Fuzzy Associative Memory for Humanoid Robot Joint Control

Doug Turk, Damien Kee, Chris Myatt, Gordon Wyeth
School of Information Technology and Electrical Engineering
University of Queensland, St Lucia, Australia
{turk, kee, wyeth}@itee.uq.edu.au

Abstract

Traditional approaches to joint control required accurate modelling of the system dynamic of the plant in question. Fuzzy Associative Memory (FAM) control schemes allow adequate control without a model of the system to be controlled. This paper presents a FAM based joint controller implemented on a humanoid robot. An empirically tuned PI velocity control loop is augmented with this feedforward FAM, with considerable reduction in joint position error achieved online and with minimal additional computational overhead.

1 Introduction

Research into control techniques for highly non-linear and systems with unknown or difficult to model dynamics is becoming more and more prevalent. Learning systems that are capable of providing adequate control of highly complex systems are quickly gaining acceptance as suitable control techniques.

In the field of humanoid robotics, learning systems are appealing for two main reasons: their ability to generate control without accurate system characterization information, and the parallels with the biological world. Humanoid robots are typically multi-jointed systems, with each joint subject to complex and varying loads as the robot moves about.

Designing controllers that maintain a high level of tracking and stability under a range of load conditions is challenging. Using traditional model-based control, the problem is addressed by calculating the forward model of the torques on the joints and applying an appropriate compensating signal.

However, humanoid robots are difficult to model mathematically. Hence analytically finding feedforward dynamics for model based control can be both a complicated and time consuming process. Additionally, contact with the unpredictable loads from the real world and human interaction further complicates the modelling problem.

The motions performed by a humanoid robot are typically cyclic, such as walking and grasping. The loads experienced by each joint are consequently cyclic, and as such, feedforward controllers may be used to great effect to predict and compensate the expected loads.

Adaptive control techniques such as Fuzzy Logic and Associative Memories can be used to implement this feedforward component without explicitly modelling the system dynamics.

This paper presents the use of a Fuzzy Associative Memory (FAM) as a feedforward addition to a traditional control schema. This component provides an additive compensating signal, effectively predicting the known disturbances of a cyclic motion.

1.1 Previous Research

The area of adaptive control has been widely researched and in particular, the use of learning systems in adaptive joint control of under-modelled systems has been previously investigated. Three main methods are used: neural networks, fuzzy logic systems, and genetic algorithms [Commuri and Lewis, 1996; Kee, 2002; Si *et al.*, 1999]. Of these methods, fuzzy associative memory is generally the most suited to on-line learning.

Collins [2003] outlines the use of a Trajectory Error Learning (TEL) schema, whereby the measured error of the system is learned by a neural network feedforward block. In a poorly tuned system, consisting of a wheelchair robot, the initial error is significant, and the compensating signal is heavily updated every iteration. As the system learns, the measured error is reduced and the compensating signal undergoes fewer modifications. Kee [2002] adapted this TEL for use in the joint controllers of a humanoid robot. Both methods employ the use of a Cerebellar Modelled Articulated Controller (CMAC) as the learning component to the system [Kee and Wyeth, 2002].

An alternative to the CMAC is the use of a Fuzzy Associative Memory as a feedforward control element for adaptive joint control. Several researchers [Commuri and Lewis, 1996; Pan and Woo, 2000] show methodologies that develop appropriate control for an *n*-degree-of-freedom planar robot arm. Both teams apply their methodology to a simulated two degree of freedom arm with successful results.

Si, Zhang and Tang [1999] built on Pan and Woo [2000], and augmented a PD controller with a Fuzzy Controller on a two-degree-of-freedom planar robotic arm.

The Fuzzy Adaptive Controller was then trained to compensate for the gravity disturbance experienced by the tool point. They used a Genetic Algorithm to learn the membership functions parameters which are traditionally determined by an expert human. The GA was used to determine the optimal membership width of a set of triangle membership functions to control a simulated inverted pendulum on a cart.

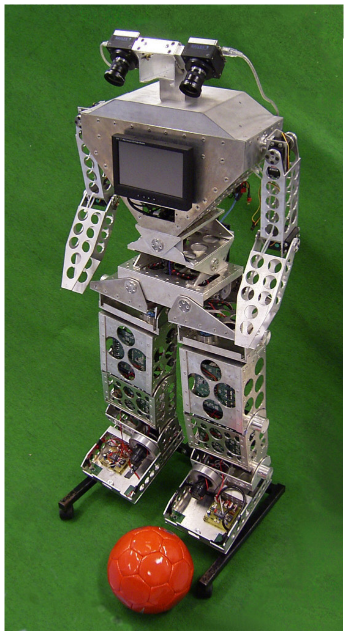
1.2 Paper Outline

Section 2 presents the experimental setup in the form of the GuRoo humanoid robot and the associated simulator. Section 3 outlines the FAM as used in this research, including the parameters of the membership functions and association tables.

Results obtained from the experimental system with both the uncompensated and compensated architecture are outlined in Section 4. These results include experiments with changing system dynamics and the impact co-evolutionary learning systems. Section 5 draws conclusions based on these results.

2 GuRoo Project

The GuRoo is a humanoid robot with 21 degrees of freedom (Figure 1). It stands 1.2m tall and weighs approximately 35kg with onboard computation and power. In addition, the project uses a graphical simulator which accurately models the dynamics of the multi degree of freedom robot.



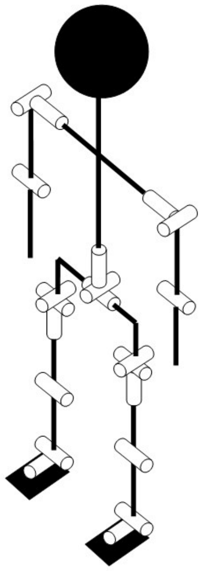


Figure 1 GuRoo Robot and location of degrees of freedom

2.1 The Robot

The joints of the robot consist of 15 high-powered DC motors to actuate the legs and spine, as well as six smaller servo motors to drive arm movements.

These motors are controlled by a distributed computing system, comprising six dedicated motor control boards and a central computer. The central computer is a mini-ITX motherboard with a 1 GHz CPU running Windows XP (Figure 2).

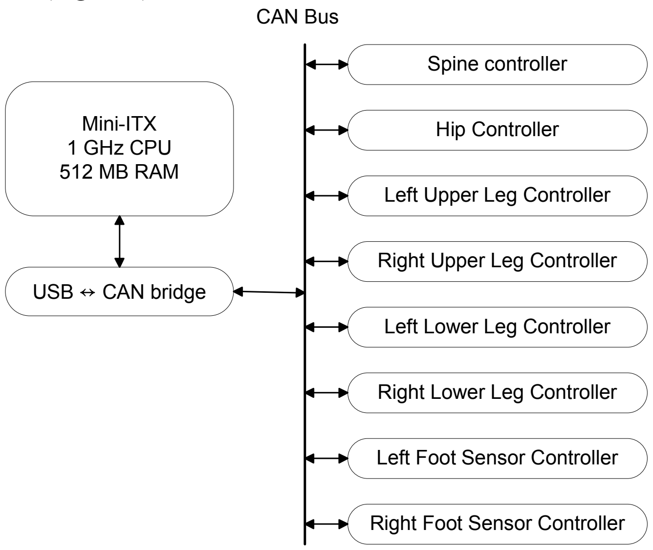

CAN Bus
Mini-ITX
1 GHz CPU
512 MB RAM
USB ↔ CAN bridge
Spine controller
Hip Controller
Left Upper Leg Controller
Right Upper Leg Controller
Left Lower Leg Controller
Right Lower Leg Controller
Left Foot Sensor Controller
Right Foot Sensor Controller

Figure 2 GuRoo distributed control architecture

Trajectories for each of the joints are calculated, and the desired joint velocities are sent to each motor control board via a CAN network. Each motor control board consists of a Motorola 68HC376 processor and discrete power electronics capable of controlling three motors. The control boards are responsible for running local control loops for these motors, and can measure and send diagnostic information back to the main controller. The local control loops run at 250 Hz, while the main trajectory generator runs five times slower, at 50 Hz.

2.2 GuRoo Simulator

The simulator is based on DynaMechs [McMillian, 1995], a dynamic simulation tool for multi-chained, star configured robots. It has been adapted to include specific characteristics of the GuRoo, including the distributed nature of the control architecture and the CAN bus. The GuRoo's chest is modelled as a mobile base with five chains arranged in a star configuration representing the arms, legs and head. The modified Denavit-Hartenburg parameters and CAD surface area graphically represent the robot, as seen in Figure 3.

Mass distribution information in the form of inertia tensors is combined with actual motor characteristics to simulate realistic interactions between links. The simulator provides the same programmatic interface as the firmware, with the ability to read encoders, measure current consumption and to transmit and receive CAN packets. This interface also provides simulated sensor information congruous with the real world sensors located on the robot.

The GuRoo has a vision system consisting of two Basler Firewire cameras. This system is capable of capturing and processing 30 frames per second. It can identify the soccer ball, coloured markers and soccer goals necessary to compete in the RoboCup Humanoid League.

Capacitive force sensors located in the feet are used to calculate the zero moment point acting on the robot.

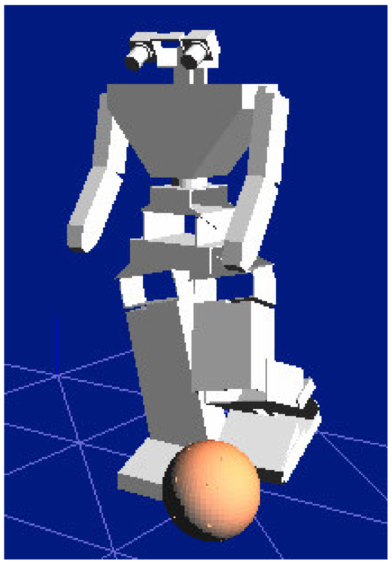


Figure 3 GuRoo Simulator

3 Fuzzy Associative Memory

A Fuzzy Associative Memory (FAM), employing the use of triangular membership functions, was implemented as the feedforward component in a Trajectory Error Learning schema. The FAM component consists of two sub-components, the fuzzy logic rules and the input/output mapping relation known as the Associative Memory.

3.1 Fuzzy logic

A fuzzy logic system contains sets used to categorise input data (*fuzzification*), decision rules that are applied to each set, and a way of generating an output from the rule results (*defuzzification*).

In the fuzzification stage, a data point is assigned a degree of membership (DOM) in the each set. The DOM is determined by a membership function. The membership function is often a triangular function [Castro, 1995] centred at a given point x_0:

$$DOM(x) = \max\left(1 - \left|\frac{x - x_0}{w}\right|, 0\right) \qquad (1)$$

The width of the membership function w is then set so that adjacent sets overlap, ensuring that the total degree of membership is constant. An example of four triangular membership functions is shown in Figure 4. Here each input is assigned to at most two of the sets “tiny”, “small”, “medium”, or “large”.

Once the input has been categorised into sets, rules are applied to the sets. The rules are if...then statements, e.g. *IF input1=tiny AND input2=small THEN output1=3*. Each rule inherits a degree of membership (or a degree of applicability) which is the product of the degrees of membership of the inputs. These rules may be input by a human expert before the system runs, but in the case of FAM, the rules are learned online.

Defuzzification is the name for a procedure to produce a real (non-fuzzy) output which combines the fuzzy rule

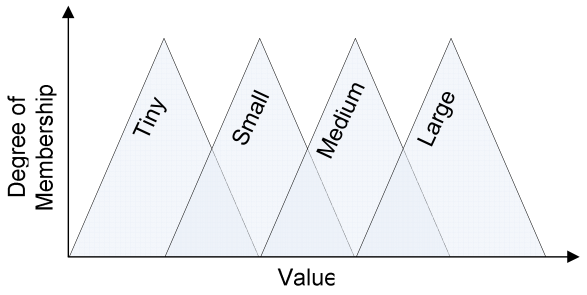


Figure 4 Example of membership functions for four fuzzy sets

results together. It generally takes one of several forms; in this case, each output is the weighted sum of applicable rules. The effect is to interpolate outputs between the points specified by the rules.

In the GuRoo system, the inputs are the phase of the current cyclic action, and the error in each of the joints. Thirty-six phase sets are used, and sixty-four error sets. Each of these sets has a triangular membership function, with a constant width. The outputs produced are the positional corrections to apply to each of the joints. Each joint has its own independent set of rules; i.e., the state of other joints is not considered when calculating corrections to a specific joint.

The number of sets to use in a particular application depends on how rapidly the output changes with respect to the inputs. A fuzzy logic control system effectively interpolates between known output values, so a large number of triangles means better interpolation, but higher memory use and less extrapolation. In the case of the GuRoo system, these numbers were chosen by knowledge of the range of expected data, and tuned by experimentation.

3.2 Associative Memory

Associative Memory is a type of memory with a generalised addressing method: the address is not the same as the data’s location, as in traditional memory [Skapura, 1995]. An associative memory allows a fuzzy rule base to be stored: the inputs are the degrees of membership, and the outputs are the fuzzy system’s output. Such a system is termed a *fuzzy associative memory* (FAM) [Si *et al* 1999]. A FAM is particularly useful if learning algorithms are applied, because it allows rules to be updated easily.

FAM may be implemented in software for restricted inputs by assigning a number to each possible input set (e.g. “tiny”=1, “small”=2, etc.), and using the numbers to index a multidimensional array (the number of dimensions being the number of input variables). Each element of the multidimensional array is then a list of the output values. In the case of this type of FAM control, the output is the position correction that must be made.

Using the associative memory presented thus far, the system has a storage method, and a way of calculating outputs, but no way of learning from its inputs. The learning process is governed by a learning rule; the type of data being learned influences the choice of learning rule. In this case, the learning rule for an output correction Δx is

$$\Delta x_{new} = \Delta x_{old} + \alpha\varepsilon$$

where α is the learning rate (which controls how much the existing knowledge is weighted relative to the new knowledge), and ε is the position error that was observed [Collins, 2003; Russell and Norvig, 2003]. This learning rule converges to a stable output state when the error is zero. In the case of the GuRoo system, the learning rule is applied to every element of the associative memory where the degree of membership for the input variables is non-zero. (Other alternatives include applying the rule to only the memory element with the highest degree of membership.)

3.3 Feedforward block

The feedforward FAM component outlined above is used as the learning component of a Trajectory Error Learning (TEL) architecture (Figure 5). A conventional feedback control loop is implemented on the motor control boards, with the desired position as the input to the system. Using the desired position and the trajectory phase as inputs to the system, the FAM generates a compensating signal which is added to the original signal's position. The updating of the table values in the associative memory component is driven by the position error measured in the system. As the error decreases, the change in values of the associative memory and hence compensating signal tends towards zero.

In general, each output should be calculated using all the joint position information available. In practice, this creates a prohibitively large FAM array with a large number of inputs: with *n* inputs, and *m* sets per input, memory usage is $O(m^n)$. So the output for each joint was determined only by its own error, not that of other joints. Effectively, each joint used an independent FAM controller.

Using a set of independent controllers can pose problems, however, because the dynamics of the joints are not actually independent. Because the dynamics are coupled, the FAM learning systems can potentially coevolve. The extent to which this coevolution helps or hinders the FAM system is investigated later in the paper.

3.4 Sensor Delay, Actuator Delay, and Temporal Credit Assignment

The fuzzy logic system as described so far presumes that the sensing and actuation is instantaneous. If sensing is not instantaneous, or there is a delay between sensing and processing the data (as is the case in GuRoo), then the desired position from the trajectory generator must be delayed so that it corresponds with the measured position. This is not difficult in the case of the GuRoo because the sensor delay is known accurately. Activity queue number 1 in Figure 5 compensates for this delay.

A more difficult problem arises when the actuator responds slowly, because of communication delays, inertia in the system, or a poorly tuned control loop. In this case, errors will not be corrected at the point in the associative memory that caused them. Further problems may flow on, such as induced unstable oscillations. This problem is known as the temporal credit assignment problem.

For a general plant, the solution is difficult to overcome, especially when no *a priori* estimate of this time is available. In the case of the GuRoo, the time delay is similar across joints because of the similar motors and hardware used, and trial solutions may be estimated and tested in simulation. One can obtain an initial estimate of the magnitude of this time delay by applying an impulse input from the associative memory and observing the system's output. Once an estimate of this actuator delay time has been made, an activity queue of this length may be used to store past inputs. When errors are observed, the portion of the table that is updated with the desired correction is determined by the earliest entry in the activity queue. Activity queues 2 and 3 in Figure 5 represent this actuator delay compensation. Their lengths were tuned as described above. This essentially places all credit or blame at the one time, a variant of the common TD learning algorithm. Because of the fuzzy nature of the table, the temporal credit is spread across two phase sets and two error sets. Fully solving the temporal credit problem in general is extremely complex, and essentially needs a complete system model of its own.

Although the input and output delays may seem symmetrical (a delay between input and output), changing the point where FAM is applied effectively changes the FAM system's objective. In the case of GuRoo, one of the objectives of FAM is to compensate for the slow system response, so separate input and output delay lines are necessary.

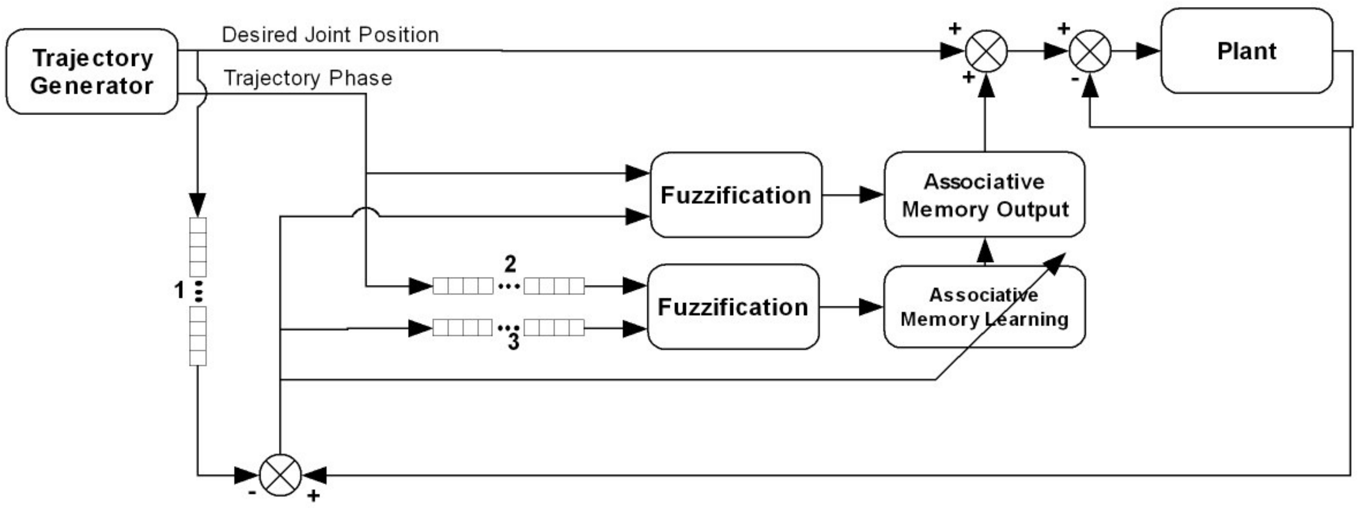


Figure 5 Fuzzy Associative Memory Implementation

4 Results

The FAM implementation was tested with the simulator to avoid damaging the robot. Initial experimentation was performed with a standard crouching movement, characterised by joint angles of 22°, 35° and 16° on the pitch axes of the hip, knee and ankles respectively. Figure 6 shows the position error present on each of these joints during a crouching motion without any compensation in the system.

FAM feedforward component was added to the hip pitch, knee pitch, and ankle pitch joints with a fast learning rate $\alpha = 0.2$. Thirty-six fuzzy sets in phase and sixty-four sets in error were used to categorise the inputs to the FAM, and inputs were scaled so this range was well-covered. An actuator delay length of 0.24 seconds was used. The robot visibly improved its balance with FAM as opposed to without FAM—Figure 7 shows the tracking. Table 1 shows the RMS error in both cases. However, when no actuator activity queues were used to address the temporal credit assignment problem, FAM created instabilities that caused the robot to overbalance, obviously worse than the behaviour with no FAM at all.

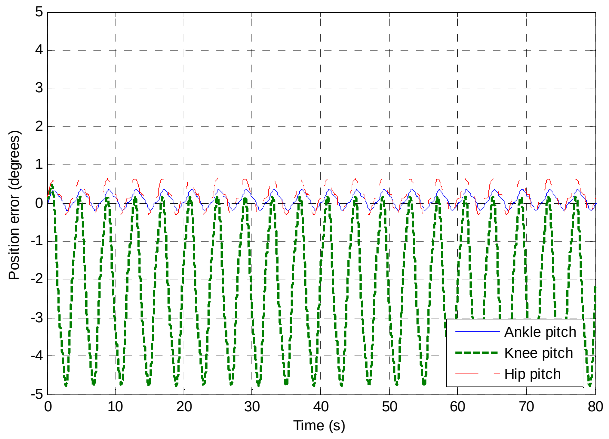


Figure 6 Joint position errors vs time during the crouching motion

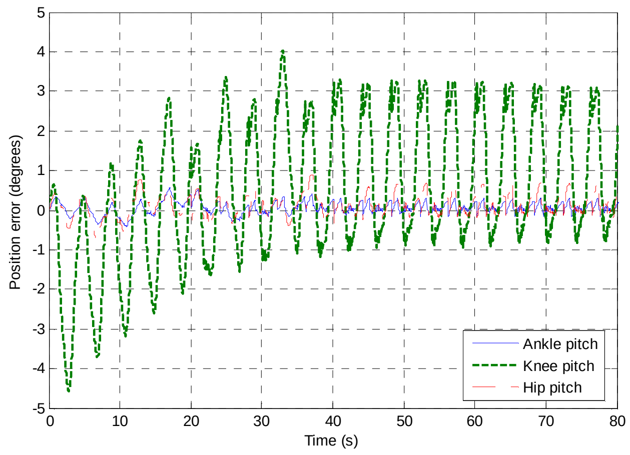


Figure 7 Joint position errors vs time during the crouching motion with FAM enabled

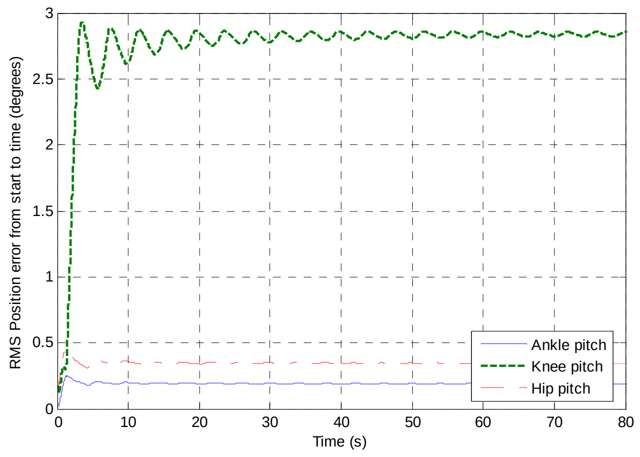


Figure 8 RMS position error with FAM disabled during the crouching motion

Joint	RMS error with FAM (degrees)	RMS error without FAM (degrees)
Ankle	0.17	0.19
Knee	1.87	2.85
Hip	0.30	0.34

Table 1 RMS joint position error with and without FAM

To demonstrate the learning effect of FAM, the RMS error was plotted. Figure 8 shows the RMS error with FAM disabled: it is essentially a constant. In contrast, with FAM enabled (Figure 9), the error decays at a rate proportional to the learning rate to its steady-state error.

In the case of the GuRoo system, FAM is not overly sensitive to the length of the delay queue. A range of actuator delay queue lengths from 3 to 18 (corresponding to actuator delays of 0.06 seconds to 0.36 seconds) were found to give a reduction in RMS error. Similarly, the GuRoo works with high or low learning rates (though higher learning rates may overcompensate), and a wide range of number of fuzzy sets. Changing the number of triangles did not impact heavily on the system. Additional triangles slowed the time taken to reach a constant output, in return

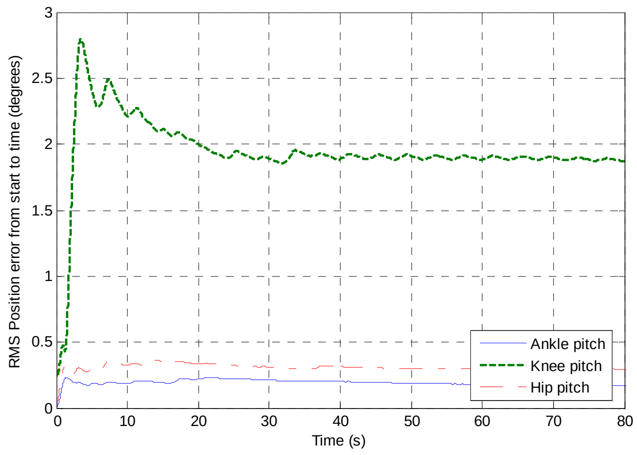


Figure 9 RMS position error with FAM enabled during the crouching motion

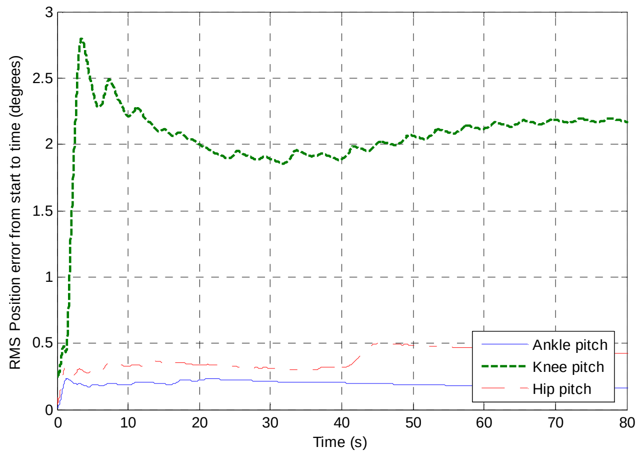


Figure 10 RMS position errors vs time when the robot lifts a 5 kg load at 40 seconds

for greater steady state stability. Fewer triangles allowed for faster learning and better generalisation.

The adaptive nature of FAM suggests testing it in a dynamic environment. The previous crouching test may be extended to a lifting situation. The robot is allowed to learn an appropriate compensation for its crouching motion, but after five cycles is given a mass of 5 kg in its arms. The FAM controller must then relearn the dynamics of the system.

Figure 10 shows the joint position errors when the robot lifts a load. The point where the load is applied is easily visible, but the motion quickly returns to its previous behaviour. A temporary increase in the error is seen, but the system converges to nearly the same long-term errors as in Figure 7.

In each of these tests, the robot started in a slightly crouched position so that it would not be moving to its joint limits. Because FAM knows nothing of the robot's joint limits, it may try to compensate past the robot's joint limits if the robot is operating near them. This creates a highly discontinuous force as the joint hits the limit, effectively violating the assumption underlying FAM that the system has continuous derivatives, and hence destabilizing the algorithm.

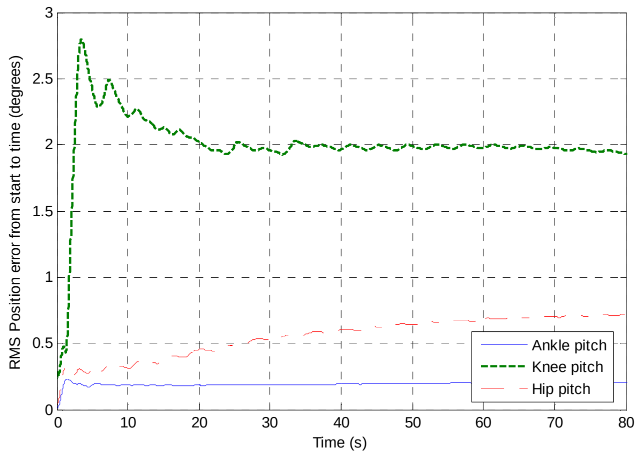


Figure 11 RMS Joint position errors vs time during the crouching motion with FAM compensation on the knee only.

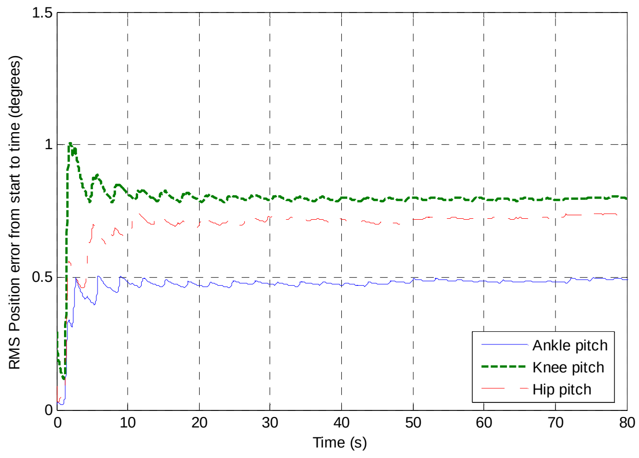


Figure 12 RMS position error vs time, with FAM disabled during a normal walking gait

The phenomenon of coevolution, where the learning of one joint affects the response of the system to the other joints, was observed, but not as dramatically as was first thought. Joint position errors in this case are shown in Figure 11. The knee error is almost the same as in Figure 7; differences are more apparent later in the evolution. In fact, the RMS knee error is slightly higher in Figure 11 (FAM applied to the knee only); it is 1.93 degrees, higher than the RMS error of 1.87 degrees when FAM was applied to all joints. Not observing unstable coevolution is heartening, because the most obvious solution to coevolution problems would be to use all joint errors as inputs to the associative memory, creating a many-dimensional lookup table that consumes exponentially more memory than the present implementation. However, other joint positions are quite well correlated with the gait phase, so the gait phase essentially encodes the other joint positions for a given motion. Although this approximation is certainly not exact, it appears to produce a favourable space/accuracy tradeoff.

For maximum utility, the one set of FAM parameters must be stable for a wide variety of period motions. To test this, the more complex motion of walking was used to test FAM's stability and effectiveness. The same FAM parameters as for crouching were used.

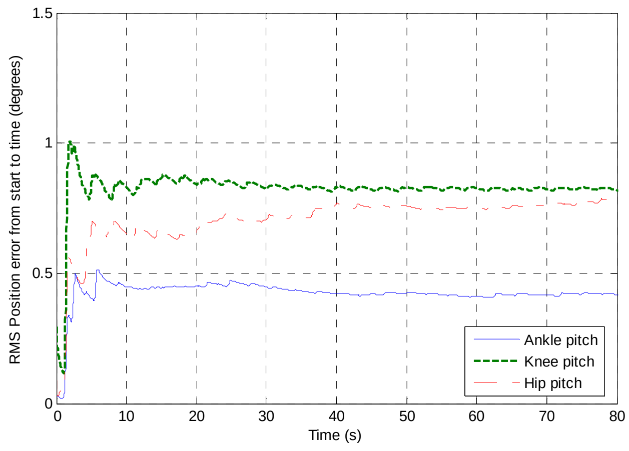


Figure 13 RMS errors with FAM enabled during a normal walking gait

Figure 12 and Figure 13 show the joint position error against time while walking with FAM disabled and enabled, respectively. The improvement with FAM is less than for the crouching motion, with the ankle pitch showing greatest improvement. It is thought that more improvement is observed in the crouching motion than the walking motion because the knees are not loaded as much in the walking motion as in the crouching motion, and the FAM system was tuned based on the crouching motion. This raises the question as to whether tuning control gains is easier than tuning FAM parameters for a particular application.

5 Future Work

FAM has been shown to improve joint-tracking errors, but there is clearly room for further improvement. In particular, the method for tuning FAM parameters is rather ad-hoc, requiring an understanding of the tradeoffs involved. Further, some more general parameters, such as the optimal set of membership functions and the method of temporal credit assignment, are very complex in general. Using other learning techniques (such as used by Si *et al* [1999]) can help tune the FAM parameters in a specific situation, but tuning all the parameters would generally take a large amount of learning time. Hence further work should focus on finding subsets of this parameter space that are likely to contain near-optimal sets of parameters.

In addition to the parameter search improvements, the techniques here could be extended to higher level robotic control tasks. For example, FAM could be used to aid toolpoint path planning.

6 Conclusions

Complex multi-degree of freedom robotic systems are difficult to model and hence difficult to control using traditional analytical techniques. Learning approaches such as FAM enable appropriate control without the need for accurate system dynamics to be determined. In this research, a FAM was implemented on a set of poorly tuned joint controllers on the GuRoo robot and resulted in an improvement in joint control.

The temporal credit assignment problem caused by the delay between commanded output and measured response was addressed through the use of activity queues to keep track of inputs and their relative impact. Use of the FAM on a simple crouching motion improved joint positional error by up to 30% in some joints.

The paper also shows the FAM's ability to compensate for changes in system dynamics. A sudden 15% increase in mass at the arms initially increased the positional error, but was quickly compensated for.

Co-evolutionary issues were initially thought to potentially impact the system, as changes in position of one joint can effect changes in another joint with constant inputs. The effects of co-evolution were found to be negligible once the system was implemented, probably due to the use of a gait phase variable to drive the learning.

When implemented on more complex motions such as walking, the improvement was not as noticeable, with only minor decreases in RMS error. The joint error present during a walking gait is less than during a deep crouch and as such, lower absolute improvements were expected.

The FAM implementation was found to provide an increase in performance but only from the basis of an initially poorly tuned controller. Even without accurate system dynamics, it was easy to tune the PI controller to give comparable results. The FAM still requires some tuning in the form of learning rates and number of membership functions.

7 References

[Castro, 1995] Fuzzy Logic Controllers Are Universal Approximators. J. L. Castro, *IEEE Trans. Syst., Man, Cybernetics*, vol. 25, no. 4, pp. 629–635, 1995.

[Collins, 2003] *Cerebellar Modelling Techniques for Mobile Robot Control in a Delayed Sensory Environment*, David Timothy Collins, PhD Dissertation, The University of Queensland, 2003.

[Commuri and Lewis, 1996] Adaptive-Fuzzy Logic Control of Robot Manipulators. *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, pp. 2604–2609, 1996.

[Kee, Wyeth, 2003] *Cerebellar Joint Compensation for a Humanoid Robot*, Damien Kee and Gordon Wyeth, presented at the 2003 Austrlalian Conference on Robotics and Automation (ACRA 2003), 2003.

[Lee, C.-C. Berenji, H.R.] An intelligent controller based on approximate reasoning and reinforcement learning. *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, pp. 200–205, 1989.

[McMillian, 1995] *Computational Dynamics for Robotic Systems on land and Underwater*, S. McMillian, PhD Dissertation, Ohio State University, 1995.

[Miller; Glanz; Kraft, 1990] CMAC: An Associative Neural Network Alternative to Backpropagation, W. Thomas Miller, III, Filson H. Glanz, L. Gordon Kraft, III, *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1561–1567, 1990.

[Pan, Woo, 2000] PD Manipulator Controller with Fuzzy Adaptive Gravity Compensation. Pan L, Woo PY, *Journal of Robotic Systems* vol. 17, no. 2, pp. 93–106, 2000

[Russell and Norvig, 2003] *Artificial Intelligence—A Modern Approach*, Second Edition. Stuart Russell and Peter Norvig, Pearson Education, Inc., New Jersey 2003.

[Self, 1990] Designing with Fuzzy Logic. Kevin Self, *Spectrum, IEEE*, vol.27, iss.11, Nov 1990.

[Si *et al* 1999] Modified Fuzzy Associative Memory Scheme Using Genetic Algorithm, Jie Si, Naiyao Zhang, and Rilun Tang, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, pp. 2002–2006, 1999.

[Skapura, 1995] *Building Neural Networks*. David M Skapura, Addison-Wesley, 1995.

[Widrow, Stearns, 1985] *Adaptive Signal Processing*, B. Widrow and S. D. Stearns, Prentice-Hall, 1985.

[Wang, Vachtsevanos, 1990] Fuzzy Associative Memories: Identification and Control of Complex Systems. Bo Hyeun Wang George Vachtsevanos, *5th IEEE Int. Symposium on Intelligent Control*, vol. 2, pp 910–915 1990.