Sensing and Direction in Locomotion Learning with a Random Morphology Robot

Karl Hedman  David Persson  Per Skoglund  Dan Wiklund  Krister Wolff  Peter Nordin
Complex Systems Group, Department of Physical Resource Theory
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
E-mail: {wolff, nordin}@fy.chalmers.se

Abstract

We describe the first instance in sensing and direction with a learning Random Morphology robot. Using sensing and genetic programming, it learns to locomote itself in different directions and by superposition of different solution candidates it can follow an arbitrary path. The robot is assembled of seven standard R/C servo motors, arbitrarily interconnected and an IR sensor that is used to determine distances to nearby objects, thus detect its movements. The architecture of the robot is two dimensional. The robot is trained to move in a direction specified by the distance to the nearest wall in the surroundings. The fitness function reward movement in the given direction with an additional bonus for moving any of the actuators at all.

1 INTRODUCTION

In traditional robot control programming, an internal model of the system is derived and the inverse kinematics can thus be calculated. The trajectory for movement between given points in the working area of the robot is then calculated from the inverse kinematics. Even though this still is a very common approach, we propose for several reason the concept of genetic programming for control programming of so called bio-inspired robots as e.g. a random morphology robot, with many dependent actuators [Dittrich et al, 1998]. The traditional geometric approach to robot control, based on modelling of the robot and derivation of leg trajectories, is computationally expensive and requires fine tuning of several parameters in the equations describing the inverse kinematics [Nolfi & Floreano, 2000]. Conventional industrial robots are designed in such a way that a model can be easily derived, but for the development of bio-inspired robots, this is not a primary design principle. Thus, a model of the system is very hard to derive or to complex so that model-based calculations of actuator commands require to much time for reactive tasks [Dittrich et al, 1998] and [Langdon & Nordin, 2001]. For a robot that is conceived to operate in an actual human living environment, it is impossible for the programmer to consider all eventualities in advance. The robot is therefore required to have an adaptation mechanism that is able to cope with unexpected situations [Andersson et al, 2000], [Nordin et al, 1998], [Banzhaf et al, 1997], [Nordin & Banzhaf, 1997] and [Olmer et al, 1995].

Our approach to evolve locomotion controllers for a Random Morphology-robot differs from previous work in that we use a sensor system to dedicate a desired moving direction to the robot. The problem was split up in several subtasks. First, a robot of arbitrary architecture was constructed and assembled. Second, a suitable environment was designed for the robot experiment. Finally, our system was ready for the experiments in genetic programming. Specified by the shortest distance to the nearest wall, the robot was set to move in a given direction. In the next stage, it should be able to move in as many directions as possible and finally, as the robot has learned to move in a number of different directions, it could be controlled by letting different solutions master the robot in different situations. Thus, it can follow an arbitrary path.

2 ROBOT PLATFORM

The Random Morphology robot is composed of seven standard off-the-shelf R/C servo motors as actuators, which are interconnected arbitrarily in a two dimensional plane. To prevent the robot from tearing itself apart during the experiments, the movements of the
robot’s various parts are limited to two dimensions. Given this restriction, the robot cannot lift any part of it self from the ground, thus it has to shuffle its way forward. Further, the connection bars should not intervene with the movement of the actuators. This means that the these are constructed in such a way that each actuator can move freely in its whole workspace. At a certain location of the robot, that we call the robot’s head, an IR distance sensor is mounted.

2.1 ACTUATORS

The Random Morphology-robot is assembled with standard off-the-shelf R/C servo motors as actuators. It is composed of seven servos which are interconnected arbitrarily in a two-dimensional plane. This kind of servo has an integrated closed loop position control circuit which detects the pulse-code modulated signal that emanates from the controller board for commanding the servo to a given position [Jones et al, 1999]. In this implementation, each servo is commanded to a given position by the robot control program by addressing it an integer value within the interval \( [0, 255] \). Each device contains of a complete servo system including DC motor, gear box, feedback device, servo control circuitry and drive circuit. We use servomotors with an output torque of 3.3 kgcm at 4.8 V. The interconnections are made up of metal bars of varying length. These can be arbitrarily connected, by thin metal joints, to both the wheels or the bodies of the actuators.

2.2 SENSOR SYSTEM

To accurately measure distances to nearby objects, a near-infrared PSD range sensor is used. It consists of an IR emitter and a Position Sensitive Detector in a single package. The principle of this sensor is based on triangulation, which means that the sensor is relatively insensitive to the texture and color of the object at which it is pointed. The emitter, placed below the detector in the package, illuminates a small spot on an obstacle with modulated IR light. A lens forms an image of the spot on the active element at the back of the detector. The output of the detector element is a function of the position on which the image is falling [Jones et al, 1999]. Within the range of about eight to 40 centimeters distance to the object, a value of sufficient accuracy (resolution < 1 millimeter) is produced.

2.3 CONTROLLER BOARD

To control the servo motors and process sensory data, we use the EyeBot MK3\(^1\) micro controller. The EyeBot MK3 consists of a 32-bit micro controller board with a graphics display, four push buttons for user input and a serial communications interface. The robot control programs are developed on a host computer. After a cross- compilation they are downloaded, in executable code format, to the EyeBot controller. The serial line is then only used for uploading experimental data to the host computer since all signal processing is carried out on the EyeBot controller itself.

2.4 FIRMWARE AND SOFTWARE

The EyeBot MK3 controller board is running an operating system which consists of two main parts, the Robot Basic I/O System, RoBIOS and the Hardware Description Table, HDT. The same RoBIOS is shared by all hardware configurations of a robot controlled by an EyeBot MK3, but the HDT differs to account for different sensors or actuators connected to the actual hardware. Each actuator has a unique workspace according to its position on the robot [Ziegler et al, 2001]. The individual workspace for all actuators are specified in the HDT file by setting suitable values. In order to control the movements of a limb, the partial movements of all involved joints must be coordinated and synchronized to get the desired motion. For this reason, a servo locomotion module has been developed [Wolff & Nordin, 2001]. All robot control programs that we have developed here are implemented in C language.

\(^1\)http://www.ee.uwa.edu.au/ braunl/eyebot/
3 GENETIC PROGRAMMING SYSTEM

The learning method is a conventional steady-state linear GP algorithm, running on the robots EyeBot MK3 computer. The genetic programming system is implemented as a register machine [Nordin, 1997]. The operators work on the three registers, with the initial values set to 100, 0 and 1, respectively. Each individual consists of a string of integer valued numbers, where every number corresponds to an instruction. Furthermore, the operations available are listed in table 1 below.

Table 1: Function set of the GP system.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>ADD(a,b), SUB(a,b), MUL(a,b), SINE(a)</td>
</tr>
<tr>
<td>Protected division</td>
<td>DIV(a,b)</td>
</tr>
<tr>
<td>Exec. delay</td>
<td>WAIT(t)</td>
</tr>
<tr>
<td>Servo command</td>
<td>SETSERVO0(a), SETSERVO1(a), ..., SETSERVO6(a)</td>
</tr>
</tbody>
</table>

WAIT(t) delays the execution of the program for t milliseconds (maximum 1000 ms) and SETSERVO0(a)-SETSERVO6(a) commands any servo to a value within the range \(0, 255\). The output from the program to the robot is the positions of the servos.

Table 2: GP system parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Tournaments</td>
<td>500-700</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Max Start Length</td>
<td>50</td>
</tr>
<tr>
<td>Max Length</td>
<td>400</td>
</tr>
<tr>
<td>Prob. of Mutation</td>
<td>0.75</td>
</tr>
<tr>
<td>Prob. of Homol. Crossover</td>
<td>0.5</td>
</tr>
<tr>
<td>Prob. of Non Homol. Crossover</td>
<td>0.5</td>
</tr>
<tr>
<td>Overall Prob. of Crossover</td>
<td>1</td>
</tr>
</tbody>
</table>

3.1 INITIALIZATION

An initial population is randomly created. The individuals in the population consists of a sequence of integer valued numbers in a certain interval, where each number corresponds to a predefined instruction. However individuals are of random length, there is a maximum length which the individuals are not allowed to exceed. The decoding of an integer into instruction is handled by the register machine.

3.2 TOURNAMENT SELECTION

The method of steady-state tournament selection is used to select individuals to breed. This implies that there are no well-defined generations but a successive change of the population [Banzhaf et al, 1998] and [Nordin, 1997]. Four different individuals are randomly picked from the population and get to compete against each other in pairs. Their performance is evaluated using a fitness function and the two winners get to breed. The offspring, produced by recombination and mutation, replace the two loser individuals in the population. The selection, evaluation and reproduction phases of the evolutionary algorithm is then repeated until the maximum number of trials is reached. The number of tournaments a certain individual can be selected for is unrestricted.

3.3 REPRODUCTION

For reproduction both mutation and crossover is used. Crossover is in this case two-point and can be done in two ways, homologous and non-homologous. The homologous takes two points that are the same on each individual and swaps the information between the points while the non-homologous takes two different points on each parent and swaps the information in-between. Note that the non-homologous crossover changes the length of the individual. To prevent the individuals from growing out of proportion, a maximum length is introduced. If an offspring turns out to be longer than the maximum length, the instructions after the maximum length are simply dropped. There is equal probability between homologous and non-homologous crossover.

Mutation randomly takes a point on one of the children and inserts a randomly selected instruction there. For every tournament there is a certain probability that any of the offspring is mutated.

4 EXPERIMENTAL SECTION

Here, we describe the setup and the physical environment of the experiments. The robot is positioned in an enclosed arena, a short distance from one of the walls. The goal for the robot is to locomote towards that wall as straight forward and fast as possible. A number of runs where made under these conditions. The objective was to find individuals able to move in different directions. In this way the robot can be remote con-
trolled to move in a maze using different individuals.

4.1 EXPERIMENTAL SETUP

The robot and the experimental environment is shown in the figure below. The Random Morphology robot is directly controlled by the EyeBot MK3 micro controller. The controller is connected, via the serial communication interface, to a standard desktop PC running Windows 98. The EyeBot MK3 and the robot are powered from separate outputs on a regular DC power supply.

During evolution, all signal processing is carried out on the EyeBot controller, hence the serial communication line is then only used for uploading experimental data to the host computer. The best fitness is transferred to the desktop PC every tenth tournament. The best individual and the best length is uploaded every 50th tournament. All data are stored on the host computer for further evaluation.

![Figure 2: The robot in its experimental environment.](image)

The robot is placed in the middle area of the arena and in the corner closest to the camera, one can see the EyeBot MK3 controller.

4.2 FITNESS EVALUATION

The primarily criteria when designing the fitness function is that individuals with the desired features is favored over less suited individuals. The fitness function promote individuals moving in the right direction and punish movement in other directions. Thus a suitable fitness function is merely the difference between the measured distance before and after each individual with the condition that fitness function should be maximized. However, in order to promote movement the individual achieves a bonus of tenth of a point each time it sets a servo to a different value than it was previously set to. There is a maximal bonus of 8 points for moving servos.

4.3 EVOLUTION

The evolution begins with an initialization of all actuators. The head is positioned and the first tournament starts. In order for the robot to find the way, it is uses a pre-defined function that positions the PSD sensor in the direction where the distance to the closest wall is the shortest. Four individuals are selected to be in the tournament and each individual is allowed to control the robot for four cycles. The robot’s distance from the wall is measured before the individual starts and after it has finished. The acquired data and the number of moves the robot makes are passed to the fitness function. In each tournament there are four individuals selected. Their performance is evaluated, using the fitness function as described above and the two winners get to breed. The offspring, produced by recombination and mutation, replace the two loser individuals in the population. The sensor is repositioned every fifth tournament.

The evolution can terminate in one out of three ways. Evolution stops if the program reaches the end of the loop, if the experimenter manually pushes a button on the EyeBot MK3 controller, or if the measured distance to the wall has become too small. In the case of when the robot has reached the wall, evolution can continue by manually selecting to stop the evaluation and reposition the robot. The reason for this is to prevent good individuals from dying during unsupervised evolution, when the robot might end up in a situation where it cannot move anyway.

5 RESULTS

The overall result of the experiment is that the GP algorithm is able to produce individuals that can locomote the robot in different directions. Thus, the robot is really able to follow an arbitrary path. Some of the individuals show a rather complex behavior since they use all of the actuators in a clearly correlated manner. The fastest and best individual however, use only three actuators in a short cycle that manage to locomote the Random Morphology robot across the arena.

Further in this section we present statistical data from the experiments.

5.1 FITNESS AND LENGTH

The fitness of the best evolved individual was measured to 151. Figure 3 clearly show how fitness is getting better and better as evolution proceed. A number
of runs were made under the same conditions. The results from some of them are selected and presented in figure 4 below. It shows that some of the runs resulted in interesting individuals able to solve the task. There were also made an experiment, considerably longer than all the other runs. Figure 5 below presents the statistics from that run. This experiment shows how the complexity (length) of the individuals rather quickly reaches the maximum value of 400. When this happens the information left over is simply dropped. In this case the crossover is not working properly and evolution is seriously harmed.

Figure 3: The fitness of the best run (top). Best individual length of the best run (bottom).

Figure 4: Fitness of some selected runs, also resulting in interesting individuals (top). Length of best individuals (bottom).

Figure 5: The fitness of the longest run (top). The best length of the longest run (bottom).

5.2 DISTANCE AND VELOCITY

In order to view how the best individual locomote the robot, the distance to the wall was measured continuously during a number of cycles. The velocity was also calculated from these data. The maximum velocity measured was 16 mm/cycle and the average velocity was calculated to 5.8 mm/cycle, where a cycle is defined to take 5 seconds. See figure 6 and figure 7.

Figure 6: The distance to the wall decreasing.

6 DISCUSSION

When building a Random Morphology robot, a lot of practical issues has to be dealt with. In our first attempt of designing the robot, the actuators were arbitrarily interconnected such that the robot could move in three dimensions. This was a flawed construction since it resulted in very little movement and the robot
broke apart very easily. At this point, it was realized
that a new approach was needed and the robot was
completely redesigned. This time we aimed for a two-
dimensional design, with all the actuators orientated
with their wheels facing up. With this two dimensional
design, the robot could now move freely without break-
ing apart. However the robot could move its different
parts relative each other, it did not move much rela-
tive the ground, due to the low friction to the surface.
To compensate for this, ballast weights were placed on
top of the actuators.
Evolving robot controller programs with real physical
hardware is a challenging task. Since the evaluation
of the individuals is to be made in real time, the tour-
naments are very time consuming. Evolved programs
are showing good performance in some ways, yet not
in others and different patterns of movements were de-
veloped.
Some improvements of our system is still to be done,
such as implement an evolutionary system with indi-
vidual programs, or solution candidates, that can move
the robot in any direction. That is not the case now,
hence several different individuals is used, one for each
single direction.

7 SUMMARY AND CONCLUSIONS

We describe the first instance in sensing and direc-
tion with a learning Random Morphology robot. The
robot is composed of seven standard off-the-shelf R/C
servomotors as actuators, which are interconnected ar-
bitrarily in a two dimensional plane. To accurately
measure distances to nearby objects, a near-infrared
PSD range sensor is used and thus movements of the
robot is detected. To control the servo motors and
process sensory data, we use the EyeBot MK3 micro
controller.
Using sensing and Genetic Programming, it learns to
locomote itself in different directions and by super-
position of different solution candidates it can follow
an arbitrary path. The learning method is a con-
ventional steady-state linear GP algorithm, running
on the robots computer. The GP system is imple-
mented as a register machine. The method of steady-
state tournament selection is used to select individu-
als to breed and for reproduction both mutation and
crossover is used. The fitness function promote indi-
viduals moving in the right direction and punish move-
ment in other directions.
The previously described experiments in genetic pro-
gramming, performed with our Random Morphology
robot, showed that the GP algorithm is able to pro-
duce individuals that can locomote the robot in differ-
ent directions. Thus, the robot is really able to follow
an arbitrary path.

References

P. Dittrich, A. Burgel and W. Banzhaf (1998). Learn-
ing to move a robot with random morphology. In Phil
Husbands and Jean Arcady Meyer, editors, First Eu-
ropean Workshop on Evolutionary Robotics (pp. 165-
178). Berlin: Springer-Verlag.
S. Nolfi and D. Floreano (2000). Evolutionary
Robotics: The Biology, Intelligence, and Technology of
Self-Organizing Machines. Massachusetts: The MIT
Press.
W. B. Langdon and J. P. Nordin (2001). Evolving
Hand-Eye Coordination for a Humanoid Robot with
Machine Code Genetic Programming. In Proceeding
of EuroGP 2001, (pp 313-324). Lake Como, Milan,
Italy: Springer Verlag.
B. Andersson, P. Svensson, J. P. Nordin and M. G.
Nordahl (2000). On-line Evolution of Control for a
Four-Legged Robot Using Genetic Programming. In
Stefano Cagnoni, Riccardo Poli, George D. Smith,
David Corne, Martin Oates, Emma Hart, Pier Luca
Lanzi, Egbert Jan Willem, Yun Li, Ben Paechter and
Terence C. Fogarty, editors, Real-World Applications
of Evolutionary Computing, volume 1803 of LNCS,
(pp. 319-326). Edinburgh, Scotland: Springer Verlag.
Evolution of World Model for a Miniature Robot
using Genetic Programming. International Journal
of Robotics and Autonomous systems, North-Holland,
Amsterdam.
Generating Adaptive Behavior using Function Regres-
sion with Genetic Programming and a Real Robot. In
Proceedings of the Second International Conference on
Genetic Programming. Stanford University, USA.
Method to Evolve Behavior and to control a Mini-
ture Robot in Real Time with Genetic Programming.
The International Journal of Adaptive Behavior, (5)
(pp. 107 - 140). USA: The MIT Press.
Evolving Real-Time Behavioral Modules for a Robot
with Genetic Programming. In Proceeding of ISRAM
Mobile Robots: Inspiration to Implementation. Mas-
sachusetts: AK Peters.
J. Ziegler, K. Wolff, J. P. Nordin and W. Banzhaf

