A Framework for Learning from Observation Using Primitives

Darrin C. Bentivegna[1,2] and Christopher G. Atkeson[1,3]

[1] ATR, Human Information Science Laboratories Department 3, Kyoto, Japan
[2] College of Computing, Georgia Institute of Technology, Atlanta, GA
[3] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

Abstract. This paper describes a method to learn task primitives from observation. A framework has been developed that allows an agent to use observed data to initially learn a predefined set of task primitives and the conditions under which they are used. A method is also included for the agent to increase its performance while operating in the environment. Data that is collected while a human performs a task is parsed into small parts of the task called primitives. Modules are created for each primitive that encode the movements required during the performance of the primitive, and when and where the primitives are performed.

1 Introduction

Learning without any prior knowledge in environments that contain large or continuous state spaces is a daunting task. For agents that operate in the real world, learning must occur in a reasonable amount of time. It is essential for an agent to have domain knowledge is if it to learn in the time scales needed [14]. Providing an agent with domain knowledge and also with the ability to use observation data for learning can greatly increase its learning rate. This paper describes a framework in which to conduct research that explores the use of primitives in learning from observation.

Virtual and hardware environments of air hockey and a marble maze game, figures 1 and 2, have been created as platforms in which to conduct this research. The virtual environments were first created and provided an invaluable tool in which to test algorithms that will be run on the hardware version. For this reason the physics of the virtual environments are programmed to match those of the hardware versions as much as possible. In all these environments the position data can be collected as a human operates in the environment. The architecture and current playing strategy of these environments will be described.

1.1 Primitives

Robots typically must generate commands to all their actuators at regular intervals. The analog controllers for our 30-degree of freedom humanoid robot are given desired torques for each joint at 420Hz. Thus, a task with a one second duration is parameterized with $30 * 420 = 12600$ parameters. Learning in this high

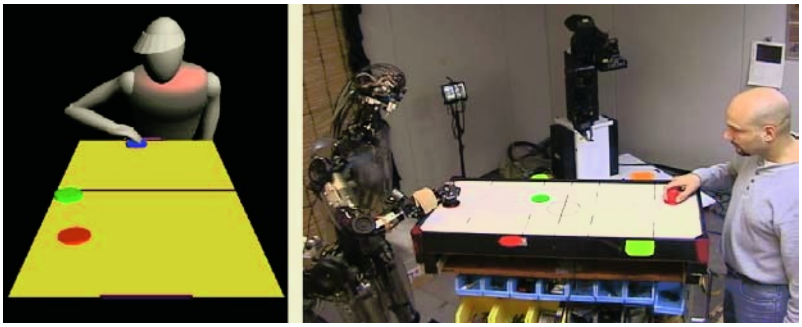

Fig. 1. The virtual air hockey environment on the left and the hardware version on the right.

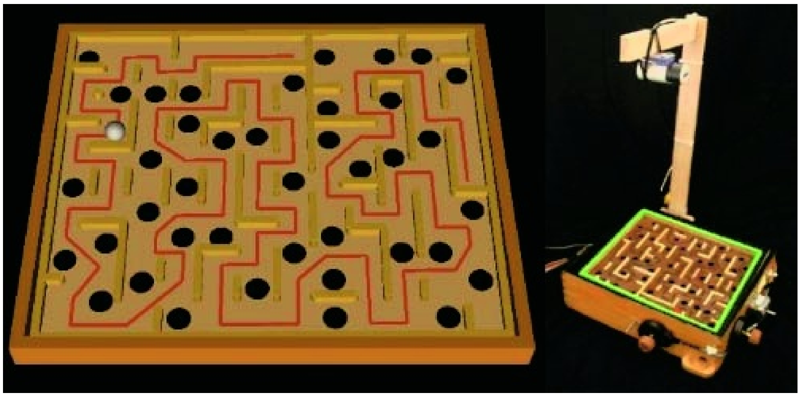

Fig. 2. The virtual marble maze game on the left modeled after the hardware version on the right.

dimensional space can be quite slow or can fail totally. Random search in such a space is hopeless. In addition, since robot movements take place in real time, learning approaches that require more than hundreds of practice movements are often not feasible. Special purpose techniques have been developed to deal with this problem, such as trajectory learning [2], learning from observation [4, 5, 9, 13, 6, 8, 10, 11], postural primitives [17], and other techniques that decompose complex tasks or movements into smaller parts [3, 7, 15].

It is our hope that primitives can be used to reduce the dimensionality of the learning problem [3, 16]. Primitives are solutions to small parts of a task that can be combined to complete the task. A solution to a task may be made up of many primitives. In the air hockey environment, for example, there may be primitives for hitting the puck, capturing the puck, and defending the goal. In this research a task expert predefines the set of primitives to be used for a given environment and algorithms are created to find the primitives in the captured data.

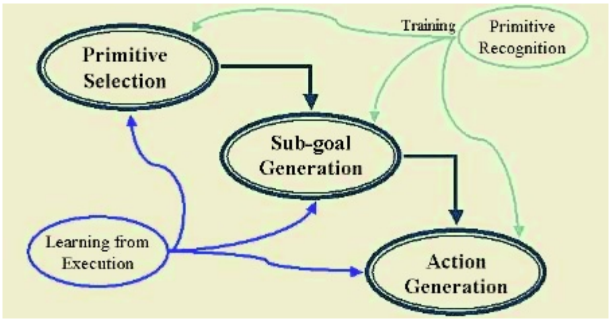

Fig. 3. Our view of a primitive.

1.2 Perceiving the Primitives

Since the observed data is continuous it must first be segmented into primitives. To accomplish this, critical events are used. Critical events are easily observable occurrences. Examples of critical events for the puck include collisions, in which the ball speed and direction are rapidly changed, and the ball traveling in a straight line with decreasing velocity. Algorithms have been created that find the primitives within the data by searching for the proper sequence of critical events.

1.3 Strategy for Primitive Use

Figure 3 shows our view of a primitive. Currently, a human, using domain knowledge, designs the candidate primitives that are to be used. The primitive recognition module segments the observed behavior into the chosen primitives. This segmented data is then used to provide the encoding for the primitive selection, sub-goal generation, and action generation modules.

The primitive selection module will provide the agent with the primitive to use for the observed state of the environment. After it has been decided which primitive to use, the desired outcome, or goal, of that primitive is specified by the sub-goal generation module. Lastly the actuators must be moved to obtain the desired outcome. The action generation module finds the actuator commands needed to execute the chosen primitive type with the current goal.

After the agent has obtained initial training from observing human performance, it should then increase its skill at that task through practice. Up to this point the agent's only high-level goal is to perform like the teacher. Its only encoding of the goal of the entire task is in the implicit encoding in the primitives performed. The learning from practice module contains the information needed to evaluate the performance of each of the modules toward obtaining a high-level task objective. This information can then be used to update the modules and improve performance beyond the teacher.

2 Air Hockey Environment

Figure 1 shows the virtual air hockey game created that can be played on a computer. A human player using a mouse controls one paddle. At the other end is a simulated or virtual player. The movement of the virtual player has been limited to match that of the humanoid robot DB (www.erato.atr.co.jp/DB/). Spin of the puck is ignored in the simulation. The position of the two paddles and the puck, and any collisions occurring within sampling intervals are recorded.

The hardware implementation, figure 1, consists of the humanoid robot, a small air hockey table, and a camera based tracking system. The robot observes the position of the ball using its on board cameras and hardware designed to supply the position of colored objects in the image. The humanoid's torso is moved during play to extend the reach of the robot. The head is moved so that the playing field is always within view.

The full list of primitives currently being explored in the air hockey environment is:

– Left Hit: the player hits the puck and it hits the left wall and then travels toward the opponent's goal.
– Straight Hit: the player hits the puck and it travels toward the opponent's goal without hitting the side walls.
– Right Hit: the player hits the puck and it hits the right wall and then travels toward the opponent's goal.
– Block: the player deliberately does not hit the puck but instead moves into a blocking position to prevent the puck from entering their goal.
– Prepare: movements made while the puck is on the opposite side from the player. The player is either preparing to setup for a shot, or preparing to defend their goal.
– Multi-Shot: movements made after a shot is attempted, but while the puck is still on the player's side. If the puck is not quickly moving toward the opponent's side, they will have the opportunity to hit it again.

3 Marble Maze Environment

In the marble maze game a player controls a marble through a maze by tilting the board that the marble is rolling on. The hardware board is tilted using two knobs and the virtual board is controlled with the mouse, figure 2. There are obstacles, in the form of holes, that the marble may fall into, and walls. In both versions the time and the board and ball positions are recorded as a human plays the game. The virtual game models the movement of the marble and treats collisions with the wall aesthetically with a significant loss of energy. The human controls the board on the hardware version by using knobs connected to encoders. The motor command generated by the encoder system is read by the computer and sent to the motors. The position of the ball is obtained using a Newtonlabs Cognacrome vision system [1]. The computer can also generate its own commands and send them to the motors.

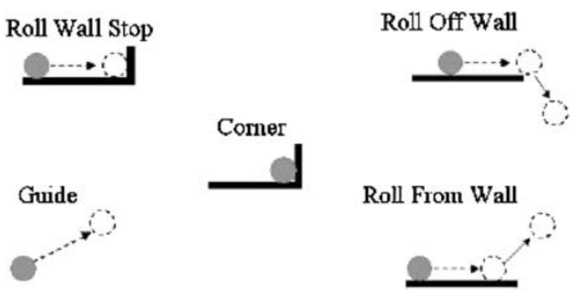

Fig. 4. Primitives used in the marble maze environment.

The primitives for the marble maze game are designed to give the agent the skills it will need to perform the task. The following primitives are currently being explored and are shown in figure 4:

- Roll Wall Stop: The ball rolls along a wall and stops when it hits another wall.
- Roll Off Wall: The ball rolls along a wall and then rolls off the end.
- Guide: The ball is moved without touching a wall.
- Roll From Wall: The ball hits, or is on, a wall and then is maneuvered off it.
- Corner: The ball is in a corner and the board is being positioned to move the marble from the corner.

4 Selecting the Appropriate Primitive and Sub-goal

As discussed in the strategy above, it is the responsibility of the primitive selection module to choose the type of primitive, based on the current state and prior observations of primitives being executed. In our implementation, the context or state in which the human has performed each primitive is extracted from the observed data, and is used by a nearest neighbor lookup process to find the past primitive executions whose context is most similar to the current context. For example the puck's position and velocity when it crosses a pre-specified line is often used as the index for a lookup. In the air hockey environment the primitives are selected and then run to completion, before the next primitive is selected and executed. In the marble maze environment a primitive can be interrupted if it is not making progress or causes the marble to fall into a hole.

The sub-goals for the primitive provide the parameters needed to perform the action. The sub-goals for the hit primitives, for example, are the desired hit location, the puck's desired post-hit velocity, and the target location. In air hockey these sub-goals are returned along with the single nearest neighbor as part of the selected primitive. The sub-goals in the marble maze game are obtained by interpolating between parameters of multiple previously executed primitives close of the selected type.

Table 1. Performance of player agent observing the number of games specified and then playing one game.

Games Observed	Time	Holes fallen into	Not making progress
1	367.3	6	5
2	257.9	6	3
3	234.6	3	3
4	189.8	5	2
5	129.9	4	1
6	72.2	3	0
7	123.4	3	1
8	73.2	4	0
9	231.0	3	3
10	243.0	3	3

5 Results

In the virtual game of air hockey an agent used data collected while observing a human to initially learn how to perform air hockey primitives and went on to increase it performance of shot primitives through practice. Agents in both the hardware and software versions have used the observed data to learn how to choose a primitive and parameters when operating in the environment. An agent in the virtual marble maze game learned how to perform primitives and an initial primitive selection strategy from observing a human. An agent also went on to increase its performance through practice. Initial research in the hardware marble maze has shown that better sensing and controlling devices are needed and a new version is currently being constructed for further research.

Table 1 shows the performance of a marble maze agent after it has observed a various number of games performed by a skilled human player. This agent only used the observed data and is not learning from practice. The observed player performed the task in about 55 seconds and never fell into a hole or was penalized for not making progress. There is an improvement in time to complete the maze up until six games are observed. But for holes fallen into, is not clear that observing more then three games has proven to be of benefit.

Since this agent is not using the learning from practice module, when it runs through the maze multiple times, it has approximately the same performance on each run. Any difference is due to noise that is purposely introduced into the simulation. A common error is for the agent to choose a primitive that has been performed on just the other side of the wall from where it is. This would create a sub-goal position that is out of reach of the marble from the current location. In this situation the marble would mostly just sit in a corner or fall in a hole that is nearby. Another frequently observed error is the agent choosing a primitive that it can not perform from the current state of the environment.

From observing the performance of this agent it can be seen that it must also have the ability to learn beyond the observation. As mentioned above, the agent computes an action using the observed data and then goes on to perform that action. To change its performance the agent must have knowledge of the overall task objective and some way to evaluate its performance toward the accomplishment of the task. It must also have a way to change its behavior as it practices to increase its performance toward completing the task.

6 Increasing Performance through Practice

There are a number of things the agent can learn to increase its performance while operating in the environment. At the primitive level it can improve the policy to become more proficient at primitive performance. The virtual air hockey player observed its own performance and collected data while practicing. This data was then used separately in a neural network and a kernel regression model to improve the hit performance of the agent. Many other methods can be used by an agent to learn a primitive through practice such as those used by Schaal and Atkeson [5] in pole balancing and of Kamon et. al. [12] in learning to grasp objects.

Agents can also learn to select more appropriate primitives and parameters. Without the learning from practice module the agent's only goal is to act like the teacher with no knowledge of a higher goal or task objectives. It is the job of the learning from practice module to provide this information so the agent's performance can be increased. The algorithms and performance of this module are currently being tested and will be presented in future research.

7 Conclusions

Agents must learn quickly if they are to operate in high dimension environments. Providing an agent with domain knowledge and the ability to learn from observation can greatly improve its learning rate. The presented framework provides much flexibility in conducting learning from observation research using primitives. The current research using this framework demonstrates its ability and future research will focus on improving the performance of the individual modules.

Acknowledgments

Support for both investigators was provided by ATR, Human Information Science Laboratories Department 3 and by National Science Foundation Award IIS-9711770. This research was supported in part by the Communications Research Laboratory (CRL).

References

1. Newton research labs homepage. http://www.newtonlabs.com.
2. C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator.* MIT Press, Cambridge, MA, 1988.

3. R. C. Arkin. *Behavior-Based Robotics.* MIT Press, Cambridge, MA, 1998.
4. C. G. Atkeson and S. Schaal. Learning tasks from a single demonstration. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA97)*, pages 1706–1712, 1997.
5. C. G. Atkeson and S. Schaal. Robot learning from demonstration. In J. D. H. Fisher, editor, *Proceedings of the 1997 International Conference on Machine Learning (ICML97)*, pages 12–20. Morgan Kaufmann, 1997.
6. P. Bakker and Y. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11, 1996.
7. D. C. Bentivegna and C. G. Atkeson. Using primitives in learning from observation. In *First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, 2000.
8. R. Dillmann, H. Friedrich, M. Kaiser, and A. Ude. Integration of symbolic and subsymbolic learning to support robot programming by human demonstration. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The Seventh International Symposium*, pages 296–307. Springer, NY, 1996.
9. G. Hayes and J. Demiris. A robot controller using learning by imitation. In *A. Borkowski and J. L. Crowley (Eds.), Proceedings of the 2nd International Symposium on Intelligent Robotic Systems*, pages 198–204, 1994.
10. G. Hirzinger. Learning and skill acquisition. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The Seventh International Symposium*, pages 277–278. Springer, NY, 1996.
11. K. Ikeuchi, J. Miura, T. Suehiro, and S. Conanto. Designing skills with visual feedback for APO. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The Seventh International Symposium*, pages 308–320. Springer, NY, 1996.
12. I. Kamon, T. Flash, and S. Edelman. Learning visually guided grasping: A test case in sensorimotor learning. In *IEEE Transactions on System, Man and Cybernetics*, volume 28(3), pages 266–276, 1998.
13. Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. In *IEEE Transactions on Robotics and Automation*, pages 799–822, 1994.
14. L.-J. Lin. Hierarchical learning of robot skills by reinforcement. In *Proceedings of the 1993 International Joint Conference on Neural Networks*, pages 181–186, 1993.
15. M. J. Mataric, M. Williamson, J. Demiris, and A. Mohan. Behavior-based primitives for articulated control. In *Fifth International Conference on Simulation of Adaptive Behavior (SAB-98)*, pages 165–170. MIT Press, 1998.
16. R. A. Schmidt. *Motor Learning and Control.* Human Kinetics Publishers, Champaign, IL, 1988.
17. M. Williamson. Postural primitives: Interactive behavior for a humanoid robot arm. In *Fourth International Conference on Simulation of Adaptive Behavior*, pages 124–131, Cape Cod, MA, 1996. MIT Press.