Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 52 (2005) 160–179
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Interactive control for Internet-based mobile robot teleoperation

Meng Wang *, James N.K. Liu

Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

Received 9 September 2004; received in revised form 24 March 2005; accepted 7 April 2005
Available online 23 May 2005

Abstract

Internet telerobotics has emerged in recent decade with direct control and supervisory control as the main teleoperation paradigms. Both paradigms, however, are difficult to use on applications operating in the unknown and dynamic real world, while they do not provide adequate feeling of interaction or a human-friendly control interface to human operator. This paper proposes a novel interactive control (i.e., active supervisory control) paradigm: telecommanding, which is used for Internet-based wheeled robot teleoperation. Telecommanding involves two parts: basic telecommanding using joystick commands, and advanced telecommanding using linguistic commands. Each joystick or linguistic command is designed to perform an independent task and is defined with multiple events (non-time action references), and the corresponding response functions. This event-driven mechanism enables the robot to deliberately respond to expected events while to reactively respond to unexpected events. Assisted by up-to-date media streaming technologies, telecommanding can help a novice operator to easily control an Internet robot navigating in an unknown and dynamic real world. Experiments, including an Internet-based teleoperation test over 1500 km from Beijing to Hong Kong, demonstrate the promising performance.

Keywords: Internet telerobotics; Interactive control; Mobile robot navigation; Supervisory control

1. Introduction

Robots can now not only make basic motions [19] but also can closely interact with people [1–15]. Internet robots can provide many different remote services with potential applications in many areas: consumer home pet services, entertainment, telemedicine, distance learning, and the sharing of laboratory resources, as well as industry automation, military, and security applications [22]. On the other hand, the Internet also entails a number of limitation and difficulties, such as restricted bandwidth, arbitrarily large transmission delays, delay jitter, and packet lost or error, all of which influence the performance of Internet-based telerobotics systems [21,22]. Naturally one new field, Internet Telerobotics, is emerging in the recent decade [16].

* Corresponding author. Tel.: +852 27667312; fax: +852 27740842.
E-mail addresses: csmwang@comp.polyu.edu.hk (M. Wang), csnkliu@comp.polyu.edu.hk (J.N.K. Liu).

0921-8890/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2005.04.001

There are two generations of Internet robots. The 1994 Mercury project [2] was one of the earliest implementations of telerobotics over the Internet, with Australia's Telerobot [1] coming online at almost the same time. Since then, about 40 such systems have been put on line by research teams around the world [46]. The first-generation of Internet robots were mainly based on robotic arms or simple mobile robots directly controlled by human operators [1,2,5]. These Internet robots operate within a well-structured environment with little uncertainty, and have no local intelligence. In contrast, research into second-generation Internet robots has recently begun to focus on autonomous mobile robots that navigate in a dynamic and uncertain environment [3,4,37]. The key features of this generation are that the robots are autonomous and reactive, which enables them to navigate and cope with uncertainty in the real world. The main focus in building this generation of Internet robots is supervisory control [20].

Existing online robots are of two types: mobile manipulators with haptic or force feedback [1,2,10–13,48], and mobile vehicles used for navigation [3–7,44,47]. Because manipulated robots and wheeled robots have different characteristics, in the context of Internet-based teleoperation, they call for different control paradigms. The manipulated robots are often located in a limited or known workspace. Direct control is the popular control paradigm for Internet-based manipulated robot teleoperation. To alleviate the problem of uncertain time delay, three approaches [22] are often used in such systems: the predictive aiding approach, the simulating, and planning display approach [12,17,18], and the event-based approach [10,11,15].

Our focus is on the field of wheeled robot teleoperation. In this area, some Internet-based telerobotic systems have used direct control [5,7,45]. The typical example is the KhepOnTheWeb system [5], in which web users, via clickable images fed back from a camera, are able to control the robot's movements as it moves within a small wooden maze. Obviously, the direct control is not suitable for Internet-based mobile robot teleoperation because of the high latency derived from the Internet, such as restricted bandwidth, uncertain time delay, packet lost or error, and so on.

It is more common for Internet-based mobile robot teleoperation to use supervisory control [3,4,37]. In this case, problems derived from the Internet are alleviated by giving the robot more local intelligence. Unfortunately, most such systems [3,4] lack adequate interaction between human operator and robot. We refer to this type of control paradigm as *passive supervisory control*. Passive supervisory control is inadequate in four ways: (1) the control interface is only able to provide single or limited available control methods (e.g., using a mouse to click a map); (2) the human operator can issue only very high-level instructions to the robot, and it is difficult to obtain the robot's running status or information about the events the robot has encountered; (3) the robot has considerable autonomy but lacks the interaction with the human operator; (4) it is often needed to let the robot know some environmental knowledge in advance for path planning or self-localization, which causes that it is difficult to be applied in an unknown and highly dynamic environment. The typical examples are Xavier, an office-exploring robot at CMU [3], and the museum tour-guide robot RHINO and MINERVA [4]. They allow web users to take the goal control, but the robots must know some global environment knowledge in advance. The control mode used on the Mars lander [33–35] can also be categorized as passive supervisory control. The human operators on earth use a web-based tool to specify multiple waypoints as the navigation subgoals in 3D views of the landing site. These waypoints were generated from images obtained using stereo cameras on the lander.

One important characteristic between human and robot is interactivity. Simmon et al. summarized their lessons from the 5 year (December 1994–1999) public Xavier experiment as follows [3].

"Autonomy can help in reducing the bandwidth requirements for control but this introduces problems of its own, particularly in the area of interactivity. People seem to prefer 'hands on' control. . . . The only real negative impact of autonomy on web-based interaction is that commanding at a high-level is not as interactive as (conventional) teleoperation."

Saucy and Mondada also pointed out the significance of interactivity over the 1 year (May 1997–1998) KhepOnTheWeb system that was accessible to the public [5].

"Another problem is obviously the delay that prevents people from having a good interaction and from taking

interests in the site. That's one reason why users do not come back."

Researchers are attempting to add more interaction between humans and robots [36–42,48], such as behavior-programming control, supervised autonomy, shared control, cooperative control, collaborative control, and fitting autonomy. We refer to these control modes as *active supervisory control* or *interactive control*. Section 6 provides a detailed analysis and comparison of these control modes. The main deficiencies of these control modes are: (1) they lack a complete framework to process the commands that can be sent continuously from human operator; (2) they are difficult to evaluate the online running performance and provide the corresponding response actions; (3) their components are interdependent, which means that one poor component may cause multiple tasks fail; (4) the interfaces are not sufficiently human-friendly and they cannot provide a multi-modal control interface. In this paper, we attempt to address the problems faced by the existing passive supervisory control and interactive control methods by proposing a novel interactive control method, telecommanding, for Internet-based mobile robot teleoperation.

This paper is organized as follows. Section 2 presents the entire telecommanding framework, including basic telecommanding using joystick commands and advanced telecommanding using linguistic commands. Section 3 proposes the design method of linguistic command function. Section 4 introduces our experimental platform. Section 5 presents the simulation and real world experiments. Section 6 gives a summary of comparison between telecommanding and other approaches. Section 7 offers our conclusion.

2. An interactive teleoperation paradigm

The following sections discuss in detail a novel interactive teleoperation paradigm: telecommanding. Section 2.1 will describe the framework of telecommanding, which involves two different but complimentary teleoperation methods: basic telecommanding using joystick commands, and advanced telecommanding using linguistic commands. In Section 2.2, we will describe the design of joystick commands. In Section 2.3, we will describe the design of linguistic commands.

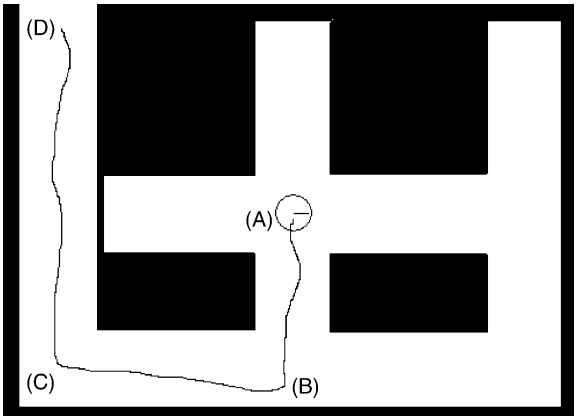

Fig. 1. The robot in a maze (A is the start and D is the goal).

2.1. The framework of telecommanding

Imagine a complex navigational task guided by a human. It may be, for example, to guide a bewildered person out of a maze (see Fig. 1) from the start A to the goal D. The human guide may use three possible methods. One is that the guide directly guides the person step by step. The second method is that the guide may simply give directions, just like someone giving instructions on how to reach the nearby post office: "Turn right and move forward 50 m, then turn right and go to the end. Next take a right, move forward 100 m. And there's the post office". The third method is to use a map and point out three waypoints B–D or only the goal D. We call the first method basic telecommanding (BT), both the second and third methods advanced telecommanding (AT).

Together, both BT and AT make up an interactive control paradigm: telecommanding, especially in terms of Internet-based mobile robot teleoperation. The framework of telecommanding is shown in Fig. 2. The control interface between human operator and robot is multimodal. For example, BT uses joystick commands that are issued via the computer keyboard or a real joystick device by remote operator. AT uses linguistic commands. The linguistic commands in the second method are often issued via an interactive command window of a graphical display interface. The linguistic commands in the third method are often issued via a computer mouse by clicking in the graphic window of a display interface. The display interface shows the visual feedback from a camera mounted on the robot,

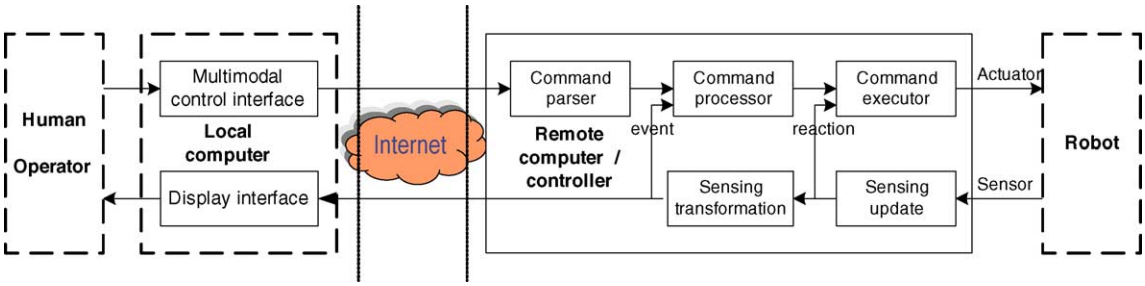

Fig. 2. The framework of telecommanding.

history, and current status, as well as visualized pose and obstacles. The Command Parser is responsible for parsing the joystick or linguistic commands from the local computer, then transferring them to the corresponding Command Processor for further command processing and execution. The function of the Sensing Update module is to capture the raw exteroception and proprioception sensing data from the robot's sensors. The modules of Command Executor, Sensing Update and the robot make up a reaction loop. This loop enables the robot to react rapidly to unexpected events derived from the dynamics of real world. The function of the Sensing Transformation module is to transform the raw sensing data into high-level data (e.g., total moving distance) to allow the detection of expected events. These expected events drive the Command Processor to autonomously make a deliberative plan and to respond to current situation encountered by the robot. The Command Processor, Command Executor, robot, Sensing Update, and Sensing Transformation modules constitute a deliberative loop. The large loop, including human operator, forms a complete telecommanding system.

BT and AT are exclusive. When human operator sends joystick commands, BT dominates the control privilege of the robot and discards all previous linguistic commands. Similarly, when the operator sends linguistic commands, AT dominates the control privilege. On the other hand, all linguistic commands can be sent continuously, and they are stored in an ordered command queue that applies a FIFO (first-in-first-out) policy.

In every robot control cycle (e.g., 100 ms in our robotic system), the deliberative loop executes an inference process to output the next set of low-level motor controls. The inference process I can be defined as a relationship between input space U and output space Y. The input space U is multi-dimensional, with each dimension μ_i corresponding to a particular input data mode derived from the sensing transformation module, e.g., distance to front obstacle, direction to the goal, total moving distance, distance to the goal. Similarly, the output space Y is multi-dimensional, with each dimension corresponding to a particular type of output, normally motor speed v and delta turn angle ω. This is expressed by:

$$I : U(u_1, u_2, \cdots, u_i, \cdots, u_n) \rightarrow \mathrm{Y}(v, \omega)$$

We define some terms in the following, which will be used in the designs of joystick commands and linguistic commands.

Definition 1 *(Event)*. Let e_i be a subspace of input space $U(u_1, u_2, \ldots, u_i, \ldots, u_n)$, $i \in \{1, 2, \ldots, m\}$, and $U = e_1 \cup e_2 \cup \ldots \cup e_i \ldots \cup e_m$, so e_i is called an event.

Definition 2 *(Event occurs)*. Let e_i be an event and x_t a input vector at the time t, $x_t \in U(u_1, u_2, \ldots, u_i, \ldots, u_n)$. If $x_t \in e_i$, so the event e_i occurs, otherwise the event e_i does not occur.

Definition 3 *(Response function)*. Let e_i be an event. When the event e_i occurs, the robot should output the response actions y_t according to a function $f_{e_i}(x_t, y_{t-1})$, where x_t is the input vector at the current time t, $x_t \in U(u_1, u_2, \ldots, u_i, \ldots, u_n)$, y_{t-1} is the output vector at the time $t-1$, $y_{t-1} \in Y(v, \omega)$, so $f_{e_i}(x_t, y_{t-1})$ is called *response function* of the event e_i.

For example, suppose u_1 is an input variable, denoting the distance between the robot and the front obstacle, so an event e_i can be defined as $e_i = \{(u_1, \ldots, u_n) | 0.5 \leq u_1 < 1\}$. If $x_t \in e_i$, the event e_i occurs,

whose physical mean is such that if the distance between the robot and the frontal obstacle is in a range of 0.5–1.0 m, the robot should not simply respond to the command from the human operator but should autonomously calculate the output actions in accordance with the response function $f_{e_i}(x_t, y_{t-1})$. For instance, a response function is simply defined as $v_t = f_{e_i}(x_t, y_{t-1}) = v_{t-1} - 100$, where v_{t-1} is the speed of the robot at the time $t-1$. This response function does not affect the turn angle of the robot.

2.2. Basic telecommanding

Basic telecommanding (BT) is in some respects similar to the direct control paradigm that we find in a car driven by a human. Like a driver using a steering wheel, human operator uses ⟨Left and Right⟩ joystick commands to turn the robot. Like the driver using the accelerator and brake, the operator uses ⟨Up and Down⟩ joystick commands to accelerate or decelerate, even to stop or to reverse the robot.

In fact, BT substantially differs from the way of a human driving a car. The latter is an example of conventional direct control, whose process has three important characteristics: (1) the driver can receive environmental information in real-time through human vision and through the car's equipment, and from this can simultaneously build a real world model; (2) the driver can respond immediately to any contingency and the driver's actions are immediately effective; (3) the car typically lacks autonomous intelligence and relies on the driver to handle unexpected events. As mentioned in Section 1, Internet-based teleoperation has to overcome many challenges, such as restricted bandwidth, uncertain time delay, and data lost or error. Given these, it is desirable for the human operator to send the remote control as few commands as possible. This is quite different from the situation of a human driver, who must continuously provide input about steering or acceleration. At the same time, the robot has an autonomous capability to respond some expected events as well as to react rapidly to contingencies, so that the human operator does not need to handle the control details.

In BT, joystick commands provide such capabilities to enable the human operator to send as few commands as possible while to equip the robot local intelligence. Since the robot will continue to execute a joystick command until otherwise instructed, the operator sends the joystick commands only if necessary. This greatly reduces the number of commands an operator must issue. In addition, the robot autonomously makes judgments about situations. If it becomes aware of some impending danger, for example, nearby obstacles, the robot autonomously decreases the speed to a reasonable value while turning toward a safer direction. If the danger is immediate, the robot stops. In such situations, the internal autonomous behavior of the robot dominates the control privilege. Potentially, the robot may not respond to the human's joystick commands until it thinks that the current danger has passed or unless the joystick command will remove it from harm's way.

Definition 4 *(Joystick command function)*. Let N be a joystick command, so the corresponding command execution function $f_N(x_t,y_{t-1})$ is called the joystick command function associated with the joystick command N, where x_t is the input vector at the current time t, $x_t \in U(u_1, u_2, \ldots, u_i, \ldots, u_n)$, and y_{t-1} the output vector at the time $t-1$, $y_{t-1} \in Y(v, \omega)$.

In the implementation of our telerobotic system, we define four joystick commands: UP, DOWN, LEFT, RIGHT as well as the corresponding joystick command functions as the Eqs. (1)–(4).

$$v_t = f_{\mathrm{UP}}(x_t, y_{t-1}) = \begin{cases} v_{t-1} + \Delta v, & \text{if } v_t < v_{\max} \\ v_{\max}, & \text{otherwise} \end{cases} \quad (1)$$

$$v_t = f_{\mathrm{DOWN}}(x_t, y_{t-1}) = \begin{cases} v_{t-1} - \Delta v, & \text{if } v_t > v_{\min} \\ v_{\min}, & \text{otherwise} \end{cases} \quad (2)$$

$$\omega_t = f_{\mathrm{LEFT}}(x_t, y_{t-1}) = \Delta\theta \quad (3)$$

$$\omega_t = f_{\mathrm{RIGHT}}(x_t, y_{t-1}) = -\Delta\theta \quad (4)$$

where v_t, ω_t are the output variables at the current time t, respectively, denoting the speed and delta turn angle of the robot. $v_{\max}$ and $v_{\min}$ are, respectively, the minimum and maximum bounds of the speed. Because our robot does not have rear sensors, we set $v_{\min} = 0$, meaning that the robot is not allowed to reverse. Δv is a constant parameter (mm/s), and $\Delta\theta$ is a constant parameter (degree/s). The joystick commands UP and DOWN affect the speed of the robot while LEFT and RIGHT affect only the robot's steering angle.

In addition, we need to define the corresponding joystick events and response functions associated with these four joystick commands. For example, we are able to define several joystick events associated with UP as $\{e_{U1}, \ldots, e_{Uk}\}$, and the corresponding joystick response functions as $\{f_{e_{U1}}, \ldots, f_{e_{Uk}}\}$, where $e_{Ui} \cap e_{Uj} = \emptyset$, $i, j \in \{1, 2, \ldots, k\}$, which guarantees that there is only one joystick associated with a joystick command that occurs.

$$e_{U2} = \{(\cdots, u_3, u_4, u_5, \cdots, u_n) | ((0 \le u_3 < 0.15) \vee (0 \le u_4 < 0.15)) \wedge (u_5 > 0)\},$$

$$e_{U1} = \{(\cdots, u_3, u_4, u_5, \cdots, u_n) | ((0.15 \le u_3 < 0.5) \vee (0.15 \le u_4 < 0.5)) \wedge (u_5 > 0)\},$$

where u_3 is the distance (m) to the front obstacle, u_4 is the distance to the lateral obstacle, u_5 is the current actual speed of the robot. Obviously, $e_{U1} \cap e_{U2} = \emptyset$. Correspondingly, we simply define two joystick response functions as the Eq. (5).

$$v_t = \begin{cases} f_{e_{U1}}(x_t, y_{t-1}) = \dfrac{\Delta v}{2}, & \text{if } x_t \in e_{U1} \\ f_{e_{U2}}(x_t, y_{t-1}) = 0, & \text{if } x_t \in e_{U2} \end{cases} \quad (5)$$

In the actual implementation of joystick response function associated with UP, we define such functions that enable the robot to autonomously decrease the speed to a reasonable value while to turn toward a safer direction.

In every robot control cycle, Algorithm 1 is called once. In this algorithm, we ignore the feedback of the running events but they should be displayed on the display interface of human operator's monitor.

2.3. Advanced telecommanding

By advanced telecommanding, human operator does not care about the low-level control details. As mentioned in Section 2.1, a person gives a stranger a series of high-level instructions to guide him to the nearby post office. The robot must follow these instructions ("linguistic commands") while at the same time autonomously handle unexpected events and avoiding any static or dynamic obstacles (e.g., humans) in its path. Therefore, AT can reduce the influence of the high latency of the Internet.

Algorithm 1: JoystickCommandProcessor( )

Input: $x_t(u_1, u_2, \ldots, u_i, \ldots, u_n)$, $y_{t-1}(v_{t-1}, \omega_{t-1})$
Output: $y_t(v_t, \omega_t)$
BEGIN:
Step 1. IF there is a new joystick command, THEN
To calculate $y_t = f_N(x_t, y_{t-1})$ according to the corresponding Eqs. (1)–(4);
ELSE
$v_t = v_{t-1}$; $\omega_t = 0$;
END IF
Step 2. Detect and respond the joystick events associated with joystick command UP:
IF $x_t \in e_{U1}$, THEN $v_t = f_{e_{U1}}(x_t, y_{t-1})$;
. . . . . .
IF $x_t \in e_{Uk}$, THEN $v_t = f_{e_{Uk}}(x_t, y_{t-1})$;
Step 3. Detect and respond to the joystick events associated with joystick commands DOWN, LEFT and RIGHT, similar to Step 2
Step 4. IF no events occur AND there is no joystick command, THEN
To maintain the current status of the robot;
ELSE
Output the low-level motor command $y_t(v_t, \omega_t)$;
END IF
END Algorithm 1

Robotics researchers are able to design any variety of linguistic commands and integrate them into this framework for adaptation to specific tasks. In our research examples, AT involves five types of linguistic command: MOVE, TURN, WANDER, GOTOEND, and COORDINATE. These commands are designed to realize specific tasks, respectively. The human operator can continuously input these commands from the interactive command window, or the special command COORDINATE via the mouse by clicking in the graphical window, without having to wait until previous linguistic command is finished. If the command is correct and there are no exceptional events, the robot may follow these commands to reach the goal. Otherwise, the robot enters the command exception handle module automatically.

Every linguistic command is stored in an ordered command queue that adopts the policy of FIFO (first-in-first-out). This command queue is a two dimensional array: *commandQueue[M][N]*, where

- m: denotes the m_{th} command. $m \in M = [0,+\infty)$;
- n: denotes the n_{th} parameter of the m_{th} command. $n \in N = \{0, 1, 2, 3, 4\}$;
- *commandQueue[m][0]*: the index number of this command type. e.g., MOVE_INDEX or COORDINATE_INDEX;
- *commandQueue[m][1]*, *commandQueue[m][2]*: two working parameters of this command, e.g., (x, y) coordinate. The parameters are set to adapt flexibly to the real world model;
- *commandQueue[m][3]*, *commandQueue[m][4]*: two performance evaluation parameters of this command. Using the two parameters, the robot can evaluate the performance (success or failure) of execution result of current command, in order to make a decision to enter either the command exception handle module or the next command execution module.

For the design of linguistic command, we define the following terms.

Definition 5 *(Target event)*. Let e_{T} be an event associated with a linguistic command N. If the event e_{T} occurs, the robot has reached the goal state. So e_{T} is called a *target event* associated with the linguistic command N.

Definition 6 *(Overrun event)*. Let e_{O} be an event associated with a linguistic command N, and e_{T} a target event associated with N. If the event e_{O} occurs and e_{T} does not occur, the robot has encountered an overrun exception. So e_{O} is called an overrun event associated with the linguistic command N.

Definition 7 *(Underrun event)*. Let e_{U} be an event associated with a linguistic command N, and e_{T} a target event associated with N. If both the event e_{U} and e_{T} occur, the robot has encountered an underrun exception. So e_{U} is called an underrun event associated with the linguistic command N.

Definition 8 *(Linguistic command function)*. Let N be a linguistic command, so the corresponding command execution function $f_N(x_t,y_{t-1})$ is called linguistic command function associated with the linguistic command N, where x_t is the input vector at the current time t, $x_t \in U(u_1, u_2, \ldots, u_i, \ldots, u_n)$, and y_{t-1} is the output vector at the last time $t-1$, $y_{t-1} \in Y(v, \omega)$.

Actually, each linguistic command can be defined as a seven-element tuple $\{N, e_{\text{T}}, e_{\text{O}}, e_{\text{U}}, f_{e_{\text{O}}}, f_{e_{\text{U}}}, f_N\}$. N is the definition of this linguistic command, involving its name and parameters. The target event e_{T} is determined by two working parameters (*commandQueue[m][1]*, *commandQueue[m][2]*) of the linguistic command. The underrun event e_{U} is determined by the first performance evaluation parameter *commandQueue[m][3]*. The overrun event e_{O} is determined by the second performance evaluation parameter *commandQueue[m][4]*. $f_{e_{\text{O}}}$ and $f_{e_{\text{U}}}$ are the corresponding overrun and underrun response functions, respectively.

We explain the physical mean of these events using a linguistic command GOTOEND. When the linguistic command GOTOEND is running and the actual moving distance of the robot already exceeds the expected maximum distance, the overrun event occurs. This means that the robot has met an incorrect command or exception (e.g., the goal is too far away or it is not reachable). When the command GOTOEND is finished due to satisfying the target event but the actual moving distance does not exceed the expected minimum distance, the underrun event occurs. This means that the robot has also encountered an incorrect command or exception (e.g., someone suddenly blocks the path). In two such situations, the robot should then enter the command exception handle module to execute the corresponding overrun or underrun response function.

For the purpose of discussing how to design a linguistic command in terms of $\{N, e_{\text{T}}, e_{\text{O}}, e_{\text{U}}, f_{e_{\text{O}}}, f_{e_{\text{U}}}, f_N\}$, we will make use of a linguistic command MOVE. The formal definition of the linguistic command MOVE in our telerobotic system is as follow:

MOVE(double Distance, double minDistanceScale = 0.5, double maxDistanceScale = 1.5).

If the human operator does not input *minDistanceScale* and *maxDistanceScale*, the default values are used. MOVE is a complex linguistic command. Its function is to enable the robot to reach a goal lying ahead of the current location, and to avoid any static or dynamic obstacles (e.g., box and human). MOVE is converted into the following style in the command queue for execution.

commandQueue(MOVE_INDEX, Distance, 0, minDistanceScale, maxDistanceScale).

The current location of the robot in the robot internal coordinate can be represented as a vector (x^0, y^0, φ^0), where φ^0 denotes the current absolute heading angle. The command goal location is (x^T, y^T, φ^T). The current location of the robot is (x', y', φ'). The expected minimum and maximum moving distance are, respectively, d_{min} and d_{max}. Therefore,

d_{min} = minDistanceScale × Distance;
d_{max} = maxDistanceScale × Distance;
$x^T = x^0 + \text{Distance} \times \cos(\varphi^0)$;
$y^T = y^0 + \text{Distance} \times \sin(\varphi^0)$;
$\varphi^T = \varphi^0$.

So a target event e_T can be defined as $e_T = \{(u_1, u_2, u_3, \cdots, u_n) | (u_1 < \lambda) \wedge (u_2 < \beta)\}$, where u_1 is the distance between current location of the robot and the goal and $u_1 = \sqrt{(x' - x^T)^2 + (y' - y^T)^2}$, u_2 is the angle difference between current heading angle of the robot and the heading angle of the goal, $u_2 = |\varphi' - \varphi^T|$, λ is a constant that denotes the minimum tolerance for u_1, β is a constant that denotes the minimum tolerance for u_2. An underrun event e_U can be defined as $e_U = \{(u_1, u_2, u_3, \ldots, u_n) | u_3 < d_{min}\}$. An overrun event e_O can be defined as $e_O = \{(u_1, u_2, u_3, \ldots, u_n) | u_3 > d_{max}\}$, where u_3 is the actual moving distance of the robot. The overrun and underrun response function can be simply designed to enable the robot to cancel all subsequent linguistic commands in the command queue while to stop the movement of the robot. A more sophisticated strategy is to enable the robot to reschedule the command queue. The linguistic command function is the most important element of this command. We discuss it in Section 3. In every robot control cycle, Algorithm 2 is called once.

Algorithm 2: LINGUISTICCOMMANDPROCESSOR( )

Input: $x_t(u_1, u_2, \ldots, u_i, \ldots, u_n)$, $y_{t-1}(v_{t-1}, \omega_{t-1})$
Output: $y_t(v_t, \omega_t)$
BEGIN:
Step 1. IF a target event e_T of current linguistic command occurs, THEN
  IF an underrun event e_U occurs, THEN
    $y_t = f_{e_U}(x_t, y_{t-1})$;
  ELSE
    IF the next linguistic command can be obtained from the command queue via FIFO, THEN
      To set the e_T, e_U, e_O of this command, and go to the Step 1 again;
    ELSE
      $v_t = 0; \omega_t = 0$; /* stop the robot */
    END IF
  END IF
ELSE
  IF an overrun event e_O occurs, THEN
    $y_t = f_{e_O}(x_t, y_{t-1})$;
  ELSE
    $y_t = f_N(x_t, y_{t-1})$;
  END IF
END IF
Step 2. Output the low-level motor command $y_t(v_t, \omega_t)$
END Algorithm 2

3. The design of linguistic command function

Each linguistic command should be designed to autonomously perform an independent task. More linguistic commands, such as light-seeking, door-crossing, wall-following, can be designed to perform more complex tasks. In this paper, the linguistic command MOVE has been discussed in Section 2.3. The definitions of other linguistic commands in our telerobotic system are as follows:

TURN(double deltaAngle)
GOTOEND(double minDistance = 0, double maxDistance = INFINITE)
WANDER(double desiredDistance, double minDistanceScale = 0.5, double maxDistanceScale = 1.5)
COORDINATE(double *x*, double *y*, double minDistanceScale = 0.5, double maxDistanceScale = 1.5).

TURN is a simple linguistic command, whose function is to enable the robot to rotate. GOTOEND is a complex linguistic command that enables the robot to reach the end of a routeway (e.g., corridor end) while avoiding any lateral obstacles. WANDER is an interesting linguistic command that enables the robot to avoid any obstacles and move towards the safer direction. COORDINATE is a complex linguistic command that enables the robot to go from the current location to the goal location while avoiding any obstacles.

As it happens, MOVE and COORDINATE share the same linguistic command function GoalReaching( ). We here discuss this function in detail as a design example. The main idea of GoalReaching( ) is to decompose the task into three behaviors (i.e.,

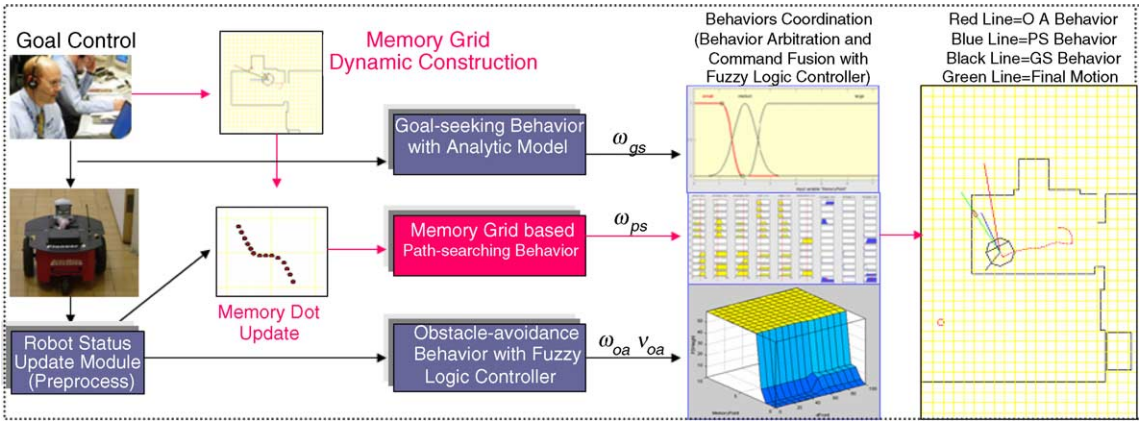

Fig. 3. The function GoalReaching decomposed into three behaviors (goal-seeking, obstacle-avoidance, and memory-grid based path searching) that coordinate with each another.

goal-seeking, obstacle-avoidance, and path-searching behaviors), and to reach the goal by coordinating them as shown in Fig. 3. Noted that the robot does not have a priori known environmental knowledge so as to find the safest path and avoid any obstacles, and it has to search online.

3.1. Preprocess

In the preprocess module, the input space, including the exteroception and proprioception sensing data, should be gathered and updated. If the input space is too large, the computational complexity should be controlled by reducing the number of dimensions. A common way to do this is to use a "situation clustering" approach [29] in which the complexity of the input space is reduced by introducing a limited number of intermediate variables that are meant to classify the different "perceptual situations" relevant to the robot's behavior. Possible intermediate variables are statements such as "facing_obstacle" or "distance_to_left_obstacle". These variables are then used by the fuzzy control rules.

3.2. Obstacle-avoidance

The obstacle-avoidance behavior [23,27], a local reactive behavior, must be effective in the face of any obstacles. Suppose the robot is mounted with five groups of proximity sensors (e.g., ultrasonic sensors): left, front-left, front, front-right, right. These sensors report the distances between the robot and the closest obstacle in each of the five sectors, namely $\{d_{\text{Left}}, d_{\text{Front-left}}, d_{\text{Front}}, d_{\text{Front–right}}, d_{\text{Right}}\}$. Each obstacle distance is represented by the three linguistic fuzzy sets {VERY NEAR, NEAR, FAR}. The output linguistic variables are the delta turn angular ω and moving speed v. Some typical turn or move rules are shown as follows.

- If d_{Front} is FAR and $d_{\text{Front-left}}$ is FAR and $d_{\text{Front-right}}$ is NEAR, then, ω is LEFT-TURN-SMALL;
- If $d_{\text{Front-left}}$ is VERYNEAR and $d_{\text{Front-right}}$ is not VERYNEAR, then, ω is RIGHT-TURN-SMALL;
- If d_{Front} is NEAR, then, v is SLOW;
- If d_{Front} is FAR, then, v is FAST.

The asymmetrical polynomial curves are used as membership functions (see Fig. 3). The fuzzy inference engine uses a Mamdani model. The defuzzification approach uses the centroid or center of gravity (COG) method [29].

3.3. Goal seeking

The goal-seeking behavior is a global behavior that enables the robot to seek the global goal location. Unlike other approach [24], we first assume that the goal-seeking behavior does not influence the speed of the robot, and contributes only to the rotational turn angle. Second, we use a very simple analytic model rather than a set of fuzzy logic navigational rules. We set the turn angle recommended by the goal-seeking behavior as the heading error between the current robot heading and goal direction as shown in Fig. 4. Thus, the value

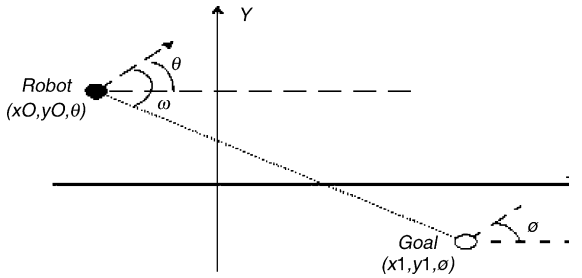

Fig. 4. The turn angle calculation of goal-seeking behavior.

domain of this turn angle is $(-180^\circ, 180^\circ]$. Positive and negative mean that the robot turns to the left and right, respectively.

3.4. Path searching

The path-searching behavior is a novel regional behavior. We have this design in order to help the robot get out of dead ends autonomously. The path-searching behavior works by, at first, constructing a memory grid. It then uses this memory grid to record the trajectory regions and the corresponding time consumed by the robot. These records are represented by a memory dot matrix. According to this memory dot matrix, it then calculates all risk factors of eight possible forward directions. To calculate the turn angle, it uses a "smallest-and-nearest" strategy. To determine the weight of path-searching behavior, it uses the total number of the memory dot of the current memory grid region, while combines with other input variables, e.g., distance to the obstacles. The details are discussed in [30].

3.5. Behavior arbitration and command fusion

Many existing behavior arbitration methods adopt High-Priority-Take-All or Winner-Take-All selection strategies but these strategies come with two disadvantages: their performance in certain situations may be inefficient, and the desirability of each behavior cannot vary from situation to situation. Other strategies have employed fusion methodologies in which each behavior is allowed to affect the final output based on the situational context [24–26]. One such strategy is context-dependent blending (CDB) [26] in which fuzzy logic is applied so that in a prevailing situation a decision can be made between behaviors.

Our behavior coordination strategy is similar to the CDB approach. It uses fuzzy context rules to express a behavior arbitration strategy. Each behavior is assigned a weighting factor, and these factors are adjusted dynamically according to the fuzzy weight rules. The weights determine the degree of influence of each behavior on the final motion command. For example, the input fuzzy variables of the fuzzy logic controller are (d_{left}, $d_{\text{Front-left}}$, d_{Front}, $d_{\text{Front-right}}$, d_{right}, K), and the output variables are (w_{oa}, w_{gs}, w_{ps}). K is the total number of memory dot of current robot location. w_{oa}, w_{gs}, w_{ps} are, respectively, the weights for obstacle-avoidance, goal-seeking, and path-searching behaviors. In turn, these variables are, respectively, represented by three linguistic fuzzy sets. The typical membership function, weight rules, and control curve can be seen in Fig. 3. Some typical weight rules are shown as follows.

- if K is LARGE, then w_{gs} is LOW and w_{ps} is HIGH.
- if K is SMALL, then w_{ps} is LOW.
- If d_{Front} is FAR and $d_{\text{Front-left}}$ is FAR and $d_{\text{Front-right}}$ is FAR and d_{left} is FAR and d_{right} is FAR, then is LOW.
- If d_{Front} is FAR and $d_{\text{Front-left}}$ is not VERYNEAR and $d_{\text{Front-right}}$ is not VERYNEAR and d_{left} is not VERY NEAR and d_{right} is not VERYNEAR and K is MEDIUM, then w_{gs} is MEDIUM and w_{ps} is MEDIUM.
- If d_{Front} is NEAR or $d_{\text{Front-left}}$ is NEAR or $d_{\text{Front-right}}$ is NEAR or d_{left} is NEAR or d_{right} is NEAR or K is MEDIUM, then w_{oa} is MEDIUM and w_{gs} is LOW and w_{ps} is LOW.

The strategy adopted in our approach is simpler than that of the CDB approach. The CDB approach uses a fuzzy preference combination to carry out command fusion but we first use a behavior arbitration module to compute all of the defuzzificated weight factors of all behaviors, and then carry out command fusion directly using these weight factors in the Eq. (6).

$$v = \frac{\sum v_i w_i}{\sum w_i}, \qquad \omega = \frac{\sum \omega_i w_i}{\sum w_i} \tag{6}$$

where v and ω are, respectively, the desired final speed and the delta turn angle values while v_i and ω_i are, respectively, the speed and turn angle preference values suggested by each individual behavior and w_i is

the defuzzificated weight factors that are output by the behavior arbitration module.

4. Experimental platform

Our research is tested on a mobile robot (vehicle) with eight forward ultrasonic sonars. The control commands transfer through radio Ethernet devices, and the video data is fedback through a set of A/V transmitter–receiver from a pan-tilt-zoom camera mounted on the robot deck.

4.1. PolyUiBot

Our project, PolyUiBot, creates a teleoperation platform that offers the Internet users (clients) the opportunity to remotely control a mobile robot in response to live streaming video captured by the camera mounted on the robot (see Fig. 5). The robot server connects the robot and camera over a wireless channel, obviating the problems associated with cables. The streaming server captures and encodes the real-time video from the camera on the robot under the instructions of the robot server. The compressed video images are streamed to transfer to the master client and slave clients. The robot server and streaming server services can both be distributed from the same computer to the Internet. The robot server interprets and activates the intelligent robot navigation algorithms, as well as the low-level motion control of the robot via the wireless channel. The remote control includes the pan-tilt-zoom commands of the camera on the robot.

Only one master client dominates the full control privilege to interact with the robot server through the Internet. The other slave clients can simultaneously watch the streaming video using the streaming player but they have no control privileges unless the master client hands over his privilege to another client. The slave clients include both the PC clients using a 33.6 Kbps modem to connect to the Internet through a telephone line, and the clients using mobile devices (e.g., PDA, mobile phones).

4.2. A hybrid approach using streaming technology and data visualization

To overcome information capturing, uncertain time delay, packet lost, low-bandwidth over the unreliable Internet, we provide a hybrid approach integrating the streaming technology with data visualization (e.g., sonar reading visualization, virtual trajectory display) to improve sensibility (i.e., virtual telepresence) as much as possible.

The most intuitive and informative way to improve sensibility for Internet-based robot teleoperation is via vision feedback. Researchers have approached this problem in a variety of ways. Early researchers used

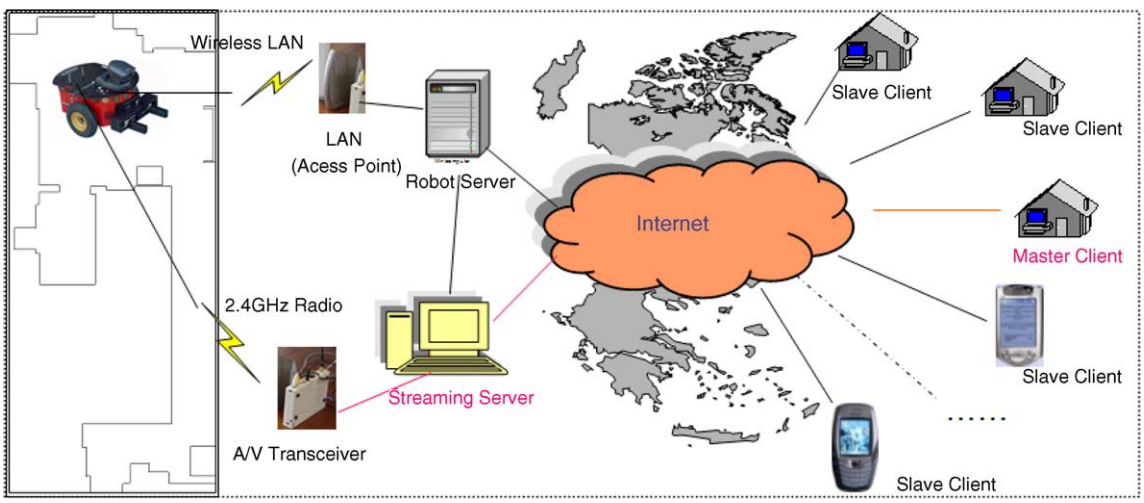

Fig. 5. PolyUiBot: the platform for Internet-based mobile robot teleoperation.

a picture transmission scheme (e.g., JPEG or GIF) or hybrid image-virtual reality [1–4]. The drawback of picture transmission is the very low frame rate and large time delay (over 10–20 s). A more serious problem is that Internet performance degrades, such as reductions in bandwidth, may cause service-stop errors. Researchers have now begun to use video conferencing systems instead of picture transmissions, but the crucial video coding algorithms of these systems are obsolete (e.g., H.261, H.263) [10]. The best current candidates for transferring multimedia perception information with the best quality of service (QoS) through the Internet are emerging streaming technologies such as MPEG4, RTP, and MMS [28]. We conduct the PolyUiBot platform based on the streaming technology, using a "pull" streaming transmission scheme and a constant bit rate (CBR) video encoding method, WMV9 as the video codec, and MMS as the streaming protocol [31].

The data visualization has to be developed to complement the video transmission technique. The streaming video provides global environment feedback, and truly improves the quality of services over the low-bandwidth, uncertain Internet transmission channel, producing a more stable system, higher image resolution, and smoother images [28]. On the other hand, the video transmission time delay is large (about 8 s through the campus Internet, and about 12 s through the 33.6 Kbps Internet connection over the telephone line). This time delay is caused mainly by the encoder and decoder buffers, which are used to guarantee the quality of service. So, we have developed the data visualization using sonar readings and dead reckoning data in order to obtain more timely perceptual feedback (less than 1 s to transfer in our campus Internet) and to improve the efficiency of the teleoperation.

4.3. Internet-based data transmission

Currently, unlike the other existing Internet telerobotics projects [1–7], we have not built our own web-based data transmission system. Instead, we use a Microsoft remote desktop connection service or Ultr@VNC service [32] for simplification of development workload, which is a reliable and convenient way for remote operators to connect with the robot server. The display and control interface is shown in Fig. 6. The human operator can obtain the global context information of the robot through the video feedback, and the local context information through the data visualization. This enables the operator to easily predict the next control command, and improves the teleoperation sensibility. On the other hand, the remote desktop connection service or VNC service is not an efficient teleoperation service since it consumes extra bandwidth for unnecessary data transmission. Moreover, through this service, the human operator actually dominates the control privilege of the robot server so that it is not enough for real public applications.

5. Experimental results

5.1. Teleoperation using basic telecommanding

5.1.1. Basic telecommanding simulation

We performed a basic telecommanding simulation as shown in Fig. 7. The human operator uses the joystick commands ⟨UP, DOWN, LEFT, RIGHT⟩ to control the robot to go from the start A to the goal F. The trajectory is displayed by the chain of black circles. The circle is drawn once by the program every 0.5 s. A denser concentration of circles (e.g., B to C) thus indicates that the robot is travelling more slowly. The relation of the robot speed and joystick command ⟨UP⟩ from the human operator is shown in Fig. 8. The robot begins to increase its speed from location A. Every time the robot receives an ⟨UP⟩ joystick command (see Fig. 8 2nd, 3rd, 4th, 14th, 17th, etc. ⟨UP⟩ command), the speed increases 100 mm/s. Once the speed reaches the predefined maximum bound (400 mm/s) (see Fig. 8 5th, 27th), the robot will not again increase its speed. When the robot is approaching obstacles, it may not respond to the ⟨UP⟩ command from the human operator and may autonomously decrease its speed to a reasonable value 100 mm/s (see Fig. 8 6th, 16th, 20th). When the robot is very close to obstacles, its speed is autonomously set to 0 mm/s and it does not respond to the next ⟨UP⟩ command (see Fig. 8 7–13th). When the robot is brought out of danger by ⟨LEFT⟩ or ⟨RIGHT⟩ commands from the human operator, it once again responds to the ⟨UP⟩ command (see Fig. 8 14th, 17th, 21th, 24th). During this experiment, we define the joystick events and corresponding joystick response functions simply, for simplicity of the implementation.

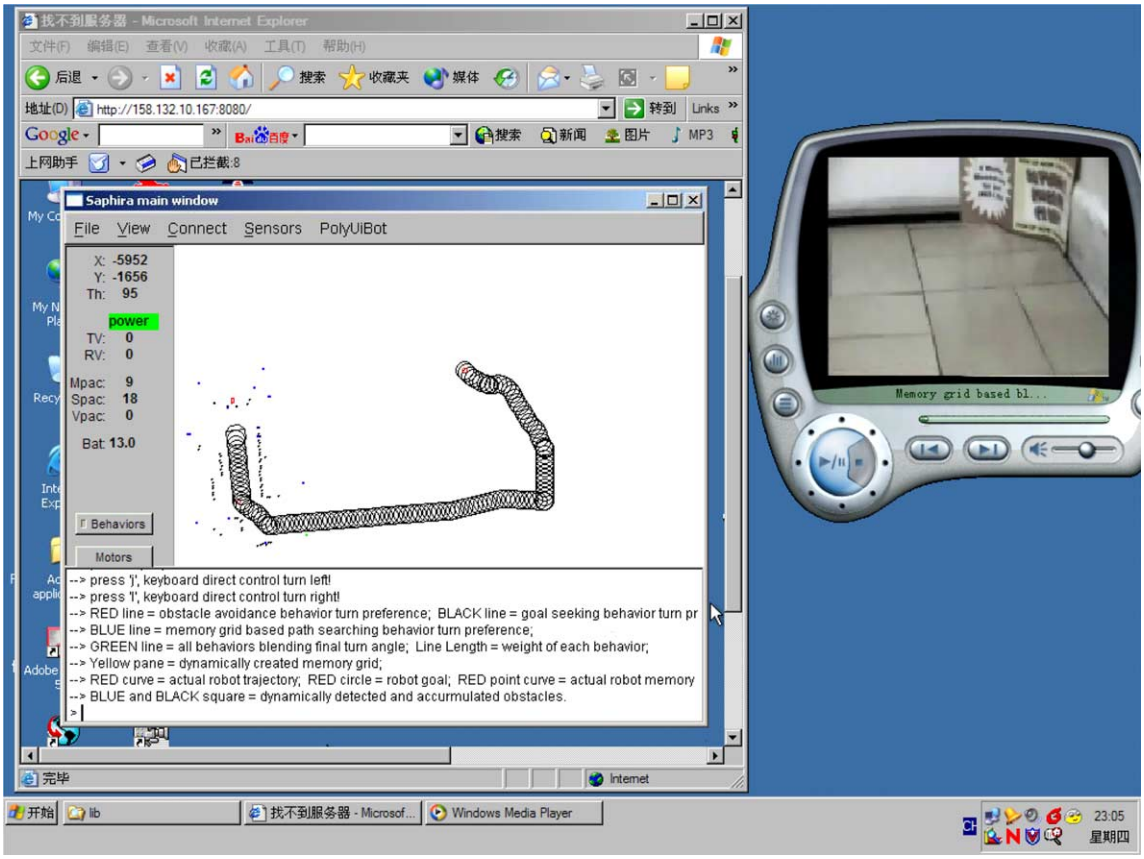

Fig. 6. The display and control interface: web-based data transmission via VNC service using IE, and streaming technology based video feedback.

5.1.2. Basic telecommanding in a corridor of real world

We conducted a real world experiment (in our department corridor, see Fig. 9) in which a remote human operator controls a real robot through the campus Internet using basic telecommanding control. Based on the experimental platform and control interface discussed in Section 4, it is convenient for the remote operator to control the robot using joystick commands. The operator does not need any robotic expertise for this teleoperation. It is just doing like playing a game using the keyboard's ⟨Up, Down, Left, Right⟩ keys. The velocity of the robot is bounded by a maximum speed 400 mm/s and the robot possesses autonomous intelligence, e.g., it autonomously decelerates and moves toward another safer direction if it is approaching an obstacle and stops if the danger is immediate (e.g., someone suddenly blocks the path). As a result, during the teleoperation experiment, the robot was able to avoid collisions with obstacles (e.g., walls, desks, boxes, and humans), even though at the time we purposely disconnected the network cable in order to let the Internet connection lost for a period.

5.2. Teleoperation using advanced telecommanding

5.2.1. Advanced telecommanding simulation

The purpose of this experiment is to demonstrate how to control a robot so that it can navigate in a complex and unknown space using advanced telecommanding for Internet-based teleoperation. As shown in Fig. 10(a), the remote operator continuously sends a series of linguistic commands to perform the guide task. First move forward (MOVE) to the location B; then, take a right turn 45° (TURN) and go to the end

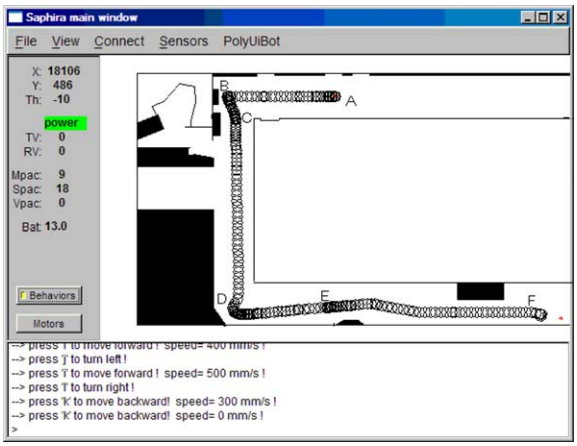

Fig. 7. Basic telecommanding simulation using joystick commands (UP, DOWN, LEFT, RIGHT) from the start A to the goal F. The trajectory is displayed by the chain of black circles.

C (GOTOEND); next turn right 90° (TURN) and go to the end D (GOTOEND); at this time turn a right angle 45° (TURN) and move forward (MOVE) to the location E; finally turn right 45° (TURN) and go to the end (GOTOEND), that is, the final goal F. During the experiment, the subsequent commands are stored in a command queue and allowed to execute only once the previous command is finished successfully. This test is successful without any overrun or underrun events.

We obtain a new linguistic command GOTO_ROOM_F when all linguistic commands in the command queue of the test are stored in the robotic system. We perform this learned linguistic command GOTO_ROOM_F from the start A again as shown in Fig. 10 (b). Because the sensing data is not identical with the last test, the trajectory of the robot is a little different from the last one. When the command GOTOEND( ) is running from location C, the target event does not occur at location D, but occurs instead at location E. The subsequent MOVE(1000) is wrong because

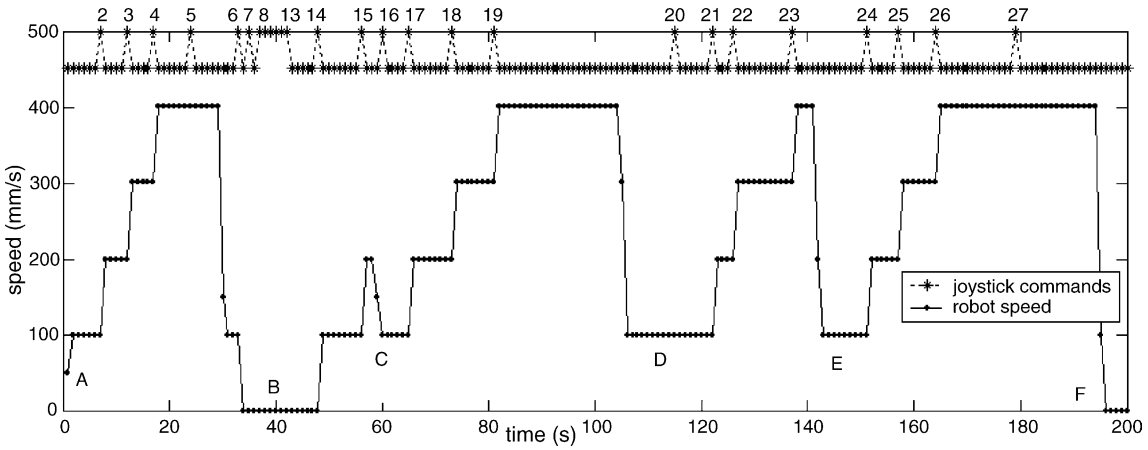

Fig. 8. The relation of robot speed and joystick command ⟨UP⟩ from human operator, which are recorded during the simulation from A to F. Each peak in the curve (*) represents an ⟨UP⟩ command. There are 27 ⟨UP⟩ commands in total. (A–F) correspond to the locations in Fig. 7.

Fig. 9. Basic telecommanding for use in Internet-based teleoperation. The robot moves in our department corridor under remote control. The corridor trajectory is about 45 m long and contains two corners and a number of obstacles.

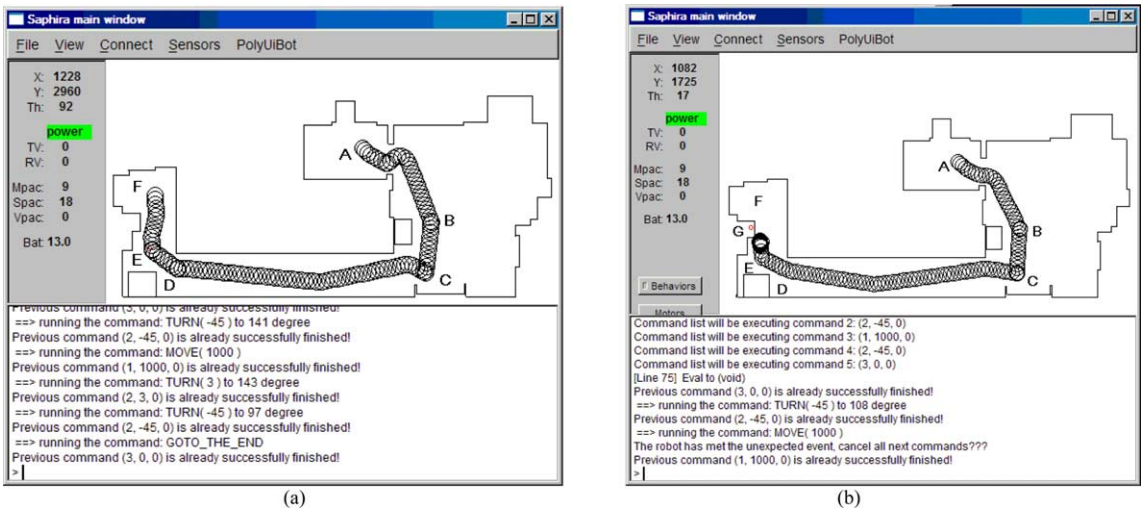

Fig. 10. Advanced telecommanding simulation using linguistic commands from the start A to the goal F. (a) GOTO_ROOM_F = {MOVE(3000),TURN(−45),GOTOEND( ),TURN(−90), GOTOEND( ),TURN(−45),MOVE(1000),TURN(−45),GOTOEND( )}; (b) learned command GOTO_ROOM_F is executed. An overrun event occurs when MOVE(1000) is running at location E.

it causes the robot to move toward location G, which is not reachable. This leads to an overrun event when the robot tries to move around to reach the location G. This activates the corresponding command exception handle module. In our telerobotic system, the robot is simply stopped autonomously so that the human can send further commands.

These linguistic commands greatly reduce the length and complexity of the command list, making it suitable for Internet-based teleoperation. For example, the robot is expected to move forward three meters from the start A to the location B. But a corner blocks its path (see Fig. 10). Our MOVE command provides a convenient way to achieve the goal. The operator only needs to send MOVE(3000) rather than to send a lot of low-level commands to avoid the obstacle. These linguistic commands are useful particularly in the dynamic real world.

5.2.2. Advanced telecommanding in a hall of real world

We performed an Internet-based teleoperation using advanced telecommanding to navigate a robot in a complex house (see Fig. 11). The remote operator observes, through the streaming video feedback, a global environment that is a priori unknown. The perceptual data visualization provides more timely local information. The remote operator sends a series of linguistic commands to let the robot move to the end of the path, return to the cross, then turn and go to the final goal (see Fig. 11). Because the robot is highly autonomous, the maximum velocity of the robot is set at 200 mm/s. The experimental task took about 100 s without any collisions, and the trajectory is about 13 m long, making the average velocity 13,000/100 = 130 mm/s. We ran the same Internet-based teleoperation test using direct control without robot intelligence. The task took over

Fig. 11. Advanced telecommanding in real world for Internet-based teleoperation The remote operator sends a series of linguistic commands: {GOTOEND(),TURN(180), GOTOEND(),TURN(40), MOVE(2000),TURN(90),MOVE(1000),TURN(50),MOVE(3000)}.

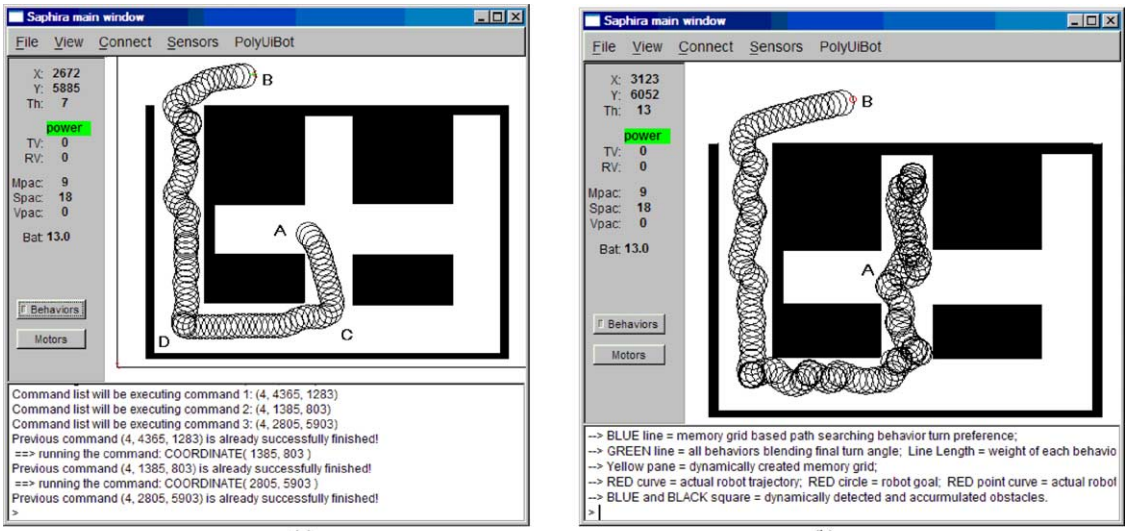

Fig. 12. The goal-oriented navigation in a maze from the start A to the goal B. (a) Three COORDINATE command (A−>C−>D−>B) by the use of the command queue; (b) One COORDINATE command (A−>B): the robot autonomously makes a decision by coordinating three behaviors: memory-grid based path searching, obstacle-avoidance, and goal seeking behaviors.

300 s and on several occasions the robot collided with walls or doors because of the large time delay derived from the video feedback.

5.3. The goal-oriented navigation in a maze

The purpose of this experiment is to demonstrate that the COORDINATE command enables the robot to have local intelligence to perform the complex navigational task in an unknown environment. As shown in Fig. 12, the robot does not have a priori knowledge of the map of the maze. It is required to go from the start A to the goal B. By clicking a mouse, the human operator is able to send the COORDINATE linguistic command. The simplest way is to continuously give out three COORDINATE commands (see Fig. 12 (a)). This allows the robot to pass by the waypoints C and D from the start A to the goal B, thereby avoiding the dead end. We propose a memory grid technique and a novel autonomous navigation strategy coordinated by three behaviors: memory-grid-based path searching, obstacle-avoidance, and goal seeking behaviors. The details can be found in [30]. By simply applying a COORDINATE command, the robot can autonomously search the correct path and avoid the dead end trap. The experimental results are encouraging. The robot can autonomously find the correct path out of the maze (see Fig. 12 (b)).

5.4. Robot teleoperation over a long distance

We tested the telecommanding teleoperation paradigm through the Internet over a distance from Beijing to Hong Kong (over 1500 km). A remote human operator (located in Beijing) connects with the robot server (located at our department in Hong Kong) through the VNC service, and observes the robot's surroundings (our department corridor) through streaming video (50 Kbps). Combining the graphical control interface and local perceptual data visualization, the remote operator can determine and send the telecommanding commands. The experiment has tested the joystick commands and all linguistic commands discussed in Section 2. The COORDINATE command is sent via a mouse click. The operator has no robotic expertise and he is told the teleoperation commands only at the beginning of the test. The successful test demonstrates that the telecommanding teleoperation paradigm is interactive, effective, and easy to use. The robot is also able to handle unexpected events and avoid collisions by making use of its autonomous local intelligence. The PolyUiBot project was publicly demonstrated and

Fig. 13. The robot is navigating by telecommanding in the Hong Kong Convention and Exhibition Center. Some audiences think the robot is "alive" and they are interested in blocking its moving path.

was well received at the Hong Kong Convention and Exhibition Center in April, 2004. Here are some video clips (see Fig. 13).

5.5. Performance and stability analysis

As shown in our real world tests, basic telecommanding (BT) is safe and reliable even in a crowded exhibition center. BT gives the human operator a strong feeling of interaction with the robot and it is easy to use. On the other hand, the human operator does have to spend more effort on the control details if BT is used in the highly dynamic environment. It should also be noted that BT is not suitable for carrying out some more skilful tasks (e.g., finding and entering a door located in the lateral wall of the corridor) if an uncertain or long time delay exists (e.g., through the Internet or the space).

Advanced telecommanding (AT) compensates for the disadvantages of BT and, further, it can evolve and obtain more high-level linguistic commands by learning the linguistic commands queue from the human operator. AT is modular, and each linguistic command can be designed and used independently. This means that one poor linguistic command may not affect the performance of others. The AT to guiding robots is familiar and easy to accept and therefore it does not require expertise. AT is suitable for the use in environments affected by uncertain or long time delays. The linguistic commands it sends are as few as possible yet can guarantee both the detection of exceptions and feedback. The disadvantages of AT are that a linguistic command function involves quite complicated design and that we currently lack an explicit standard to determine a command schedule as well as to define exception detection and responses.

The stability of Internet-based telerobotic system is affected by the uncertain time delay that results in the loss of synchronization of time-based action references. In the telerobotic system by telecommanding, the predefined events are non-time-based action references that are independent of this uncertain time delay. These events drive the robot to output actions in accordance with predefined response functions. The literature [43] has previously proven the theoretical stability of non-time referenced Internet-based telerobotic systems, and now it has been demonstrated via our tests in simulation and in the real world.

6. Summary of comparison with other approaches

It is obvious that the direct control is unsatisfactory for use in Internet-based mobile robot teleoperation because of the high latency derived from the Internet as discussed in Section 1. Passive supervisory control is unsatisfactory mainly in that it fails to provide adequate human–robot interactivity. Table 1 provides a comparison of these approaches with our own telecommanding.

Researchers are also attempting to add more human–robot interaction in the form of behavior-programming control, fitting autonomy, supervised autonomy, shared control, cooperative control, collaborative control, and so on [36–42,48]. We refer to these strategies as active supervisory control or interactive control. Chung et al. [36] propose a control strategy consisting of three major parts: behaviors, planner, and coordinator. The coordinator produces a wake-up table that contains all behaviors which should be scheduled by a real-time robotic system. Each task action must be defined with three conditions: pre_activate, fire_condition, and post_activate. These conditions respectively denote a group of behaviors that must be activated at the corresponding time. This kind of control strategy is too complex and the running performance of the robotic system is difficult to evaluate. One poor

Table 1
Comparison of performance under various teleoperation paradigms

	Direct control	Passive supervisory control	Telecommanding (interactive control)
Command send	Continuous	One by one	Both
Command level	Low	High	High
Task efficiency	Low	High	High
Semi-autonomy	None	High	High
Stationary environment	Feasible	Feasible	Feasible
Dynamic environment	Dangerous	Feasible	Feasible
Internet connection lost	Dangerous	Safe	Safe
Complex task	Difficult	Feasible	Feasible
Human–robot interactivity	Good	Poor	Good
Real world applicability	Good	Poor	Good

behavior could cause the failure of multiple tasks. The human operator also finds it difficult to provide suggestions for action via the command parameters. Similar problems arise in the behavior-programming control mode proposed by Luo and Chen [37]. In behavior-programming control mode, the event derived from a motion assistant is used on the robot to select a behavior that is suitable in the encountered situation. Vieira et al. proposed a concept of fitting autonomy [48], which allows the mobile agents to adapt its high-level abstract plan to the exact environment it finds in remote places and to execute the adapted plan including the execution monitoring and error recovery. This concept is evaluated in the field of mobile manipulator teleoperation.

Cheng and Zelinsky propose a teleoperation paradigm: supervised autonomy [42]. This allows some qualitative instructions (e.g., Go Forward, Go Toward, Go Between, or Keep To) to be implemented using a vision-based approach. These instructions are slightly similar to our linguistic commands, but they lack the performance evaluation and it allows instructions to be sent only one at a time. More importantly, this paradigm does not have a complete framework for command processing and event response. The shared control in literature [40] is similar to our joystick command, but it is rather simple and lacks the corresponding events definition and response functions. Literature [39] used cooperative control to control an intelligent wheelchair. This allows, at certain times, both the robot and the human to become the supervisor. Three types of behaviors are defined: skill-based, rule-based, and knowledge-based behaviors. The collaborative control in [41] is a model-based on human–robot dialogue. Both the cooperative control and collaborative control are difficult to apply in Internet-based mobile robot teleoperation because the Internet causes uncertain time delays and the robot cannot obtain timely suggestions from the human operator.

7. Conclusion

This paper proposes a novel interactive control: telecommanding, which involves two parts: basic telecommanding using joystick commands, and advanced telecommanding using linguistic commands. The commands are designed to perform different independent tasks. Each joystick or linguistic command is defined with multiple events (non-time action references) and the corresponding response functions. Some events (e.g., overrun event or underrun event) are used to evaluate the performance of the task when the robot is executing a linguistic command. This event-driven mechanism enables the robot to deliberately respond to expected events while the robot can reactively respond to unexpected events. Telecommanding ensures the safety of Internet robot that is navigating in an unknown and highly dynamic real world, and alleviates the problems of arbitrary network delays and restricted bandwidth. Moreover, it eases the workload of the human operator, reduces the operation sequence and its complexity in the command queue, and improves the interactivity and reliability of Internet telerobotics. The experiments have demonstrated the promising performance and the advantages of this paradigm over direct control and passive supervisory control.

Acknowledgements

The authors would like to acknowledge the partial support of the Hong Kong Polytechnic University via RGC grant B-Q515 and departmental grant H-Z87.

References

[1] K. Taylor, J. Trevelyan, Australia's telerobot on the web, in: International Symposium on Industrial Robots, 1995, pp. 39–44.
[2] K. Goldberg, S. Gentner, et al., The Mercury project: a feasibility study for Internet robots, IEEE Rob. Autom. Mag. 7 (March (1)) (2000) 35–40.
[3] R. Simmons, Fernandez, et al., Lessons learned from Xavier, IEEE Rob. Autom. Mag. 7 (2) (2000) 33–39.
[4] S. Thrun, M. Bennewitz, et al., MINERVA: a second-generation museum tour-guide robot, IEEE Int. Conf. Rob. Autom. 3 (1999) 1999–2005.
[5] P. Saucy, F. Mondada, KhepOnTheWeb: open access to a mobile robot on the Internet, IEEE Rob. Autom. Mag. 7 (March (1)) (2000) 41–47.
[6] R. Siegwart, P. Saucy, Interacting Mobile Robots on the web, in: ICRA'99, Detroit, MI, USA, May 1999.
[7] H. Huosheng, Y. Lixiang, et al., Internet-based robotic systems for teleoperation, Int. J. Assembly Autom. 21 (2) (2001) 1–10.
[8] X. Wang, M. Moallem, R.V. Patel, An Internet-based distributed multiple-telerobot system, IEEE Trans. SMC, Part A 3 (September (5)) (2003) 627–634.
[9] D. Schulz, W. Burgard, et al., Web interfaces for mobile robots in public places, IEEE Rob. Autom. Mag. 7 (March (1)) (2000) 48–56.
[10] I. Elhajj, N. Xi, et al., Supermedia-enhanced Internet-based telerobotics, Proc. IEEE 91 (3) (2003) 396–421.
[11] P. Li, W. Lu, Implementation of an event-based Internet robot teleoperation system, in: Fourth World Congress on Intelligent Control and Automation, vol. 2, 2002, pp. 1296–1300.
[12] M.R. Stein, The PumaPaint project, Autonom. Rob. 15 (2003) 255–265.
[13] G. Niemeyer, J.J.E. Slotine, Toward bilateral internet teleoperation, in: K. Goldberg, R. Siegwart (Eds.), An Introduction to Online Robots, The MIT Press, London, England, 2002, pp. 193–213.
[14] J. Cui, Z. Sun, P. Li, Visual technologies in shared control mode of robot teleoperatioin system, in: Fourth World Congress on Intelligent Control and Automation, vol. 4, 2002, pp. 3088–3092.
[15] P. Li, W. Lu, Implementation of an event-based Internet robot teleoperation system, in: Fourth World Congress on Intelligent Control and Automation, vol. 2, 2002, pp. 1296–1300.
[16] R. Siegwart, K. Goldberg, Robots on the web, IEEE Rob. Autom. Mag. 7 (March (1)) (2000) 4.
[17] C.P. Sayers, Fundamentals of online robots, in: K. Goldberg, R. Siegwart (Eds.), An Introduction to Online Robots, The MIT Press, Cambridge, London, 2002, pp. 3–16.
[18] C.P. Sayers, Remote Control Robotics, Springer-Verlag, New York, 1999.
[19] U. Nehmzow, Mobile robotics: a practical introduction, Springer-Verlag, London, 2000.
[20] T.B. Sheridan, Telerobotics, automation and human supervisory control, The MIT Press, London, England, 1992.
[21] K. Brady, T.J. Tarn, Handling latency in Internet-based teleoperation, in: K. Goldberg, R. Siegwart (Eds.), An Introduction to Online Robots, The MIT Press, London, England, 2002, pp. 171–192.
[22] R.C. Luo, K.L. Su, Networked intelligent robots through the Internet: issues and opportunities, IEEE Proc. 91 (3) (2003) 371–382.
[23] S. Thongchai, S. Suksakulchai, D.M. Wilkes, N. Sarkar, Sonar behavior-based fuzzy control for a mobile robot, in: IEEE Conference on Systems, Man, and Cybernetics, vol. 5, 2000, pp. 3532–3537.
[24] H. Seraji, A. Howard, Behavior-based robot navigation on challenging terrain: a fuzzy logic approach, IEEE Trans. Rob. Autom. 18 (June (3)) (2002) 308–321.
[25] Ye Cang, Wang Danwei, A novel behavior fusion method for the navigation of mobile robots, in: IEEE International Conference on SMC, vol. 5, 2000, pp. 3526–3531.
[26] A. Saffiotti, E.H. Ruspini, K. Konolige, Using fuzzy logic for mobile robot control, in: H.J. Zimmermann (Ed.), Practical Applications of Fuzzy Technologies, Academic Publishers, Norwell, USA, 1999, pp. 185–206.
[27] A. Saffiotti, Fuzzy logic in autonomous navigation, in: D. Driankov, A. Saffiotti (Eds.), Fuzzy Logic Techniques for Autonomous Vehicle Navigation, Physica-Verlag, Heidelberg, New York, 2000, pp. 3–22.
[28] M. Wang, J.N.K. Liu, PolyUiBot: sensibility improvement using streaming technology for internet telerobotics, WSEAS Trans. Comp. 3 (July (3)) (2004) 592–601.
[29] M. Wang, J.N.K. Liu, Autonomous robot navigation using fuzzy logic controller, in: IEEE Conference on Machine Learning and Cybernetics, Shanghai, China, August 2004, pp. 691–696.
[30] M. Wang, J.N.K. Liu, Online path searching for robot autonomous navigation, in: IEEE Conference on Robotics, Automation and Mechatronics (RAM), Singapore, December 2004, pp. 746–751.
[31] J.N.K. Liu, M. Wang, F. Bo, iBotGuard: an internet-based intelligent robot security system using invariant face recognition against intruder, IEEE Trans. SMC Part C 3 (February (1)) (2005) 97–105.
[32] http://ultravnc.sf.net/.
[33] R. Volpe, T. Estlin, et al., Enhanced Mars rover navigation techniques, in: IEEE Conference on Robotics and Automation, vol. 1, USA, 2000, pp. 926–931.
[34] C.R. Weisbin, G. Rodriguez, NASA robotics research for planetary surface exploration, IEEE Rob. Autom. Mag. 7 (December (4)) (2000) 25–34.
[35] P.G. Backes, K.S. Tso, J.S. Norris, G.K. Tharp, Internet-based ground operations for mars lander and rover missions,

in: K. Goldberg, R. Siegwart (Eds.), An Introduction to Online Robots, The MIT Press, Cambridge, England, 2002, pp. 227–240.
[36] J. Chung, B.S. Ryu, H.S. Yang, Integrated control architecture based on behavior and plan for mobile robot navigation, Robotica 16 (1998) 387–399.
[37] R.C. Luo, T.M. Chen, Development of a multi-behavior based mobile robot for remote supervisory control through the Internet, IEEE/ASME Trans. Mechatronics 5 (December (4)) (2000) 376–385.
[38] P.E. Rybski, S.A. Stoeter, Sharing control [multiple miniature robots], IEEE Rob. Autom. Mag. 9 (December (4)) (2002) 41–48.
[39] G. Bourhis, Y. Agostini, Man-machine cooperation for the control of an intelligent powered wheelchair, J. Int. Rob. Syst. 22 (1998) 269–287.
[40] I.S. Lin, F. Wallner, R. Dillmann, Interactive control and environment modelling for a mobile robot based on multisensor perceptions, Rob. Autonom. Syst. 18 (August (3)) (1996) 301–310.
[41] T. Fong, C. Thorpe, C. Baur, Robot, asker of questions, Rob. Autonom. Syst. 42 (March (3–4)) (2003) 235–243.
[42] G. Cheng, A. Zelinsky, Supervised autonomy: a framework for human–robot systems development, Autonom. Rob. 10 (2001) 251–266.
[43] N. Xi, T.J. Tarn, Stability analysis of non-time referenced Internet-based telerobotic systems, Rob. Autonom. Syst. 32 (August (2–3)) (2000) 173–178.
[44] V. Mut, J. Postigo, E. Slawinski, B. Kuchen, Bilateral teleoperation of mobile robots, Robotica 20 (2002) 213–221.
[45] K.H. Han, S. Kim, Y.J. Kim, J.H. Kim, Internet control architecture for Internet-based personal robot, Autonom. Rob. 10 (2001) 135–147.
[46] K. Goldberg, R. Siegwart, Introduction, in: K. Goldberg, R. Siegwart (Eds.), An Introduction to Online Robots, The MIT Press, Cambridge, England, 2002, pp. XV–XXI.
[47] A. Halme, J. Suomela, M. Savela, Applying telepresence and augmented reality to teleoperate field robots, Rob. Autonom. Syst. 26 (1999) 117–125.
[48] W.J. Vieira, L.M. Camarinha-Matos, L.O. Castolo, Fitting autonomy and mobile agents, in: IEEE Conference on Emerging Technologies and Factory Automation, vol. 2, Portugal, 2001, pp. 471–480.

Meng Wang received the B.Sc. degree from Xiamen University, Xiamen, China, in 1996, and the M.Sc. degree from the Chinese Academy of Space Technology (CAST), Beijing, in 1999. He is currently pursuing the Ph. D. degree at the Hong Kong Polytechnic University, Hong Kong.

From 1997 to 2003, he was with CAST, conducting a number of research and engineering projects. He is an expert in the area of satellite ground applications, majoring in signal processing, system automation, digital signal processor (DSP) embedding system, video compression and transmission, and database. He has served in several vital national projects for satellite ground stations, involving FY-2A, DFH-3, Resource no. 2, FY-2B, HY-1 satellites, as well as video wireless transmission system in the national plan. His current research interests are autonomous robot navigation, Internet telerobotics, image-based pattern recognition, video compression and transmission. He is a member of IEEE.

James Liu received the B.Sc. (Hons) and M.Phil. degrees in mathematics and computational modeling from Murdoch University, Perth, Australia, in 1982 and 1987 respectively. He received his Ph.D. degree in artificial intelligence from La Trobe University, Melbourne, Australia, in 1992. While working on his degree, he worked as a computer scientist at Defence Signal Directorate in Australia from 1988 till 1990. He joined the Aeronautical Research Laboratory (ARL) of Defence Science and Technology Organization in Australia as a research scientist in 1990. At ARL, he helped perform AI research in areas of human factors and mission enhancement.

Dr Liu joined the Hong Kong Polytechnic University, as a faculty member in 1994, and currently holds the position of Associate Professor in the Department of Computing. He has published technical papers on subjects in expert system verification, forecasting systems, pattern recognition, biometrics technology and e-Commerce applications. His current interests include telerobotics, intelligent business computing, multilingual system development, weather simulation and forecasting, data mining and Web-based information systems. He is a member of IEEE and AAAI.