than directly on the car to give a more realistic driver's eye view. (Even though the car is a scale model the world around it is not. The ankle height of a scale position camera gives a very strange world view.) We slightly modified the 75 Mhz AM radio transmitter so that it could be connected to a custom made computer interface. The radio transmitter conventionally used two joysticks, one for the throttle the other for steering.

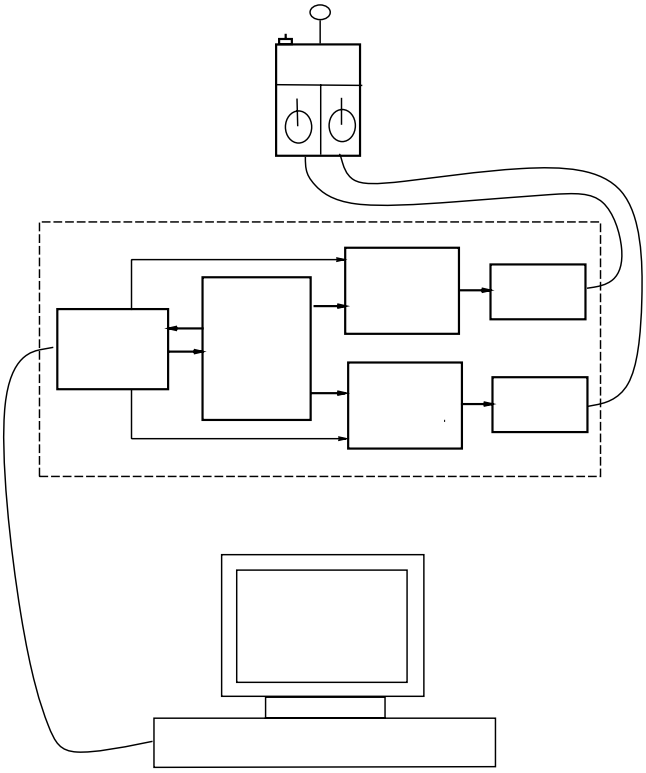

Figure 2: Interface electronics

The computer interface (Figure 2) uses the parallel printer (Centronics) port on an SGI Indigo workstation, but is suitable for any machine with a Centronics port. The interface does the standard Centronics handshaking and passes the eight bit parallel data either to a D/A converter that controls the throttle, or to a D/A converter that controls the steering depending on the high order bit. The outputs of the D/A converters are buffered through operational amplifiers connected as Voltage Controlled Voltage Sources (VCVS) because the D/A converters have insufficient current drive.

The throttle operational amplifier's output is connected to the the radio's controls via half of a double pole double throw switch. The switch disconnects the potentiometer (joystick) and connects the operational amp, or vice versa if one desires manual control of the radio transmitter. The steering operational amplifier is connected similarly to the other joystick's output.

3. The Software (Local Operation)

There are two versions of software used, one for local operation with full motion video and another for network operation using compressed video. For local operation, the software runs as an X-Window application. Two windows are used. One is the mouse tracking window, the other is the video window. The video window is set to be the same size as the mouse tracking window and placed over the top of it. The mouse is tracked in the hidden bottom window and its scaled (-1,+1) X and Y coordinates are passed to the program. The program sets its throttle (forward or reverse) proportional to the Y axis position and its steering proportional to the X axis position. For example, if the mouse is in the upper left corner of the video window, this is interpreted as turning left, while moving forward. The throttle setting is only valid if mouse button 1 is pressed, otherwise the throttle is neutral. We have found this is essential to make the car usable. The central ten percent of the X and Y axes are a dead zone with neutral steering and neutral throttle. The dead zone is to minimize the hand-eye coordination needed to control the car - to stop the car, the driver simply has to enter this zone.

4. Network Software

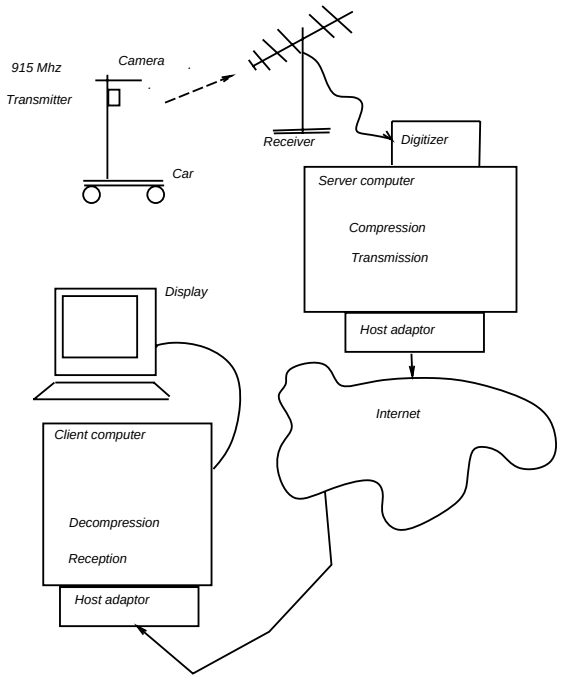

Figure 3: Sending side software modules

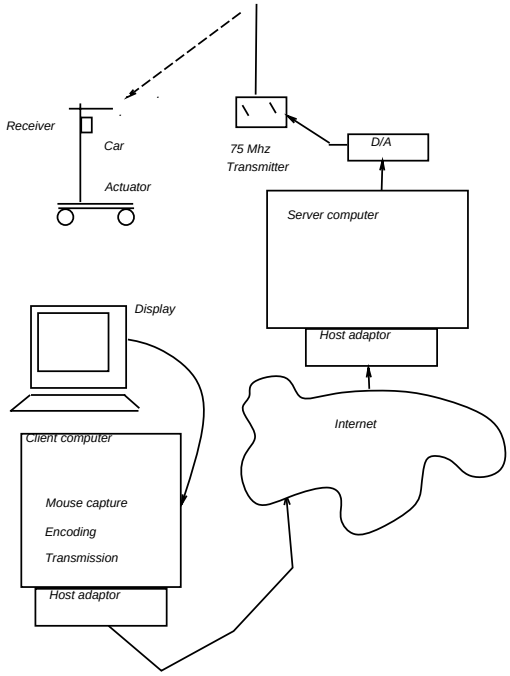

Figure 4: Receive side software modules

Once the software was written and the car test locally we built software for use over any IP network, in particular the IP multicast backbone (MBONE)[1]. The controlling host is connected to the car as before and gets input video as before. It is, naturally, the server for a remote client (Figures 3 and 4). The process

running on the server accepts mouse tracking and button data from the client and returns to the client a few frames per second of software compressed video. There are two client processes. One process decompresses and displays the video, the other tracks a mouse in the video window and forwards that information to the server. The video is sent using UDP packets; the mouse tracking information is sent using TCP packets.

The software used to send the video was derived from the widely used network video program - *nv* [2, 3]. This program is widely used for videoconferences on the Internet, particularly on the multicast backbone (MBONE) [MBONE]. A Tk/Tcl [4] interface was added to standard *nv* to allow only an authorized user to control the car. The authentication for this user is via a challenge-response password. If the car's control computer is on a multicast backbone, others may watch the video while the authorized user drives the car. This feature comes free by using *nv* as the basis of the video transmission software.

Several modifications were made to the *nv* program in order to improve its video compression rate, while still retaining interoperability with standard *nv* programs. First, the change from YUV format video to the internal representation used by *nv* is now done only for blocks that need to be updated, rather than for all blocks as in the original program. Second, the central loop used for the conversion has been unrolled. Third, this loop was optimized for our RISC workstation by using word rather than byte operations wherever possible. With these speedups, given unlimited network bandwidth, the compression speed for scenes with comparable motion increases from about 1.5 frames/sec to 15 frame/sec.

5. Experience Driving The Car

The difference between driving the car over the network and driving the car directly from the control computer is that the video changes from full NTSC video to a few frames per second of compressed video. Surprisingly, at least for driving the car around the narrow corridors of our facility, this is of little practical consequence. The reason is that the car has a high power to weight ratio and so one needs to pulse the throttle by pulsing the mouse button. Thus, the best way to drive the car is to point the wheels, pulse the throttle for a fraction of a second and then see what progress has been made. For this operation one really only uses a few frames per second of information: all of the intermediate frames are a waste. This is quite unlike video conferencing over the Internet where the slow frame rate is very disconcerting and, in fact, the video is sometimes more of an annoyance than a help.

We have found that pointing the mouse in the video window is an effective way to drive the car. We have calibrated the steering so that to drive to an object, the driver merely has to point to the object and click on it. This has made it possible for users to gain a feel for driving the car within minutes: on several occasions, we have invited people walking down the hall to play with the car, and in every case, they were able to do so. The use of a mouse to drive seems to be not too big a problem. We can, alternatively, point by looking at the window with a "head mouse" that we have developed - here, using a Polhemus sensor mounted on a bicycle helmet allows the computer to move the mouse to wherever the user's head points to.

While it is in some ways more natural to drive the car with a driver's eye view of the world, it is actually more difficult to drive it this way than it is to drive the car in the conventional manner of looking down on the car and its environment while controlling it with a hand held radio with joystick controls. This is true even when we operate the car locally and have full motion video available. There are, we believe, three reasons for this, though the last one is dominant.

First, the radio control's joystick is more natural to use than a mouse for car control applications, because it provides proportional tactile (force) feedback that a mouse does not.

Second, vision driving the car is monocular rather than stereoscopic. For a moving vehicle in an ordinarily scaled world this makes little difference. We get depth clues from having moving views and from the scale of objects in the scene. Stereoscopic vision is easy to achieve via an optical lens attachment and an optical viewer in front of the terminal.

Third, the camera is in a fixed position relative to the car. (We are in the process of rectifying this. The camera is small enough to be turned by being directly mounted on standard model servos. If one connects two servos at right angles to each other and mounts the camera on those a panning and tilting through about 90 degrees is simply achieved. One does need to use a four channel radio for this and modify the control board and software to deal with the four channel setup.) Because one cannot look around at will

relative to the car it is much more difficult to understand where the car is within the environment than it otherwise would be. We find that, given this constraint, to drive well one needs to 'internalize' the environment in which one is going to drive before attempting to do it. This works well except when one finds himself facing a blank wall and able to see no other features. At this point being able to look around directly would be a major help. The current way to accomplish this is to back up while turning the wheels sharply until one again recognizes where the car is in the environment. (We have driven the car mostly in the corridors of Bell Labs which are narrow and uniformly beige except for doors. This is as difficult a case as one is likely to encounter.)

6. Other Presences

The setup we have allows operation with any presence using a two channel radio. We had so much fun with the toy car we decided to fly a model airplane. The airplane requires a four channel radio and controller. The ailerons and elevator are controlled analogously to the car's steering and throttle respectively. The throttle and the rudder are adjusted (trimmed really) using keys on the keyboard for increasing and decreasing them. The challenge is to build a unit controllable over an Ethernet that can do software video compression with small enough delay so that we can avoid crashing. (We have completed the airplane, but haven't flown it.)

We would like to try a model sailboat, because one of the authors found that sailing a radio controlled model sailboat was much more difficult than sailing a real boat. This was due mostly to the difference in visual perspective from the edge of the pond compared to the cockpit of a boat. (A lot of the fine control in sailing depends on tactile feedback.) The ultimate would be to fly a toy helicopter, but this will involve a different input scheme and more complex software and hardware, since model helicopters use five channel radios.

We have contemplated building a telepresence for a long distance telecommuting application. Basically, one needs an enhanced version of the model car with two way audio. The unit needs to have more battery power and some safety limit switches but is otherwise quite similar to the toy car. We decided against doing this because of problems in the building environment. Such a remote presence in our environment needs to range through a large complex of buildings and requires substantial battery power. It needs to be strong enough to open doors and agile enough to do so as well as to call elevators. All of this requires a large, heavy device that would be a danger if it were to, say, fall down a flight of stairs. Its grip could likewise be a danger. If we had a door-less single floor in which to operate, such a presence could be made both more simply and safely. It would still need limit switches, but no longer needs to really grip. It could simply come in to chat and observe and would offer a long distance commuter a much better sense of belonging to a group.

7. Potential Applications

Beyond the obvious application as a toy controller for the well- heeled hobbyist, what might be done with this technology particularly when it is used over a network? We have hardware to allow the car to be controlled from a slightly modified voice grade line videophone, in addition to the IP network control already mentioned. In a network the car or something similar becomes useful for remote surveillance. One can easily imagine a doctor wandering around a clinic looking a patients. One could also tour a house for sale from a realtors office. A potential application is in the making of movies or videos. We have heard that model helicopters, conventionally flown, are already occasionally used for this. Certainly somewhat similar of technology has been used to look around dangerous environments before, but networking would make this sort of application even more appealing. Of course, the car might have to be modified considerably for many dangerous environments for reasons of functionality or safety.

8. Conclusions

We have described a telepresence based on a modified model car, which is equipped with a video camera and can be controlled over the Internet. The key contributions of our work are to recognize that such presences can be built with off the shelf components. Second, we have given much thought to an easy to use interface, which has proved to be successful. Third, our software is compatible with existing packet video standards, and thus can be widely used. Finally, we have tuned the low bandwidth available from the

Internet to the capabilities provided by the car.

We are aware of a few other similar projects. Some of these involve tele-operation of cameras or robots over the Internet. These efforts lack the rich functionality provided by our telepresence - for example, Internet cameras cannot be moved around, only panned or zoomed. Other presences are confined to the local area, and do not leverage the existing packet video infrastructure in the Internet. We believe that our presence is the first to marry the mobility of a car to the packet video standards in the Internet and the MBONE.

Operating a model car, airplane or boat from a computer screen gives us a different and more realistic feel than the normal method of operation. Using a network to operate a vehicle is entirely compatible (for cars, not airplanes) with the slow video provided by the Internet or a videophone and is, in fact, a more useful application for this than the too slow "talking heads" application for which they are normally used. Teachers of the handicapped understand that video is more useful for dealing with things than people. There is a saying among them that being deaf cuts you off from people and being blind cuts you off from things. Perhaps this is the right way to use packet video, not to look at people, but to look at (and eventually manipulate) objects.

9. References

1. S. Casner and S. Deering, First IETF Internet Audiocast, *ACM SigComm Computer Communication Review 22*, 3 (July 1992).
2. R. Fredericks, `nv-3.2` network video tool, *Available for anonymous FTP from ftp.parc.xerox.com*, 1993.
3. R. Fredericks, Experiences with Real Time Software Video, *6th Packet Video Conference, Portland, OR*, September 1994.
4. J. Ousterhout, Tcl and the Tk Toolkit, *Addison Wesley, Reading, MA*, 1994.