# KaBaGe-RL: Kanerva-based Generalisation and Reinforcement Learning for Possession Football

**Kostas Kostiadis and Huosheng Hu**

Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
Email: kkosti@essex.ac.uk, hhu@essex.ac.uk

**Abstract**

The complexity of most modern systems prohibits a hand-coded approach to decision making. In addition, many problems have continuous or large discrete state spaces; some have large or continuous action spaces. The problem of learning in large spaces is tackled through generalisation techniques, which allow compact representation of learned information and transfer of knowledge between similar states and actions. In this paper Kanerva coding and reinforcement learning are combined to produce the KaBaGe-RL decision-making module. The purpose of KaBaGe-RL is twofold. Firstly, Kanerva coding is used as a generalisation method to produce a feature vector from the raw sensory input. Secondly, the reinforcement learning uses this feature vector in order to learn an optimal policy. The efficiency of KaBaGe-RL is tested using the "3 versus 2 possession football" challenge, a sub-problem of the RoboCup domain. The results demonstrate that the learning approach outperforms a number of benchmark policies including a hand-coded one.

**Keywords:** Kanerva coding, Reinforcement learning, RoboCup, Decision-making, Generalisation.

## 1. Introduction

In simple learning tasks, estimates of value functions can be represented using a table with one entry for each state or for each state-action pair. Although this approach is very clear and easy to use, it is limited to tasks with small numbers of states and actions. Most problems normally have continuous or large discrete state spaces; some have large or continuous action spaces. Using tables to store estimates of value functions in such cases makes inefficient use of experience and normally means impractical memory requirements. The problem of learning in large spaces is tackled through generalisation techniques, which allow compact representation of learned information and transfer of knowledge between similar states and actions. In large smooth state spaces, it is expected that similar states will have similar values and similar optimal actions. Therefore it should be possible to use experience gathered through interaction with a limited subset of the state space and produce a good approximation over a much larger subset.

In most reinforcement learning (RL) architectures, the storage of a variety of mappings is required, including State → Action (policy), State → Reward (value functions), and (State × Action × State)→[0, 1] (transition probabilities). Some of these mappings can be learned using straightforward supervised learning. Popular function approximation techniques for supervised learning include neural network methods [16], fuzzy logic [3, 13], and CMACs [1]. Other mappings, especially the policy mapping, need specialised algorithms because training sets of input-output pairs are not available.

Numerous generalisation techniques, both over input and over actions, are presented in [8]. In this paper the interest is on a special case of gradient-descent methods known as linear methods. In linear methods, the approximate function is a linear function of the parameter vector. Linear methods can be very efficient in practice both in terms of data and computation [19]. Whether or not this is true depends critically on how the states are represented in terms of features. Some of the most popular feature representation techniques include coarse coding [6], tile coding [1], and radial basis functions (RBF) [4]. In this paper the focus is on a feature representation method based on work originally presented in [9] also known as Kanerva coding. The terms "tile coding" and "Kanerva coding" are due to [19]. Although a number of systems have combined linear methods and reinforcement learning [17, 18, 22], to the best of the authors' knowledge there are no reinforcement learning systems that use Kanerva coding.

The rest of the paper is organised as follows. Section 2 introduces the possession football challenge. Section 3 explains how possession football maps onto the episodic RL framework. KaBaGe-RL, the proposed decision making module is presented in section 4, together with its implementation details. Section 5 presents experimental results demonstrating the performance of the proposed learning module. Related work is briefly reviewed in section 6. Finally, conclusions and future work are presented in section 7.

## 2. The Possession Football Challenge

The efficiency of KaBaGe-RL is tested using the "3 versus 2 possession football" challenge, a subtask of the RoboCup

domain [10]. The RoboCup domain provides a challenging testbed for decision-making problems since each agent has to operate in a complex, real-time, noisy environment with the presence of both teammates and adversaries. The 3 versus 2 (or 3v2 for short) possession football challenge was originally introduced by [17] in what they termed the "keepaway" problem. The rest of this section is partly based on their description of the "keepaway" task.

In the 3v2 possession football challenge there is one team of *Attackers*, trying to maintain possession of the ball within a limited region, while another team, the *Defenders,* is trying to gain possession of the ball. Each episode starts with the *Attackers* having possession. As soon as the ball goes outside the region of play or the *Defenders* take possession of the ball, the episode ends and the players are reset for another episode (having possession of the ball means that the ball is within kicking distance from the player). The experiments presented in this paper use a region of 20m x 20m with 3 attackers and 2 defenders, as shown in figure 1.

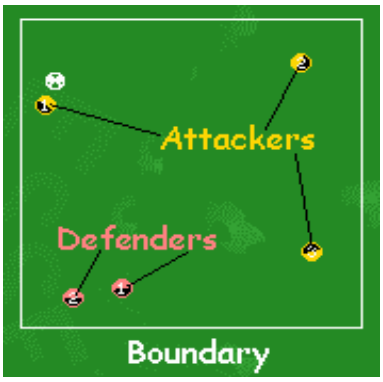



Figure 1: Screen shot from a 3v2 possession football episode in a 20m x 20m region

All the experiments were done using version 6.07 of the standard RoboCup football simulator [14]. An omniscient coach agent manages the play, ending each episode when the ball goes outside the region of play or when the defenders gain possession. The coach then resets the position of the ball and players and starts the next episode. At the beginning of each episode both defenders are placed semi-randomly within certain distance from the bottom-left corner of the play region. The attackers are then placed in the same manner, one in each of the remaining 3 corners of the play region. The ball is placed close to the attacker of the top-left corner. Figure 1 shows a screen shot of an initial configuration for an episode.

To concentrate on the efficiency of the decision-making process of each agent, a simplification was made to the original version of the simulator. In the original version, each agent has a limited field of view (by default each player can only see objects within a 90-degree view cone), and the precision of an object's sensed location degrades with distance. For these experiments the agents are equipped with a full 360-degree noise free vision (actions retain their original noisy characteristics). It is not yet clear whether this simplification is necessary or helpful.

Possession football is a subtask of robotic football. Since there are fewer players involved and the game takes place in a smaller region, each agent can concentrate on the same high-level goal without the need to balance offensive and defensive considerations. Nevertheless, learning to maintain possession of the ball in a smaller region is of great importance in full-scale 11 vs. 11 games, and is one of the first things human players learn to do. The Attackers and Defenders are based on a modified version of the Essex Wizards 2000 multi-agent system (MAS) [7]. A detailed description of the overall design and architecture of the Essex Wizards MAS can be found in [11, 12].

## 3. Possession Football and RL

The RoboCup simulator operates in discrete time steps, t = 1, 2, …, each representing 100ms of simulated time. As soon as one episode ends (either when the ball goes outside the region of play or when the Defenders gain possession), another begins. Each agent is presented with a sequence of these episodes. This can be seen as an instance of the discrete-time, episodic RL framework. As stated earlier, the Defenders' task is to gain possession of the ball, while the Attackers try to maintain possession for as long as possible. For this task, each agent is equipped with the following high-level behaviours:

- ***HoldBall()*** – Remain stationary while keeping the ball in a position that is as far away from the opponents as possible.
- ***PassBall(t)*** – Kick the ball directly towards teammate t.
- ***InterceptBall()*** – Move as fast as possible towards the quickest interception point between the ball's trajectory and your position.
- ***GetOpenToReceive()*** – Move to the least congested position within the region of play in order to receive a pass.

The Defenders' policy space is quite simple. Each defender always executes ***InterceptBall()***. This is necessary since two defenders are required in order to steal the ball from an attacker. If only one defender attempts to steal the ball, the attacker can execute the ***HoldBall()*** behaviour, and maintain possession for a long period of time. Once the ball gets within a defender's kicking range the episode ends. The Attackers' policy space is more complex and can be seen in figure 2.

The most important decision point for an attacker occurs when the ball is within its kicking range. This is

the moment when the agent has to decide whether to hold the ball, pass to teammate 1 (nearest teammate), or pass to teammate 2 (furthest teammate). For this subset of the Attackers' policy space (i.e. {***HoldBall()***, ***PassBall (Teammate 1)***, ***PassBall(Teammate 2)}***), reinforcement learning has been applied in order to optimise their decision making and hence adapt their policies and maximise possession time. This decision-making mechanism was implemented using a variation of the Q-Learning algorithm [23], known as *residual Q-learning* [2].

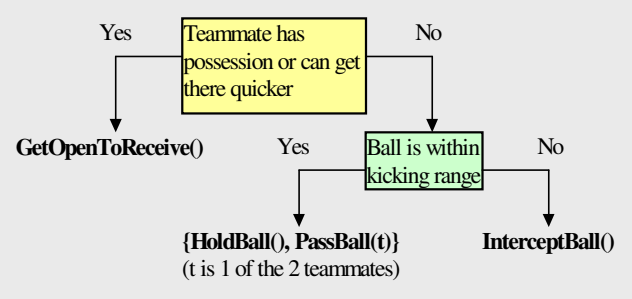



Figure 2: The Attackers' policy space

## 4. Implementing KaBaGe-RL

KaBaGe-RL, the proposed decision making module, (figure 3) is responsible for receiving sensory inputs and producing a desirable action. The first stage filters the raw sensory data and produces a feature vector using Kanerva coding. The second stage uses Q-Learning in order to choose an optimal action given the current input.

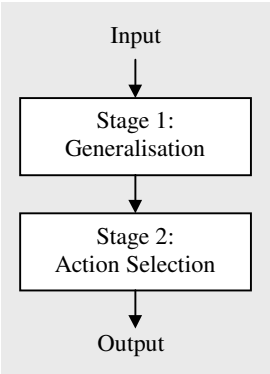



Figure 3: The KaBaGe-RL Decision-Making Module

### Stage 1: Generalisation

The generalisation module (figure 4) is implemented using 4 components connected in a serial manner. The output of each component in the series becomes the input to the next. Initially the raw input is converted to a format that is suitable for use with Kanerva coding (binary format). This step is optional, since the original input signal may already be binary. The binary input is then matched against each prototype. The result of this bit-wise comparison between the two vectors is a positive integer that indicates the number of bits at which they match. This value is then normalised and compared against a threshold. If the normalised result is greater than or equal to the threshold the result is 1, otherwise it is 0. A result of 1 indicates that the given feature exists in the input, whereas a result of 0 indicates that it does not.

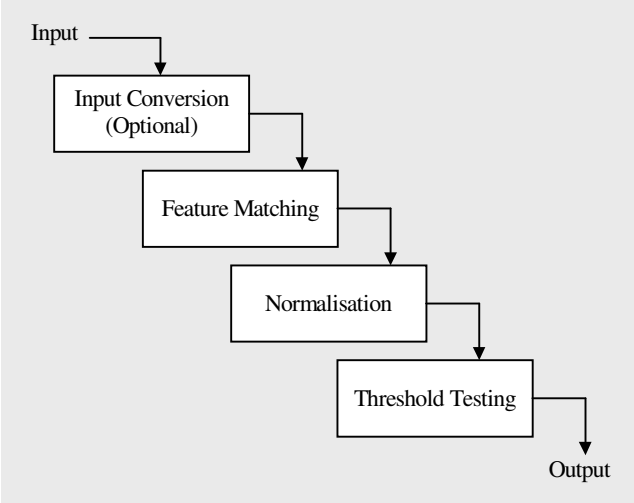



Figure 4: Generalisation Module

The performance of KaBaGe is directly related to the feature representation scheme. A good candidate is one that encapsulates all the useful information needed by an agent in order to make a decision. In addition, for the 3v2 challenge, the chosen scheme must be able to recognise symmetric configurations of the players and the ball. For example, an agent that successfully learned to choose optimal actions when positioned near the top-left corner should be able to use this experience when positioned near any of the remaining three corners. Therefore, the chosen scheme should not be based on the positions of the players and the ball but rather on the distances and angles between them. Such a scheme already exists and was originally used by [17] in a tile coding implementation. According to this representation, the environment is encoded using the following 13 state variables: Dist(A1, C), Dist(A1, A2), Dist(A1, A3), Dist(A1, D1), Dist(A1, D2), Dist(A2, C), Dist(A3, C), Dist(D1, C), Dist(D2, C), Min(Dist(A2, D1), Dist(A2, D2)), Min(Dist(A3, D1), Dist(A3, D2)), Min(Ang(A2, A1, D1), Ang(A2, A1, D2)), Min(Ang(A3, A1, D1), Ang(A3, A1, D2))

where:

- **C** – is the centre of the region of play

- **A1** – is the attacker in possession of the ball (ball shown as a dark circle in figure 5)
- **A2** – is the closest teammate (attacker) to A1
- **A3** – is the second closest teammate (attacker) to A1
- **D1** – is the closest opponent (defender) to A1
- **D2** – is the second closest opponent (defender) to A1
- **Dist(A1, C)** – is the distance between attacker 1 and the centre of the play region
- **Ang(A2, A1, D1)** – is the angle shown in figure 5.

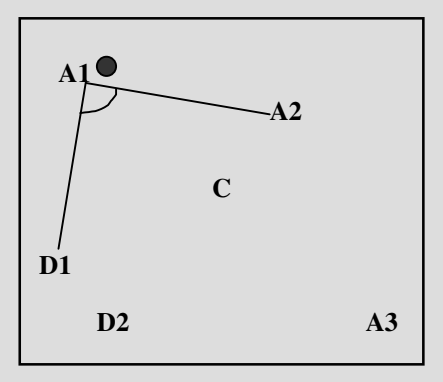



Figure 5: The state variables used by KaBaGe

Recall that Kanerva's implementation only deals with binary features. However, the distances and angles used are not in a binary format. As mentioned earlier, the first step in the KaBaGe module is that of input conversion if the input is not in a binary format. The method used converts all variables into 3-bit binary features. Out of these bits only one is set to 1 and the remaining two are set to 0. For distances those bits represent how close the two objects are. If the objects are very close the first bit is set and the feature appears as (1, 0, 0). Similarly if the two objects are far apart the last bit is set and the feature appears as (0, 0, 1). If the two objects are not too close or too far, then the middle bit is set and the feature appears as (0, 1, 0). For angles the bits represent how wide the angle is and the features are constructed in a similar manner.

The core of the KaBaGe module consists of 5000 semi-randomly generated prototypes, which have 39 bits each (since there are 13 features of 3 bits each). A threshold is used that activates a given prototype only if it has 7 or more features in common with the input vector. It may seem surprising at first how generalisation is achieved by expanding the original input. It may also not be immediately clear how semi-randomly constructed prototypes can yield acceptable levels of performance. In fact [20] have shown that such random representation systems can outperform other methods (such as nearest neighbour and perceptron learning) and the performance of the system only improves as the number of prototypes increases.

**Stage 2: Action Selection**

Recall that the subset of the policy space of each Attacker that is of interest consists of the 3 actions ***{HoldBall(), PassBall(Teammate 1), PassBall(Teammate 2)}***. Also recall that the input to the action selection component is the output of the generalisation component (figure 3). In other words, the 5000 dimensional vector that is produced using Kanerva coding is to be mapped to the 3 actions. This is achieved by connecting all bits of the Kanerva vector to all the actions as shown in figure 6. Since there are 5000 bits and 3 actions, there are 15000 links.

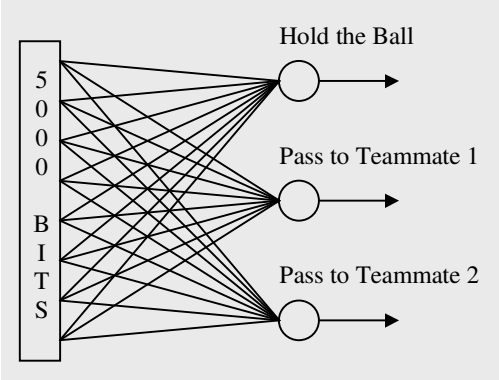



Figure 6: Action Selection

Each of these links is associated with a weight $w_{i\alpha}$ where $i$ is one of 5000 bits and $\alpha$ is one of 3 actions. The action selection is done as follows:

– Add all the weights $w_{i\alpha}$ for each action $\alpha$ and each bit $i$ that is active.

– Execute the action with the highest sum. (Ties were broken according to an arbitrary ordering of the actions.)

Initially, all weights are set to 0. Learning occurs by adjusting these weights given a specified reward. Only the weights for the links between the action that was executed and the bits that were active are updated. All rewards are 0, apart from the reward received at the end of an episode (i.e. when the ball goes outside the region of play or the Defenders gain possession) which is –1. Each of the 3 attackers learns individually, therefore they might learn to behave similarly, or they might learn different, complementary policies. Learning occurs for the attacker that last had possession of the ball. It would be unfair to punish all 3 attackers for the mistake of a single individual. Therefore, when the ball passes safely from one attacker to the next, the attacker that initiated the pass receives a reward of 0 when the ball reaches its teammate successfully. The last attacker that had possession of the

ball before an episode ended receives a reward of –1. The learning mechanism should try to postpone this final negative reward for as long as possible. Also the optimistic initialisation of the weights ensures that all actions will be tried at least once.

The update rule that is used to adjust the weights is that of *residual Q-learning* presented in [2]:

$$\Delta w_r = -\alpha(R + \gamma \max_{u_{t+1}} Q(s_{t+1}, u_{t+1}) - Q(s_t, u_t))$$
$$\times(\phi\gamma \frac{\partial}{\partial w} \max_{u_{t+1}} Q(s_{t+2}, u_{t+1}) - \frac{\partial}{\partial w} Q(s_t, u_t))$$

where $\alpha$ is the learning rate, $\gamma$ is the discount factor, $R$ is the reward, and $\phi \in [0,1]$ is a value that determines the ratio between direct and residual gradient algorithms. The experiments presented in this paper use an $\alpha = 0.000001$, $\gamma = 0.99$, and $\phi = 0.3$.

## 5. Experimental Results

To evaluate the performance of KaBaGe-RL, it was compared against 5 benchmark policies:

(i) ***AlwaysHold()*** – Always hold the ball, never passes.

(ii) ***AlwaysPassNear()*** – Immediately passes to the nearest teammate.

(iii) ***AlwaysPassFar()*** – Immediately passes to the furthest teammate.

(iv) ***Random()*** – Randomly execute one of the 3 possible actions.

(v) ***HandCoded()***:

- *IF* (no Defenders within 10 meters) *THEN* ***HoldBall()***
- *ELSE IF* (teammate t is in a better position *AND* the pass is likely to succeed) *THEN* ***PassBall(t)***
- *ELSE* ***HoldBall()***

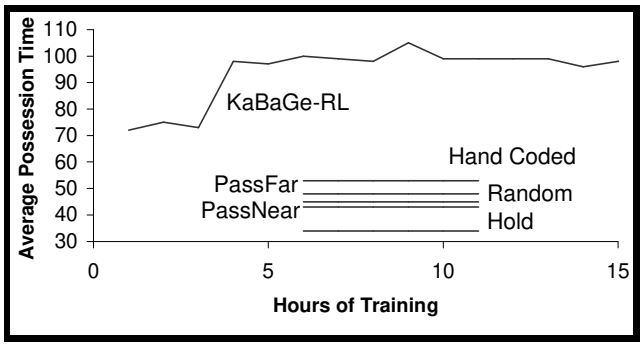



Figure 7: Possession Time vs. Hours of Training (bins of 1000 episodes)

It should be noticed that the first 3 benchmark policies correspond to the 3 actions in the Attackers policy space. Figure 7 shows the average possession time in bins of 1000 episodes against the total training time. The graph clearly demonstrates that the KaBaGe-RL module learns to outperform all benchmark policies, even after the first 1000 episodes.

## 6. Related Work

Reinforcement learning systems have been quite popular within the RoboCup domain. [21] applied vision-based RL on real football playing robots. A robot firstly learnt to shoot the ball into a goal given the state space in terms of the size and the positions of both the ball and the goal in the image, then learnt the same task but with the presence of a goalkeeper. This task differs from possession football in that there is a well-defined goal state (i.e. to score a goal). [15] applied RL in order to learn low-level skills such as kicking, dribbling, and ball interception, as well as co-operative behaviour for the "2 attackers vs. 1 defender" problem (where the 2 attackers try to score a goal against 1 defender). The difference with the work presented here is that they use the full sensor space as the input representation, they use a neural network as a function approximator, and no more than 2 attackers learned to co-operate.

The most closely related work to the one reported here is by [17] who introduced the possession football problem as a subtask of the RoboCup domain. The main differences are that they use CMAC as a function approximator and also in some of their experiments allowed a centralised omniscient coach to estimate the value function from each attacker's perspective. They also use a much softer episode termination condition.

## 7. Conclusions and Future Work

This paper presented a novel decision making module that combines Kanerva coding and reinforcement learning (RL). A number of systems have used RL in the past but to the best of the authors knowledge there are no RL systems that use Kanerva coding. The results presented here have demonstrated that KaBaGe-RL can outperform a number of benchmark policies in a challenging sub-problem of the RoboCup domain.

Future work involves further experimentation in order to combine Kanerva coding with other popular action selection algorithms such as Sarsa. A long-term research goal is also to compare Kanerva coding with other generalisation methods such as principal component analysis [5].

## Acknowledgements

The authors would like to thank the University of Essex for the financial support by providing the Research Promotion Fund DDPB40. Special thanks also go to Gianluca

Baldassarre, Matthew Hunter, Peter Stone, and Rich Sutton for their useful comments and suggestions during development.

**References**

[1] Albus J. S., *Brains, Behaviour, and Robotics*, BYTE Books, Subsidiary of McGraw-Hill, Peterborough, New Hampshire, 1981.

[2] Baird L., *Residual Algorithms: Reinforcement Learning with function approximation*, Proceedings of the 12th International Conference on Machine Learning, pp. 30-37, San Francisco: Morgan Kaufmann, 1995.

[3] Berenji H. R., *Artificial neural networks and approximate reasoning for intelligent control in space,* In American Control Conference, pp. 1075 - 1080, 1991.

[4] Broomhead D. S, and Lowe D., *Multivariable functional interpolation and adaptive networks,* Complex Systems, 2:321-355, 1988.

[5] Castleman K. R., *Digital Image Processing*, Prentice Hall, 1996.

[6] Hinton G. E., *Distributed representations*, Technical report CMU-CS-84-157, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[7] Hu H., Kostiadis K., Hunter M., and Kalyviotis N., *Essex Wizards 2000 Team Description*, In Stone P., Balch T., and Kraetszchmar G., editors, RoboCup-00: Robot Soccer World Cup IV. Springer Verlag, Berlin, 2001.

[8] Kaelbling P. Leslie, Littman L. Michael, Moore W. Andrew, *Reinforcement Learning: a survey,* Journal of Artificial Intelligence 4, pp. 237-285, 1996.

[9] Kanerva P., *Sparse Distributed Memory*, The MIT Press, Cambridge, MA, 1988.

[10] Kitano H., *RoboCup: The Robot World Cup Initiative*, In proceedings of the 1st Inte. Conf. on Autonomous Agents (Agents-97), Marina del Ray, The ACM Press, 1997.

[11] Kostiadis K. and Hu H., *A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition*, In Veloso M., Pagello E., and Kitano H., editors, RoboCup-99: Robot Soccer World Cup III. Springer Verlag, Berlin, 2000.

[12] Kostiadis K., Hunter M., and Hu H., *The Use of Design Patterns for the Development of Multi-Agent Systems*, Proceedings IEEE International Conference on Systems, Man, and Cybernetics (SMC2000), Tennessee, October 2000.

[13] Lee C. C., *A self-learning rule-based controller employing approximate reasoning and neutral net concepts,* International Journal of Intelligent Systems, 6(1):71-93, 1991.

[14] Noda I., Matsubara H., Hiraki K., and Frank I., *Soccer Server: A tool for research on multiagent systems*, Applied Artificial Intelligence, 12, pp. 233-250, 1998.

[15] Riedmiller M., Merke A., Meier D., Hoffman A., Sinner A., Thate O., and Ehrmann R., *Karlsruhe brainstormers – a reinforcement learning approach to robotic soccer*, In Stone P., Balch T., and Kraetszchmar G., editors, RoboCup 2000: Robot Soccer World Cup IV, Berlin: Springer Verlag, 2001.

[16] Rumelhart D. E., and McClelland J. L., editors, *Parallel Distributed Processing: Explorations in the microstructures of cognition,* Vol. 1: Foundations, The MIT Press, Cambridge, MA, 1986.

[17] Stone P., Sutton R. S., and Singh S., *Reinforcement Learning for 3 vs. 2 Keepaway*, RoboCup-2000: Robot Soccer World Cup IV, Stone P., Balch T., and Kreatzschmarr G., editors., Springer Verlag, Berlin, 2001.

[18] Sutton R. S., *Generalisation in reinforcement learning: Successful examples using sparse coarse coding*, In Touretzky D. S., Mozer M. C., and Hasselmo M. E., Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference, pp. 1038-1044, The MIT Press, Cambridge, MA, 1996.

[19] Sutton R. S., and Barto A. G, *Reinforcement Learning an Introduction,* The MIT Press, Cambridge, MA, 1998.

[20] Sutton R. S., and Whitehead S. D., *Online Learning with Random Representations*, Proceedings of the 10th International Conference on Machine Learning, pp. 314-321, Morgan Kaufmann, 1993.

[21] Uchibe E., Asada M., Noda S., Takahashi Y., Hosoda K., *Vision-Based Reinforcement Learning for RoboCup: Towards Real Robot Competition,* Proceedings IROS '96 Workshop on RoboCup, 1996.

[22] Waltz M. D., and Fu K. S., *A heuristic approach to reinforcement learning control systems*, IEEE Transactions on Automatic Control, 10:390-398, 1965.

[23] Watkins C. J. C. H., *Learning from Delayed Rewards*, PhD thesis, King's College, Cambridge, U.K., 1989.