Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Robotics and Autonomous Systems 42 (2003) 245–258

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

The Personal Rover Project: The comprehensive design of a domestic personal robot

Emily Falcone*, Rachel Gockley, Eric Porter, Illah Nourbakhsh

The Robotics Institute, Carnegie Mellon University, Newell-Simon Hall 3111, 5000 Forbes Ave., Pittsburgh, PA 15213, USA

Abstract

In this paper, we summarize an approach for the dissemination of robotics technologies. In a manner analogous to the personal computer movement of the early 1980s, we propose that a productive niche for robotic technologies is as a long-term creative outlet for human expression and discovery. To this end, this paper describes our ongoing efforts to design, prototype and test a low-cost, highly competent personal rover for the domestic environment.
© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Social robots; Educational robot; Human–robot interaction; Personal robot; Step climbing

1. Introduction

Robotics occupies a special place in the arena of interactive technologies. It combines sophisticated computation with rich sensory input in a physical embodiment that can exhibit tangible and expressive behavior in the physical world.

In this regard, a central question that occupies our research group pertains to the social niche of robotic artifacts in the company of the robotically uninitiated public-at-large: *What is an appropriate first role for intelligent human–robot interaction in the daily human environment*? The time is ripe to address this question. Robotic technologies are now sufficiently mature to enable interactive, competent robot artifacts to be created [4,10,18,22].

The study of human–robot interaction, while fruitful in recent years, shows great variation both in the duration of interaction and the roles played by human and robot participants. In cases where the human caregiver provides short-term, nurturing interaction to a robot, research has demonstrated the development of effective social relationships [5,12,21]. Anthropomorphic robot design can help prime such interaction experiments by providing immediately comprehensible social cues for the human subjects [6,17].

In contrast, our interest lies in long-term human–robot relationships, where a transient suspension of disbelief will prove less relevant than long-term social engagement and growth. Existing research in this area is often functional, producing an interactive robot that serves as an aide or caregiver [13,16,19]. The CERO figure is of particular interest due to its evaluation as a robot interface representative in an office environment over a period of several months.

Note that such long-term interaction experiments often revisit the robot morphology design question. Anthropomorphism can be detrimental, setting up long-term expectations of human-level intelligence or perception that cannot be met. Robots such as eMuu

* Corresponding author. Tel.: +1-412-268-6723; fax: +1-412-268-7350.
E-mail address: etf@andrew.cmu.edu (E. Falcone).

0921-8890/03/$ – see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0921-8890(02)00379-2

and Muu2 exemplify the same aesthetic principles of non-anthropomorphic expressiveness sought by our research group [3].

Most closely aligned to the present work are those projects in which the robot's role is to be a vessel for exploration and creativity. Billard's Robota series of educational robots provide rich learning experiences in robot programming [4]. Coppin's [8] Nomad rover serves as a telepresence vehicle for the public. Although the human–robot relationship is secondary, the robot nonetheless provides displaced perception and exploration, inspiring users with regard to both robotics and NASA exploration programs. Educational robotics kits such as LEGO Mindstorms [14] also provide inspiration regarding science and technology. Such kits provide, in the best case, an iconic programming interface. Without depending upon previous programming experience, this enables a child to guide the behavior of their robotic creation over the short term. Teaching by example and durative scheduling are aspects of robot expression that are not addressed by these kits.

Our aim is to develop a comprehensive example of long-term, social human–robot interaction. Our functional goal is to develop a robot that can enter a direct user relationship without the need for a facilitator (e.g. an educator) or a specially prepared environment (e.g. a classroom).

We propose that an appropriate strategy is to develop a robot functioning within the human domestic environment that serves as a creative and expressive tool rather than a productive appliance. Thus the goal of the Personal Rover Project is to design a capable robot suitable for children and adults who are not specialists in mechanical or electrical engineering. We hypothesize that the right robot will help to forge a community of creative robot enthusiasts and will harness their inventive potential. Such a *personal rover* is highly configurable by the end user: a physical artifact with the same degree of programmability as the early personal computer combined with far richer and more palpable sensory and effectory capabilities.

The challenge in the case of the personal rover is to ensure that there will exist viable user experience trajectories in which the robot becomes a member of the household rather than a forgotten toy relegated to the closet.

A User Experience Design study conducted with Emergent Design, Inc., fed several key constraints into the rover design process: the robot must have visual perceptual competence both so that navigation is simple and so that it can act as a videographer in the home; the rover must have the locomotory means to travel not only throughout the inside of a home but also to traverse steps to go outside so that it may explore the backyard, for example, finally, the interaction software must enable the non-roboticist to shape and schedule the activities of the rover over minutes, hours, days and weeks. In the following sections, we present corresponding details of the comprehensive design of the robot mechanics, teaching interface and scheduling interface.

2. Rover mechanics and control

2.1. Rover hardware

The rover's size and shape are born from practical constraints regarding the home environment together with the goal of emulating the aesthetics of the NASA exploratory rovers. Users should be able to easily manipulate the rover physically. Also, the rover must be small enough to navigate cramped spaces and large enough to traverse outdoor, grassy terrain and curbs.

The fabricated rover's physical dimensions are 18 in. × 12 in. × 24 in. (length, width, height). Four independently powered tires are joined laterally via a differential. Each front wheel is independently steered by a servomotor, enabling not only conventional Ackerman steering but also the selection of any center of rotation along the interior rear axle. Two omni-wheels behind the main chassis provide protection against falling backward during step climbing and also enable a differential-drive motion mode. The most unusual mechanical feature of the personal rover is the swinging boom, which is discussed below due to its critical role for step climbing.

The CMUcam vision system [20] is mounted atop a pan-tilt head unit at the top end of the swinging boom (Fig. 1). This vision sensor is the single most important perceptual input for the personal rover. Images are sufficient for basic robot competencies such as obstacle avoidance and navigation [1,2,11,23,24].

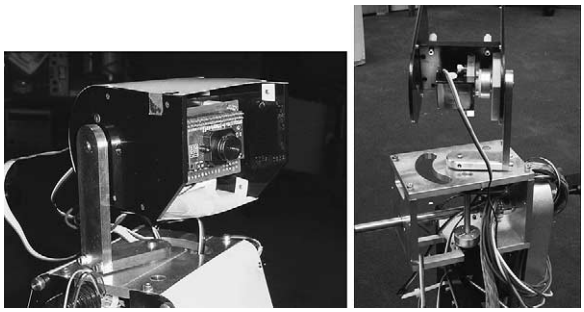

Fig. 1. A CMUcam is mounted in the rover's head, where it can pan and tilt.

But even more importantly images are an exciting data collection tool: the personal rover can act as a video and photo documentary producer. At the interaction design level, a robot that responds visually, and does so using fast pan-tilt control, communicates a compelling level of awareness [5].

A Compaq iPAQ on the rover provides 802.11 networking, communicates with the CMUcam, and sends motion commands to the Cerebellum microcontroller [7]. The iPAQ serves both as a wireless to serial bridge for networked communication and as a fast sensorimotor controller that can servo the rover's pan-tilt mechanism to physically follow an object being tracked by CMUcam. The Cerebellum controls the servo motors, reads infrared (IR) range finders, and provides four PIC-based daughter boards (one for each wheel) with speed commands. Based on quadrature encoders attached to the motors, the daughter boards use proportional integral derivative (PID) control to adjust the duty cycle and report current levels to the Cerebellum as feedback.

2.2. Low-level control

Command packets from the controlling computer to the rover can specify any combination of the following commands: speed, turn angle, boom position, camera pan and tilt angles, and finally all camera commands. Each single-threaded communication episode consists of one or more directives regarding the above degrees of freedom. The rover responds with a *state vector* packet containing rover velocity, encoder counts, wheel duty cycles, IR range readings, servo positions and boom position.

2.2.1. Encoders

The controlling computer calculates the rover's approximate position and angle by integrating the encoder values. Because the turning radius can be inferred from steering servo positions, only one wheel's encoder value is required for the calculations. With four encoders, the problem is over constrained, but this redundancy enables limited error handling and improves accuracy empirically. Encoder values that are kinematically inconsistent are discarded, then remaining values are averaged.

Position integration is performed classically, computing the distance the robot has moved on a circle of fixed radius. Given r, the positive radius, and α, the angle in radians, the rover has moved around the circle, we can calculate the new location of the rover with the following formulas:

$$x_1 = r\left[\cos(\theta_0 + \alpha - \tfrac{1}{2}\pi) + \cos(\theta_0 + \tfrac{1}{2}\pi) + x_0\right]$$

$$y_1 = r\left[\sin(\theta_0 + \alpha - \tfrac{1}{2}\pi) + \sin(\theta_0 + \tfrac{1}{2}\pi) + y_0\right]$$

$$\theta_1 = \theta_0 + \alpha$$

2.2.2. Motion control

Two simple movement functions, GoTo and TurnTo, use closed-loop control to translate and rotate the rover to new goal poses. While the rover is moving, a global x, y, and θ are continuously updated. We implement vision-relative motion control functions using the CMUcam tracking feedback loop executed on the iPAQ. The function called "landmark lateral" moves the rover to a specified offset relative to a visually tracked landmark, using the pan angle of the camera to keep the rover moving straight, and using the global coordinate frame to track the rover's overall position. We calculate the position of the landmark in the global coordinate frame by using the pan and tilt angles of the camera, together with the known height of the camera above the ground (Fig. 2).

2.2.3. Climbing

One of the biggest engineering challenges in deploying a personal rover is creating the locomotory means for a robot to navigate a typical domestic environment. Houses have steps and a variety of floor surfaces. Most homes have staircases, doorjambs between interior rooms and steps between rooms and outside

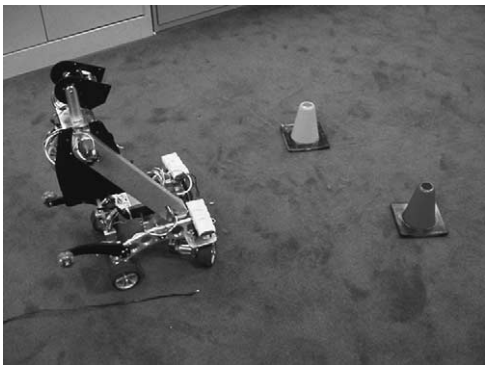

Fig. 2. The rover uses landmarks to navigate.

porches. Although brute force solutions to step climbing clearly exist (e.g. treaded and very large robots), it is a daunting task to create a mechanism that is both safe for the robot and safe for the environment.

Several recent robots have made significant advances in efficient climbing. The EPFL Shrimp can climb a range of terrain types, including regular steps, using six powered wheels and an innovative passive jointed chassis [9]. The RHex robot demonstrates highly terrainable legged locomotion using sophisticated leg position control [15].

For the personal rover we pursued a wheeled design due to efficiency on flat ground, the anticipated

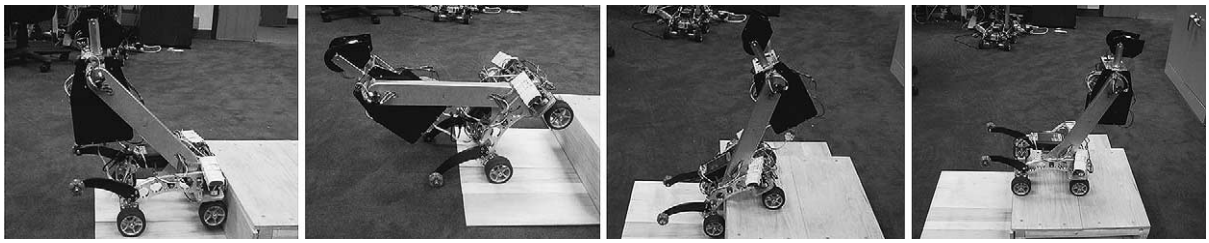

Fig. 3. Four different stages in climbing up a step.

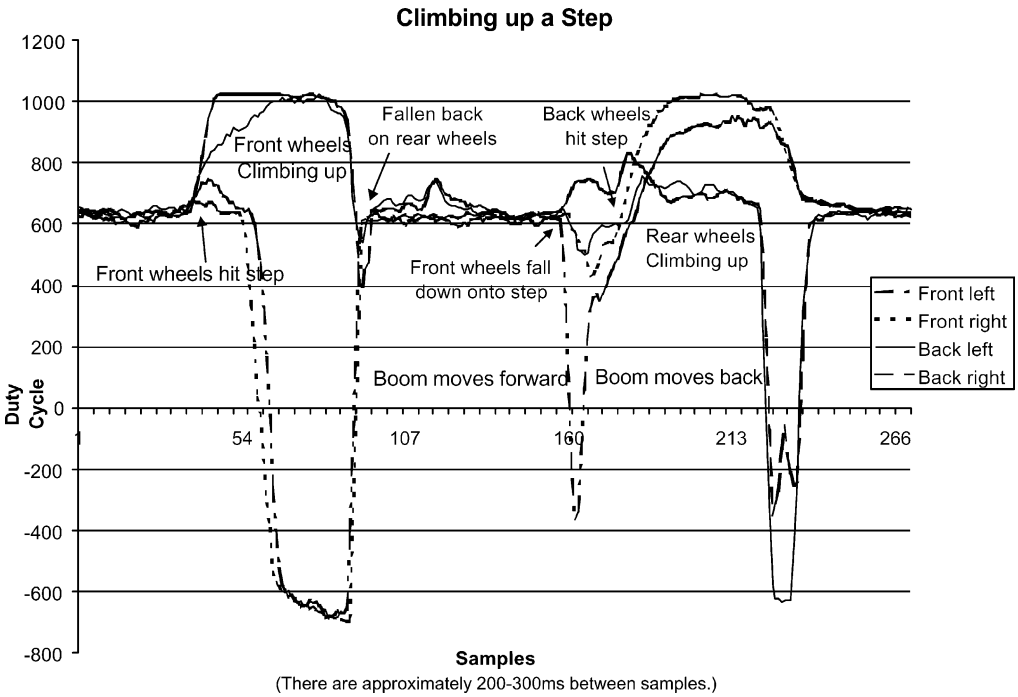

Fig. 4. Back-EMF trajectories during step climb.

environment for most rover travels. In order to surmount large obstacles such as steps, the rover employs a heavy swinging boom that contains the main processor, daughter boards, batteries and CMUcam. By moving its center of gravity, the rover can climb up steps many times the diameter of its wheels. Currently the omni-wheels can be moved to allow the rover to climb up steps 7 in. tall.

Fig. 3 shows several stages in the step climbing process. During this process, motor current data is used by the control program to infer the terrain beneath the rover (Fig. 4). With the boom moderately aft, the rover approaches the step edge while monitoring wheel currents. When both front wheels have contacted the step edge, the back wheels are moving forward with full power and the front wheels are actually applying current in the negative direction to keep them from moving too quickly, due to the geometry of this fixed-speed approach.

Next, the rover moves the boom aft, causing the rover to fall backwards onto the omni-wheels, and detects this event. Finally, with the front wheels over the top face of the step, the rover moves the boom fore, positioning its center of gravity just behind the front wheels. Because there are necessarily no omni-wheels at the front of the robot, it is in danger of falling forward during the step climbing procedure, and thus the boom position must be modulated to maintain maximum pressure on the front wheels while keeping the center of gravity behind the front wheels.

3. Interaction design

The interaction design process started, as described earlier, using a user-centered experience design process commonly used for commercial toy and vehicle development. A critical requirement borne from this analysis was that the non-technological user must be able to shape and schedule the activities of the rover over hours, days and weeks. Two basic requirements of such an interface have been addressed thus far: teaching and scheduling. First, a successful interface should facilitate *teaching* the rover new types of tasks to perform while maximally building upon the rover's prior competencies. Second, a scheduling interface should enable the long-term behavior of the rover to be planned, monitored and changed.

3.1. Perception-based teaching

A successful interface must address the question of how one can teach the rover to navigate a home environment reliably. Given price and complexity constraints, we are strongly biased toward vision as a multi-use sensor. As an incremental step toward passive, vision-based navigation, we simplify the visual challenge by placing high-saturation landmarks in static locations throughout the test area.

Our goals in developing a teaching interface for the rover include:

- The user environment must be highly intuitive.
- The language must be expressive enough to navigate a house.
- The navigation information must be stable to perturbations to the physical environment.

3.1.1. Definitions

The basic data structures underlying the teaching environment are Actions, LandmarkViews, Landmarks, Locations and Paths.

- *Action*. Any basic task that the rover can perform. Actions include things such as pure dead-reckoning, driving to landmarks, turning in place, and checking for the presence of landmarks. Examples of Actions include:
  - *ClimbAction*: climb up or down a step;
  - *DriveToAction*: dead-reckon driving;
  - *DriveTowardMarkAction*: drive toward a landmark, stopping after a set distance;
  - *LookLandmarkAction*, check for the presence of a landmark;
  - *SendMessageAction*: send the user a message;
  - *StopAtMarkAction*: drive toward a landmark, stopping at a location relative to the landmark (e.g. 2 ft to the left, 12 in. in front, etc.);
  - *TurnToAction*: turn a set number of degrees;
  - *TurnToMarkAction*: turn until facing a landmark.
- *LandmarkView*. What a landmark looks like; its "view". This can be thought of as a landmark "type", that is, it contains information about a landmark but not positional information. It keeps track of the color, name, and image of the landmark.

- *Landmark*. A landmark with positional information. A Landmark object contains a LandmarkView object as well as pan and tilt values for where the rover expects to see this landmark.
- *Location*. A location is identified by a set of Landmarks and a unique name. A Location also stores the known paths leading away from that location. The rover neither independently determines where it is, nor compares stored images with what the camera currently sees. Rather, the user must initially tell the rover where it is, at which point it can verify whether it can see the landmarks associated with that location. If it cannot see these landmarks, then it can query the user for assistance.
- *Path*. A series of Actions, used to get the rover from one Location to another. A Path executes linearly; one action is performed, and if it completes successfully, the next executes. Paths actually have a tree structure, so that they have the capability of having alternate Actions specified. Thus, for example, a Path from point A to point B might be "drive to the red landmark, but if for some reason you cannot see the red landmark, drive to the green one and then turn 90°".

3.1.2. User interface

While the rover can dead-reckon locally with a high degree of accuracy, navigation robustness in the long-term depends on the reliable use of visual landmarks. Designing the user's teaching method to be a wizard-based interface is a promising direction. The wizard constrains user control of the rover to the atomic actions available to the rover itself as an autonomous agent. Without the ability to manipulate the rover's degrees of freedom directly, the user must view the world from the robot's point of view, then identify the appropriate visual cues and closed-loop controls to effect the desired motion. This is critical to overall system stability because each atomic rover behavior can be designed to be robust to local perturbations (i.e. rover translation and rotation).

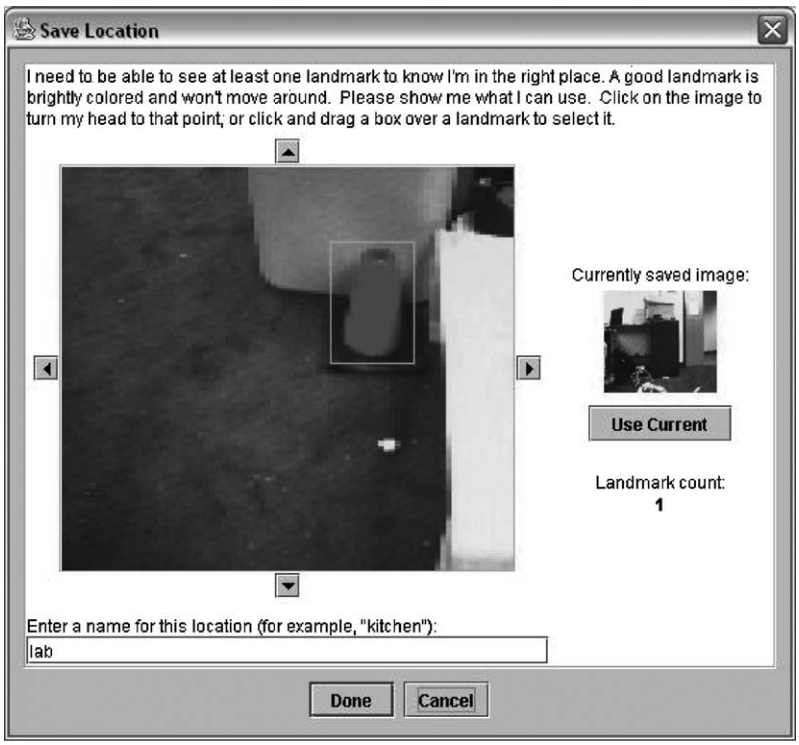

Fig. 5. This screen shot, taken during a trial run, shows the user selecting a landmark while saving a location.

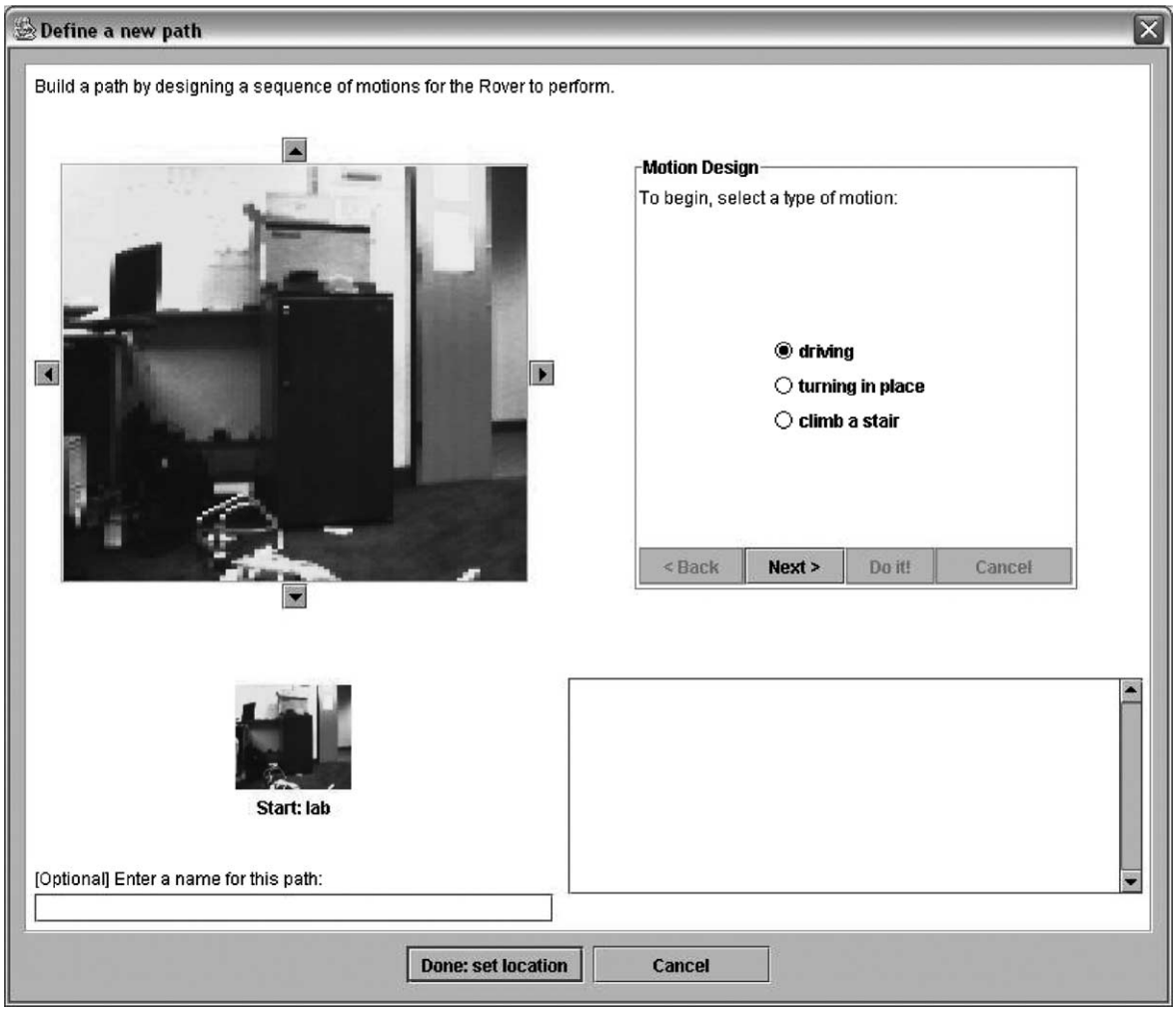

Fig. 6. This screen shot, taken during a trial run, shows the start of path teaching.

For example, the teaching interface allows the user to specify a landmark by outlining a box around the desired landmark on the displayed camera frame (Fig. 5). If the rover is able to track the landmark that the user selected, it compares the new landmark to all the previously seen and named LandmarkViews. If no match is found, the rover asks the user whether she would like to save this new type of landmark. Saved landmarks can then be used offline in mission design, discussed below.

To begin teaching the rover, the user must first specify the rover's current location. To do this, the user selects one or more landmarks, so that the rover can identify the location in the future (Fig. 5).

To teach the rover paths between points in a home, the user is presented with a wizard-based interface to define each step of the path. Each of these steps maps directly to Actions, and may be something like "drive until you are directly in front of a landmark", "climb up a step", or "turn 90°". Fig. 6 depicts the start of path teaching. The user is presented with an image of what the rover can see, the wizard for instructing the rover, a box where the history of the actions performed will be displayed, and other information relevant to

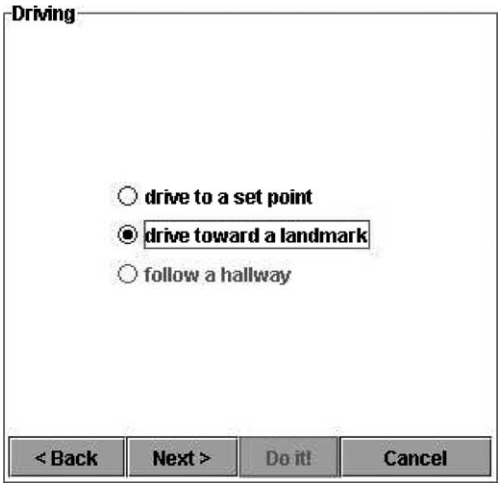

Fig. 7. Driving options.

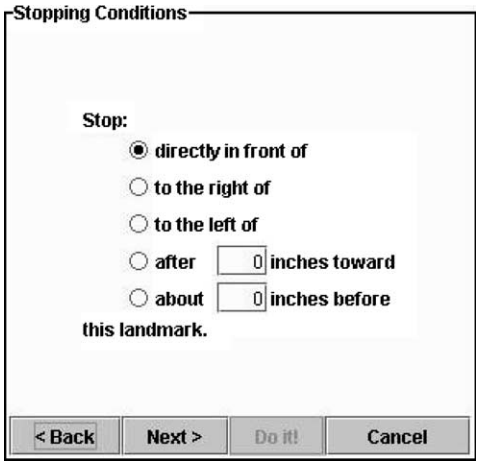

Fig. 9. Stopping conditions.

this path. By progressing through a series of panels, such as those shown in the screen shots in Figs. 7–10, the user can instruct the rover exactly as necessary. The full wizard, along with the Actions that can be produced, is shown in Fig. 11.

3.2. Mission design, scheduling, and execution

The rover's daily activities are controlled through the design and execution of autonomous *missions*. Each mission is a task or experiment that the user has constructed from a set of individual rover movements and actions. Personal rover missions may mimic the exploratory and scientific missions performed by NASA's Mars Rover or may accomplish new goals created by the user. For example, the rover could make a map of the house or chart the growth of a plant. Missions are fairly autonomous, with varying degrees of user interaction in the case of errors or insurmountable obstacles. Mission scheduling allows the rover to carry out missions without requiring the user's presence.

Fig. 8. Selection of a landmark.

Fig. 10. Summary.

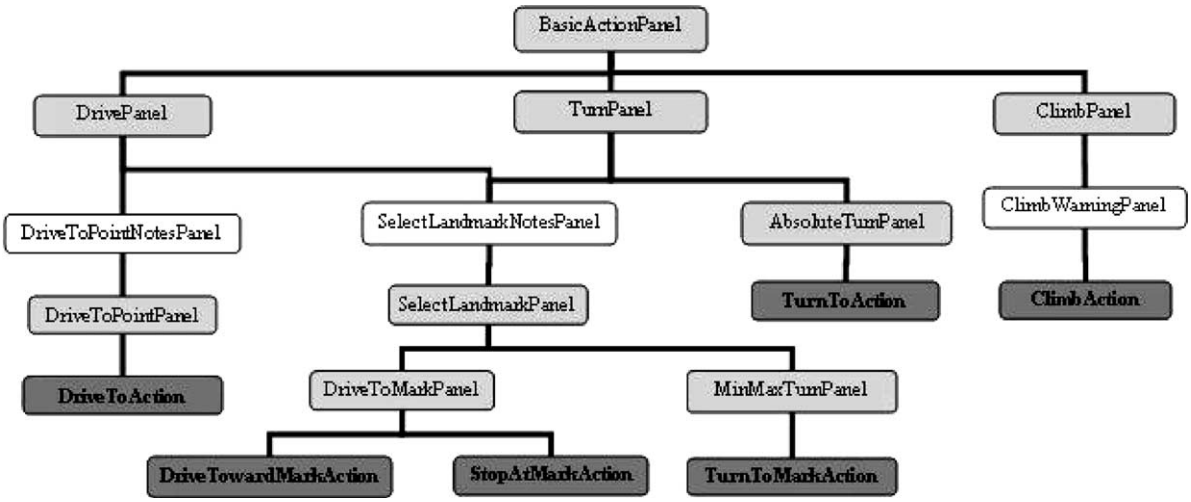

Fig. 11. Flow of ActionPanels in action design wizard. Actions are shown in dark gray, panels which request user input are shown in light gray, and panels which merely provide information are shown in white.

Our goals in developing a user interface for mission design, scheduling, and execution include:

- The mission design interface should allow the user to design and program creative missions by combining individual actions. The interface should be intuitive enough so that the user can begin using it immediately, but flexible enough so as not to limit the user's creativity as they grow familiar with the rover.
- Mission scheduling should make the user think beyond the rover's immediate actions to the rover's long-term future over days and even months.
- Mission execution should offer adjustable degrees of human–machine interaction and control for mission reporting and error handling.
- The software should support communication of the rover's status through different means such as email, PDA, or cell phone.

3.2.1. *Mission development*

To build a mission, the user first clicks on the Mission Development tab of the user interface. Here there is a set of *blocks* grouped by function, with each block representing a different action that the rover can perform. Some of the blocks are static, such as the block used to take a picture. Others can be defined and changed by the user through the teaching interface. For example, the block used to follow a path allows the user to choose any path that they have previously taught the rover.

The user can select a block by clicking on it with the mouse. While a block is selected, clicking in the Mission Plan section will place the block and cause a gray shadow to appear after it. This shadow indicates where the next block in the mission should be placed. To build a mission, the user strings together a logical set of blocks (Fig. 12).

As each block is placed, a popup window is displayed. Here the user can enter the necessary details for the action, for example, the starting and ending location of a path (Fig. 13).

We have currently implemented two different types of blocks. The first simply represents a single action that can be followed directly by another action, for example, sending a message (Fig. 14). The second represents a conditional action, in which different actions can be taken based on the outcome. For example, when looking for a landmark, one action can be taken if a landmark is found and a different action can be taken if the landmark is not found (Fig. 14). These blocks can have any number of conditions. As well as the true and false conditions shown in the landmark example, blocks can condition on equality and inequality. For example, one could implement a block for checking if the IR range

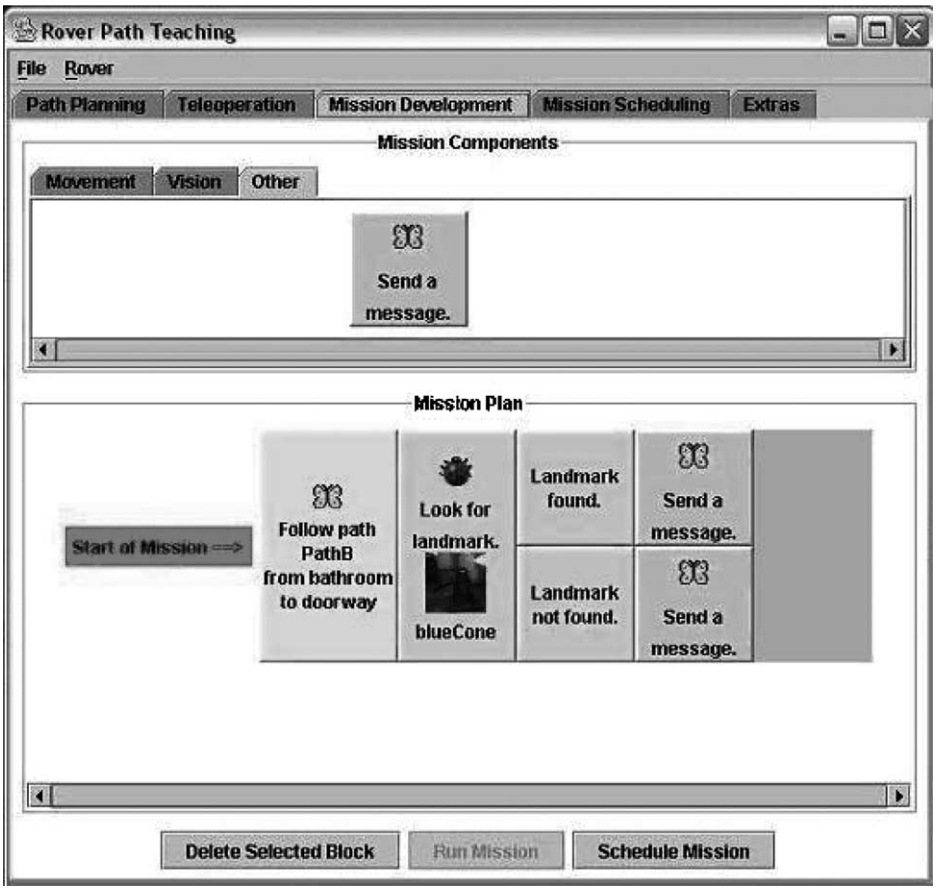

Fig. 12. Screen shot of a user building a mission by placing individual action blocks together.

finder value is less than x, equal to x, or greater than x.

It is possible to build a mission that cannot be run by the rover. For example, the simple mission "follow a path from A to B then follow a path from C to D" does not make sense. A red X icon indicates the blocks where there are errors (Fig. 15). The user can delete the bad blocks, or right click on a block to display the popup window and edit the details for that block. Other than mismatched locations, currently supported errors are invalid paths and invalid landmark selections.

One planned future improvement in the area of mission development is to implement two new block types. One type of block will allow sections of the mission to be repeated. The user will be able to choose a number of times to repeat the section, or to repeat until a certain condition is met. The other block type will allow the user to define her own subroutine blocks. These user-defined blocks can then be used as functions, allowing a set of actions to be added to the mission as a group. The user-defined blocks will also allow the same set of actions to be easily added to multiple missions.

3.2.2. *Mission scheduling and execution*

After designing a mission, the user has the option to run the mission immediately or schedule the mission to run at a later time. Scheduling the mission allows the user to select a starting time and date as well as how often and how many times the mission should be repeated. The user also gives the mission a unique name. Fig. 16 shows the scheduling wizard.

Fig. 13. Screen shots of the popup window that prompts the user to select a starting location and then an appropriate ending location to create a path. Ending locations that would create an invalid or unknown path are disabled.

Before accepting a mission schedule, we check for conflicts with all of the previously scheduled missions. If any conflicts are found, we prompt the user to reschedule the mission, cancel the mission, reschedule the conflicts, or cancel the conflicts as shown in Fig. 17. In the future, we plan to allow both the precise scheduling currently implemented and a less rigid scheduling method. For example, the user could schedule a mission to run around a certain time or whenever the rover has free time. For these more flexible missions, the rover will handle conflict avoidance without requiring additional user input.

All the scheduled missions can be viewed by clicking on the Mission Scheduling tab of the user interface. The user can select any of the scheduled missions to view the details of the schedule. The user can also cancel a mission or edit the schedule. In the future we plan to implement a graphical view of the rover's schedule. The Mission Scheduling panel will include a calendar showing all of the scheduled missions.

Fig. 14. Sending a message is an unconditional action. Looking for a landmark is a conditional action with two different possible outcomes.

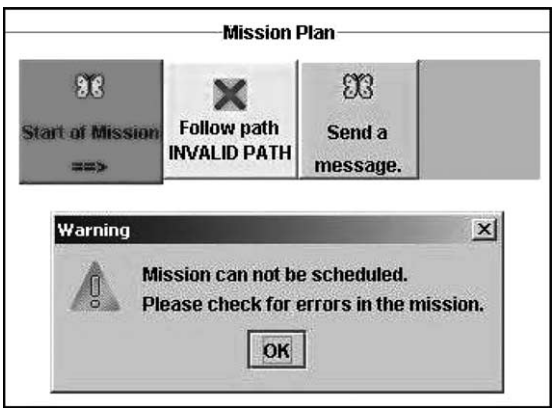

Fig. 15. A red X icon indicates any blocks with errors. The mission may not be run or scheduled until the errors are corrected or removed.

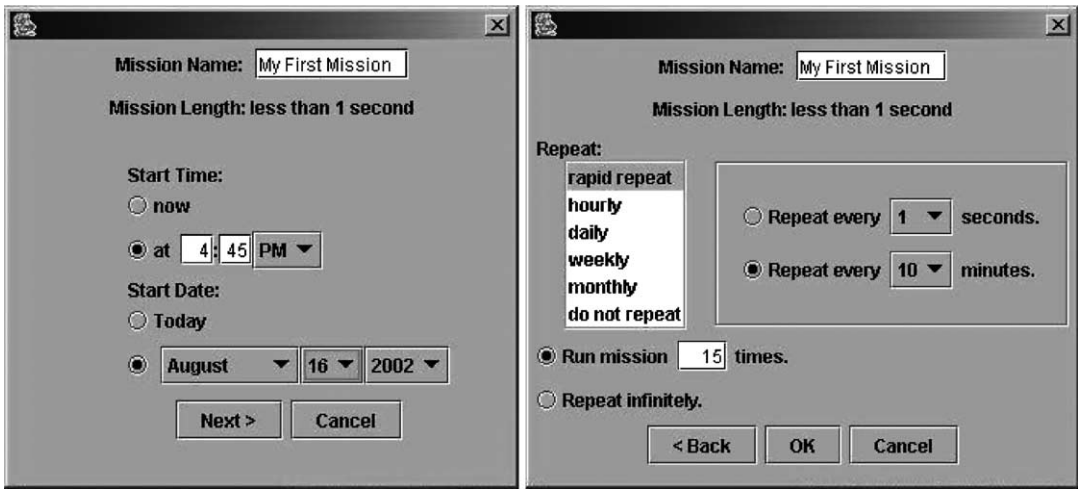

Fig. 16. When scheduling a mission the user selects the start time and date as well as how often and how many times to repeat the mission.

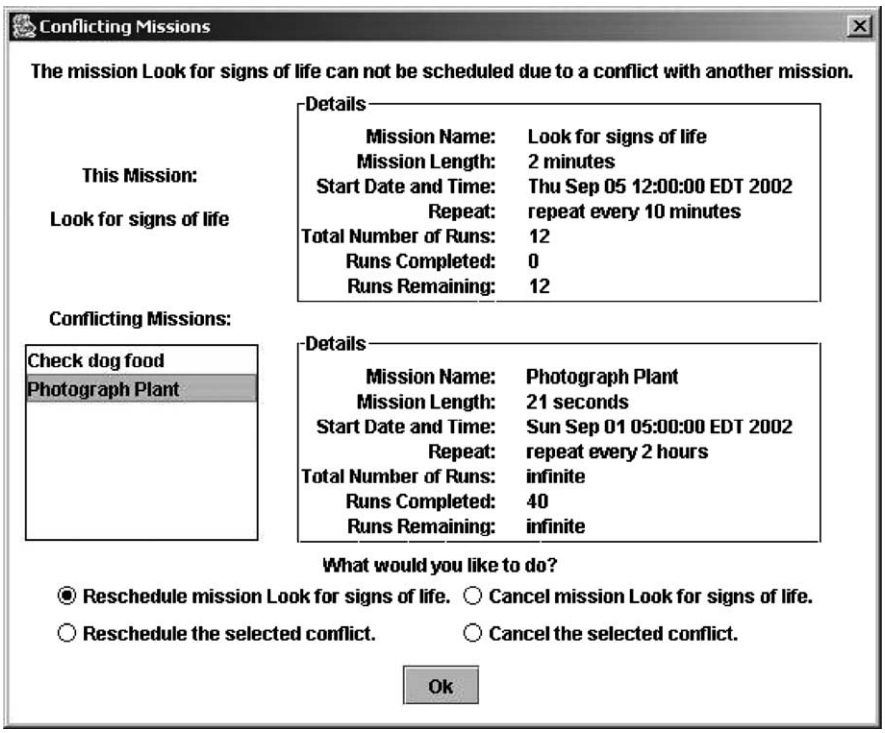

Fig. 17. When there is a scheduling conflict, a dialog prompts the user to resolve the conflict.

4. Conclusions

The personal rover combines mechanical expressiveness with a simple-to-use interface designed explicitly for a long-term human–robot relationship. Currently, three prototype rovers have been fabricated to prepare for preliminary user testing. Both from a mechanical and user interface point of view, the rover is not yet sufficiently advanced to accompany a subject home for the month. Thus, initial user testing of the interface will take place at Carnegie Mellon University's campus over the duration of a day. Planned rover improvements include making landmark recognition less dependent on lighting conditions, increasing feedback and interaction during path following and mission execution, giving the rover the ability to ask for and receive help, increasing battery life, and making step climbing faster.

Acknowledgements

We would like to thank NASA-Ames Autonomy for their financial support, Peter Zhang and Kwanjee Ng for the design and maintenance of the rover electronics, and Tom Hsiu for the design of the rover hardware.

References

[1] Y. Abe, M. Shikano, T. Fukuda, F. Arai, Y. Tanaka, Vision based navigation system for autonomous mobile robot with global matching, in: Proceedings of the International Conference on Robotics and Automation, Detroit, MI, 1999, pp. 1299–1304.

[2] J.R. Asensio, J.M.M. Montiel, L. Montano, Goal directed reactive robot navigation with relocation using laser and vision, in: Proceedings of the International Conference on Robotics and Automation, Detroit, MI, 1999, pp. 2905–2910.

[3] C. Bartneck, M. Okada, Robotic user interfaces, in: Proceedings of the Human Computer Conference, 2001.

[4] A. Billard, Robota: clever toy and educational tool, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 2002. Also: Robotics and Autonomous Systems 42 (2003) 259–269 (this issue).

[5] C. Breazeal, B. Scassellati, A context-dependent attention system for a social robot, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI99), Stockholm, Sweden, 1999, pp. 1146–1151.

[6] A. Bruce, I. Nourbakhsh, R. Simmons, The role of expressiveness and attention in human–robot interaction, in: Proceedings of the ICRA, 2002.

[7] The cerebellum microcontroller. http://www.roboticsclub.org/cereb.

[8] P. Coppin, A. Morrissey, M. Wagner, M. Vincent, G. Thomas, Big signal: information interaction for public telerobotic exploration, in: Proceedings of the Workshop on Current Challenges in Internet Robotics, ICRA, 1999.

[9] T. Estier, Y. Crausaz, B. Merminod, M. Lauria, R. Piguet, R. Siegwart, An innovative space rover with extended climbing abilities, in: Proceedings of the Space and Robotics 2000, Albuquerque, USA, February 27–March 2, 2000.

[10] M. Fujita, H. Kitano, Development of an autonomous quadruped robot for robot entertainment, Autonomous Robots Journal 5 (1998).

[11] I. Horswill, Visual collision avoidance by segmentation, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1994, pp. 902–909.

[12] H. Kozima, H. Yano, A robot that learns to communicate with human caregivers, in: Proceedings of the First International Workshop on Epigenetic Robotics, Lund, Sweden, 2001.

[13] H. Hüttenrauch, K. Severinson-Eklundh, Fetch-and-carry with CERO: observations from a long-term user study, in: Proceedings of the IEEE International Workshop on Robot and Human Communication, 2002.

[14] B. Mikhak, R. Berg, F. Martin, M. Resnick, B. Silverman, To mindstorms and beyond: evolution of a construction kit for magical machines, in: A. Druin (Ed.), Robots for Kids: Exploring New Technologies for Learning Experiences, Morgan Kaufman/Academic Press, San Francisco, CA, 2000.

[15] E.Z. Moore, M. Buehler, Stable stair climbing in a simple hexapod, in: Proceedings of the Fourth International Conference on Climbing and Walking Robots, Karlsruhe, Germany, September 24–26, 2001.

[16] National/Panasonic Press Release, Matsushita Electric (Panasonic) develops robotic pet to aid senior citizens with communication, Matsuthita Corporation, March 24, 1999.

[17] NEC Press Release, NEC develops friendly walkin' talkin' personal robot with human-like characteristics and expressions, NEC Corporation, March 21, 2001.

[18] I. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, A. Soto, An affective mobile robot educator with a full-time job, Artificial Intelligence Journal 114 (1–2) (1999) 95–124.

[19] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, S. Thrun, Towards robotic assistants in nursing homes: challenges and results, in: Proceedings of the Workshop on Social Robots (IROS 2002), 2002.

[20] A. Rowe, C. Rosenberg, I. Nourbakhsh, CMUcam: a low-overhead vision system, in: Proceedings of the IROS 2002, Switzerland, 2002.

[21] J. Schulte, C. Rosenberg, S. Thrun, Spontaneous, short-term interaction with mobile robots, in: Proceedings of the ICRA, 1999.

[22] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, Probabilistic algorithms and the

interactive museum tour-guide robot minerva, International Journal of Robotics Research 19 (11) (2000) 972–999.
[23] I. Ulrich, I. Nourbakhsh, Appearance-based obstacle detection with monocular color vision, in: Proceedings of the AAAI 2000, Menlo Park, CA, 2000.
[24] I. Ulrich, I. Nourbakhsh, Appearance-based place recognition for topological localization, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2000.

Emily Falcone received her BS degree in Computer Science from Carnegie Mellon University in 2002. She is currently employed at the Carnegie Mellon University Robotics Institute where she performs research in the Mobile Robot Programming Lab. Her research focuses on scheduling, interface design, and long-term human–robot interaction. She is also interested in the application of robots in education.

Rachel Gockley is an undergraduate at Carnegie Mellon University. She expects to receive her BS degree in Computer Science and Philosophy with a Minor in Robotics in 2003. She plans to pursue a Ph.D. in Robotics. Her research interests include mobile robots and human–robot interaction. She is a member of the honor society of Phi Kappa Phi.

Eric Porter is a Senior Computer Science major at Carnegie Mellon University. He is pursuing a Minor in Robotics and his research interests include mobile robotics and computer vision.

Illah R. Nourbakhsh is an Assistant Professor of Robotics in The Robotics Institute at Carnegie Mellon University. He received his Ph.D. in Computer Science from Stanford University in 1996. He is co-founder of the Toy Robots Initiative at The Robotics Institute. His current research projects include electric wheelchair sensing devices, robot learning, theoretical robot architecture, believable robot personality, visual navigation and robot locomotion. His past research has included protein structure prediction under the GENOME project, software reuse, interleaving planning and execution and planning and scheduling algorithms. At the Jet Propulsion Laboratory he was a member of the New Millennium Rapid Prototyping Team for the design of autonomous spacecraft. He is a founder and chief scientist of Blue Pumpkin Software, Inc.