Learning to Drive a Bicycle using Reinforcement Learning and Shaping

Jette Randløv
CATS, Niels Bohr Institute,
University of Copenhagen,
Blegdamsvej 17,
DK-2100 Copenhagen Ø, Denmark
randlov@nbi.dk

Preben Alstrøm,
alstrom@cats.nbi.dk

Abstract

We present and solve a real-world problem of learning to drive a bicycle. We solve the problem by online reinforcement learning using the Sarsa(λ)-algorithm. Then we solve the composite problem of learning to balance a bicycle and then drive to a goal. In our approach the reinforcement function is independent of the task the agent tries to learn to solve.

1 Introduction

Here we consider the problem of learning to balance on a bicycle. Having done this we want to drive the bicycle to a goal. The second problem is not as straightforward as it may seem. The learning agent has to solve two problems at the same time: Balancing on the bicycle and driving to a specific place. Recently, ideas from behavioural psychology have been adapted by reinforcement learning to solve this type of problem. We will return to this in section 3.

In reinforcement learning an agent interacts with an environment or a system. At each time step the agent receives information on the state of the system and chooses an action to perform. Once in a while, the agent receives a reinforcement signal r. Receiving a signal could be a rare event or it could happen at every time step. No evaluative feedback from the system other than the failure signal is available. The goal of the agent is to learn a mapping from states to actions that maximizes the agent's discounted reward over time [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998]. The discounted reward is the sum $\sum_{i=0}^{\infty} \gamma^i r_{t+i}$, where γ is the discount parameter.

A lot of techniques have been developed to find near optimal mappings on a trial-and-error basis. In this paper we use the Sarsa(λ)-algorithm, developed by Rummery and Niranjan [Rummery and Niranjan, 1994, Rummery, 1995, Singh and Sutton, 1996, Sutton and Barto, 1998], because empirical studies seem to suggest that this algorithm is the best so far [Rummery and Niranjan, 1994, Rummery, 1995, Sutton and Barto, 1998]. Figure 1 shows the Sarsa(λ)-algorithm. We have modified the algorithm slightly by cutting of eligibility traces that fall below 10^{-7} in order to save calculation time. For replacing traces we allowed the trace for each state-action pair to continue until that pair occurred again, contrary to Singh and Sutton [Singh and Sutton, 1996].

1. Initialize all eligibility traces $e_0 = 0$.
2. Set $t = 0$.
3. Choose action a_t.
4. If $t > 0$ then learn
   $w_t = w_{t-1} + \alpha [r_{t-1} + \gamma Q_t - Q_{t-1}] e_{t-1}$.
5. Calculate $\nabla_w Q_t$ with respect to the chosen action.
6. Update accumulating traces as
   $e_t = \gamma \lambda e_{t-1} + \nabla_w Q_t$.
   Update replacing traces as
   $e_t(s) = \begin{cases} \nabla_w Q_t & \text{if } \nabla_w Q_t \neq 0, \\ \gamma \lambda e_{t-1}(s) & \text{otherwise.} \end{cases}$
7. Perform action, receive reinforcement-signal.
8. If the system has entered a terminal state, then
   $t \leftarrow t + 1$ and jump to point 3.
9. Otherwise perform the learning (point 4) with
   $Q_t = 0$.

Figure 1: The Sarsa(λ)-algorithm.

2 Learning to balance on a bicycle

Our first task is to learn to balance. At each time step the agent receives information about the state of the bicycle,

the angle and angular velocity of the handle bars, the angle, angular velocity and acceleration of the angle from the bicycle to vertical. For details of the bicycle system we refer to appendix A.

The agent chooses two basic actions. What torque should be applied to the handle bars, $T \in \{-2\,\mathrm{N}, 0\,\mathrm{N}, +2\,\mathrm{N}\}$, and how much the centre of mass should be displaced from the bicycle's plan, $d \in \{-2\,\mathrm{cm}, 0\,\mathrm{cm}, +2\,\mathrm{cm}\}$ —a total of 9 possible actions. Noise is laid on the choice of displacement, to simulate an imperfect balance, $d = d_{\text{agents choice}} + sp$, where p is a random number within $[-1; 1]$ and s is the noise level measured in centimeters. We use $s = 2\,\mathrm{cm}$.

Our agent consists of 3456 input neurons and 9 output neurons, with full connectivity and no hidden layers. The learning rate is $\alpha = 0.5$. The continuous state data is discretised by non-overlapping intervals in the state-space, such that there is exactly one active neuron in the input layer. This neuron represent state information for all the different state variables. The discrete intervals (boxes) are based on the following quantization thresholds:

The angle the handle bars are displaced from normal, θ: 0, $\pm 0.2, \pm 1, \pm\frac{\pi}{2}$ radians.

The angular velocity of the angle, $\dot{\theta}$: 0, ± 2, $\pm\infty$ radians/second.

The angle from vertical to bicycle, ω: 0, $\pm$ 0.06, $\pm$ 0.15, $\pm\frac{1}{15}\pi$ radians.

The angular velocity, $\dot{\omega}$: 0, $\pm$ 0.25, $\pm$ 0.5, $\pm\infty$ radians/second.

The angular acceleration, $\ddot{\omega}$: 0, ± 2, $\pm\infty$ radians/second^2.

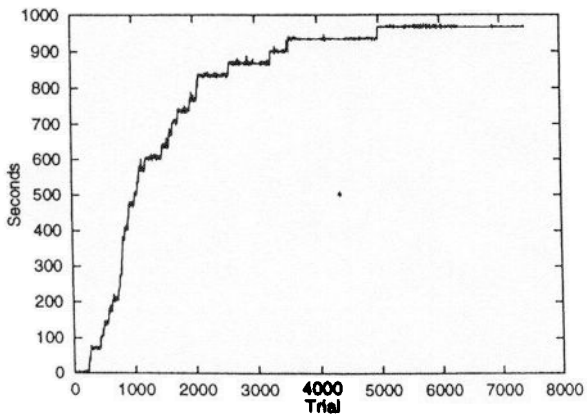

Figure 2: Number of seconds the agent can balance on the bicycle, as a function of the number of trials. Average of 40 agents. (After the agent has learned the task, 1000 seconds are used in calculation of the average.)

Figure 2 shows the number of seconds the agent can balance on the bicycle as a function of the number of trials. When the agent can balance for 1000 seconds, the task is considered learned. Here $\lambda = 0.95$ and $\gamma = 0.99$. Several CMAC-systems (also know as generalized grid coding) [Watkins, 1989, Santamaría et al., 1996, Sutton, 1996, Sutton and Barto, 1998], were also tried, but none of them gave the agent a learning time below 5000 trials.

Figures 3 and 4 show the movements of the bicycle at the beginning of a learning process seen from above. Each time the bicycle falls over it is restarted at the starting point. At each time step a line is drawn between the points where the tyres touch the ground.

Both accumulating and replacing eligibility traces were tried. The results are shown in figure 5. The results found support the general conclusions drawn by Singh and Sutton [Singh and Sutton, 1996]: Replacing traces make the agent perform much better than conventional, accumulating traces. Long traces help the agent best.

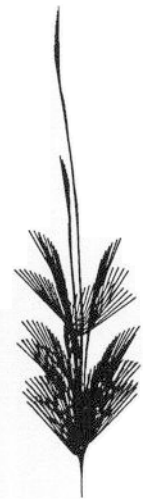

Figure 3: The first 151 trials seen from above. The longest path is 7 meters.

3 Shaping

The idea of shaping, which is borrowed from behavioural psychology, is to give the learning agent a series of relatively easy problems building up to the harder problem of ultimate interest [Sutton and Barto, 1998]. The term originates from the psychologist Skinner [Skinner, 1938], who studied the effect on animals, especially pigeons and rats.

To train an animal to produce a certain behavior, the trainer must find out what subtasks constitute an approximation of the desired behavior, and how these should be reinforced [Staddon, 1983]. By rewarding successive approximations to the desired behavior, pigeons can be brought to pecking a selected spot [Skinner, 1953, p. 93], horses to do clever tricks in a circus like seemingly recognize flags of nations or numbers and to do calculation [Jørgensen, 1962, pp. 137-139], and pigs to perform complex acts as eating breakfast at a table and vacuuming the floor [Atkinson et al., 1996, p. 242]. Staddon notes that human education as well is built up as a process of shaping if behavior is taken to include "understanding" [Staddon, 1983, p. 458].

Figure 4: The same route as figure 3 a little later. Now the agent can balance the bicycle for 30–40 meters. The agent starts each trial in a equilibrium position $(\theta, \dot{\theta}, \omega, \dot{\omega}, \ddot{\omega}) = (0, 0, 0, 0, 0)$. During the first trials it learns to avoid disturbing this unnecessarily, i.e. it learns to keep driving straight forward. Now the most difficult part of the learning remains: To learn to come safe though a dangerous situation. A weak (random) preference for turning right (instead of left) is strengthened during the learning as the agent gets better at handling problematic situations and therefor receives less discounted punishment than expected.

Shaping can be used to speed up the learning process for a problem or in general to help the reinforcement learning technique scale to large and more complex problems. But there is a price to be paid for faster learning: We must give up the *tabula rasa* attitude that is one of the attractive aspects of basic reinforcement learning. To use shaping in practice one must know more about the problem than just under which conditions an absolute good or bad state has been reached. This introduces the risk that the agent learns a solution to a problem that is only locally optimal.

There are at least three ways to implement shaping in reinforcement learning: By lumping basic actions together as macro-actions, by designing a reinforcement function that rewards the agent for making approximations to the desired behavior, and by structurally developing a multi-level architecture that is trained part by part.

Selfridge, Sutton and Barto showed that transferring knowledge from solving an easy version of a problem such as the classical pole mounted on a cart can ease learning a more difficult version [Selfridge et al., 1985].

McGovern, Sutton and Fagg have tested macro-actions in a gridworld and found that in some cases they accelerate the learning process [McGovern et al., 1997].

Dorigo, Colombetti and Borghi have worked with shaping for real robots [Dorigo and Colombetti, 1993, Colombetti et al., 1996, Dorigo and Colombetti, 1997]. They use reinforcement learning as a mean to translate suggestions from an external trainer. The trainer is a programme in itself with a high-level representation of the desired behavior that provided immediate reinforcement. For instance in the "The Hamster Experiment" [Colombetti et al., 1996] the robot's task is to collect pieces of food (colored cans) and bring them to its nest. The trainer provides the agent with a reinforcement signal for approaching the food. This signal is proportional to the decrease in the distance between the robot and the pieces of food. The training of the agent boils down to translating the high-level trainer to a low-level control programme. This method of shaping by a trainer has a number of advantages as well as disadvantages. The agent does not have to solve the delayed reinforcement problem. But on the other hand, the programmer of the trainer must know in advance what high-level behavior is desired, and to such a degree that the trainer can judge how well a single move fits into the desired behavior.

Mataric has studied the possibility of putting implicit domain knowledge into the agent by construction a more complex reinforcement function than commonly used [Mataric, 1994]. Again the theory was tested on a real robot moving cans to a nest. Here the constructed function did

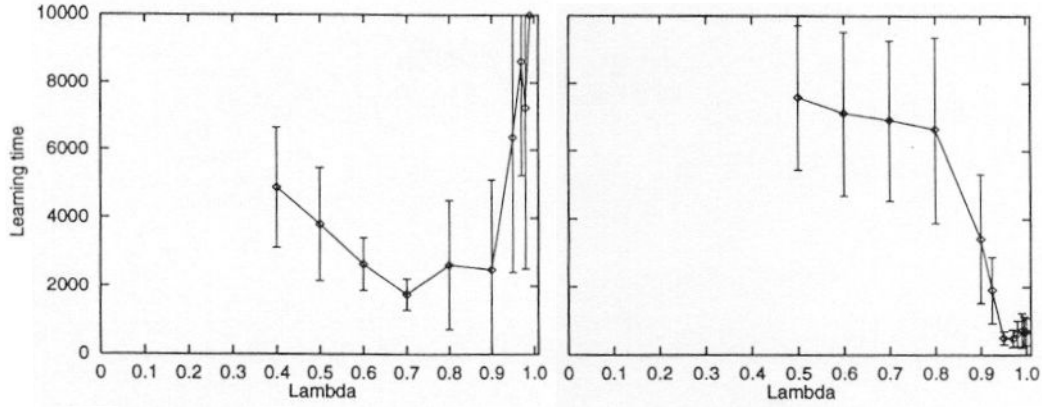

Figure 5: Learning time for different values of λ for accumulating eligibility traces (left) and replacing traces (right). Each point is an average of 30 simulations.

not eliminate the need for solving the delayed reinforcement problem.

Gullapalli has studied two implementations of shaping [Gullapalli, 1992]. In the first the complexity of the control task is gradually increased during learning, and the reinforcement function used is changed accordingly. In this way most of a training run is used in learning the approximation to the current target behavior. This system was used to make a simulated robot hand perform a series of key strokes on a calculator. The actual task consisted of six subtasks. Secondly Gullapalli considered structural shaping: An incremental development of the learning system where a multi-level architecture is trained in parts.

Gerald Tesauro's Backgammon playing agent achieved master level play through self-play [Tesauro, 1992, Tesauro, 1994, Tesauro, 1995]. This can be considered as a very succesfull example of the use of shaping. Self-play is a sort of shaping, since at first the agent plays against a nearly random opponent and thereby solves an easy task. The complexity of the task then grows as the agent gets better at playing.

In Gullapalli's experiments [Gullapalli, 1992] and Selfridge, Sutton and Barto's [Selfridge et al., 1985], as well as in Dorigo, Colombetti and Borghi's [Colombetti et al., 1996, Dorigo and Colombetti, 1997], the agent received a different reinforcement signal over time for the same behavior. This is not in agreement with the original inspiration of the reinforcement signal as being a hardwired signal inside the brain of a animal. To solve this problem, we need the reinforcement function to be independent of what task the agent tries to learn to solve. Our approach in general is to let the most basic tasks result in the lowest reinforcement signals and more advanced tasks correspond to larger signals.

Say, we want a robot to learn to move forward like a child (see figure 6). As a child grows stronger it discovers more complex and faster ways of moving. Performing each way of moving can be seen as a task that is more difficult than the former. The robot starts by learning to roll. Having done so, it might discover how to crawl. The reinforcement signal for crawling is greater than rolling, and greater than what the agent expects to receive, and therefore it acts as a reward. Later after having learned to walk, failing to walk and falling back on crawling makes the robot receive a smaller reinforcement signal than it expected, and the internal reinforcement signal becomes negative—that is the signal acts as a punishment.

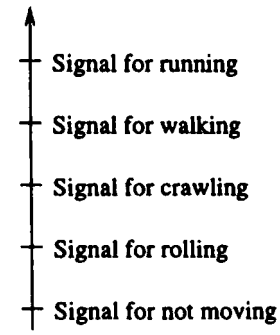

Figure 6: Reinforcement signals for the movements of a child-robot.

Can these basic ideas of shaping be applied to reinforcement learning, and make it possible to solve a complex problem with more than one goal? We will now turn to a practical study of these theoretical issues.

4 Learning to drive to a goal using shaping

We want to study shaping on the composite problem of learning to balance a bicycle and then drive to a goal. In contrast to other experiments with shaping, we want the agent to be totally in charge of when to switch task. When

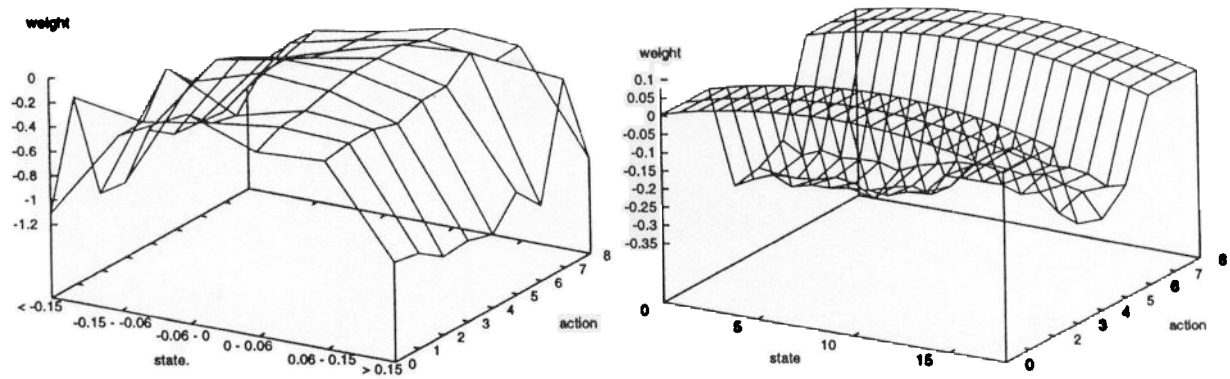

Figure 7: The weight, from the ω-oriented input neurons (left) and the weights from the angle oriented input neurons (right). Note the difference of the scale.

one drives a bicycle in the morning to the institute and hits a hole in the road, one instantaneously forgets about where to go and focus attention on the balance. We want the agent to be able to switch task equally swiftly when it find the situation appropriate.

The bicycle starts out at the origin heading west. The goal is a circular spot (10 meter radius) positioned 1000 meters to the north of the starting point.

We enlarge our basic network by 20 more input neurons, with full connectivity to the 9 output neurons. The angle between the driving direction and the direction to goal is discretised by 18° intervals, one for each neuron. Now there are exactly two active neurons in the agents input layer—one for the state of the bicycle and one for the driving direction relative to the goal. The learning rate for the weights from the angle-input neurons is chosen to be 0.01—much smaller than the rate for the other weights, in order to reflect the different time scales in the learning tasks: We do not want the weights in the angle oriented part to grow large while the agent learns to balance the bicycle. The odds are against these weights ending up containing anything useful.

The reinforcement function is independent of the task the agent tries to learn to solve. If the bicycle falls over, the agent always receives -1, if the agent reaches the goal it is rewarded by $r = 0.01$, and otherwise the agent receives $r = (4 - \psi_g^2) \cdot 0.00004$, where ψ_g is the angle between the driving direction and the direction to goal measured in radians. The agent is punished when driving away from the goal and rewarded when driving towards it. This reinforcement function is inspired by the signal used by Colombetti, Dorigo and Borghi [Colombetti et al., 1996] mentioned earlier. Note that the agent still have to solve the delayed reinforcement problem. As one can see, the numerical value of this signal is quite small. We tried larger values, which made the agent learn to drive in the correct orientation without being able to balance. After a few hundred trials the agent at the starting point immediately threw the bicycle to the right. The positive reinforcement it received due to the correct orientation in several time steps was large enough to make up for the punishment from falling.

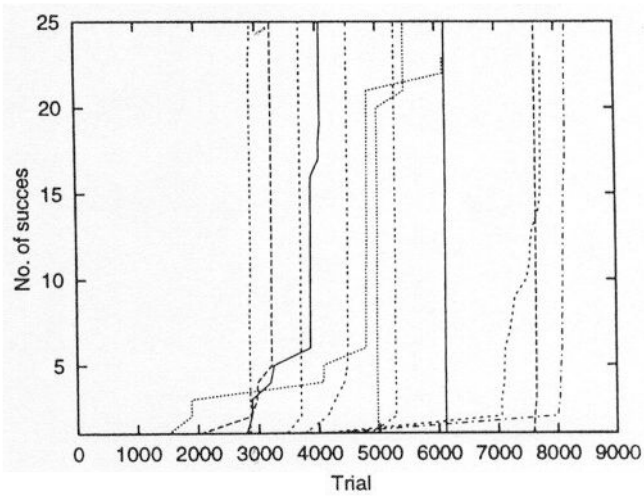

Figure 8: Number of times an agent drives the bicycle to the goal for twelve agents.

Figure 8 shows the number of times twelve agents reach the goal. In a typical learning process it takes the agent 1700 trials to learn to balance (i.e. drive more than 1000 s without falling), and after about 4200 trials it gets to the goal for the first time. After a total of approximately 5700 trials it drives to the goal more or less every time.

Figure 7 shows the values of some of the important weights

after learning. The ω-weights shown are an average of weight values around $\theta = 0$, $\dot{\theta} = 0$, $\dot{\omega} = 0$ and $\ddot{\omega} = 0$. If the agent drives along in balance, the weights with values in the relatively flat upper area are active for the balance oriented input neurons, and the values of the angle oriented neurons matter for the choice of action. The weights belonging to the balance oriented input neurons makes the agent prefer action 3, 4 and 5 (which corresponds to $T = 0$), but the weights belonging to the angle oriented neurons decide which one. But if the state of the bicycle enters an area of unbalance, the balance oriented input neurons have far greater differences in values of the weights, and as a result the angle oriented input neurons do not make any difference for the choice of action. In other words: The agent swiftly shifts attention from the task of finding the goal to the task of balancing the bicycle if required.

Figure 9: A typical route when the agent reaches the goal for the first time.

Figure 9 and 10 shows routes from the starting point to the goal (the grey circle on the y-axis). The first drives to the goal can be as long as 200 km, but the agent soon learns to drive to the goal driving "only" 7 km. A driving distance as short as 1680 m has been observed.

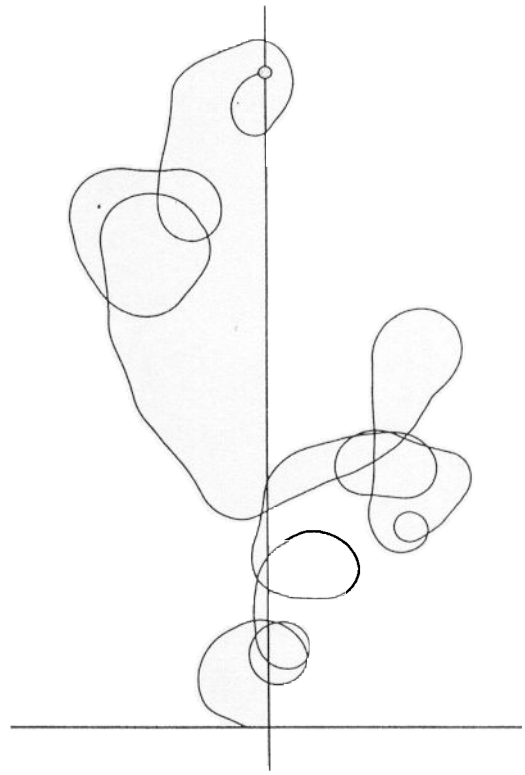

Figure 10: Already after 10 drives to the goal the agent navigates a little better.

The goal is not reached just by coincidence. The probability for hitting the goal at random is quite small. An estimate for the time required to reached the goal by doing a correlated random walk is 10^{10} time steps. (The bee line from the starting point to the goal is $3.6 \cdot 10^4$ time steps.) In other words: If the agent had to solve the problem of learning to drive to the goal without access to the shaping reinforcement signal, i.e. the tabular rasa approach, it would take enormous amounts of time before it hits the goal for the first time and experiences the reward for getting there.

We agree with Mataric [Mataric, 1994] that these heterogeneous reinforcement functions have to be designed with great care. In our first experiments we rewarded the agent for driving towards the goal but did not punish it for driving away from it. Consequently the agent drove in circles with a radius of 20–50 meters around the starting point. Such behavior was actually rewarded by the reinforcement function, furthermore circles with a certain radius are physically very stable when driving a bicycle because of the cross terms in eqs. (2) and (3) in the appendix.

5 Conclusion

Our results demonstrate the utility of reinforcement learning on a difficult, dynamical real world problem. It is possible to learn to balance a bicycle by pure reinforcement learning with only one (rare) reinforcement signal. Furthermore it is possible to learn a solution to the double problem of balancing on the bicycle and driving to a goal by combining reinforcement learning with shaping. The application of shaping accelerated the learning process immensely. Without shaping, it would not have been practical to wait for the agent to discover the goal and the reward for getting there.

Acknowledgements

We would very much like to thank Andrew G. Barto for several good discussions and stimulating ideas.

References

[Atkinson et al., 1996] Atkinson, R. L., Atkinson, R. C., Smith, E. E., Bem, D. J., and Nolen-Hoeksema, S. (1996). *Hilgard's Introduction to Psychology*. Harcourt Brace College Publishers, 12'th edition.

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

[Colombetti et al., 1996] Colombetti, M., Dorigo, M., and Borghi, G. (1996). Robot shaping: The hamster experiment. Technical Report TR/IRIDIA/1996-6, Université Libre de Bruxelles.

[Dorigo and Colombetti, 1993] Dorigo, M. and Colombetti, M. (1993). Robot shaping: Developing autonomous agents though learning. Technical Report TR-92-040, International Computer Science Institute, Berkeley. Labeled: To appear in Artificial Intelligence Journal.

[Dorigo and Colombetti, 1997] Dorigo, M. and Colombetti, M. (1997). Précis of "Robot Shaping: An Experiment in Behavior Engineering". *Adaptive Behavior*, 5(3–4). Précis of the book from MIT Press, Oct. 1997.

[Gullapalli, 1992] Gullapalli, V. (1992). *Reinforcement Learning and Its Application to Control*. PhD thesis, University of Massachusetts. COINS Technical Report 92–10.

[Jørgensen, 1962] Jørgensen, J. (1962). *Psykologi – paa biologisk Grundlag*. Scandinavian University Books. Munksgaard, København.

[Mataric, 1994] Mataric, M. J. (1994). Reward functions for accelerated learning. In Cohen, W. W. and Hirsh, H., editors, *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, CA.

[McGovern et al., 1997] McGovern, A., Sutton, R. S., and Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *1997 Grace Hopper Celebration of Women in Computing*.

[Rummery, 1995] Rummery, G. A. (1995). *Problem Solving with Reinforcement Learning*. PhD thesis, Cambridge University Engineering Department.

[Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University.

[Santamaría et al., 1996] Santamaría, J. C., Sutton, R. S., and Ram, A. (1996). Experiments with reinforcement learning in problems with continuous states and action spaces. Technical Report 96-088, COINS.

[Selfridge et al., 1985] Selfridge, O. G., Sutton, R. S., and Barto, A. G. (1985). Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference in Artificial Intelligence*, pages 670–672. Morgan Kaufmann, CA.

[Singh and Sutton, 1996] Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

[Skinner, 1938] Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis*. Prentice Hall, Englewood Cliffs, New Jersey.

[Skinner, 1953] Skinner, B. F. (1953). *Science and Human Behavior*. Collier-Macmillian, New York.

[Staddon, 1983] Staddon, J. E. R. (1983). *Adaptive Behavior and Learning*. Cambridge University Press.

[Sutton, 1996] Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. The MIT Press, Cambridge.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press/Bradford Books.

[Tesauro, 1992] Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

[Tesauro, 1994] Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. Technical report, IBM, Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

[Tesauro, 1995] Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38.

[Watkins, 1989] Watkins, C. J. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University.

A Details of the Bicycle Simulation

The bicycle must be held upright within $\pm 12°$ measured from vertical position. If the angle from the vertical to the bicycle falls outside this interval, the bicycle has fallen, and

the agent receives punishment -1. The Bicycle is modeled by the following non-linear differential equations. One simplification was made to ease the derivation of the equations: The front fork was assumed to be vertical, which is unusual but not impossible. This, however, made the task a bit more difficult for the agent.

There are two important angles in this problem: The angle θ of the direction of the bicycle from straightforward, and the angle ω the bicycle is tiled from vertical. The conservations of angular momentum of the tyres results in some important cross terms.

The equations do not model a bicycle exactly, as some second order cross effects were ignored during the derivation. However we believe that the largest problem of transferring to a real bicycle would be to build hardware that could withstand falling over a thousand times—not just without crashing but also without changing and thereby make the system unstationary.

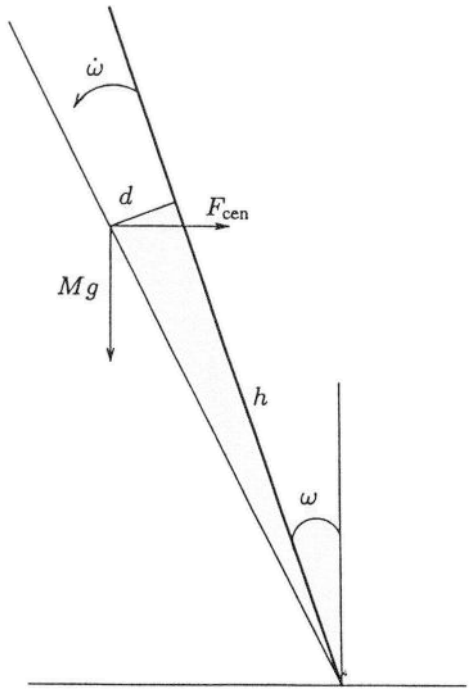

Figure 11: The bicycle as seen from behind. The thick line represents the bicycle. CM is the centre of mass of the bicycle and cyclist.

The following equations describe the mechanics of the system. (See figure 11.) The angle φ is the total angle of tilt of the centre of mass, and is defined as:

$$\varphi \stackrel{\text{def}}{=} \omega + \arctan\left(\frac{d}{h}\right) \tag{1}$$

The angular acceleration $\ddot{\omega}$ can be calculated as:

$$\ddot{\omega} = \frac{1}{I_{\text{bicycle and cyclist}}}\Bigg(Mhg\sin\varphi - \cos\varphi\Big(I_{dc}\,\dot{\sigma}\dot{\theta} + \text{sign}(\theta) \cdot v^2\Big(\frac{M_d r}{r_f} + \frac{M_d r}{r_b} + \frac{Mh}{r_{\text{CM}}}\Big)\Big)\Bigg) \tag{2}$$

This equation is the mechanical equation for angular momentum. The physical contents of the right hand side are terms for the gravitation, effects of the the conservation of angular momentum of the tyres and the fictional centrifugal force. The term $I_{dc}\,\dot{\sigma}\dot{\theta}$ is important for understanding why it is relative easier to ride a bicycle than to keep the balance on a bicycle standing still. The cross effects that originate from the conservation of angular momentum of the tyres stabilize the bicycle, and this effect is proportional to the angular velocity of the tyres $\dot{\sigma}$ and thereby to the velocity of the bicycle.

The angular acceleration $\ddot{\theta}$ of the front tyre and the handle bars is:

$$\ddot{\theta} = \frac{T - I_{dv}\,\dot{\sigma}\,\dot{\omega}}{I_{dl}} \tag{3}$$

These equations are not an exact analytical description, as some second (and higher) order terms have been ignored. The values of ω, $\dot{\omega}$, $\ddot{\omega}$, θ, $\dot{\theta}$ are send to the agent at each time step. The agent returns the value of d and the torque T.

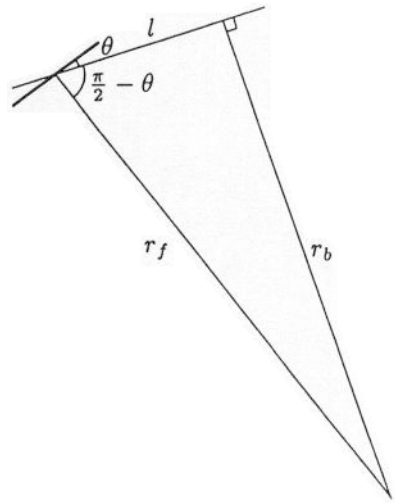

Figure 12: Seen from above. The thick line represents the front tyre.

The front and back tyres follow different paths in a curve with different radii (see figure 12). The front tyre follows

the longest path. The radius for the front tyre is:

$$r_f = \frac{l}{|\cos(\frac{\pi}{2} - \theta)|} = \frac{l}{|\sin\theta|} \tag{4}$$

And for the back tyre:

$$r_b = l \left|\tan\left(\frac{\pi}{2} - \theta\right)\right| = \frac{l}{|\tan\theta|} \tag{5}$$

For the CM the radius can be calculated as:

$$r_{CM} = \left((l-c)^2 + \frac{l^2}{(\tan\theta)^2}\right)^{\frac{1}{2}} \tag{6}$$

The equations of the position of the tyres for the front tyre:

$$\begin{pmatrix} x_f \\ y_f \end{pmatrix}_{(t+1)} = \begin{pmatrix} x_f \\ y_f \end{pmatrix}_{(t)} + v\,dt \begin{pmatrix} -\sin(\psi + \theta + \text{sign}(\psi+\theta)\arcsin(\frac{v\,dt}{2r_f})) \\ \cos(\psi + \theta + \text{sign}(\psi+\theta)\arcsin(\frac{v\,dt}{2r_f})) \end{pmatrix}$$

And for the back tyre:

$$\begin{pmatrix} x_b \\ y_b \end{pmatrix}_{(t+1)} = \begin{pmatrix} x_b \\ y_b \end{pmatrix}_{(t)} + v\,dt \begin{pmatrix} -\sin(\psi + \text{sign}(\psi)\arcsin(\frac{v\,dt}{2r_b})) \\ \cos(\psi + \text{sign}(\psi)\arcsin(\frac{v\,dt}{2r_b})) \end{pmatrix}$$

We estimated the values of the moments of inertia to:

$$I_{\text{bicycle and cyclist}} = \frac{13}{3} M_c h^2 + M_p (h + d_{CM})^2 \tag{7}$$

The various moments of inertia for a tyre was estimated to (see figure 13):

$$I_{dc} = M_d r^2 \tag{8}$$

$$I_{dv} = \frac{3}{2} M_d r^2 \tag{9}$$

$$I_{dl} = \frac{1}{2} M_d r^2 \tag{10}$$

Table 1 shows the values of the parameters used for the bicycle system.

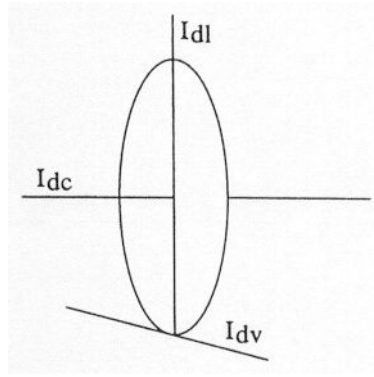

Figure 13: Axis for moments of inertia for a tyre.

Notation		Value
c	Horizontal distance between the point, where the front wheel touches the ground and the CM.	66 cm
CM	The Centre of Mass of the bicycle and cyclist as a total	
d	The agent's choice of the displacement of the CM perpendicular to the plan of the bicycle	
d_{CM}	The vertical distance between the CM for the bicycle and for the cyclist.	30 cm
h	Height of the CM over the ground	94 cm
l	Distance between the front tyre and the back tyre at the point where they touch the ground	111 cm
M_c	Mass of the bicycle	15 kg
M_d	Mass of a tyre	1.7 kg
M_p	Mass of the cyclist	60 kg
r	Radius of a tyre	34 cm
$\dot{\sigma}$	The angular velocity of a tyre	$\dot{\sigma} = \frac{v}{r}$
T	The torque the agent applies on the handlebars	
v	The velocity of the bicycle	10 km/h

Table 1: Notation and values for the bicycle system.