

PdaDriver: A Handheld System for Remote Driving

Terrence Fong

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
terry@ri.cmu.edu

Charles Thorpe

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
cet@ri.cmu.edu

Betty Glass

CIS – SAIC
8100 Shafer Parkway
Littleton, CO 80127
betty@cis.saic.com

Abstract

PdaDriver is a Personal Digital Assistant (PDA) system for vehicle teleoperation. It is designed to be easy-to-deploy, to minimize the need for training, and to enable effective remote driving through multiple control modes. This paper presents the motivation for PdaDriver, its current design, and recent outdoor tests with a mobile robot.

Keywords: vehicle teleoperation, remote driving, personal digital assistant, handheld user interface

1 Introduction

For some remote driving applications, installing control stations with multiple displays and high-bandwidth communication is infeasible or prohibitive. For other applications, the vehicle is driven by operators for whom extensive training is impractical. In these situations, we need a driving system that requires minimal infrastructure and that does not tightly couple performance to operator experience[1].

To satisfy this need, we have developed *PdaDriver*, a Personal Digital Assistant (PDA) system for remote driving. *PdaDriver* uses multiple control modes to make vehicle teleoperation fast and efficient. We designed *PdaDriver* to minimize the need for training, to enable rapid command generation, and to improve situational awareness.

We first developed *PdaDriver* in 2000 to teleoperate a military training robot[1]. This version ran on a Casio Cassiopeia and used a wired, serial link. In 2001, we extended *PdaDriver* to support “collaborative control”, a system model in which the robot asks questions to the human to obtain assistance with cognitive tasks[2]. We used this second version to study collaborative navigation and exploration [2, 3].

During 2002, we modified *PdaDriver* to run on a Compaq iPAQ PocketPC with wireless ethernet (Figure 1, left). We did this for several reasons: the iPAQ display works well in direct sunlight (important for field use); the ethernet link provides high bandwidth and low latency; and the iPAQ has greater battery life than the Casio. In addition, we integrated



Figure 1: *Left*, PdaDriver (version 3) on a Compaq iPAQ; *right*, Remote driving with PdaDriver

PdaDriver with SAIC’s *MarsScape* architecture and the Autonomous All Terrain Vehicle Intelligence Lab (AATVIL) robot (Figure 1, right)[10].

AATVIL is based upon a modified Honda Rubicon ATV chassis (Figure 2, left). It has multiple on-board computers and actuation for steering, brake, throttle, and transmission. AATVIL is capable of traversing natural terrain at 5 m/s and can traverse moderately steep slopes (30 deg incline, 20 deg side). AATVIL is equipped with real-time kinematic differential GPS, a north-finding module, and numerous cameras (Figure 2, right) for navigation in day and at night.

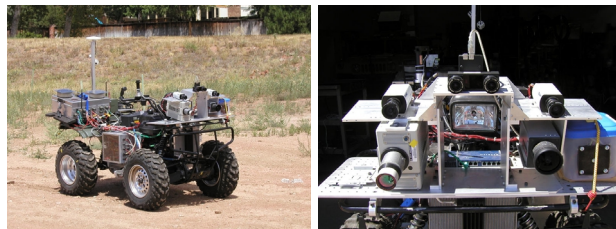


Figure 2: *Left*, Autonomous ATV Intelligence Lab (AATVIL) vehicle; *right*, Camera head with digital color stereo, progressive-scan high-resolution color, multi-spectral, and infrared

2 Related Work

One of the earliest (if not the first) PDA interfaces for remote driving was developed at the Naval Research Laboratory[8]. This PDA interface ran on a Palm Pilot™ and was part of a multi-modal system that incorporated natural language, visual gesturing, and synthetic gestures. A map display and pen (touchscreen) gestures were used to direct an autonomous robot and to help disambiguate natural language inputs.

Several other Palm Pilot interfaces have since been developed. In [9], Rybski et al. describe a simple command interface for operating small scout robots. In [7], Lu, Castellanes, and Rodrigues describe an interface for teleoperating a robot dog. With this system, low-bandwidth video (from the robot’s camera) and four buttons are used for simple rate control. In [12], Skubic et al. describe a system for communicating robot navigation tasks. The system works by having a user sketch a rough environment map and trajectory on the PDA.

Since 2000, there has been growing interest in Windows CE-based PDAs, primarily because these devices have better displays and communication options than their PalmOS counterparts. In [5], Hüttenrauch and Norman describe several iPAQ interfaces for operating an indoor robot. In [13], Suomela and Halme described an iPAQ interface designed for an operator who must work in close, physical proximity to a large service robot.

As the distinction between PDAs and cellular telephones continues to blur, it is only natural for remote driving interfaces to be deployed on the latter. Fujitsu, for example, has recently developed a home robot that can be teleoperated by a NTT DoCoMo mobile phone[4]. With the mobile phone, users can issue direct motion commands or command navigation to pre-defined locations, while watching robot camera images on the phone’s display.

Although all these interfaces are similar in some respects to PdaDriver, there are several important differences. First, PdaDriver is designed for remote driving in unstructured, unknown environments. Thus, unlike other interfaces, which are used for short-range operation or in known environments, PdaDriver emphasizes multiple control methods (e.g., image-waypoint) that are appropriate for exploration, particularly in low-bandwidth and high-delay situations.

Second, PdaDriver is designed for rapid integration. Specifically, the choice of simple messaging protocol and control paradigms enables PdaDriver to be used with a variety of robot systems. For example, we have used the same interface to control indoor research robots and an ATV-based mobile robot. In addition, we are currently working to remotely drive a robot Jeep (CMU Navlab 11).

Finally, with the exception of [5], none of the other PDA interfaces was developed using HCI methods. As such, their designs are *ad-hoc* and provide poor affordances. With PdaDriver, however, we relied heavily on user-centered design techniques, such as heuristic evaluation, to produce an effective interface. We feel strongly that when an interface is well-crafted it becomes transparent: users take it for granted and can use it with minimal effort. On the other hand, if an interface is poorly crafted it is difficult to understand, difficult to use and limits performance.

3 Architecture

The PdaDriver architecture is shown in Figure 3. The user interface (described in Section 4) runs on a PDA and connects to a mobile robot controller, typically located on-board the robot, via a network communication link and TCP sockets.

Two modules, the *PDA Gateway* and the *Image Server*, are customized for each robot and integrated into the robot’s controller. These modules are described in the following section. Although the architecture is optimized for single vehicle operation, simultaneous (switched) control of multiple vehicles from a single user interface is supported.

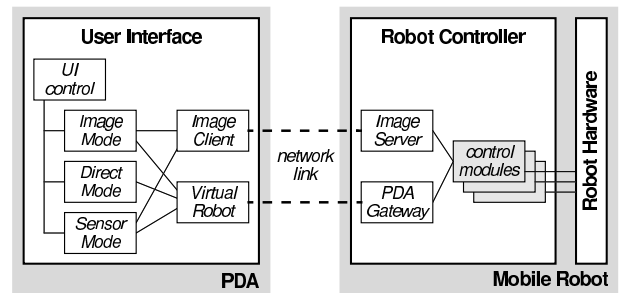


Figure 3: PdaDriver (ver. 3) architecture

When the user interface is running, it maintains a continuous connection to the PDA Gateway. This gateway processes commands from the user interface and outputs robot data (pose, health, etc.). To obtain camera images, the user interface also intermittently connects to the Image Server, which provides JPEG compressed images from the robot’s camera.

Both the PDA Gateway and the Image Server communicate using an open-source network communication library[11]. This design facilitates integration of PdaDriver with different mobile robots. In particular, all robot-specific code (messaging, hardware control, etc.) is isolated to these two modules. Thus, the user interface can be used without modification (except for simple parameter changes) with different robot controllers and hardware.

3.1 PDA Gateway

The PDA Gateway is designed as a proxy server for the user interface[2]. It provides access to robot controller services (motion control, localization, etc.) while hiding the controller’s design and implementation. The PDA Gateway communicates with a simple protocol that works well even over low-bandwidth connections. The protocol is text-based (which speeds integration testing) and synchronous (to reduce latency and to improve operational safety).

Whenever the user interface is connected, the gateway continually monitors data transmission. If it detects a communication problem (network outage, loss of connection, etc.) or that the interface is no longer responding, the PDA Gateway immediately closes the connection, then stops and safeguards the robot. Thus, the PDA Gateway ensures that operator commands are only executed while the interface is connected and functioning correctly.

3.2 Image Server

As an alternative to video, we use an event-driven Image Server[2]. This server helps minimize bandwidth consumption by transmitting images only when “significant” events occur. Specifically, the Image Server captures a frame, compresses it into a JPEG image, and sends the image only when the operator issues a request, the robot stops, an obstacle (static or moving) is detected, or an interframe timer expires.

Event-driven imagery is a flexible mechanism. For example, if an application allows a high-bandwidth, low-latency communication link, we set the interframe timer to a low value. This produces an image stream that approximates low-rate video. Alternatively, if the link is low-bandwidth, we set the timer to a high value. In this case, images are transmitted only when key events occur, thus minimizing link usage.

4 User Interface

The PdaDriver user interface is written in Personal Java™ and runs under the Insignia Solutions Jeode™ Java Virtual Machine. Because the interface requires high-resolution color, we usually employ Windows CE-based PDAs. However, we have also used Sharp’s Linux-based Zaurus¹ and are currently evaluating Sony’s Clie.

Vehicle teleoperation in unstructured, unknown environments requires flexible control. Because both the task and the environment may vary (depending on situation, over time, etc.), no single control mode is optimal for all conditions. Thus, we designed and implemented three modes in the PdaDriver user interface:

¹The Zaurus is thinner and sleeker than other Windows CE-based PDAs, but has poorer display back lighting than the iPAQ.

1. Direct (“raw teleop”) mode. A “virtual” joystick combined with low-rate video.
2. Image mode. The operator designates a path by clicking a sequence of waypoints in a still image.
3. Sensor mode. Allows selection of camera used for remote driving.

We designed each interface mode using a combination of heuristic design, heuristic evaluation, and cognitive walkthrough[3]. We chose these methods because they are rapid, can be used throughout the development process, and have been shown to produce high-quality interfaces in a variety of domains.

4.1 Direct Mode

Direct mode supports two-axis, rate control remote driving. This mode is appropriate when the terrain is benign (e.g., few dangerous obstacles), when high operator workload is acceptable, and when latency (i.e., communication delay and image update rate) can be tolerated. Direct mode is often used for performing long-distance, high-speed traverses.

In direct mode, camera images and a graduated cross are shown on the PDA screen (Figure 4). Pressing the vertical cross axis commands translation and the horizontal axis commands steering curvature. The iPAQ “joypad” (a 4-button cursor control) may also be used to command fixed driving rates (forward, turn left, etc). Releasing the screen press, or the joypad, stops the robot.

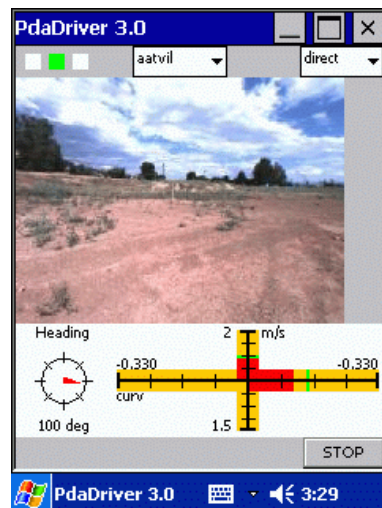


Figure 4: Direct mode enables rate control driving with low-rate video.

As the robot moves, the direct mode display is updated to show current compass heading, translation rate (m/s) and curvature ($1/m$). Commanded translation and curvature are shown as green markers on the graduated cross. Current translation and curvature are shown as filled red bars.

4.2 Image Mode

Image mode supports image-based waypoint driving, inspired by Kay’s STRIPE system[6]. However, instead of Kay’s ad-hoc calibration, Image mode uses Tsai’s method for camera calibration and distortion correction[14]. Image mode (Figure 5) shows an image from the robot’s camera. Horizontal lines overlaid on the image indicate the projected horizon and robot width. The user specifies a path by clicking a series of waypoints (red circles) on the image. When the *go* button is pressed, PdaDriver sends the projected world points to the robot. As the robot moves, a status bar displays the robot’s progress.



Figure 5: Image mode supports waypoint driving via a sequence of projected image points.

To transform image points to world points, we assume that the ground-plane is locally flat and use simple perspective projection. We perform forward projection by first computing undistorted coordinates (Tsai dewarp) and then transforming from the image point to the world frame. Although this procedure computes 3D world points, we only use 2D coordinates (i.e., ground points) for driving. With this approach, we typically achieve less than 5% downrange projection error.

4.3 Sensor Mode

One of the difficulties with outdoor remote driving is that environmental characteristics may vary with time, location, and situation. For example, scene illumination can rapidly change due to sun position, shadows, and other factors (fog, dust, etc). Sensor mode addresses this problem by allowing the operator to select which camera is used for remote driving. With AATVIL, for example, six forward-mounted cameras are available, three of which are shown in (Figure 6).

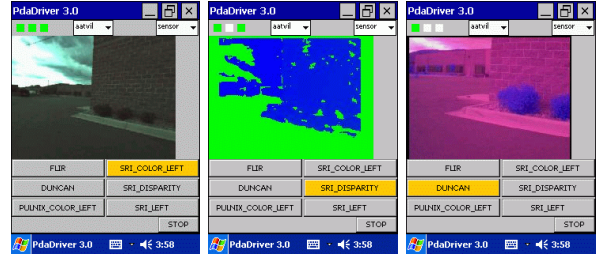


Figure 6: Sensor mode supports multiple cameras. *From left:* color, disparity (range), and multispectral.

5 Experiment Results

In August 2002, we conducted field tests at SAIC (Littleton, Colorado), using PdaDriver to remotely drive AATVIL over natural terrain (dirt and loose gravel with sparse vegetation) and in a parking lot (adjacent to a brick building surrounded by shrubs). During the tests, the operator was located 50-100m from AATVIL, but did not directly observe the robot.

5.1 Direct Mode

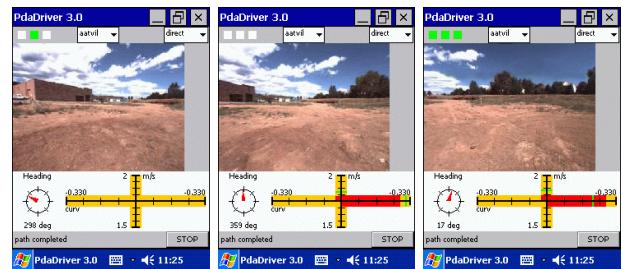


Figure 7: *Top*, AATVIL operating on natural terrain; *bottom*, Direct mode display sequence

Figure 7 shows a typical direct mode driving sequence with AATVIL on natural terrain. In this test, the robot was initially stopped, then was commanded to drive forward and to turn to the right. Image latency (time from capture to PDA display) was measured at 800 msec. Control latency (PDA command to robot execution) was approximately 500 msec. Because of these delays, some operator training was required to effect smooth vehicle control.

5.2 Image Mode



Figure 8: *Top*, AATVIL driving on a dirt road; *bottom left to right*, Image mode display sequence (color camera): designated path with three waypoints, first-half of path achieved, approaching final waypoint.

Figure 8 shows a typical image mode driving sequence using a color camera. In this test, AATVIL was commanded to drive on a dirt road (Figure 8, top), parallel to several mounds of dirt located to the left of the vehicle. To do this, the operator began by designating a path with three waypoints in the image (Figure 8, bottom left). He then pressed the *go* button and AATVIL then autonomously moved along the path (Figures 8, bottom center and right).

Because image mode commands points in a world reference frame, we found that vehicle motion accuracy was highly dependent on localization performance. In addition, although image-based driving is an efficient command mechanism, it sometimes fails to provide sufficient contextual cues for good situational awareness. One remedy for this is to use sensor-derived maps, which provide reference to environmental features and explored regions[3].

5.3 Multiple sensors

Figure 9 demonstrates the value of having multiple sensors for remote driving. In this test, which was conducted just prior to a storm, AATVIL was driven towards a building surrounded by small shrubs. Due to the harsh lighting conditions, vegetation was difficult to discern in color and disparity images (Figure 6). With the multi-spectral camera, however, shrubs surrounding the building were easy to identify (Figure 9).



Figure 9: *Top*, AATVIL approaching a building bordered by small shrubs; *bottom*, Image mode display sequence (multispectral camera).

6 Future Work

Although both rate and image-waypoint control are efficient command mechanisms, neither may provide sufficient contextual cues for good situational awareness. Maps can remedy this by providing reference to environmental features, explored regions and traversed path. Thus, previous versions of PdaDriver included a mode for map-based waypoint driving.

The current PdaDriver, however, does not use maps. This is partially due to AATVIL's limited map building capability, but also because we considered the field test environment to be benign (uncluttered, few dangerous obstacles, etc). Yet, even in such an environment, we found that having a map would be useful. For example, some operators had difficulty judging depth and spotting nearby obstacles in the image displays. Thus, there is a clear need to incorporate sensor-based map displays, if not a complete map-based waypoint driving mode.

An additional improvement to PdaDriver would be to develop a "pendant" mode that supports direct control of vehicle actuators and display of vehicle status (health, pose, etc.). Such a mode would greatly improve vehicle field deployment, particularly manual egress/regress and vehicle check-out. Pendant mode would also be useful for quickly activating and deactivating robot controller modules, thus aiding fault detection and isolation.

Finally, although we developed PdaDriver for remote driving, its design is well-suited for other tele-

operation functions. For example, PdaDriver could easily be extended for high-level tasking: visual servo target designation, payload deployment, etc. In addition, PdaDriver has significant potential as a highly portable, field control unit. Specifically, we believe the interface could be used with little modification to operate almost any type of computer controlled device, such as military training targets, remote sensors, UAV's, etc.

Acknowledgments

We would like to thank James McKenna, Matt Morgenthaler, and Jeremy Myrtle for supporting AATVIL integration. This work was partially funded by grants from the DARPA ITO Mobile Autonomous Robot Software (MARS) program and SAIC.

References

- [1] T. Fong et al., "Novel interfaces for remote driving: gesture, haptic, and PDA", In Proceedings of the SPIE Telemanipulator and Telepresence Technology, 2000.
- [2] T. Fong, Collaborative control: a robot-centric model for vehicle teleoperation, Technical Report CMU-RI-TR-01-34, Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, 2001.
- [3] T. Fong et al., "A personal user interface for collaborative human-robot exploration", In Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space, 2001.
- [4] Fujitsu develops mobile phone-controlled robot for the home, Press Release, 7 October 2002, Fujitsu Laboratories, Inc., 2002.
- [5] H. Hüttenrauch and M. Norman, "PocketCERO – mobile interfaces for service robots", In Proceedings of the Mobile HCI, International Workshop on Human Computer Interaction with Mobile Devices, 2001.
- [6] J. Kay, STRIPE: Remote driving using limited image data, Technical Report CMU-CS-97-100, Ph.D. dissertation, Computer Science, Carnegie Mellon University, 1997.
- [7] W. Lu, J. Castellanes, and O. Rodrigues, Remote robot control using a Personal Digital Assistant, B.A. thesis, Computer Science, Ryerson Polytechnic University, 2001.
- [8] D. Perzanowski et al. "Towards seamless integration in a multi-modal interface", In Proceedings of the Workshop on Interactive Robot Entertainment, 2000.
- [9] P. Rybski et al., "System architecture for versatile autonomous and teleoperated control of multiple miniature robots", In Proceedings of the IEEE International Conference on Robotics and Automation, 2001.
- [10] Mobile Autonomous Robot Software (MARS) Self-Composing Adaptive Programming Environment, Final Report, SAIC, Littleton, CO, 2002.
- [11] Simple Communications Library (SCL), Fourth Planet, Inc., Los Altos, CA, 2000.
- [12] M. Skubic et al., "Extracting navigation states from a hand-drawn map", In Proceedings of the IEEE International Conference on Robotics and Automation, 2001.
- [13] J. Suomela and A. Halme, "Novel interactive control interface for centaur-like service robot", In Proceedings of the International Federation of Automatic Control, 2002.
- [14] R. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision", In Proceedings of the Computer Vision and Pattern Recognition, 1986.