Using A Case-Base of Surfaces to Speed-Up Reinforcement Learning

Chris Drummond
Department of Computer Science
University of Ottawa
150 Louis Pasteur, P.O. Box 450 Stn A
Ottawa, Ontario, Canada, K1N 6N5
fax: (613) 562-5187, phone: none
email: cdrummon@csi.uottawa.ca

Abstract

This paper demonstrates the exploitation of certain vision processing techniques to index into a case-base of surfaces. The surfaces are the result of reinforcement learning and represent the optimum choice of actions to achieve some goal from anywhere in the state space. This paper demonstrates how strong features that occur in the interaction of the system with its environment can be detected early in the learning process. Such features allow the system to identify when an identical, or very similar, task has been solved previously and to retrieve the relevant surface. This results in an orders of magnitude reduction in learning rate.

Review Type: Research Paper

Keywords: Reinforcement Learning, Case-based Learning, Feature Extraction, Snakes

Using A Case-Base of Surfaces to Speed-Up Reinforcement Learning

Chris Drummond
Department of Computer Science, University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
cdrummon@csi.uottawa.ca

Abstract

This paper demonstrates the exploitation of certain vision processing techniques to index into a case-base of surfaces. The surfaces are the result of reinforcement learning and represent the optimum choice of actions to achieve some goal from anywhere in the state space. This paper demonstrates how strong features that occur in the interaction of the system with its environment can be detected early in the learning process. Such features allow the system to identify when an identical, or very similar, task has been solved previously and to retrieve the relevant surface. This results in an orders of magnitude reduction in learning rate.

1 Introduction

One important research issue for case-based learning is its combination with other learning methods. As Aamodt and Plaza [1] point out, generally the machine learning community aims to produce "a coherent framework, where each learning method fulfills a specific and distinct role in the system." This paper discusses one such approach, combining case-based and reinforcement learning. An advantage of the latter method is that it learns even when the information available is very limited. It requires only knowledge of its present state and infrequent real values of reward to learn the actions necessary to bring a system to some desired goal state. As often occurs in achieving such generality, the learning rate is slow. It is important then to exploit the results of prior learning and it would be an definite advantage to quickly recognise that the task, or similar one, has been solved previously.

This paper explores the idea of storing these solutions in a case-base. A critical issue, as in most cased-based systems, is how to recognise and index these cases. The central intuition of this work is that there are features in an environment which are stable and easy to recognise accurately. The environment here includes the system. The features are not “in the world” but in the system’s interaction with the world. For the notion of strong features to be effective, these features must largely dictate the shape of the control surface representing this interaction. Recent research [9] has shown that a function can be accurately reconstructed from just a record of its steep slopes. In this application, tasks exhibiting similar features result in surfaces the same at some qualitative abstraction. If the features differ by a small amount one would expect the surface to differ by a small amount.

In the experiments of section 3, the features are due to the robot’s interaction with walls. It is the existence of such objects with well defined boundaries that makes the notion of edge extraction central to much of vision processing. This paper takes much of its insight from work done in vision and is closely related to Marr’s early work [10]. The features are essentially Marr’s zero crossing points. It will demonstrate the use of a snake, a popular technique in object recognition [12], to locate and characterise these features. One important difference in this research, is that recognising the task is not an answer in itself but is used to initialise another learning process. Thus the recognition process has only to be accurate enough to produce a significant speedup in the learning process, averaged over many learning episodes.

But what of other situations where objects with well defined boundaries are not present? In Herve et al’s research [5] in robot arm control, there are also strong features, the singularities in the arm’s work space. In Aha’s paper [2] learning is particularly successful where the control surface is smooth, but encounters difficulties in the area of strong non-linearities. This work aims to identify such features, not only because they are easy to recognise but also because they have such an impact on learning.

In the rest of the paper, section 2 will discuss the basic form of the case base. Section 3 presents experiments demonstrating feature extraction and how this leads to a substantial reduction of learning rate. Section 4 will present background material on the generation of the snake, the underlying spline representation and reinforcement learning.

2 The Case-Base

The case-base itself consists of two parts: the cases and the indices. A case is a surface obtained using reinforcement learning on a particular task. The case-base will contain a number of such surfaces representing the result of different learning episodes. An index is based on strong features of these surfaces determined through a contour method called a snake.

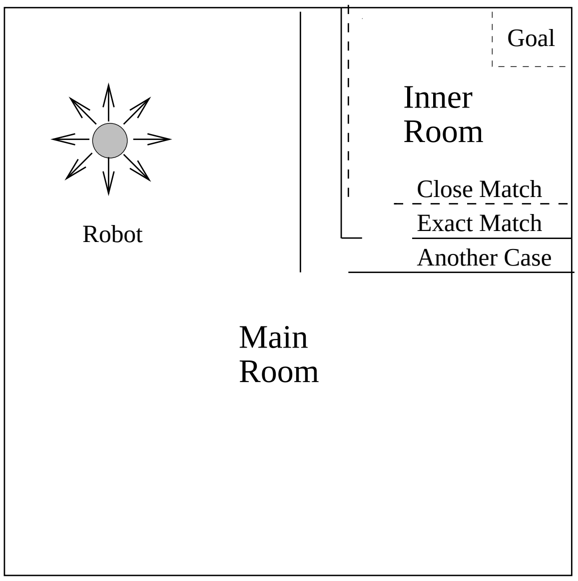

Figure 1: A room within a room

Figure 1 shows a simple reinforcement learning problem. A robot moves in any of 8 directions within an environment of a main room and an inner room containing the goal. In the reinforcement learning framework, an action taken that moves the robot from one state to another, results in a percentage of the value of the new state being assigned to the originating state. Initially only the goal states are non-zero, but eventually this process will propagate back values to all states. This forms a surface over the state space, which in this research becomes the case. Once all these values have stabilised, a greedy algorithm moving from one state to the neighbouring state with the highest value, will take the shortest path to the goal. Figure 2 shows such a surface produced by the learning process (denoted "another case" in figure 1). In this example the "strong" features are the two steep slopes associated with the walls of the inner room.

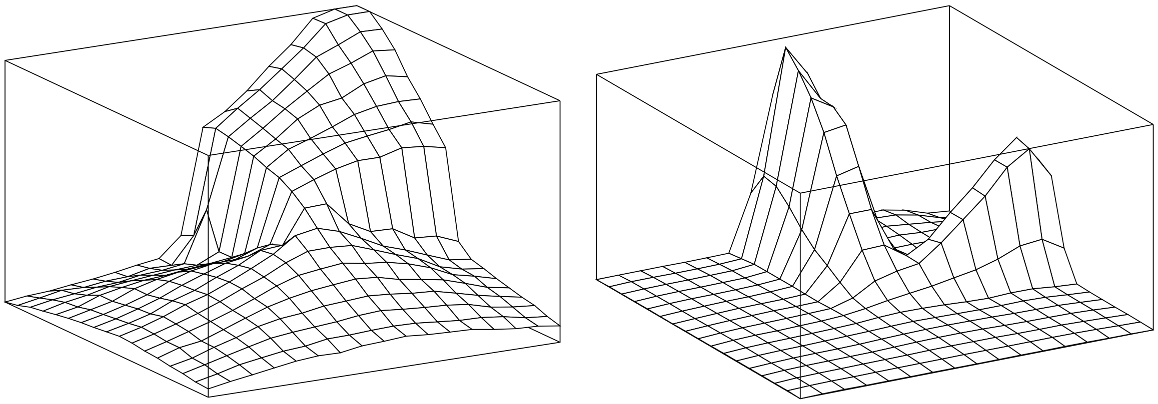

Figure 2: A Surface and its Gradient Vector

Figure 2 also shows a plot representing the magnitude of the gradient vector of this surface. This is the absolute value of the largest gradient in any direction, it forms hills coincident with the steep parts of the surface. The snake is started off in an approximate initial position on this gradient surface. Then the process is essentially one of hill climbing, with the snake ending up lying along the ridge of the hills, figure 3. The process being terminated when the snake is stationary. A function that represents the characteristics of the gradient along the snake can then be extracted. In this example, the ridge dips close to zero, at a point related to the slow ramp between the two steep slopes in the original surface. The height of the snake over this differential will be close to zero in this region and much larger elsewhere.

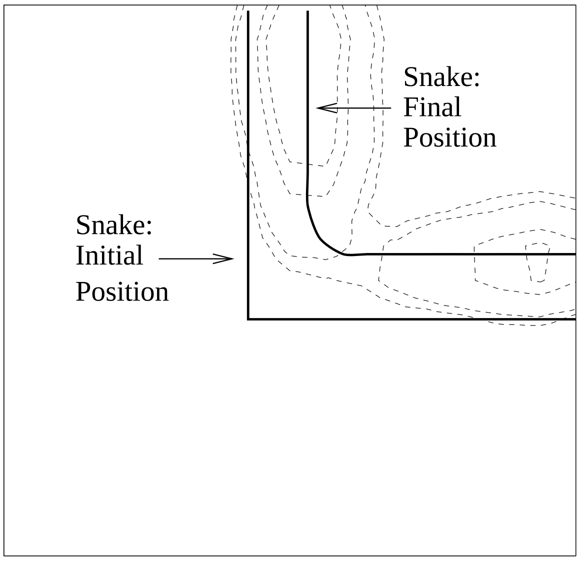

Figure 3: Fitting The Snake

The index is constructed from these two parts: the position of the snake on the surface and the shape of the extracted function. When a new task is being learnt a snake is applied to the surfaces produced early in learning process. The aim is to establish the surfaces with similar feature positions and shape characteristics. To achieve this a distance measure between the new snake and the index is used, based on the weighted sum of two norms: one for their relative positions and one for their similarity of function shape. The best match is then removed from the case-base, and the case itself used to initialise the learning system with the surface that closely matches the task in hand. If the match is exact, this results in the learning being complete, if not, further refinement is necessary but much less than in a normal learning process.

3 Experiments

The following experiments demonstrate that the features can be detected very early in the learning process and are effective in indexing into a case-base. Even when there is not an exact match, using a surface from a close match results in a considerable improvement in learning rate.

The experiments use the environment discussed in section 2, as shown in figure 1. The goal remains fixed, but 10 different sized inner rooms with different door positions and widths were used. The state space was divided into a 17 by 17 array of discrete cells. The basic Q-learning algorithm was used, with a discrete function approximator as discussed in section 4.2. The algorithm was run for each room configuration until there was no appreciable change in the surface. An initial position for the snake was established and then the method discussed in section 4.1 used to locate the features. The snake position and related function were extracted to form the index, the discrete functional approximation the case. One of these surfaces was designated the test case and is shown in figure 4. This is associated with environmental configuration denoted "exact match" in figure 1. Adjacent to that is the magnitude of its gradient vector.

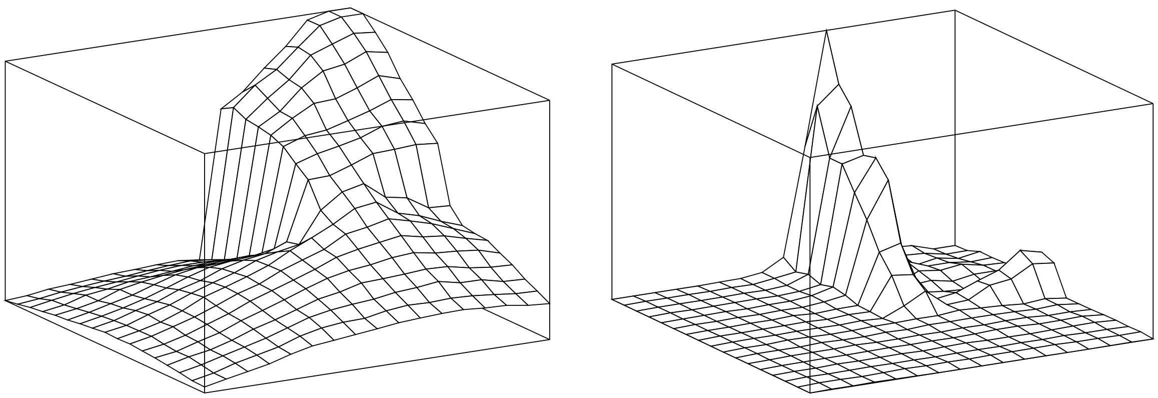

Figure 4: Test Case

To start a new learning episode the arrays were initialised to the values for the goal position without any walls present, effectively the default case in the base. The algorithm was then run in increments of 1000 actions up to 100000 and the Euclidean norm of the difference between the current surface and converged surface recorded. This was then repeated but this time, the snake was fitted to each of the resulting surfaces at each increment and the function extracted. If the mean value of the function exceeded a threshold, used to prevent the updating of the surface prematurely and set at 0.5, the best matching case was retrieved. The best match was determined using a distance measure between the old and new snake. First, the extracted functions were thresholded using the mean of their values. Then, an Euclidean norm of the difference of the result was computed. The snakes' positions were compared by the same norm for both x and y values. The weighted sum of the two norms determined the distance, with a weight of ten for the position and one for the function.

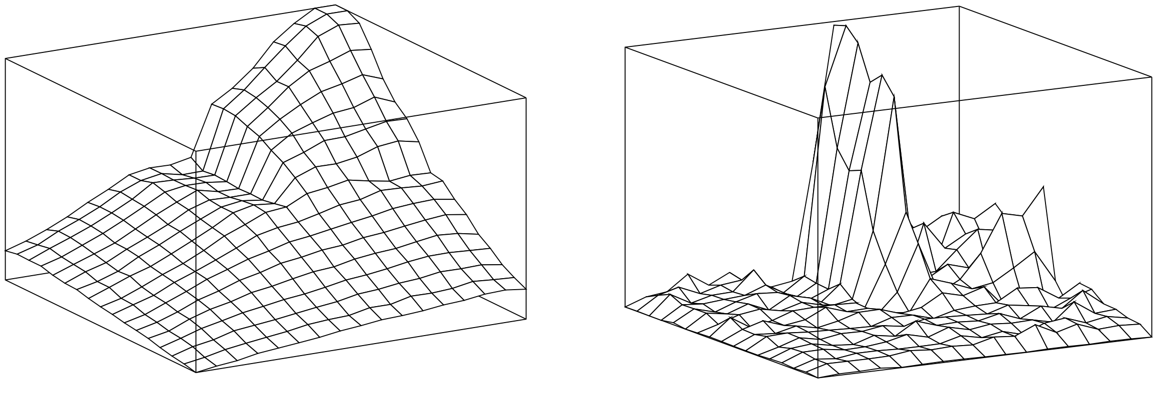

Figure 5: Surface After 2000 Actions

In the first experiment the test surface was left out of the case-base. The algorithm chose as the best match a slightly smaller room with a door in a similar position, denoted "close match" in figure 1. The arrays were then updated from those in the case base. In the second experiment the test surface was added to the case-base and on repeating the procedure the actual surface was selected. In both experiments, a case was selected after

2000 actions, based on features extracted from the surface shown in figure 5. The magnitude of the gradient vector shows roughly the same form as the test case and this allowed the successful extraction of the features so early in the learning process.

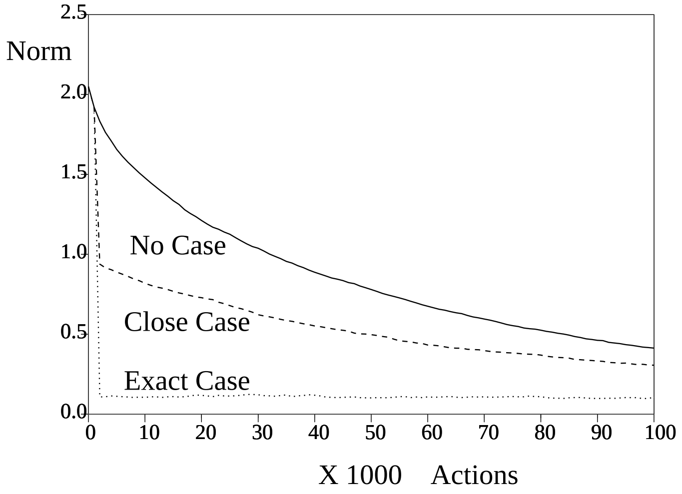

Figure 6: The Learning Rate

Figure 6 shows the learning curve produced in these experiments, as measured by taking the Euclidean norm. Each curve starts from the default surface being introduced. The topmost curve shows the normal learning rate every 1000 actions, as can be seen at the lower end it is slowly converging to an asymptotic value close to zero. It may not actually reach zero as there is some stochastic variation in the surface due to the selection policy of the action, but it will oscillate around a very low value. In both subsequent experiments, after 2000 actions the new snake is considered to be sufficiently close to an index in the case-base. The second curve shows the result of using a surface that is close to the desired one. The learning rate drops dramatically as the old surface is introduced. From then on it slowly drops towards the asymptotic value. The curve is effectively the result of shifting the position at 40000 actions back, so the learning rate is increased by a factor of twenty. When the exact solution is used the learning rate drops to its minimum value.

4 Some Technical Details

This section gives some of the technical details on producing a snake and its underlying spline representation, and the reinforcement learning algorithm used.

4.1 Splines and Snakes

This work has made extensive use of splines: in one dimension to represent the snake and in two dimensions to fit the discrete data to allow the extraction of various differentials. A spline produces a function that are piecewise polynomials. Although it is quite smooth, by giving up continuity in some of the higher derivatives the shape of the function is more locally controlled than say polynomials. This improves its stability and makes operations on matrices fast due to their sparsity. Splines are typically constructed from basis functions which are translates of a single canonical function. One such family is the b-splines. The cubic b-spline in figure 7 is of order four, each piece is a cubic polynomial and the second derivative is continuous. Typically higher dimension basis functions are produced by a simple product, the basis function value at a point is just the product of the values of one dimensional basis functions at their respective coordinates.

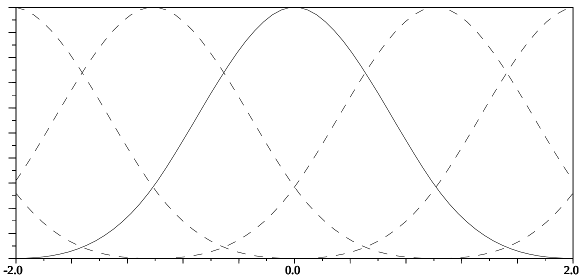

Figure 7: Cubic B-splines

A snake is a deformable contour with a couple of elastic properties: resistance to stretching and resistance to bending. The snake is represented by a spline in a parametric form, equation 1 to allow it to produce closed contours. To produce such a form requires two splines one for the x coordinates, equation 2, another for the y, equation 3.

$$g(u) = (x(u), y(u)) \quad (1) \qquad x(u) = \sum_{j=1}^{m} c_j \phi_i(u) \quad (2) \qquad y(u) = \sum_{j=1}^{m} k_j \phi_i(u) \quad (3)$$

The contour that the snake is intended to approximate is where there is a sharp increase in value of the surface. This occurs where magnitude of the gradient vector of the surface is at its maximum, at the zero crossing of the second derivative. To determine the position of the snake it is necessary to find the function g that minimises its energy, given in equation 4. So the field is defined as the negative of this quantity, equation 5.

$$E_{snake}(g) = \int_R (E_{field}(g(u)) + E_{internal}(g(u)))du \tag{4}$$

$$E_{field} = -|\nabla(x, y)|^2 = -[(\frac{\partial Surface}{\partial x})^2 + (\frac{\partial Surface}{\partial y})^2] \tag{5}$$

This energy equation balances the internal energy of the snake with that due of the gradient. The internal energy is made up of two parts: the term containing the first derivative which gives it the resistance to stretching and the term containing the second derivative that gives it resistance to bending, equation 6. The effect of these terms may be varied along the length of the snake by the use of the weighting functions $\alpha(s)$ and $\beta(s)$.

$$E(g) = -\int_R E_{field} du + \int_R \alpha(s)(\frac{dg(u)}{du})^2 du + \int_R \beta(s)(\frac{d^2 g(u)}{du^2})^2 du \tag{6}$$

By differentiating with respect to the coefficients of the basis functions and equating the resulting equations to zero, a set of simultaneous equations is produced. Equation 7 shows one row for the x coordinates another set is required for the y coordinates. With an initial position of the snake, the left hand side of equation can be solved by Gaussian quadrature. The resulting matrix equations are shown in equations 8 and 9. They must be solved iteratively until the snake is stationary. Additional terms are added to smooth the iteration process, and which also improves the condition of the matrices [6].

$$\Sigma_j c_j \left(\int_R \alpha(u) \left(\frac{d\phi_i(u)}{du} \frac{d\phi_j(u)}{du} \right) + \beta(u) \left(\frac{d^2\phi_i(u)}{du^2} \frac{d^2\phi_j(u)}{du^2} \right) du \right) = \int_R \frac{\partial E_{field}}{\partial x} \phi_i(u) du \tag{7}$$

$$Hc^t = b_x(c^{t-1}, k^{t-1}) \tag{8}$$

$$Hk^t = b_y(c^{t-1}, k^{t-1}) \tag{9}$$

4.2 Reinforcement Learning

Reinforcement learning [13] is a family of algorithms that addresses the problem of credit assignment. In many learning situations feedback on the success of actions is infrequent and delayed. So actions that receive no direct feedback but are stage setting to those that do, must somehow be rewarded. To achieve this a system is trained not by waiting until the final outcome of the experiment but rather by correlating it to its subsequent predictions. Thus in a system designed to predict win in a game of chess, if the subsequent position predicts a win, the present position is biased towards predicting a win.

The basic idea in reinforcement learning is to determine the expected discounted reward for each state. Both the reward and result of an action can be stochastic. The system explores the world, learning for each action in each state the expectation of the sum of future rewards. These are often discounted the farther they are into the future, $R = E[\sum_{t=1}^{\infty} \gamma^t r_t]$ where γ, the discount factor, is less than or equal to one. The two algorithms most frequently used to address this problem are Q-learning [17] and TD(λ) [14]. This research adopts the Q-learning algorithm, the simpler of the two. This algorithm assumes the world is a discrete Markov process thus both states and actions actions are discrete. For each action a in each state s, Q-learning maintains a rolling average of the immediate reward r plus the value for the best action a' of the next state s', formula 10. Watkins and Dayan [17] proved that this will converge to the optimal value with certain constraints on the reward and the learning rate α. Because of the nature of the problem representation, the optimal solution from any state is to take the best action from the present state and then take the optimal solution from then on.

$$Q_{s,a}^{t+1} = (1-\alpha) Q_{s,a}^t + \alpha(r + \gamma max_{a'} Q_{s',a'}^t) \tag{10}$$

The extension to continuous spaces can be done using function approximation. The simplest method is to divide both the state and action dimensions into intervals. Each resulting cell then represents the average of a range of actions taken from a region in the state space. In many applications this method is successful but there exists no general proof of its convergence.

5 Limitations and Future Work

There are a number of limitations in the present form of the system. Firstly, a single snake is used and it will only to locate features that are connected to the boundaries of the state space. In environments of a more complex collection of rooms and walls, it will be necessary to use multiple snakes. This should present no real technical problem but will somewhat complicate the matching algorithm. Secondly, at present surfaces that only approximately match a new task are used without modification. Once detecting the position of the features, it should be possible to modify the case, as is typical in many case-based systems, to better initialise the learning process. Future work will involve solving these problems and investigating the idea of using a planning system to construct a new more complex surface from several existing cases. This would then be further refined through reinforcement learning, ultimately becoming itself a new case that can be combined with others.

6 Related Work

The most closely connected work, to that presented in this paper, is the various forms of instance based or case-based learning that have been used in conjunction with reinforcement learning. Their use has principally been to address two problems: the economical representation of the state space and dealing with hidden state. The former uses learnt instances combined with various forms of local regression. Both Peng [11] and Tadepalli [15] use linear regression over a set of neighbouring points. McCallum [7] uses a tree representation to expand the state representation to include prior states to remove ambiguity due to hidden states. In further work [8] he uses a single representation to address both the hidden state problem and the general problem of representing a large state space by using a case-base of sequences of states associated with various trajectories. Unlike this other research, in the work presented here the case is not an example of the value function during learning. It is the result of a complete learning episode, so the method should be complementary to these other approaches.

This work is also related to case-based planning [4, 16]. This is partly through the general connection of reinforcement learning and planning. But also, as discussed in the previous section, through one of the overall goals of this work in that the learnt surfaces will become cases in a planning system. Last but not least, is the connection with object recognition in vision research [12, 3]. In many ways model bases are similar to case bases. In this work, many of the methods if not the final application has come from such work.

7 Conclusions

This paper has shown how the use of a case-base of surfaces can significantly improve the speed of reinforcement learning. It introduced the notion of using vision processing techniques to extract features from the evolving surface. The experiments demonstrated that the features are stable and can be extracted early in the learning process. Further work is needed to see if this technique is as effective in domains where there are no well defined objects in the environment.

References

[1] Agnar Aamodt and Enric Plaza (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* AICom - Artificial Intelligence Communications, V. 7, No. 1 pp 39-37

[2] Aha, D. W., and Salzberg, S. L. (1993). *Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks.* In Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics. Unpublished. pp. 363-368

[3] C. h. Chin and C. R. Dyer (1986) *Model-based recognition in Robot Vision* Computing surveys V. 18 No 1 pp 67-108

[4] Kristian J. Hammond (1990) *Case-Based Planning: A Framework for Planning from Experience* The Journal of Cognitive Science V 14 no. 3

[5] Jean-Yves Herve and Rajeev Sharma And Peter Cucka *The Geometry of Visual Coordination.* Proceedings of the Ninth National Conference on Artificial Intelligence pp 732-737

[6] Frederic Leymarie and Martin D. Levine. (1993) *Tracking Deformable Objects in the Plane Using an Active Contour Model* IEEE Transactions On Pattern Analysis And Machine Intelligence V 15 No. 6 pp 617-634

[7] R. A. McCallum, (1995). *Instance-based utile distinctions for reinforcement learning.* Proceedings of the Twelfth International Conference on Machine Learning. pp 387-395

[8] R. A. McCallum (1995). *Instance-based state identification for reinforcement learning.* Advances in Neural Information Processing Systems 7. pp 377-384

[9] Staphane Mallat and Sifen Zhong (1992). *Characterization of Signals from Multiscale Edges.* IEEE Transactions On Pattern Analysis And Machine Intelligence V 14 No. 7 pp 710-732

[10] David Marr (1982) Vision : a computational investigation into the human representation and processing of visual information pub W.H. Freeman

[11] Jing Peng (1995) *Efficient Memory-Based Dynamic Programming.* Proceedings of the Twelfth International Conference of Machine Learning pp 438-439

[12] P. Suetens and P. Fua and A. Hanson (1992) *Computational strategies for object recognition.* Computing surveys V. 24 No. 1 pp 5-61

[13] R.S. Sutton (1988) *Learning to Predict by the Methods of Temporal Differences* Machine Learning V. 3 pp 9-44

[14] R.S. Sutton (1990) *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming* Seventh International Conference on Machine Learning pp 216-224

[15] P. Tadepalli and D. Ok (1996) *Scaling up Average Reward Reinforcement Learning by Approximating the Domain Models and the Value Function.* Proceedings of International Machine Learning Conference. pp 471-479

[16] Manuela M. Veloso and Jaime G. Carbonell (1993) *Derivational Analogy in PRODIGY: Automating Case Acquisition, storage and Utilization* Machine Learning V. 10 No. 3 pp 249-278

[17] Christopher J.C.H. Watkins and Peter Dayan (1992) *Technical Note:Q-Learning* Machine Learning V. 8 No 3-4 pp 279-292