Robotica (1999) volume 17, pp. 325–334. Printed in the United Kingdom © 1999 Cambridge University Press

Becoming incrementally reactive: on-line learning of an evolving decision tree array for robot navigation

G.H. Shah Hamzei*, D.J. Mulvaney**, and I. Sillitoe†

(Received in Final Form: October 1, 1998)

SUMMARY

This paper proposes a novel hierarchical multi-layer decision tree for representing reactive robot navigation knowledge. In this representation, the perception space is decomposed into a hierarchical set of worlds reflecting environments which are homogeneous in nature and which vary in complexity in an ordered manner. Each world is used to produce a corresponding decision tree which is trained incrementally. The instantaneous perception of the robot is used to select an appropriate rule from the decision tree and a sequence of rule activations form the complete trajectory. The ability to keep the knowledge complexity manageable and under control is an important aspect of the technique.

KEYWORDS: Incremental learning; Hierarchical learning; Decision trees; Machine learning; Intelligent navigation; Robotics; Incremental tree induction (ITI).

1. INTRODUCTION

Robotics offers a large supply of challenging problems suitable for investigation using learning techniques. The vast majority of recently-developed techniques can be subsumed within the categories of fuzzy logic,[1–4] connectionist approaches,[5,6] symbolic methods[7–9] and genetic algorithms.[10,11] In addition, hybrid versions have been developed which combine two or more of these techniques with the aim of improving or optimising certain control criteria as a result of complementary effects.[12,13]

Reactive control has shown to be an appropriate paradigm for autonomous robots,[14,15] as this allows the robot to respond to its immediate perception while exploring a dynamically changing environment without the need for mapping. Reactivity has variously been described as: learning behaviour which is established by some *a priori* knowledge of the environment,[1,16] pre-formulation of perception-actions into explicit rules,[3] or behaviours which are learned entirely from scratch.[17,18]

Decision trees[19–21] provide a symbolic approach to learning and have been successfully applied to a wide range of scientific and engineering problems to implement intelligent decision-making solutions.[12,22,23] Their recent application to scientific and medical problems,[22,24] as well as to industrial systems,[12,23] demonstrate their establishing popularity as reliable predictors. More relevant to the current work is their application to robotics and intelligent control.[9,25,26] In reference [9], features from the echoes of an array of ultrasonic sensors mounted on the circumference of a robot are collected and used to train a decision tree. This is then used to classify the contours of the obstacles in the robot surroundings enabling the robot to perceive and identify corners, edges and poles, and so aid navigation. Shah-Hamzei et al employ decision trees in references [25] and [26] and train these in an off-line manner to allow a robot to learn new behaviours. The generated trees are searched and used to synthesise appropriate control rules to navigate a robot in unseen environments.

This paper builds on recent advances in reactive control by proposing a novel approach to behaviour learning in complex and unknown robot environments. In addition, the work introduces the concept of incrementally evolving multiple decision trees in order to manage knowledge complexity. The learning mode is on-line and the robot environment is broken down, in terms of object configurations, into a number of worlds which are organised in a hierarchy based on the complexity of the perception. Since this approach has a biological motivation as its foundation, namely that of survival, at the outset the robot environment needs to be sufficiently simple to enable the robot to learn *efficient* and *simple* behaviours only in order to *survive*. These behaviours could be, for instance, how to reach a food source, a light source or other desired destination (target-seeking behaviour). This is mimicked by first training the robot in the simplest environment of the hierarchy in which there are no obstacles. To reach the set target, the robot typically performs a sequence of random motions whose quality is assessed under continuous evaluation. Movements which are evaluated as "useful" are incorporated into the training of an incrementally evolving decision tree. A tree so-modified is searched in order to provide a suitable movement for the robot. Once the basic survival mechanism has been established, the environment can be transformed to one more "hostile" by introducing obstacles to populate the environment. The robot can then be allowed to learn suitable new reactive behaviours in the more complex environment by shaping the previously learned knowledge.

The fact that the proposed algorithm is not tuned to a certain environment suggests some similarities with the

* Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, Leics LE11 3TU (UK)
E-mail: G.H.Shah-Hamzei@lboro.ac.uk
** Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, Leics LE11 3TU (UK)
E-mail: D.J.Mulvaney@lboro.ac.uk
† Department of Technology, University of Borås, 501 15 Borås (Sweden)
E-mail: Ian@adm.ing.hb.se

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

work of Braunstingl et al, who describe a wall-following fuzzy controller.[27] Both systems perform robot guidance in a reactive manner by using a stored representation of the robot's immediate perceptions: walls of arbitrary shape and obstacles are not modelled and path planning is basically of a local nature. However, the fundamental differences between the two approaches are that in the work presented in this paper the combination of co-operating locally intelligent path planners provides global navigation, and that these planners are able to learn and cope with an additional behaviour, namely that of target seeking.

2. THE RATIONALE OF HIERARCHICAL LEARNING DESIGN

A robust navigation system relies heavily on safe and simple motions. Since safe and simple motions are local in nature,[16] they can be provided in the current work by an array of locally-intelligent simple decision rules encoded in a finite number of decision trees. Global motion planning is accomplished by shaping local intelligence into global behaviours by cyclic and consecutive sampling of decision tree array elements.

Sammut et al[28] developed a distributed approach incorporating a number of decision trees representing the dynamic sub-systems of a Cessna aeroplane in an attempt to produce an artificial autopilot. Perhaps surprisingly, the decision trees trained using data supplied by less experienced pilots were able to perform better than those trees trained using data from experienced pilots.[29] This may have been due to the less experienced pilots being able to supply the boundaries of successful piloting. The constructed autopilot successfully managed to fly an entire flight mission.

3. NOTATION

In the current work, *world* is defined to be the instantaneous robot perception. The experiments are conducted using the simulated mobile robot Khepera (see section 10 for more details). The input perception P is defined to be that provided by the set of six circumference sensors as $P = \{S_0, S_1, S_2, S_3, S_4, S_5,\}$ in which the range of each sensor value is defined by $s_i = \{0, 1, 2, 3, \ldots, 1023\}$. Any perceived world is an element of $W = \{w_0, w_1, w_2, w_3, w_4\}$, where w_0 is the simplest and w_4 the most complex world. w_4 is not represented as an independent rule layer in the hierarchy as the action space in w_4 is limited to a single class regardless of the goal location, making the generation of a corresponding tree unnecessary. Training vectors need to have the format $f_0, f_1, \ldots, f_i, \ldots, f_n, c_i$, in which f_i is a feature with $f_i \in F$ and $F = \{f_0, f_1, \ldots, f_i, \ldots, f_n\}$ is the feature vector. Features can take either continuous or descrete values. c_i is a class with $c_i \in C$ from the set of available classes $C = \{c_0, c_1, \ldots, c_i, \ldots, c_n\}$ in which each class has a finite number of discrete values.

4. CONTROL SYSTEM ARCHITECTURE

Figure 1 shows the schematic view of the system architecture. This is composed of three main modules:

(i) *Environment*. The interface providing data from the physical world.

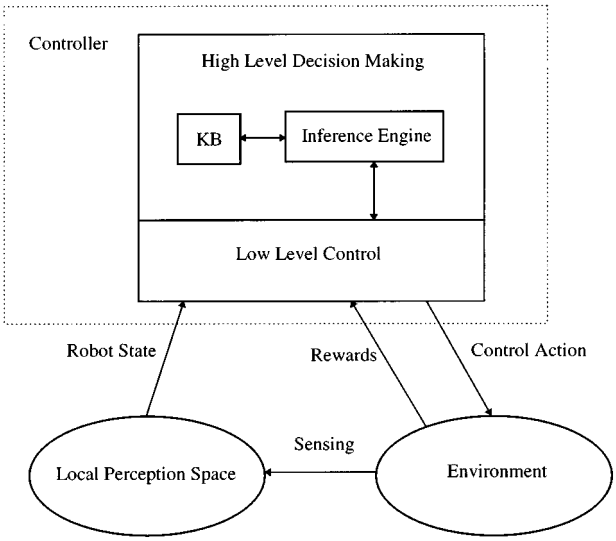


Fig. 1. The overall system architecture.

(ii) *Local Perception Space*. Each perception is mapped on a certain state that falls into a unique world category. The output is a state vector containing a finite number of state variables, and this is used to infer or re-infer tree networks.

(iii) *Controller*. This accommodates two sub-modules, namely high-level decision making (whose architecture is shown in Figure 2), and low-level control. The former performs predictions based on the current state vector, whereas the latter generates an appropriate control action based on the current decision in order to provide a movement demand to the robot. The control action $a(\beta, \nu)$ is a state variable defined in terms of turning angle β and a turning velocity ν,

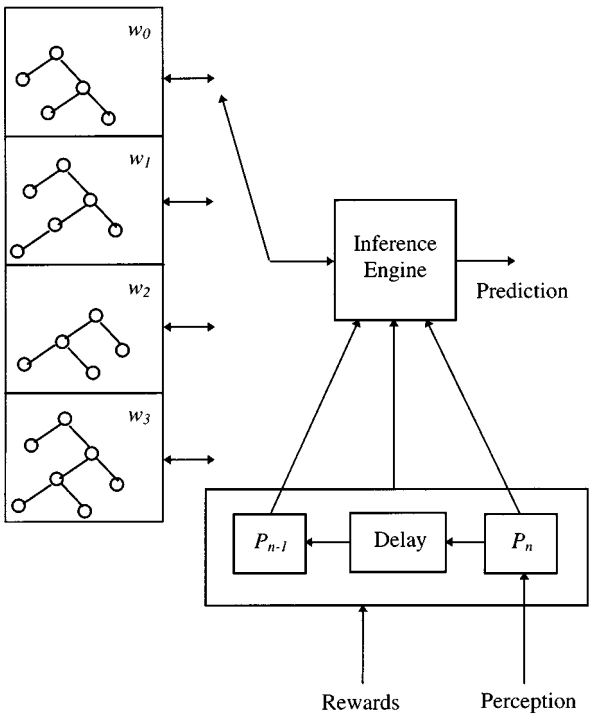


Fig. 2. High level decision module with parallel co-existing decision trees as intelligent local planners.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

As shown in Figure 2, the current state vector P_n is used to predict the robot motion, whereas the previous state, P_{n-1} (if positively rewarded) is used to restructure the tree to incorporate the new item of knowledge.

5. INCREMENTAL TREE EVOLUTION

The algorithm houses in its core the Incremental Tree Inducer (ITI-2.8)[30] as the learning module. ITI is driven in its incremental mode and on-line. This means that appropriate knowledge entities are given to ITI either for incorporation into an appropriate existing decision tree or to instantiate a new tree if none exists already.

5.1 Feature selection

Concept learning requires that the knowledge to induce decision trees is "sculptured" into a finite number of pre-defined vector entities. This is independent of the node (either batch or incremental) in which the tree induction is performed.

To produce trees with high discriminatory powers, the current work uses as features both the proximity sensor values and the relative location of the target to the robot. To restrict the dimensionality of the world-dependent trees, the format of the training examples is configured to be a function of the world complexity. This means that in world w_0 with no obstacles, each training example is reduced to the relative location of the goal (GoalRelLoc) and a single class, allowing the production of a linear decision tree which is highly goal oriented. In higher order worlds, proximity sensor values are added to the existing features resulting in the modified format of feature vector, S_0, S_1, S_2, S_3, S_4, S_5 *GoalRelLoc*, c_i. In the above, *GoalRelLoc* $\equiv \{N, NE, E, SE, SW, W, NW\}$ and $C \equiv \{F, FR, R, BR, BL, L, FL\}$ where F is "front", FR is "front right", BL is "back left", and so on.

5.2 Automatic knowledge acquisition and class prediction mechanism

One aspect of the approach that facilitates the population of the parallel co-existing trees in incremental mode, is the automatic generation of training examples. This is carried out to collect classification knowledge without the intervention of human experts. Since the robot motion directions are generated randomly at the training stage, we define the usefulness $U = f(C)$ as a measure of goodness of the performed actions. Following the transition from the previous state $n-1$ to the current state n, the cost C associated with each motion is calculated according to the heuristic function

$$C = \left(\frac{E_{n-1} - E_n}{d} \right) \mathrm{sgn}(\cos(\theta_n - \theta_{n-1})) \qquad (1)$$

and the usefulness is define as

$$U = \left\{ \begin{array}{l} 1 \text{ if } C \geq 0 \\ 0 \text{ if } C < 0 \end{array} \right\} \qquad (2)$$

In Equation (1), d is the total travelled distance, E is the Euclidean distance from the robot to the goal, and θ is the robot divergence angle whose details and derivation can be found in reference [18]. Only those states delivering a usefulness of unity are positively reinforced by being remembered, the remainder are forgotten. Each individual remembered experience is supplied to ITI-2.8 as part of the training data for the appropriate tree.

5.3 Local independence and global coupling of decision trees

Figure 2 illustrates that each tree network is an individual computational entity in itself and is able to make predictions when activated. In environments of greater complexity, with arbitrarily shaped walls and corners, a range of different trees will generally be sampled in order to generate the appropriate control actions, a suitable sequence of which forms the global path.

By analysing the process of tree generation in a finite window of time, one can observe how each individual tree evolves from a single-class entity to a mature decision tree capable of predictions, hence the notion of "tree evolution". Table 1 provides an illustration of tree evolution, showing that the effect of behaviour learning is to improve the robot navigation performance incrementally over a finite number of iterations.

6. ALGORITHMS

The complete system incorporates the two algorithms described below:

(i) *Behaviour learning*. This algorithm precedes navigation in the executional context as it is able to set the robot in exploratory mode in order to acquire knowledge. It also generates a signal to invoke ITI for tree induction. The pseudocode for this algorithm is shown below.

IncrementalLearningMode (Tree, World)

Step 1: **IF** Goal is not reached
Step 2: **THEN** Choose a direction at random
Step 3: Evaluate the previous motion
Step 4: **IF** Previous motion interesting (output of the cost function)
Step 5: **THEN** Set up an appropriate training vector
Step 6: (Re-) Infer previous tree to absorb new knowledge
Step 7: Knowledge = UseKnowledge (Tree, World)
Step 8: **IF** Knowledge
Step 9: **THEN IF** Goal
Step 10: **THEN** Go to step 14
Step 11: **ELSE** Go to step 3
Step 12: **ELSE** Go to step 2
Step 13: **ELSE** Go to step 2

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

Step 14: END.

Note that in step 6 only *new* knowledge entities are evaluated for possible tree induction. The sub-routine UseKnowledge allows the tree to be searched concurrently with new knowledge being acquired, enabling the robot to have access to its previous experiences. This sub-routine is defined as follows.

UseKnowledge (Tree, World)

Step 1: IF Useful knowledge for the current world
Step 2: THEN Fetch knowledge
Step 3: Generate an appropriate control action using the tree
Step 4: Go to step 5
Step 5: END.

Table I. An illustration of how the linear tree evolves over a series of generations. The tree is highly goal-oriented and is used for predictions in w_0 (an environment with no obstacles). Where a class is present in the tree it is shown together with the number of training examples used in its derivation

F(1)	GoalRelLoc=N F(1) BL(1)	GoalRelLoc=N F(2) BL(1)	GoalRelLoc=N F(2) GoalRelLoc=NW BL(1) L(1)
a) 1 generation	**b) 2 generations**	**c) 3 generations**	**d) 4 generations**
GoalRelLoc=N F(2) GoalRelLoc=NW BL(1) GoalRelLoc=SW L(1) FR(1)	GoalRelLoc=N F(2) GoalRelLoc=E FR(1) GoalRelLoc=NE BR(1) GoalRelLoc=NW BL(1) BL(1)L(1)	GoalRelLoc=N F(2) GoalRelLoc=SW BL(1)L(1) GoalRelLoc=NW BL(1) GoalRelLoc=E FR(1) GoalRelLoc=NE BR(1) FR(1)BR(1)	GoalRelLoc=N F(2) GoalRelLoc=NW BL(1)FL(1) GoalRelLoc=SW BL(1)L(1) GoalRelLoc=NE BR(2) GoalRelLoc=E FR(1) FR(1)BR(1)
e) 5 generations	**f) 7 generations**	**g) 9 generations**	**h) 11 generations**
GoalRelLoc=N F(2) GoalRelLoc=NW BL(1)FL(2) GoalRelLoc=SW BL(2)L(1) GoalRelLoc=W L(1) GoalRelLoc=NE BR(2) GoalRelLoc=E FR(1) FR(1)BR(1)	GoalRelLoc=N F(3) GoalRelLoc=NW BL(1)FL(2) GoalRelLoc=SW BL(2)L(1) GoalRelLoc=W BL(1)L(1) GoalRelLoc=NE BR(3) FR(2)BR(2)	GoalRelLoc=N F(3) GoalRelLoc=NW BL(2)FL(2) GoalRelLoc=NE BR(4) GoalRelLoc=SW BL(2)L(1) GoalRelLoc=W BL(1)L(1) GoalRelLoc=E FR(2)BR(1) FR(1)BR(1)	GoalRelLoc=N F(3) GoalRelLoc=NE BR(4) GoalRelLoc=E FR(2)BR(1) GoalRelLoc=SE FR(1)BR(1) GoalRelLoc=NW BL(3)FL(2) GoalRelLoc=SW BL(2)L(1) BL(1)L(1)FL(1)
i) 14 generations	**j) 18 generations**	**k) 21 generations**	**l) 23 generations**
GoalRelLoc=N F(2) GoalRelLoc=SW F(1)FL(1)R(1)BR(1)L(1) GoalRelLoc=NW FL(2)BL(1)L(2) GoalRelLoc=W L(1) GoalRelLoc=SE FR(2) GoalRelLoc=E FR(2)R(1)BR(3) FR(3)R(1)BR(3)	GoalRelLoc=NE FL(2)BL(1)L(4) GoalRelLoc=SW F(1)FL(1)R(1)BR(1)L(1) GoalRelLoc=N F(2) GoalRelLoc=W L(1) GoalRelLoc=SE FR(2) GoalRelLoc=E FR(2)R(1)BR(4) FR(4)R(1)BR(4)	GoalRelLoc=NW FL(3)BL(2)L(7) GoalRelLoc=SW F(1)FL(1)R(1)BR(1)L(1) GoalRelLoc=N F(2) GoalRelLoc=W L(1) GoalRelLoc=E FR(2)R(1)BR(4) GoalRelLoc=SE FR(2) FR(7)R(1)BR(6)	
m) 28 generations	**n) 33 generations**	**o) 42 generations**	

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

(ii) *Intelligent navigation*. This part implements the worlds w_0 to w_3 as finite state machines (FSMs) in which each world is represented as an independent state. The dynamic behaviour of the robot occurs as a result both of the actions of the decision trees and of the transitions between trees, which are controlled by the FSMs. The procedure for the intelligent navigation process is shown below.

NavigationMode (Tree, World)

In state *n* world, w_x is detected ($\boldsymbol{w}_x \in$ **World**)
Step 1: **IF** Goal is not reached
Step 2: **THEN IF** tree *x* exists (**tree** $\boldsymbol{x} \in$ **Tree**)
Step 3: **THEN IF** useful knowledge available for current world
Step 4: **THEN** Fetch knowledge
Step 5: Generate an appropriate control action
Step 6: NavigationMode (Tree, World)
Step 7: **ELSE**
Step 8: **WHILE** $(x > 0)$
Step 9: $x = x - 1$
Step 10: Search tree *x* for knowledge to train world w_{x+1}
Step 11: Go to step 3
Step 12: IncrementalLearningMode (Tree, World)
Step 13: **ELSE** Go to step 12
Step 14: **END.**

7. ILLUSTRATION OF A LOCALLY TRAINED BINARY DECISION TREE

In the vast majority of cases, a decision tree consists of a finite number of decision nodes and terminal nodes (classes) linked together to form a complete network. In extreme cases, the tree can lead to only a single class (see Table I(a)). Decision nodes are generated in descending order from the root of the tree and each accommodates a particular feature with a unique outcome. The order of each test node in the hierarchy is arranged either information theoretically[20] or is based on other merits, for example Kalmogrov-Smirnoff distance in reference [31]. Unlike C4.5,[20] which is an ID3 descendant, the decision tree networks produced by ITI-2.8 are of a binary nature. This implies that, at any given test node, if the test result is positive (yes) the tree branches to the left, otherwise the search is directed to the right. This process to find a match for a given pattern proceeds until the leaf is reached which provides a classification for the pattern.

Figures 3 to 5 show local predictors (the DTs for w_0 and w_2) which are chosen from the hierarchy, and used in combination with the other DTs to provide a plan for the global path. The behaviour dominance in the tree hierarchy changes in a complementary fashion, meaning that, if goal-orientedness is the dominant behaviour in a layer such as w_0, reactivity is of minor influence. Conversely, in a highly reactive layer such as w_3, target seeking behaviour is practically non-existent.

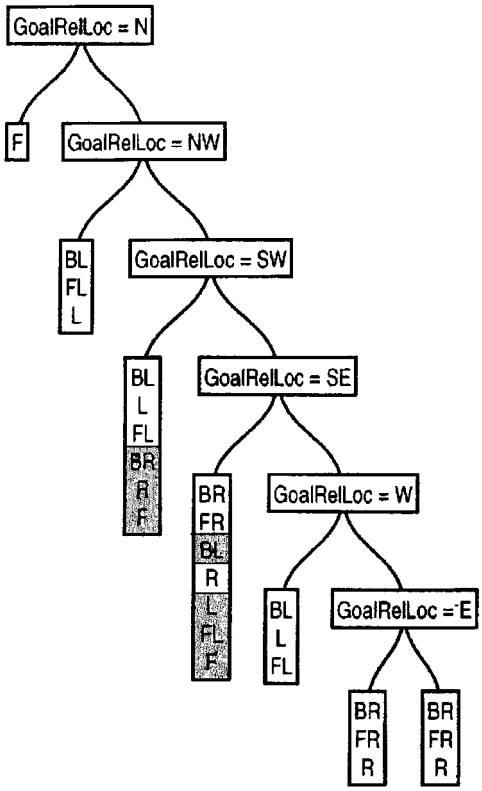


Fig. 3. A linear decision tree representing w_0 (world zero of the hierarchy). This is used for prediction in environments without any objects.

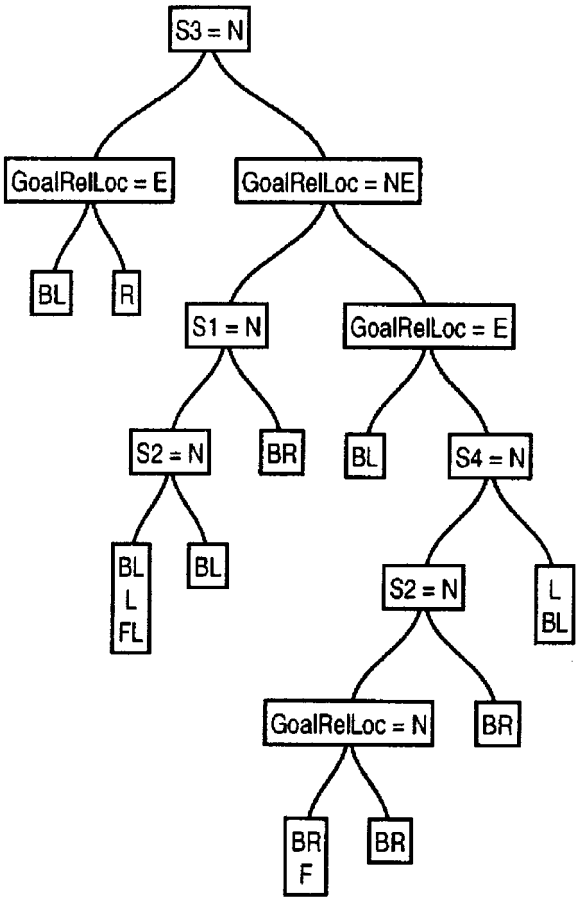


Fig. 4. An intermediate stage in the evolution of w_2.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

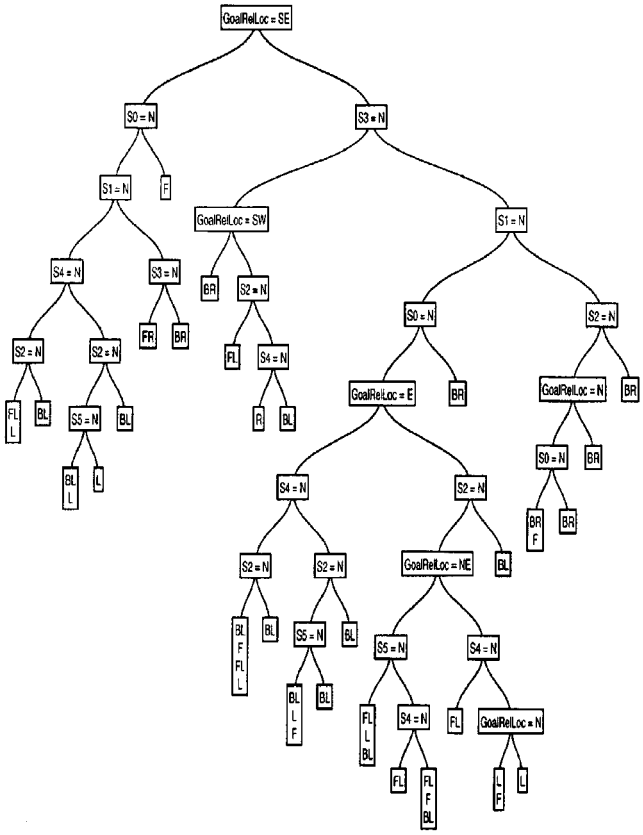


Fig. 5. The decision tree for w_2.

8. ACTION SELECTION AND CONFLICT RESOLUTION

From a decision tree point of view, inconsistent training examples occur when more than one class would be able to classify the same input pattern.[30] In the current robotics application, for a given perceptual pattern more than one rule can be fired to generate a control action. To resolve the conflict, a single value is generated from a rectangular distribition to produce a percentage measure. ITI-2.8 also associates every rule with a "frequency tag", and that rule is fired whose frequency tag matches the randomly generated frequency measure. This results in consistent action commands being sent to the actuation level to drive the robot forwards.

9. DYNAMIC RULE INHIBITION

Another important aspect of the proposed learning algorithm is the monitoring and long term assessment of incrementally-learned rules. In the learning phase, when the goal direction is located between the lines of maximum sensitivity of two adjacent sensor groups, rules can be evaluated by the system as useful, although they would quantitatively be considered to deliver poor performance under normal navigation circumstances. This effect is also a source of multiple-class generation. To exclude these types of rule, each individual rule is assessed after it has been fired depending on whether the resulting performance was satisfactory and the performance is used to give an indication of its overall usefulness. If the overall usefulness drops below a pre-set level, the rule is flagged and inhibited. Even though inhibited rules physically exist in the network, they have no actual contributions. These rules are shown shaded in Figures 3 to 5, where appropriate, to highlight the inhibiting mechanism.

10. RESULTS

Each of the aforementioned modules, namely w_0 to w_3, is a unique computational entity with a dynamic life time. Some have a process-long life whereas other types of entity can be generated during the process, but may also be destroyed to avoid an unnecessary increase in computational cost.

10.1 The simulated robot

In the current stage of the work, the control algorithm is verified in different scenarios using the Khepera[32] robot simulator. Khepera is a commercially-available miniature

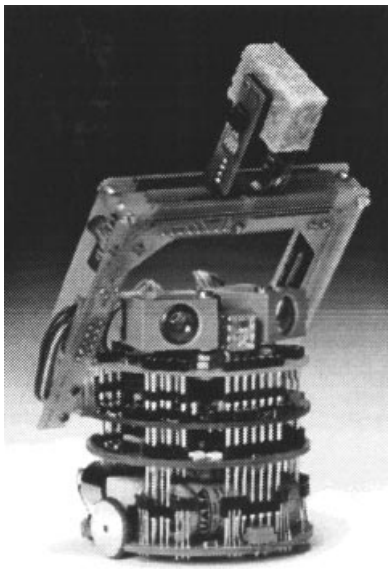


Fig. 6. Actual Khepera robot equipped with vision and gripper (Photo by Alain Herzog, Courtesy of EPFL Laboratory, Lausanne).

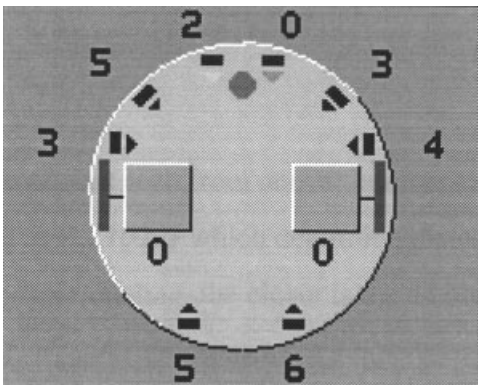
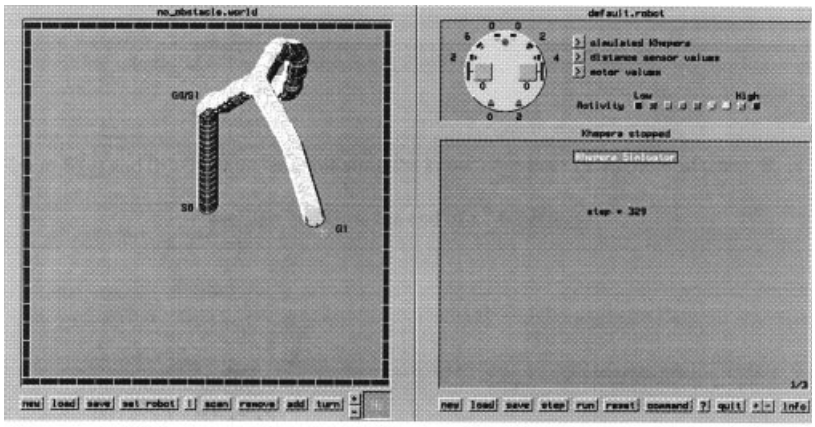
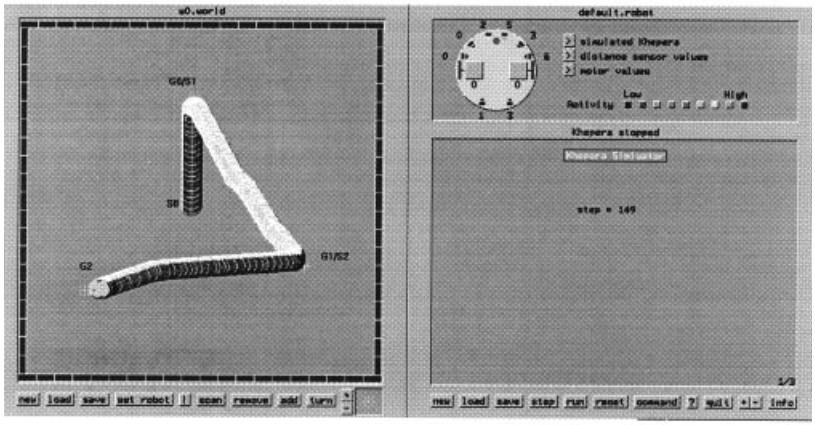


Fig. 7. Simulated Khepera robot seen from above; numbers associated with sensors are from the set of $D=\{0, 1, 2, 3, \ldots, 1023\}$ which determine the corresponding distance from obstacles. The greater the sensor reading, the closer is the an obstacle to the corresponding sensor.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

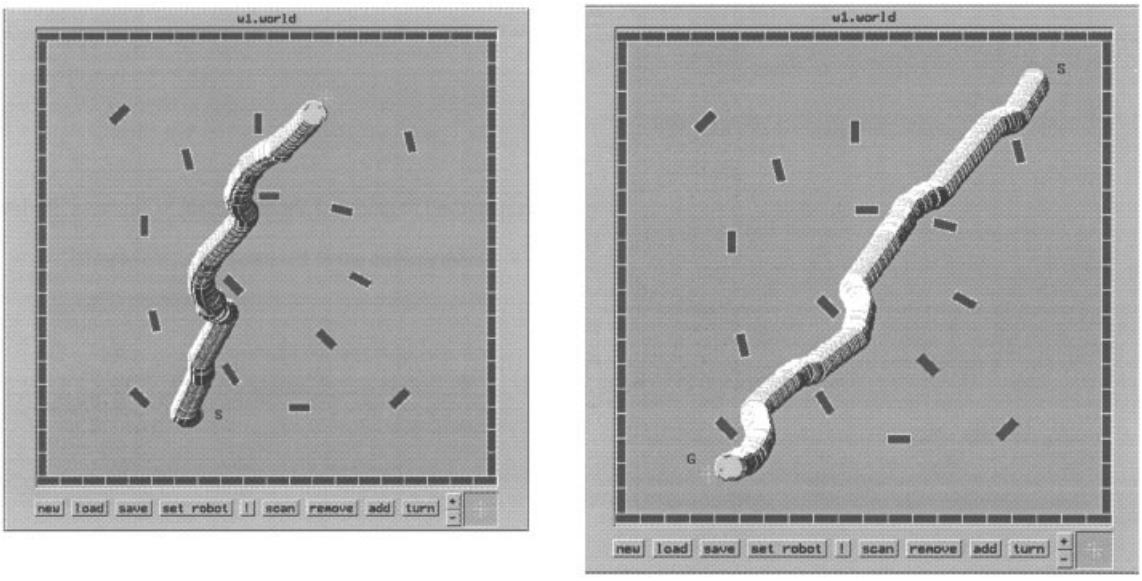


(a) Training in a "two-leg" trajectory



(b) Target-seeking in a "three-leg" trajectory

Fig. 8. (a) and (b): Training and initiating the knowledge base in w_0 (no obstacles). S_x is the start and G_x is the end of x^{th} trajectory.



(a) A training scenario on w_1

(b) Navigation in w_1 after sufficient training

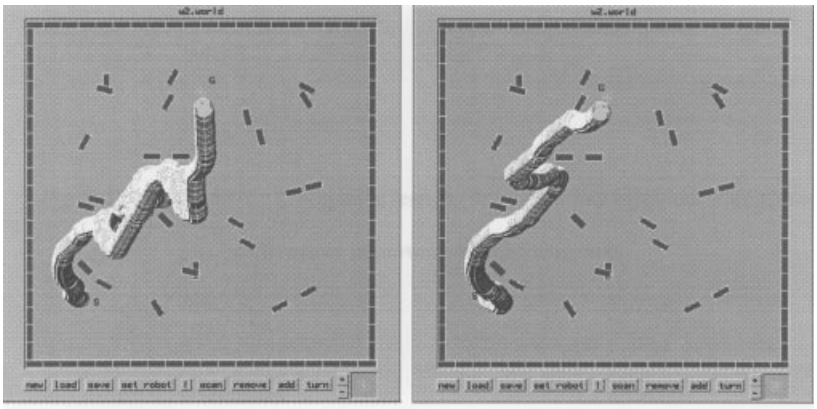
Fig. 9. (a) and (b): Example of learning in w_1. Learning to become reactive by avoiding simple obstacles scattered in the environment.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

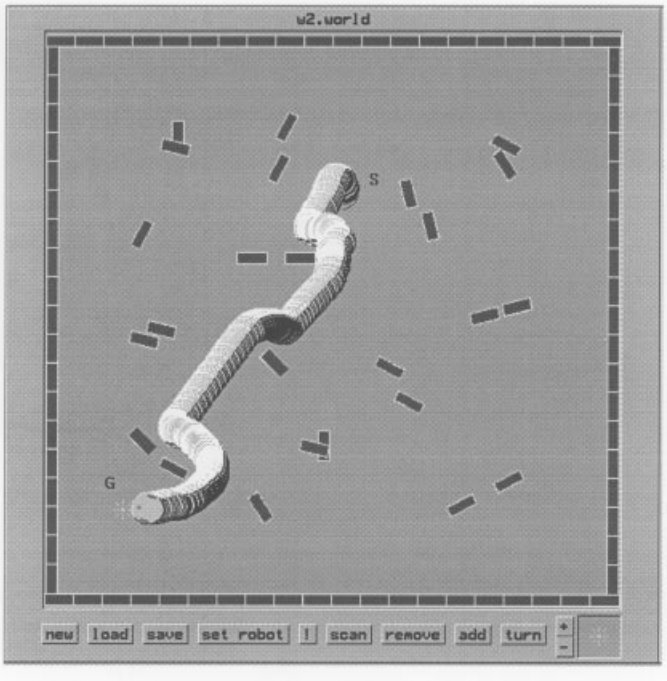
mobile robot, shown in Figure 6. Figure 7 shows the simulated counterpart. Khepera has access to an array of eight light sensors as well as eight infra-red proximity sensors for range measurement. The current work uses only the six frontal infra-red distance sensors to scan the areas to the front and sides of the robot for object detection, and each returns a value in the range 0 to 1023 depending on the nature and the range of the detected objects. In general, the greater the magnitude of a sensor return value, the closer is the obstacle. The simulated robot is steered differentially by controlling the wheels individually. The Khepera simulator has been designed to incorporate realistic assumptions and to minimise the presence of unrealistic suppositions, facilitating the transfer of the simulation results without major change directly to the real Khepera robot.[33]

10.2 Examples of homing tasks

As discussed previously, global learning is initiated in w_0 and propagates up the hierarchy to the more complex worlds. This is carried out by adjusting the dimensions of the previous rule layer to adapt to new knowledge in the current layer. In all the examples, the robot is initially set at the starting point S and is expected to reach the light source G, also indicated by a star symbol. The navigation fields chosen consist of typical situations that can occur in a navigatory task, for example sharp corners and edges, right angle corners, and both straight and rounded walls with or without discontinuities. The different stages of Figure 8 demonstrate an example of the learning process in w_0: it illustrates how the learning is formed (Figures 8(a)) and improved in two-leg and in multi-leg trajectories (Figures 8(b)). Figures 9, 10 and 11 illustrate learning and adaptation in the worlds w_1, w_2 and w_3. It is revealing of the operation of the learning mechanism to view the reaction of the robot to a new environment (Figure 10(a)) while trying to adapt to new knowledge. After the corresponding tree has incorporated sufficient knowledge, the robot is able to navigate smoothly towards the set target (Figure 10(c)). Behaviour



(a) First stage of learning in w_2 (b) An intermediate stage of learning in w_2



(c) A navigation example after sufficiently training the robot in w_2

Fig. 10. (a), (b) and (c): Examples of learning in w_2. Showing behaviour learning and performance improvement in homing tasks.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

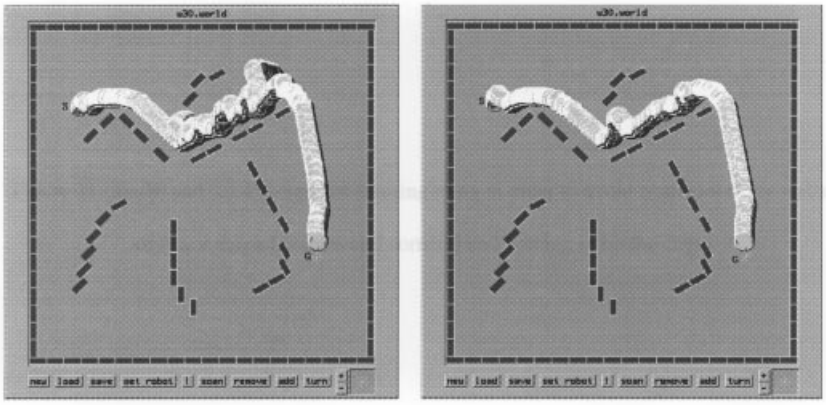
dominance in a number of the decision trees can trigger "zigzag" motions when the robot follows a long wall, and this can be seen in Figures 11(a), (b) and (c).

11. CONCLUSIONS AND FURTHER WORK

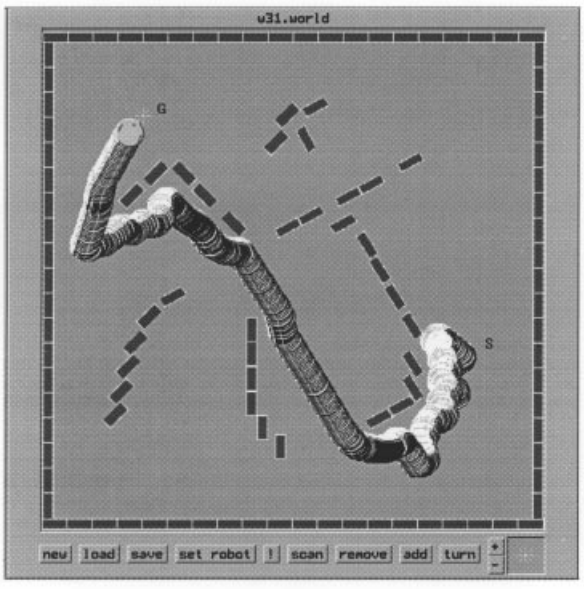
This paper has introduced a new approach to behaviour learning and global navigation which is biologically inspired and uses decision tree learning. The robot environment is decomposed into a set of homogeneous perceptual worlds of differing complexity. The robot learning system uses the experience of exploring the robot environment to train each of the layers on-line and incrementally. At the navigation stage, each perceptual world is mapped to a unique rule layer which can be searched for prediction purposes.

The core of the system is composed of two algorithms for on-line learning and navigation, distributed over a string of four intelligent local predictors that are independent computational entities with dynamic life times. They are used in combination and are sampled continuously to perform globally-shaped motion predictions. An important attribute of the system is the dynamic rule inhibition which is based on a long term assessment of the performance of the rule layers in the navigation task. The control system ensures safe and globally tuned predictions: safety is achieved by individual local predictors and global shaping by their coupling.

A shortcoming of the current implementation which is exhibited in some of the homing tasks (section 10.2), is the unsmoothness of robot trajectories when following long walls. This effect has also been reported by Reignier in reference [34]. Two possible sources for this "zigzag" motion are the presence of multiple-class rules and abrupt behaviour switching as control is moved between rule layers. In the current work, the effect of the former is not significant as the classes can generally be considered to have overlapping areas of operation. However, the effect of the latter source is significant. In response to this drawback, it is the intention to introduce fuzziness into our algorithm to blend behaviours and optimise global paths.



(a) First stage of learning to follow walls (b) Improvement of wall following behaviour



(c) A homing task in w_3 while following walls and corners

Fig. 11. (a), (b) and (c): Example of learning in w_3 in order to avoide obstacles (long walls, arbitrary shaped objects and corners) and homing in on the target.

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press

In the current work it is argued that a suitably accurate robot localisation method combined with the control concept proposed in this paper can be merged to produce a robust, computationally inexpensive and easy to implement intelligent path planner for real-world environments. The symbolic nature of this technique gives the advantage of control laws which are intelligible to human users, in contrast to those obtained as a result of applying connectionist methods.

References

1. P. Reignier, "Fuzzy logic techniques for mobile robots obstacle avoidance", *Robotics and Autonomous Systems* **12** 143–153 (1994).
2. C.J. Wu, "Fuzzy robot navigation in unknown environments", *IEEE International Workshop on Emerging Technology and Factory Automation* (August 11–14, 1992) pp. 624–628.
3. H. Surmann, J. Huser and L. Peters, "A fuzzy system for indoor mobile robot navigation", *FUZZ-IEEE*, Yokohama, Japan (1995) pp. 83–86.
4. A. Saffiotti, E.H. Ruspini and K. Konolige, "Robust execution of robot plans using fuzzy logic", *IJCAI'93, Fuzzy Logic in Artificial Intelligence*, Springer-Verlag (1994) pp. 24–37.
5. A.Dubrawski and J. Crowley, "Self-supervised neural system for reactive navigation", *IEEE International Conference on Robotics and Automation* (1994) **Vol. 3**, pp. 2076–81.
6. J. Tani and N. Fukumura, "Learning goal-directed sensory-based navigation of a mobile robot", *Neural Networks*, **7**, No. 3, 553–563 (1994).
7. T. Shibata, T. Abe, K. Tanie and M. Nose, "Motion planning of a redundant manipulator-criteria of skilled operators by Fuzzy-ID3 and GMDH and Optimisation by GA", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (1995) **Vol. 4**, pp. 3730–3735.
8. T. Zrimec and P. Mowforth, "Learning by an autonomous agent in the pushing domain", **In:** (W. Van de Velde, Ed) Toward Learning Robots (The MIT Press, Cambridge, Mass., 1993).
9. I. Sillitoe and T. Elomaa, "Learning decision trees for mapping the local environment in mobile robot navigation", *Proceedings of MLC-COLT Workshop on Robot Learning* New Brunswick, N.J. (July, 1994) pp. 119–125.
10. A.C. Schultz, "Learning robot behaviours using genetic algorithms", *Proceedings of the International Symposium on Robotics and Manufacturing* (August 14–18, 1994) pp. 607–612.
11. J. Grefenstette, "Evolutionary algorithms in robotics", *Proceedings of the Fifth International Symposium on Robotics and Manufacturing, ISRAM'94* (1994) pp. 127–132.
12. J. Bala, J. Huang, H. Vafai, K. DeJong and H. Wechsler, "Hybrid learning using genetic algorithms and decision trees for pattern classification", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada (August 19–25, 1995) pp. ???.
13. R. Braunstingel, J. Mujika and J.P. Uribe, "A wall following robot with a fuzzy logic controller optimised by a genetic algorithm", *FUZZ-IEEE-IFES'95 Fuzzy Robot Competition*, Yokohama (March, 1995) pp. 77–82.
14. D.T. Lawton, R.C. Arkin and M. Cameron, "Qualitative spacial understanding and reactive control for autonomous robots", *IEEE International Workshop on Intelligent Robots and Systems, IROS'90* (1990) **Vol. 2**, pp. 700–714.
15. A. Fagg, D. Lotspeich and G. Bekey, "A reinforcement-learning approach to reactive control polocy design for autonomous robots", *IEEE International Conference on Robotics and Automation* (1994) pp. 39–44.
16. M. Kaiser, V. Klingspor, J. Millan and M. Accame, "Using machine learning techniques in real-world mobile robots", *Intelligent Robot Systems, IEEE Expert*, 37–45 (1995).
17. A. Dubrawski and J. Crowley, "Learning locomotion reflexes: A self-supervised neural system for a mobile robot", *Robotics and Autonomous Systems* **12**, 133–142 (1994).
18. G.H. Shah Hamzei, D.J. Mulvaney and I. Sillitoe, "Multi layer hierarchical rule learning in reactive robot control using decision trees", *Int. J. Intelligent and Robotic Systems* 1–26 (1998).
19. J.R. Quinlan, "Decision trees and decisionmaking", *IEEE Transactions on Systems, Man, And Cubernetics* **20**, No. 2, 339–346 (1990).
20. J.R. Quinlan, *C4.5: Programming For Machine Learning* (Morgan Kaufmann Publishers, 1992).
21. P.E. Utgoff, "Incremental induction of decision trees", *Machine Learning* **4**, 186 (1989).
22. I. Kononenko, "Inductive and baysian learning in medical diagnosis", *Applied Artificial Intelligence* **7**, 317–337 (1993).
23. B. Cestnik, I. Bratko and I. Kononenko, "Assistant 86: A Knowledge Elicitation Tool for Sophisticated Users", *Progress in Machine Learning (Proceedings of the 2nd Euroipean Working Session on Learning)* (I. Bratko and N. Lavrac, Eds.) (Sigma Wilmslow, UK, 1987) pp. 31–45.
24. S. Salzberg, R. Chander, H. Ford, S.K. Murthy and R. White, "Decision trees for automated identification of cosmic ray hits in bubble space telescope images", *Publications of Astronomical Society of the Pacific* **107**, 1–10 (May, 1995).
25. G.H. Shah Hamzei, D.J. Mulvaney and I. Sillitoe, "Batch-mode decision tree learning applied to intelligent reactive robot control", *Sixth IEEE International Conference on Emerging Technologies and Factory Automation ETFA'97*, Los Angeles, USA.
26. G.H. Shah Hamzei and D.J. Mulvaney, "System Instability and Oscillation Resolution in Reactive Robotics Using Decision Tree Based Approach to Learning", *International Conference of Artificial Intelligence and Soft Computing*, Banff, Canada (July 27–August 1, 1997) pp. 411–414.
27. R. Braunstingl, P. Sanz and J.M. Ezkerra, "Fuzzy Logic Wall Following of a Mobile Robot Based on the Concept of General Perception", *ICAR'95, The Seventh International Conference on Advanced Robotics*, Sant Feliu de Guixols, Spain (September 1995) pp. 367–376.
28. C. Sammut, S. Hurst, D. Kedzier and D. Michie (Eds.), "Learning to Fly", *Proceedings of the Ninth Machine Learning Conference* (Morgan Kaufmann, 1992) pp. 385–393.
29. J. Cussens, "Machine learning", *Computing & Control Engineering Journal* **7**, No. 4, 164–168 (1996).
30. P.E. Utgoff, "Decision Tree Induction Based On Efficient Tree Restructuring", *Technical Report* 95–18 (March 17, 1995).
31. P.E. Utgoff and J.A. Clouse, "A Kolmogrov-Smirnoff Metric for Decision Tree Induction", *Technical Report* 96–3 (1996).
32. O. Michel, Mage Team, I3s Laboratory, CNRS, University of Nice-Sophia Antipolis, France, downloadable from: http://alto.unice.fr/~om/khep-contest.html.
33. O. Michel and P. Collard, Artificial Neurogenesis, "An Application to Autonomous Robotics", *Proceedings of the Eighth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press (1996) pp. 207–214.
34. P. Reignier, "Fuzzy logic techniques for mobile robots obstacle avoidance", *Robotics and Autonomous Systems* **12**, 143–153 (1994).

https://doi.org/10.1017/S0263574799001319 Published online by Cambridge University Press