Robbie the Running Robot: A Distributed Learning System

Kynan Eng, Alec P Robertson and Deane R Blackman
Department of Mechanical Engineering
Monash University, Clayton
Victoria, Australia

Abstract

A unique distributed control system has been developed to allow a hexapod robot to learn how to walk. It models competing directives which drive the robot to walk as quickly as possible without falling over given minimal initial knowledge. The system architecture features a highly decentralised arrangement of cooperating modules with no central controller. This has many advantages over conventional control systems, including robustness, scope for incremental development and the ability to facilitate accurate pre-prototype simulation. Accelerated learning was achieved by using the robot's intrinsic knowledge of its own symmetry to infer additional information about its surroundings. This was found to improve the robot's performance, particularly at the start of the learning process when information was limited. These concepts can be applied to autonomous robots for use in deep sea and other inhospitable environments.

Introduction

Conventional mechatronic systems tend to be highly specific with the steps required to perform a task built-in to the machine's design. This makes the system efficient at performing its intended function in optimal conditions, but inefficient at dealing with unexpected situations. The main reason that most mechatronic systems deal poorly with unexpected events is that the predefined information in the system is too specific. A better strategy in some applications is to allow the system to independently determine the best way to perform the task, given the ability to interact with its environment and analyse the success of its actions. Of course, providing such meta-knowledge to a mechatronic system must have practical limits defined by the learning time required and the cost of failed trials during the learning process.

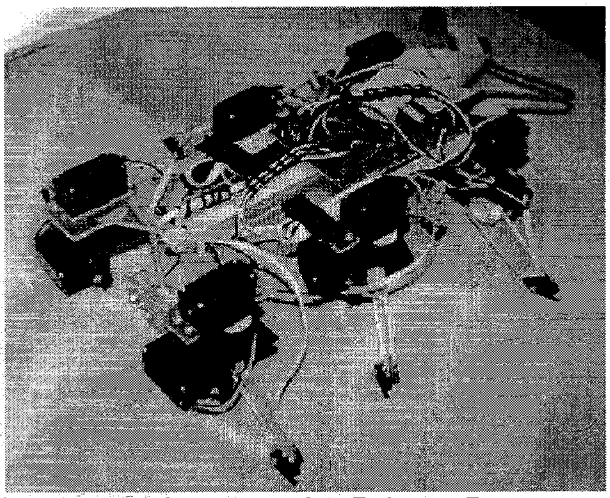
Fig 1. Robbie the Running Robot at Rest

This paper describes the development of a six-legged robot that learns how to walk as quickly as possible with minimal initial knowledge regarding appropriate walking actions or terrain type. The main focus of the project was on the learning processes involved rather than the details of dynamic real-time robot control systems. Given the means to control each of its legs, the robot was designed to work out sequences of leg movements that allow it to walk without falling over. To counteract the possibility that the robot could avoid falling simply by standing still, the incentive to learn how to walk was given by an in-built "curiosity". The competing directives affecting the robot can be summarised as follows:

Directive	**Means to achieve directive**
Walk	Repeat a leg movement sequence forever
Avoid pain	Don't repeat "painful" leg movements
Improve	Try new random leg movement sequences
Compete	Repeat the last sequence of leg movements a little faster

System Architecture

Earlier studies in this field have been undertaken by Brooks [1,2] with *Genghis*, a walking robot based on his Subsumption Architecture. His work was heavily process based, where each module in the architecture was responsible for some action: raising legs, lower legs, etc.

The distributed learning architecture developed for Robbie (Figure 2) directly models the competing directives which drive the robot's actions. It is layered in a similar way to the ISO networking reference model. The lowest layer is the physical realisation of the robot (sensors and actuators), while the higher layers deal with progressively more abstract learning concepts. The directives can be seen as the vertical alignment of different blocks in the architecture.

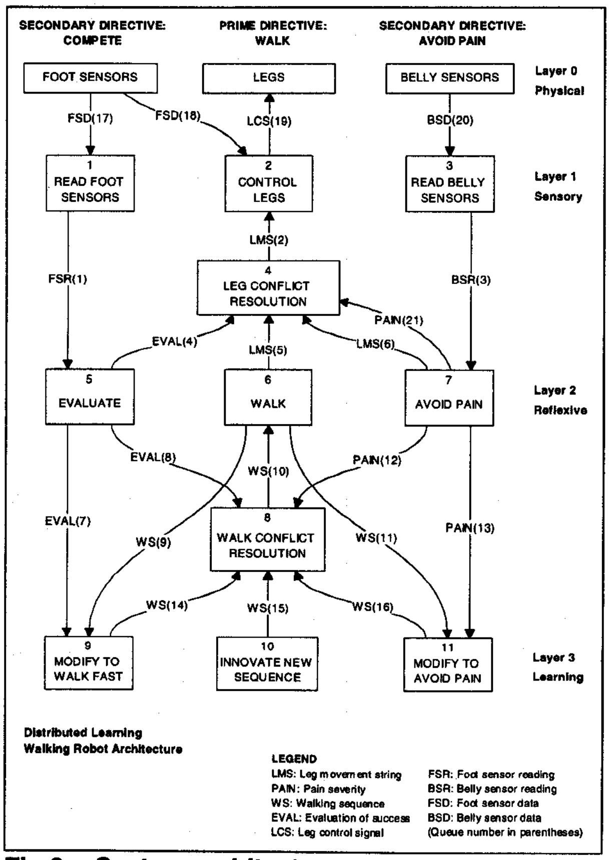

Fig 2. System architecture

Each module is run as a separate independent process. Asynchronous message queues provide for interaction between modules. The architecture is organised so that feedback paths progress directly up the layers. Feed forward paths, on the other hand, are regulated by special conflict resolution blocks. These blocks accept multiple inputs from one or both adjacent layers and produce a single feed-forward output, ie. the block makes a decision about the most appropriate input or combination of inputs to feed forward.

The interactions between the components are what causes the system as a whole to "learn". By designing some of the complexity of the system at the architectural level, the individual components can be made much simpler than might otherwise be possible in a learning system.

The distributed learning architecture presented here represents one way of dealing with the given learning problem. When defined in this way, the distributed learning architecture is simply a high-level algorithm. Thus the algorithms used for the individual components are obviously affected by the overlying architecture.

Conceivably, if the learning architecture was sufficiently refined with each component being defined as a separate learning architecture then eventually each component would become trivial to implement. However, this is not the case, as this would imply that at some stage there was a trivial way of learning an arbitrary piece of knowledge. In practice, the learning architecture needs to be just complex enough to learn the task at hand — at least one of the components must contain some in-built knowledge about the task. Extra complexity lends extra adaptability, but it must be contained in order to maintain practicality. This is the difference between learning in the pure sense (learning with no predefined knowledge) and learning in a practical application (some predefined knowledge without compromising adaptability).

Analogues in Other Systems

Many recent developments in learning systems use other (often natural) systems as a basis, for example the human immune system [3]. While no existing system was consciously used to develop Robbie's learning architecture, two analogies suggested themselves when the topology of the design had been finalised:

- The structure can be seen as a simplified version of the human nervous system, in which the physical assets correspond to the nerve endings and the message queues represent the transmission of messages along the nerves.
- The striking similarity of the system architecture to the ISO networking reference model reflects the similarities between hiding complexity in learning and remote data access.

Learning Algorithms

Several novel algorithms were developed to enable the robot to learn how to walk. They deal with two potentially conflicting goals: avoiding falling over and maximising walking speed. The algorithms were designed with practical performance in mind rather than learning in the pure sense, however prior knowledge of the learning problem was kept to a minimum to ensure adaptability over a wide variety of different terrains.

Avoiding Falling Over

A rule-based strategy was used to enable the robot to learn how to avoid falling over. It trials different combinations of leg movements and records the results of these trials. In this way, after a certain number of trials it has amassed sufficient knowledge of the terrain to move its legs in ways that prevent it from falling over.

The speed of the learning process is improved by using the symmetry of the robot to generate tentative rules. Dubbed Symmetry Learning, this technique allows the robot to access (up to) four times as much data as is available directly from its sensors from a single trial.

The method by which the extra data is generated is illustrated in Figure 3. When the robot receives data about the result of a trial, it is stored with an "actual" priority. The data is then mirrored about the horizontal and vertical planes to generate three more rules with priorities of "high", "medium" and "low". The lower priorities indicate that a rule was generated by the use of symmetry and should be considered as a tentative rule only. Any new information received in a later trial that conflicts with an existing rule is only stored if it is of an equal or higher priority than the existing rule. Using this technique, the robot learns (increases the number of rules available) very quickly by inferring its own tentative rules from the available data. Even if the tentative rules are incorrect, later trials will eventually weed out the invalid rules.

Two different methods were used to modify the robot's actions in response to data gained from the stored rules. The first of these involves randomly altering the sequence of the leg movement times where the robot found leg configurations that caused it to fall over. The second method notes which leg actions most likely caused the robot to fall over and ensures that the same actions are not repeated on the next attempt. Neither of these methods require the robot to learn what needs to be done; it was deemed that from a practical viewpoint the simplicity of these algorithms did not warrant extra learning software.

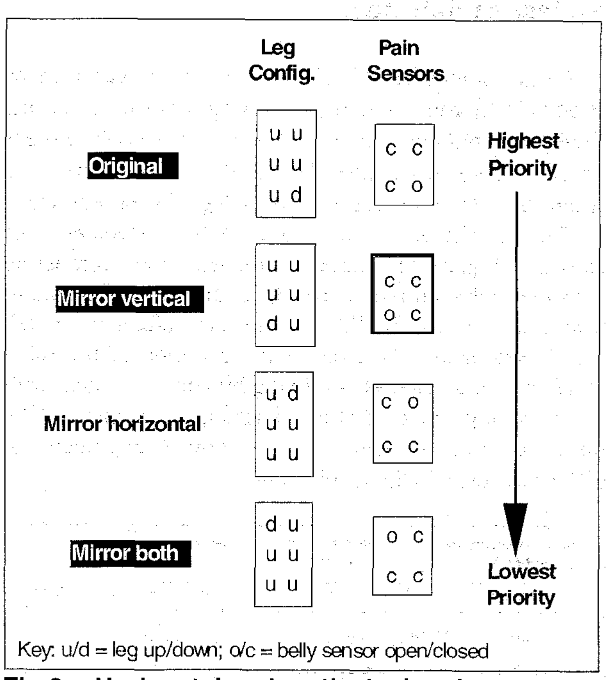

Fig 3. Horizontal and vertical mirroring

Maximising Walking Speed

The approaches to maximising the robot's walking speed fall into two broad categories:

- A simple proportional reduction scaling transformation which instructs the robot to do what it did before, but a little bit faster.
- Non-linear transformations which seek to normalise and eliminate redundant periods (times when none of the robot's legs are moving).

The non-linear transformations are useful for rapid performance improvements, while the proportional scaling is used for fine-tuning performance when the robot is already walking near its maximum potential speed.

Note that although these approaches were found to work well, neither of them involve the robot actually learning the appropriate actions to take. The knowledge of what to do in order to walk faster is predefined in the algorithms used.

Another strategy has been proposed but not yet implemented where the robot creates lists of rule "cells" for each leg which contain statistics about the robot's performance over the previous time period. New and improved walking sequences are generated by combining the rule cells of most merit for each leg.

Simulation Software

A simulation of the robot was developed to test each learning algorithm as it was implemented. The simulation display contains a representation of the system architecture and allows easy tracing of data and process states. It deals exclusively with the learning components of the system; the control of the robot was dealt with separately.

Robot Controller Architecture

The main aim in the design of the robot controller was to keep the cost and development time required to an absolute minimum. An architectural diagram is shown in Figure 4.

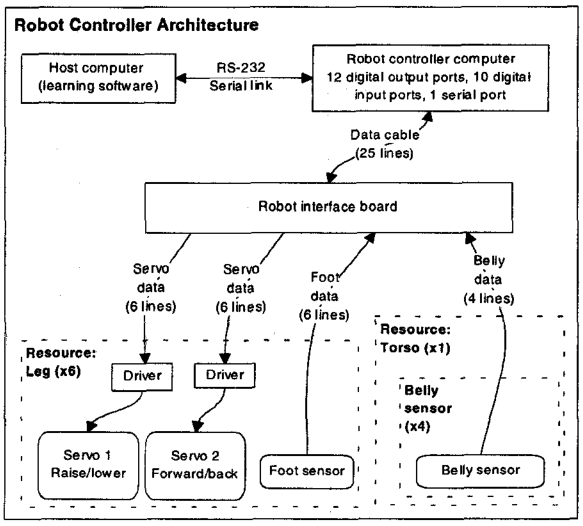

Fig 4. Robot controller architecture

Both the host computer and robot controller computers are standard PCs running MS-DOS. The host PC uses a single RS-232 serial port connected to a controller PC which interfaces to the robot via two standard 8-bit parallel ports.

The interface protocol between the host and controller computers is a simple fixed packet length serial protocol. No error checking or packet retransmission facilities were implemented due to time constraints.

Robot Controller Software

The software for controlling the robot performs the following tasks in real time:

- Pulse Width Modulation (PWM) position-based control of twelve servo motors
- Continuous polling of six foot sensor switches and four belly sensor switches
- Interrupt-driven RS-232 serial communications with the learning software computer
- Logging of system operations to disk
- Calculation of performance statistics
- Display of current status on the computer screen
- Processing of user keyboard commands

The core of the controller is a timer interrupt routine which times the pulses for controlling the servo motors. The pulse width is between about 1 and 2 ms for each servo motor, with a new pulse being required for each servo motor every 25-30 ms. After each motor has been pulsed once, the sensors are polled and their status updated. This gives a polling rate of about 40 Hz, which is more than adequate for this application.

At the same time as this, the interrupt-driven serial communications library receives and sends packets when instructed to by the learning software computer. Both the communications queue and the keyboard are polled periodically and actions taken depending on any new inputs received.

Although the learning software deals only with walking forwards, the controller software also provides for backwards walking and turning left and right.

Robot Construction

The physical robot is about 300 mm long and has a total mass of about 1 kg. It is constructed mainly of CNC milled 3 mm acrylic. There are three major structural components:

- The chassis is a rectangular sheet of acrylic which provides a platform for the servo motors and interface PCB. Large sections are cut out of the rectangular sheet to minimise weight. A narrow acrylic spine fixed with double-sided adhesive tape runs the length of the robot to provide additional stiffness.
- Servo mounts provide a connection between the fore-aft and raise-lower servos on each leg.
- The legs are single pieces of 3 mm thick acrylic directly mounted to the leg drive servo–motor.

Standard RC servo motors are used to actuate each of the two-degree-of-freedom legs. Each of the six legs has a microswitch mounted in the foot. Microswitches mounted on the torso enable the robot to sense when it has fallen down.

Testing Performed

The robot was tested in offline mode, using the simulation as well as online with the real robot. The simulated mode tests revealed how the algorithms and the robot (simulation) performed under deterministic conditions, while the field tests showed how the algorithms and the physical robot performed in reality.

Real robot testing was conducted on the following types of terrain:

- Flat ground (carpet)
- Inclined wooden plane (6 degrees and 14 degrees), uphill and downhill (see Figure 5)
- Corrugated plastic roofing with a peak-to-peak height of about 20 mm and a wavelength of about 75 mm

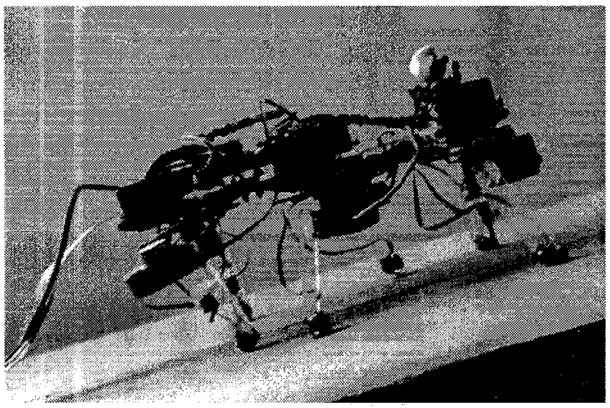

Fig 5. Robbie climbing an inclined plane

The following data was collected for each of three separate test runs for each algorithm and the results averaged:

- The position (and hence the velocity) of the robot at regular time intervals
- The number of times the robot fell down during the run and at what times the falls occurred

The robot was also run in tripod gait on each type of terrain, yielding the maximum possible speed of the robot as a reference value.

Results and Analysis: The R-Rating

To provide a consistent way of comparing the performance of different algorithms on different terrains, an index called the R-rating was developed. Its formulation is as follows:

$$\varsigma = \frac{n_{falls\ \mathrm{tripod}}}{v_{av\ \mathrm{tripod}}}$$

$$R_i = 100\frac{v_{av}}{\mathrm{n}_{\mathrm{falls}}}\cdot\varsigma \qquad \mathrm{i} = 1..\mathrm{N}$$

where: $\mathrm{n}_{\mathrm{falls\ tripod}}$ – number of falls per metre for tripod gait

$v_{av\ \mathrm{tripod}}$ – average velocity over course for tripod gait

ς - scaing factor

$\mathrm{n}_{\mathrm{falls}}$ – number of falls per metre

v_{av} – average velocity over course

N - number of tests performed

R_{i} - performance index (R - rating) of test i

The results are normalised to the results obtained from manually running the robot in the tripod gait on flat ground, which is assigned an R-rating of 100, assuming one fall per metre (the robot must fall to learn). The robot walking in tripod gait on flat ground managed a sustained speed of 8.2 cm/s.

It was found that the pain avoidance learning algorithms aided by symmetry and the nonlinear performance improvement transformations gave the best performance on all terrain types. The learned gaits gave a maximum speed of about 2 cm/s (R-Rating = 13) for flat and corrugated terrain, and about 1 cm/s (R-Rating = 2) on the 6 degree uphill incline. The 14 degree incline proved too steep to climb at all. On flat ground, the best non–learning algorithm registered an R–Rating of 4 indicating that there is a three–fold performance difference using the learning algorithms. The robot was able to attain maximum speed when walking without falling over in less than 60 seconds.

Comparison With Other Robots

Robbie's performance is most readily compared with the *Genghis* robot developed by Rodney Brooks at MIT. Both use pain feedback information and are designed to walk as quickly as possible. Genghis is fully autonomous and has some extra features such as variable leg lifting and force balancing to allow it to walk over uneven terrain more easily. Both take around 2 minutes to reach a maximum speed of between 2 and 3 cm/s. However, because Robbie was constructed very rapidly and with limited financial resources, its performance was almost certainly well below its full potential. It was also lacking in the navigational and force balancing features. Overcoming these limitations should be a relatively straightforward development process and should result in a significant increase in performance.

Applications

There are obvious applications for the architecture and algorithms developed for Robbie to autonomous walking robots in inhospitable environments and/or in situations where the terrain is unknown. The learning components can be readily extended to include extra sensors, path planning and navigational control. The economic and rapid construction of Robbie (total cost about AUD400) also opens up possible commercial applications in education and entertainment.

Limitations, Conclusions and Further Work

The distributed learning control system and the design process used were found to have many advantages during the course of Robbie's construction, including:

- The problem was able to be decomposed easily into small, manageable pieces.
- The architectural layout was suggested naturally by the competing directives in the problem definition. The horizontally layered structure bound by vertical directives imposed a degree of order on the design without imposing excessive restrictions.
- A simulation was created which reflected the performance of the robot remarkably well long before the real robot was built, enabling off-line simulation and testing of algorithms.
- Each module could be implemented relatively independently of the others, and improved incrementally where deemed necessary.
- The overall system proved to be quite robust; the robot walked successfully even when major bugs were present in some of the modules.

Several novel algorithms were also developed which addressed the specific problems associated with robot walking. The first was Symmetry Learning, which uses the robot's own symmetry to speed up the creation of rules to walk without falling over. Several non-learning algorithms were also developed to provide rapid walking speed improvements by removing redundant and wasted time from a given robot walking sequence without compromising stability.

The main limitation of the distributed learning architecture is that it is specific to the problem of learning how to walk. It may be possible to define a more general distributed learning architecture (perhaps using a different architectural topology) that is applicable to a wider set of problems in the same way that neural networks are. Further research can be directed towards development of additional levels of abstraction to broaden the scope of application of the architecture and algorithms.

Acknowledgment

Thanks go to Associate Professor Deane R Blackman and Dr Bijan Shirinzadeh, both of Monash University, for their valuable guidance in the course of the design and construction of Robbie the Running Robot. Other people who have provided assistance include Keith Hoy, Peter Koch (equipment and supplies for construction) and André Spierings (photography of Robbie).

References

1. R A Brooks: A Robust Layered Control System for a Mobile Robot, Proc Robotics Research: Third International Symposium, Gouvieux, 1985, pub. IEEE, pp 365-372.
2. R A Brooks: A Robot that Walks; Emergent Behaviours from a Carefully Evolved Network, Proc International Conference on Robotics and Automation, Arizona, pub. IEEE, 1989, pp 692-694.
3. A Ishiguro, S Ichikawa, Y Uchikawa: A Gait Acquisition of a 6-Legged Robot Using Immune Networks, Proc International Conference on Intelligent Robots and Systems, Location unknown, 1992, pub. IEEE, pp 1034-1041.