Effective Vehicle Teleoperation on the World Wide Web

Sébastien Grange[1], Terrence Fong[2] and Charles Baur[1]

[1]Institut de Systèmes Robotiques
L'Ecole Polytechnique Fédérale de Lausanne
CH-1015 Lausanne EPFL, Switzerland

[2]The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA

Abstract

Our goal is to make vehicle teleoperation accessible to all users. To do this, we are developing easy-to-use yet capable Web tools which enable efficient, robust teleoperation in unknown and unstructured environments. Web-based teleoperation, however, raises many research issues, as well as prohibiting the use of traditional approaches. Thus, it is essential to develop new methods which minimize bandwidth usage, which provide sensor fusion displays, and which optimize human-computer interaction. We believe that existing systems do not adequately address these issues and have severely limited capability and performance as a result. In this paper, we present a system design for safe and reliable Web-based vehicle teleoperation, describe an active and dynamic user interface, and explain how our approach differs from existing systems.

1 Introduction

Since the 1950's, vehicle teleoperation has centered primarily on rate-controlled systems for hazardous environments (e.g., underwater ROV's). In these systems, a trained operator controls the vehicle's motion (rotation and translation rates) via hand-controllers and receives feedback from cameras and sensors. Recent systems have emphasized the use of multi-modal operator interfaces and supervisory control[2][9][16].

Yet, despite decades of research, vehicle teleoperation remains problematic. In particular, operator interfaces are often cumbersome, need significant infrastructure, and require extensive training. Many interfaces overwhelm the user with multiple displays while simultaneously demanding high levels of cognition and motor skill. As a result, vehicle teleoperation is a domain for experts.

Our objective is to make vehicle teleoperation accessible to all users, novices and experts alike. To do this, we need to make operator interfaces that are easy to deploy, easy to understand and easy to use. Thus, we are developing Web-based tools to enable efficient, robust vehicle teleoperation in unknown, unstructured and dynamic environments.

Using Web-based tools as a teleoperator interface raises numerous research issues and concerns. For example, data transmission through the Internet is often irregular and unreliable. Consequently, the system must be designed to handle potentially unbounded delay or loss of data. Additionally, the available network bandwidth varies greatly, depending on network hardware and network load. Thus, the amount and type of information that can be exchanged between a remote system and a user is severely limited.

In the following sections, we present a system for safe and efficient, Web-based vehicle teleoperation. We begin by discussing related research and design issues. We then describe our system architecture and implementation. Finally, we present some experiences with this system.

2 Related Research

2.1 Web-based telemanipulation

The first telerobot appeared on the Web in 1994[4]. Since then numerous other Web-based robots have been deployed, notably [14]. The common characteristic of these systems is that they only provide interaction with robot manipulators. This is because the interaction with the user does not need to be dynamic: a known environment is subjected to discrete changes caused only by the manipulator. Typically, the following scheme is used:

- the user monitors system state via a static HTML document (e.g., a still image and numerical data)
- the user enter parameters for the robot to execute (normally a single motion)
- the robot executes the command without supervision or further user interaction
- once the command is completed, or if an error occurs, a new HTML document (containing the updated system state) is generated

If these systems show that interaction with physical devices through the Internet is technically possible, they do not provide tools to deal with unexpected events, such as dynamic changes in the remote environment. Even though some systems present the user with enhanced imagery (e.g., graphic overlays), no real collaboration between the user and the robot is possible; primarily because the robot provides no feedback during the motion execution phase.

2.2 Web-based vehicle teleoperation

More recently, vehicle teleoperation systems have appeared on the Web[10][12][13][15]. These systems attempt to provide user-friendly interfaces for remote driving. As with Web-based telemanipulation, most of these operated within controlled environments.

KhepOnTheWeb, allows the user to control a Khepera robot in a static environment[10]. The interface, uses *CGI*[1] programs and *server-push* video[2]. An overhead camera provides continuous views of robot position and an on-board camera shows the environment. Even though position can always be seen, the transmission delay and the lack of orientation aids, makes the system hard to use. Moreover, because the Khepera operates in a closed environment, free motion and exploration are impossible.

Another CGI system is the WebPioneer[15], in which the user drives a Pioneer[3] robot in an unknown (i.e. not described *a priori*) room. No external view of the robot is offered, but server-push video from an on-board camera provides feedback. A graphical command set (translation, rotation) is used to control the robot. The simplicity of the interface, though easy to use, severely limits the interactivity. Additionally, the single video stream is not sufficient for the user to get or maintain situational awareness.

With both these Web-based vehicle teleoperation systems, on-board sensors are only used to prevent collision with immediate obstacles, and no sensor feedback is given to the user. Moreover, the user is forced to rely entirely on video feedback to navigate and to detect obstacles. However, since delay-free transmission cannot currently be guaranteed on the Web, remote driving with these systems is often arduous and confusing.

3 Design issues

3.1 Vehicle teleoperation

The design of a vehicle teleoperation system is governed by many factors. For most systems, however, the primary constraint is the communication link. In particular, bandwidth directly restricts the quantity and quality of information available to the operator for decision making. Additionally, transmission delay affects the reliability of remote operation. Beyond a certain delay, manual control of the vehicle may become highly error prone or impractical[8]. Thus, when we design a vehicle teleoperation system, it is critical that we consider limitations imposed by the communication link.

Human beings are far from perfect. With manual control, teleoperation is limited by the operator's motor skills and his ability to maintain situational awareness. Fatigue, lack of concentration, inadequate displays, and poor feedback all contribute to reduced performance. Additionally, humans have difficulty building accurate mental models of remote environments. Distance estimation, obstacle detection and attitude judgement are problematic, especially for untrained operators[8]. Consequently, we must design the operator interface so that it maximizes usability.

The number and type of sensors carried by a vehicle also directly influence the user's ability to perceive the remote environment and to identify hazards. Even with an ideal communications link (i.e., unlimited bandwidth and delay-free), insufficient or inadequate sensors can dramatically reduce teleoperation system performance. As a result, we need to ensure that we employ appropriate sensing for the vehicle and environment.

Finally, a robot operating in an unknown or unstructured environment may be exposed to dynamic hazards. This is particularly problematic when the hazards are time-critical or malicious. If the operator is unable to perceive moving obstacles, or if he sees them too late due to communication delays, then safeguard functions have to be managed by the robot itself. Thus, it becomes important for the robot to have some level of autonomy.

Overall, a well-designed vehicle teleoperation system requires effective and efficient use of the communication link, a clear and compelling operator interface, and an architecture which supports local autonomy.

3.2 Web-based teleoperation

In the case of Web-based teleoperation, a computer network provides a communication link with severely limited bandwidth and variable delay. Given this constraint, data exchange between the robot and the user must be carefully managed. As a consequence, both the robot and the operator interface must perform data pre-processing and encoding before transmission. Additionally, bandwidth-expensive components (e.g., video) are discouraged and other mechanisms must be used to convey information.

To date, most Web-based systems have been built using CGI programs. Due to limitations of CGI, however, these programs cannot listen to user commands and provide useful feedback at the same time. Moreover, a CGI program opens and closes a new network connection each time the program is invoked, resulting in significant overhead and delay. The use of multiple simultaneous CGI programs (e.g., to support a interface with multiple HTML frames)

[1] Common Gateway Interface

[2] a Netscape™ extension

[3] Pioneer is a registered trademark of ActivMedia, Inc.

only exacerbates the problem. As a result, we need to seek alternatives to CGI for handling Web interaction.

The design of any user interface often involves a trade-off between ease of use and the capacity for complex tasks. Modern interface design approaches this problem by focusing on the user; i.e., the interface is designed to maximize usability by a target set of users. Yet, this can be difficult for Web-based teleoperation because the interface needs to support users having diverse skills, knowledge, and experience. In particular, Web interfaces need to be designed so that novices will feel comfortable, yet must not unduly constrain experts. If the interface fails to do either, users will quickly lose interest and avoid using the system.

To support a wide range of users, a Web-based teleoperation interface must present data clearly and concisely. Complex data, such as robot position and coordinate frame transforms, should be hidden from the user. Instead, the interface should present the data using readily accessible displays (e.g., a moving map).Parameters and operations which are best handled by the robot (e.g., collision avoidance) should be removed from user control (i.e., the interface only needs to provide feedback). Finally, the interface must inform the user about the reliability of the data it displays (so that the user can make informed decisions).

A Web-based vehicle teleoperation interface, however, has to also consider other parameters. Because of varying transmission delay, the information displayed may not match the actual state of the remote system. Thus, the interface has to be designed so that commands will not affected by delay. Specifically, motion commands should be sent to the robot relative to the state displayed by the interface. Moreover, the interface must be robust enough to deal with communication failure and warn the user if the connection to the robot is lost.

4 System design

4.1 Overview

We have created a system, *WebDriver*, for safe and reliable Web-based vehicle teleoperation. Our system supports a wide range of users, novice and experts alike. We designed WebDriver to minimize bandwidth usage, to provide a compelling and active user interface, and to optimize human-computer interaction[5].

The WebDriver architecture (Figure 1) has 3 elements:

User Interface. The user interface is a Java applet which runs in a Web browser. It accepts user commands and provides continuous feedback from the robot's sensors. The user interface is connected to the rest of the system via a persistent network link.

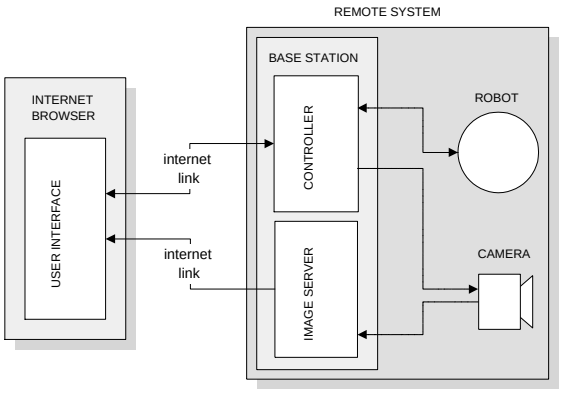

Figure 1. WebDriver system architecture

Base Station. The base station performs three processing functions: communication with the user interface, image processing, and high-level robot control.

Robot. The teleoperated vehicle is equipped with on-board sensors (including a pan-tilt camera) and a motion controller. It is connected to the base station via a radio modem and analog video transmitter.

4.2 User interface

The WebDriver user interface is shown in Figure 2. The interface contains two primary tools, the *dynamic map* and the *image manager*, which allow the user to generate commands and to receive feedback. The interface also has controls for manipulating the display and for positioning (pan angle) the robot's on-board camera. A proximity light indicates distance between the robot and nearby obstacles.

We designed the interface for ease-of-use and flexibility: the user is able to see system status at a glance and can specify robot motion commands (rotation, translation) in multiple ways. The interface is also designed to support single-touch interaction (to enable use with touchscreen displays in Internet kiosks).

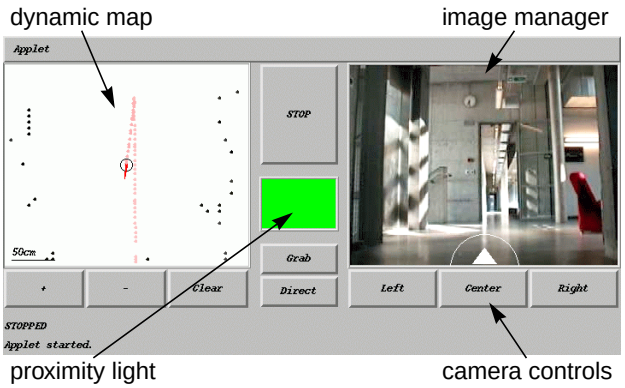

Figure 2. Web interface for vehicle teleoperation

Dynamic Map

The dynamic map (see Figure 3) is constructed using ultrasonic sonar and robot position. Sensor data is filtered, stored and then displayed as colored points: gray for sensed obstacles, red for robot position. The color of each point reflects the confidence value for the point, i.e., certainty that an obstacle exists in that location or that the robot was in a specific position. Dark colors indicate high confidence.

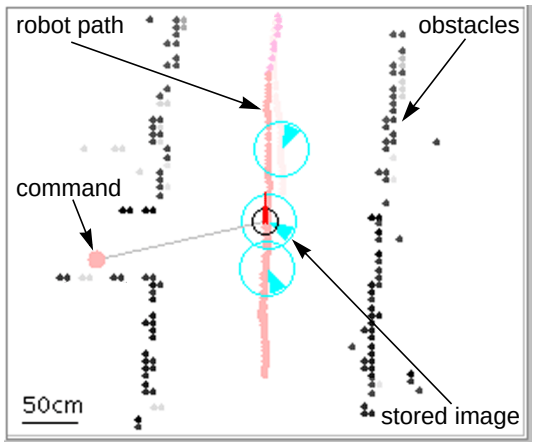

Figure 3. Dynamic map

The user can drive the robot by clicking on the map at the location he wants the robot to go. The robot will then try to move to that point, avoiding obstacles it encounters on the way. The map also displays locations (blue circles) at which images were stored by the image manager.

Image manager

The image manager displays and stores images from the robot's camera. Unlike other Web-based systems, such as [10] or [12], we do not use server-push video because it excessively consumes bandwidth. Instead, we use an event-driven model to retrieve images when:

- user issues request (for new or stored image)
- robot is stopped and is awaiting command
- obstacle detected
- interframe timer expires (5 seconds)

The timer ensures that the image manager always displays a recent image (i.e., if no other event occurs, a new image will be retrieved when the timer).

Figure 4 shows typical image manager displays. On each image, the camera orientation (pan angle) and indicators for detected obstacles (based on proximity sensor readings) are overlaid. Additionally, if a stored image is shown (Figure 4, right), a "replay" symbol is added. The user can change the robot's heading by clicking in the image (i.e., the robot will turn so that the selected feature becomes centered in the image). Clicking in the white area at the bottom of the image commands a forward translation.

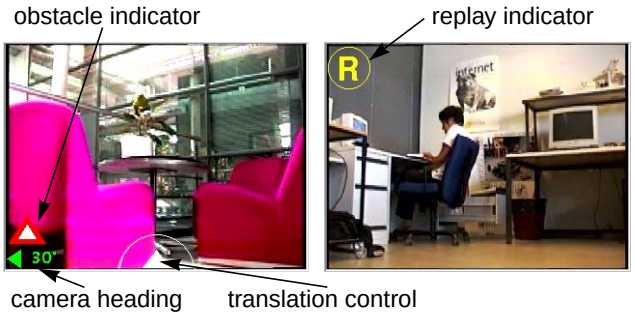

Figure 4. Image manager
current image (left), stored image (right)

User Interface Design

We implemented the user interface as a Java applet. This allows the interface to be easily deployed and used in any Java-enabled Web browser. Figure 5 shows the Java object structure of the user interface.

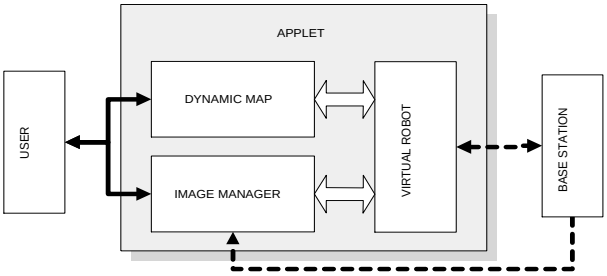

Figure 5. User interface object structure

The primary Java object is the *virtual robot*. When the user interface is active, the virtual robot maintains an open connection and mirrors the robot's state. Messages (status, sensor readings, etc.) received from the base station are used to update the virtual robot. In this way, all applet objects have continuous access to current data.

The other user interface objects process user input and generate displays. When the user gives a robot command, the object receiving the input invokes a method in the virtual robot. The virtual robot then forwards this command to the real robot. This modular design enables control of different mobile robots with the same user interface. Only messages sent by the virtual robot need to be customized for each type of robot.

By using a Java applet (instead of HTML/CGI) we are able to maintain a persistent network connection between the Web interface and the robot controller. This enables us to perform data transfer through the connection at any time, in either direction and with minimal delay. Additionally, the continuous link allows us to avoid repeated connection setup/teardown overhead, which reduces network resource usage and improves overall system responsiveness.

4.3 Base station

The base station is responsible for communication with the user interface, image processing, and high-level robot control. It manages all information exchange between the user interface and the robot, performing data encoding and compression to speed transmission, reduce delay and minimize bandwidth consumption.

When the user interface is connected, a base station process continually sends messages containing robot state and sensor information. The same process also receives robot commands from the user interface and relays the messages to the high-level robot controller.

Image processing is performed by a TCP socket-based image server. Whenever the user interface requests an image, the server captures a video frame, converts and compresses it into a JPEG image, and returns it to the user interface. Our image server executes quickly (1/20 second for 1/4 size NTSC), thus providing an efficient, network friendly (i.e., it consumes minimal bandwidth) solution for sending video imagery.

The base station performs high-level robot control using a controller based on Saphira[7]. The controller supports complex robot behaviors such as motion planning, collision detection and obstacle avoidance. This gives the robot the ability to override user commands that would otherwise jeopardize its safety. We feel this is particularly important in a context where a command might not arrive in time to the robot, or when the user might not have sufficient information to make an appropriate decision.

4.4 Robot

We designed the WebDriver system for use with a Pioneer AT mobile robot. The PioneerAT is a skid-steered wheeled vehicle which is capable of traversing moderately rough natural terrain. The PioneerAT has an on-board microprocessor-based controller which manages vehicle sensors and performs motion control.

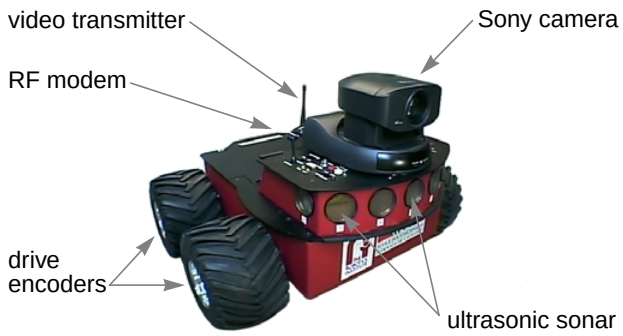

Figure 6. PioneerAT configuration

Our PioneerAT (see Figure 6) is equipped with a Sony EVI-D30 pan-tilt-zoom color CCD camera, a ring of seven Polaroid piezoelectric ultrasonic sonars, power monitoring, and drive encoders. An analog NTSC video transmitter and a RF modem (9600 baud) are used for wireless communications with the base station.

5 Experiences

We have found that the WebDriver architecture effectively frees the system from bandwidth limitations and transmission delay imposed by the Web, thus enabling efficient and effective control of the robot. Specifically, the architectural features that enable this are:

- pre-processing and compressing message data
- avoiding direct control: no time dependent commands, absolute positioning commands only
- giving the robot enough autonomy to ensure safeguarding functions

We conducted informal user tests with the WebDriver. Anecdotal evidence from a range of users suggests that the system is quite reliable and robust. Additionally, since the robot performs safeguarding functions (avoiding obstacles autonomously) continuous user attention is not required, thus making the system safe and easy to use. Users have found it easy to explore different rooms and drive along corridors using only on-board camera images and dynamic map representations of the environment.

Limitations

The WebDriver system has several limitations. Most significantly, we estimate robot position using dead-reckoned odometry. Since, we use the robot position to register sensor data, the dynamic map accuracy is limited by dead-reckoning error. Consequently, the map only presents locally correlated information. Furthermore, the map's reliability decreases precipitously with movements involving large wheel slip.

The limited number and reliability of proximity sensors (ultrasonic sonar) also impact system performance. The standard PioneerAT is equipped with 7 forward-facing sonar devices which provide coverage of a 180 degree range. Since obstacle detection is limited to the front and sides of the vehicle, hazards (especially dynamic ones) located behind the vehicle pose significant problems. Additionally, since ultrasonic sonar data is often noisy (due to multiple reflections and environment characteristics) and imprecise (due to wide beam cone), the resulting dynamic map may also be noisy and imprecise. Filtering the sonar data, such as [1], can improve the situation, but is certainly not a panacea.

Future Work

A number of improvements would make the WebDriver system more reliable. For the user interface, better sensor fusion schemes would make the dynamic map more reliable over long distances and duration. A higher level of autonomy would make map self-calibration and automatic repositioning possible. Also, including the user in the map building process could improve performance.

On the hardware side, increasing the number and accuracy of proximity sensors would provide better obstacle avoidance. Better positioning could be achieved using an external system, such as radio beacons or carrier-phase dGPS. Wheel-slip and wheel-blocked sensors would also improve safeguarding and provide additional positioning information.

6 Conclusion

We have created a system, the WebDriver, for efficient, robust Web-based vehicle teleoperation. The WebDriver supports a wide range of users with diverse skills, knowledge, and expertise. We designed WebDriver to minimize network bandwidth usage, to provide an easy-to-use yet compelling operator interface, and to optimize human-computer interaction.

Our system differs from existing Web-based telemanipulation systems because it supports teleoperation in unknown, unstructured and dynamic environments. Our system differs from other Web-based vehicle teleoperation systems because it provides greater interactivity and safety through an active interface and safeguarding autonomy.

Acknowledgments

We would like to thank the Institut de Systèms Robotiques (Département de Microtechnique, EPFL) for providing research facilities, infrastructure and support.

References

[1] Borenstein, J, and Koren, Y., "Histogramic In-motion Mapping for Mobile Robot Obstacle Avoidance", IEEE Journal of Robotics and Automation, Vol. 7, No. 4, 1991.

[2] Fong, T., et. al., "Operator Interfaces and Network-Based Participation for Dante II", *SAE 25th International Conference on Environmental Systems*, San Diego, CA, 1995.

[3] Fong, T., et. al., "Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation", *AAAI 1999 Spring Symposium*, Stanford, CA, March 1999.

[4] Goldberg, K., et. al., "Desktop Teleoperation via the World Wide Web", *IEEE Conference on Robotics and Automation*, Nagoya, Japan, May 1995.

[5] Grange S., "'Interface Utilisateur Avancee", Microengineering Project Report, Swiss Federal Institute of Technology, Lausanne, February 1999.

[6] Kay, J., *STRIPE: Remote Driving Using Limited Image Data*, Ph. D. Thesis, Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.

[7] Konolige, K. and Myers, K. *The Saphira Architecture for Autonomous Mobile Robots*, in *Artificial Intelligence and Mobile Robots* (Bonasso, R. and Murphy, R., eds.), MIT Press, Cambridge, MA, 1997.

[8] McGovern, D., "Experiences and Results in Teleoperation of Land Vehicles", SAND87-0646, Sandia National Lab., Albuquerque, NM, 1990.

[9] Meier, R., et. al., "A Sensor Fusion Based User Interface for Vehicle Teleoperation", *Field and Service Robots Conference*, Pittsburgh, PA, August 1999.

[10] Michel, O. et al., "KhepOnTheWeb: An Experimental Demonstrator in Telerobotics and Virtual Reality", *VSMM'97*, Geneva, Switzerland, September 1997.

[11] Sheridan, T., "Space teleoperation through time delay review and prognosis", *IEEE Transactions on Robotics and Automation*, October 1993.

[12] Siegwart, R., and Saucy, P., "Interacting Mobile Robots on the Web", workshop, *IEEE Conference on Robotics and Automation*, Detroit, MI, April 1998.

[13] Simmons, R., "Where in the World is Xavier, the robot?", *Robotics and Machine Perception*, Special Issue: Networked Robotics, Vol. 5, No. 1, March 1996.

[14] Taylor, K., Trevelyan, J., "A Telerobot on the World Wide Web", *National Conference of the Australian Robot Association*, Melbourne, July 1995.

[15] WebPioneer Website, ActivMedia, Inc., `http://webpion.mobilerobots.com`, 1998.

[16] Wettergreen, D., et. al., "Operating Nomad During the Atacama Desert Trek", *Field and Service Robotics Conference*, Canberra, Australia, 1997.