In Working Notes,AAAI Spring Symposium To Boldy Go Where
No Human- Robot Team Has Gone Before, 2006.

Demonstration-Based Behavior and Task Learning

Nathan Koenig and **Maja Matarić**
nkoenig|mataric@cs.usc.edu
Computer Science Department
University of Southern California
941 West 37th Place, Mailcode 0781
Los Angeles, CA, 90089-0781

Abstract

To be truly useful, robots should be able to handle a variety of tasks in diverse environments without the need for re-programming. Current systems, however, are typically task-specific. Aiming toward autonomous robots capable of acquiring new behaviors and task capabilities, we describe a method by which a human teacher can instruct a robot student how to accomplish new tasks. During the course of training, the robot learns both the sequence of behaviors it should execute and, if needed, the behaviors themselves. It also stores each learning episode as a case for later generalization and reuse.

Introduction and Motivation

Recent years have witnessed a promising increase in the prevalence of robotics in society. From vacuum cleaners to reconnaissance airplanes to autonomous vans driving in the dessert, robots are slowly entering human environments. However compelling these developments have been, they have required elite groups of highly skilled people to develop and maintain the robots. In contrast, typical consumers tend to gravitate towards technologies that work out of the box. Consequently, multi-purpose and easily reprogrammable robots will be the only economical and practical method to meet our ever-growing demands for automation. An easily programmable robot will allow domain experts to instill it with their expertise, and users to customize it to their personal needs.

While the benefits of such technology are obvious, its implementation is not. Programming must be simple and intuitive, and the storage, retrieval, and use of previous programming automatic. Creating a code library is not an option, as the human learning curve would be much too steep. An alternative we propose is *demonstration-based behavior and task learning coupled with case-based reasoning* (Carbonell 1986). In this approach, the robot is programmed much like an apprentice watching a master. Knowledge gained in such a manner is then stored in a case library for later use.

Learning new behaviors and tasks is not simple. Understanding and generalizing from observations of human actions or commands is a key requirement. The student robot must present an affable interface to the human teacher and still garner useful information from the teaching process. Ideally no technical skill requirements should be placed on the teacher. The teacher-learner interface must abstract away the details of the robot's sensors and actuators, while providing meaningful feedback to the teacher.

The learning mechanism must also store and generalize training instances for use in different future situations. After training, the robot will likely encounter different environments, component failures, and complex goals that require composition of various components of prior training examples. In all these cases the robot must leverage its stored knowledge to gracefully handle new situations.

This method of behavior and task learning presents a unique opportunity for humans to instruct robots in a manner not far removed from teaching other humans. Through an intuitive interface, a teacher can easily instruct an autonomous robot. These capabilities allow a robot to easily work with a human, while potentially learning ever more complex tasks. Building on these theories, one can move toward robot-robot teaching scenarios, and learning tasks that require multiple robots and/or humans.

Related Work

Programming by demonstration is a well studied topic for user interfaces, graphics applications, and office automation. Each of these systems attempts to learn action sequences by observing user actions. These action sequences are then applied when similar situations are encountered. In the robotic domain, programming by demonstration has been applied to industrial robotics (Münch *et al.* 1994). In this work, focus was placed on parameter learning of elementary operations. Besides parameter learning, robots have also learned behavior network representations for complex tasks. Nicolescu & Matarić(2003) developed such a system by using an interactive teaching methodology . Our work builds on this through the use of behavior networks, and adds the ability to learn completely new behaviors.

Imitation learning in humanoid robots is concerned with learning parametric models and policies of motion from human demonstrations. An articulated arm successfully learned an appropriate policy to balance a pole based on a reward function and task model (Atkenson & Schaal 1997). Biped humanoids have learned to achieve human-like locomotion through motion primitives of demonstrated trajectories (Nakanishi *et al.* 2004). General formalisms for performance metrics on humanoid imitation tasks has also been studied (Billard *et al.* 2004). This form of imitation allows articulated robots to learn complex gestures and motions by observing a human.

Our research also relies heavily on case-based reasoning for storage and retrieval of past experiences for use in new environments and tasks. More precisely, we use case-based reasoning in conjunction with behavior learning.Ram & Santamaria(1997; 1993) incorporated reinforcement learning with a case library to autonomously learn and select appropriate behavior parameters. In a similar manner, Likhachev, Kaess, & Arkin(2002) used gradient descent to tune behavior parameters that are stored in a case library indexed on environmental features. Both these techniques focus on autonomously optimizing parameters of a static set of behaviors for varying environmental conditions. Our goal, on the other hand, focuses on learning the behaviors and their temporal ordering.

An essential component of many robotic systems is a planning strategy. Our research takes a means-ends analysis approach that was derived from Veloso's work on PRODIGY/ANALOGY (Manuela M. Veloso 1993). That system relied on case-based reasoning and derivational analogy to create high-level robot plans using means-ends analysis and backward chaining. A graphical user interface, GUI, was also developed that allowed a user to interact with the planning process (Cox & Veloso 1997). This incorporated human decisions into the planning process. While we leverage human expertise at the teaching phase rather than in the planning, this work demonstrates the usefulness and practicality of allowing a human to actively interact with an intelligent system.

An important aspect of our research is the ability of the robot to learn continually. This notion of *lifelong learning* relies on past knowledge to achieve complex and extended goals. Thrun(1994) successfully used Q-learning with an explanation based neural network to store, reuse, and modify prior knowledge to inductively learn functions. In a similar vein, we store and reuse past knowledge to improve performance and increase the usefulness of a robot.

The work presented in this paper can be viewed as merging teaching by demonstration, lifelong learning, case-based reasoning, and behavior-based robotics. We are strongly motivated towards increasing the usefulness of robots by improving their abilities to interact with and learn from humans. Each of the above listed components plays a vital role toward meeting this goal. While successfully used independently, we believe if properly combined in a hierarchical system they have the potential to move intelligent agents from sterile research environments into real-world scenarios with direct human collaboration.

Overview

The method we employ is hierarchical in nature, incorporating teaching, behavior-based (Arkin 1998; Matarić 1997) learning, and case-based memory. Throughout this paper, teaching (training) is assumed to be between a single human teacher and a single robot student. The teacher passes instructions to the student in the form of commands to execute that are understandable by the robot, such as "move forward 5m".

A single training episode (see Figure 1) entails placing the robot at a pre-defined start state and instructing (controlling) it to a specific goal state. From this training instance, the robot generates a set of behaviors. Each behavior generalizes a portion of the task execution into a self-contained, time-extended, goal-oriented controller.

The generated behavior network is stored in a case library for later use. Each training example creates a single behavior network which is encapsulated into a case, and labeled by the start and goal states. Cases are unique and indexed based on their label. Following training, a stored case can be activated when the robot's current state and user-provided goal state match a label in the case library (see Figure 2).

The above-described process allows a teacher to easily instill knowledge about a specific task. In this context, we use the term "knowledge" to refer to the state of knowing how to act in relevant situations, which is a set of labeled and composable behavior networks. We do not incorporate any "forgetting" into the system at this time; thus the robot can learn up to the capacity of its memory.

State Representation

The robot has access to two types of state information: sensory and model-based. Sensory information consists of data from local physical sensors on the robot. Model-based information consists of processed local sensor data as well as global data, such as a map of the

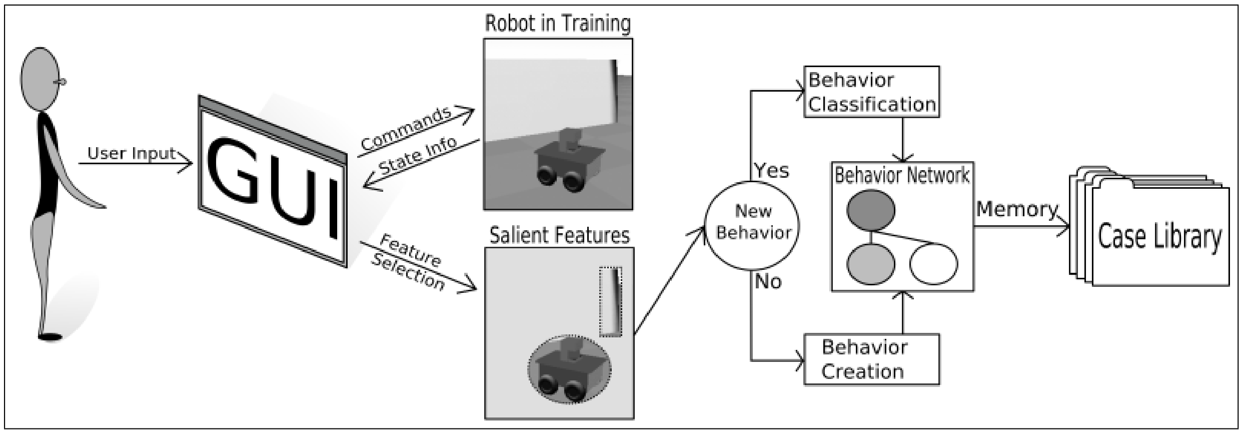

Figure 1: Information flow during training. The teacher instructs the robot how to act using a GUI interface, and receives feedback in the form of robot's state information. When salient features are indicated by the teacher, the robot classifies them or creates new behaviors. New behaviors are added into a control network representing the learning trial, which is labeled and stored in a case library for later use.

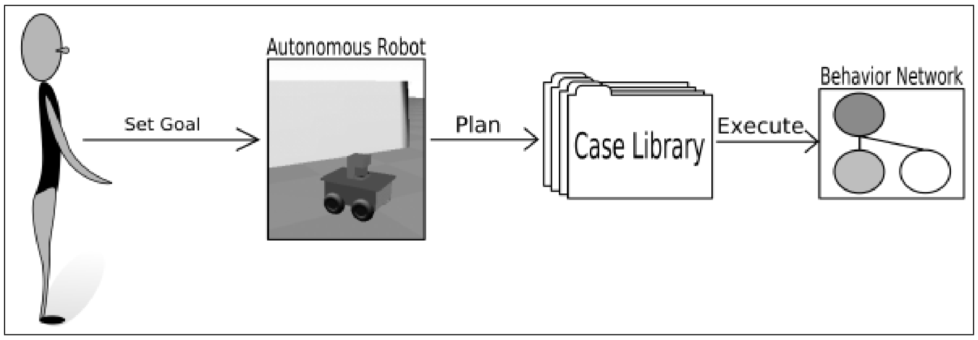

Figure 2: In autonomous mode, the robot receives a goal from the user and derives a strategy to achieve that goal by using its case library. The strategy is a series of one or more cases. In situations when the robot is unable to derive a complete strategy, it will notify the user of the error and await new instructions.

environment, portion of the map explored, and important locations within the map. The amount and type of model-based information available to the robot must be decided prior to training.

The state space for the robot is large. Therefore, a subset of the state data is used by the robot during learning. The subset is composed of the most important, i.e., *salient features* used by the teacher when a control decision is made. A *feature* refers to an individual element of the complete state description, such as the robot's position or the current laser range scan.

Salient features are recognizable either directly or indirectly. Direct feature recognition occurs when the robot actively asks the teacher to indicate important features in the state space. When posed this type of question, the teacher must select a subset, ideally a minimal one, of the current state information used to make the particular control decision. Indirect feature recognition occurs when the robot has seen a particular state and action pairing before. It can then determine the similar features between the previous and the current instances.

The work in this paper is applied to (but not limited to) the domain of mobile navigation. In that context, we make a few assumptions concerning what state information is available to the robot. Specifically, a map of the environment is provided, and the robot is capable of localizing itself within the environment.

Human Robot Interface

Our method of teaching a robot requires an intuitive interface between the teacher and the student. The interface must inform the teacher of the robot's current state, allow the teacher to instruct the robot, and be simple to use. In order to meet these demands we rely on a standard personal computer running a graphical user interface, GUI, that processes raw state information and displays it in a meaningful way.

The GUI must be able to render all the state information either as an image or text. The rendering method is dependent on the feature type. In many cases, a feature is represented as an image or animation. For example position information and range data are displayed on a map, battery levels as a bar meter, and camera data as images.

A mechanism must also exist for controlling the robot during teaching. In this case the robot actuators must map to user input in an intuitive way. Since we have assumed the robot has a map of the environment which is displayed by the GUI, the user can easily control the robot's position by indicating map locations. These global positions are interpolated into a trajectory for the robot to follow. This abstraction allows the teacher to ignore the dynamics and kinematics of the robot hardware, and instead focus on the learning task. Similarly, grippers can be controlled by a set of buttons that indicated opening, closing, and lifting. Such interfaces must be designed for each actuator the robot possesses.

The final requirement of the interface involves a method by with the robot can query the teacher for salient features. When this query occurs, the user is shown a standard message indicating a request to select a set of features currently visible on the GUI. In response to this message, the user highlights the most important features and acknowledges a completed response to the query.

Learning

Our method of learning is focused on two important issues. The first is understanding demonstrations provided by a human teacher. A suitable mechanism is required to accurately translate actions performed by a human into knowledge understandable by a robot. Such a mechanism must be transparent to the instructor and general enough to capture relevant information across a significant spectrum of domains.

The second and related point lies in generalizing the knowledge gained from a teacher, efficiently storing the knowledge, and reusing it at appropriate occasions. This must also take place in real time without human intervention.

These two issues can be labeled as knowledge acquisition and knowledge understanding. In both cases minimal burden is placed upon the teacher. This attempts to maximize the ease of teaching without reducing its effectiveness, thus creating more intuitive and natural method of collaborating with a robot.

Behaviors

We take a *behavior-based* approach throughout our work. Behaviors are time-extended actions that achieve or maintain a set of goals. These behaviors can describe fairly basic concepts such as `avoid-obstacles` and `follow-wall`, as well as more complex such as `find-box` and `locate-target`. They provide a suitably general definition to classify demonstrated actions, and can be specific enough to achieve meaningful tasks.

The robot starts with a set of basic behaviors, such as `avoid-obstacles`, `localization`, `follow-wall`. This set does not define all the behaviors the robot will require, but rather gives the robot a rudimentary set of skills to bootstrap the training process.

Every behavior starts with a set of *preconditions* and ends with a set of *postconditions*. Preconditions and postconditions consist of one or more state features. The precondition defines a certain state that the robot must be in before the behavior is activated. The postcondition defines the state to reach or maintain.The behaviors are organized into a behavior network (Nicolescu & Matarić 2002), where links between behaviors specify their temporal ordering within a well-defined temporal logic. A behavior cannot activate unless its predecessor behavior has been active. The resulting behavior network represents the complete strategy (plan) used by the teacher to move from the start state to the goal state.

Behavior Matching

The human teacher does not need to understand what behaviors are or how they work in order to instruct a robot. The teacher simply commands the robot to perform the necessary actions to complete a task. During such a demonstration, the robot must learn the important features in the state space and how to respond to them. The approach we take decomposes the continuous action and sensory information during a demonstration into discrete behaviors that are organized into a network representing the complete task plan.

The first step in the decomposition process is to recognize salient features in the state space. We are only concerned with features punctuated temporally by a user command. An instruction from the teacher is significant, as it marks a point in time when a strategic decision was made. This decision must have been based on a subset of the robot's state space, i.e., a set of features. It is therefore incumbent upon the robot to determine what features the teacher used during their decision process.

The second step involves matching behaviors to the time period between salient features. We make the assumption that actions between salient feature sets are similar enough to group into a single behavior. This assumption allows the robot to reduce a continuous task into a significantly smaller discrete behavior representation.

Matching behaviors to time periods is done by selecting a behavior whose preconditions and postconditions match the first and second salient feature sets delimiting the time period. The behaviors are further constrained by the order in which they were encountered during training. For example, if behavior B occurs after behavior A, then B has the constraint that A must be active immediately before B.

These rules allow the robot to generate a behavior network, thereby describing a continuous training example as a discrete set of behaviors. Each node in this network is a single behavior generated from the training example. Links between the nodes define the activation preconditions and postconditions.

Behavior Creation

A robot that is only capable of behavior matching is limited to learning tasks that can be composed of pre-programmed behaviors. This is a major constraint on the robot's breadth and depth of learning. We approach this problem by allowing the robot to autonomously generate new behaviors.

The process by which new behaviors are generated is similar in many respects to behavior matching. When behavior matching fails, the robot defines a new behavior with the preconditions and postconditions of the start and ending salient features of the non-matching period. With the conditions established, the new behavior now requires a controller.

The controller's role is to execute actions necessary to achieve the goals of the behavior. Typically, a behavior's controller is pre-programmed and remains static. However, we require the robot to autonomously create a controller. In order to accomplish this the robot must know how its actions affect its state and the order in which the actions should be executed.

For the robot to know the affects of its actions, it must have a model of its state/action mapping i.e., its controller. Such a model should be learned, rather than hard coded, as a separate process prior to teaching due to its complexity and dependence on the structure and capabilities of the robot. Pierce & Kuipers(1997) presented a solution to this problem through the use of a sequence of statistical and generate-and-test methods to learn a hierarchical model of the robot's sensorimotor apparatus . Alternatively, Schmill *et al.*(1998) learned context dependent decision trees that related primitive actions to changes in sensor readings. We will look to these works as inspiration for our learning method, which is not the core of our approach.

The order in which actions should be executed is described by the changes in features seen during teaching. A complete history of the features is logged between the start and end of the behavior. This history describes the sequence of changes in the state features. With this information a function can be fit to each salient feature. The functions are assumed to be linear, and can be approximated using standard techniques such as least squares.

With the preconditions and postconditions, a controller model, and a means to generate a behavior, the robot has enough information to compose a completely new behavior. Pre-programmed and new behaviors are treated identically. The former are inserted into the behavior network according to the same rules, and can be reused as appropriate.

Case-based Memory

When the behavior network is generated, the training instance must be stored in memory. A memory component allows a robot to store, reuse, and adapt previous learning.

The robot's memory relies on a case-based knowledge library. Each training instance is considered a single case and is stored in the case library. Cases in the library are stored and retrieved via an index. The initial start state of the training example combined with the final, or goal, state form the index for a case. The case contains the behavior network generated from the training data. With a library of cases, a robot can use its current state and goal information to select an appropriate case and execute the associated behavior network.

The robot maintains a single case for each start and goal state pair. In other words, the robot maintains only a single strategy for a case. After initial instruction, the teacher is able to overwrite and modify the previous training through re-training. This simplifies the case library based on the assumption that humans intend to teach the robot the best strategy. It is also possible to have the robot learn multiple alternative strategies; our continuing work will address that extension.

Planning

When the robot is presented with a new task to perform, it must plan a strategy for reaching the task goal given its start state. A strategy is formed by first decomposing the task into a set of subgoals. This decomposition is achieved using *means-end analysis* based on the case-library. The selected cases are executed in order, as long as they remain valid. If during execution a case becomes invalid, either by the goal changing or the state changing, the robot must replan.

It is probable that the robot will encounter situations where it will be unable to decompose a task into a set of subgoals. This will occur whenever the robot has not received sufficient training to reach the goal. In these situations the robot will inform the user that it has insufficient knowledge to complete the task, and will await further instruction.

Even with a complete plan, the robot will likely enter error states where it is unable to reach a subgoal. This can result from improper training, changes in the environment, and device failures. In situations where the robot is still functional, it will stop and regenerate a plan. If it is unable to successfully create a new case plan, the robot will backtrack to the last case's goal state and attempt to replan. The robot will continue to backtrack until it can successfully create a new case plan, or reach the start state, at which point the robot will declare a general failure.

Algorithm 1 Runtime Algorithm

```
Require: $start\_state$, $goal\_state$
 1: $current\_state = start\_state$
 2: $case\_plan \leftarrow$ backward chaining
 3: while $current\_state != goal\_state$ do
 4:     $current\_case = top(case\_plan)$
 5:     $sub\_goal\_state \leftarrow current\_case$
 6:     if $current\_state != sub\_goal\_state$ then
 7:         $behavior \leftarrow behavior\_network$
 8:         execute $behavior$
 9:     else
10:         $pop(case\_plan)$
11:     end if
12: end while
```

Teaching

Teaching begins by placing the robot in the relevant environment with a given set of starting state features. The teacher sends commands to the robot using the GUI until it reaches the desired goal feature set.

It is important for the robot to recognize salient features in the environment. These features indicate important events, and occur when the teacher instructs the robot. The robot capitalizes on this fact by recording when the teacher sends instruction to it, and may ask the instructor what subset of the total current state features was used to make the instructional decision.

Salient features are difficult for a learning agent to autonomously recognize. Rather than guess an appropriate feature set and risk learning incorrect behavior, the robot will query the instructor. The robot will simply ask the teacher to indicate the most important features used to decide what action the robot should take. These salient features are stored along with the commanded action.

Constantly questioning the instructor is bothersome. To help alleviate this problem, the robot first attempts to select the salient features automatically. The robot searches its history of received commands and salient features. If a similar command is found paired with a similar features, then the same feature set is applied to the current situation.

The robot processes new salient features by matching behaviors. All feature information between the previous and current salient feature sets are classified as a behavior. The behavior's preconditions and postconditions are set to the two boundary feature sets, respectively. Finally, the behavior is inserted into the network by creating a link from the previous behavior to the newly matched behavior. This behavior network maintains a certain probability of correctly modeling the teacher's actions. Upon the completion of training, the behavior network has essentially reduced the continuous state information into a discrete set of behaviors that conditioned on a few salient features.

We make a few assumptions concerning the training process. First, training should be incremental in task complexity. The robot should be taught simple tasks before complex ones. The robot uses previous training knowledge during new training instances. It is likely that a complex task can be decomposed into a series of sequential simple tasks. By teaching the easier tasks first, there may be no need to teach the complex task as the robot's planning mechanism can produce the necessary task plan. Also, the simple tasks can be combined in numerous ways, whereas a complex task is unlikely to be generally applicable.

It is also assumed the teacher will never purposefully provide incorrect instructions. The learning mechanism does not attempt to detect malicious behavior. The teacher is also required to indicate salient features when requested. Failing to do so will result in the robot learning potentially incorrect behaviors.

The environment in which robots are taught can be either in simulation or the real world. Simulated environments are desirable for teaching simple tasks that require simple worlds. Simulations also reduce the complexity of the teaching scenario by eliminating potential hardware problems. They however present an idealized world to the robot. Therefore, tasks that require complex and dynamic worlds should be taught in real environments. In these, the teacher will also receive important feedback on sensor and actuator noise and how the robot behaves in dynamic environments. With this data, the teacher can alter the training process to better match what the robot is likely to encounter in the future.

Teaching is accomplished by providing a teacher with a mechanism to control the robot through which the robot can be moved towards a desired goal. This control mechanism is an interface through which the teacher can visualize the current state of the robot and issue commands. It is important that the teacher uses only information also available to the robot. If the teacher uses other information from the environment, for example from the human vision system, the teacher will likely make control decisions that the robot is incapable of understanding and learning. Based on this constraint, the teacher must use a GUI that displays the robot's sensor and model-based state information.

Example

The purpose of this example is to illustrate the goal of this research and how the proposed system will function. For simplicity, we have chosen an exploration task

Algorithm 2 Teaching Algorithm

```
Require: $start\_state$, $goal\_state$
 1: $current\_state = start\_state$
 2: $salient\_features = \{\phi\}$
 3: while $current\_state != goal\_state$ do
 4:   if $!new\_instruction$ then
 5:     continue
 6:   end if
 7:   $salient\_features \subset current\_state$
 8:   if $salient\_features == \{\phi\}$ then
 9:     query instructor for salient features
10:   end if
11:   $behavior =$
12:     $classify(salient\_features_{t-1...t})$
13:   if $behavior == \{\phi\}$ then
14:     $behavior =$
15:       $create(salient\_features_{t-1...t})$
16:   end if
17:   $network\_insert(behavior)$
18: end while
```

in which the robot's objective is to maximize the observed area of a rectangular enclosed area. The robot is a two wheeled mobile base with a scanning laser range finder, and makes use of preprogrammed obstacle avoidance and localization algorithms.

The first step involves teaching the robot how to explore. This is accomplished by first placing the robot at a start location, observing the robot's state information, and then passing commands to the robot. These instructions are high level in nature, and take the form of ¡x,y,theta¿ goal poses. As commands are received, the robot may query the user as to which features are currently salient. The teacher responds via the GUI by selecting which features most heavily influenced their command choice. Teaching proceeds until the robot has sufficiently explored the enclosed area. Sufficiency is determine by the teacher.

Upon completion of the training exercise, the robot processes logged information. The sequences of data are matched to known behaviors through behavior matching. If a sequence cannot be matched a new behavior is created through behavior creation. During this process of matching and creation the behavior network is created based on the order in which the behaviors were encountered. The final network is stored in the robot's case library with an index corresponding to the start state and goal.

For our example of exploration of an enclosed area, we will assume the teacher used a wall following strategy. With this strategy, the teacher may command the robot to move towards the first corner indicating the salient features include distance to the wall parallel to the direction of travel and distance to the corner. This process is repeated for the next three corners. Once the robot has again reached its starting location, the teacher indicates that training is complete.

Using the logged information, the robot is able to break the data into four sequences, one for each side of the room. With no matching behaviors, the robot creates a new behavior for the first sequence. The robot is now able to match the next three sequences with the newly created behavior. The resulting network consists of a single behavior that follows a wall to a corner, a link from the behavior back to itself, and a stopping condition when the current state again equals the start state. This behavior network is then inserted into the case library. When the robot next encounters a situation where it is next to wall with a corner in sight and a goal of exploration, it can use this stored case.

Let us now suppose the teacher takes a different strategy towards exploration. Instead of following the walls, the teacher instructs the robots to perform a zig-zag pattern down the room. In this case the teacher would indicate the salient feature is distance to the next wall perpendicular to the direction of motion. The robot would in turn learn a new behavior where it moves towards an opposing wall, turns almost completely around, and repeats until it runs into the end of the room.

Ideally the robot will be taught simple tasks first, upon which it can learn more complex tasks. For example if the robot has learned exploration behaviors and a behavior to pick up objects, it can more easily learn a task involving finding an object and bring it back to a specific location. In this case object identification must be learned, however exploration and picking up an object it already knows how to do.

It will often be the case that new behaviors must be learned during a training example. These behaviors can be difficult to learn without active human interaction which reduces the feature space and provides the robot with accurate task examples. Storage and reuse of these behaviors allows the robot to handle more complex tasks, and also eases the burden on the teacher.

Future Work

This research has important limitations that will be addressed in our continuing work. The first of these is the constraint of a single strategy per case. We currently maintain a case library where each case is unique and each case has only one behavior network. With these constraints, a robot is capable of learning only one solution for a case. Ideally the robot would store numerous labeled solutions for a case, selecting the one which is most appropriate.

Secondly, we have defined the teaching interface to be composed solely of robot state information. We would eventually like to display this information on a portable device. This would give the teacher a better understanding of the environment, and make the teaching process more personal and interactive.

Conclusion

This paper has described a novel method of interactively teaching robots how to perform complex tasks. With this architecture, the human teacher takes on a more personal and intuitive role in "programming" a robot. This is meant to improve the effectiveness of the robot, increase its usefulness and longevity, and allow for a maximal number of teachers to impart knowledge into the robot.

Ideally we see this research aiding in the transition of robots from single purpose and highly specialized machines toward general purpose tools that are useful and simple. In the short term, this work creates interesting opportunities for robots to work closely with and aid humans in specialized fields such as search and rescue, exploration, and construction.

References

Arkin, R. C. 1998. *Behavior-Based Robotics*. CA: MIT Press.

Atkenson, C. G., and Schaal, S. 1997. Robot learning from demonstration. In Douglas H. Fisher, J., ed., *Machine Learning: Proceedings of the Fourteenth Internation Conference (ICML '97)*, pp. 12–20. San Francisco, CA: Morgan Kaufmann.

Billard, A.; Epars, Y.; Calinon, S.; Cheng, G.; and Schaal, S. 2004. Discovering optimal imitation strategies. *Robotics and Autonomous Systems* 47:2-3.

Carbonell, J. G. 1986. *Machine Learning, An Artificial Intelligence Approach*, volume II. Morgan Kaufman. chapter Derivational analogy: A theory of reconstructive problem solving and expertise acquisition., 371–392.

Cox, M. T., and Veloso, M. M. 1997. Supporting combined and human and machine planning: An interface for planning by analogical reasoning. In Leake, D., and Plaza, E., eds., *Case-Based Reasoning Research and Development, Proceedings of ICCBR-97, the Second International Conference on Case-Based Reasoning*, 531–540. Providence, Rhode Island: Springer Verlag.

Likhachev, M.; Kaess, M.; and Arkin, R. 2002. Learning behavioral parameterization using spatio-temporal case-based reasoning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, 1282–1289. Washington, DC: IEEE.

Manuela M. Veloso, J. G. C. 1993. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning* 10(3):249–278.

Matarić, M. 1997. Behavior base control: Examples from navigation, learning, and group behavior. *Experimental and Theroretical Artificial Intelligence* 9(2-3):323–326.

Münch, S.; Kreuziger, J.; Kaiser, M.; and Dillmann, R. 1994. Robot programming by demonstration (rpd) - using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. In *25th International Symposium on Industrial Robots (ISIR '94)*, 685–693.

Nakanishi, J.; Morimoto, J.; Endo, G.; Cheng, G.; Schaal, S.; and Kawato, M. 2004. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems* 47(2-3):79–81.

Nicolescu, M., and Matarić, M. J. 2002. A hierarchical architecture for behavior-based robots. In *International Joint Conference on Autonomous Agents and Multiagent Systems*.

Nicolescu, M., and Matarić, M. J. 2003. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *International Joint Conference on Autonomous Agents and Multiagent Systems*.

Pierce, D., and Kuipers, B. J. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* 92(1-2):169–227.

Ram, A., and Santamaria, J. C. 1993. A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robot navigation. In *Second International Workshop on Multistrategy Learning*.

Ram, A., and Santamaria, J. C. 1997. Continuous case-based reasoning. *Artif. Intell.* 90(1-2):25–77.

Schmill, M. D.; Rosenstein, M. T.; Cohen, P. R.; and Utgoff, P. 1998. Learning what is relevant to the effects of actions for a mobile robot. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pp. 247–253. Minneapolis, Minnesota, United States: ACM Press.

Thrun, S. 1994. A lifelong learning perspective for mobile robot control. In *IEEE/RSJ/GI Conference on Intelligent Robots and Systems*, 23–30.