MUTUAL LEARNING BY AUTONOMOUS MOBILE ROBOTS.

Ian D. Kelly*, David A. Keating & Kevin Warwick

Department of Cybernetics, University of Reading, Whiteknights, Berkshire, RG6 6AY, UK.

* email : cybidk@cyber.reading.ac.uk

Abstract

This paper describes a reinforcement learning algorithm for small mobile robots based on sets of fuzzy automata. The task the robots have to learn is how to reactively avoid obstacles. In mutual learning two robots learn simultaneously, with the experiences of one robot being passed to the second robot. We show that the robot that receives the other robots experiences learns more quickly and robustly than the robot not receiving such information.

1.0 Introduction

Over recent years we have been building autonomous mobile robots which adhere to the philosophies of Artificial Life (AL). Traditional, Artificial Intelligence (AI) practitioners build robots that employ fully specified, top down, sequential control strategies, using complicated computers to process the data from complex sensors like video cameras. We believe that it is "better" to start with robots that have simpler sensors and hence simpler (although not trivial) control strategies. Our first group of robots, which are known as the Seven Dwarfs, do not even have a microprocessor, instead they use a pre-programmed look up table stored in EEPROM. In conjunction with simple sonar sensors this look up table allows the Seven Dwarfs to successfully wander around an environment avoiding both stationary and moving obstacles at a speed of 1ms-1.

Recently there has been much interest in the development of intelligent machines which can learn from their past experiences in their environment [4], [7], [10]. To this end we developed a robot that learns how to react to its environment [5]. This learning robot also uses simple sonar sensors and a low power microprocessor (a 4MHz Z80). This successfully learned, in a few minutes, how to avoid objects. With the development of communicating robots [3] we have recently started studying dual learning. This involves one robot sharing its experiences with another robot, hence decreasing the overall time taken for the robots to learn.

2.0 The Seven Dwarfs

Each of the Seven Dwarfs (as shown in figure 1), has two pairs of ultrasonic transducers, which each have a useful range of about 150 to 500mm. The sonar system processes data from these transducers to produce 4 bits of information, one for which side the nearest obstacle is on (left or right) and the other three give the range to the obstacle. Four mode select switches produce another four bits of information, which allow each robot to have 16 different control programs. The address formed by the data from the sonar and the mode select switches is used to access an Electrically Erasable Programmable Read Only Memory (EEPROM) which forms a look up table of the required motor actions for each possible obstacle position. Eight bits of data are produced by the EEPROM. Each motor driver requires four bits of information, one bit for direction and three for speed (0=stop, 7=full speed). Each motor driver produces a pulse width modulated signal which controls the motor speed via a H-bridge of MOSFETs. The sonar and look up table are read forty times a second allowing the robots to easily travel at 1ms-1. The seven dwarfs are manually programmed, and so far the strategies include avoid obstacles, follow nearest object, and puppy mode; where an object is followed until it gets too close, upon which the robot will back away. A simple avoid program in C is :-

```
if (Range==7)  /* nothing visible, forward at full speed */
{
        LeftSpeed=7; LeftForward=TRUE;
        RightSpeed=7; RightForward=TRUE;
}
else if (Range<4)  /* object very close, back away */
{
        LeftSpeed=3; LeftForward=FALSE;
        RightSpeed=3; RightForward=FALSE;
}
else if (LeftNotRight)  /* object on left, turn right */
{
        LeftSpeed=3; LeftForward=TRUE;
        RightSpeed=3; RightForward=FALSE;
}
else  /* object on right, turn left */
{
        LeftSpeed=3; LeftForward=FALSE;
        RightSpeed=3; RightForward=TRUE;
}
```

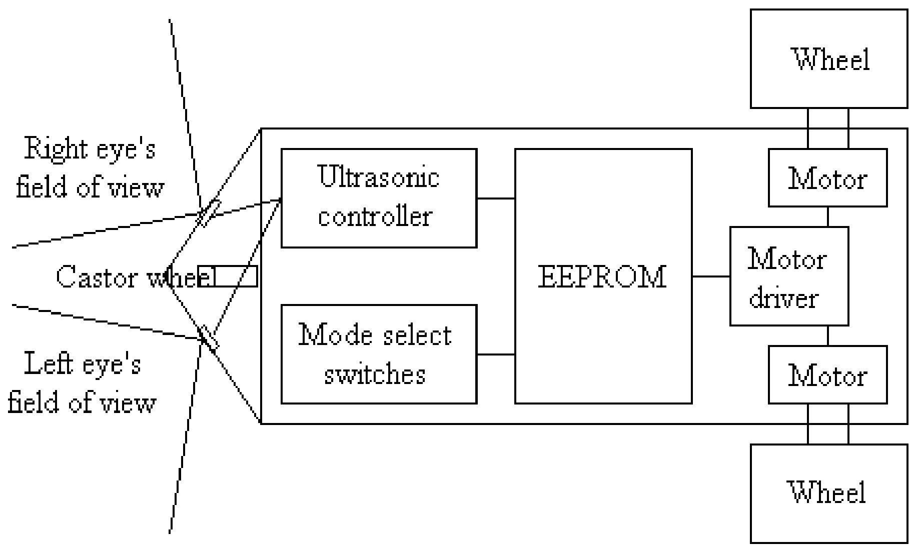

Figure 1, block diagram of one of the Seven Dwarfs.

Each of the basic actions of the Seven Dwarfs can be considered as a straightforward reactive response [9]. In this way obstacle avoidance essentially means that a robot will move around in a forwards direction, moving to the left or right in order to avoid hitting obstacles on the right of left respectively. Alternatively by following along behind an object the robots tend to behave in a way that can be considered to be rather like ducklings. This is extended in "puppy mode" to include backing away if the object gets too close.

Other versions of robots developed from the Seven Dwarfs include infrared transmitters and receivers for position detection. These have been used to allow a robot to identify beacons which represent predators, prey or charging stations, from the infrared signal they generate, and to react accordingly, i.e. to flee, chase or dock [2]. For the docking behaviour the robot monitors how much charge has been drawn from its battery and calculates when it needs to go and recharge its battery. Recharging is achieved by finding and docking the robots front bumpers with a charging station. Further to this robot, a group of robots have been equipped with a similar infrared localisation and communication system. These robots can use their infrared system to flock with each other in a similar manor to sheep [3].

3.0 The mutual learning robots

Two autonomous mobile robots have been constructed which are equipped with a radio communication system and an improved ultrasonic sonar for detecting obstacles. The obstacle detection system consists of three sets of ultrasonic transducers, one set looking forward, another to the front-left and the third to the front-right. Unlike the Seven Dwarfs, this sonar system returns the range to the nearest obstacle in front of each set of ultrasonic transducers. To allow the robots to detect close obstacles, echoes have to be detected whilst the ultrasonic pulse is being transmitted, thus requiring a high threshold. For the detection of objects further away, a much lower threshold is required to allow for the large signal loss. To allow the robots to detect both near and distant obstacles, a varying threshold system is used which is initially large and decreases with time to a pre-set minimum. A time-out system is used to determine if there are no objects within range. Each set of ultrasonic sonar transducers is scanned ten times per second, and has a range of 30mm to 1m and a resolution of better than 5mm. Both of the robots are equipped with a radio transmitter and receiver, working at 418MHz, hence time division multiplexing is required for two way communication. The rate of data transfer over this communications system is 19,200baud.

Physically the robots are small, having a width of 140mm, a length of 130mm and a height of 140mm, see figure 2. Each robot also weighs less than 600 grams. Movement is provided by two small d.c. motors with in-line gear boxes connected to the back wheels. The front of each robot is supported by a single castoring wheel. At present motor control is provided by open-loop pulse width modulated controllers, providing several different speeds (up to 1ms-1) and directional control. Each robot is controlled by a single 8MHz Z80 CPU. The processor receives an interrupt when the radio receiver detects a carrier wave from another robot.

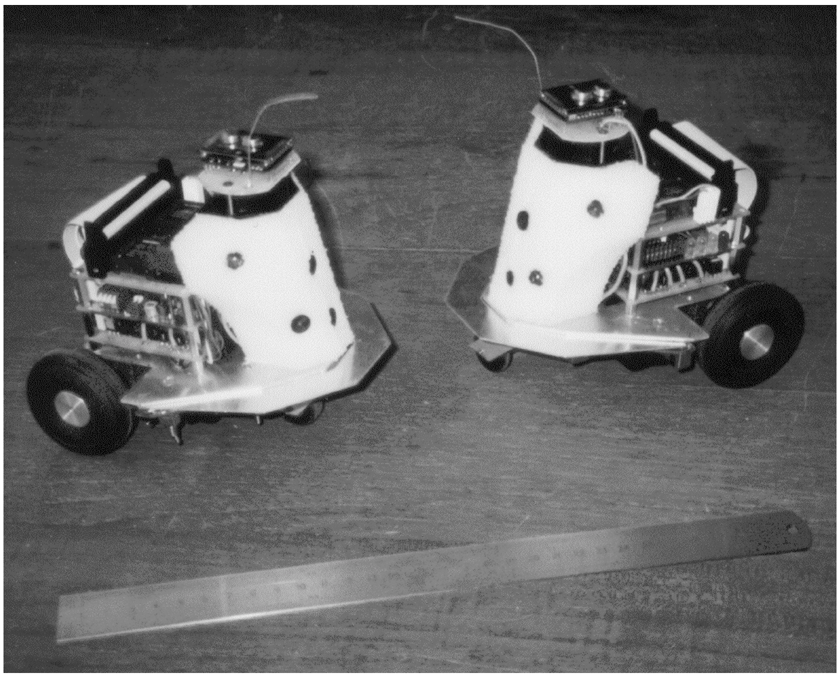

Figure 2, the pair of learning robots.

Power is provided by four 1.2V, 1800mAH nickel metal hydride rechargeable batteries. In order to keep the centre of gravity of each robot low, the batteries are placed underneath the robots. With maximal activity, in the worst case, the battery life is greater than five hours from a full charge.

4.0 The learning algorithm

The task of each robot is to learn to associate the "best" motor action to its current situation so that it moves around whilst avoiding obstacles. The learning algorithm is based on sets of fuzzy automata, similar to those described in [6], and is similar to one step Q-learning [8]. With two motors each with three possible speeds (forward at full speed, backwards at full speed and stop) there are nine available actions (a1..a9). The five different situations that a robot can find itself in are :

case 1 : no object near robot
case 2 : obstacle in distance to the right
case 3 : obstacle in distance to the left
case 4 : obstacle relatively near the right
case 5 : obstacle relatively near the left

It should be stated that this learning strategy does not try to build a map of the environment, instead the robot learns how the react to obstacles in the different positions. Each of these different circumstances has its own fuzzy automaton associated to it. Each automaton, which is effectively a set of motor actions (a1..a9), has a set of probabilities of taking the associated action (p1..p9). A weighted roulette wheel technique is used to randomly select the most appropriate action for the given input (obstacle position). The action with the highest probability is the most likely to be chosen. This method is similar to techniques that are used in genetic algorithms [1]. The chosen action is executed for a short period of time and is then evaluated. If the action was successful its probability of being selected is increased. If the action was a failure its probability of being selected is decreased. In both cases the other probabilities of the chosen automaton are adjusted so as to keep the total probability constant. If the performance value is Alpha (positive meaning successful and negative meaning unsuccessful), n is the action that was chosen and j is all the actions except for n, then the rules for adjusting the probabilities are :

```
Alpha = eAlpha                        (make Alpha non-linear)
Alpha = Alpha / Factor         (scale Alpha)
IF Alpha >= 0 THEN             (the action was successful)
    pn = pn + Alpha            (increase probability of chosen action)
    pj = pj - ( Alpha / 8 )    (decrease other probabilities)
ELSE                           (the action was unsuccessful)
    pn = pn - Alpha            (decrease probability of chosen action)
    pj = pj + (Alpha / 8 )     (increase other probabilities)
END
```

To emphasise both very good and very bad actions alpha is made exponential. It is then divided by one of three different values (Factor = 4, 2 or 1) which allow the effects of different learning rates to be investigated. Since we are only using a slow processor all of the probabilities are stored as signed 16 bit integers rather than as fractions.

In order to evaluate Alpha a definition is required of what actions are good and what ones are bad. Rules were chosen which were general and hence would not give the robot any information about which motor actions to select. The basic rules are that if there is no object within range it is good to go forward but if an object is relatively near it is good to get further away from it. Lastly if the object is in the distance it is still good to go forwards but it is also good to get further away from it. If the possible motor speeds are +1 (forwards), -1 (backwards) and 0 (stop), and FrontDir is positive if getting further away from the nearest object and negative otherwise, the rules for calculating Alpha are :

```
case 1 :                                   (no object near robot)
    Alpha = ( LeftSpeed + RightSpeed ) * 3

case 2 and case 3 :                        (obstacle in distance)
    Alpha = RightSpeed + LeftSpeed + ( FrontDir * 3 )

case 4 and case 5 :                        (obstacle relatively near)
    Alpha = FrontDir * 3
```

The main adaptive loop of the program is :

```
Choose probability set (automaton) according to obstacle position,
Choose action, an, of this automation,
Move insect for short period of time,
Evaluate Action and update probabilities.
```

On the original version of the learning robot the distance to the nearest obstacle was determined using the left and right sonar channels. However, in practice it was found that if the robot turned whilst facing an obstacle this could lead the robot into thinking that it was successfully getting further away from the obstacle when it was actually getting nearer to it. This is shown by the dotted lines in figure 3. This problem is overcome by using the front sonar channel to determine whether the robot is successfully getting further away from the object, as shown by the solid line in figure 3.

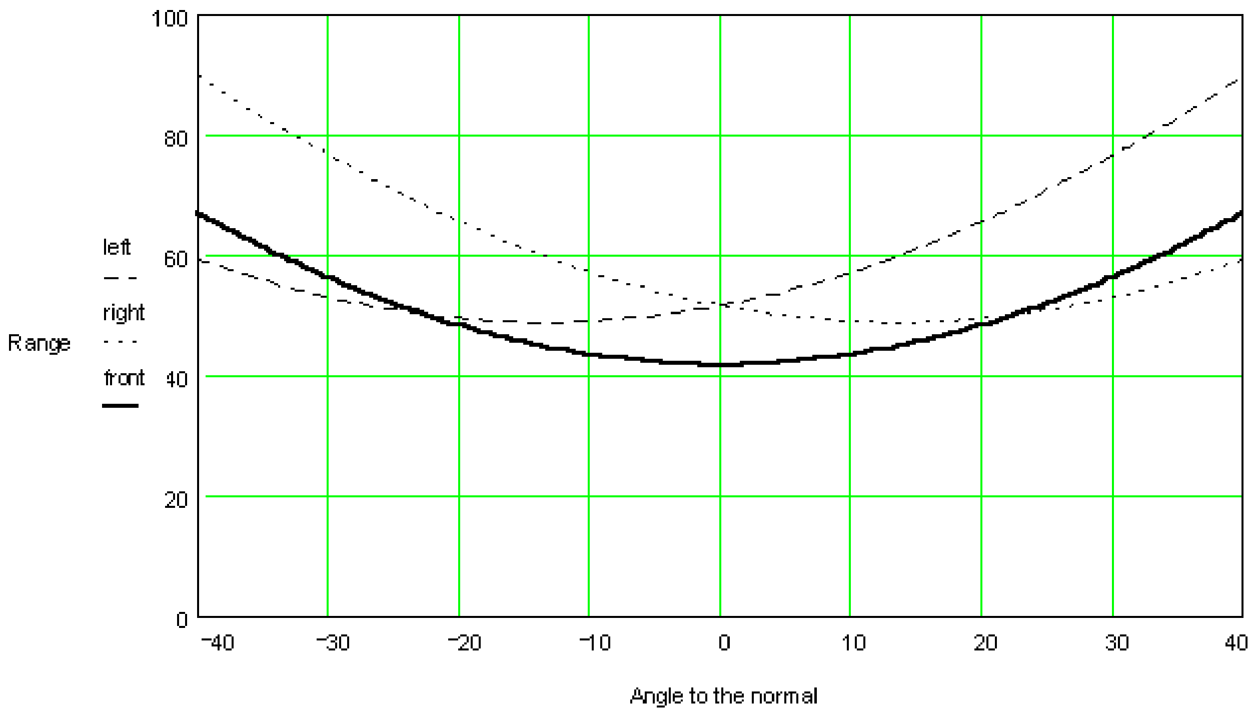

Figure 3, range returned by each sonar channel with changing angle.

5.0 Mutual Learning

For bi-directional dual learning each robot has to transmit the position of the object (e.g. case 1..5) that it reacted to, the action that it chose and how good or bad that action was (i.e. the Alpha factor). As well as this information a start of packet code and a check sum are also required, in total this gives a four byte packet. With the 10 bytes of synchronisation that are required for the receiver, the time to send a packet is 7ms. Since the main adaptive loop takes 200ms, five packets of data are sent and five packets of data are received every second. Since only 0.35% of processor time is required for receiving data, interrupts can be used. As stated above, when a transmitter is switched on an interrupt is generated on the other robot by the carrier detect line from the radio module. The main adaptive loop of the program now becomes :

```
    Choose probability set (automaton) according to obstacle position,
    Choose action, an, of this automation,
    Move insect for short period of time,
    Evaluate Action and update probabilities,
    Transmit information to other robot.
    (at any time an interrupt will cause information from the other robot
     to update one set of probabilities)
```

6.0 Results

The test set-up consisted of two robots each in its own environment bounded by an elliptical wall of 7.5m circumference and 120mm height. Both robots ran the learning algorithm but one had its receiver disabled preventing it from using the other robots experiences to learn. The goodness factor was calculated by comparing the values in the table of probabilities with "ideal" values determined by the authors. The goodness factor was evaluated five times a second and the average logged every second. Four runs were performed under each test condition to allow an average learning rate to be determined. Graphs of goodness factor vs. time are shown in figures 4 to 7.

Figure 4 shows the learning with the normal Alpha factor which had previously been determined to be the best compromise between speed of learning and robustness. The results for the four runs are shown for the single robot and dual learning cases, with the averages of the four runs shown as a bold line. The final graph shows the two averages for comparison. Figure 5 shows the same graphs but with an Alpha factor of half the normal value while figure 6 shows the same graphs again but with an Alpha factor of double the normal value. In each case the shared experience shows an improved learning rate over the individual case. With a high value of Alpha the learning is smoother when the robot has the other robots experiences to help reduce the effects of noise. This is shown clearly in figure 6.

Figure 7 shows a comparison between one robot learning with a given Alpha factor and dual learning with each robot having half the single robot’s Alpha factor. In the first case the single robot has the normal Alpha factor whilst in the second case the single robot has double the normal Alpha factor. When the single robot has the optimal value of Alpha the dual learning with half the value shows no advantage in speed of learning but shows a greater immunity to noise. When the Alpha factor is increased the learning rate for the dual learning shows not only increased noise immunity but is also faster.

7.0 Conclusions

The results clearly show that joint learning is faster and more robust than individual learning, although this advantage is smallest when the individual learning is optimal. The use of such learning also allows the advantages of increased Alpha factors to be exploited where the effects of noise would have prevented this in the individual case. The authors are currently investigating the effects of true mutual learning where two or more robots share each others experiences.

8.0 Acknowledgements

We would like to thank C. King for all his hours constructing the chassis of the robots, and P. Minchinton for designing the gate array circuits to control the processor and the sonar.

9.0 References

[1] GOLDBERG, D.E. "Genetic algorithms." Addison Wesley. 1989.

[2] KELLY, I.D. & HOLE M. "Robot Insect II." Internal research report. Dept. Of Cybernetics. University of Reading. 1993.

[3] KELLY, I.D. & KEATING, D.A. "Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots." Proc. Mechatronics 1996.

[4] MATARIC, M.J. "Learning in Multi-Robot Systems." In Adaptation and Learning in Multi-Agent Systems, Gerhard Weiss and Sandip Sen, eds., Lecture Notes In Artificial Intelligence (LNAI), Vol. 1042, Springer-Verlag. 1996.

[5] MITCHELL, R.J., KEATING, D.A. & KAMBHAMPATI, C. "Neural network controller for mobile robot insect." Proc. EURISCON 1994.

[6] NARENDRA, K. & THATHACHAR, M. "Learning Automata : an introduction.", Prentice-Hall. 1989.

[7] SCHAERF, A., SHOHAM, Y., & TENNENHOLTZ, M. "Adaptive load balancing : A study in multi-agent learning." Journal of Artificial Intelligence Research 2. May. 1995.

[8] WATKINS, C.J. & DAYAN, P. "Technical note : Q-learning." Machine Learning. Vol. 8(3-4), pp. 279-292. 1992.

[9] WARWICK, K., KELLY, I., GOODHEW, I. & KEATING, D. "Behaviour and learning in completely autonomous mobile robots." Proc. IEE Colloq. on The Design and Development of Autonomous Agents. London, pp 7/1 - 7/4. 1995.

[10] YAMAGUCHI, T., MASUBUCHI, M., FUJIHARA & K. YACHIDA, M. "Real-time reinforcement learning for a real robot in the real environment." IROS 96. Osaka. Japan. 1996.

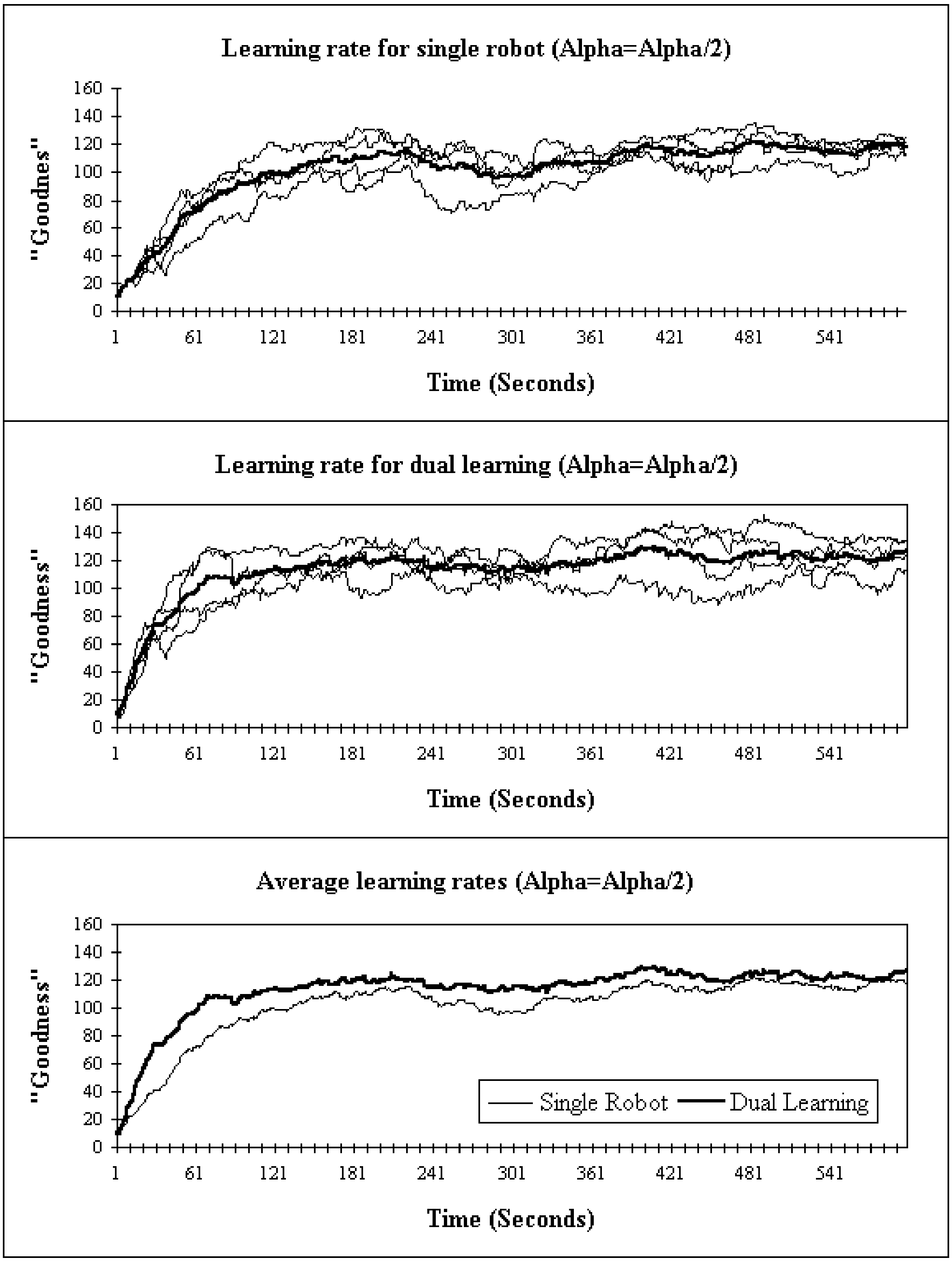

Figure 4, learning rates with optimal Alpha factor.

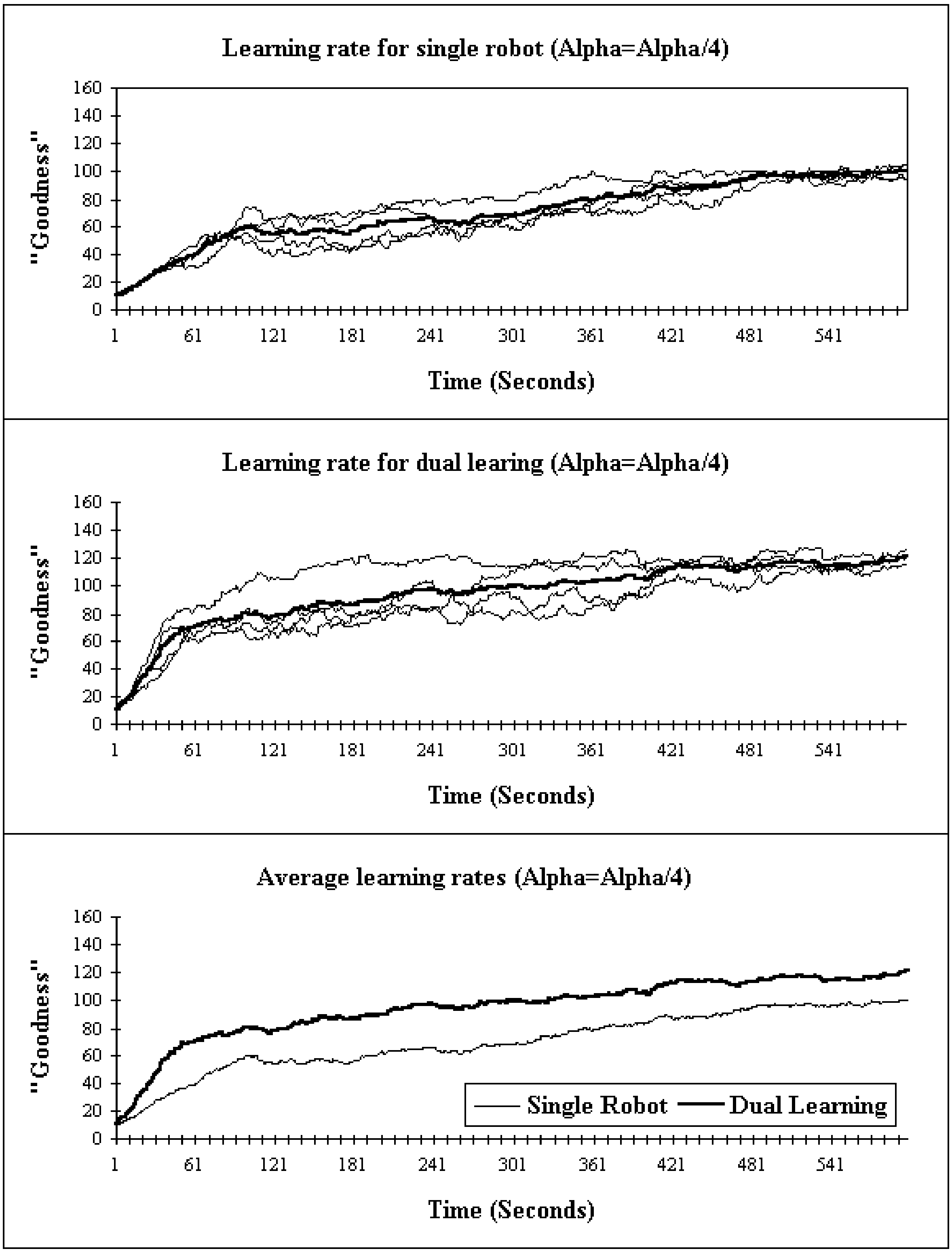

Figure 5, learning rates with half of the optimal Alpha factor.

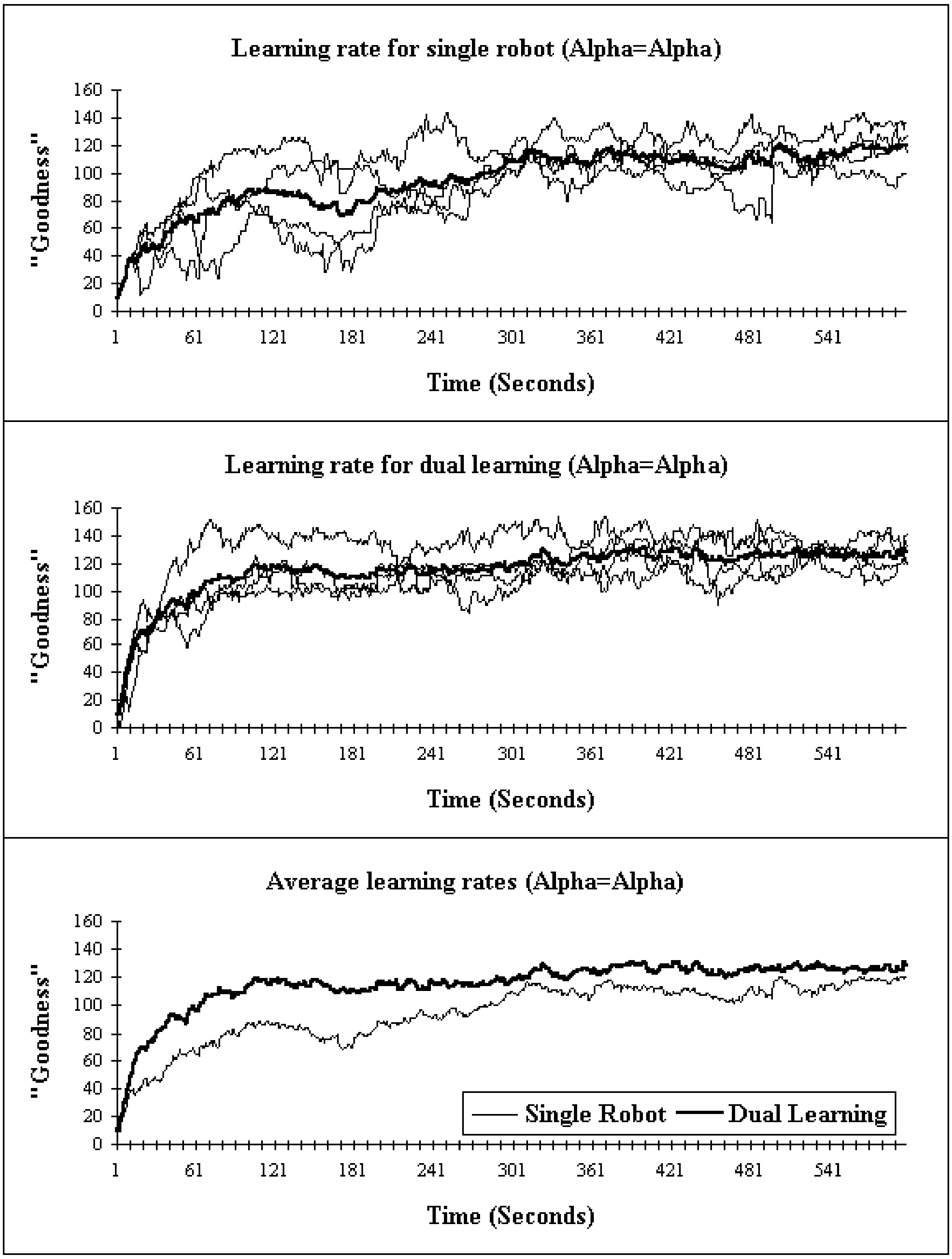

Figure 6, learning rates with twice the optimal Alpha factor.

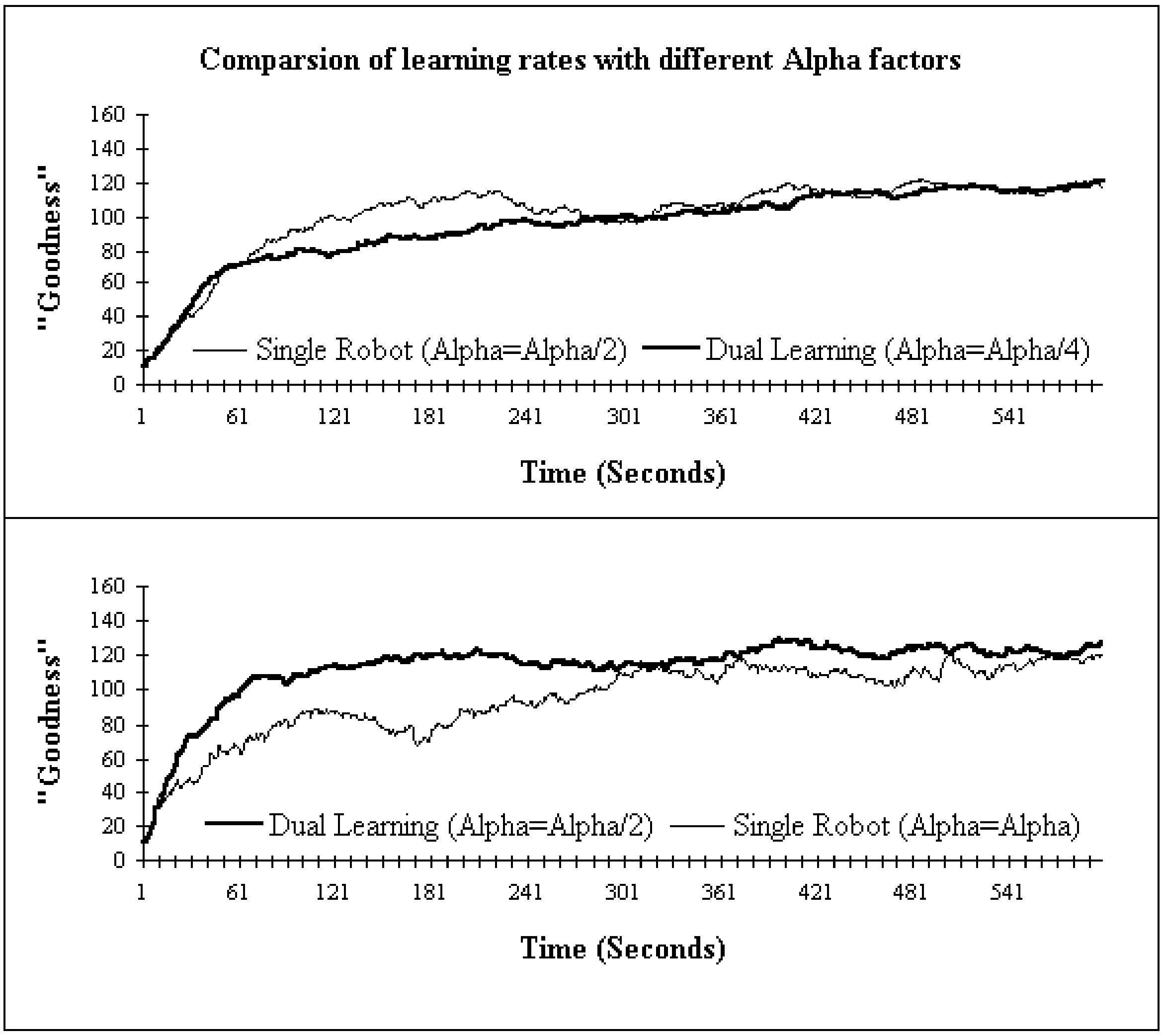

Figure 7, comparsion of learning rates with different Alpha factors.