Learning From Observation Using Primitives

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

[1]Information Sciences Division, ATR International
[2]College of Computing, Georgia Institute of Technology
[3]Robotics Institute, Carnegie Mellon University

Abstract

This paper describes the use of task primitives in robot learning from observation. A framework has been developed that uses observed data to initially learn a task and then the agent goes on to increase its performance through repeated task performance (learning from practice). 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. The feasibility of this method is currently being tested with agents that learn to play a virtual and an actual air hockey game.

1 Introduction

Human learning is often accelerated by observing a task being performed or attempted by someone else. If robots can be programmed to use such observations to accelerate learning their usability and functionality will be increased and programming and learning time will be decreased. This paper describes research that explores the use of primitives in learning from observation. Our ultimate goal is to show that the use of primitives accelerates learning, and that primitives can be learned automatically by observing a teacher's performance. This paper describes how a set of predefined primitives can be used in learning from observation.

Figure 1 shows a virtual air hockey game that was created that allows a person to play against a virtual player. How the virtual air hockey playing agent learns its behavior from observing a human will be described. Research in this environment is also being performed using a humanoid robot (www.erato.atr.co.jp/DB/) and a camera based tracking system, figure 2. Learning using primitives on the hardware version has just begun. The architecture and current playing strategy of this environment will be described.

A camera based motion capture system can easily be used to collect data in a hardware implementation [9, 17]. Spong and others have programmed a robot arm to play air hockey [8, 18, 20].

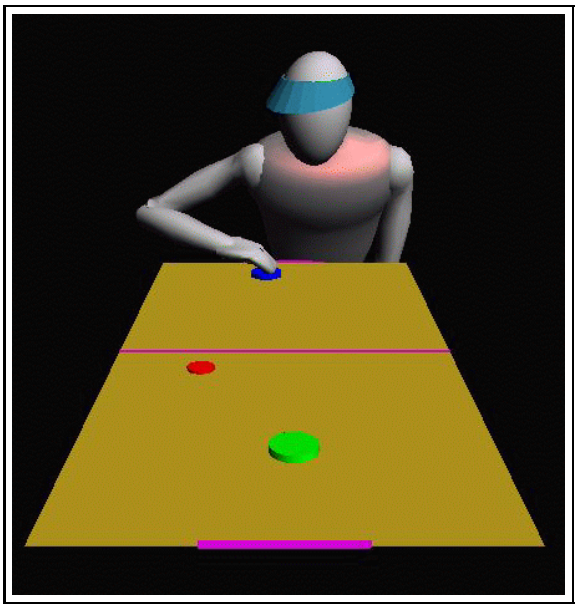

Figure 1: The virtual air hockey environment. The disc shaped object near the centerline is a puck that slides on the table and bounces off the sides, and the other two disc shaped objects are the paddles. The virtual player controls the far paddle, and a human player controls the closer paddle by moving the mouse. The object of the game is to score points by making the puck hit the opposite goal (the purple/light area at the ends of the board).

2 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 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,

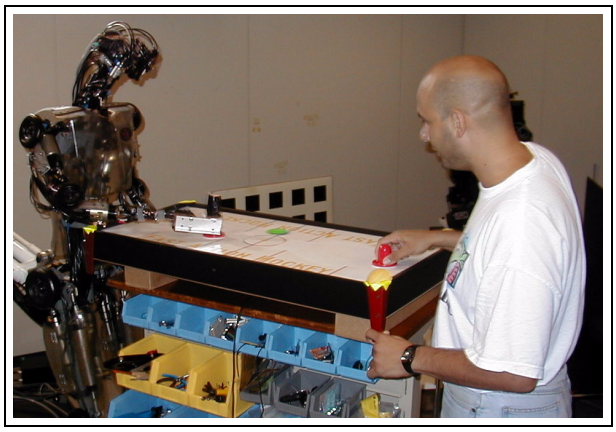

Figure 2: The hardware air hockey environment.

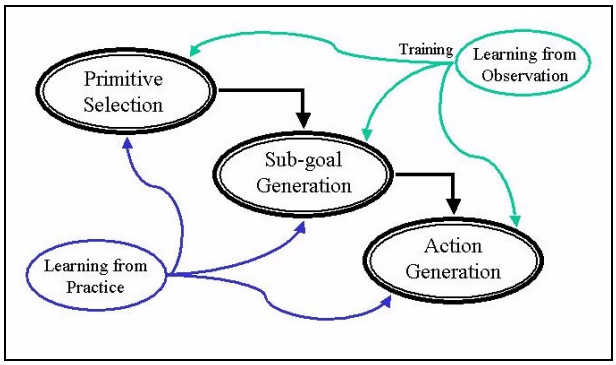

Figure 3: Our view of a primitive.

such as trajectory learning [1], learning from observation [4, 5, 11, 15, 6, 10, 12, 13], postural primitives [21], and other techniques that decompose complex tasks or movements into smaller parts [2, 7, 16].

It is our hope that primitives can be used to reduce the dimensionality of the learning problem [2, 19]. 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. There are many possible primitives, and it is often possible to break a primitive up into smaller primitives.

2.1 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 learning from observation 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.

3 Virtual Air Hockey

Air hockey is a game played by two people. They use round paddles to hit a flat round puck across a table. Air is forced up through many tiny holes in the table's surface, which creates a cushion of air for the puck to slide on with relatively little friction. The table has an edge around it that prevents the puck from going off the table, and the puck bounces off this edge with little loss of velocity. At each end of the table there is a goal area. The objective of the game is to hit the puck so that it goes into the opponent's goal area while also preventing it from going into your own goal area.

Figure 1 shows the virtual air hockey game created that can be played on a computer. The game consists of two paddles, a puck and a board to play on. A human player using a mouse controls one paddle. At the other end is a simulated or virtual player. The software for this game can be obtained at www.cc.gatech.edu/projects/Learning_Research/. The movement of the virtual player has very limited physics incorporated into it. The paddle movement is constrained to operate with a velocity limit. Paddle accelerations are not monitored and therefore can be unrealistically large. The virtual player uses only its arm and hand to position the paddle. For a given desired paddle location, the arm and hand are placed to put the paddle in the appropriate location, and any redundancies are resolved so as to make the virtual player look “human-like”. If the target is not within the limits of the board and the reach of the virtual player the location is adjusted to the nearest reachable point. The torso is currently fixed in space but could be programmed to move in a realistic manner. The virtual player's head moves so that it is always pointing in the direction of its hand, but is irrelevant to the task in this implementation.

The paddles and the puck are constrained to stay on

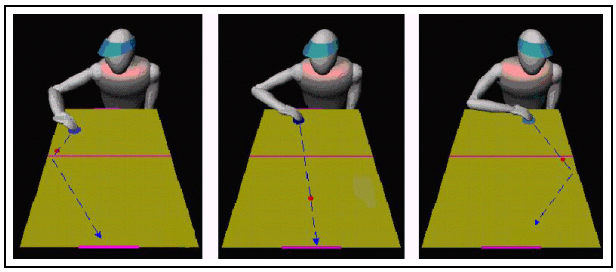

Figure 4: Three hit primitives being performed by the virtual player: right, straight, and left.

the board. There is a small amount of friction between the puck and the board's surface. There is also energy loss in collisions between the puck and the walls of the board and the paddles. 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.

4 Air Hockey Primitives

As explained above, human domain knowledge was used to define a set of primitives to work with initially. Three hit primitives are shown in figure 4. The full list of primitives used 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.

4.1 Perceiving the Primitives

The observed data must first be segmented into the above primitives. To accomplish this, critical events are used. Critical events are easily observable occurrences. The puck mostly travels in a straight line with a gradually decreasing velocity. Critical events for the puck include collisions, in which the ball speed and direction are rapidly changed.

The prepare primitive is performed whenever the puck is on the side opposite the player and will continue until a shot is to be attempted or a block is to be performed. In all the remaining primitives the puck is on the same side as the player, so discovering which primitive is being performed requires taking into account the position and velocity of the puck and collisions of the puck with the player's paddle.

The hit primitives are parameterized by the incoming puck position and velocity (when it crossed the center line), the hit location, the outgoing puck velocity and the target position. To find this primitive in the captured data, a collision of the puck with the player's paddle is searched for. When this event has been found, the puck's velocity vector is then observed to determine the target of the hit, and the state of the environment under which this primitive was selected. A simple idealized model is used to determine where the puck would hit the back wall if the opponent did not block it. This use of a model enables the learning agent to estimate the target being attempted without the shot having to be completed. The accuracy of the model can be critical in producing useful training data. Other methods can be used to reduce the reliance on the model, such as only considering shots that have actually hit the back wall without having hit any other walls or paddles.

If it is determined that the puck did not travel from the other side prior to a shot being made, this shot will be classified as a multi-shot primitive. The multi-shot primitive may be performed after a failed shot attempt or a blocking primitive and the puck is moving very slowly within hitting range. Here the player has plenty of time to setup and make a shot. If, after a collision with the observed player's paddle has been detected, the puck's observed trajectory does not fit the requirements for one of the hit primitives, the blocking primitive is considered. If the player's paddle is near their goal at the time of the collision, this will be classified as a blocking primitive.

4.2 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 po-

sition and velocity when it crossed the centerline is often used as the index for a lookup. In this implementation the primitives are selected and then run to completion, before the next primitive is selected and executed.

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. Currently these sub-goals are returned along with the single nearest neighbor as part of the selected primitive. A future implementation will obtain the sub-goals by interpolating between parameters of previously executed primitives of the selected type.

4.3 Finding the Right Paddle Motion

Once the primitive to perform has been decided upon, and the parameters are obtained, the agent must then figure out how to move the paddle to obtain the sub-goal. This can be done in many ways. Three methods have been tried; an idealized model based on physics, neural networks, and kernel regression [3].

The model based on idealized physics contains a simulation algorithm and computes the required paddle movements to hit the puck to a desired location with the desired output velocity. The computed movement is the minimum movement needed to obtain the correct hit. Paddle velocity that is perpendicular to the normal of the paddle-puck collision does not affect the puck's movement. This method ignores puck spin. Using this model produces extremely accurate results but does not take into account the information obtained from the observation. The accuracy of the model largely determines the results of this method. If a model based on domain knowledge is not available some other method must be used.

For the neural network and kernel regression methods information is extracted from the captured data so as to have the virtual player move the paddle the way that the human moved the paddle to make a shot. The observed data is segmented into the primitives and a database is created for each of the primitives. The hit primitives contain the following information:

Input:

- The XY location of the puck when it was hit.
- The velocity components of the puck when it was hit.
- The absolute velocity of the puck just after it is hit.
- The position on the back wall that the puck would hit if unobstructed.

Output:

- The paddle's velocity components at the time of the collision.
- The location of the paddle relative to the puck at the time of contact.

This information is used in the action generation module to tell the agent the paddle's velocity components and relative position that are needed to make the desired shot. The query to the generation module is the puck's velocity (from the game state) and desired hit location, the desired velocity of the puck after it is hit, and the desired location to shoot for on the back wall (from the sub-goal generation module). The module then outputs the information needed by the virtual player to make the shot.

The kernel regression method uses a weighed average of the closest n points to the query point to compute the needed information. To find the closest points in the state space the Euclidean distance of each data point from the query point is computed. This distance is computed as follows: $D(\boldsymbol{x}, \boldsymbol{q}) = \sqrt{\sum_j (\boldsymbol{x}_j - \boldsymbol{q}_j)^2}$ where $\boldsymbol{x}$ and $\boldsymbol{q}$ are the locations of the data point and the query point in state space. j represents the jth component of the vector. A kernel function uses the distance to compute the weight of that data point. The kernel function chosen for this module is $K(D) = \exp^{-D^2}$. The output components at the query point use n data points and are computed using the following equation: $y(\boldsymbol{q}) = \frac{\sum y_i K(D(\boldsymbol{x}_i, \boldsymbol{q}))}{\sum K(D(\boldsymbol{x}_i, \boldsymbol{q}))}$ where i ranges from 1 to n.

5 Hardware Air Hockey

The hardware implementation, figure 2, consists of the humanoid robot and a small air hockey table. The robot observes the position of the ball using its onboard 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.

5.1 Computing Joint Angles

This task uses 16 degrees of freedom. The following joints are used in the air hockey task: shoulder (2 joints), elbow, arm rotation, wrist rotation, hand (2 joints), waist (3 joints), head (2 joints, nod and rotation), and eyes (2 joints each eye, pan and tilt). Using all the joints above, except for the head and eyes, the robot must be positioned so that the puck is flat on the board and moves smoothly from one location to another. The other joints are used to position the head and eyes so that the entire board is in view at all times. We have manually positioned the robot in several positions on the board while maintaining these constraints, figure 5. To get joint angles for any desired puck position, we interpolate using the four surrounding training positions and use an algorithm similar to that used in graphics for texture mapping [14]. This approach allows us to solve the inverse kinematics of the robot with extreme redundancy in a simple way.

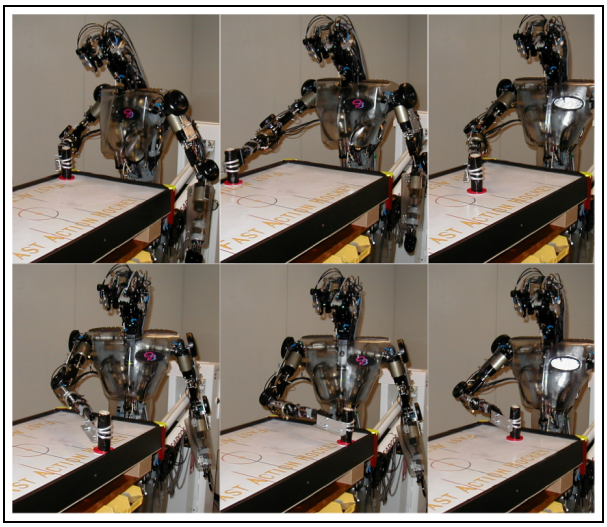

Figure 5: The six given configurations of the robot used to compute all enclosed configurations.

5.2 Vision

It is the job of the robot's vision system to provide the location of the puck and the two paddles on the playing field. Due to the movements of the torso, the head of the robot, and therefore the eyes, move. A simple object tracking method that takes into account the constrained playing field is used to compensate for this movement and provide accurate object tracking. The vision system consists of cameras located in the head of the robot and a color tracking system for each camera output. The system tracks the four corners of the board, the puck, and a paddle. Figure 6 shows the four corners and the puck displayed in the vision system. Using the four known corners and the fact they are in a plane, the location of an object within that polygon and on the plane can be computed. The state of the puck consists of its position and velocity. The puck velocity is numerically computed using filtered positions. The vision system runs at 60 frames per second and as long as the four corners are in view, board positions can be computed.

6 Differences Between the Software and Hardware Air Hockey

In the virtual game sensing is perfect and collision information is accurately known because it can be obtained from the simulator. In the hardware version only position data is obtained. From this data critical events such as collisions must be determined. We discover collisions by searching for rapid changes in the puck's velocity. The positions of the objects on the board are used to determine what item the puck collided with. In all other ways the primitives are searched for in a similar manner as in the software version.

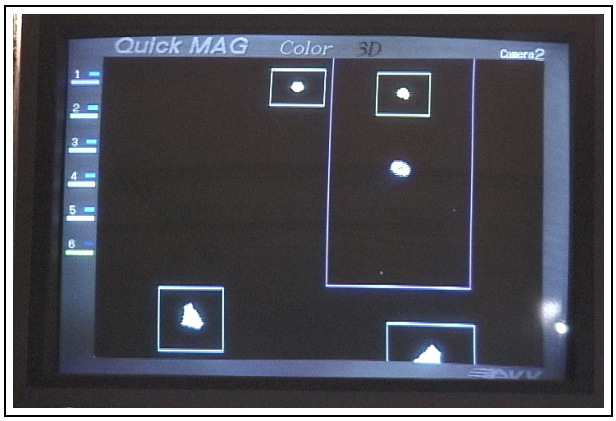

Figure 6: The four corners of the board and the puck in the middle as seen by the robot.

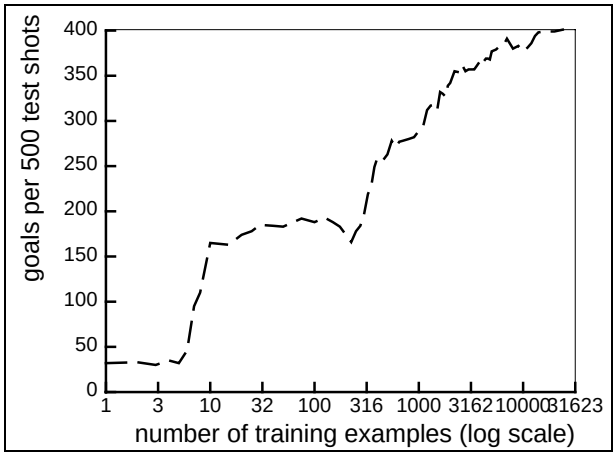

Figure 7: The number of goals obtained by the virtual agent as an increasing number of shots are observed. Note the log scale on the horizontal axis.

In the software version the agent may be placed without regard to physics. But in the hardware version the robot has many limitations such as maximum speed and command delay time. In the software version the shot is not considered until the puck crosses the centerline, giving it just a very small time to setup and make the shot. This strategy will not provide enough time for the hardware robot to make the shot. Therefore the hardware agent must start considering shots and setting up for them as soon as possible.

Predicting the trajectory of the puck is used in segmenting the data and in deciding on hit locations. The software version uses a simulator that is noise free and it is therefore very accurate in predicting the future state of the puck. In the hardware version on the other hand, the environment cannot be as accurately modeled. Additionally there will be slight variability from game to game due to factors such as the board being tilted or the friction changing due to fluctuations in the blower output.

7 Results

The learning from observation algorithms for virtual air hockey is fully implemented. The virtual air hockey player agent provides a fun and challenging opponent. Figure 7 shows a graph of the performance of the hit primitives as the agent observes an increasing number of hits. The agent first observes a number of shots taken by the human. The puck is then shot toward the agent 500 times just as if the opponent hit it. The number of times the agent scores for each observation set is shown in the graph. During these trials the opponent's goal is left undefended. As the number of observed hits increases, the agent's performance increases. We are currently devising other ways to evaluate the performance of the air hockey player agent in a quantitative way.

The same learning algorithms are currently being adapted for use by the humanoid robot. The humanoid air hockey environment has been set up and we have demonstrated the humanoid's ability to hit the puck and move around the board. Using a predefined playing strategy the humanoid has proven to be a fun opponent and the opportunity to increase its performance using this type of learning can be seen.

8 Conclusions

Virtual and hardware versions of an air hockey game have been created that allow data to be captured while the games are being played. Humans, using domain knowledge, select primitives to use and create software needed to parse the captured data. Modules are then created that use this data to select when and where primitives should be performed, the parameters needed for the performance of the primitives, and the low-level movements needed during the actual performance of the primitive. The agent then uses these modules to perform the task. A virtual air hockey player has learned a shot strategy, how to hit, and prepare from observing a human. A humanoid robot has been programmed to observe the state of the air hockey environment and play using a predefined strategy.

9 Acknowledgments

Support for both investigators was provided by the ATR-International Information Sciences Division and by National Science Foundation Award IIS-9711770.

References

[1] C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator*. MIT Press, Cambridge, MA, 1988.

[2] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[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] B. Bishop, P. Shirkey, and M. Spong. An experimental testbed for intelligent control. In *American Control Conference, Seattle*, 1995.

[9] B. E. Bishop and M. W. Spong. Vision-based control of an air hockey robot. *IEEE Control Systems Magazine*, 19(3):23–32, June 1999.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. E. N. J. F. Blinn. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, Oct. 1976.

[15] 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.

[16] 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.

[17] T. Ohshima, K. Satoh, H. Yamamoto, and H. Tamura. AR2 hockey: A case study of collaborative augmented reality. In *IEEE Virtual Reality Annual International Symposium*, pages 268–275, 1998.

[18] C. B. Partridge and M. W. Spong. Control of planar rigid body sliding with impacts and friction. In *International Journal of Robotics Research*, pages 336–348, 2000.

[19] R. A. Schmidt. *Motor Learning and Control*. Human Kinetics Publishers, Champaign, IL, 1988.

[20] M. W. Spong. Robotic air hockey. http://cyclops.csl.uiuc.edu, 1999.

[21] 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.