SELF-EVOLVING CONTROLLERS FOR ROBOT SYSTEMS

Mark Foyle

MEng Computer Science & Cybernetics, siu00mf@reading.ac.uk

ABSTRACT

Research into robotic control and learning methods is an area of great interest within the scientific community.

This paper describes the development of a novel self-evolving control system for controlling robots. In particular, it discusses the application of the system to a robotic arm and the series of experiments that were carried out.

In these experiments, the control system enabled the arm to successfully adapt such that it was able to throw a ball into a target cup, and even show a noticeable improvement in the distance obtained when throwing the ball as far as it could.

The control system used a particularly novel approach, in which Fourier series were used to generate control signals defining the motion of the robot, and these signals could then be adjusted by modifying the coefficients of the Fourier series.

Finally, the paper discusses how the application of this approach to robot control shows great potential, and how the system may be applied to other similar robots, and even biped walking robots.

1. INTRODUCTION

Robot learning is defined as the establishment of a capability within a robot agent that was not available at the design stage [1].

This project involved the study of computer-controlled robots, and how ‘intelligence’, could be introduced in the form of a self-evolving controller, allowing the robot to adapt its operation in order to fulfil an objective.

The use of robots for performing tedious, repetitive tasks is well known. A robot can be programmed to perform a certain task, and can then perform the task over and over again in exactly the same way. The problem with programming robots in such a way is that this approach results in non-adaptability. Thus if the environment in which the robot is operating changes, the robot is unable to adapt to this environment change and still carry out its task.

In [2], the authors describe a method of teaching a robot to throw a ball, based upon previously obtained data and a simulation. Indeed, a lot of learning robot algorithms are based upon simulations, however, the problem with this is that the simulation has to be incredibly accurate to be useful, and even then, the real robot may perform differently.

![Figure 1. Robotic Arm and Interface Circuit](image)

In this project, a robotic manipulator arm (as shown in Figure 1) was used for the development of a control system and learning algorithm, with the goal of getting the robotic arm to throw a ping-pong ball into a cup placed in front of the robot. Thus the aim is to get the arm to learn how to throw a ball a specific distance, and in a certain direction.

The arm is controlled by 5 R/C servos, therefore allowing a large variation of movement. These servos can be controlled by connecting them through an interface circuit to a computer, and then using a piece of software on the computer to send signals to the robot.

This paper describes the robot control software and the novel learning methods used so that the robot could be taught how to throw the ball successfully.
2. DESIGN

By its very nature, the project consists of two main components – the control system for interfacing with and controlling the robot, and the learning algorithm, which adjusts the way in which the robot moves in order to fulfill a goal.

The control system used in the project involved the physical hardware interface between the computer and the robot, and also the associated control software.

The control system was mainly the work of the author, Mark Foyle, whilst David Barrett [3] conducted the majority of the work on the development of the learning algorithm.

The following sections describe in detail the hardware and software aspects of the control system, and an overview of the learning algorithm.

3. CONTROL HARDWARE

The hardware interface between the robotic arm and the computer constitutes one half of the control system. Since the movement of the arm is controlled by the servos, and it is necessary to control the positions and movements of the servos from the computer, an interfacing circuit between the computer and the card had to be introduced. Whilst this could have been constructed specifically for this project, it was noted that a servo controller card, which can communicate with a computer serial port, existed and had the functionality required for the project.

3.1. Interface Circuit

The controller card, manufactured by Milford Instruments [4], provides a way of controlling the position of up to 8 servos, by sending signals from a PC serial port to the card, using a specified protocol. In order to move one of the servos connected to the card, a string of 3 bytes is sent, as defined in Table 1.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Synchronisation Byte</td>
<td>255</td>
</tr>
<tr>
<td>Byte 2</td>
<td>Servo Number</td>
<td>0.7</td>
</tr>
<tr>
<td>Byte 3</td>
<td>Position</td>
<td>0.255</td>
</tr>
</tbody>
</table>

The communication speed for the card could be set to either 2400 or 9600 Baud, but for the purpose of this project it was set to 9600, allowing quicker communication with the board, and hence more control over the robot’s movements.

3.2. Robotic Arm

The robotic arm consists of 5 servos, each of which can be controlled through the hardware interface circuit.

Since each servo has a range of 180 degrees, and there are 256 possible positions that can be set, each servo therefore has an approximate resolution of 0.7 degrees per position unit, and therefore provides a fair amount of control over the movement of the robot as a whole.

4. CONTROL SOFTWARE

The other major component of the control system was the control software, which comprises of the software required to communicate with and operate the servo controller card, and also the signal generation system.

The chosen implementation language was Object Pascal, as used within the Delphi programming environment. This was chosen in preference to other available languages, due to its simplicity and the speed of the development this allows.

The communication aspect involved the implementation of the communication protocol that is used to send commands to the servo controller board. Thus a way of sending commands to the computer’s serial port was required. This was implemented using a freeware component [5] that provides an interface for the serial port, allowing bytes of data to be written into a buffer in memory, and then sent down the serial port to the attached servo controller card.

The signal generation system was one of the novel features of the control software, and was implemented so that it was possible to produce different signals based upon a set of parameters. This was necessary due to the fact that it would be easier to get the learning algorithm to output a set of numbers (parameters) than it would be to output a set of functions describing the movement of each servo.

It was decided that the signal generation system should be biologically inspired, and thus produce smooth sinusoidal signals, in a way similar to Central Pattern Generators (CPGs). These CPGs essentially are a group of neurons that produce rhythmic patterns, without any kind of sensory input.

In order to build a system that would be able to both produce these sinusoidal signals and also allow these signals to be specified by a set of parameters, research was done into Fourier theory [6], wavelets and Walsh functions.

Fourier theory specifies that it is possible to represent any periodic signal as a sum of an infinite number of sines and cosines. Therefore, by using Fourier series to build up the control signals, it is likely that it would be possible even using just a few sine waves to build up different waveforms.
A function was written as part of the signal generation system that constructed the Fourier series signals for each of the servos, and could calculate the output value of each of the signals for any given point in time. Due to the nature of the signals, the output for the signals at any point in time is usually a real number. Since the servo has 256 positions, all real numbers produced had to be mapped to one of these 256 positions. This was done by taking the integer value of the output, and sending this to the servos, through the communication software.

In order to ensure that this value would still remain in the range of 0 to 255, each of the coefficients were bounded, to ensure that the output would not normally be outside of this range. In the very rare case where this might be so, then the value would be clamped to the nearest boundary value, so if a value of 256.8 was produced from the signal, this would be converted to a value of 255.

The boundaries set for the coefficients were that each of the coefficients for the servos were in the range -64 to 64, and the T variable was in the range 0.5 to 4.5. These boundaries were set based upon observations made during testing.

During the signal generation, the actual signal generated is only between 0 and 0.5T as opposed to 0 and T. This is due to the fact that the segment of the signal from 0.5T to T is identical to the segment preceding it, except that it has been inverted. This has no practical use in this case, since it simply causes the servos to perform the same motion but backwards.

In addition, part of the graphical interface of the software produced graphs of the control signals, so that they could be seen graphically, and aid with testing that the whole system was working correctly.

One other important feature that was implemented was a database feature that stored the coefficients used for every throw. When a throw has been made, the software prompts for the ‘output’ value from the robot’s throw. In the case of throwing the ball into a cup, this represents the distance that the ball lands from the cup, although in any general situation, it represents a measure of ‘goodness’ of the output, where a low value represents a good, positive outcome. This value is then stored with the coefficients in the database, so that every experiment was logged. This meant that the learning algorithm would be able to study a history of the throws made, which aids the derivation of new coefficients.

5. LEARNING ALGORITHM

The learning algorithm constituted the other main part of the project. This part of the project was mainly conducted by David Barrett [3], although a brief overview of the learning algorithm will be supplied here.

As mentioned previously, the learning algorithm is required to manipulate the coefficients that are used by
the signal generation system, in order to ensure that the robot system achieves its goal.

Effectively, the robot can be thought of as a function. Initiating its inputs (the Fourier coefficients) to different values results in different outputs. Obviously a core consideration is how this output of the robot is measured.

Since the robot arm is being used to throw a ball, the output of the system is related to how far the ball is thrown. As an example, the straight-line distance from the robot to the place where the ball first hits the floor can be used. The only problem here is that a higher output value (bigger distance), is the goal for the robot, however most optimisation functions work by minimising the output value. This was not a problem though, since the distance was simply scaled and inverted so that a higher distance corresponded to a lower output value.

The chosen optimisation function that would be used within the learning algorithm was the Simplex function. Thus the simplex function would work within the learning algorithm to minimise the output of the robot function (after the distance had been converted as described above).

The robot can be thought of as a function. Equation 2 shows how this function can be represented, based on the factors involved.

\[ O = F_R[e, F_S(a_1, a_2, a_3, b_1, ..., e_2, e_3, T)] \]  \hspace{1cm} (2)

O represents the output of the function, the value that the learning algorithm is trying to minimise. \( F_R \) represents the actual function of the robot, which is constructed from \( F_S \), which is the output signals from the signal generation system, and _ represents the other aspects of the system, such as its physical makeup, friction, noise in the system, etc.

The learning algorithm was implemented in Matlab, due to its support of matrix multiplication, which is widely used in the algorithm. In addition, several extra functions were written allowing the control system software to interface and interact with the learning algorithm, despite the different programming languages used.

A more detailed account of the learning algorithm may be found in [3] written by David Barrett.

6. TESTING & RESULTS

Once the control system and the learning algorithm had been completed, the first tests could be carried out to see how effective the overall system was.

6.1. Testing of the Control System

Initially a few simple tests were done to ensure that the control system was operating correctly. This was done by generating random sets of coefficients, and then generating the control signals for the robot. By comparing the graphical on-screen representation of the signals with the motion of the robot, it was possible to confirm that the control system was indeed working correctly.

6.2. Testing of the whole system

Two significant types of experiment were performed, involving the robot throwing the ball. However, before these could be carried out, a set of random throws had to be performed, so that some starting points for the learning algorithm could be found. This involved generating random coefficients, which were then used to throw the ball. Those sets of coefficients that resulted in a good throw (above 50cm) were then used as seed values for the learning algorithm when the experiments were carried out.

The first experiment involved the robot learning to throw the ball as far as possible. This required measuring the distance from the base of the robot to the place where the ball first hit the ground. As mentioned previously, in this situation a larger distance is better, and therefore in order to get the learning algorithm to maximise the throwing distance, the actual distance needs to be manipulated before it is fed into the learning algorithm. It was discovered through various trials that the best method of doing it was by using equation 3 below:

\[ \text{Output} = \frac{400}{\text{Dist.} + 0.01} \] \hspace{1cm} (3)

Several runs of this experiment were tried, each with different starting points, and the learning algorithm modified the coefficients such that the throws improved, and the ball travelled further. After the success of these trials, it was decided it would be interesting to try and get the robot to throw the ball at a target.

The second experiment was slightly more complex, and involved the robot trying to throw the ball at a target. A teacup, with containing tissue paper to stop the ball bouncing out, was placed in front of the robot at a distance of 50cm. The algorithm was then started, and the ball thrown. The straight-line distance from the centre of the teacup to the place where the ball first hit the ground was then measured and entered directly into the learning algorithm as a distance in cm. This produced a measure of how effective the coefficients were.

As the algorithm executed, this process was repeated, with different coefficients, as generated by the learning process. Figure 2 below shows how the distance that the ball landed from the cup changed over time.
As it can be seen, the graph shows a trend of the distance from the ball to the cup reducing, therefore indicating an improvement in performance.

The experiment was run many times, using different starting points for the learning algorithm, and in all cases, the distance between the ball and cup could be seen to be decreasing.

Based upon these results, it was therefore concluded that the learning algorithm was operating correctly, and it was successfully ‘teaching’ the arm how to throw the ball in a better way such that it would be able to get the ball to land within the target, and therefore satisfying the goal of the problem.

Figure 3 and Figure 4 show a comparison of the control signals that were used in the first throw of the experiment (with random starting coefficients) with the control signals from the final throw (with the improved coefficients) in which the ball was successfully landing in the cup.

It can be seen from these graphs that the control signals changed fairly significantly, highlighting the ability of the system to be able to vary the control signals to produce significantly different movements.

7. FURTHER WORK

Following the work that has been to date, it would now be possible to use different robots with the system, due to the way in which the system has been designed.

Currently, a bipedal robot (known as ‘T-Rex’) is being developed consisting of 2 legs with knee joints and a tail. The robot features 5 servos for controlling its movement, and will therefore work with the control system with no modification (the robotic arm also has 5 servos), other than that the signals will be repeated to ensure a constant motion.

As with the robotic arm, a goal needs to be defined for the walking robot in order for the learning algorithm to work correctly. Since the robot will be trying to walk, a good goal for it would be to walk as far as it can. Thus coefficients can be used to generate control signals (as before), which will then make the robot move. The distance it walks will then be used to judge how good the set of coefficients are. Obviously if the robot does not move or falls over, then this will be a bad set of coefficients, and so the learning algorithm will change them to a set of coefficients more likely to yield in a walking gait.

Once these experiments have been carried out, the possibility of installing an ultra-sonic system to the robot can be considered, such that the speed at which the robot is moving can be monitored. This can then be used instead of the distance the robot moves as a measure of how ‘good’ the coefficients are. This method would result in less human interaction being needed and thus could potentially speed up the learning process.

Finally, it would be interesting to experiment with other robots, to see if the system can be used to ‘teach’
other robots to perform a certain action. This may require further work on the system (for example to use more servos), but the resultant could potentially be a system that can be used to aid any robot in adapting to a certain task.

8. CONCLUSION

This paper has discussed the development of a novel control system and learning algorithm, which have been applied to a robotic arm, and yielded positive results. Both the hardware and software have been studied, and the justifications behind the decisions made have been discussed.

The results show that it was possible to get the robot to adapt in a way that enabled it to solve two different problems, without the robot being aware of the context of the problem. This capability to solve different problems suggests that it may be possible to get the robot to perform other tasks, such as more complex types of throw.

The success of the experiments suggest that the use of Fourier series in the signal generation system was appropriate, since the control system was able to make the robot move in significantly different ways during experimentation. In addition, the use of Fourier series allows for periodic signals to be produced, therefore allowing the robot to perform a repetitive task.

In conclusion, the methods employed provide a solid foundation for further development, suggesting that other applications of the system may be possible.

Acknowledgements: The author would like to acknowledge the work and assistance of his project partner, David Barrett, with whom many ideas have been shared. The author would also like to express his gratitude to Dr. William Harwin, the project supervisor, whose assistance and guidance has been crucial in the success and completion of the project. In addition, the author would also like to thank Dr. Peter Kyberd for his input at the start of the project.

9. REFERENCES