To appear in the AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, 2003

Integrating User Commands and Autonomous Task Performance in a Reinforcement Learning Framework

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

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

Abstract

Making robot technology accessible to general end-users promises numerous benefits for all aspects of life. However, it also poses many challenges by requiring increasingly autonomous operation and the capability to interact with users that are generally not skilled robot operators. This paper presents an approach to variable autonomy that integrates user commands at varying levels of abstraction into an autonomous reinforcement learning component to permit faster policy acquisition and to modify robot behavior based on the preferences of the user. User commands are used here as training input as well as to modify the reward structure of the learning component. Safety of the mechanism is ensured in the underlying control substrate as well as by an interface layer that suppresses inconsistent user commands. To illustrate the applicability of the presented approach, it is employed in a set of navigation experiments on a mobile and a walking robot in the context of the MavHome project.

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 could perform repetitive and potentially dangerous tasks and assist in operations that are physically challenging for the user. However, moving robot systems from factory settings into more general environments where they have to interact with humans poses large challenges for their control system and for the interface to the human user. The robot system has to be able to operate increasingly autonomously in order to address the complex environment and human/machine interactions. Furthermore, it has to do so in a safe and efficient manner without requiring constant, detailed user input which can lead to rapid user fatigue (Wettergreen, Pangels, & Bares 1995).

In personal robots 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 instructions at high levels of detail. For the user interface and the integration of human input and autonomous control system this implies that they have to facilitate the incorporation of user commands at different levels of abstraction and at different bandwidth. This, in turn, requires operation at varying levels of autonomy (Dorais *et al.* 1998; Hexmoor, Lafary, & Trosen 1999) depending on the user feedback available. An additional challenge arises because efficient task performing strategies that conform with the preferences of the user are often not available *a priori* and the system has to be able to acquire them on-line while ensuring that autonomous operation and user 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 to 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 and their influence is limited such as to not prevent task achievement. As a result, the robot can seamlessly switch between fully autonomous operation and the integration of high and low level user commands.

In the remainder of this paper, the user interface and the 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 use of intermittent low level instructions using direct teleoperation are introduced. Finally, their use is demonstrated in the context of a navigation task with a mobile and a walking robot as part of the MavHome project.

Integrating User Input and Autonomous Learning for Variable Autonomy

The approach presented here addresses 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 be used to impose *a priori* safety constraints that limit the overall behavior of the system (Huber & Grupen 1999; Ramadge & 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, Bradtke, & Singh 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 in 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 reinforcement learning facilitates a seamless transition between user operation and fully autonomous mode based on the availability of user input. Furthermore, it permits user commands to alter the performance of autonomous control strategies without the need for complete specification of a control policy by the user. Figure 1 shows a high level overview of the components of the system.

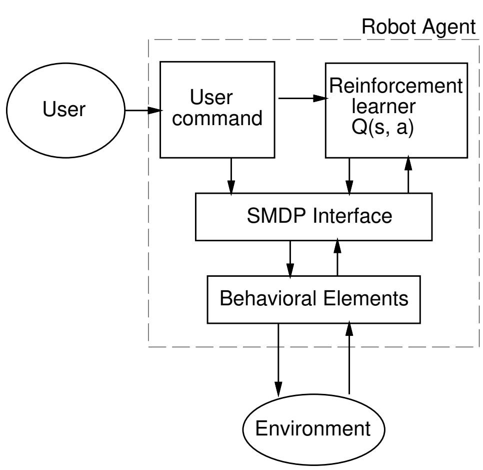

Figure 1: Overview of the Control System

In the work presented here, user commands at a higher level of abstraction are presented in the form of temporary subgoals in the SMDP model or by specifying an action 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 are used as teaching input to the learning component to optimize 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

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

where r is the reward obtained.

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.

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, the approach presented here uses the 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

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

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

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 specified by the task reward function r_T. This is important particularly in personal robot systems 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. Under these circumstances, the user rewards introduced above would cancel out and no longer influence the policy learned. Similarly, the user might give a sequence of commands which, when followed, form a loop 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 [1].

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

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

Experiments

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

Navigation Task

The goal of the robot navigation task used here to demonstrate the integration of user commands and autonomous learning component is to learn to optimally navigate the environment and reach a specific target. The environment itself consists of a set, V, of via points on a collection of maps consisting of a 50×50 grid of square cells. 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 DEDS Supervisor produces topological plans in the form of sequences of via points.

High-Level User Commands

To illustrate the capabilities of high-level user commands in the form of subgoals to accelerate learning and to modify autonomous behavior while avoiding unreliable user commands, two experiments were performed on the Pioneer 2 mobile robot shown in Figure 2.

[1] For details on the derivation of the bounds see (Papudesi 2002).

Figure 2: Pioneer Robot

Task Modification The first experiment is aimed at demonstrating the capability of the proposed approach to use sparse user input to modify the learned control policy in order to more closely reflect the preferences of the user. For this purpose, a navigation task was learned first and then a single user command in the form of an intermediate subgoal outside the chosen path was specified. Figure 3 shows the effect of the single user command on the learned policy for the navigation task.

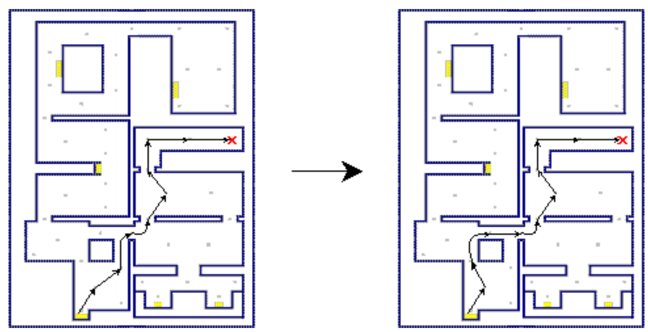

Figure 3: Change in Control Policy due to a User Command

In this figure the cross indicates the task-specific goal and the arrows show the optimal policy before the subgoal was provided by the user (left) and after the robot was told once to move behind the cubicle in the bottom left room (right). Figure 4 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 5 shows the robot performing the navigation task.

Filtering of Inconsistent User Commands The second experiment illustrates 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 6 shows the loop specified by the user commands and the learned loop-free policy that the robot executes after learning.

While the robot will execute the loop as long as the user explicitly commands it, it reverts to a policy that fulfills the

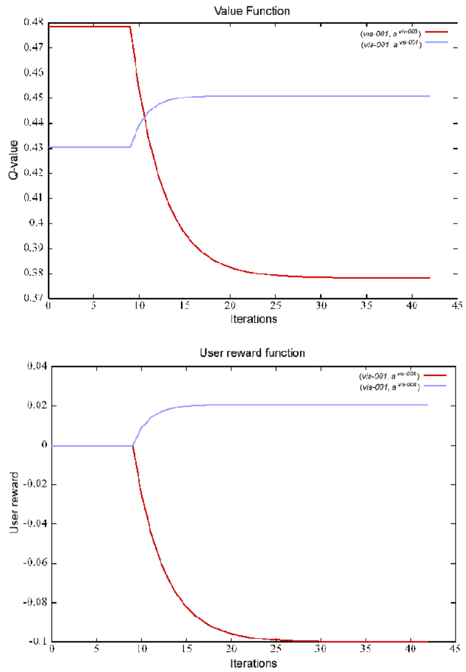

Figure 4: Change in the Q-value Function (top) and User Reward Function (bottom) due to a User Command

Figure 5: Pioneer Robot Navigating in the MavHome Lab

original task objective as soon as no further user commands are received. Figure 7 shows the corresponding Q-value and user reward functions.

These graphs show how the user reward function first modifies the behavior until it reaches the formal bounds imposed to prevent loops. Once the bounds are enforced, the Q-values of the actions re-establish the original order and the initially best action is chosen again, leading the robot to return to the initial path.

Multi-Level User Input

A second set of experiments was performed using a robot dog, Astro, operating in the MavHome smart home environment at the University of Texas at Arlington. 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. Figure 8 shows Astro.

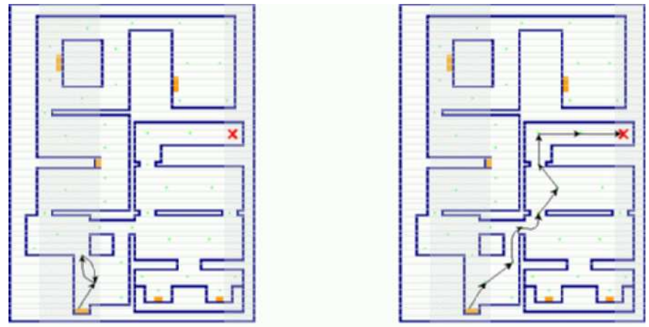

Figure 6: User Specified Loop (left) and Resulting Loop-Free Policy (right)

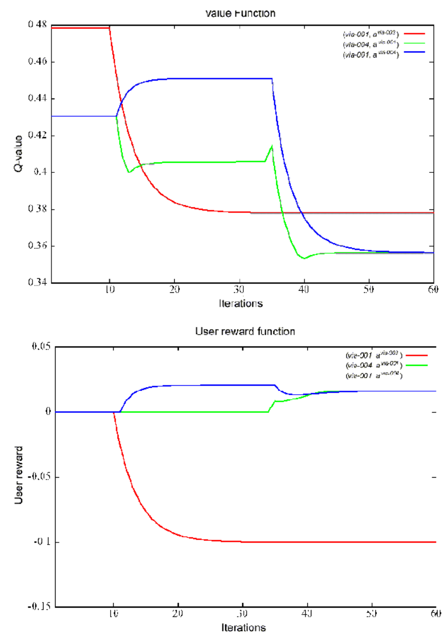

Figure 7: Change in the Q-value Function (top) and User Reward Function(bottom) due to a User Specified Loop

The goal of the MavHome project is to create a home that acts as an intelligent agent, perceiving the state of the home through sensors and acting upon the environment through device controllers (Das *et al.* 2003). The agent selects actions that maximize comfort of the inhabitants while minimizing utility costs. The MavHome architecture is hierarchical, so operations selected by the home agent may be handed to a lower-level agent to execute.

Astro represents one of the low-level MavHome agents. Initially, Astro will be trained to perform repetitive tasks such as fetching small items and conducting surveillance of the house. To accomplish any of these tasks, the first step is to equip Astro with the functionalities of navigating and locating himself in the house.

In this navigation task, user commands guide the robot at multiple levels of abstraction. First, user-specified subgoals

Figure 8: Robot Dog *Astro*

are provided in the form of 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 a wheeled robot implementation of the 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, dog. As a result, we provide three movement options for Astro: straight line, harmonic motion, or a two-segment path around corners and obstacles.

A reinforcement learning algorithm is used to select the movement pattern that is best for any pair of via points. Between any two via points, the user is allowed to select a direction for the robot to follow or allow the algorithm to select a motion consistent with the learned policy. If the user selects a direction, the dog moves in the given direction for a fixed distance. The executed path is compared with the path that would be generated using one of the three predetermined movement patterns, and the movement types are given reward based on the difference between planned and selected movement paths.

This algorithm is validated using Astro in a navigation task in the MavHome environment. 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 9.

In this experiment, point via-005 in Figure 9 is used as the goal. Initially, Astro chose via-003 as his first subgoal. 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 actually might cause him to approach the corner again. 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

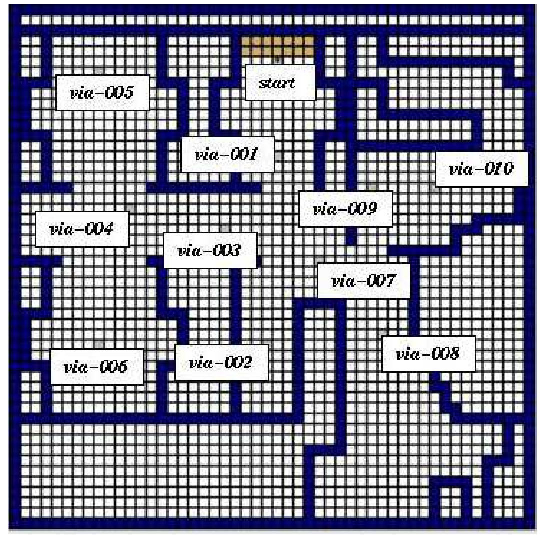

Figure 9: Map of the MavHome Lab with Via Points

then to the goal. When repeating the same task, Astro improves his movement efficiency based on the high-level and low-level feedback from the user. Figure 10 shows the robot dog executing the learned navigation strategy in the MavHome lab.

Conclusions and Future Work

To enable personal robot technologies to be used by general end-users it is important that operational safety can be assured and that a user interface is available that permits unskilled users to command the robot. The control and interface approach presented in this paper attempts to address these issues by means of a formal control structure and through the integration of various types of user commands into an autonomous reinforcement learning component which provides the robot with variable modes of autonomy. Here, user commands are used to train the robot and to modify task performance such as to more closely match the preferences of the user. At the same time, formal DEDS mechanisms and bounds on the user reward function are used to prevent inconsistent user commands from interfering with the overall task. The experiments presented in this paper show the effectiveness of this approach in the presence of high level commands in the form of subgoals as well as low-level commands provided by means of a joystick.

In the future, the approach will be further augmented by providing additional modes of human/robot interaction such as imitation capabilities. The goal here is a system that can seamlessly switch between different levels of autonomy depending on the available user input while maintaining operational safety. Furthermore, additional techniques will be investigated that permit user input to adjust the internal model of the robots' behaviors based on experimental feedback.

Acknowledgments

This work was supported in part by NSF ITR-0121297

Figure 10: Astro Performing a Learned Navigation Policy in the MavHome Lab

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, Amherst, MA.

Clouse, J., and Utgoff, P. 1992. A teaching method for reinforcement learning. In *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* 10(7):931–946.

Das, S. K.; Cook, D. J.; Bhattacharya, A.; Heierman, E. O. III; and Lin, T. 2003. The Role of Prediction Algorithms in the MavHome Smart Home Architecture. to appear in *IEEE Personal Communications*.

Dorais, 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.; Littman, M. L.; ; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4.

Kawamura, K.; 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.. 2002. *Integrating Advice with Reinforcement Learning*. M.S. Thesis, University of Texas at Arlington, Arlington, TX.

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 *Proc. IROS*, 274–279. Pittsburgh, PA: IEEE.