Applying Behavioural Cloning to Robotics

Claude Sammut
ARC Centre of Excellence for Autonomous Systems
School of Computer Science and Engineering
University of New South Wales
Sydney 2052 Australia
claude@cse.unsw.edu.au

***Abstract* – Programming an autonomous system can be a difficult and time-consuming task. One method for speeding up this process is to make the system teachable. That is, the system learns to perform a task by observing a human operator. We call this *behavioural cloning* when the learnt behaviour closely matches that of the trainer. Behavioural cloning has been applied to a variety of problems, originally in simulation. However, it is now being applied, with some success, to physical systems. In This paper, we review previous work, describe recent applications and discuss some of the outstanding problems in applying behavioural cloning to robots.**

I. Introduction

In the early years of Artificial Intelligence research, a major goal was to build a "general purpose problem solver", that is, a system that had a universal method for decision-making, requiring only a little domain knowledge. It was eventually realised that no such system could be built in practice and that domain specific problem solvers were the only way of build a system that was both practical and highly skilled. However, knowledge-based systems encountered the "knowledge acquisition bottleneck", the term given to the difficulty in acquiring adequate domain knowledge to make the system perform at an expert level. This problem lead to increased interest in Machine Learning approaches to building knowledge bases. If a human finds it difficult to discover or express domain knowledge, then give the system the ability to learn for itself.

There are strong analogies to this story in robotics. Ideally, we would like a task planning system that could work in any robotic domain. We would like a vision system that could cope with any environment that is presented to it. We would like a locomotion system that could traverse a wide variety of terrains. As with problem solving, experience has taught us that high performance comes from carefully engineering a solution to suit a specific task. And as with problem solving, this is a difficult and costly exercise. Once again, Machine Learning is seen as a way of reducing the cost of building autonomous systems and also as a way of improving their robustness.

Behavioural cloning has been gaining in popularity as an approach to acquiring control knowledge for a robot. The system learns to perform a task by observing a human operator. Logs of the human performance are input to a Machine Learning algorithm to produce a control strategy.

Various approaches to behavioural cloning have been successfully demonstrated, both in simulation and using physical systems. However, one of the biggest obstacles to more widespread deployment of behavioural cloning is the complexity of the sensory data collected by a robot. If the state of the system can be characterised by a small number of parameters, then the problem remains almost entirely within the realm of Machine Learning. If the state can only be characterised by complex data such as from a vision system then we must also develop methods for representing the data in a form that is suitable for learning.

A further problem that must be considered falls in the domain of Human-Computer Interaction. If the human operator has access to information that the robot does not then it may be difficult or even impossible for the robot to learn an operational control strategy. Therefore, the operator must be limited to a user interface that presents only the information available to the robot. This present problems since the way the information is presented can strongly influence the human's ability to perform well.

In the remainder of this paper, we give a brief introduction and history of behavioural cloning and then present recent work in applying it to physical systems. We describe attempts to address the problems of representation and human-computer interaction and conclude with a discussion open problems.

II. Behavioural Cloning

Michie [1] first formulated behavioural cloning as the process of capturing records of the human performance of a skill and using machine learning to transform the data into symbolic control rules that not only show run-time dependability but also transparency. That is, an important goal of behavioural cloning is to gain an understanding of the human skill as well as being able to reproduce it.

Initial experiments acquired knowledge for some quite complex systems but entirely in simulation: pole balancing, flying a fixed-wing aircraft [2], operating a container crane [3] and scheduling a production line [4]. These experiments used relatively straightforward method of representation control knowledge, namely, situation-action rules.

In the case of flying an aircraft, each time the pilot performs a control action, such as moving the stick or changing settings for the throttle and flaps, we record that action and the state of the aircraft at a time slightly preceding the action. That is, we record, the position and orientation of the aircraft, the rates of change of these variables and we

record all the other control settings. We then used a standard decision-tree learning algorithm [5] to induce a set of control rules for each control action.

Variations on the above methodology include decomposing the learning task into acquiring rules for different stages of the operation, for example, taking-off, turning, landing, etc. Within a single behaviour, multiple rules may also be learned for "goal-seeking" when the system is far from its target and "station-keeping" when the system is near its target. Kerr and Kibira [4] post-processed their decision trees by "fuzzifying" them. It was found that the scheduler was rapidly switching plans when the system was near a decision point and oscillating around it. The fuzzification smoothed this behaviour. While they introduced traditional membership functions, the same effect can also be achieved by using a machine learning algorithm that outputs class probabilities rather than strict classifications.

While situation-action rules are effective in relatively simple domains, they tend to be brittle in more complex environments [6]. For example, if the aircraft in the flight simulator drifts into regions of the state space that are not covered by the situation-action rules, there is no way of recovering. Recent work in behavioural cloning as focussed on creating representations of the skill that are more goal-oriented. These representations are more robust since the control does not attempt to map every possible state to an appropriate action. Instead, the controller tries to determine setting for particular goal parameters and then follows with a set of rules that apply actions to achieve those goals values.

Bain [7] constructed a system consisting of two learned rule-sets in which one was applied to the current state of the simulation to determine the desired values of predefined goal parameters. The system them "backward chained" with another rule set to find an action that would move the system closer to the goal values. Isaac [8] extended this architecture so that the second set of rules was implemented as a model tree that yielded PID values for a switching controller. The operation of this system is best illustrated by an example. The following rules show how a turn can be implemented. The goal rule is as follows:

t = goal turn rate

if $azimuth \le -3.6$ then
  if $distance \le 1715$ then
    $t = 0.163 \cdot azimuth + 0.0013 \cdot distance - 1.90$
  else
    $t = 0.073 \cdot azimuth + 0.0003 \cdot distance - 1.92$
else
  $t = 0.006 \cdot azimuth + 0.0022 \cdot distance - 1.93$

This rule determines the desired turn rate given the aircraft's heading and distance from the target. The rule is learnt by a regression tree program in which the leaf nodes are constructed by linear regression on the training data in the leaves. Once the desired turn rate has been found, the system uses the following rule to determine that setting for the ailerons.

INNER AILERON CONTROL RULE

if $i < 11.7$ then
  $a_{in} = -0.053 \cdot e - 0.0004 \cdot i - 0.003 \cdot d - 0.013$
else
  $a_{in} = -0.043 \cdot e - 0.0236 \cdot i - 0.012 \cdot d + 0.330$

OUTER AILERON CONTROL RULE

$a_{out} = -0.049 \cdot e - 0.0016 \cdot i - 0.012 \cdot d - 0.002$

$e = current\_turn\_rate - t$

$i =$ integral of e

$d =$ derivative of e

Two rules are given. The "inner" rule is used when the aircraft is close to the desired turn rate and the "outer" rule when further away. Each leaf node in the rules above is a PID controller. That is, each control action is determined by:

$$control = P \cdot error + I \cdot \int error \, \mathrm{d}t + D \cdot \frac{\mathrm{d}error}{\mathrm{d}t}$$

Thus, the system learns a switching controller. This approach had proved to be more robust than situation-action rules for complex tasks.

Suc [9] has taken the approach of learning "qualitative strategies" as constraints on learning a controller. From the human trainers actions, an algorithm for learning qualitative models is used to generate rules such as, "if the aircraft is approximately straight and level and if the aircraft's pitch is below the desired pitch, pull back on the stick". While this does not give the numerical values required to implement the action, the rule serves as a constraint for further learning. Given a set of rules that characterise the pilot's behaviour throughout a flight, functions to produce a numerical action value are randomly generated, constrained by the qualitative rules. The function that best achieves the desired result is retained.

There are other approaches to learning from demonstration, for example by Atkeson and Schaal [10]. However, they place less importance on building human-readable control rules.

III. Learning Control Strategies

D'Este, O'Sullivan and Hannah [11] gave a simple demonstration of behavioural cloning in robotics. The task was to teach a Pioneer II robot to follow a target and avoid obstacles. The robot was equipped with a colour camera and a sonar array. The target was painted bright green so that a simple vision system could isolate the target and determine its heading and distance. The sonar array was used to for obstacle detection. Thus, the inputs for learning are relatively simple: head and distance to target and the return values from the 16 ultrasonic sensors. Some experiments also included a "memory" in the sense that the values from the previous decision cycle were preserved and were also available for learning.

The human trainer controlled the robot via workstation and attached joystick. In many respects this is similar to the problem addressed by Pomerleau [12] when training the ALVINN vehicle to drive on roads. In both cases, it was found that the situation-action formulation only succeeded if the training data included a wide variety of conditions. For example, a "safe" driver who always stays in the middle of the road or a robot that stays well away from any obstacle provides no experience for what to do when the car does stray too close to the side or the robot wanders to near an obstacle. Pomerleau solved this problem by artificially creating data sets that contained additional examples. In the case of the Pioneer, the investigators combined data from a safe operator with those of a "dare devil" who allowed the robot to come close to barriers and fences. This experience of having to cover a wide variety of situations was one of the reasons for pursuing the goal-directed approaches described in the previous section.

The main feature of these examples is that while we are dealing with relatively complicated tasks, the input data are simple. Even in ALVINN, the vision system was highly restricted and the learner's task was mainly to relate curvature of the road with an appropriate steering angle. Even much more complex problems, such as flying a helicopter, have the same feature. They can be solved because the inputs are simple.

There have been several recent demonstrations of learning to control a helicopter [13, 14] has demonstrated. Ng's approach has some features in common with behavioural cloning. Behavioural traces of a human pilot are recorded but rather than using the data to construct a control strategy directly, a locally weighted linear regression algorithm is used to create a model of the helicopter. Once this was obtained it could be used in a simulation environment for applying reinforcement learning. Once learnt in simulation, the control policy is transferred to the real helicopter. Because reinforcement learning algorithms typically require many trials to learn a satisfactory policy, building a model is currently the only way of applying this kind of learning in practice. Ironically, one of the original claims for reinforcement learning was that no model is required. Ng's work has some similarities with Suc's [9] in that Suc also builds a model of the plant but this is used when refinancing the qualitative control rules to numerical ones.

As impressive as helicopter control is, the system can still be characterised by a small set of state variables, as in the flight simulator. Thus, solving this problem does not require dealing with complex representations of the world. In the following sections, we discuss two domains: robot soccer and autonomous ground vehicles in unstructured environments where representation becomes a serious issue.

IV. Teaching a Robot to Play Soccer

RoboCup[1] is an international competition held annually to encourage developments in robotics. It consists of several leagues of different kinds of robots. One of them is the Sony legged robot league. A team consists of four Sony Aibo robots. Each one has an onboard visions system and operates autonomously, although it can communication with its team mates via an 802.11b wireless network. We describe here some of our experiences in programming this robots to play soccer[2] and the implications for machine learning.

The robot's software system has evolved into quite a complex system as illustrated in Figure 1.

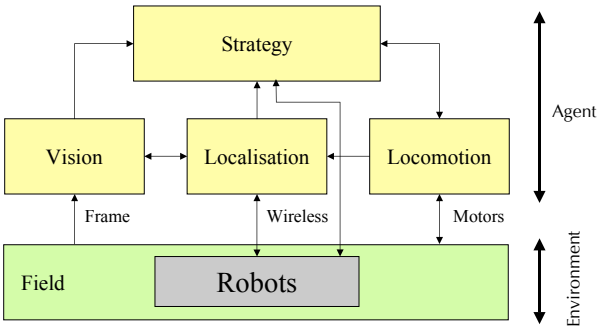

Figure 1. Robot Architecture

- The robot's vision system is primarily based on segmenting the image by colour and looking for edges that represent field markings. Because images can be very noisy and background objects may confuse object recognition, much effort goes into heuristics that filter out unlikely candidates.
- To localise, that is find where the robot is in relation to other objects in the field, the robot looks for the corner posts and goals, which are specially coloured and the field markings. Each robot constructs its own world model, which is a map of the field with objects placed where the robot thinks each one is.
- Communication between the robots is mainly used to assist in localisation. Each robot broadcasts its world model to its team mates. This is most helpful when one robot is unable to see the ball but another robot can. Data fusion is therefore critical component of the localisation module.
- Locomotion in a four-legged robot is quite complex and significant part of the effort in programming the robots is devoted to optimising leg movements for speed and to tailor them for specific skills such as kicking, dribbling, getting behind the ball, etc.
- The strategy module determines the appropriate action to perform for a given situation. For example, if the robot is in the mid-field and has a clear shot at the goal, it may choose a long-range kick. If it is near the goal, it may choose a less powerful but more accurate kick. If it cannot see the goal, it may continue to dribble the ball. The strategy module also determines the role of the robot: attacking, supporting the attacker or defending.

In the early years of the competition, some teams attempted to develop elegant software architectures based on generic designs. However, these were quickly abandoned as it was realised that to achieve a high level of performance, a great deal of domain specific knowledge was required. The generic systems were simply too slow and were not robust.

[1] http://www.robocup.org

[2] The University of New South Wales team has won the competition three times. Team report may be found at http://www.cse.unsw.edu.au/~robocup.

The robots would always do something unexpected because either the vision system was confused by an unusual scene or the interactions with other robots were so complex that it was impossible to predict the outcome of some behaviour.

The most successful methodology for developing the software for the robots has been to begin with simple-minded approaches, put the robots into a game, observe their behaviour and modify the code to eliminate weaknesses. Playing many games has been essential to discovering unique but important situations that the robots might encounter. Of course, this is slow and painstaking work. Ideally, we would like to put the robots into the field and let them learn as humans do, through coaching, practice and observation of other players.

A. Learning Low-Level Skills

Machine learning has been used very successfully in several aspects of the legged league. Colour detection is a difficult problem because the perception of colour is strongly affected by lighting conditions. Consequently, machine learning has been applied to learning colour calibration [15]. Sample images are obtained. They are manually classified to create a training set for learning. In our case, we use Quinlan's C4.5 [16].

Another area in which learning has been very beneficial is in improving the gait of the robot. Specify the kind of motion we want the robot to execute by describing a trajectory for the paws at the end of the leg. Many different shapes for the trajectory have been tried. Figure 2 shows an example.

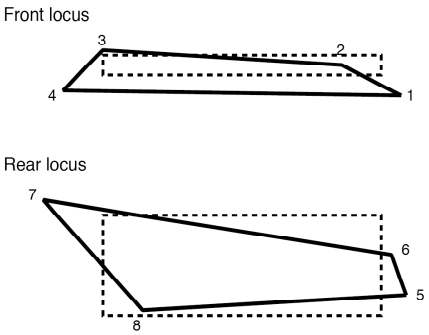

Figure 2. Locomotion Trajectory

Different surfaces may require a slightly different gait. For example, a smoother carpet could cause more slippage and so the angle of attack on the descent of the leg or the lift-off may have to be adjusted. Kim and Uther [17] used an optimisation technique to improve the gait. A robot is placed on the field and made to walk back and forth along a path of known length. The robot is able to time its traversal and therefore measure improvements in speed. Between each repetition, the parameters are adjusted following a gradient descent approach to find a better trajectory. In the 2003 competition this yielded approximately a 25% increase in straight line speed.

B. Learning Behaviours

We saw that supervised learning can be applied to learning low-level vision operations and trial-and-error learning can be used for locomotion. Learning behaviours would also be very useful because these are also difficult to devise. Some examples of simple behaviours are:

- If the robot cannot see the ball first look in the direction last indicated by the world model. If that fails, scan the head around. If that fails, rotate the whole robot 360º. Note that the robot should turn clockwise if the robot is defending its own goal and it is on the left hand side of the field. This is because the ball may be close to the robot but out of its field of view. When the robot turns, it may knock the ball. If that happens, we want the ball to be knocked away from our own goal. If the robot is attacking the other goal, it would turn in the opposite direction so that the all is knocked towards the goal.
- If two team mates are approaching the ball, the one that has the best angle on the ball should take precedence over the other robot. For example, if a robot is approaching the ball facing its own goal, it should get out of the way of a team mate coming from the other direction.
- If a robot is approaching the ball and there is no opponent nearby, it can take the time to trap the ball, turn and shoot in best direction. If there are opponents near by, the robot should not try to be accurate, it should use a simple kick to move the ball down the field and away from opponents.

The code is full of such heuristics. Various teams implement behaviours as state machines or decision trees but they all contain many *ad hoc* rules that are created as a result of watching games and devising new behaviours or patching old one when a failure occurs. To try to automate this process, we are developing a behavioural cloning system to capture behaviours from a human operator. However, before we can do any learning, we have to deal with problems in user interface design.

The setup is as follows. During practice sessions, a human can operate a robot by remote control. Information from the robot is displayed on a workstation screen and the motion of the robot is controlled by a joystick. We do not want the operator to observe the robots on the field directly. The reason is that the human would then have much more information than is available to the robot. The robot's field of view is very narrow whereas the operator looking at the field would have a global view. Therefore, the operator could make decisions based in information that the robot does not have. If that is the case, it may be impossible for the robot to learn when to trigger a behaviour. Therefore, we restrict the human to only knowing what the robot knows.

Figure 3. Robot view of a scene

Figure 2 shows a scene taken through the robot's camera; this gives a good indication of the narrow field of view. Figure 3 shows a similar scene after blob formation and object recognition. The bounding boxes indicate that an object is recognised. The pink-on-blue object is one of the corner posts.

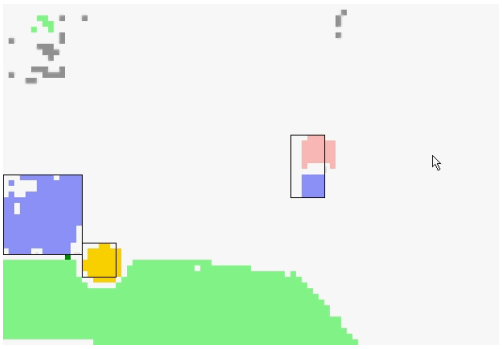

Figure 4. After object recognition

Even this display is too rich since the human may still do some edge detection or other operations that we cannot control. So the display is distilled even further to exactly those elements that would be seen by the robot (Figure 5).

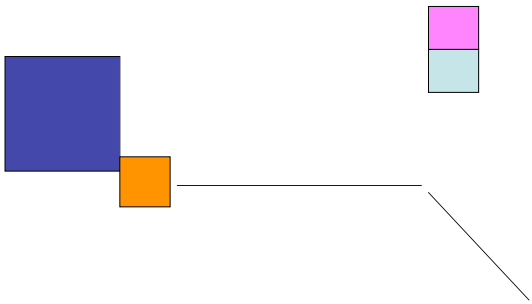

Figure 5. The object display

The joystick must allow control of the head to pan around. Motion control is simplified since the operator is not responsible for low-level locomotion. The robot moves and turns in the direction indicated by the joystick.

There are features of the tele-operation that are still under investigation. One is the degree to which we must slow down a game. The human reaction time is very slow compared with the robots. So when a team of human-controlled robots plays a fully autonomous team, the humans have little chance of winning. Thus, we must run the autonomous team in slow motion. However, this could affect the game. So we do not expect to be able to transfer cloned behaviours to a robot without modification.

The second feature under investigation is whether the human should see a display of the robot's world model. As described earlier, the outcome from the localisation module is a map of the field indicating the estimated positions of the different objects. An example of the world model display is shown in Figure 6. This shows the location of two blue (with arrows) and three red robots and the ball. The ellipses indicate the degree of uncertainty about the position of the object.

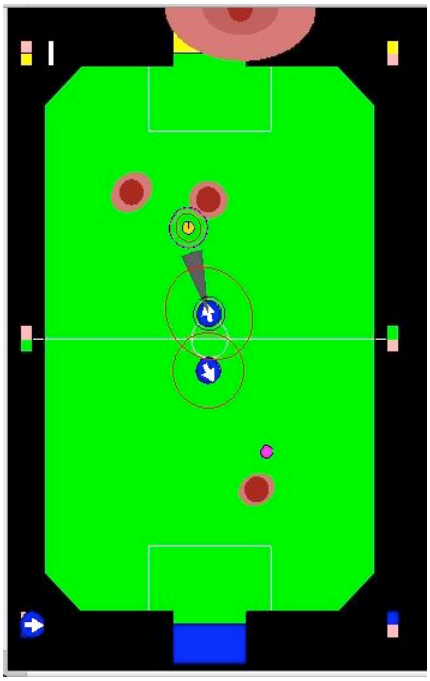

Figure 6. A world model for a robot facing the ball

Figure 6 shows the world model that has been constructed by the upward facing blue robot. It also contains information transmitted from the other blue robot. The second blue robot believes the ball is in front of it. This can occur if there is some background object that is being mistaken for the ball.

Because the operator only sees a very small window onto the field through the robot's camera, it is difficult to remember the location of objects that go out of view. So there is a reason for showing the operator the world model display. On the other hand, the world model can be misleading, as in the example above, so it may be better to let the operator maintain his own model. Both these modes of operation will be tested.

There are many other user interface issues yet to be settled. For example, the robot's head has three degrees of freedom so the image may be rotated. The robot's vision system uses the head joint angles to transform object positions. Should the human be shown the corrected display or the unrotated display? This is again a matter to be determined empirically.

As in previous behavioural cloning experiments, we capture the operator's actions, along with the state of the system at the time the action was taken. Unlike other experiments, the information in the images is *relational*. We are often interested in whether one blob appears above, below, left or right of another object. Whether it appears bigger or smaller, closer of further away. So there is further work required in determining an appropriate representation for the training data.

V. Open Problems and Conclusion

In the previous section, we saw how a rich sensory system such as vision can complicate attempts at behavioural cloning. In RoboCup, the vision problem is relatively easy because the environment has been carefully structured. Now

let us consider robots operating in unstructured environments. Some examples are autonomous ground vehicles operating off-road or RoboCup rescue in which a robot is required to navigate through a mock collapsed building to locate survivors.

If we wish to have a robot drive through a forest, we have no reliable way of recognising landmarks in such a complicated scene. An alternative to segmenting an image to find local features is to treat the image as a whole and operate on global features. This is the approach widely used in content-based image retrieval. There has been growing interest in dimensionality reduction techniques [18] to store and retrieve images based on global features. An image is mapped into a high-dimensional space in such a way that similar images should be mapped into neighbouring regions. Clustering then tells us how to group images. Now consider how we navigate when driving, say, from home to the office. How do we recognise that we should turn at a particular street? We may recognise a building or some other landmark. However, the same affect could be achieved if we recognise the scene, as a whole, as being similar to other scenes where a turn has been successful.

These methods suggest that it may be possible to train an autonomous vehicle to navigate through a complex environment. A possible approach is to allow a human driver to control the vehicle, collecting images as it goes. The images are stored in a database along with associated actions. To retrace the path autonomously, as the vehicle captures images, they are matched with images in the database. If there is a match, the associated action is performed. What is less clear at present is how such a system could be generalised so that the vehicle will make a sensible decision when it encounters a new set of scenes.

VI. Conclusion

Behavioural cloning has been used to considerable effect in robotic applications. However, the tasks that have been tackled so far have not required complex representations. As we move to less structured environments, it is necessary to find ways of representing the robot's sensory inputs so that a correspondence can be made between the human's perception of the world and the robot's. Only then can the robot learn to associate the appropriate actions with its inputs.

References

[1] D. Michie, M. Bain, and J. E. Hayes-Michie, "Cognitive models from subcognitive skills," in *Knowledge-base Systems in Industrial Control*, M. Grimble, S. McGhee, and P. Mowforth, Eds.: Peter Peregrinus, 1990.

[2] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to Fly," presented at Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, 1992.

[3] T. Urbancic and I. Bratko, "Reconstructing Human Skill with Machine Learning," presented at Proceedings of the 11th European Conference on Artificial Intelligence, 1994.

[4] R. M. Kerr and D. Kibira, "Intelligent reactive scheduling by human learning and machine induction," presented at IFAC Conference on Intelligent Manufacturing Systems, Vienna, 1994.

[5] J. R. Quinlan, "Learning Efficient Classification Procedures and Their Application to Chess End Games," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Palo Alto: Tioga, 1983.

[6] I. Bratko, T. Urbancic, and C. Sammut, "Behavioural Cloning of Control Skill," in *Machine Learning and Data Mining*, R. S. Michalski, I. Bratko, and M. Kubat, Eds., 1998, pp. 335-351.

[7] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, K. Furukawa, D. Michie, and S. Muggleton, Eds.: Oxford University Press, 1999, pp. 103-129.

[8] A. Isaac and C. Sammut, "Goal-directed Learning to Fly," presented at Proceedings of the Twentieth International Conference on Machine Learning, Washington, D.C., 2003.

[9] D. Suc, I. Bratko, and C. Sammut, "Learning to Fly Simple and Robust," presented at European Conference on Machine Learning, Pisa, 2004.

[10] C. G. Atkeson and S. Schaal, "Robot Learning From Demonstration," in *Proceedings of the Fourteenth International Conference on Machine Learning*, D. H. Fisher, Ed. Nashville, Tennessee: Morgan Kaufmann, 1997, pp. 12 - 20.

[11] C. D'Este, M. O'Sullivan, and N. Hannah, "Behavioural Cloning and Robot Control," in *Robotics and Applications*, M. H. Hamza, Ed. Salzburg: IASTED/ACTA Press, 2003, pp. 179-182.

[12] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA, 1989.

[13] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," presented at International Conference on Robotics and Automation, South Korea, 2001.

[14] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry, "Autonomous Helicopter Flight via Reinforcement Learning," presented at Advances in Neural Information Processing Systems 16, Cambridge, MA, 2003.

[15] S. Chen, L. Siu, T. Vogelgesang, T. F. Yik, B. Hengst, S. B. Pham, and C. Sammut, "The UNSW RoboCup 2001 Sony Legged Robot Leage Team," in *RoboCup 2001: Robot Soccer World Cup V*, *Lecture Notes in Artificial Intelligence*, A. Birk, S. Coradeschi, and S. Tadokoro, Eds.: Springer, 2002, pp. 39-48.

[16] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[17] M. S. Kim and W. Uther, "Automatic gait optimisation for quadruped robots," presented at Australasian Conference on Robotics and Automation, Brisbane, Australia, 2003.

[18] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323-2326, 2000.