JOINT CONSTRAINT MODELLING USING EVOLVED TOPOLOGY GENERALIZED MULTI-LAYER PERCEPTRON (GMLP)

GLENN JENKINS[1], PAUL ANGEL[2]

[1]School of Applied Computing, Faculty of Applied Design and Engineering,
Swansea Metropolitan University, Swansea, SA1 6ED, Wales
Email: glenn.l.jenkins@smu.ac.uk
[2]Department of Computing and Mathematics, Faculty of Advanced Technology,
University of Glamorgan, Pontypridd (Cardiff), CF37 1DL, Wales
Email: pangel@glam.ac.uk

Abstract: The accurate simulation of anatomical joint models is important for both medical diagnosis and realistic animation applications. Quaternion algebra has been increasingly applied to model rotations providing a compact representation while avoiding singularities. This paper describes the application of artificial neural networks topologically evolved using genetic algorithms to model joint constraints directly in quaternion space. These networks are trained (using resilient back propagation) to model discontinuous vector fields that act as corrective functions ensuring invalid joint configurations are accurately corrected. The results show that complex quaternion-based joint constraints can be learned without resorting to reduced coordinate models or iterative techniques used in other quaternion based joint constraint approaches.

Keywords: Anatomical joint constraint, quaternion, piece-wise linear, discontinuous, evolved neural network, topological evolution, NetJEN.

1. INTRODUCTION

Joint systems are important constituents of anatomical models, they are used in simulation to retain anatomically correct movement and ensure limbs do not separate or intersect. Current techniques are limited by their underlying representation or their abstraction of the joint function. Demand is increasing for anatomically correct joints for applications in animation and medicine [1, 2]. However in current applications increasing accuracy incurs additional complexity and therefore computational cost [3-5].

Dynamics solutions can be used to produce realistic behaviour based on input, contact and constraint forces [6]. Depending on the complexity of the simulation, the outcome of dynamics-based behaviour can be difficult to predict. Inverse-Kinematics (IK) based approaches however allow the precise placement of end effectors as constraints [3]. IK solvers attempt to resolve constraints within a constraint system, a problem compounded by the existence of zero or more solutions [3].

Kinematics based solvers can be classified as analytical, often resorting to reduced coordinate formalisms, or numerical, using iterative approaches to solve a system of constraints. An important aspect of this is how the constraint is represented. This work builds on previous work in joint constraint modelling by extending quaternion abstractions of phenomenological joints (whose behaviour can be modelled without reference to the underlying joint anatomy). Artificial Neural Network (ANN) techniques are used to learn vector field functions that describe the joint's behaviour and correct invalid joint configurations directly in quaternion space without resorting to iterative approaches.

1.1 Previous work

Primitive joint constraints have been parameterised using Euler angles [7-9]. However inter-dimensional dependencies are not represented [10] and singularities or "Gimbal Lock" are encountered [11]. Inter-dimensional dependencies between Euler angle components can be expressed using equations [12], such equations can provide mathematical descriptions of rotational constraint boundaries. Here geometric functions are fitted to a given dataset, examples include spherical [13] and conical polygons [1, 14].

Approaches such as special orthogonal matrices have been used to overcome the problem of singularities [2, 15]. More recent research has focused on the use of quaternions to model rotations and joint constraints. Quaternion algebra allows rotational models to be represented without the presence of Gimbal Lock [11]. Quaternions are an extension of complex numbers, composed of 1 real and 3 imaginary components where $q = \langle s, i, j, k \rangle$.

Multiplying complex numbers results in rotation in the complex plane, giving rise to the complex identity i^2 = -1. This is extended in a subset of quaternion space, where all quaternions are of unit length, to i^2 = j^2 = k^2 = -1. Cyclic-coordinate-decent, a IK technique popular in real time applications, was implemented with a set of simple quaternion based constraints by Lee [16].

Lee [16] decomposes a single quaternion into two quaternions each representing rotation in a single plane (effectively swing and twist for conic and axial constraints). In each case the centre of the constraint is known, a quaternion describing the swing of the joint can be created based on the angle between the centre and its image rotated by the subject quaternion and the axis calculated from the cross product of the constraint centre and its rotated image. The second quaternion representing the rotation around the axis can then be calculated by calculating the twist alone, (removing the swing component) the axis and angle of this quaternion can then be calculated. Conic, axial and revolute constraints are defined and can be used to model basic constraints, more complex constraints can be defined with a union of these basic types. Interrogation of these shapes (to ascertain the validity of a joint configuration,) is presented, but no method of calculating a correction to the nearest valid orientation is defined.

Liu and Prakash [17] build on Lee's work. Using a sampled boundary they create a function to constrain the decomposed quaternion that can be used for both constraint validation and clamping to the boundary.

In the quaternion iso-surface approach of Herda *et al* [3, 18] limb rotations were recorded and represented in quaternion space. A set of four-dimensional unit quaternions describing the valid joint rotations are projected to a cloud of points in three-dimensions. This overcomes the problem of ambiguity in quaternion space and allows the creation of an iso-surface surrounding the cloud of valid points defining the boundary between valid and invalid rotations. An iterative approach can then be employed to resolve invalid joint configurations.

Johnson [19] also reduced the dimensionality of the quaternion by projecting one half of the unit quaternion hyper-sphere onto a three-dimensional tangent space. A set of quaternions expressing valid joint and pose constraints are then generated. Constraints are based on the maximum deviation from the mean of the collected data and corrections implemented by recursively moving an invalid point closer to the mean.

Artificial Neural Networks (ANNs) have been applied to IK and some approaches have included joint constraint using both recurrent [20-26] neural networks with unsupervised learning and feed forward [27, 28] neural networks with supervised learning. Feed-forward network architectures such as that of the Multi-layer Perception have been popular since their resurgence in the mid eighties [29]. These are trained to give certain outputs in response to given inputs by repeatedly adjusting the strengths of the interconnections between neurons. This optimises a boundary delineating regions within a multi-dimensional feature space.

Though ANNs have been successfully applied to various problems, no analytical rule has been developed governing the optimal topology of the network [30]. Large neural networks with high connectivity show improved approximation but poor generalization while smaller ANNs with low connectivity show better generalization capabilities but poor approximation.

Huber, Mayer and Schwaiger [30] attempted to solve to this problem by means of Evolutionary Algorithms (EA) which search for a problem-adapted neural network topology. EA are used to evolve the topology while traditional a traditional supervised training is used on each of the networks in the evolved population.

The paper is structured as follows. Section 2 provides a description of our methodology with reference to the techniques employed. Section 3 outlines the experiments carried out and their results while Section 4 discusses and draws conclusions from these results.

2. METHODOLOGY

2.1 Constraint Modelling

This work extends our previous work by modelling discontinuous vector fields in quaternion space [31, 32]. In this scheme, a joint is modelled as a single unit-length quaternion that describes the relative rotation between the joint's constituent components. The constraint behaviour is learned using evolutionary ANN techniques from data-sets that describe the rotational extrema of the joint to be modelled.

The behaviour of the joint constraint is modelled as a vector field in $\Re 4$ that maps invalid rotations to the nearest valid rotation, as illustrated in Fig. 1.

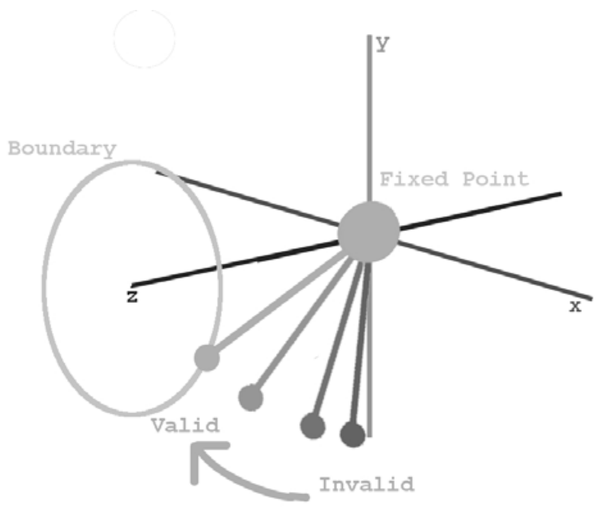

Fig. 1 – Constraint boundary modelling in quaternion space.

2.2 Evolutionary Algorithms

NetJEN is a Java based implementation of NetGEN [30, 33, 34] developed by researchers at the University of Salzburg. NetJEN [35] boasts several impressive features and provides an intuitive user interface in addition to reporting tools and other useful functionality. A brief outline of the system they developed follows based on published work [30, 33-40].

Before GA techniques can be applied to topological evolution the ANNs underlying structure, the phenotype, must be considered as a genotype (a blue print for the construction of the network.) This must be encoded such that GA techniques can be applied. There are two common approaches; *Indirect Encoding* encodes a set of constraints that govern the construction of individual neural networks within the population. These constraints are evolved indirectly impacting on the neural networks generated. In NetJEN *Direct Encoding* is used, a network topology is created and encoded minimising the decoding effort to map between the genotype and corresponding phenotype. The encoding scheme used is called the *Modified Miller Matrix*, an extension of the *Miller Matrix* [41].

The genome structure comprises *Learn Parameters,* which describe the values required to train the neural network. *Activation Function Template Parameters* are used to describe one or more activation functions present in the network. *Neuron Parameters* indicate the type of neuron and *Structure Parameters* explicitly specify each connection within the network [34]. Markers (binary inhibitors) are used to regulate the expression of wild-type genes, for example hidden neurons, while other problem dependent genes such as output neurons are fixed [34]. As a result the bit string includes some non-coding regions (*Introns*), these have been shown to reduce the effects of crossover and are common in biological systems [38-40].

The structure and neuron parameters are represented by a linearized binary adjacency matrix [34] as shown in Fig. 2. As the network architectures are feed-forward the triangle above the main diagonal must be zero, the main diagonal is used to represent the activation function index (zero if not expressed) [34]. The maximum size of the network is set in advance and so the size of the structures does not change during evolution. The activation function template parameters and activation functions were not evolved during the following experiments but are included in descriptions of the genome for completeness

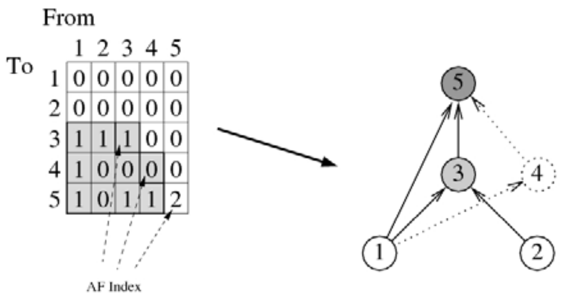

Fig. 2 - The Genotype/Phenotype mapping, here the presence of a one in indicates a forward connection from the node identified by the row number to the node identified by the column number. As there are no links from a node to itself the main diagonal represents the Activation Function (AF) Index is the index of the activation function of a given neuron. The above figure shows all the notes of the system node 4 however despite having an activation function is not part of the generated phenotype is not represented in the genotype as it has no input connections. The image reproduced from Mayer and Schwaiger [34] with their permission.

The system comprises of a Simple Genetic Algorithm (or SGA), the Genotype Phenotype Mapping and the Neural Network Manager. The Neural Network Manager (NNM) in NetGEN was the *Stuttgart Neural Network Simulator (SNNS)* [42], and the SGA from Smith *et al* [43] an implementation of previous work by Goldberg [44]. In the Java implementation the NNM used is BOONE, developed by August Mayer at the University of Salzburg.

The SGA generates blueprints for a random population of ANNs which are validated and passed to the NNM where they are constructed and trained using *Resilient Back-propagation* [45]. The SGA then assigns fitness values to each network using a fitness function. This *Composite Fitness Function* comprises a measurement of the networks performance (*Model Fitness*) and a complexity regularization term (*Complexity Fitness*,) as expressed in equation 1.

$$F = \alpha 1 \frac{1}{1+\varepsilon m} + \alpha 2 \frac{1}{1+\varepsilon c} \tag{1}$$

Where,

$$\alpha 1 + \alpha 2 = 1.0 \tag{2}$$

In equation 1, *F* is the fitness of the neural network *εm* is the M*odel Error* (Sum Squared Error or SSE) and *εc* the complexity regularization term. $\varepsilon c = |C_{total}|$ with C_{total} being the total set of neural network connections. The regularization weight (*α2*) has been shown to be most effective in the range 0.001 to 0.01 to guide the evolution towards networks of low complexity [34]. The error weight (*α1*) is derived from regularization weight (*α2*).

The SGA uses *Binary Tournament Selection* to select the best networks of the population to breed, *n* individuals (typically two) are selected and the individual with the highest fitness is placed in the breeding pool. The selection itself is weighted, the higher an individuals fitness the more likely it is to be chosen [46]. Binary Tournament Selection has been found to be superior to Proportional Selection methods [39].

An entirely new generation of individuals is created through crossover and mutation of the fittest individuals selected from the last generation. This completes the evolutionary cycle that continues for a specified number of generations. It should be noted that the fittest individuals of the last generation will appear in the breeding pool more than once and inbreed generating identical offspring in the new generation [46]. This ensures that the best genetic patterns are passed on to the next generation. Crossover and mutation are implemented on a linearized *Modified Miller Matrix* allowing standard two-point crossover, this has a more global effect on the bit string than the exchange of rows and columns used in the original *Miller Matrix* approach [38-40].

2.3 Vector Field Modelling

A vector field is defined as a mapping that assigns each input to an output via some vector function [47]. Dynamic behaviour (such as that exhibited in joint constraints) can be described as a change in state that is determined by a function dependent on the current state. There is clear similarity between the mappings required for vector fields and those involved in the description of the kinematic behaviour of joint constraints [48].

Feed forward [48, 49] and Adjoint [50, 51] ANNs have been used to approximate continuous two dimensional vector fields and have been successfully used to reconstruct continuous vector fields in three dimensions [52].

Neural networks have been utilised for physics based animation by Grzeszczuk, Terzopoulos and Hinton [53]. In their approach complex forward dynamics equations required for physics based animation were replaced with neural networks, predicting the complex vector mapping from the current state to a future state based on the current state, the applied force and external forces.

In earlier work the authors successfully used topologically evolved neural networks to approximate discontinuous vector fields representing corrective constraints with inter-dimensional dependencies in one, two and three dimensions [32].

2.4 Neural Network Configuration and Training

The neural networks that were evolved during training consisted of three layers, an input layer, hidden layer and output layer. The nodes of the hidden layer have sigmoidal functions, while those of the output layer have linear functions. The number of hidden nodes and connection topology are initially randomised then evolved using EA. The weights of the interconnections are also initially randomised then updated using the resilient back-propagation algorithm.

The input and output layers contained four nodes representing the four components of a unit quaternion. The input quaternion describes the current joint configuration while the output quaternion response is dependent on the validity of the input orientation with respect to the learned constraint model. If the input quaternion is recognized as being invalid, the network response represents a quaternion rotation to correct the input quaternion to the nearest valid quaternion rotation. However, if the input quaternion is valid, the network outputs the identity quaternion, which represents no rotation. In effect, the network learns an implicit boundary between valid and invalid joint configurations.

The number of generations, training epochs and hidden nodes were restricted to reduce training times. Each experiment was repeated five times to ensure the consistency of the results. The regularization weight was chosen based on publications by the systems authors [34], as was the learning rate [39], the stopping MSE for the networks was identified though experimentation. The size of the population, number of generations and the number of training patterns were suggested by a co-author of the NetJEN system Dr. Helmut Mayer in private correspondence. The evolution and training parameters were configured as shown in TABLE I

TABLE I
EVOLUTION AND TRAINING SETTINGS

Parameter	Description	Value
Regularization function	Secondary fitness function.	Number of links
Hidden Nodes	Maximum no. of hidden nodes.	20
Number of Generations	No. of generations over which the ANN were evolved.	50
Population Size	Size of the populations evolved.	20
Fitness Function	Primary fitness function.	Inverse SSE
Regularization Weight	Regularization weight (*α2*) this term controls the effect network size on the fitness function.	0.01
Evolve number of Links	Networks are pruned down from fully connected networks.	On
Evolve number of Hidden Nodes	Evolves the no. of hidden nodes.	On
Evolve number of epochs	Evolves the no. of training epochs	On
Learning Rate	Learning rate used when training the ANN.	0.1
Stopping Error	MSE at which the ANN are stopped.	0.001
Training Function	Training function used to train the weights of the ANN.	Resilient back-propagation
Max Epochs	Maximum number of training epochs	500

Through experimentation, it was determined neural networks with sigmoid activation functions in the hidden layer and linear activation functions in the output layer produced good results. This distribution of activation functions was used throughout these experiments a similar distribution were employed for vector field approximation by Grzeszczuk *et al* [53], linear output layers have also been used with bi-polar sigmoid hidden layers [48, 49]. Each experiment was repeated five times to ensure the consistency of the results.

3. RESULTS

3.1 Regular Constraint Boundaries

Neural networks were successfully evolved and trained to model discontinuous vector fields representing regular (spherical) constraints. This is reflected both by the low Mean Squared Error (MSE) values and the structure of the neural networks - indicated by the number of hidden nodes as shown Fig. 3. The number of hidden nodes was limited to reduce training times (as indicated in TABLE I,) and the size of the hidden layer for each trained network was close to this maximum throughout. The number of hidden nodes appears to increase with the MSE (as shown in Fig. 3). This indicates that more complex networks were required for the given constraint ranges and that the high error is contributed to by the restriction on network size.

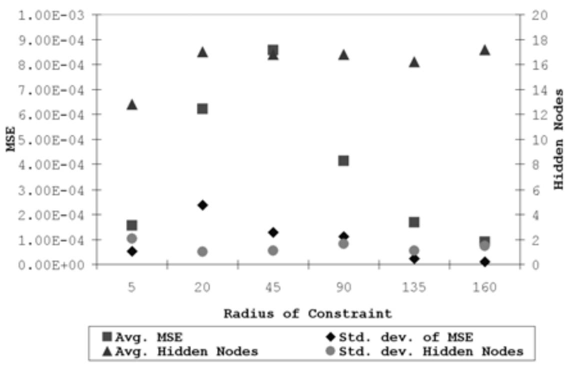

Fig. 3 – Graph showing the effect of range variation on MSE and network complexity (hidden nodes).

The increase in MSE between a 20° and 90° constraint radius does not correspond to changes in the constraint size. To understand this behaviour further, correlations are sought with respect to the distribution of training patterns in quaternion space. Principle Component Analysis (PCA) gives a set of eigenvectors that describe the principle directions of variation while the eigenvalues describe the contribution of each vector to the overall variation of the dataset. The eigenvectors produced were compared to the identity quaternion and the change in rotation between them noted. The fourth principle component has an eigenvalue of zero and is thus ignored.

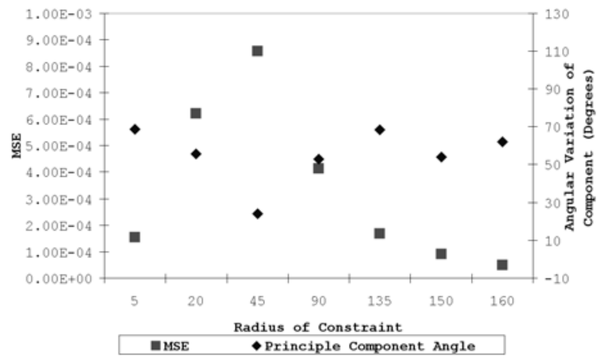

Fig. 4 – Graph showing the effect of range variation on the MSE and the change in orientation of the principle component.

MSE is a holistic measurement of error masking regional variations in the error over the vector field. To quantify these errors, we observe the behaviour of a virtual joint constrained by the trained neural networks. The *l2*norm (Pythagorean distance) between the ideally corrected and neural network corrected endpoint of a virtual limb is observed. This more direct comparison demonstrates that the error is highest around the boundary separating the valid and invalid regions, as shown in Fig. 5.

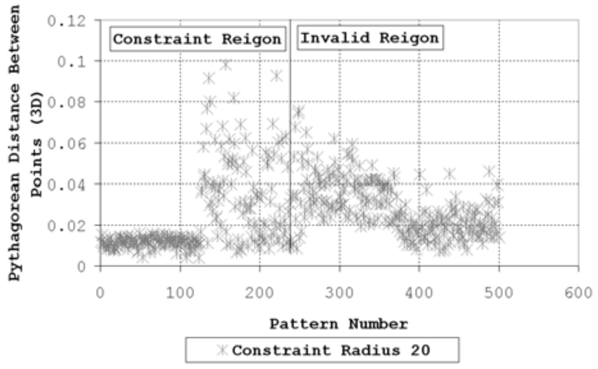

Fig. 5 – Plot showing the four-dimensional Pythagorean error between the ideal quaternion correction and the corresponding neural network output.

Fig. 6 illustrates the ideal and actual correction vectors for a virtual limb constrained by a regular boundary constraint. The dark dashed line shows the result of the corrections from the training data while the solid line shows the results of the neural network corrections, both lines lighten from their initial positions to their corrected positions. The boundary can be clearly identified and it is noted that all corrected rotations reside on or close to the constraint boundary.

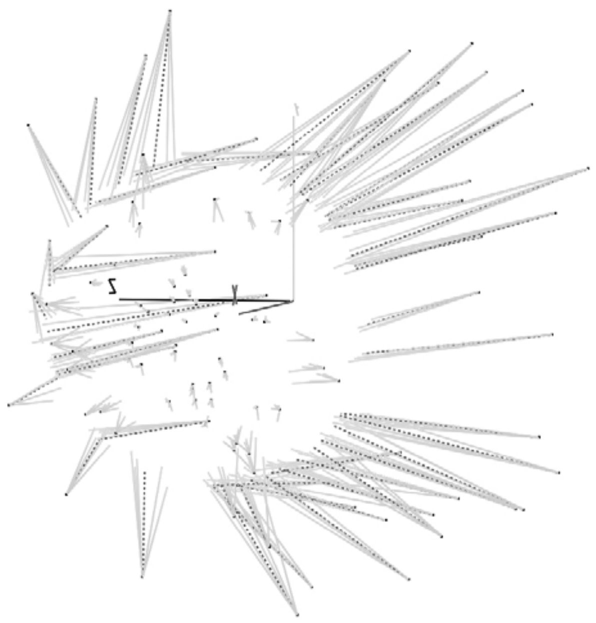

Fig. 6 – Ideal and neural network corrected rotations. Ideal corrections are shown as dark dashed lines; neural network corrections (for each pattern) are shown as light solid lines.

The network's behaviour can be further understood by looking at the distribution of training patterns. Fig. 7 shows the test inputs for a regular constraint with 20° radius. The input rotations of the test set are shaded according to their error when corrected by the neural network (compared to the ideal corrections of the test set). The darker points represent lower error and lighter points represent higher error. Salient regions of higher error are those around the constraint boundary and in the region diametrically opposite the constrained region on unit sphere.

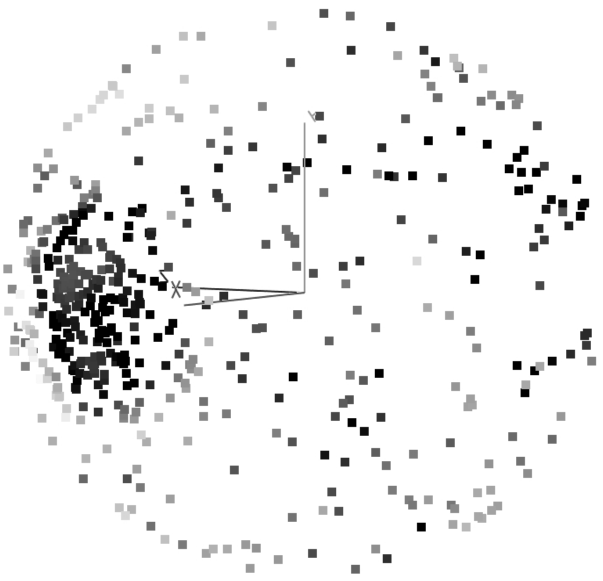

Fig. 7 - Test patterns shaded to represent the *l2* norm (i.e. error) of the relative neural network output. The darker patterns represent low error and the lighter ones high error.

Overlaying the patterns used in training (‘x’) and evolution (‘+’) of the neural network it is observed that regions of sparse training data correspond to regions of high error. In Fig. 7 the camera is placed inside the same sphere of points shown in Fig. 6. High error points are observed in an area where there are few training set points. Low error points are observed in regions well populated with training points. The validation set points (‘+’), used to assess network fitness, have less effect.

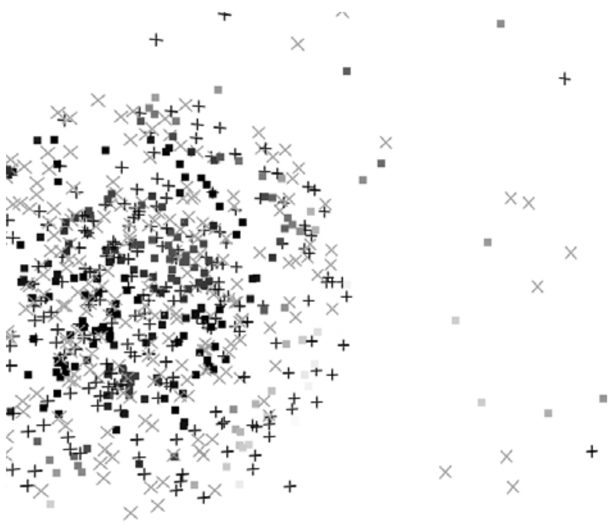

Fig. 8 - Regions displaying poor performance. Dark squares depict low error and light spots high error. Training patterns (‘x’) and validation patterns (‘+’) are overlaid.

Experiments were carried out to ascertain the effect of aligning the constraint centre of the virtual limb with each of the principle axes in 3D space. The variations in the effects of the principle components are indicated by their eigenvalues, where smaller values indicate a more evenly distributed dataset. When considered in relation to the average Pythagorean error there is a strong indication that the performance of a neural network is linked to the distribution of training patterns in quaternion space (as shown in Table II).

TABLE II

EFFECT OF EIGENVALUE VARIANCE ON MSE

Limb Start Alignment	Variance of Eigenvector Contributions	Average MSE	Average Pythagorean Error
Y	310.9959	5.61E-04	0.023673
Z	311.4465	4.60E-04	0.02448747
X	317.2301	4.88E-04	0.0257511

The results show that the networks performance (indicated by the average Pythagorean error for all patterns) increases as the variation in eigenvalues increases, i.e. as the dataset becomes less evenly distributed in quaternion space.

The quaternion representation on which the neural networks are trained is inherently redundant. Quaternions occupy an S3 hyper-sphere in four-dimensional space, representing 4π rotations, so quaternions on opposite sides of the S3 hyper-sphere (q and $-q$) represent identical rotations [3]. There is ambiguity between the position in quaternion space and the rotation represented.

The datasets used to train the neural networks consist of valid and invalid quaternions created from a simulation of a 3D virtual joint. In mapping the joint’s valid and invalid positions to quaternion space, multiple regions may be created due to the inherent redundancy of the quaternion representation. This is turn may result in multiple constraint boundaries on the quaternion sphere.

Experiments were undertaken to ascertain the effect of these factors on neural network approximation. It was postulated that mapping quaternions to one side of the hyper-sphere would simplify the four-dimensional vector field and improve training. We refer to quaternions that are not mapped as ambiguous (AMB) and those that are mapped as non-ambiguous (non-AMB).

The results summarised in Fig. 9 indicate that mapping quaternions to one side of the hyper-sphere increases the error on larger constraint ranges. To help interpret these results, PCA was performed on ambiguous and non-ambiguous datasets over the tested ranges. The results show that below 90° no points were mapped and the PCA gave identical eigenvectors and eigenvalues. However above 90°, an increase in mapped points results in an increase in network error and an increase in the difference between principle components.

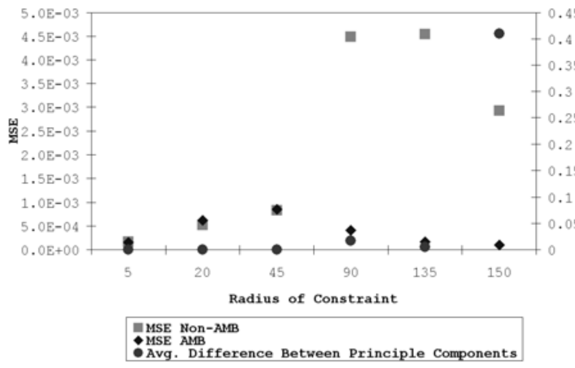

Fig. 9 - A comparison of the MSE recorded over various ranges for ambiguous (AMB) and non-ambiguous (non-AMB) results, plotted against the length of the difference quaternion between the principle components of each.

The length of a four-dimensional vector representing the difference between the principle component matrices (4 x 4) for each set (ambiguous and non-ambiguous) is used as a metric. In Fig. 9 a clear correlation can be observed between the increasing difference in the distribution of patterns in quaternion space and the increase in error.

To further understand how the patterns in quaternion space are changing, the orientation (with respect to a static unit vector) and influence of the principle components for a constraint radius was investigated. Fig. 10 and Fig. 11 show the difference in the orientation of the principle components and their contribution compared to the average MSE of the ambiguous and non-ambiguous datasets. That is, the percentage of the variation that can be attributed to the component.

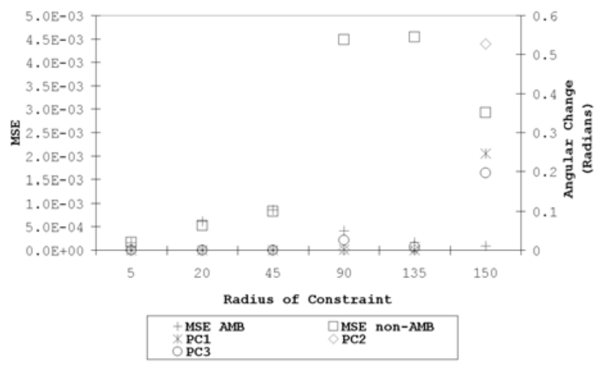

Fig. 10 -The difference in the orientation of principle components plotted against the average MSE for the ambiguous (AMB) dataset (with quaternions on both sides of the hyper-sphere,) and the non-ambiguous (non-AMB) dataset (with all quaternions on one side of the hyper-sphere.)

Mapping quaternions to one side of the hyper-sphere has a major effect on the orientation of the third principle component. There is initially no difference in the distribution of the datasets, however above 45 degrees the error on the ambiguous and non-ambiguous datasets diverges, (as shown in Fig. 10) At the same point the orientation of the third principle component and to a lesser extent the first and second principle components begins to change.

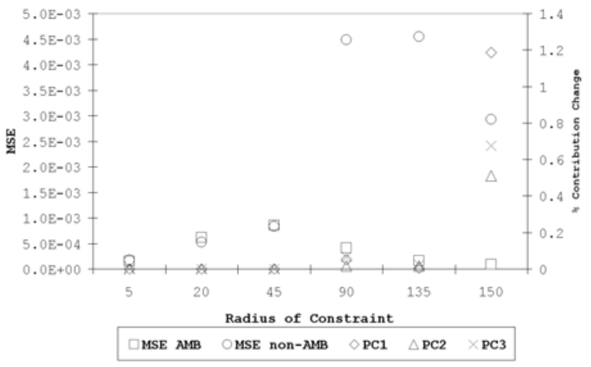

Fig. 11 -The difference in principle components contribution plotted against the average MSE for the ambiguous (AMB) dataset (with quaternions on both sides of the hyper-sphere and the non-ambiguous (non-AMB) dataset (with all quaternions on one side of the hyper-sphere.)

There are also changes in the contribution of the principle components once again at the same point that the plots of error on the ambiguous and non-ambiguous datasets diverge, (as shown in Fig. 11). Changes in the principle component direction and the contribution of the principle components indicate that the distribution of the data in quaternion space changes.

3.2 Irregular Constraint Boundaries

In human anatomy most of the rotational boundaries encountered are irregular. Therefore the performance of our technique on such boundaries is an important consideration. In the following experiments irregular boundaries designed to test the capabilities of our constraint modelling approach are implemented.

Experiments using an ambiguous dataset, where invalid rotations were corrected to the closer of two boundaries (one on either side of the quaternion hyper-sphere), did not train successfully. Based on the results of earlier experiments the valid quaternion and the boundary to which invalid quaternion were corrected were forced to inhabit the same side of the quaternion hyper-sphere. Invalid quaternions from both hyper-spheres are corrected to the same boundary giving a continuous vector field in this region.

The results show the neural network was able to learn the irregular boundary, though the error was higher and the networks evolved subsequently more complex than in the regular boundary case (as shown in Table IV).

TABLE III

COMPARATIVE PERFORMANCE ON TEST DATA

	Min / Max Error	**Avg. Error**	**Avg. Hidden Nodes**	**Avg. Pythag. Error**
Regular	4.79E-05 / 7.86E-04	3.31E-04	16	0.0197
Irregular	9.41E-03 / 1.41E-03	1.24E-03	18.4	0.0430

Visualisation of the error demonstrates similarities with the regular boundary case, where errors are clustered in regions of sparse training data and around the constraint boundary. An additional region of high error is also present where rotations are corrected towards the concave region of the boundary (as shown in Fig. 12).

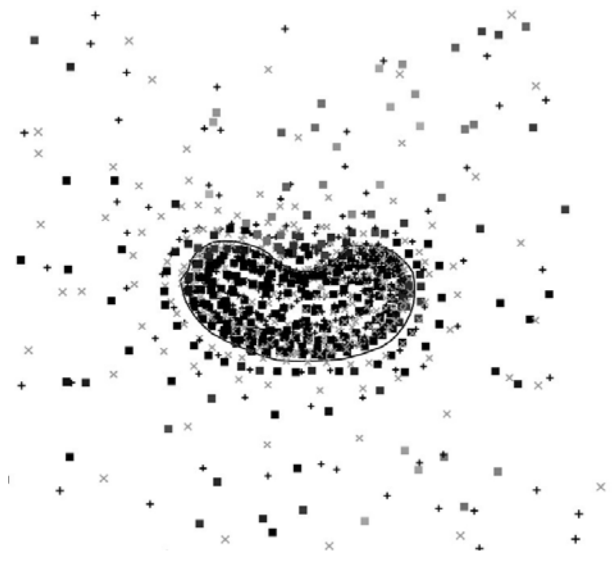

Fig. 12 - Visualisation of error distribution. Test patterns depicted as squares graded according to their error (darker points represent higher error). Overlaid by training patterns (x) and test patterns (+).

4. DISCUSSION & CONCLUSIONS

The results show that artificial neural networks trained with evolutionary genetic algorithms are capable of learning vector field models that can be applied to joint constraint problems. This has the advantage that constraints can be described and corrected with neural network models using quaternion representations directly, without resorting to reducing the dimensionality of (or otherwise pre-processing,) the quaternion as in Herda et al [18, 54], Lee [16] and Johnson [19]. Unlike Lee [16] a method for correcting to a valid orientation is presented while avoiding the need for iterative methods as used by Herda et al [18, 54] and Johnson [19].

The results indicate that the distribution of the vector field in quaternion space has a significant effect on the complexity of the mapping and therefore the success of neural networks training. PCA indicates that the fluctuations in error as the radii of the modelled regular constraint increases are the result of changes in orientation of the principle components and hence changes in the distribution of the data in quaternion space as shown in Fig. 4. Furthermore initial limb orientation results suggest that a more regular distribution (lower variance in the contributions of principle components) may results in improved quaternion learning (as shown in TABLE II).

The regional continuity of the mapping has a significant effect on the complexity of the vector field. The failure of experiments that implemented boundaries on both halves of the quaternion hyper-sphere indicates that the addition of ambiguity and additional discontinuities in quaternion space increase the complexity of the mapping. Forcing all quaternions to one side of the quaternion hyper-sphere produce poor results as the radii of the constrained region increased (as shown in Fig. 9). The PCA of these datasets shows a change in their distribution coinciding with the increase in MSE suggesting that discontinuity or ambiguity have been introduced to the dataset (again shown in Fig. 10).

In both the regular and irregular boundary experiments significantly improved results were obtained when quaternions on either side of the hyper-sphere were corrected to a single boundary. This preserves the continuity of the distinct regions and produces superior results. In conclusion evolved neural networks are capable of learning discontinuous vector fields in quaternion space provided the dataset is appropriately distributed and regional continuity is maintained.

This work has implications in a number of areas where accurate anatomical joint models are required. In animation for example where character specific models are required to reduce animation effort (in avoiding unnatural limb orientations) and to maintain consistency with live action characters (where motion is limited due to clothing, age or injury).

The neural network approach was in the regular boundary case slower than the dataset generation program due in part to the simplicity of the dataset generator and more importantly to the design of BOONE (Basic Object Orientated Network Evaluator) a part of NetJEN. BOONE (the NNM) sequentially fires nodes until there are no further changes in any nodes output to allow the creation, evolution and evaluation of recurrent networks. However BOONE is significantly faster than the corresponding dataset generator in the case of the irregular boundaries, indicating that the technique may have significant advantages in anatomical applications were simplistic geometric constraints are not sufficient.

In future work we will continue to address the neural network modelling of discontinuous vector fields in quaternion space for anatomical joint constraint focusing on the creation of appropriately distributed datasets.

5. ACKNOWLEDGEMENTS

The authors would like to thank; Helmut Mayer (University of Salzburg) for all his advice in optimising the configuration of the NetJEN system and August Mayer (University of Salzburg) for his advice, continued development the NetJEN system.

REFERENCES

[1] W. Manurel and D. Thalmann. "Human shoulder joint modelling including scapulo-thoratic constraints and joint sinus cones." *Computers and Graphics*. vol. 24, pp. 203-218, 2000.

[2] J.D. Feikes, J.J. O'Connor, and A.B. Zavatsky. "A constraint-based approach to modelling the mobility of the human knee joint." *Journal of Biomechanics*. vol. 36, pp. 125-129, 2003.

[3] A. Watt and M. Watt. "Forward vs inverse kinematics in computer animation." in *Advanced animation and rendering techniques*, New York: ACM Press, 1992, pp. 371-384.

[4] Y. Zhang and J. Wang. "A dual neural network for constrained torque optimization of kinematically redundant manipulators." *IEEE Transactions on System, Man and Cybernetics: Part B*. vol. 32, pp. 654-662, 2002.

[5] A. D'Souza, V. S, and S. Stefan. "Learning inverse kinematics." presented at International Conference on Intelligent Robots and System, Maui, Hawaii, USA, 2001.

[6] P.M. Issacs and F.C. Micheal. "Controlling dynamic simulation with kinematic constraints, behaviour functions and inverse dynamics." *ACM Transactions on Computer Graphics*. vol. 21, pp. 215-223, 1987.

[7] J.J. Faraway, X. Zhang, and D.B. Chaffin. "Rectifying postures reconstructed from joint angles to meet constraints." *Journal of Biomechanics*. vol. 32, pp. 733-736, 1999.

[8] J. Eng and D.A. Winter. "Kinematic analysis of the lower limbs during walking: What information can be gained from a 3d model." *Journal of Biomechanics*. vol. 28, pp. 753-758, 1995.

[9] T. Furuta, T. Tawara, Y. Okumura, M. Shimizu, and K. Tomiyama. "Design and construction of a series of compact humanoid robots & development of bipedal walking control strategies." *Robotics and Autonomous Systems*. vol. 37, pp. 81-100, 2001.

[10] P. Baerlocher. "Inverse kinematics techniques for the interactive posture control of articulated figures." PhD Thesis, Ecole Polytechnique Federal De Lausanne, Lausanne, 2001.

[11] A. Watt and M. Watt. "Parameterisation of rotation." in *Advanced animation and rendering techniques*, New York: ACM Press, 1992, pp. 356-368.

[12] A. Maciel, L.P. Nedel, and C.M. Freitas. "Anatomy based joint models for virtual human skeletons." presented at IEEE Computer Animation, Geneva, Switzerland., 2002.

[13] J.U. Korein. *A geometric investigation of reach*. Massachusetts: MIT Press, 1984.

[14] A.E. Engin and S.T. Tumer. "Three dimensional kinematic modelling of the human shoulder complex part 1: Physical model & determination of joint sinus cone." *Journal of Biomechanical Engineering*. vol. 111, pp. 107-112, 1989.

[15] D.R. Wilson, J.D. Feikes, and J.J. O'Connor. "Ligaments and articular contact guide passive knee flextion." *Journal of Biomechanics*. vol. 31, pp. 1127-1136, 1998.

[16] J. Lee. "A hierarchical approach to motion analysis and synthesis for articulated figures." PhD Thesis, Korean Advanced Institute of Science and Technology, Daejeon, 2000.

[17] Q. Liu and E. Prakash, C. "The parameterization of joint rotation with the unit quaternion." presented at 7th Digital Image Computing: Techniques and Applications, Sydney, 2003.

[18] L. Herda, R. Urtasun, P. Fua, and A. Hanson. "Automatic determination of shoulder joint limits using quaternion field boundaries." *International Journal of Robotics Research*. vol. 22, pp. 419-444, 2003.

[19] M.P. Johnson. "Exploiting quaternions to support expressive interactive character motion." PhD Thesis, Massachusetts Institute of Technology, Massachusetts, 1995.

[20] Y. Zhang, J. Wang, and Y. Xia. "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits." *IEEE Transactions on Neural Networks*. vol. 14, pp. 658-667, 2003.

[21] H. Ding and S.K. Tso. "A fully neural network-based planning scheme for torque minimization of redundant manipulators." *IEEE Transactions on Industrial Electronics*. vol. 46, pp. 199-206, 1999.

[22] W.S. Tang and J. Wang. "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators." *IEEE Transactions on System, Man and Cybernetics: Part A*. vol. 23, pp. 199-206, 2001.

[23] H. Ding and J. Wang. "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators." *IEEE Transactions on System, Man and Cybernetics: Part A*. vol. 29, pp. 269-276, 1999.

[24] Y. Zhang and J. Wang. "A dual neural network for constrained torque optimization of kinematically redundant manipulators." *IEEE Transactions on System, Man and Cybernetics: Part B*. vol. 32, pp. 654-622, 2001.

[25] Y. Xia and J. Wang. "A dual neural network for kinematic control of redundant robot manipulators." *IEEE Transactions on System, Man and Cybernetics: Part B*. vol. 31, pp. 147-154, 2001.

[26] Y. Zhang, J. Wang, and Y. Xu. "A dual neural network for bi-criteria kinematic control of redundant manipulators." *IEEE Transactions on Robotics and Automation*. vol. 18, pp. 923-931, 2002.

[27] J. De Lope, T. Zarraonandia, R. Gonzalez-Carega, and D. Maravall. "Solving the inverse kinematics in humanoid robots: A neural approach." presented at 7th International Work-Conference on Artificial and Neural Networks, Menorca, 2003.

[28] Z. Mao and T.C. Hsia. "Obstacle avoidance inverse kinematics solution of redundant robots by neural networks." *Robotica*. vol. 15, pp. 3-10, 1997.

[29] K. Mehrotra, C.K. Mohan, and S. Ranka. "Introduction." in *Elements of artificial neural networks*, Massachusetts: MIT Press, 1997, pp. 1-40.

[30] R. Huber, H.A. Mayer, and R. Schwaiger. "Netgen - a parallel system generating problem-adapted topologies of artificial neural networks by means of genetic algorithms." presented at Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Frachgruppe 1.1.3, Dortmund, 1995.

[31] G. Jenkins. "Evolved neural network approximation of discontinuous vector fields in unit quaternion space (s^3) for anatomical joint constraint." PhD Thesis, University of Glamorgan, Treforest, 2007.

[32] G. Jenkins and P. Angel. "Evolved topology genralized multi-layer percerptron (gmlp) for joint constraint modelling." presented at 9th International Conference on Computer Modelling and Simulation, Oriel Collage Oxford, 2006.

[33] H.A. Mayer and R. Schwaiger. "Evolution of cubic spline activation functions for artificial neural networks." presented at 10th Portuguese Conference on Artificial Intelligence (EPIA 2001), 2001.

[34] H.A. Mayer and R. Schwaiger. "Differentiation of neuron types by evolving activation function templates for artificial neural networks." presented at World Congress on Computational Intelligence, International Joint Conference on Neural Networks, Honolulu, Hawaii, USA, 2002.

[35] H.A. Mayer and P. Maier. "Evolution of neural go players." *Osterreichische Gesellschaft fur Artificial Intelligence*. vol. 24, pp. 8-16, 2005.

[36] H.A. Mayer, M. Strapetz, and R. Fuchs. "Simultaneous evolution of structure and activation function types in generalized multi-layer perceptrons." presented at WSES International Conference on Neural Networks and Applications, Puerto De La Cruz, Tenerife, Spain, 2000.

[37] H.A. Mayer. "A taxonomy of the evolution of artificial neural systems." presented at Scientific Computing in Salzburg, 2005.

[38] H.A. Mayer. "Symbiotic co evolution of artificial neural networks and training data sets." presented at 5th International Conference on Problem Solving from Nature, Amsterdam, The Netherlands, 1998.

[39] H.A. Mayer, R. Huber, and R. Schwaiger. "Lean artificial neural networks - regularization helps evolution." presented at 2nd Nordic Workshop on Genetic Algorithms and their Applications, Vaasa, Finland, 1996.

[40] H.A. Mayer and R. Schwaiger. "Evolutionary and coevlolutionary approaches to time series prediction using generalized multi-layer perceptrons." presented at Congress on Evolutionary Computation, Washington DC, 1999.

[41] G.F. Miller, P.M. Todd, and S. Hegde, U. "Designing neural networks using genetic

algorithms." presented at Third International Conference on Genetic Algorithms, San Mateo, California, 1989.

[42] A. Zell, G. Marmier, M. Vogt, N. Mach, R. Huebner, K.U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, A. Hatzigeogiou, S. Doering, and D. Posselt. "Stuttgart neural network simulator, user manual," University of Stuttgart, Stuttgart 1994.

[43] R.E. Smith, D.E. Goldberg, and J.A. Earickson. "Sga-c: A c-language implementation of a simple genetic algorithm.," University of Alabama, Tuscaloosa TCGA 91002, May 1991.

[44] D.E. Goldberg. "Computer implementation of a genetic algorithm." in *Genetic algorithms in search, optimization, and machine learning*, D. Edward, Ed., Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989.

[45] M. Riedmiller and H. Braun. "A direct adaptive method for faster backpropagation learning: The rprop algorithm." presented at IEEE International Conference on Neural Networks, San Francisco, CA, 1993.

[46] D.E. Goldberg. "Genetic algorithms revisited: Mathematical foundations." in *Genetic algorithms in search, optimization, and machine learning*, D. Edward, Ed., Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989.

[47] G. Arfken. "Vector analysis." in *Mathematical methods for physicist*, 3rd ed, G.B. Arfken and H.J. Weber, Eds., Orlando: Academic Press, 1985.

[48] N.H.R. Goerke and R. Eckiller. "A neural network that generates attractive vector fields for robot control." presented at Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, 1996.

[49] N.H.R. Goerke, F. Kintzler, A. Rabe, D. Roggisch, and R. Eckmiller. "Controlling the khepera robot by neural network modules." presented at First International Khepera Workshop, Paderborn: HNI-Verglasschriftenreihe, 1999.

[50] Y. Kuzo, M. Mitsui, H. Kawakimi, and T. Mori. "A learning method for vector feild approximation by neural networks." presented at IEEE World Congress on Computational Intelligence, Ankhorage, AK, USA, 1998.

[51] I. Kimura, Y. Susaki, R. Kiyohara, A. Kaga, and Y. Kuroe. "Gradient-based piv using neural networks." *Journal of Visualization*. vol. 5, pp. 363-370, 2002.

[52] Y.N. Kulchin and A.V. Panova. "Neural networks for reconstruction of signal from distributed measuring system of optical amplitude sensors." *Pacific Science Review*. vol. 3, pp. 1-4, 2001.

[53] R. Grzeszczuk, Terzopoulos, D, Hinton, G. "Nuroanimator: Fast neural network emulation and control of physics-based models." presented at 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998.

[54] L. Herda, R. Urtasun, and P. Fua. " Hierarchical implicit surface joint limits for human body tracking," Computer Vision Lab, Ecole Polytechnique Federal de Lausanne (EPFL), Lusanne CH-1015, 2004.

Dr. Glenn Jenkins is a Lecturer at the Swansea Metropolitan University, specialising in Games Development and Software Engineering. He obtained his PhD in Computer Science focusing on the Simulation of Anatomical Joint Constraints using Neural Networks. His research interests include artificial neural networks their evolution and application to the simulation of anatomical joint constraints.

Dr. Paul Angel is a Senior Lecturer in the School of Computing at the University of Glamorgan, specialising in Computer Graphics, Visualisation and Software Design. His research interests include volumetric modelling and visualisation, image analysis, artificial neural networks and concurrent / parallel programming techniques. He obtained his PhD in Computer Science focusing on the application of wavelet feature extraction techniques applied to biological image data.