User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment

Y. Wang, M. Huber, V. N. Papudesi, and D. J. Cook

Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

Abstract

Autonomous robots hold the possibility of performing a variety of assistive tasks in intelligent environments. However, widespread use of robot assistants in these environments requires ease of use by individuals who are generally not skilled robot operators. In this paper we present a method of training robots that bridges the gap between user programming of a robot and autonomous learning of a robot task. With our approach to variable autonomy, we integrate user commands at varying levels of abstraction into a reinforcement learner to permit faster policy acquisition. We illustrate the ideas using a robot assistant task, that of retrieving medicine for an inhabitant of a smart home.

1 Introduction

The application of robot technologies in complex, semi-structured environments and in the service of general end-users promises many benefits. In particular, such robots can perform repetitive and potentially dangerous tasks, as well as assist in operations that are physically challenging for the user. In the context of intelligent environments, assistive robots have a variety of functions to offer. They can move through the environment making sure that the contents and inhabitants are secure. They can also perform simple tasks such as cleaning and retrieving needed objects.

Moving robot systems from factory settings into more general environments, particularly environments requiring interaction with humans, poses large challenges for their control system and for the interface to the human user. The robot system must be able to operate based on direct user guidance or increasingly autonomously as the environment, robot experience, and task complexity dictates. Furthermore, it must do so in a safe and efficient manner without requiring constant, detailed user input which can lead to rapid user fatigue (Wettergreen et al., 1995).

For personal robot applications, such as robot assistive tasks in intelligent environments, this requirement is further amplified by the fact that the user is generally not a skilled engineer and can therefore not be expected to be able or willing to provide constant, detailed instructions. An inhabitant of a smart home, for example, would like to request that needed medicine be retrieved without giving detailed instructions of how to accomplish the task. For the user interface and the integration of human input into an autonomous control system, this implies that a robot system must facilitate the incorporation of user commands at different levels of abstraction and at different bandwidths. This, in turn, requires operation at varying levels of autonomy (Dorais et al., 1998; Hexmore et al., 1999) depending on the available user feedback.

An additional challenge arises because efficient task-performing strategies that conform with the preferences of the user are often not available *a priori*. As a result, the system has to be able to acquire them on-line while ensuring that autonomous operation and user-provided commands do not lead to catastrophic failures.

In recent years, a number of researchers have investigated the issues of learning and user interfaces (Clouse & Utgoff, 1992; Smart & Kaelbling, 2000; Kawamura et al., 2001). However, this work was conducted largely in the context of mission-level interaction with the robot systems using skilled operators. In contrast, the approach presented here is aimed at the integration of potentially unreliable user instructions into an adaptive and flexible control framework in order to adjust control policies on-line. The learned policies should more closely reflect the preferences and requirements of the particular end-user. To achieve this, user commands at different levels of abstraction are integrated into an autonomous learning component. Their influence speeds learning of the control policy, but is limited to not prevent ultimate task achievement. As a result, the robot can seamlessly switch between fully autonomous operation and the integration of high and/or low-level user commands.

In the remainder of this paper, our approach to variable autonomy is presented. In particular, fully autonomous policy acquisition, the integration of high-level user commands in the form of subgoals and the user of intermittent low-level instructions using direct teleoperation are introduced. Their use is demonstrated in the context of an intelligent environment task using a

walking robot, that of retrieving an object as requested by the environment inhabitant.

2 Combining User Input and Autonomous Learning for Variable Autonomy

The approach presented here introduces a method of achieving variable autonomy by integrating user input and autonomous control policies in a Semi-Markov Decision Process (SMDP) model that is built on a hybrid control architecture. Overall behavior is derived from a set of reactive behavioral elements that address local perturbations autonomously. These elements are endowed with formal characteristics that permit the hybrid systems framework to impose *a priori* safety constraints that limit the overall behavior of the system (Huber & Grupen, 1999; Ramadge and Wonham, 1989). These constraints are enforced during autonomous operation as well as during phases with extensive user input. In the latter case, they overwrite user commands that are inconsistent with the specified safety limitations and could thus endanger the system. The goal here is to provide the robot with the ability to avoid dangerous situations while facilitating flexible task performance.

On top of this control substrate, task-specific control policies are represented as solutions to an SMDP, permitting new tasks to be specified by means of a reward structure r_T that provides numeric feedback according to the task requirements. The advantage here is that specifying intermittent performance feedback is generally much simpler than determining a corresponding control policy. Using this reward structure, reinforcement learning (Barto et al., 1993; Kaelbling et al., 1996) is used to permit the robot to learn and optimize appropriate control policies from its interaction with the environment. When no user input is available, this forms a completely autonomous mode of task acquisition and execution.

User input at various levels of abstraction is integrated into the same SMDP model. User commands temporarily guide the operation of the overall system and serve as training input to the reinforcement learning component. Use of such training input can dramatically improve the speed of policy acquisition by focusing the learning system on relevant parts of the behavioral space (Clouse & Utgoff, 1992). In addition, user commands provide additional information about user preferences and are used here to modify the way in which the robot performs a task. This integration of user commands with the help of, and as a jumpstart for, reinforcement learning facilitates a seamless transition between user operation of the robot and fully autonomous execution, based on the availability of user input. Furthermore, it permits user commands to alter the performance of autonomous control strategies without the user needing to provide a complete specification of the control policy. Figure 1 shows a high-level overview of the components of the control system.

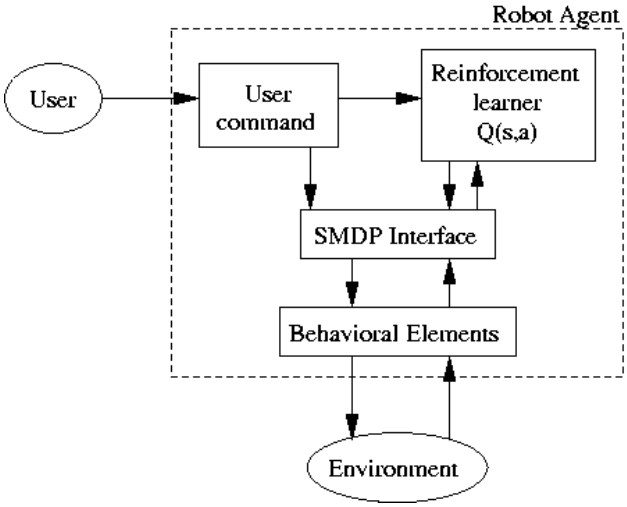

Figure 1. Overview of the control system.

In the work presented here, user commands at a high level of abstraction are presented to the SMDP model in the form of temporary subgoals to be achieved or suggested specific actions to execute. This input is used, as long as it conforms with the *a priori* safety constraints, to temporarily drive the robot. At the same time, user commands play the role of training input to the learning component, which optimizes the autonomous control policy for the current task. Here, Q-learning (Watkins, 1989) is used to estimate the utility function, *Q(s,a)*, by updating its value when action *a* is executed from state *s* according to the formula

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)),$$

where *r* represents the obtained reward.

Low-level user commands in the form of intermittent continuous input from devices such as a joystick are included in the same fashion into the learning component, serving as temporary guidance and training information.

3 User Commands as Reward Modifiers

To address the preferences of the user beyond a single execution of the action and to permit user commands to have long-term influence on the robot's performance of a task, we employ user commands to modify the task-specific reward structure to more closely resemble the actions indicated by the user. This is achieved by means of a separate user reward function, r_u, that represents the history of commands provided by the user. User input is

captured by means of a bias function, *bias(s,a),* which is updated each time a user gives a command to the robot according to the function

$$bias(s,b) \leftarrow \begin{cases} bias(s,b) + (n-1) & \text{if } b = a \\ bias(s,b) - 1 & \text{otherwise} \end{cases}$$

$$r_u(s,a) = f(bias(s,a)),$$

where action a in state s is part of the user command and there are n possible actions in state s. The total reward used by the Q-learning algorithm throughout robot operation is then

$$r = r_t + r_u,$$

leading to a change in the way a task is performed even when operating fully autonomously.

Incorporating user commands into the reward structure rather than directly into the policy permits the autonomous system to ignore actions that have previously been specified by the user if they were contradictory, if their cost is prohibitively high, or if they prevent the achievement of the overall task objective as specified by the task reward function, r_t. This is particularly important in personal robot systems such as assistive robots in intelligent environments, where the user is often untrained and might not have a full understanding of the robot mechanism. For example, a user could specify a different, random action every time the robot enters a particular situation (e.g., a different fetch operation from a different location). Under these extreme circumstances, the user rewards introduced above would cancel out and no longer influence the learned policy. Similarly, the user might give a sequence of commands which, when followed, form a loop (e.g., perform sentry duty over the entire house, returning to start location) and thus prevent the achievement of the task objective. To avoid this, the user reward function has to be limited to ensure that it does not lead to the formation of spurious loops. In the approach presented here, the following formal lower and upper bounds for the user reward, r_u, applied to action *a* in state *s*, have been established and implemented. Details on the derivation of the bounds are reported elsewhere (Papudesi, 2002).

$$-\max_{a \in A} Q(s,a) - r < r_u < Q(s,a)(1-\gamma) - r_T$$

These bounds ensure that the additional user reward structure does not create any loops, even if explicitly command by the user. As a result, the system can successfully achieve the overall task objective provided by the task reward, r_T.

4 Experiments

To demonstrate the power and applicability of the model of variable autonomy introduced here, a number of experiments in simulation and on mobile and walking robot tasks have been performed. These experiments demonstrate that the approach presented here provides an effective interface between robot and human as well as a valuable robot training mechanism.

4.1 High-Level User Commands

Our first experiment demonstrates the integration of user commands and autonomous learning. The goal of the robot navigation task is to learn to optimally navigate the environment and reach a specific target. The environment itself consists of a set, *V*, of via points superimposed on a collection of maps consisting of a 50x50 grid of square cells. These via points represent user-guided bias and thus affect the problem reward.

Actions are specified as instances of geometric motion controllers that permit the robot to move safely between subsets of the via points. These actions directly handle the continuous geometric space by computing collision-free paths to the selected via point, if such a path exists. Targets represented by via points are directly reachable by at least one controller. However, controllers are only applicable from a limited number of states, making it necessary to construct navigation strategies as a sequence of via points that lead to the target location. Here, harmonic path control (Connolly & Grupen, 1993), a potential-field path planner is used to generate continuous robot trajectories while ensuring that the robot does not collide with an object. By abstracting the environment into a set of via points, the agent is capable of a combination of geometric and topological path planning. At the lower level, each harmonic controller generates velocity vectors that describe the path geometrically. At the higher level, the D-EDS Supervisor produces topological plans in the form of sequences of via points.

To illustrate the guidance of the robot using high-level user commands in the form of subgoals, two experiments were performed on the Pioneer 2 mobile robot. These experiments demonstrate the ability of high-level user input to accelerate learning and modify autonomous behavior while avoiding unreliable user commands.

First, we demonstrate the capability of the approach to use sparse user input to modify the learned control policy. This forces the learned policy to more closely reflect the preferences of the user. We demonstrate this capability on a navigation task, which is first learned without user input and then modified by incorporating a single user command in the form of an intermediate subgoal. Because the subgoal is outside the chosen path, the

learned path is modified based on user input, as shown in Figure 3. Here, the end location is marked with an X and the learned paths are highlighted. Figure 3 shows the corresponding changes in the Q-value and user reward functions for the previously best action (black line) and the new best action (grey line). These graphs illustrate the effect of the command on the reward function for the task and, as a result, on the value function and policy. Figure 4 shows the robot performing the navigation task.

Figure 2. Change in control policy by user command.

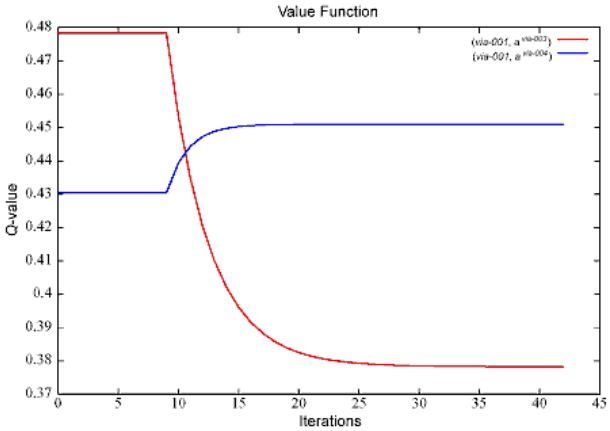

Figure 3. Change in the Q-value function due to the input user command.

Figure 4. Pioneer robot executing navigation task.

Second, we illustrate the capability of the presented approach to overwrite inconsistent user commands that would invalidate the overall task objective. Here, the user explicitly commands a loop between two via points. Figure 5 shows the loop specified by the user commands the the learned loop-free policy that the robot executes after learning.

Figure 5. User-specified loop (left) and resulting loop-free learned policy (right).

Although the robot will execute the loop as long as the user explicitly commands this, it reverts to a policy that fulfills the original task objective as soon as no further user commands are received.

4.2 Multi-Level User Input

A second set of experiments was performed using a walking robot dog, Astro (shown in Figure 6), to demonstrate user-guided robot learning at multiple levels of abstraction. In these experiments, high-level subgoals as well as low-level joystick commands were integrated to demonstrate the capabilities of the presented model for variable autonomy.

Once again, the robot task is to navigate to a specified location, but user guidance takes multiple forms. First, user-specified subgoals represent via-points that Astro should visit en route to the goal location. Second, user interaction guides the selection of low-level movement patterns for Astro to make. In the wheeled robot navigation task, a harmonic path is calculated for the robot to circumvent corners in the space that could cause collisions. However, this motion is inefficient for the smaller, and potentially more agile, walking dog. As a result, we provide two movement options for Astro: straight line and harmonic motion.

A reinforcement learning algorithm is used to select the movement pattern that is best for any pair of via points. As we mentioned before, the system has two controllers, namely a line controller and a harmonic controller, to determine the most appropriate moving pattern of the robot. If line controller is chosen, the robot will travel between via-points along a straight line, while with the harmonic controller, the robot will walk along a curve. For a given pair of via points, the user selects a direction for the robot to follow or allows the algorithm to select a motion consistent with the learned policy. If the user selects a direction, the dog moves in this direction for a fixed distance. The executed path is compared with the path generated using one of the two predetermined movement patterns, and the movement choices are given

rewards based on the difference between planned and selected movement paths.

This combination of high-level and low-level user guidance is validated in a walking robot-based navigation task. Here, Astro is successfully taught the best moving style to follow from one location to another based on joystick-controlled direction from the user as well as the via points shown in Figure 6. Localization for this task is based on heuristics, but future implementations will make use of paw joint angles to further improve estimation of the robot current location.

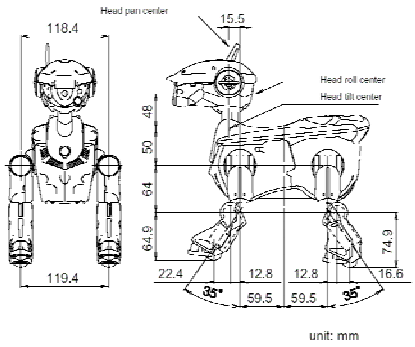

Figure 5. Walking robot dog, Astro.

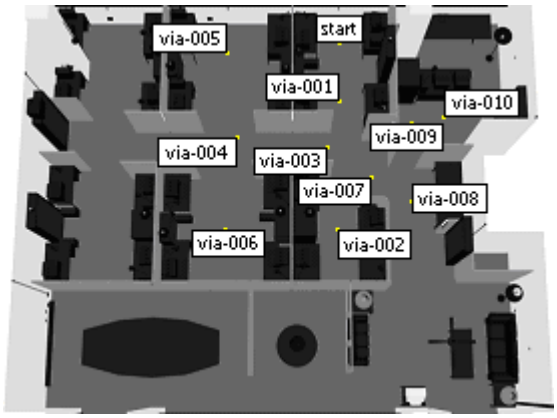

Figure 6. MavHome layout with via points.

In this experiment, point via-005 is specified as the goal. Initially, Astro chose via-003 as the first subgoal. However, the user discourages that choice because it would move too close to the wall. Astro then selects via-005 as a subgoal. Because there is a wall between the start and goal locations, this choice also ultimately fails and Astro selects via-001 as the next choice.

Although the subgoal choice is viable, Astro selects a harmonic motion to reach via-001. The user intercedes using a joystick to flatten the path and the movement policy is refined based on this interaction.

After reaching point via-001, Astro begins to move straight toward point via-005, which again leads him to the corner. The user maneuvers the joystick to avoid this. After reaching via-003 using a curved motion, Astro wisely chooses point via-004 as a pass-through point and then walks straight to via-004 and finally to the goal. When repeating the same task, Astro improves his movement efficiency based on the same high-level and low-level feedback from the user.

4.3 Intelligent Environment Robot Task

For our final experiment, we utilize our variable autonomy approach to accomplish a retrieval task. This class of robot tasks is an important component of the MavHome smart home environment. The goal of MavHome is to view the home as an intelligent agent, able to make decisions to control the environment in a way that maximizes comfort for inhabitants while minimizing resource utilization (Das et al., 2002).

Robot agents in MavHome can perform a wide variety of assistive tasks. One such task is to retrieve an object at the request of an inhabitant. For example, a bedridden individual at home alone may request the robot to fetch some needed medicine when the person cannot get it himself. This experiment equips Astro with the capability of bringing medicine to a patient.

In this experiment, the patient is near the start point shown in Figure 6 and commands Astro to retrieve the medicine located at via-005. The task thus consists of navigating to point via-005, picking up the medicine, and returning to the start location.

In addition to the abstract navigation and low-level motion controller actions discussed in the previous sections, this application adds actions to search for a target object and to pick up the object. To accommodate retrieval tasks, we design a pink basket to hold small objects, which the robot can identify and lift with its head. Driven by high-level user commands, the robot arrives at via-005 as he performs the navigation task, then he needs to conduct the pickup action. Navigation is driven by the high and low-level control policies learned earlier. The robot then needs to identify the pink basket, and adjusts its position based on the current neck angle and the distance to the basket. After Astro adjusts his position, he uses his neck to pick up the basket with the medicine. Finally, Astro carries the medicine back to the start point where the patient is. Figure 7 shows Astro executing this task in the MavHome environment.

5 Conclusions and Future Work

To enable person assistive robot technologies to be used by general end-users, such as inhabitants of intelligent environments, user guidance at multiple levels of abstraction must be integrated into the robot learning task to speed learning and guide the robot policies toward

the user preferences. The control and interface approach presented in this paper attempt to address these issues by means of a formal control structure and through integration of various types of user commands into an autonomous reinforcement learning component, which provides the robot with variable modes of autonomy. We validate our approach with wheeled and walking robot tasks, including an intelligent environment task, that benefit from high-level and low-level user guidance.

Our future work in this area will provide additional modes of human/robot interaction such as imitation capabilities. The goal here is a system that can seamlessly switch between different modes of autonomy depending on the available user input, while maintaining operational safety. We will test the ideas on a greater variety of intelligent environment tasks, and will enhance the technique to permit user input to adjust the internal model of robot behaviors based on experimental feedback.

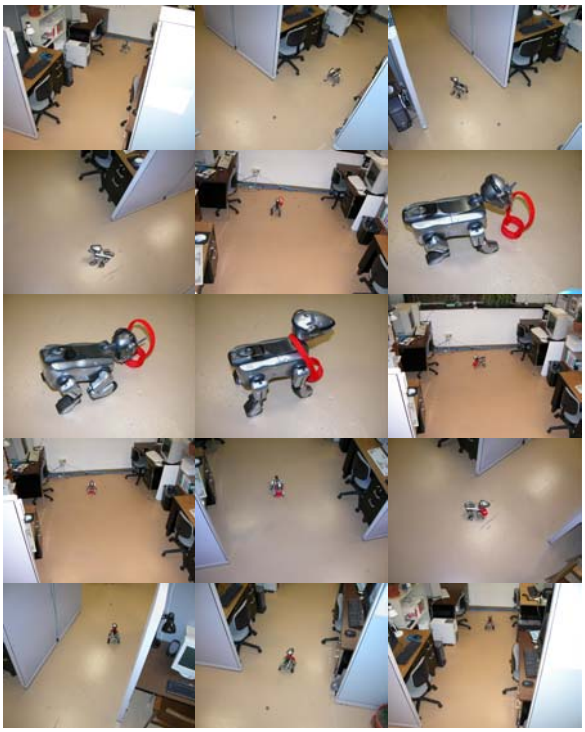

Figure 7. Astro performing MavHome retrieval task.

Acknowledgements

This work was supported in part by NSF IIS-0121297.

References

Barto, A. G., Bradtke, S.J., and Singh, S. P. 1993. Learning to act using real-time dynamic programming. Technical Report 93-02, University of Massachusetts.

Clouse, J., and Utgoff, P. 1992. A teaching method for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 92-101. San Mateo, CA: Morgan Kaufmann.

Connolly, C.I., and Grupen, R. A. 1993. The applications of harmonic functions to robotics. *Journal of Robotics Research*, 19(7):931-946.

Das, D. K., Cook, D. J., Bhattacharya, A., Heierman, E. O. III, and Lin, T. 2002. The role of prediction algorithms in the MavHome smart home architecture. *IEEE Wireless Communications*, 9(6):77-84.

Dorias, G., Bonasso, R. P., Kortenkamp, D., Pell, B., and Schreckenghost, D. 1998. Adjustable autonomy for human-centered autonomous systems on Mars. In *Mars Society Conference*.

Hexmoor, H., Lafary, M., and Trosen, M. 1999. Adjusting autonomy by introspection. Technical Report SS-99-06, AAAI.

Huber, M., and Grupen, R. A. 1999. A hybrid architecture for learning robot control tasks. In *AAAI 1999 Spring Symposium: Hybrid Systems and AI-Modeling, Analysis and Control of Discrete + Continuous Systems*. Stanford University, CA: AAAI.

Kaelbling, L. P., Peters, R. A. II, Johnson, C., Nilas, P., and Thongchai, S. 2001. Supervisory control of mobile robots using sensory egosphere. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation* 531-537. Banff, Alberta, Canada.

Papudesi, V. N., Wang, Y., Huber, M., and Cook, D. J. 2003. Integrating user commands and autonomous task performance in a reinforcement learning framework. In *AAAI 2003 Spring Symposium: Human Interaction with Autonomous Systems in Complex Environments*.

Papudesi, V. N. 2002. Integrating advice with reinforcement learning. M.S. Thesis, University of Texas at Arlington.

Ramadge, P. J., and Wonham, W. M. 1989. The control of discrete event systems. *Proceedings of the IEEE* 77(1):81-97.

Smart, W. D., and Kaelbling, L. 2000. Practical reinforcement learning in continuous spaces. In *Proceedings of the International Conference on Machine Learning*.

Watkins, C. J. C. H. 1989. Learning from delayed rewards. Ph.D. Dissertation, Cambridge University, Cambridge, England.

Wettergreen, D., Pangels, H., and Bares, J. 1995. Behavior-based gait execution for the Dante II walking robot. In *Proceedings of IROS*, 274-279. Pittsburgh, PA: IEEE.