Terrain Prediction for an Eight-Legged Robot

Stephen Urwin-Wright and David Sanders*
University of Portsmouth
Portsmouth, England PO1 3DJ

Sheng Chen
University of Southampton
Southampton, England SO17 1BJ

Received 12 April 2001; accepted 25 October 2001

Most legged robots must negotiate unknown environments with little or no terrain knowledge, as autonomous terrain mapping for robots is limited. A predictive terrain contour mapping strategy is proposed, which employs the use of a feed-forward neural network to predict the contours in environments, based on the positions of the neighboring legs. The predicted performance is better than previous implementations. © 2002 John Wiley & Sons, Inc.

1. INTRODUCTION

Roughly half the land surface of the earth is inaccessible to conventional wheeled or tracked vehicles. A walking machine that could travel where terrain difficulties make wheeled or tracked vehicles ineffective would be useful.

There are advantages of using legged robots over traditional wheeled vehicles in rough and unstructured terrain;[18] in particular, these include higher speed, better fuel economy, greater mobility, better isolation from terrain irregularity, and less environmental damage.

Robug IV is a compact and powerful general purpose teleoperated eight-legged robot. Its capabilities include plane transition between surfaces, traverse of a 0.305 m–deep pit, and autonomous omnidirectional climbing. To minimize the weight of the robot, the prototype was constructed of aluminum. *Robug IV* has the ability to carry 50 m of umbilical and have a 5 kg payload. With a weight of only 40 kg, *Robug IV* can be carried by humans.

The electromechanical design concept of the vehicle followed the same design rules and principles that were applied to *Robug III*. The design was a

*To whom all correspondence should be addressed; e-mail: david.sanders@port.ac.uk.

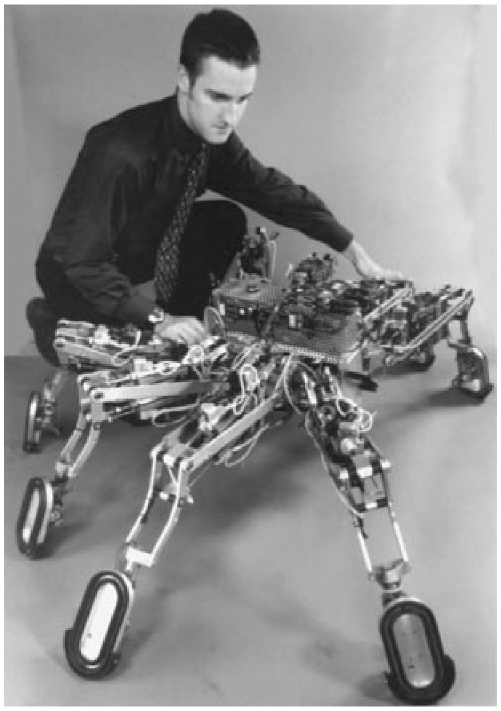

Figure 1. *Robug IV*.

"spiderlike" vehicle, comprising a central body and eight peripheral two-link legs. Each leg is 0.7 m long, consists of two links, and has four actuated points: abductor, hip, knee and ankle. The body is 0.3 m long by 0.45 m wide and 0.4 m high, and is shown in Figure 1.

The orientation of each joint in the robot leg was controlled using a double-acting pneumatic cylinder. Pneumatic actuators were used, as they are lighter and environmentally more rugged than geared electric motors.

Force control could be provided directly by the use of pneumatic actuation. Several design structures were considered, and a kinematic chain like that of the PUMA 560 robot was selected, comprising a vertical first axis of rotation and two mutually parallel horizontal axes for the second and third joints.

Pneumatic cylinders are nonlinear devices. Their behavior varied both on the small scale of a complete rod stroke and on a larger scale with the operating temperature of the system. Leaks are common in pneumatic systems, and difficult to prevent completely.

The loading on different joints in the robot legs varied widely in normal operation. This could have been due to the load being carried by the robot to the point in the walking gait sequence, or simply due to the position of the leg itself.

2. THE NEED FOR PREDICTION

Each leg was not a continuous locomotion element like a wheel. Therefore it needed to be lifted at the end of its effective stroke, returned, and placed to begin another support stroke. This creates a phasing problem, which is defined by the term "gait." A Fuzzy-Logic Adaptive Gait (FLAG) algorithm[6,11,17] was employed to determine when to lift and when to place the leg.

The FLAG algorithm was capable of navigating *Robug IV* across terrains. Problems arose because the FLAG algorithm followed the contours of the obstacles too faithfully. For example, when *Robug IV* detected a collision in the placement phase, a blind foothold search strategy was executed. In Figure 2, the algorithm deduced that its leg was in the vicinity of the lower kinematic workspace. Therefore, the course of action to take was to lift the leg higher in the Z-axis. To execute this process, a small force was exerted in the direction of the detected obstacle during lifting the foot in order to *feel* for the top surface of the obstacle.

Figure 2 shows that *Robug IV* took a number of steps before it placed its foot. A terrain mapping system, capable of accurately predicting unknown terrain, would be beneficial in assisting smooth and efficient walking motions of legged robots in unstructured environments, preventing a "juddery" action of the leg moving from A to C. This could be important when the autonomous terrain mapping accuracy of vision-based systems is reduced, as in poorly lit or smoke-filled environments. In the local range, the prediction systems may even provide more accurate information. The better the predictive strategy, the smoother *Robug IV*'s path, free of turnings and vibrations.

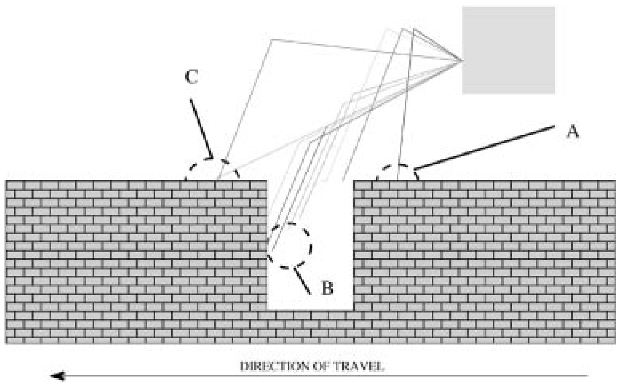

Figure 2. Obstacle.

Table I. Average Prediction Error for Current Implementation.

Terrain type	Average error (in mm)
Flat surface	14.2
Linear ramp	8.8
Set of steps	31.2
Rough terrain	24.1

3. PREDICTIVE STRATEGY IMPLEMENTED

Robug IV already had a terrain mapping system capable of predicting the Z-axis of the terrain.[5] This network was trained for 5000 epochs to predict the next step, based on data from 5 previous steps. The average prediction errors for four real terrain surfaces are shown in Table I.

The sensors used in the robot had an error after calibration of ± 6 mm. It was therefore the aim to train any artificial neural network (ANN) to be within this error. The prediction error shown in Table 1 was large compared to the sensor error. The other problem with the current prediction method was that the y-axis was not predicted.

4. IMPROVEMENT TO THE CURRENT STRATEGY

Neural networks have had a good representation in their application in control and prediction problems.[13] Their ability to capture and model information from nonlinear systems and generalize information from learned data makes them suitable for terrain prediction. Under certain conditions it may be possible to extend the suitability theories that exist in traditional control theory to systems that include neural networks. This is important if an intelligent control structure for a walking machine is going to be adopted.

The proposed solution uses the feed-forward neural network (FFNN), trained using a back-propagation[9] supervised training algorithm.

5. THE ANN

The layout of *Robug IV* is shown in Figure 3. Considering a single leg, the variables that are being predicted for each leg are shown in Figure 4.

If *Robug IV* were traversing forward in a straight line, it would be assumed that the odd legs (1, 3, 5, and 7) would encounter the same terrain as their opposite even legs (0, 2, 4, and 6) within 0.45–1.85 m (from Section 1: leg length 0.7 ($\times 2$) and width 0.45). With this in mind, it seemed logical that an implemented neural network would benefit from having some of these inputs fed into it. Therefore, prediction might only be required for the leading legs.

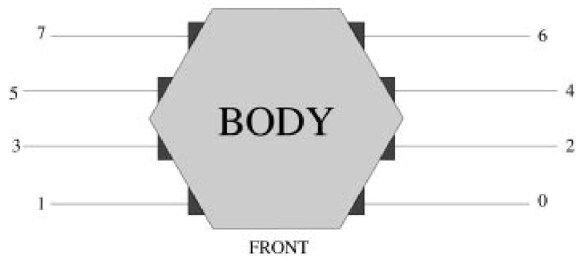

Figure 3. *Robug IV* plan.

Robug IV was built as a modular robot, primarily to overcome leg redundancy. The FLAG algorithm stated that at least three legs must be supporting the robot before a leg was lifted and placed. When there were fewer than two legs on each side, *Robug IV* went into a lockdown state to prevent damage to itself. Therefore, the number of legs on each side of the body may vary from two to four, assuming the robot is moving. If the network is designed to have all four legs providing inputs to it and two of these legs are redundant, what should these inputs be?

The constraint of available onboard processor power also had to be considered when designing the network. Each leg had four Siemens C167 microcontrollers. This limited the memory available, so the design had to be compact, compared to the power of the desktop PC used for simulation.

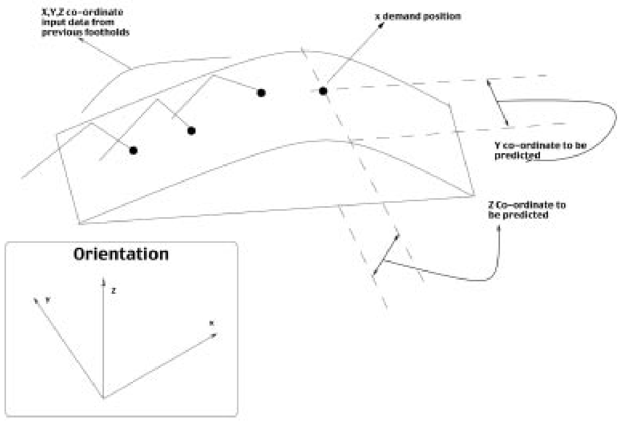
X,Y,Z co-ordinate input data from previous footholds
x demand position
Y co-ordinate to be predicted
Z Co-ordinate to be predicted
Orientation
y
z
x

Figure 4. Predicting (x, y, z).

6. SIMULATION

According to Brooks[2], simulation can be a terrible thing, convincing people that robots work which then prove unable to cope in practice, and generally confusing the issue. However, *Robug IV* was an expensive piece of equipment, capable of damaging itself. Simulation provided a safe and easy way to test different gait strategies, and reject ideas that really do not work.

There was also a lack of workspace to test all the adaptive gait strategics. *Robug IV* would probably have to walk for a considerable distance over a variety of terrains. The only real way to test *Robug IV*'s step-climbing ability is to take it to a set of steps and let it traverse them. Problems arise with gaining permission to try this and moving the equipment, the air supply, and *Robug IV* to the test site.

A simulation, called RobSim, consists of an environment with a robot walking in it. This simulator tested the FLAG algorithm incorporated into *Robug IV*. This code had been proved to emulate *Robug*, so it seemed logical to modify this code to give leg positions to be fed into the neural network.

The simulation could be static (not deal with acceleration), which simplified things. The basic environment was a surface, which the robot walked over. The robot needed to be able to find out if a given point in space was above or below the surface (i.e., whether it had made contact). The simplest way to model the surface was as a series of planar tiles.

7. TRAINING & TESTING

Essentially, training and testing were the same thing, as the training sequence is close to the testing sequence (both come from the same source). Using RobSim, data was acquired for a range of terrains. As RobSim had been proved, the mathematical calculations remained the same. The x-step that was originally used was 0.26 m[6], therefore one step was roughly 0.26 meters.

This data was then formatted into previous step, next step, training, and testing data. The total sample size was 4000 meters, roughly 15,000 steps. From this, the two sets—training and testing—were derived. These sets were then applied to a genetic FFNN. The number of inputs was then varied along with the number of hidden neurons, within the constraints mentioned in Section 5, in order to find the optimum configuration.

8. RESULTS

It was assumed that if a rule was developed for leg 0 in Figure 3, then it could be applied to the other legs (from the theory in Section 5). So leg 0 was selected to be a trial leg and then the results were implemented on the other legs.

As it was not possible for *Robug IV* to maneuver with fewer than two legs per side, *Robug IV* went into a lockdown state when this occurred; it was therefore assumed, to overcome the problem of which legs to feed into the network (Section 5), that the network would be designed with only two legs as the inputs. So the previous steps from leg 2 constituted the other input to the network.

The optimum number of input and hidden layer neurons was defined as the point when there was no more advantage in increasing them. The first value to be set was the number of input neurons. Galt[5] stated that the number of input neurons should be five and the number of hidden layers should be four, trained for 5000 epochs.

The number of previous steps that were being fed back into the network ranged from one to three. Above three, the size of the FFNN started to become high, from constraints in Section 5; below one, the FFNN became inane. The RMS prediction of the y- and z-axis plotted against the number of hidden neurons is shown in Figures 5 and 6.

From the RMS prediction error of the y-axis (Fig. 5), all the previous steps fed into the network appear in ascending order. Using more than two hidden neurons significantly reduced the error of the prediction, and using more than eight reduced the RMS of the prediction to less than 1.0 cm.

The RMS prediction of the z-axis (Fig. 6) is interesting. Feeding three previous steps into the network and having less than four hidden neurons results in the network performing worse than when two

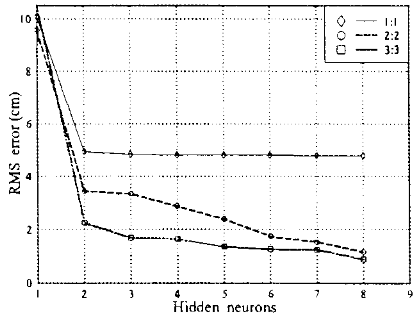

Figure 5. y-axis RMS error.

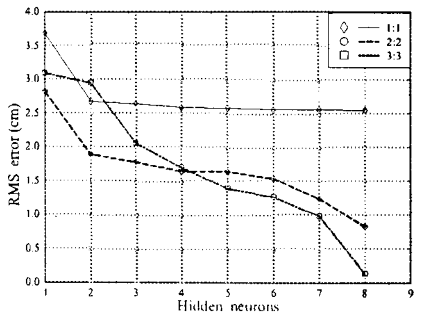

Figure 6. *z*-axis RMS error.

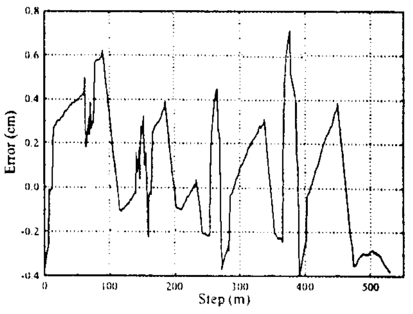

Figure 8. Prediction error of the ramp terrain.

previous steps are fed into the network. As with only using two previous steps, having more than eight hidden neurons reduced the RMS error to below 1.0 cm and even close to 0.25 cm.

Taking into consideration the error of the pneumatic cylinder once it is calibrated, which is ±6 mm (Section 3), to reduce the RMS error to within this amount would require at least eight hidden neurons and three previous steps to be fed into the network. Problems arose here, as achieving this increased the demand on the microcontroller (Section 5). So it was decided that an RMS error of 1.6 cm was tolerable, taking advantage of the fact that using two previous steps the network performs better.

Therefore, two previous steps fed were into the FFNN, with four hidden neurons. The following results show how the network performed over the four terrains.

The figures illustrating these results use error (in cm) rather than error squared. Because the error is small, squaring would make the graphs into straight lines.

8.1. Predicting the *y*-Axis

Figure 7 shows the position of the leg in the *y*-axis when *Robug IV* was walking up a ramp. The error in the prediction of this ramp is shown in cm in Figure 8; the RMS error of this is 0.450374 cm.

Figure 9 shows the position of the leg in the *y*-axis when *Robug IV* was walking across rough terrain. The error in the prediction of this terrain is shown in cm in Figure 10; the RMS error of this is 2.418234 cm.

Figure 11 shows the position of the leg in the *y*-axis when *Robug IV* was walking across smooth terrain. The error in the prediction of this terrain is shown in cm in Figure 12; the RMS error of this is 0.422875 cm.

Figure 13 shows the position of the leg in the *y*-axis when *Robug IV* was walking up a set of stairs. The error in the prediction of these stairs is shown in cm in Figure 14; the RMS error of this is 8.256384 cm.

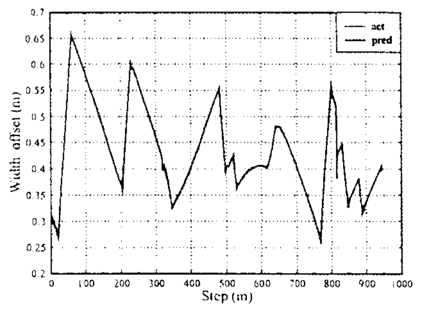

Figure 9. A rough terrain.

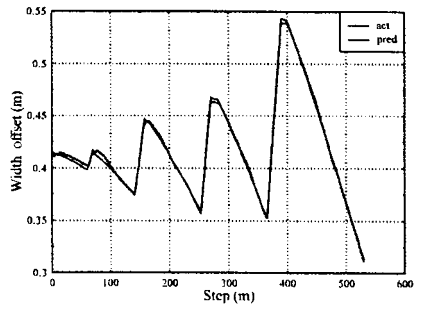

Figure 7. A ramp terrain.

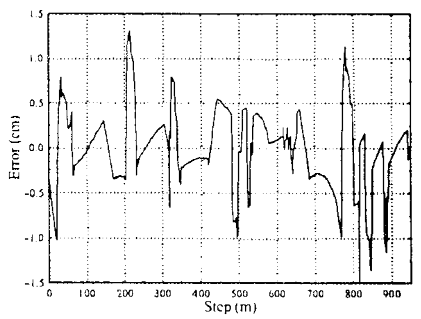

Figure 10. Prediction error of the rough terrain.

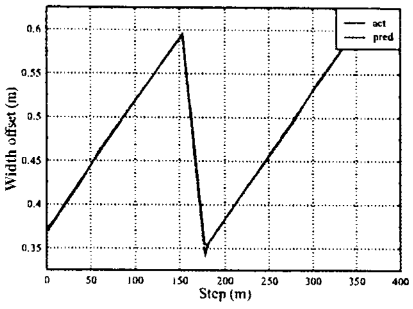

Figure 11. A smooth terrain.

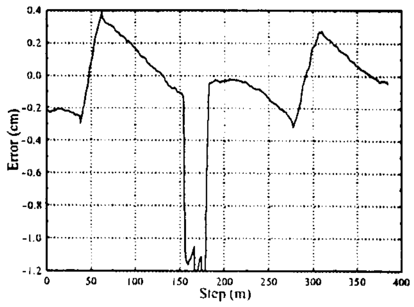

Figure 12. Prediction error of the smooth terrain.

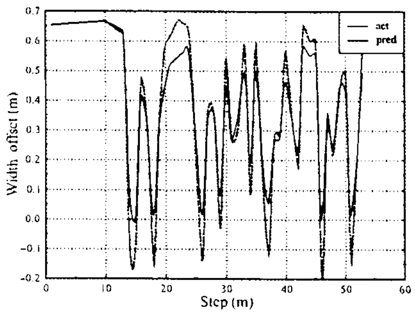

Figure 13. A stair terrain.

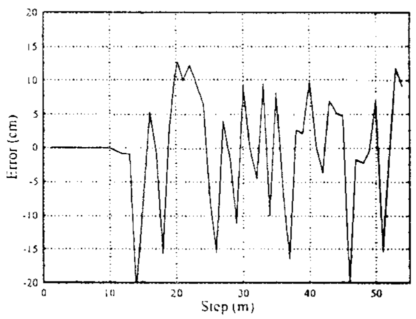

Figure 14. Prediction error of the stair terrain.

8.2. Predicting the *z*-Axis

Figure 15 shows the position of the leg in the *z*-axis when *Robug IV* was walking up a ramp. The error in the prediction of this ramp is shown in cm in Figure 16; the RMS error of this is 2.541913 cm.

Figure 17 shows the position of the leg in the *z*-axis when *Robug IV* was walking across rough terrain. The error in the prediction of this terrain is shown in cm in Figure 18. The RMS error of this is 2.541913 cm.

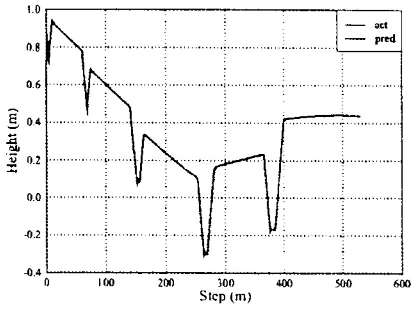

Figure 15. A ramp terrain.

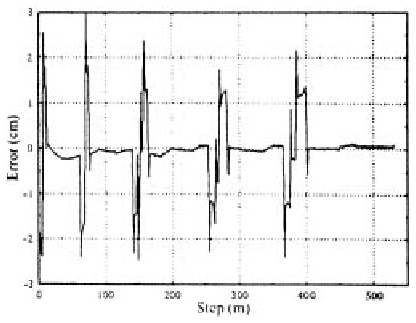

Figure 16. Prediction error of the ramp terrain.

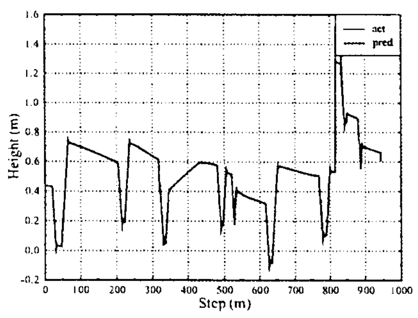

Figure 17. A rough terrain.

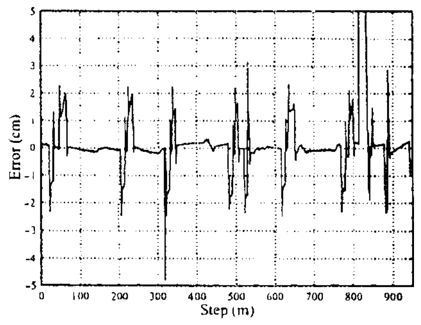

Figure 18. Prediction error of the rough terrain.

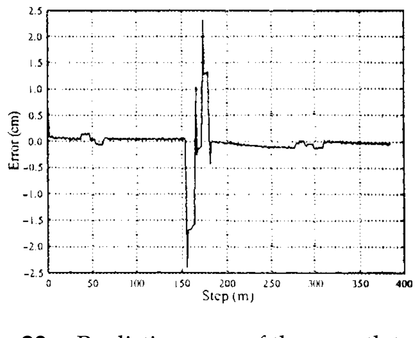

Figure 20. Prediction error of the smooth terrain.

Figure 19 shows the position of the leg in the z-axis when *Robug IV* was walking across smooth terrain. The error in the prediction of this terrain is shown in cm in Figure 20; the RMS error of this is 0.352966 cm.

Figure 21 shows the position of the leg in the z-axis when *Robug IV* was walking up a flight of stairs. The error in the prediction of these stairs is shown in cm in Figure 22; the RMS error of this is 2.541913 cm.

8.3. Learning Rate (Epochs)

From Figure 6, the two best predictions were obtain with four and eight hidden neurons. The learning rate of the three different inputs is shown in Figures 23 and 24.

Figure 23 suggests that there is no significant advantage in network learning with more than 2000 epochs, as the RMS prediction error does not improve by much. With eight hidden neurons (Fig. 24) the network learning levels off at around 2600 epochs. As only four hidden neurons were decided upon (Section 8), the optimum epoch training sequence would have only 2000 epochs.

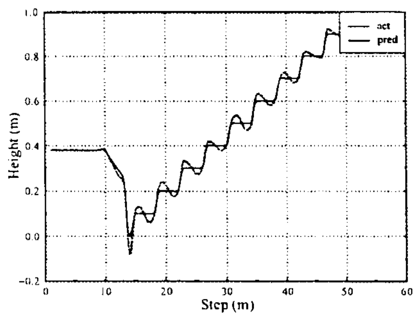

Figure 21. A stair terrain.

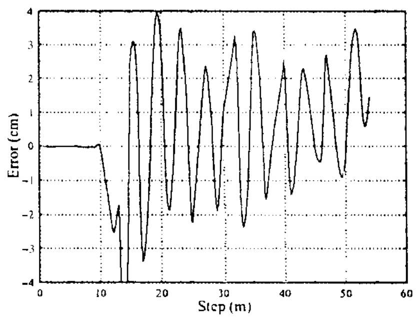

Figure 22. Prediction error of the stair terrain.

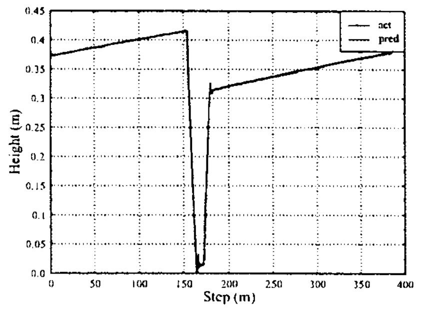

Figure 19. A smooth terrain.

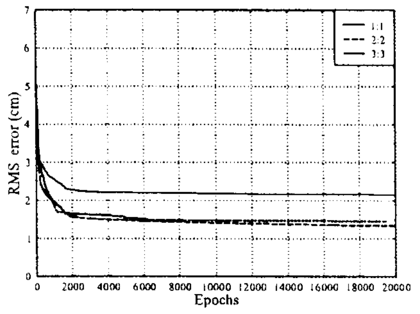

Figure 23. Four hidden neurons.

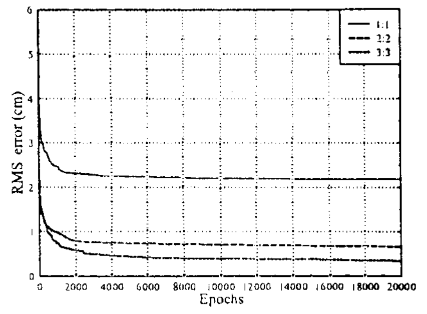

Figure 24. Eight hidden neurons.

9. CONCLUSION

A summarized comparison of the results for the four terrains is given in Table II. The terrain prediction does not predict the y-axis location of the robot's leg, so there is no comparison for the y-axis prediction; a summary of results is given in Table III.

Assuming that the terrains tested in the current implementation (see Section 3 and Table 1) are the same as those tested with the new FFNN, then the new network performs better than the currently implemented network on a rough, smooth, and stair surface. The ramp terrain (Fig. 15) is not completely linear, and this may account for the poorer result in the z-axis. It was also assumed that "average error" quoted in the previous results was actually the RMS error.

Table II. z-axis Prediction Comparisons.

Terrain type	RMS error (cm) of new prediction	Average error (cm) of current prediction
Ramp	2.54	0.88
Rough	1.50	2.41
Smooth	0.35	1.42
Stair	2.13	3.12

Table III. y-axis Prediction Error.

Terrain type	RMS prediction error (cm)
Ramp	0.45
Rough	2.41
Smooth	0.42
Stair	8.26

The number of epochs is significantly smaller. The current implementation uses 5000 epochs; the new implementation uses less than half that, at 2000 epochs. This reduces implementation time.

REFERENCES

1. P.J. Corke and B. Armstrong-Hélovry, A search for consensus among model parameters reported for the PUMA 560 robot, IEEE, 1994, pp. 1608–1613.
2. R. Brooks, Intelligence without reason. MIT AI lab memo 1(1293) www.leglab.ai.mit.edu (1991).
3. T. White, B. Luk, D.S. Cooke, and N. Hewer, Implementation of modularity in Robug IV—preliminary results, Professional Engineering Publishing, Bury St. Edmunds, UK, 1999.
4. K. Warwick, G. Irwin, and K. Hunt, Neural Network Applications in Control, IEEE control engineering series, 53. Academic Press, Exeter, England 1995.
5. S. Galt and B. Luk, Predictive terrain contour mapping for a legged robot, Fifth Int Conf on Artificial Neural Networks, 1997, pp. 129–133.
6. S. Galt, B.L. Luk, S. Chen, and R.H. Istepanian, Intelligent walking gait generation for legged robots, Int J Smart Engg Syst Des (2001).
7. David E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison Wesley, Reading, MA 1988.
8. A. Jones, Neural networks and genetic algorithms for prediction and control of dynamic systems, Technical report, Imperial College, London, 1995.
9. M. Lehtokangas, Modeling with constructive backpropagation. Neural Networks 12:(4-5) (1999), 701–716.
10. T.M. Lehtokangas, Additive neural networks and periodic patterns, Neural Networks 12:(4-5) (1999), 617–626.
11. B.L. Luk, S. Galt, and S. Chen, Using genetic algorithms to establish efficient walking gaits for an eight-legged robot, Int J Syst Sci (2000).
12. O. Firschein and M. Fischler, The eye, the brain, and the computer, Addison Wesley, UK and USA 1987.
13. A. Pipe, M. Randall, and A. Winfield, Adaptive neural control of walking robots with guaranteed stability, CLAWAR Conference, Professional Engineering Publishing (1999), 111–122.
14. S. Perantonis, N. Ampazis, and J. Taylor, Dynamics of multilayer networks in the civility of temporary minima, Neural Networks 12:(1) (1999), 43–54.
15. S. Nair, Modeling and simulation of a six-legged walking robot power system, Simulation 58:(3) (1992), 185–195.
16. W.S. Kendall, O.E. Barndorff-Nielson, and J.L. Jensen, Networks and chaos—statistical and probabilistic aspects, Chapman & Hall, 1993.
17. S. Chen, R. Istepanian, S. Galt, and B. Luk, Intelligent walking gait generation for legged robots, Professional Engineering Publishing, Bury St. Edmunds, UK 1999.
18. Shin-Min Song and Waldron, Machines that walk: The adaptive suspension vehicle. MIT Press, Cambridge, MA 1988.
19. M.L. Vaughn, Derivation of the multilayer perception weight constraints for direct network interpretation and knowledge discovery, Neural Networks 12:(9) (1999), 1259–1271.