In: Hassanien, A.E., Suraj, Z., Slezak, D., Lingras, P., Eds.: Rough Computing. Theories, Technologies, and Applications. Information Science Reference, Hersey, NY (2008) 186-203.

Monocular Vision System that Learns with Approximation Spaces

J.F. Peters, Maciej Borkowski, Christopher Henry, Dan Lockery

Department of Electrical & Computer Engineering, University of Manitoba
75A Chancellor's Circle, Winnipeg, Manitoba R3T 5V6 Canada
{jfpeters,maciey,chenry,dlockery}@ee.umanitoba.ca

Abstract. This paper introduces a monocular vision system that learns with approximation spaces to control the pan and tilt operations of a digital camera that is tracking a moving target. This monocular vision system has been designed to facilitate inspection by a line-crawling robot that moves along an electric power transmission line. The principal problem considered in this article is how to use various forms of reinforcement learning to control movements of a digital camera. Prior work on the solution to this problem was done by Chris Gaskett using neural Q-learning starting in 1998 and more recently by Gaskett in 2002. However, recent experiments have revealed that both classical target tracking as well as other forms of reinforcement learning control outperform Q-learning. This article considers various forms of the Actor Critic (AC) method to solve the camera movement control problem. Both the conventional AC method as well as a modified AC method that has a built-in run-and-twiddle (RT) control strategy mechanism, are considered in this article. The RT mechanism introduced by Oliver Selfridge in 1981 is an action control strategy, where an organism continues what it has been doing while things are improving (increasing action reward) and twiddles (changes its action strategy) when past actions yield diminishing rewards. In this work, RT is governed by measurements (by a critic) of the degree of overlap between past behaviour patterns and a behavior pattern template representing a standard are carried out within the framework provided by approximation spaces introduced by Zdzisław Pawlak during the early 1980s. This paper considers how to guide reinforcement learning based on knowledge of acceptable behavior patterns. The contribution of this article is an introduction to actor critic learning methods that benefit from approximation spaces in controlling camera movements during target tracking.

Keywords: Actor critic method, approximation space, monocular vision, reinforcement learning, rough sets, run-and-twiddle, target tracking.

1 Introduction

The problem considered in this paper is how to guide action choices by an actor that is influenced by a critic governed by the evaluation of past actions. Specifically, one might ask how to measure the value of an action relative to what has been learned from experience (i.e., from previous patterns of behavior), and how to learn good policies for choosing rewarding actions. The solution to this problem stems from a rough set approach to reinforcement learning by cooperating agents. It is an age-old adage that experience is a good teacher, and one learns from experience. This is at the heart of reinforcement learning, where estimates of the value of an action are based on past experience.

In reinforcement learning, the choice of an action is based on estimates of the value of a state and/or the value of an action in the current state. A swarm learns the best action to take in each state by maximizing a reward signal obtained from the environment. Three different forms of actor-critic (AC) method are investigated in this article, namely, a conventional AC method and a form of AC method that includes an adaptive learning strategy called run—and—twiddle (RT) played out in the context of remembered behavior patterns that accumulate in what are known as ethograms. An *ethogram* is a table of stored behavior patterns (i.e., vectors of measurements associated with behavior features) borrowed from ethology by Tinbergen (1963). Quantitative comparisons of past behavior patterns with a template representing "normal" or desirable behavior are carried out within the framework an approximation space. Approximation spaces were introduced by Zdzislaw Pawlak (1981) during the early 1980s, elaborated by Orlowska (1982), Pawlak (1982), and generalized by Skowron and Stepaniuk (1995) and Stepaniuk (1998). The motivation for considering approximation spaces as an aid to reinforcement learning stems from the fact that it becomes possible to derive pattern-based action preferences (see, *e.g.*, Peters and Henry (2005a, 2005b)).

Prior work on the reinforcement learning control was done by Chris Gaskett (2002) using neural Q-learning starting in 1998. However, recent experiments have revealed that both classical target tracking as well as other forms of reinforcement learning control outperform Q-learning. Consideration of Gaskett's particular version of Q-learning and neural networks as means of camera movement control is outside the scope of this article. This article considers various forms of the Actor Critic (AC) method to solve the camera movement control problem. AC methods have been studied extensively (see, *e.g.*, Barto (1983), Berenji (2003), Bertsekas (1996), Konda (2000), Rosenstein (2003), Sutton and Barto (1998), Watkins and Dayan (1992), Wawrzynski (2005)}). The conventional actor critic method evaluates whether things have gotten better or worse than expected as a result of an action-selection in the previous state. A temporal difference (TD) error term δ is computed by the critic to evaluate an action previously selected. An estimated action preference in the current state is then determined by an actor using δ. Swarm actions are generated by a policy that is influenced by action preferences. In the study of swarm behavior of multiagent systems such as systems of cooperating bots, it is helpful to consider ethological methods (see, *e.g.*, Tinbergen (1963)), where each proximate cause (stimulus) usually has more than one possible response. Swarm actions with lower TD error tend to be favored. A second form of actor critic method is defined in the context of an approximation space (see, *e.g.*, Peters(2004a), Peters (2004b), Peters (2005), Peters and Ramanna (2004), Peters and Henry (2006), Peters, Skowron, Synak, Ramanna (2003), Skowron and Stepaniuk (1995), Stepaniuk (2002)) and which is an extension of recent work with reinforcement comparison (Peters (2005), Peters and Henry (2005a), Peters and Henry (2005b), Peters and Henry (2006)). This form of actor critic method utilizes what is known as a reference reward, which is pattern-based and action-specific. Each action has its own reference reward which is computed within an approximation space that makes it possible to measure the closeness of action-based blocks of equivalent behaviors to a standard. The contribution of this article is an introduction to a form of monocular vision system that learns with approximation spaces used as frameworks for pattern-based evaluation of behavior during reinforcement learning.

This article is organized as follows. Rough set theory is briefly introduced in Sect. 2. The basic idea of an approximation space is presented in Sect. 3. An approach to commanding a monocular vision system is described in Sect. 4. A description of a testbed for learning experiments with a monocular vision systems is given in Sect. 5. A model for the design of a monocular vision system that learns with approximation spaces is given in Sect. 6. A rough coverage form of actor critic method is presented in Sect. 7. Finally, the results of learning by a monocular vision system in a non-noisy and in a noisy environment are presented in Sections 8 and 9.

2 Rough Sets: Basic Concepts

This section briefly presents some fundamental concepts in rough set theory resulting form the seminal work by Zdzisław Pawlak (for an overview, see, *e.g.*, Peters and Skowron (2006, 2007)) and that provides a foundation for a new approach to reinforcement learning by collections of cooperating agents. The rough set approach introduced by Zdzisław Pawlak (1981a, 1981b, 1982) and Pawlak and Skowron (2007a, 2007b, 2007c) provides, for example, a ground for concluding to what degree a set of equivalent objects are covered by a set of objects representing a standard. The term ``coverage" is used relative to the extent that a given set is contained in a standard set. An overview of rough set theory and applications is given by Polkowski (2002). For computational reasons, a syntactic representation of knowledge is provided by rough sets in the form of data tables. A data (information) table IS is represented by a pair (U, A), where U is a non-empty, finite set of elements and A is a non-empty, finite set of attributes (features), where $a: U \rightarrow V_a$ for every $a \in A$ and V_a is the value set of a. For each $B \subseteq A$, there is associated an equivalence relation $Ind_{IS}(B)$ such that $Ind_{IS}(B) = \{(x, x') \in U2 \mid \forall a \in B, a(x)=a(x')\}$. Let $U/Ind_{IS}(B)$ denote a partition of U determined by B (i.e., $U/Ind_{IS}(B)$ denotes the family of all equivalence classes of relation $Ind_{IS}(B)$, and let $B(x)$ denote a set of B-indiscernible elements containing x. $B(x)$ is called a block, which is in the partition $U/Ind_{IS}(B)$. For $X \subseteq U$, the sample X can be approximated from information contained in B by constructing a B-lower and B-upper approximation denoted by B_*X and B^*X, respectively, where $B_*X = \cup\{ B(x) \mid B(x) \subseteq X\}$ and $B^*X = \cup\{ B(x) \mid B(x) \cap X \neq \emptyset\}$. The B-lower approximation B_*X is a collection of blocks of sample elements that can be classified with full certainty as members of X using the knowledge represented by attributes in B. By contrast, the B-upper approximation B^*X is a collection of blocks of sample elements representing both certain and possibly uncertain knowledge about X. Whenever B_*X is a proper subset of B^*X, i.e., $B_*X \subset B^*X$, the sample X has been classified imperfectly, and is considered a rough set.

2 Approximation Spaces

This section gives a brief introduction to approximation spaces. The basic model for an approximation space was introduced by Pawlak (1981a), elaborated by Orlowska (1982) and Pawlak (1981b), generalized by Skowron and Stepaniuk (1995), Stepaniuk 1998), and applied in a number of ways (see, e.g., Peters (2005), Skowron, Swiniarski, and

Synak (2005), Peters and Henry (2006)). An approximation space serves as a formal counterpart of perception or observation (Orlowska (1982)), and provides a framework for approximate reasoning about vague concepts. A *generalized approximation space* is a system $GAS = (U, N, \nu)$ where

- U is a non-empty set of objects, P(U) is the powerset of U,
- $N : U \rightarrow P(U)$ is a neighborhood function,
- $\nu : P(U) \times P(U) \rightarrow [0, 1]$ is an overlap function.

A set $X \subseteq U$ is definable in a GAS if, and only if X is the union of some values of the neighborhood function. In effect, the uncertainty function N defines for every object x a set of similarly defined objects. That is, N defines a neighborhood of every sample element x belonging to the universe U (see, e.g., Peters, Skowron, Synak and Ramanna (2003)). Specifically, any information system $IS = (U, A)$ defines for any feature set $B \subseteq A$ a parameterized approximation space $AS_B = (U, N_B, \nu)$, where $N_B = B(x)$, a B-indiscernibility class in the partition of U. The overlap function ν computes the degree of overlap between two subsets of U. Let $P(U)$ denote the powerset of U. We are interested in the larger of the two sets (assume that the $card(Y) \geq card(X)$) because we want to see how well Y ``covers" X, where Y represents a standard for evaluating sets of similar objects. Standard rough coverage (SRC) ν_{SRC} can be defined as in Eq. 1.

$$\nu_{SRC}(X,Y) = \begin{cases} \frac{|X \cap Y|}{|Y|}, & if\ Y \neq \varnothing, \\ 1, & if\ Y = \varnothing. \end{cases} \quad (1)$$

In other words, $\nu_{SRC}(X, Y)$ returns the degree that Y covers X. In the case where $X = Y$, then $\nu_{SRC}(X, Y) = 1$. The minimum coverage value $\nu_{SRC}(X, Y) = 0$ is obtained when $X \cap Y = \varnothing$ (i.e., X and Y have no elements in common).

3 Environment for Line-Crawling Robot

This section briefly describes the environment to be inspected by a second generation Autonomous Line-Crawling robot named ALiCE II, which is a member of a family of line-crawling bots designed to inspect electric power transmission towers (see Figs. 1 and 2).

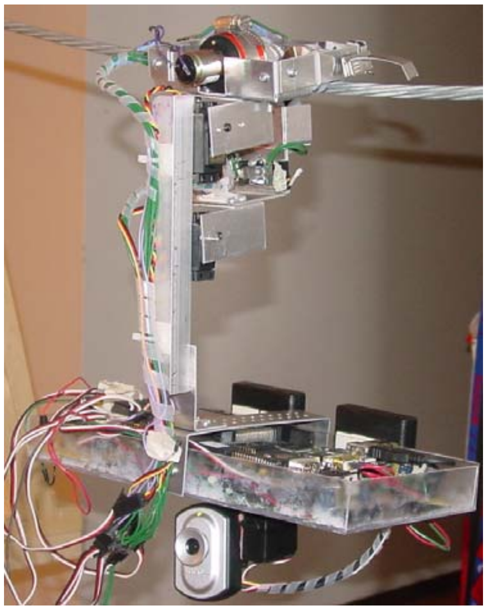


Fig. 1. ALiCE II Camera

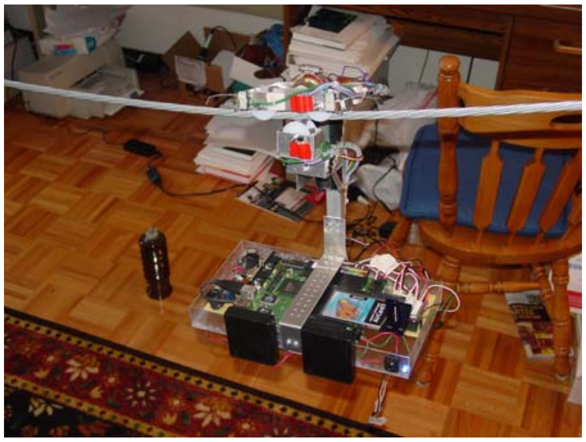


Fig. 2. ALiCE II Wireless Card

ALiCE II has been designed to crawl along a wire stretched between a particular class of steel towers in the Manitoba Hydro power transmission system. Sample steel towers in the Manitoba hydro system are shown in Figs. 3 and 5. ALiCE II has been designed to crawl along the top, lightning guard wire (called a sky wire) of a tower like the one shown in Fig. 5. A bot suspended from a sky wire stretched between electric power transmission towers usually sways from side--to--side as it moves along a wire due to buffeting by the wind. For example, in Manitoba towers usually range from 20 to 50 metres in height (see, e.g., Fig. 4). The tallest transmission towers in the Manitoba Hydro system are more than 100 metres high to support the long crossing of the Nelson River (see Berger, R.P. (1995)). A broad range of target objects (e.g., insulators, pins, bolts, cross braces) would part of the repetoire of objects that ALiCE II would inspect. A sample insultator group is shown in Fig. 6 (notice that the top insulator in Fig. 6 is damaged, and needs to be replaced).

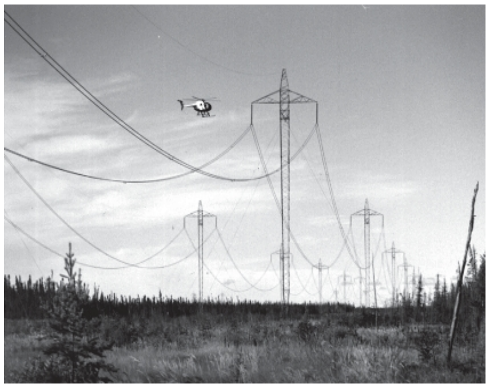


Fig. 3. Sample Steel Towers

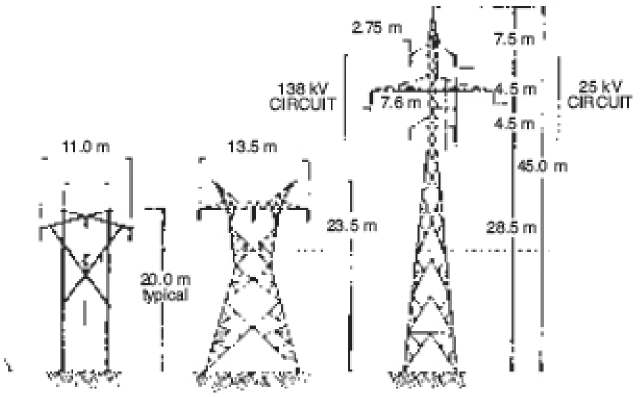


Fig. 4. Sample Tower Measurements

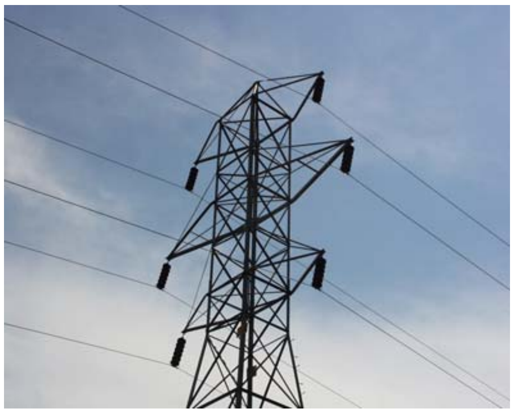


Fig. 5. Sample Steel Tower

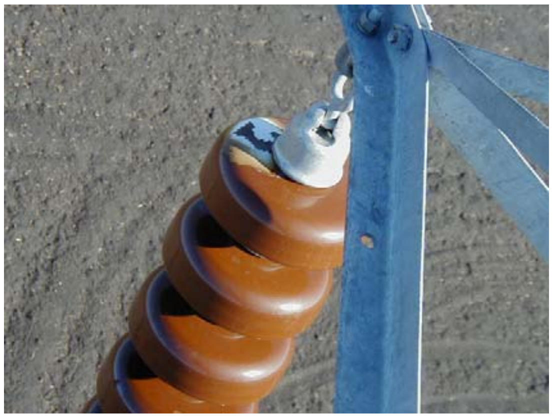
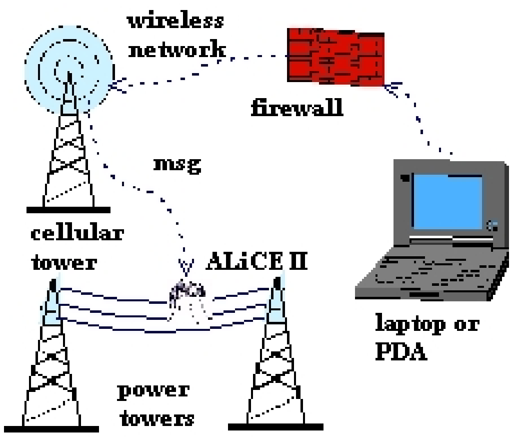


Fig. 6. Sample Insulator Group

Hence, once a bot identifies a target (e.g., insulator or tower cross--brace), target tracking is necessary to move a camera to compensate for wind action so that it continues to inspect a target object.

4 Commanding

Commanding of ALiCE II is made possible by a wireless network as shown in Fig. 2. Specifically, ALiCE II uses the Sierra AirCard 580 Wireless WAN Modem (see, e.g., Fig. 2) to establish the connection to the Point-To-Point Protocol (PPP) server over the Telus cellular network. PPP has three main components corresponding to RFC 1332 from the IETF (2006), a method for sending data over the connection between two nodes, a Link Control Protocol (LCP) for maintaining the data link, and a suite of Network Control Protocols (NCPs) used to establish and configure different network-layer protocols. Specifically, the NCP protocol used to establish the IP address is the PPP Internet Protocol Control Protocol (IPCP) which negotiates the IP address only after the data link has been established and tested by the IETF (2006). What follows is a comparison of several types of tracking systems using classical as well as reinforcement learning methods.



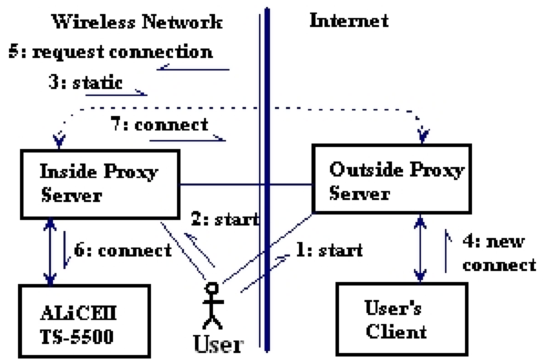


Fig. 7. ALiCE II Wireless network Fig. 8 UML Comm. Dialogue

Only the tilt servo is visible in Fig. 2. A bot suspended from a sky wire stretch between electric power transmission towers usually sways from side--to--side as it moves along a wire. Hence, once a bot identifies a target (e.g., insulator or tower cross--brace), target tracking is necessary to move a camera so that it continues to inspect a target object. What follows is a comparison of several types of tracking systems using classical as well as reinforcement learning methods.

5 Monocular Vision System Testbed

This section gives a brief overview of a testbed for the ALiCE II vision system (see Figs. 1, 2, and 9). The main processing unit of ALiCE II is the TS-5500 compact full-featured PC compatible Single Board Computer based on the AMD Elan520 processor from Elan (2006). The TS-5500 is running the Linux operating system (see Torvalds, L. (2006)) on the Elan 0x586 class processor from Technologic (2006) using the Elan microcontroller. The TS-5500 is configured to use an Orinoco wireless ethernet card which can be programmed as a wireless Access Point (AP), or to connect to an existing AP such as the DLink wireless router to send camera images to a base station. Furthermore, the TS-5500 is also configured to use a Sierra 555 wireless network card which can connect to the internet through a cellular CDMA 1X network (see Fig. 7). The main constraint limiting the possible algorithms which can be used for target tracking is the relatively low computational power of the CPU, which is run by a 133 MHz clock. As a result, the TS-5500 has computational power comparable to a Pentium III processor with an approximately 70 MHz clock. The TS-5500 is also equipped with a Creative NX Ultra WebCam that is mounted on two Hobbico mini servo's. The servo's provide the camera with two degrees of freedom (DOF), because it is mounted on two servos (one servo for swinging the camera in a horizontal plane (panning) and a second servo for swinging the camera up and down (tilting)) . Each servo's movement can be controlled separately. Two additional computers (IBM Thinkpads) are included in the hardware test setup (see Fig. 9). One IBM ThinkPad display's a randomly moving target. A second Thinkpad serves as an Observation PC, which is connected to a server running on the TS-5500. This second Thinkpad provides visual feedback and records data for comparison purposes.

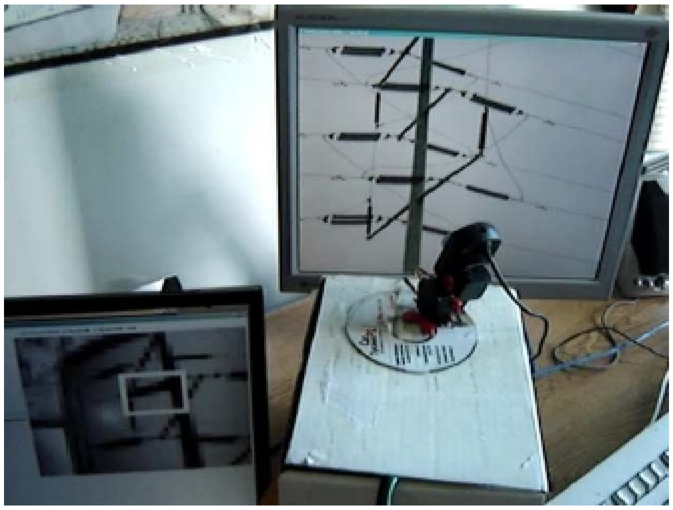


Fig. 9 Testbed for Monocular Vision System

6 Monocular Vision System

The goal of the monocular vision system is to point a camera at a moving target, continuously. To achieve this goal, a fast and efficient image processing technique is required to detect the position of a target in an input frame obtained from the webcam. The approach utilized in our system is called template matching. However, before template matching can occur, preprocessing of the input image from the camera is necessary due to the computational restraints of the TS-5500 (comparable to a Pentium III processor with a clock speed ≈ 70 MHz). Preprocessing consists of both spatially decimating the input image as well as transforming the RGB colours into grey levels. Spatial decimation is performed in each dimension (both *x* and *y*) by taking every n^th^ pixel. In our case, we selected every 4th pixel to transform the smallest available output of the webcam from 160 × 120 pixels to 40 × 30 pixels. Furthermore, conversion into grey levels is preformed using Eq. 2.

$$I = \frac{\left(R + G + B\right)}{3}. \tag{2}$$

The result of preprocessing is an image that is over 40 times smaller (160 × 120 × 3 = 57600 bytes compared with 40 × 30 × 1 = 1320 bytes) and still contains sufficient information to perform template matching. Template matching is implemented in this system using the Sum of Squared Differences (SSD), which is similar to the approach used by Gaskett (2002, 2005). The idea is to find a match of the template in the input image, where it is assumed that the template is the smaller of the two images. The SSD model is given in Eq. 3, and is calculated for every position of the template in the input image.

$$SSD(x,y) = \sum_j \sum_k \left[input\left(j,k\right) - template\left(j-k, k-y\right)\right]^2 . \tag{3}$$

The result is that Eq. 3 is minimal at the point in which the template occurs in the input image. Next, the coordinates of the minimum are passed on to the target tracking algorithms. Finally, it is important to note that the coordinates of the target refer to the centre of the template and will only span a subset of size 28 × 21 of the 40 × 30 image. For example, consider Fig. 10, which is a simple example showing in grey all the possible coordinates of the centre of a template of size 5 × 5 within an image of size 8 × 8.

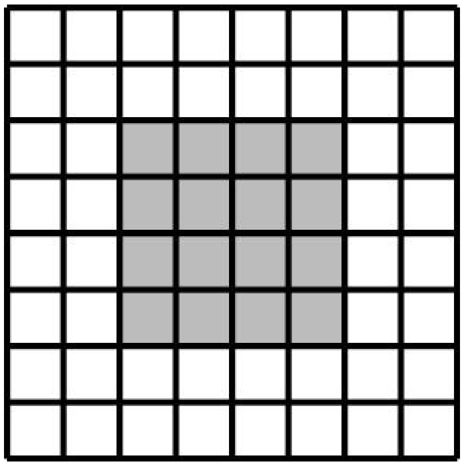


Fig. 10. Image Template

s20 -1 1	s21 -1 1	s22 0 1	s23 1 1	s24 1 1
s15 -1 1	s16 -1 1	s17 0 1	s18 1 1	s19 1 1
s10 -1 0	s11 -1 0	s12 0 0	s13 1 0	s14 1 0
s5 -1 -1	s6 -1 -1	s7 0 -1	s8 1 -1	s9 1 -1
s0 -1 -1	s1 -1 -1	s2 0 -1	s3 1 -1	s4 1 -1

Fig. 11 System States

6.1 Reinforcement Learning by Vision System

At the heart of reinforcement learning is the principle of state and state transitions. The learning agent determines the state from observed changes in its environment. Furthermore, for each perceived state there is one or more desirable actions that will cause a change in the environment and produce a transition into a beneficial state. Rewards can then be defined by the perceived desirability of the new state. States in the monocular vision system are based on the coordinates of the target obtained from the template matching process and are shown in Fig. 11. These states are applied to the 28 × 21 subset described above. The selection of states in a learning environment is more of an art than a science. The states given in Fig. 11 were arrived at through trial and error. The goal was to select areas small enough to allow a few ideal servo movements in each state, but not so small as to severely limit the set of possible movements. Similarly, the actions available in each state are based on the maximum distance from the centre of the 28 × 21 area to the outside edge in any single dimension which is approximately 14 pixels. Consequently, the actions available in each state range from 1 - 12 and represent increments to the servos position. Each step increment to the servos provides a rotation of 0.9 degrees with an accuracy of ±0.25%. Furthermore, the two numbers located below each state identifier in Fig. 11 represent the direction of the pan and tilt servo's respectively. Finally, reward is calculated using Eq. 4.

$$reward = 1 - \frac{distance}{maxdistance}, \tag{4}$$

where distance is calculated using Eq. 5.

$$distance = \sqrt{x^2 + y^2}\ . \tag{5}$$

Note that *x* and *y* represent the coordinates of the target, and the origin is selected as the centre of the camera's field of view, thus, there is no need to include the coordinates of the centre in the distance calculation. A result computed using Eq. 4 is a normalized reward which equals 1 when the target is located at the centre, and 0 when the target is at the outside edge of the field of view.

6.2 Classical Target Tracking

This section presents the classical tracking system implemented in the monocular vision system. The classical target tracking method is a deterministic algorithm and does not perform any learning. Consequently, this algorithm defines its state space separately than the reinforcement learning algorithms. In effect, every possible target coordinate can be considered a state and the only action available in each state is to move the servo's the calculated distance (using Eq. 5) to the target. In a sense, it selects the right action to take in every state. As a result, this algorithm is important because it demonstrates the desired behaviour of the reinforcement learning algorithms and provides a baseline for comparison when plotting the results. The classical target tracking method is provided in Alg. 1.

```
Algorithm 1: Classical Target Tracking
Input: States s ∈ S; //1 state for each possible set of target coordinates
Output: Deterministic policy π(s); //selects the same action in every state.
while (true) do
  get current state; //i.e., coordinates of target
  pan ← target's horizontal distance from center of camera view;
  tilt ← target's vertical distance from center of camera view;
  move servos by (pan, tilt);
end
```

6.3 Actor-Critic Learning Method

Actor-critic (AC) learning methods are temporal difference (TD) learning methods with a separate memory structure to represent policy independent of the value function used. The AC method considered in this section is an extension of reinforcement comparison in Sutton and Barto (1998). The estimated value function *V(s)* is an average of the rewards received while in state *s*. After each action selection, the critic evaluates the quality of the selected action using δ in Eq. 6, which represents the error (labeled the TD error) between successive estimates of the expected value of a state.

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{6}$$

where γ is the discount rate, and the value of a state implemented by the critic is given in Eq. 7.

$$V(s_t) = V(s_t) + \alpha_t \delta_t, \tag{7}$$

where α_t is the critic's learning rate at time *t*. For simplicity, the subscript *t* is omitted in what follows. If $\delta > 0$, then it can be said that the expected return received from taking action a_t is larger than the expected return in state s_t resulting in an increase to action preference *p(s, a)*. Conversely, if $\delta < 0$, the action a_t produced a return that is worse than expected and *p(s, a)* is decreased (see, e.g., Wawrzynski (2005)). The run-and-twiddle (RT) method introduced by Selfridge (1989) and elaborated by Watkins (1989) is a control strategy inspired by behaviour that has been observed in biological organisms such as E. Coli and silk moths, where an organism continues its current action until the strength of a signal obtained from the environment falls below an acceptable level, and then it ``twiddles'' (i.e., works out a new action strategy). This idea can be applied to the value of $\delta$ in Alg. 2. Whenever $\delta < 0$ occurs too often, it can be said that an agent is performing below expectations, and that a ``twiddle'' is necessary to improve the current situation. The preferred action *a* in state *s* is calculated using Eq. 8.

$$p(s,a) = p(s,a) + \beta\delta, \tag{8}$$

where β is the actor's learning rate. The preferred action *p(s, a)* is employed by an actor to choose actions stochastically using the Gibbs softmax method as shown in Eq. 9.

$$\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}. \tag{9}$$

Alg. 2 gives the Actor-Critic method that is an extension of the reinforcement comparison method given in Sutton and Barto (1998). It is assumed that the behaviour represented by Alg. 2 is episodic, and is executed continuously.

```
**Algorithm 2**: Actor-Critic Method
**Input**: States $s \in S$, Actions $a \in A$, initialized $\alpha$, $\gamma$, $\beta$.
**Output**: Policy $\pi$(s, a) //controls selection of action $a$ in state s.
**for** (all $s \in S$, $a \in A$) **do**
  p(s, a) $\leftarrow$ 0;
```

```
      $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
end
while (true) do
  initialize s;
  for (t = 0; t < $T_m$; t = t + 1) do
    choose a from s using π(s, a);
    take action a, observe r, s';  //s' = $s_{t+1}$
    $\delta = r + \gamma V(s') - V(s)$;
    $V(s) = V(s) + \alpha\delta$;
    $p(s,a) = p(s,a) + \beta\delta$;
    $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
    s ← s';
    end
end
```

7 Actor-Critic Method using Rough Coverage

This section presents a modified actor critic method using rough coverage derived within the context of an approximation space, which is constructed relative to a decision table known as an ethogram. An ethogram is a decision table where each object in the table represents an observed behavior, which has features such as state, action and proximate cause inspired by Tinbergen (1963). This is explained in detail in Peters, Henry and Ramanna (2005a), and not repeated, here.

7.1 Average Rough Coverage

This section illustrates how to derive average rough coverage using an ethogram. During a swarm episode, an ethogram is constructed, which provides the basis for an approximation space and the derivation of the degree that a block of equivalent behaviours is covered by a set of behaviours representing a standard (see, e.g., Peters (2005b), Tinbergen (1963), Peters, Henry and Ramanna (2005a). Let *xi*, *s*, *PC*, *a*, *p*(*s*, *a*), *r*, *d* denote i^th observed behaviour, current state, proximate cause (Tinberben (1963)), possible action in current state, action-preference, reward for an action in previous state, and decision (1 = choose action, 0 = reject action), respectively.

Assume, for example, $B_a(x) = \{y \in U_{beh} \mid x\mathrm{IND}(B \cup \{a\})y\}$. Let $\mathrm{B} = \{ B_a(x) \mid x \in \Omega\}$ denote a set of blocks representing actions in a set of sample behaviours Ω. Let $\overline{r}$ denote average rough coverage as shown in Eq. 10.

$$\overline{r} = \frac{1}{card(\mathrm{B})} \sum_{i=1}^{card(\mathrm{B})} \nu_{\mathrm{B}}\left(B_a(x), B_*D\right), \qquad (10)$$

where $B_a(x) \in B$. Computing the average lower rough coverage value for action blocks extracted from an ethogram implicitly measures the extent that past actions have been rewarded. What follows is a simple example of how to set up a lower approximation space relative to an ethogram. The calculations are performed on the feature values shown in Table 1.

Table 1: Sample Ethogram

x_i	s	PC	a	$p(s,a)$	r	d
$x0$	1	3	4	0.010	0.010	0
$x1$	1	3	5	0.010	0.010	1
$x2$	1	3	4	0.010	0.010	0
$x3$	1	3	5	0.020	0.011	1
$x4$	1	3	4	0.010	0.010	0
$x5$	1	3	5	0.031	0.012	1
$x6$	1	3	4	0.010	0.010	0
$x7$	1	3	5	0.043	0.013	1
$x8$	1	3	4	0.010	0.010	1
$x9$	1	3	5	0.056	0.014	0

$$B = \{s_i, PC_i, a_i, p(s,a)_i, r_i\},$$
$$D = \{x \in U \mid d(x) = 1\} = \{x1, x3, x5, x7, x8\}$$
$$B_a(x) = \{y \in U_{beh} \mid xIND(B \cup \{a\})y\},\ hence$$
$$B_{a=4}(x0) = \{x0, x2, x4, x6, x8\}, B_{a=5}(x1) = \{x1\},$$
$$B_{a=5}(x3) = \{x3\}, B_{a=5}(x5) = \{x5\}, B_{a=5}(x7) = \{x7\}, B_{a=5}(x9) = \{x9\},$$
$$B_*D = \cup\{B_a(x) \mid B_a(x) \subseteq D\} = \{x1, x3, x5, x7\}$$
$$\nu_B(B_{a=4}(x0), B_*D) = 0, \nu_B(B_{a=5}(x1), B_*D) = 0.25,$$
$$\nu_B(B_{a=5}(x3), B_*D) = 0.25, \nu_B(B_{a=5}(x5), B_*D) = 0.25,$$
$$\nu_B(B_{a=5}(x7), B_*D) = 0.25, \nu_B(B_{a=5}(x9), B_*D) = 0,$$
$$\overline{r} = \overline{\nu_B(B_a(x), B_*D)} = 0.1667$$

7.2 Rough Coverage Actor Critic Method

The rough coverage actor critic (RCAC) method is one among many forms of the actor critic method (see, *e.g.*, Barto (1983), Berenji (2003), Bertsekas and Tsitsiklis (1996), Konda (1995), Peters and Henry (2005), Sutton and Barto (1998), Watkins and Dayan (1992), Wawrzyński and Pacut (2004), Wawrzyński (2005)). Common variations include additional factors which vary the amount of credit assigned to selected actions. This is most commonly seen in calculating preference, $p(s, a)$. The rough coverage form of the Actor-Critic method calculates preference values as shown in Eq. 11.

$$p(s,a) \leftarrow p(s,a) + \beta[\delta - \bar{r}], \tag{10}$$

where $\bar{r}$ denotes average rough coverage computed using Eq. 10. This is reminiscent of the idea of a *reference reward* used during reinforcement comparison. Recall that incremental reinforcement comparison uses an incremental average of all recently received rewards as suggested by Sutton and Barto (1998). Intuitively, this means action probabilities are now governed by the coverage of an action by a set of equivalent actions which represent a standard. Rough coverage values are defined within a lower approximation space. Alg. 3 is the RCAC learning algorithm used in the monocular vision system. Notice that the only difference between Algorithms 2 and 3 is the addition of the reference reward $\bar{r}$, which is calculated using rough coverage.

```
**Algorithm 3**: Rough Coverage Actor-Critic Method
**Input**: States $s \in S$, Actions $a \in A$, initialized $\alpha$, $\gamma$, $\beta$.
**Output**: Policy $\pi(s, a)$ //controls selection of action $a$ in state s.
**for** (all $s \in S,\ a \in A$) **do**
  p(s, a) ← 0;
  $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
**end**
**while** (true) **do**
  initialize s;
  **for** (t = 0; t < $T_m$; t = t + 1) **do**
    choose a from s using $\pi(s, a)$;
    take action a, observe r, s'; //s' = $s_{t+1}$
    $\delta = r + \gamma V(s') - V(s)$;
    $V(s) = V(s) + \alpha\delta$;
    $p(s,a) = p(s,a) + \beta[\delta - \bar{r}]$;
    $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
    s ← s';
    **end**
    Extract ethogram table $IS_{swarm} = (U_{beh}, A, d)$;
    Discretize feature values in $IS_{swarm}$;
    Compute $\bar{r}$ as in Eq. 10 using $IS_{swarm}$;
**end**
```

7.3 Run-and-Twiddle Actor Critic Method

This section briefly presents a run-and-twiddle (RT) form of AC method in Alg. 4. Both AC methods use preference values to compute action selection probabilities. However, the RT AC method uses $\bar{v}_a$ (average rough coverage of all the blocks containing action *a*) to control the rate of learning instead of β in the actor. A ``twiddle" entails advancing the window of the ethogram (recorded behavior patterns of ecosystem organisms, or, in the monocular vision system, behavior patterns of a moving camera) and recalibrating $\bar{v}_a \forall a \in A$. This form of twiddling mimics the behaviour of E. Coli bacteria (diminishing food supply results in change in movement) or a male silk moth following the perfume emitted by a female silk moth (diminishing perfume signal results in a change of the search path), which is described by Selfridge (1984). This idea can be applied to the value of δ in Alg. 3. When $\delta < 0$ occurs too often, then it can be said that the agent is performing below expectations, and that a ``twiddle" is necessary to improve the current situation. This is the basic approach underlying Alg. 4.

```
**Algorithm 3**: Rough Coverage Actor-Critic Method
**Input**: States $s \in S$, Actions $a \in A$, initialized $\alpha$, $\gamma$, $\beta$, $th$. //$th$ = threshold
**Output**: Policy $\pi(s, a)$ //controls selection of action $a$ in state s.
**for** (all $s \in S$, $a \in A$) **do**
  $p(s, a) \leftarrow 0$;
  $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
**end**
**while** (true) **do**
  initialize s; $\nu \leftarrow 0$;
  **for** ($t = 0$; $t < T_m$; $t = t + 1$) **do**
    choose a from s using $\pi(s, a)$;
    take action a, observe r, s'; //$s' = s_{t+1}$
    $\delta = r + \gamma V(s') - V(s)$;
    $V(s) = V(s) + \alpha\delta$;
    $p(s,a) = p(s,a) + \beta[\delta - \bar{r}]$;
    $\pi(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
    **if** ($\delta < 0$) **then**
      $\nu \leftarrow \nu + 1$;
      **if** ($\nu < th$) **then**
         Extract ethogram table $IS_{swarm} = (U_{beh}, A, d)$;
         Discretize feature values in $IS_{swarm}$;
         Compute $\bar{v}_a \forall a \in A$;
         $\nu \leftarrow 0$;
      **end**
    **end**
    $s \leftarrow s'$;
  **end**

**end**
```

8 Results

The values shown in the plots in Figs. 12(a)-12(d) represent the RMS error between the distance of the target from the centre of the field-of-view of each camera image and a running average of the distance values. Preliminary experiments suggest that the RC AC method has the poorest performance among (see sample RMS plots in Figs. 12(a), 12(b), 12(c)). This is due largely to the fact that the RC AC method must construct a lower approximation space at the end of each episode (in contrast to the run and twiddle method which only constructs a lower approximation space whenever things appear to be going badly). As a result, the RC AC algorithm hangs due to the limited processing power of the TS-5500 and, as a result, the target moves a significant distance from the centre of the image. However, note that the RC AC method performs quite similar to the AC and RT AC when observing the normalized totals of the state value function V (see Fig. 13). This suggests that the RC AC method is comparable to the other two AC methods in terms of converging to an optimal policy. The RC AC method just does poorly in this real-time learning environment because its ``calculate hang'' causes the algorithm to lose the target. This can be corrected by implementing the RC AC method on a system with more processing power.

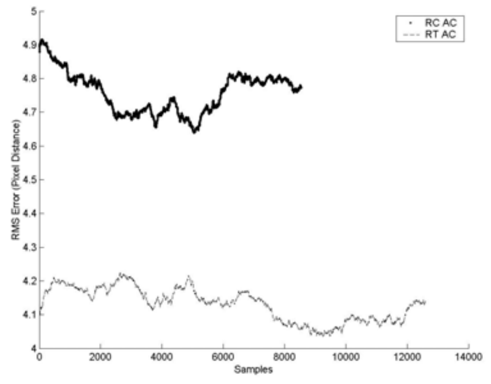


Fig. 12(a) RC AC vs. RT AC

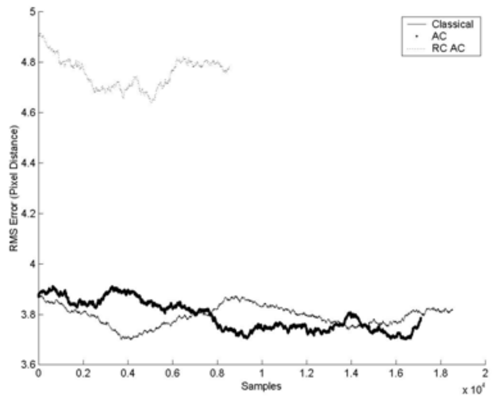
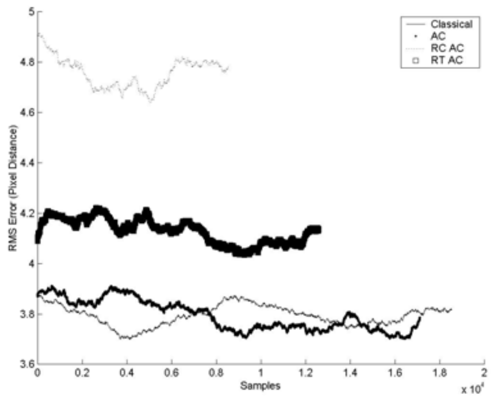


Fig. 12(b) AC, RC AC, classical



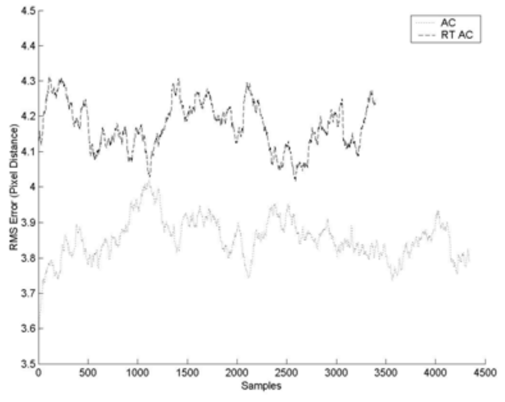


Fig. 12(c) RMS for 4 AC methods Fig. 12(d) AC vs. RT AC

In preliminary tests with camera control in the monocular vision system, the performance of run-and-twiddle (RT AC) method compares favorably the actor critic (AC) method. Evidence of this can be seen in the plots in Fig. 12(a) and Fig. 12(d).

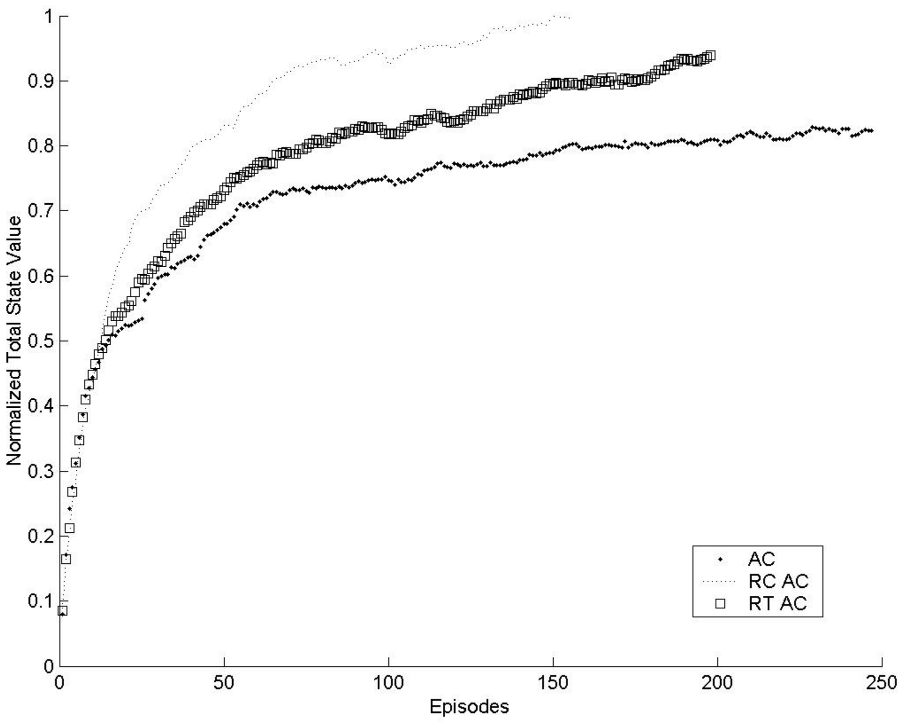


Fig. 13 Comparison of Normalized Total State Values

Finally, it is important to note that there are two reasons why the classical tracking method outperforms the reinforcement learning algorithms in the sample RMS plots in Figs. 12(b), 12(c). At the beginning of the tests, the RL algorithms are performing more exploration than greedy selection to determine the best possible action to select in each state. Toward the end of the tests (in later episodes), the RL algorithms consistently converge to an optimum policy. However, even during later episodes, the AC methods still perform exploration some of the time. As a result, the classical method should out-perform the RL methods every time. The RL algorithms are ideally suited to deal with noise, which has not be considered in the form of SSD given in Eq. 3. In this context, the term *noise* denotes irregular fluctuations in either movements of a target or in the environment that influences the quality of camera images. Noise has a number of sources (e.g., camera vibration due to buffeting of the wind, electromagnetic field, and weather conditions such has rain, glare from the sun). This is very important because of the noisy environment in which the monocular vision system must operate. In the presence of noise, the classical algorithm will not do as well as RL because is has no provision for exploration whenever it is not clear what action should be performed next to find a moving target.

9 Learning Experiments in Noisy Environment

To induce noise, the platform that the camera was sitting on (see Fig. 9) was purposely placed on a slope rather than a flat surface. The result of this placement caused the entire platform to vibrate when the camera servos were activated. This provided a degree of realism analogous to what would be experienced in a suspended system where any type of motion onboard ALiCE II will disturb the system somewhat. The extra motion was intended to provide a more challenging movement of the target as it varied with the step sizes, greater motion resulted from larger step sizes. The resulting motion of the platform was a rocking action from back to front only.

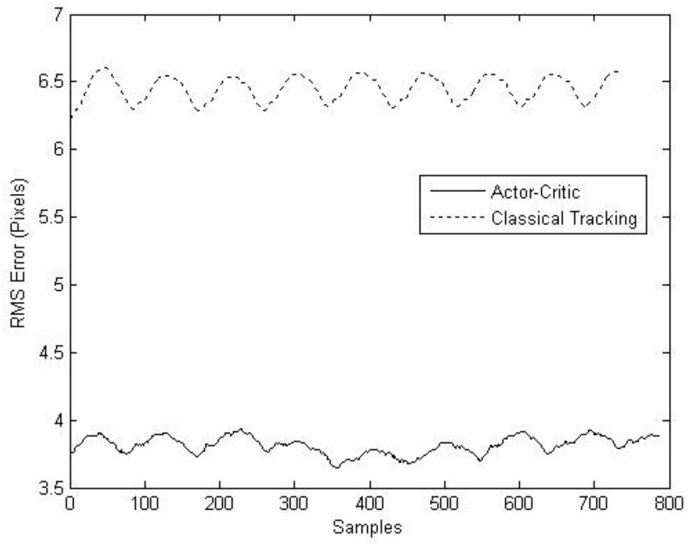


Fig. 14 Actor Critic in Noisy Environment

The illumination for the experiments was kept as uniform as possible with the addition of papers surrounding the monitor in an attempt to restrict the available ambient light. The experiments were all conducted for 5 minutes each, providing a reasonable time period to get an idea of their individual performance. Overall, this resulted in a controlled environment with some extra fluctuations in movement and a slightly more difficult target tracking problem than the previous set of experiments.

One thing that can readily be seen in Figs. 14-16 is that there is a periodicity in the experimental procedure. This is due to the trajectory that the target takes. The peaks in the cycle correspond to the highest error and this coincides to part of the trajectory where the target travels down the right hand side where the field of view can only see part of the target and often the camera would lose the target depending on the method in question. Note that the performance of the actor critic method (both the convention AC with sample RMS values in Fig. 14 and rough coverage AC with RMS values in Figs. 15-16) is excellent and it has managed to out-perform the classical tracking method.

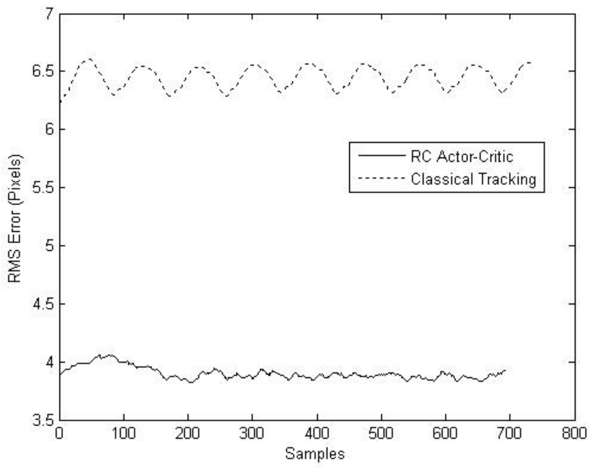


Fig. 15 Rough Coverage Actor Critic in Noisy Environment

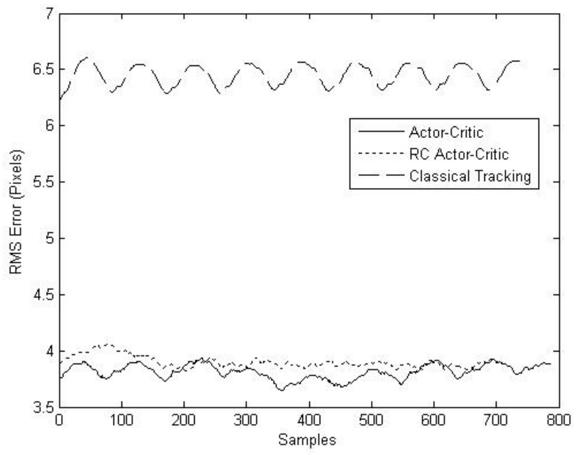


Fig. 16 RC AC, AC, Classical Tracking in Noisy Environment

The classical tracking system is being out performed by the AC and RC AC learning methods. The likely explanation for the worsened performance of the classical tracking algorithm is that the extra motion from the camera platform and a tendency for the target to slip down so that the field of view of the camera includes only part or none of the target. With these new problems, the AC and RC AC learning methods led to improved performance, since these reinforcement learning methods were able to explore in a somewhat noisy environment to discover a better tracking profile. Notice that the rough coverage form of actor critic method tends to have slightly better performance than the conventional AC method (see, for example, Fig. 16).

10 Conclusion

This article introduce a monocular vision system that learns with the help of approximations. This results in a vision system with actions that are influenced by patterns of behavior discovered in the context of approximation spaces that have been constructed periodically from ethograms. In a noisy environment, the rough coverage form of actor critic method tends to do better than the classical tracking method. This is in keeping the basic approach introduced in 1981 by Zdzisław Pawlak, who suggested classifying objects by means of their features. In the context of target tracking by a monocular vision system, an object an observed behavior of the system. Classification of vision system behavior is made possible by comparing feature values of observed system behaviors in different states, which have been captured in ethogram tables. The basic approach in the design of the vision system described in this chapter is derived from ethology introduced by Tinbergen in 1963. The current work on vision systems is part of a swarm intelligence approach, where pairs of cameras on separate, cooperating bots synchronize to form binocular vision systems that learn. Future work will consider n-ocular vision systems that learn with approximation spaces as well as various reinforcement learning methods not considered, here.

Acknowledgements

Many thanks to Wes Mueller, who has been a great help in obtaining images showing hydro transmission equipment. This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC) grant 185986 and grant T247 from Manitoba Hydro.

References

Barto, A.G., Sutton, R.S., Anderson, C.W. (1983). Neuronlike elements that can solve difficult problems. IEEE Trans. on Systems, Man, and Cybernetics 13, 834-846.

Berenji, H.R. (2003). A convergent actor-critic-based FRL algorithm with application to power management of wireless transmitters. IEEE Trans. on Fuzzy Systems 11/4, 478-485.

Bertsekas, D.P., Tsitsiklis, J.N. (1996). Neuro-Dynamic Programming. Athena Scientific, Belmont, MA.

Berger, R.P. (1995). Fur, Feathers & Transmission Lines:
http://www.hydro.mb.ca/environment/publications/fur_feathers.pdf

Elan (2006). Elan SC520 Microcontroller User's Manual at http://www.embeddedarm.com/

Gaskett, C. (2002). Q-Learning for Robot Control. Ph.D. Thesis, Supervisor: A. Zelinsky, Department of Systems Engineering, The Australian National University.

Gaskett, C., Ude, A., Cheng, G. (2005). Hand-Eye Coordination through Endpoint Closed-Loop and Learned Endpoint Open-Loop Visual Servo Control. In Proc. Int. J. of Humanoid Robotics, 2 (2), 203-224.

Internet Engineering Task Force (2006). RFC 1332 The PPP Internet Protocol Control Protocol (IPCP) at http://www.ietf.org/rfc/rfc1332.txt

Konda, V.R., Tsitsiklis, J.N. (1995). Actor--critic algorithms, Adv. Neural Inform. Processing Sys., 345--352.

Pawlak, Z. (1981a). Classification of Objects by Means of Attributes. Institute for Computer Science, Polish Academy of Sciences Report 429.

Pawlak, Z. (1981b). Rough Sets. Institute for Computer Science, Polish Academy of Sciences Report 431.

Pawlak, Z. (1982). Rough sets, International J. Comp. Inform. Science, 11, 341—356.

Peters, J.F., Pawlak, Z. (2007). Zdzisław Pawlak life and work (1906-2006), Information Sciences 177, 1-2.

Pawlak, Z., Skowron, A. (2007a). Rudiments of rough sets, Sciences 177, 3-27.

Pawlak, Z., Skowron, A. (2007b). Rough sets: Some extensions, Sciences 177, 28-40.

Pawlak, Z., Skowron, A. (2007c). Rough sets and Boolean reasoning, Sciences 177, 41-73.

Peters, J.F., Henry, C., Ramanna, S. (2005a). Rough Ethograms : Study of Intelligent System Behavior. In: M.A. Kłopotek, S. Wierzchoń, K. Trojanowski (Eds.), New Trends in Intelligent Information Processing and Web Mining (IIS05), Gdańsk, Poland, 117-126.

Peters, J.F. (2005b). Rough ethology: Towards a Biologically-Inspired Study of Collective Behavior in Intelligent Systems with Approximation Spaces. Transactions on Rough Sets, III, LNCS 3400, 153-174.

Peters, J.F., Henry, C. (2005). Reinforcement learning in swarms that learn. Proc. IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology (IAT 2005), Compiègne Univ. of Tech., France, 400-406.

Peters, J.F., Henry, C. (2006). Reinforcement learning with approximation spaces. Fundamenta Informaticae 71 (2-3) , 323-349.

Peters, J.F., Skowron, A., Synak, P., Ramanna, S. (2003). Rough sets and information granulation. In: Bilgic, T., Baets, D., Kaynak, O. (Eds.), Tenth Int. Fuzzy Systems Assoc. World Congress IFSA, Instanbul, Turkey, Lecture Notes in Artificial Intelligence 2715, Physica-Verlag, Heidelberg, 370--377.

Peters, J.F., Skowron, A. (2006). Zdzislaw Pawlak: Life and Work, Transactions on Rough Sets V, 1-24.

Polkowski, L. (2002). Rough Sets. Mathematical Foundations. Springer-Verlag,Heidelberg.

Selfridge, O.G. (1984). Some themes and primitives in ill-defined systems. In: Selfridge, O.G., Rissland, E.L., Arbib, M.A. (Eds.): Adaptive Control of Ill-Defined Systems. Plenum Press, London.

Skowron, A., Stepaniuk, J. (1995). Generalized approximation spaces. In: Lin, T.Y.,Wildberger, A.M. (Eds.), Soft Computing, Simulation Councils, San Diego, 18-21.

Skowron, A., Swiniarski, R., Synak, P. (2005). Approximation spaces and information granulation, Transactions on Rough Sets III, 175-189.

Stepaniuk, J. (1998). Approximation spaces, reducts and representatives. In: Polkowski, L., Skowron, A.: (Eds.), Rough Sets in Knowledge Discovery 2, Studies in Fuzziness and Soft Computing 19. Springer-Verlag,Heidelberg, 109-126.

Sutton, R.S., Barto, A.G. (1998). Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA.

Tinbergen, N. (1963). On aims and methods of ethology, Zeitschrift fűr Tierpsychologie 20, 410--433.

Torvalds, L. (2006). Linux Operating System at http://www.linux.org/

Technologic (2006). TSUM User's Manual at http://www.embeddedarm.com/

Watkins, C.J.C.H. (1989). Learning from Delayed Rewards, Ph.D. Thesis, supervisor: Richard Young, King's College, University of Cambridge, UK.

Watkins, C.J.C.H., Dayan, P. (1992). Technical note: Q-learning, Machine Learning, 8, 279-292.

Wawrzyński. P., Pacut, A. (2004). Intensive versus nonintensive actor-critic algorithms of reinforcement learning. Proc. 7th Int. Conf. on Artificial Intelligence and Soft Computing, Springer 3070, 934-941.

Wawrzyński. P. (2005). Intensive Reinforcement Learning, Ph.D. dissertation, supervisor: Andrzej Pacut, Institute of Control and Computational Engineering, Warsaw University of Technology.