The Recognition and Analysis of Animate Objects using Neural Networks and Active Contour Models

Ken Tabb, Neil Davey, Rod Adams & Stella George
{K.J.Tabb, N.Davey, R.G.Adams, S.J.George}@herts.ac.uk
http://www.health.herts.ac.uk/ken/vision/
Department of Computer Science,
University of Hertfordshire, College Lane, Hatfield, Herts UK. AL10 9AB

Abstract: In this paper we describe a method for tracking walking humans in the visual field. Active contour models are used to track moving objects in a sequence of images. The resulting contours are then encoded in a scale-, location-, resolution- and control point rotation-invariant vector. These vectors are used to train and test feedforward error-backpropagation neural networks, which are able to distinguish both static and dynamic human objects from other classes of object, including horses, dogs and inanimate objects. Experimental results are presented which show the neural network's ability to successfully categorise objects which have become partially occluded. Classes of object can be distinguished by the network, and experimental results are presented which show how the representational vectors used as input patterns can be used to identify, classify and analyse the temporal behaviour of pedestrians.

Keywords: Active contour model, Snake, Pedestrian, Tracking, Shape, Neural network, Axis crossover

1 Introduction

This paper describes a novel incorporation of an active contour model with a neural network categoriser. Combined, these systems provide a means of automatically tracking a moving object, and of determining whether or not that object is human.

The detection of objects in an image or sequence of images can be achieved using a number of techniques such as optic flow, threshold-based segmentation and template matching [2, 3], although few techniques produce complete and unambiguous vector-based shapes as a result. By using active contour models [1, 6, 10], shapes are produced which can then be categorised by a neural network in terms of whether they are 'human' or 'non-human'; if methods which produce bitmaps were used instead of active contour models, the network would have to analyse the entire bitmap, and a classification of 'human' would not necessarily indicate where the human is in the image.

Alternative methods exist for determining the class of an object being tracked [5, 8], although these techniques generally rely either upon the object being totally visible throughout the process, or upon complex models of the target object being formed on the fly, requiring large amounts of computation. The method we present involves training a neural network in advance, leaving very little computation to be performed while the object is being tracked. Furthermore the neural network's generalisation skills allow for correct object classification when the target object is becoming partially occluded.

Following an overview of active contour models, we show how an active contour model can be used to track moving objects in an image. These contours are then re-represented using axis crossover vectors, making them suitable for input to a feedforward error-backpropagation neural network, such that the shape information in the contour becomes scale-, location-, resolution- and control point rotation-invariant. Experiments validating the axis crossover's ability to encode human shape information are documented, using a number of different supervised multilayer perceptron neural network architectures. MLPs were chosen as they offer a well known method of generalisation in ill-defined problem domains. In addition, MLPs require only a small operational computational overhead once trained, when compared to alternative techniques such as template matching or lookup tables. The most effective level of granularity for encoding human shape information in an axis crossover vector is then investigated and identified. The resulting supervised neural network is able to classify static simulated and real human shapes, being tracked by the active contour model, to a high degree of accuracy, even when the target object becomes partially occluded. Furthermore we show how the same neural network's output can be used to identify an object's motion idiosyncrasies, in terms of how motion is cyclical for a given

individual, how objects within the same class exhibit subtle differences in their movement, and how motion patterns between object classes differ to a much greater extent than those within an object class.

2 Tracking Objects using Active Contour Models

In this section we present an overview of active contour models, and display our implementation of Fast Snakes being used to track humans in real world complex outdoor scenarios. A broader coverage of active contour models is given in [1].

2.1 An Overview of Active Contour Models

Active Contour Models, commonly referred to as 'snakes', were originally developed to assist users in identifying objects' outlines in images or sequences of images [6]. Users would initialise a contour loosely around the target object in an image by using a mouse to place the contour's control points in the image. The model would then use energy minimisation to lock the contour onto the object's outline, providing a faster and often more accurate means of obtaining the exact shape of the object than by users abstracting the object's shape by manually identifying its edges.

Snakes include tracking properties so that if a contour is initialised around a target object in the first frame of a sequence of images, the snake can then obtain that object's shape not only in the first frame of the image sequence, but in all frames. This allows an object to be tracked as it moves around, even if it changes shape during the image sequence [6]. Once the target object has been locked onto by the snake, the resultant contour, or contours if tracking an object through a sequence of images, can then be used in higher level cognitive processes, for example human or artificial intelligence examination of the object's shape [9].

A snake itself consists of two parts: a list of control point coordinates, and an energy function. The control points form the shape of the contour being moved in the image, each control point having its own (x,y) location in the image. The energy function contains a set of mathematical criteria, or 'rules', which govern the movement of control points. When displaying the contour on-screen for human interpretation, the control points are usually linked together to enable the user to identify which control

points are adjacent to which, and therefore the shape of the contour. This graphical linking of control points usually takes the form of either a polygon or a spline being approximated to the control point locations [Figure 1]. In reality, the links drawn are purely for visualisation and serve no purpose to the model itself. The snake contains only the (x,y) locations of its control points, and the mathematical rules of movement, encapsulated in the energy function. The energy function is defined by the user; by changing the definition of the energy function, or by changing parameters within the function, the contour will exhibit different behaviours, and consequently the contour will move to different areas of the image. The energy function definition is therefore critical to the active contour model's success at being able to lock onto particular types of object.

The original model depended upon user interaction to guide the snake; without the user 'steering' the snake, it would catch on local minima in the image and would form inaccurate shapes as a result [6]. In our work, we have adopted the Fast Snake model [10], as it allows for automated tracking of objects, and therefore minimal user interaction, given a suitable energy function. A comparison of the original and Fast Snake models is given in [11].

In our implementation of active contour models for tracking human objects, the model's energy function comprises two types of energy: internal energy and image energy. Internal energy is dependent upon the geometric shape which the contour has formed, and is independent of any features in the image which have been locked onto by the control points. Examples of internal energy include the contour's curvature and surface area (size). A control point's image energy, on the other hand, is entirely dependent on the image features which the control point has locked onto, and is independent of both the shape of the contour and of the image features which the other control points have locked onto. Image energy might include pixel contrast value, to allow the snake to lock onto edges in the image, and pixel colour. For example, by decreasing control point energy whenever a control point is near red areas of the image, the control points, and therefore contour, will be attracted towards red objects. In addition, if areas of high contrast in the image are also defined as having low energy, then the areas of lowest energy will be the edges of red areas in the image. Consequently the contour will be attracted towards the edges of red objects in the image, allowing the shape of red objects to be identified.

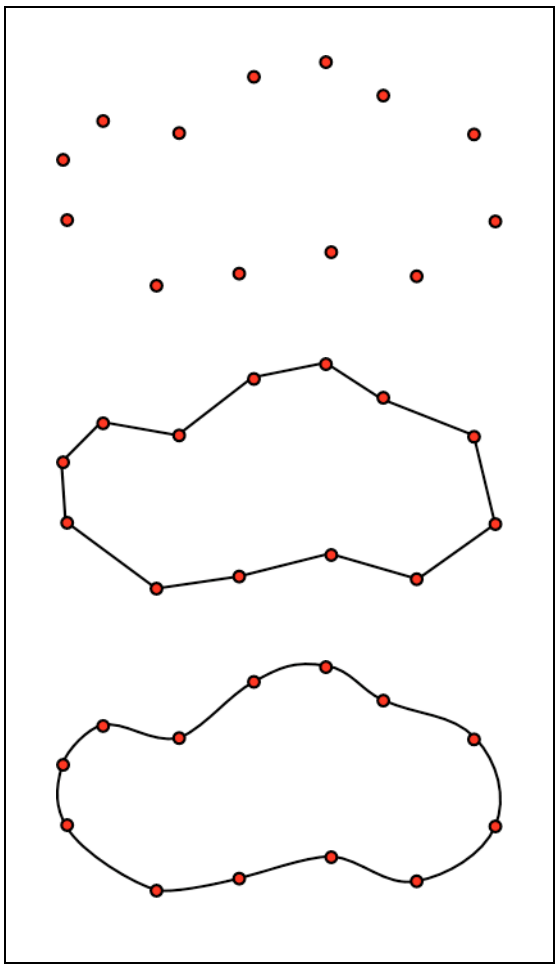

Figure 3. Alternative visual representations of contours. Each of the three visual representations in this figure are formed from the same contour, and therefore the same set of control points. **[Top]** The control points which make up the contour. **[Middle]** The control points joined using straight lines, forming a polygonal contour. **[Bottom]** Interpolated Bézier splines form a more organic shape, at extra computational cost

In addition to the energy function criteria, each criteria has a weighting parameter. This allows the user to set the influence of a particular energy function criterion over and above the other criteria. For example, it might be more important in some tasks to lock onto edges of red regions in the image than for the contour to maintain a certain size.

In some active contour model implementations, the number of control points in a model can grow or shrink as needed during the contour's movement [1]. In our implementation, all active contours keep the same number of control points from the moment they are introduced in an image to the moment they are removed; different

active contours may contain a different number of control points, but a given active contour will keep the same number of control points throughout its lifecycle.

The energy function for a given control point in our implementation can be defined as:

$$controlPointEnergy_i = \alpha \cdot internalEnergy_i + \beta \cdot imageEnergy_i \quad \textbf{[1]}$$

where i represents the given control point, $controlPointEnergy_i$ represents that control point's total energy level, and $internalEnergy_i$, and $imageEnergy_i$, represent the control point's internal and image energy levels respectively. α and β represent the user-defined weighting parameters for the internal and image energy values respectively

Continuity energy is part of the snake's internal energy, and aims to make all control points equidistant from their neighbours. By spreading the control points out around the contour, the overall shape of the desired outline can be identified; if control points were to bunch up, then certain areas of the desired outline would be identified at a high resolution, but other parts of it would be sparsely represented [Figure 2]. Of course, if the continuity energy weighting parameter is sufficiently small, then the influence of continuity energy might not be significant enough to prevent control points from bunching up.

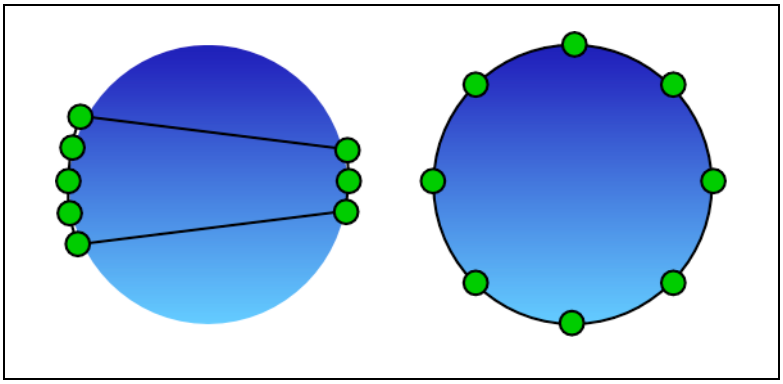

Figure 2: **[Left]** When an ACM's control points are unevenly spaced around an object, parts of the object's shape may be excluded from the ACM. **[Right]** Evenly spacing control points around an object enables its shape to be accurately encapsulated

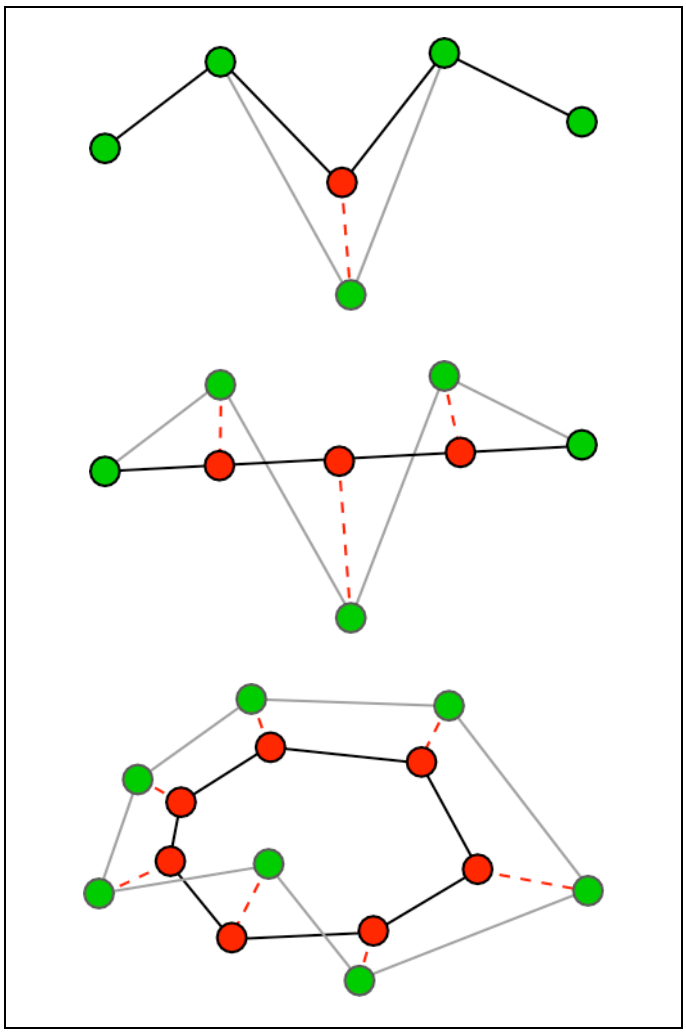

Figure 3: Curvature energy enables or disables corners from forming on the contour by positioning a control point directly between its neighbours. **[Top]** The effect of reducing curvature energy on one control point in an open-ended contour; the middle control point (green) is moved to a position nearer its two neighbours (red). **[Middle]** The effect on the entire open-ended contour following several iterations of the curvature energy function; the contour is seen to straighten out. **[Bottom]** The effect on a closed loop contour following several iterations around the contour; the contour shrinks slightly, but fits a smoother spline curve

Curvature energy is the other component of the snake's internal energy. It aims to govern the frequency and sharpness of any corners formed in the contour [Figure 3]. By controlling the circumstances in which corners are allowed to form on the snake, a trade-off is available between the complexity of shape able to be formed by the snake, and the likelihood of control points snagging on noise in the image [11]. Having a sufficiently large number of control points on the snake ensures that a snake can still

form a complex shape even with a high value of β, provided curvature between adjacent control points is low [Figure 4].

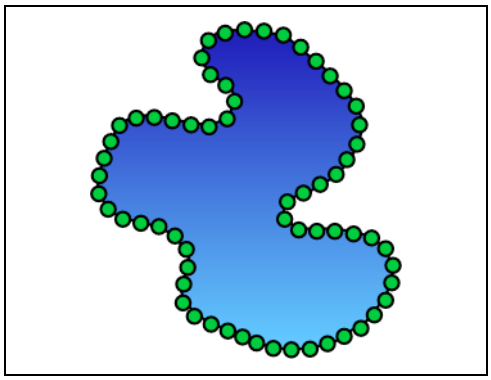

Figure 4: With sufficient control points, curvature energy need not prevent a snake from forming a complex shape; provided curvature between adjacent control points is low, a snake can still form a complex shape

Image energy in our implementation is based on a number of image preprocessing algorithms [Figure 5]. Firstly, edges are detected in both the current image and a previous 'reference' frame using a Sobel edge mask [2]. Motion detection is then performed on the two edge maps, by means of differencing the edge maps. At this point only moving objects remain in the image. 'Blobs', that is, regions that are too small to be considered useful, are then masked out, removing most of the noise caused by leaves blowing, water rippling, or distant moving objects.

To move the active contour, each of its control points in turn has its energy minimised. This involves, for each control point around the contour, searching the control point's neighbouring locations to determine, using the energy function, whether any of these locations would offer a lower energy value for the control point than its current location. Having searched its neighbourhood, the control point is then moved to the location in the neighbourhood offering the lowest energy [Figure 6]. In instances when a control point is already at the location with the lowest energy, it is not moved as doing so would increase its energy. Once every control point in the contour has had its energy minimised in this way, the contour is said to have moved. This process is repeated until the snake is relaxed, that is, until a sufficiently large percentage of the

control points no longer move to other areas in their neighbourhood, whereby the snake remains relatively still between iterations.

Figure 5: The image enhancement process prior to the introduction of active contour models to the image. **[Top]** The source video frame. **[Middle]** The image after being passed through a Sobel edge mask. **[Bottom]** After motion detection and blob removal procedures have been performed on the middle image, only the edges of moving objects remain in the image. This simplifies the active contour model's task, as most local minima have by this stage been removed from the image

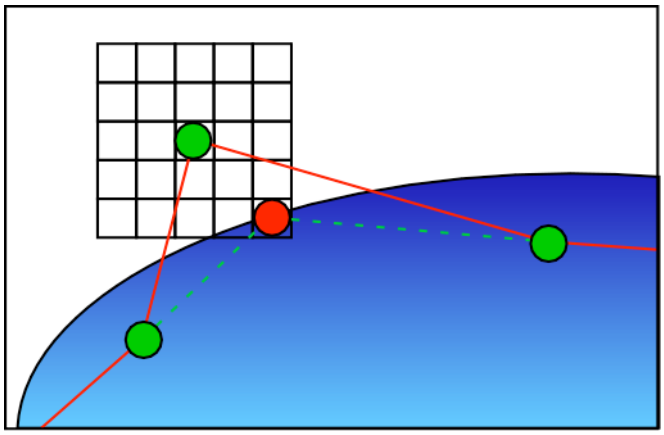

Figure 6: Control point movement is determined by measuring the energy for a given control point's neighbours, and determining if any of those locations would offer lower energy, that is, a better position. If so, the control point is then moved to that location. If there is no advantage to moving, a control point remains in its current location

2.2 Tracking Humans using Snakes

In our work, we use snakes to track non-occluded humans as they move around the visual field. Using the energy function described above, snakes can be seen to be reasonably successful at tracking moving humans, provided the human is not occluded [Figure 7].

Figure 8 shows the results of a snake tracking a sample human over 105 frames of video footage. No objective measure exists of an active contour's success at tracking objects [1], but the model can be seen to be successfully locating the target human outlines.

As can be seen from [Figure 8], the snake only obtains the outer edge of an object; 'holes' in between the human's legs do not form part of the contour (as illustrated in frame 30 of the figure). Despite the variation that humans adopt as they walk, their outline was identified in 30 different cases.

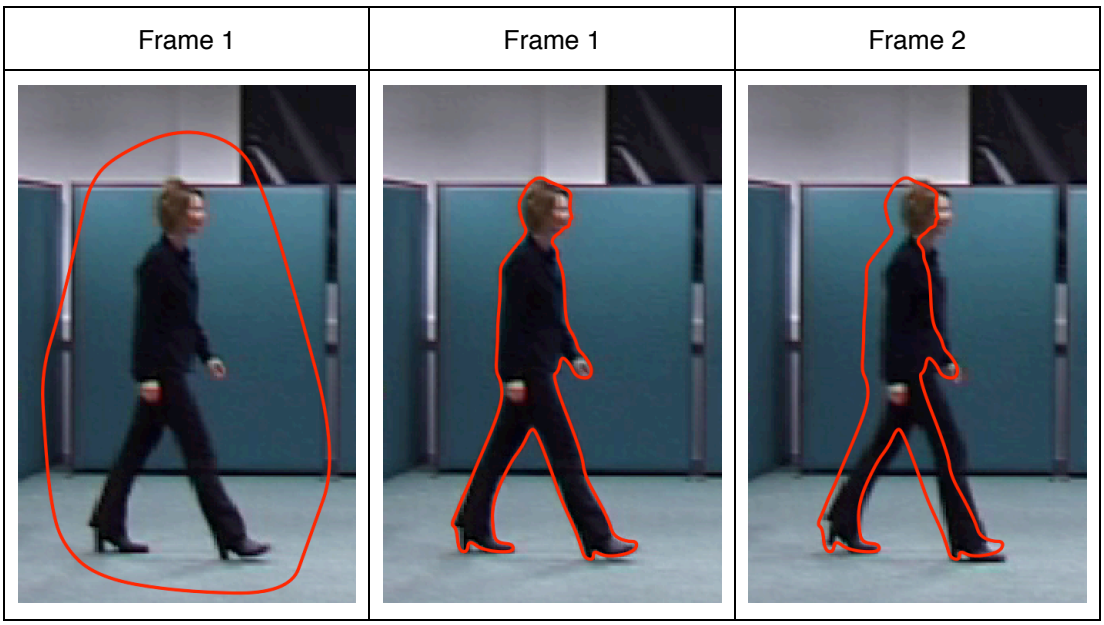

Figure 7: A snake relaxing on a human shape. **[Left]** The user initialises a contour around the target human. **[Middle]** Minimising the snake's energy function forces the snake to relax onto the human outline. **[Right]** Once relaxed, the snake is initialised in the next frame using its relaxed position as a starting point. Its energy is then minimised again to relax it onto the human's new position

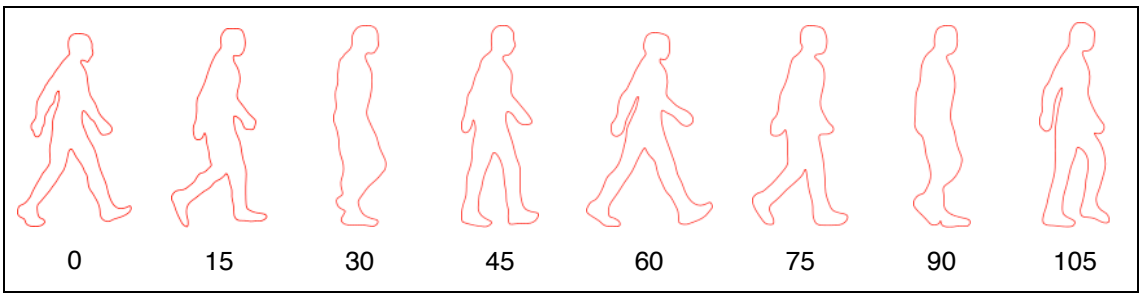

Figure 8: A target human detected and tracked through a sequence of 105 frames using an active contour. The frame numbers are shown below each frame; intermediate frames have been omitted

In cases where the human becomes either partially or totally occluded, for instance if they walk behind a tree, or if another human walks between the target human and the camera, the snake will fail in obtaining the target shape. This is because the image preprocessing stages will not be able to enhance the target object's outline if all or part of that outline is not in the image to begin with. Typically in a situation where

the target human walks behind something, the snake will remain tracking the visible part of the human until they disappear behind the obstruction, at which point it is left with nothing to track, resulting in unpredictable behaviour. Often the snake collapses in on itself at the point where the target object disappeared. The snake is rarely able to regain tracking the target human when they reappear elsewhere in the image, however, as the human has usually moved far enough across the image that the snake cannot find them in any of its control points' neighbourhoods.

3 Using Snakes and Neural Networks for Shape Classification

In this section we discuss the issues preventing an active contour from being used unaltered as an input pattern to a neural network. We propose a solution in the form of the axis crossover vector, and identify the most appropriate level of granularity for encoding human shape information, following a series of systematic experiments with neural networks and axis crossover vectors.

3.1 Obtaining Generic Shape Descriptions from Snakes

Snakes themselves have no way of determining what type of object they are tracking, they merely track visible outlines in the image, irrespective of the object(s) that the outline belongs to. In order to determine whether or not the object being tracked by a snake is human, the shape of the object somehow needs to be analysed. Due to the large variation of human poses and subsequent shapes, the wide variety of non-human objects which may be encountered in real world situations, and the general ill-defined nature of the problem, we have opted to use a feedforward error-backpropagation neural network as the means by which an object being tracked using a snake is classified human or non-human. More details of the neural network, and of experiments involving the network, are presented in the following sections.

As the snake has already obtained the shape of the target object it would be convenient, and more computationally efficient, to analyse the snake's shape rather than obtain the original object's shape from the image again using a second technique.

An active contour stores its shape information in a vector of paired (x,y) coordinates, where a given pair of coordinates in the vector represents a given control

point's (x,y) location in the image. Consequently the length of the vector corresponds to the number of control points defining the contour.

Absolute image coordinates are inherent in the contour definition, thus the native snake representation exhibits many undesirable qualities when used as a generic shape description. Such contours are not location-invariant; two equally shaped and sized contours residing in different parts of the image will result in different coordinate vectors. Additionally the vectors of contours which have the same shape but different sizes will also differ, thus the contour's native representation is not scale-invariant either. Thirdly, if one contour has more control points than another equally shaped contour, they will have different vector lengths, making the vector dependent upon the resolution of the contour, that is, upon the number of points defining the shape. Finally, if two contours are of the same shape, in the same image location, and have the same number of control points, but each contour's first control point is on a different part of the outline, their vectors will be in a different order. All of these factors make the comparison of contours difficult.

In order to use relaxed snakes' contours as input data for a neural network performing a shape analysis task, these qualities need to be removed. This re-representation therefore must provide for scale-, location-, resolution- and rotation-invariance if it is to be useful as a generic shape description. Without removing these factors, a shape's size or location in an image will still be intertwined in the shape representation, complicating any analysis. Furthermore the representation needs to remove the pairing of data, so that the neural network is not expected to have to group each control point's x and y locations together. This in itself presents a challenge when devising an input pattern representation for the neural network.

The solution which we have developed is the axis crossover vector [9]. The centre of the contour is calculated, which in our implementation is simply the mean control point location. From that centre, a pre-defined number of axes are projected outwards at specified angles, to the furthest edges of the contour [Figure 9]. The distance from the contour's centre to its furthest edge along that axis is then stored in a vector [Figure 10]. The vector length is equal to the number of axes being projected. This allows the vector to be location-invariant, as no image locations are stored in the axis crossover vector, only distances from the centre of the shape to its edges, along the axes. In addition the vector is resolution-invariant, that is, independent of the number of control points on the

contour. Also, it does not matter where on the contour the first control point resides, and so is rotation-invariant.

Once all of the axes have been measured and their distances stored in the vector, the vector is normalised. This ensures that the vector is scale-invariant, as the largest value in the vector will at this point be 1.0, which will be the longest of the axes projected.

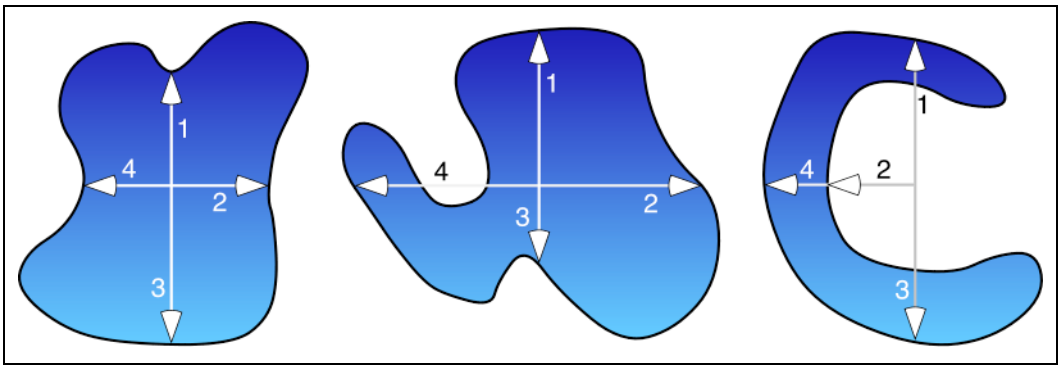

Figure 9: The axis crossover representation. **[Left]** The axis crossover being applied to a closed contour using four axes; the lengths of the four arrows form the contour's axis crossover representation. **[Middle]** In situations where an axis crosses over the contour's edge more than once, the longest distance is used. **[Right]** In situations where the contour's centre is not within its boundary, opposite axes may project in the same direction, as with axes 2 and 4; in this example, axis 2 would contain a negative value

The resulting normalised vector can then be used as a training or test input pattern for a neural network, with each vector element being a different input neuron's input [Figure 11]. The only limitation to the axis crossover representation when used with neural networks is that the number of axes must be equal to the size of input layer in the neural network. For this reason, all axis crossover vectors used to train or test a given neural network must all contain the same number of axes. Consequently, in order to ascertain the most appropriate number of axes to use for representing a given object class' shapes, it was necessary to develop several neural networks, each with a different input layer size, and to determine which network gave the best results.

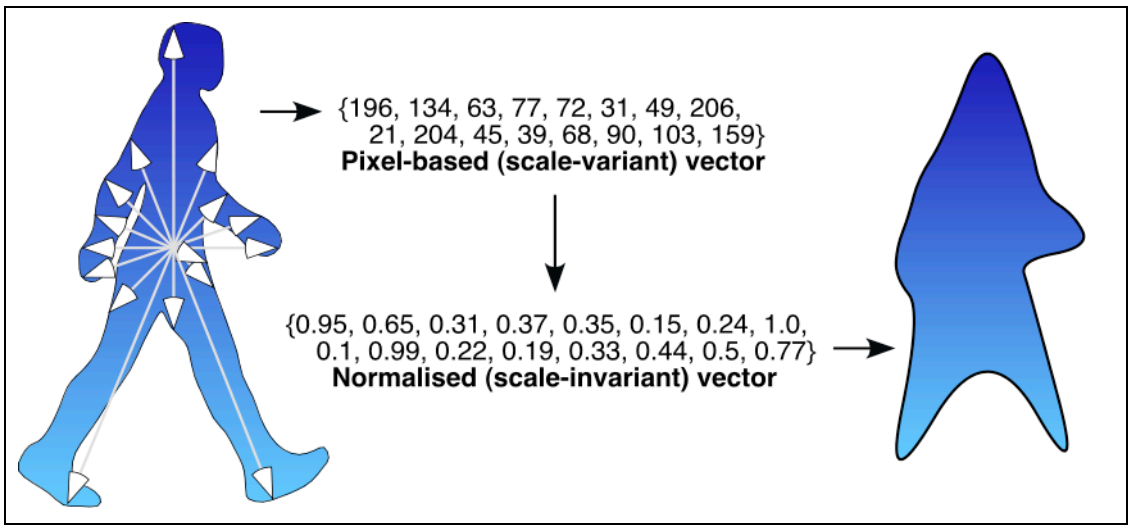

Figure 10: A pedestrian contour being re-represented as a 16-axis crossover vector. **[Left]** The contour has 16 equally spaced axes projected from its centre to its edges, giving 16 measurements in pixels **[top vector]**. **[Bottom vector]** This vector is then normalised to make it scale-invariant, providing a compact representation for use in neural networks, which is visualised as a Bézier spline on the right. In this figure, the contour's centrepoint and axis measurements are for illustration purposes only

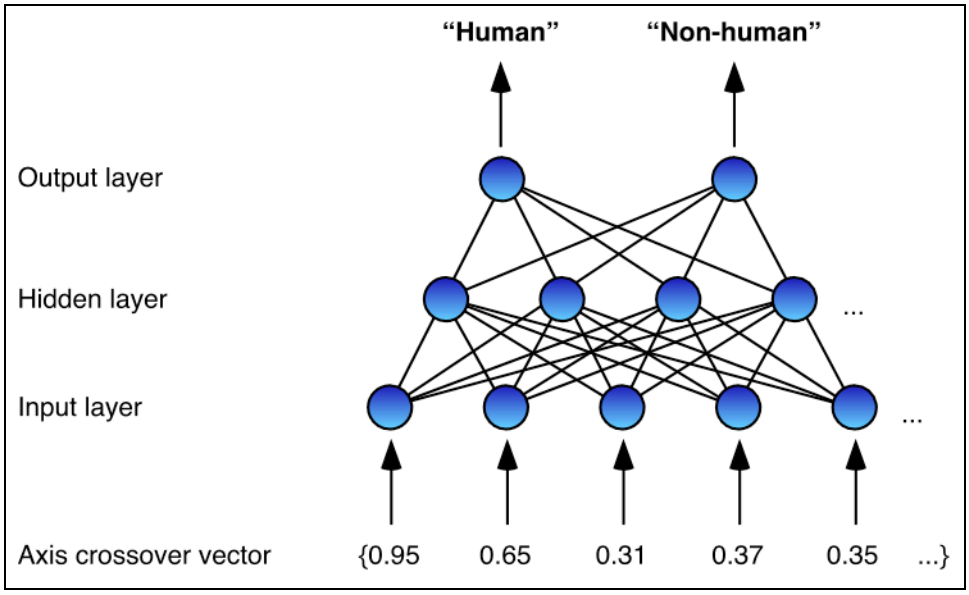

Figure 11: Axis crossover vectors as input patterns for neural networks. A given vector element maps onto a given input unit in the neural network, thus the size of input layer and axis crossover vector must equate

In our studies involving deformable objects' shapes, we do not know in advance what pose or orientation the human will take. For this reason, we project all axes evenly around 360° [as in Figure 10]. In addition we wish to be able to differentiate entire human shapes from entire non-human shapes. In some situations where an object's shape and orientation may be well known prior to analysis and where only certain parts of an object need to be inspected by the neural network, for example in well defined production line systems, it may be profitable to focus the axes upon a particular area of the object, rather than to focus on the entire shape of the object. Projecting axes at custom angles during encoding may generate a more compact and / or useful representation for the object class being detected [Figure 12]. Similarly, the granularity of shape description can be tailored to a specific task by using a larger or smaller number of axes. However, using more axes in a representation results in a more complex categorisation task for the neural network.

Whichever configuration of axis crossover vector is used, it should be remembered that, in order to be able to use these vectors to compare and contrast shapes, all axis crossover vectors generated for an experiment must have had the same configuration imposed upon them, so that, for example, the same number of axes are projected at the same angles for every vector being compared.

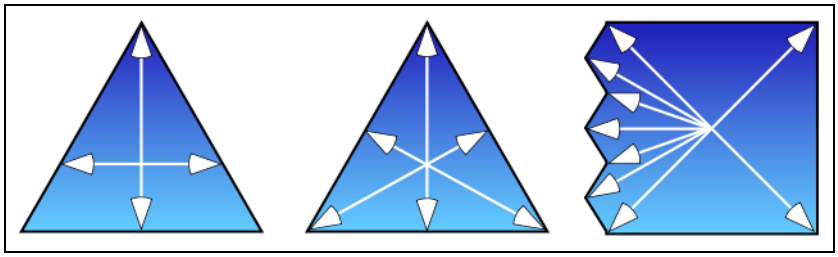

Figure 12: The axis crossover representation can be customised for encoding particular object classes. **[Left and middle]** Encoding contours using 6 axes may generate a more robust representation of an object's triangular properties than if only 4 axes are used. **[Right]** Axes can be projected at irregularly spaced angles if necessary, allowing areas of interest to be encoded in detail whilst omitting irrelevant areas from the representation

3.2 Verification of Axis Crossover Vectors as Generic Shape Descriptors

It was necessary to test whether or not a neural network could distinguish one group of crossover vectors (pedestrians) from other groups of crossover vectors (non-pedestrians). The vectors were tested with simple hidden layer backpropagation networks as the task was vector categorisation.

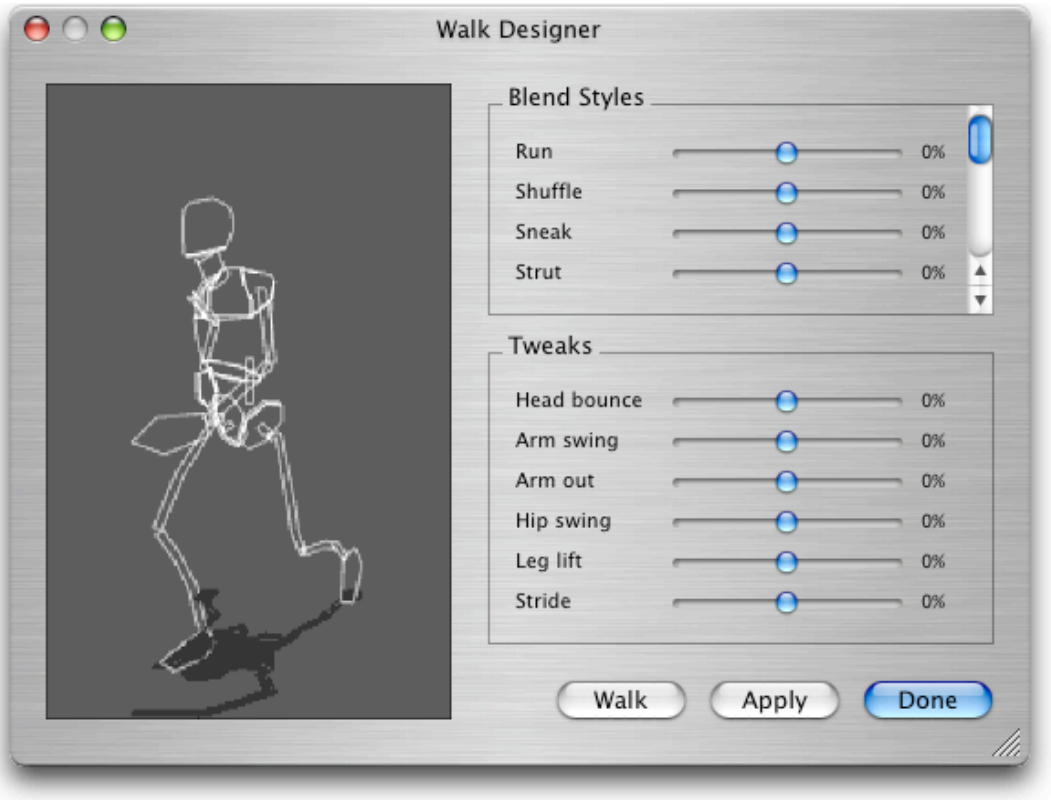

Figure 13: Using 3D modelling and animation software allows a fine degree of control over many movement and / or shape parameters of a human, such as gender, height, weight, age, gait and direction of movement. Inverse kinematics features ensure realistic movement, allowing customisable training and test sets to be generated

In order to classify shapes as human or non-human using a supervised neural network, it is necessary to obtain a training and test set of examples and counter examples. All objects in the training and test sets of experiments reported in this section were simulated using 3D modelling and animation software [Figure 13]. This allowed for much more control over objects, and meant in the case of animate objects such as

humans, that their gender, height, weight, age, direction of movement, and gait could be finely controlled.

The axis crossover representation allows for different numbers of axes to be used in the contour representation. Having fewer axes simplifies the neural network's task, however enough axes need to be used that the pedestrian qualities of the contours are encapsulated in the vectors, so that they can be differentiated from the non-pedestrian vectors. It was decided to test several different numbers of axes used in the representations, which in turn meant testing several neural networks, each with as many input units as there were axes in the representations. It was hoped that these experiments would identify the optimal number of axes to use in representing the particular class of contours relevant to this project. Initially neural networks with a single output unit were experimented on (section 3.2.1), as only two output values were needed: 'pedestrian' and 'non pedestrian'. Following these experiments, double output unit networks were used (sections 3.2.2 and 3.2.3) so that categorisation confidence values, which are described in section 3.2.4, could be analysed. In all experiments, network output was allowed some lenience; an output of 0 - 0.2 was classed '0', and an output of 0.8 - 1 was classed '1'.

3.2.1 Single output unit network experiments

The networks were trained with a range of different hidden layers, to allow the network with the optimal generalisation skills to be identified. The training set contained 150 pedestrian vectors and 150 non-pedestrian vectors. Training was stopped when the network had reached 15% or less error. It was then tested with 10 unseen pedestrian and 10 unseen non-pedestrian vectors. The non-pedestrian vectors in both training and test patterns were of indoor objects, for example lampshades and teapots.

Figure 14 shows the average results of the various single output unit networks when tested with the 20 unseen vectors (10 pedestrian and 10 non-pedestrian); each network had been trained and tested 10 times using different initial weight matrices. Of note is that the network which used the most complex contour representation being tested (24-axis) failed to learn the task adequately, irrespective of the size of its hidden layer. It can be seen from the graph that networks trained on 16-axis vectors were most successful at classification.

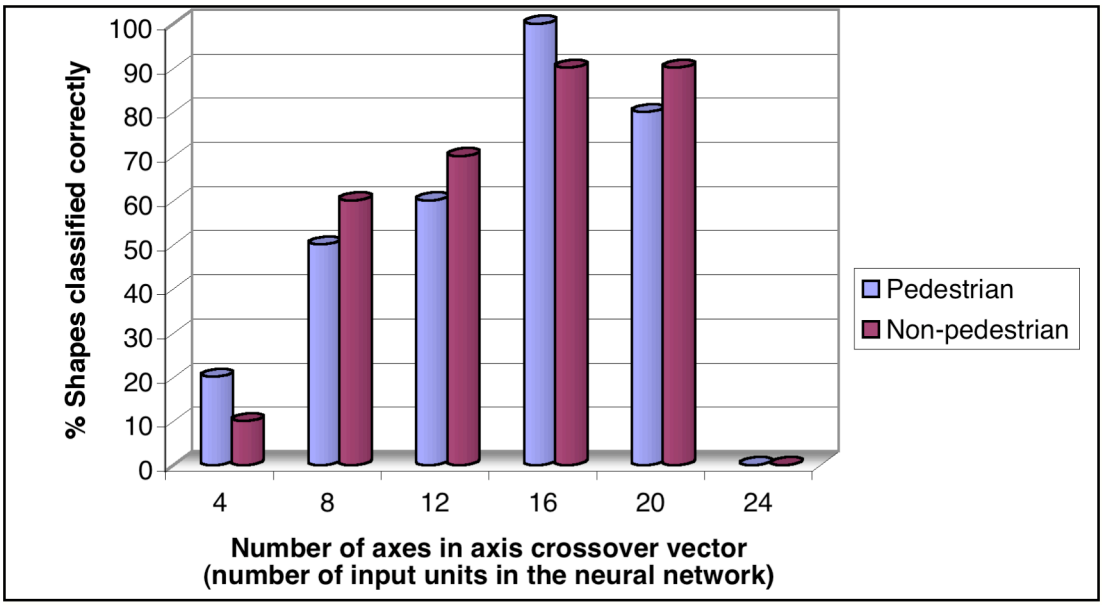

Figure 14: Results from experiments on single output unit neural networks trained using simulated human and simulated indoor non-human shapes. Each score represents the averaged score of ten identical neural networks, each trained with a different initial weight matrix. Experiments were run using 4-, 8-, 12-, 16-, 20- and 24-axis crossover representations, although the network which used 24 axes failed to learn the training data. Classifications were said to be correct if the output values were between 0.8 - 1.0 for a desired output of '1', and 0.0 - 0.2 for a desired output of '0'

3.2.2 Double output unit network experiments with indoor non-pedestrian data

The same experiments, training and test data described in section 3.2.1 were repeated with a double output unit architecture, with previously-desired outputs of '0' now becoming '1 0', and previously-desired outputs of '1' now becoming '0 1'.

The results can be seen in [Figure 15]. Although all of the networks learnt the task, their general performance was worse than the single output unit networks. As with the single output unit results, networks trained with 16-axis vectors were most successful (although the optimal number of hidden units differed from the single unit networks).

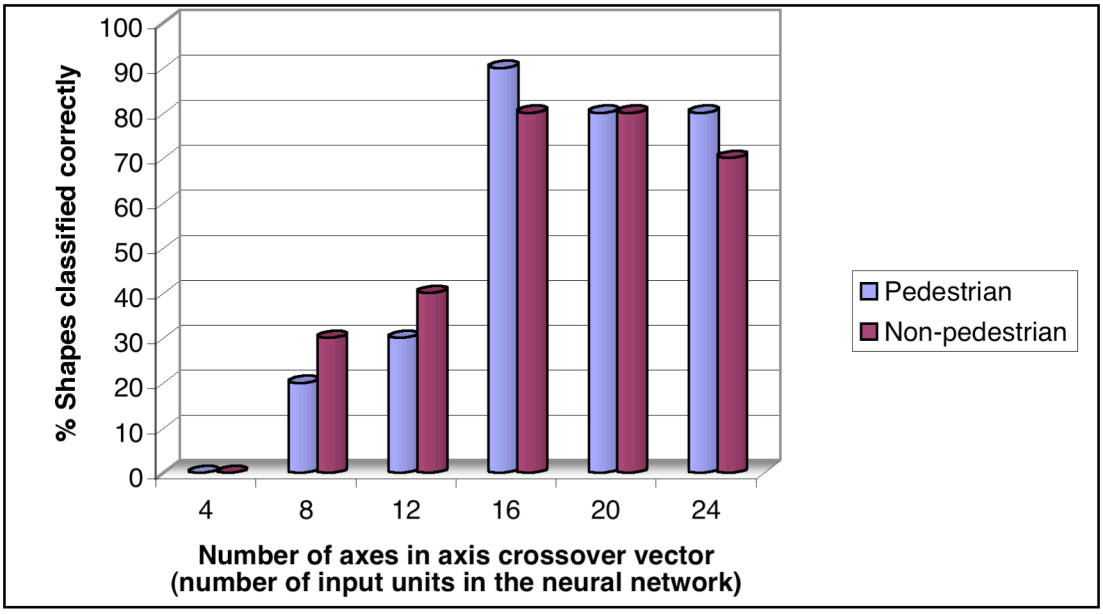

Figure 15: Results from experiments on double output unit neural networks trained using simulated human and simulated indoor non-human shapes. Each score represents the averaged score of ten identical neural networks, each trained with a different initial weight matrix. Experiments were run using 4-, 8-, 12-, 16-, 20- and 24-axis crossover representations. Classifications were said to be correct if the output values were between 0.8 - 1.0 and 0.0 - 0.2 respectively for a desired output of '1 0', and vice versa

3.2.3 Double output unit network experiments with outdoor non-pedestrian data

At this stage in the experimentation the axis crossover representation had been shown to be sufficiently descriptive of pedestrian shapes. Nevertheless the project involves outdoor pedestrian scenes, so it was necessary to use more relevant non-pedestrian objects, such as cars, streetlights and traffic lights; the 150 indoor object vectors used in training were changed for 150 outdoor object vectors accordingly, as were the 10 unseen non-pedestrian vectors used in the test set. Apart from the different non-pedestrian data used, the experiments and neural network architecture were the same as for section 3.2.2.

The results of the double output unit networks can be seen in [Figure 16]. The networks are able to classify pedestrian vectors more accurately than non-pedestrian vectors. Interestingly the best scores were still obtained by those networks which had used 16 axes in their crossover representations. It can be seen from the graph that

networks trained on 16-axis vectors were able to classify 90% of unseen human shapes and 60% of unseen non-human shapes correctly.

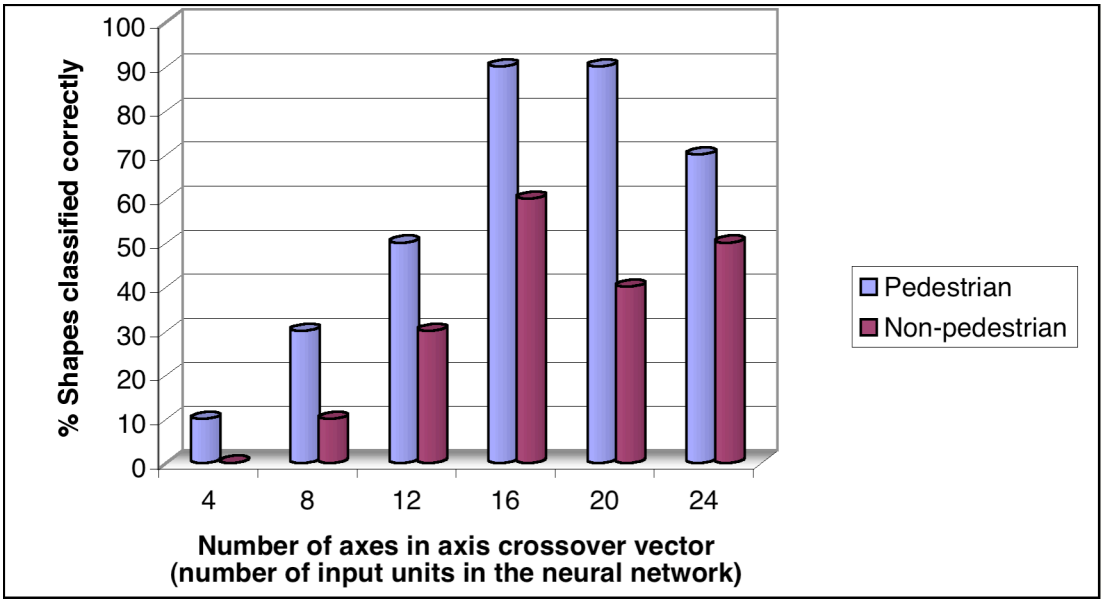

Figure 16: Results from experiments on double output unit neural networks trained using simulated human and simulated outdoor non-human shapes. Each score represents the averaged score of ten identical neural networks, each trained with a different initial weight matrix. Experiments were run using 4-, 8-, 12-, 16-, 20- and 24-axis crossover representations. Classifications were said to be correct if the output values were between 0.8 - 1.0 and 0.0 - 0.2 respectively for a desired output of '1 0', and vice versa

3.2.4 Analysis and Discussion

For the purposes of encapsulating human shape properties, we have found that 16 axes per vector is sufficient [9]. Less than 16 axes prevents the neural network from being able to accurately discriminate between human and non-human objects, and using more than 16 axes gave no benefit in performance whilst requiring more epochs before the network had finished learning. The neural network with 16 input units was therefore used henceforth.

Although the neural network with 16 input neurons and 1 output neuron had outperformed its 2 output unit counterpart, it was difficult to establish why the single

output network had misclassified its incorrect answers. For instance, an output of 0.5 could mean that the input shape, when compared to the training set, had both strong 'human' and strong 'non-human' characteristics, or it could mean that it had neither. It was felt that the single output network's ambiguity made it difficult to determine which types of extra and/or replacement training patterns would be needed.

Conversely, it was felt that the 2 output unit network could be used to produce more qualitative information, which would more clearly identify any weaknesses or omissions in the training set. It was decided to determine the confidence value from the two output units' values as this could be used as a threshold for identifying how 'pedestrian' an object is. A confidence value of a network's classification can be obtained by differencing its two output units' values. For instance, a confidence value of 0.0 indicates that the shape was as 'human' as it was 'non-human'. A confidence value of +1.0 indicates strong 'human' and weak 'non-human' characteristics, whereas a confidence value of –1.0 indicates weak 'human' and strong 'non-human' characteristics.

To test the chosen network's confidence in its categorisations, it was necessary to look at the average difference between the values output by both of its output units during the experiments described in section 3.2.3 to see how 'confident' it was that a pedestrian vector was pedestrian and that a non-pedestrian vector was not.

Figure 17 shows the average results for the 10 pedestrian and 10 outdoor non-pedestrian vectors. The network's output units are split into a 'non-pedestrian' unit and a 'pedestrian' unit (labelled OU1 and OU2 respectively in [Figure 17]) which should ideally fire mutually exclusively of one another, as something cannot be both pedestrian and non-pedestrian. However, as was found in section 3.2.3, the network is not 100% accurate, resulting in some uncertainties about the answers it gives. Nevertheless a clear divide can be seen between the values output when classifying a pedestrian vector correctly versus classifying a non-pedestrian vector correctly.

The average confidence value when classifying a human shape correctly is 0.81 (the difference between the two output unit values), with 1.0 representing complete confidence that the shape is human, whereas the average confidence value when classifying a non-human shape correctly is -0.65, with -1.0 representing complete confidence that the shape is non-human. A confidence value of 0 represents the least confident classification. The confidence values shown in [Figure 17] reflect the trend

seen in [Figure 16], that is, the network is more accurate at classifying human shapes than non-human shapes. In particular, the average output values when given non-human shapes only just fall within the 'classified correctly' zones of 0 - 0.2 and 0.8 - 1.

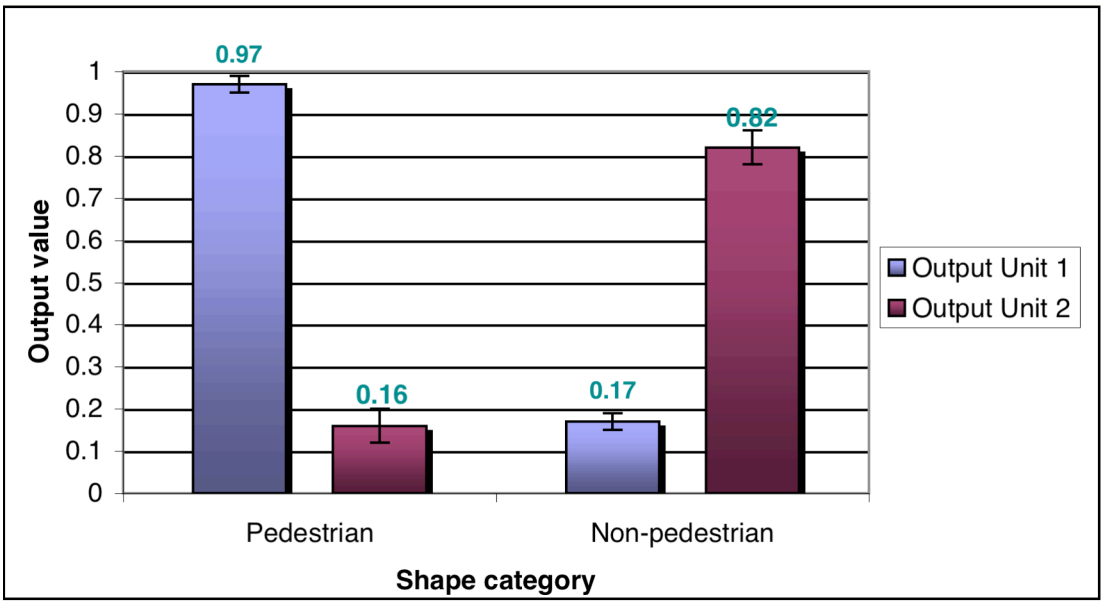

Figure 17: Mean test set classification results, from experiments on ten identical double output unit neural networks trained using simulated human and simulated outdoor non-human shapes, each trained with a different initial weight matrix

The choice of counter examples is clearly a deciding factor in the accuracy of the system; the networks were less accurate at categorising pedestrian vs. outdoor non-pedestrian objects than they were with indoor non pedestrian objects. Nevertheless we decided to remain using outdoor counter examples, as these objects were more likely to be encountered in the actual visual field being used.

4 Classification of Static Objects using Neural Networks

In this section we investigate how the shape representation developed in sections 2 and 3 can be used to produce neural networks that can distinguish between a variety

of simulated and real object classes. The network developed is able to continue to distinguish human objects even when those objects are subjected to partial occlusion.

This section uses exclusively a network with 16 input units and 2 output units, where one output unit was trained to identify human shapes, and the other non-human shapes.

4.1 Simulated and Real Object Classification

The neural network developed in the previous section was re-trained, this time with 400 examples each of simulated human and simulated non-human shapes in the training set [Figure 18]. The non-human shapes consisted of inanimate outdoor objects such as trees, cars and streetlights; and deformable animate objects, namely shapes of dogs and horses, generated using the software described in section 3.2. All non-human shapes were trained to produce a 'non-human' classification. Each shape in the training set had a snake locked onto it, and the snake's contour was then re-represented as a 16-axis crossover vector, which was used as the input pattern. The network was trained to within an error margin of 0.05.

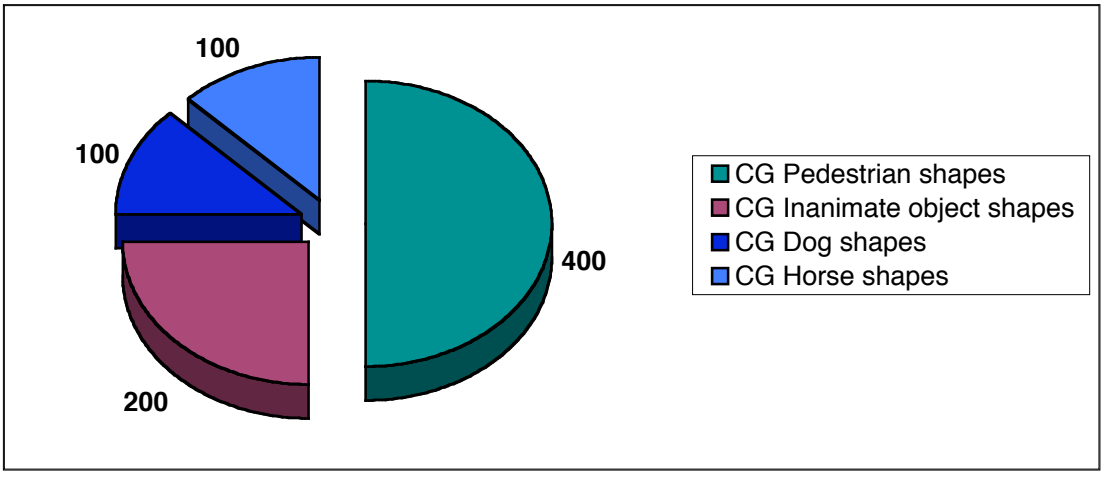

Figure 18: Training set patterns for the static object classification experiments. Each pattern was generated by detecting an object in a static image using an active contour model, and converting that contour into a 16-axis crossover vector. All shapes used to train the neural network were simulated

The test set for this categorisation experiment contained 100 unseen simulated human shapes, 100 unseen real human shapes, and 100 unseen simulated non-human shapes [Figure 19]. The real human shapes were obtained from complex outdoor scenes, as illustrated in [Figure 7], and included male and female humans of varying heights, weights and ages. Within the non-human test patterns, a previously unseen object class was introduced to the network: velociraptors. Velociraptors were used as they constituted unseen non-human bipeds, and so could further determine how distinct the network considered human objects to be.

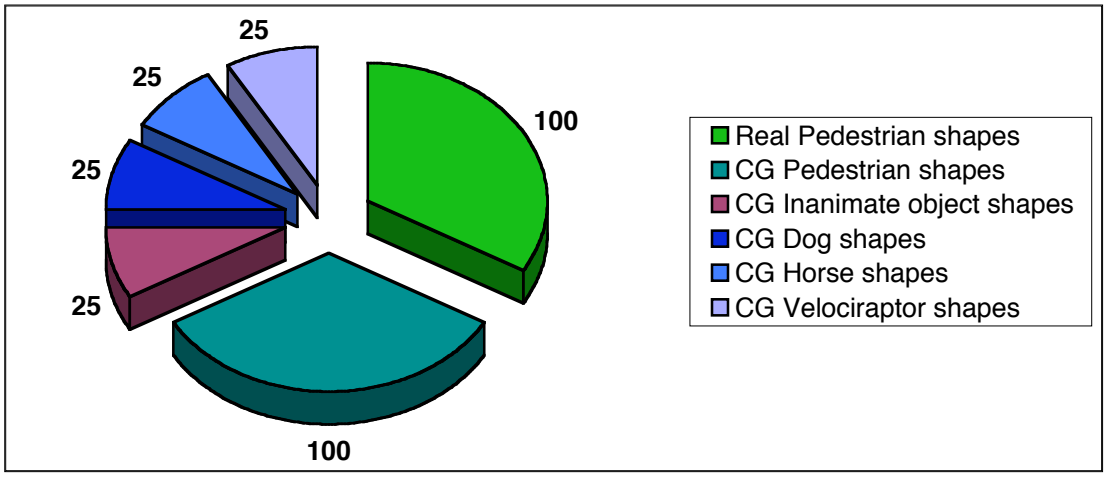

Figure 19: Test data set for the static object categorisation experiment. All shapes used as test patterns were previously unseen by the neural network. In addition the previously unseen velociraptor object class was introduced, to test the network's generalisation abilities when presented with a new bipedal object class

The results in [Figure 20] show that the network clearly identifies unseen simulated humans as 'human' with a high level of confidence, and unseen simulated dogs and horses as 'non-human' with a high, albeit lesser, level of confidence. The simulated velociraptors are classified as more human that non-human, but the network still clearly distinguishes between the simulated humans and the simulated velociraptors, with mean confidence values of 0.94 and 0.63, respectively. Real humans were not classified as accurately as simulated humans, but were nevertheless categorised as 'human' with more confidence than the simulated velociraptors,

obtaining a mean confidence value of 0.69. It is highly significant that the network, trained only on simulated human / non-human shapes, can then correctly classify real humans.

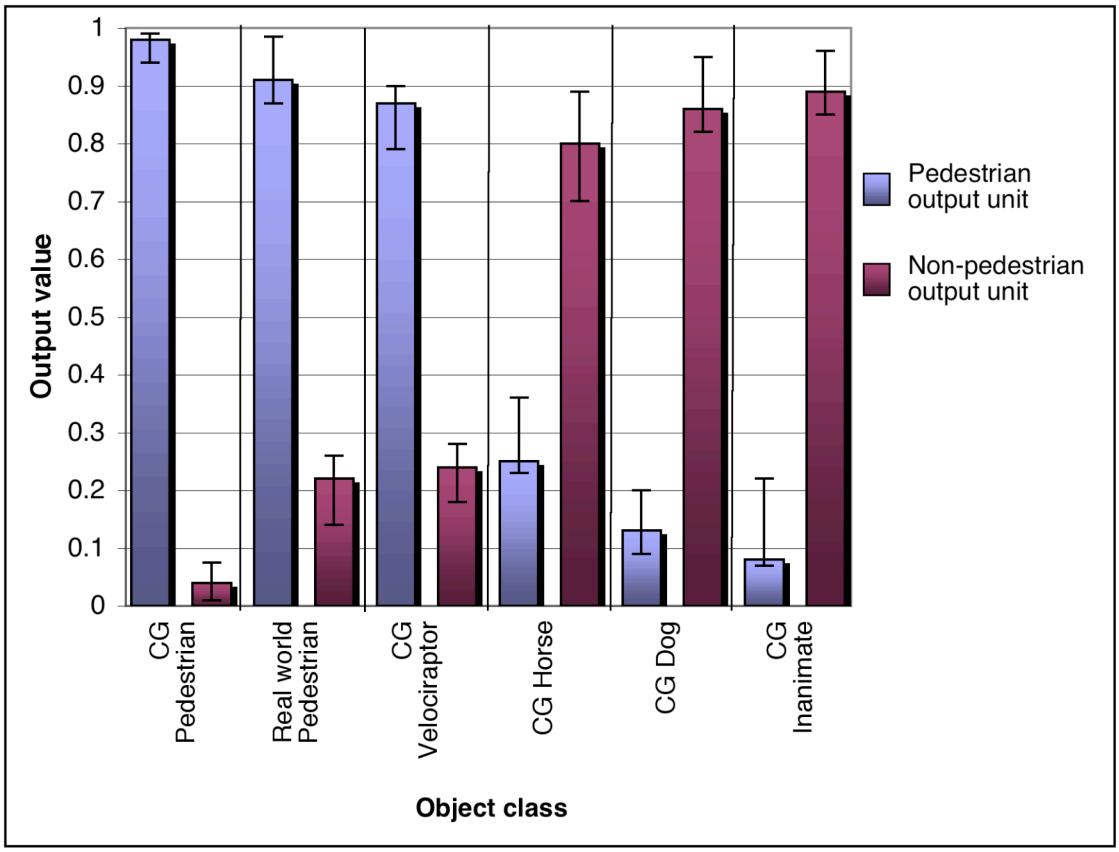

Figure 20: Analysis of neural network categorisation values for different types of unseen objects. The network's confidence value for a given shape can be obtained by subtracting the first output unit's value from the second. Results shown reflect the values given for the test set shown in [Figure 19]. Also shown are the standard deviations for each object class

4.2 Classifying Partially Occluded Objects

It was interesting to test how robustly the network behaved when presented with partially occluded simulated human shapes. The partially occluded shapes were generated by removing percentages of complete simulated human shapes, so that the simulated human could be envisaged as walking behind an invisible screen; all occlusion took place on the leading edge of the simulated human [Figure 21]. The

network was not re-trained with partial shapes, training included only the complete simulated human and complete simulated non-human shapes used in the previous experiments.

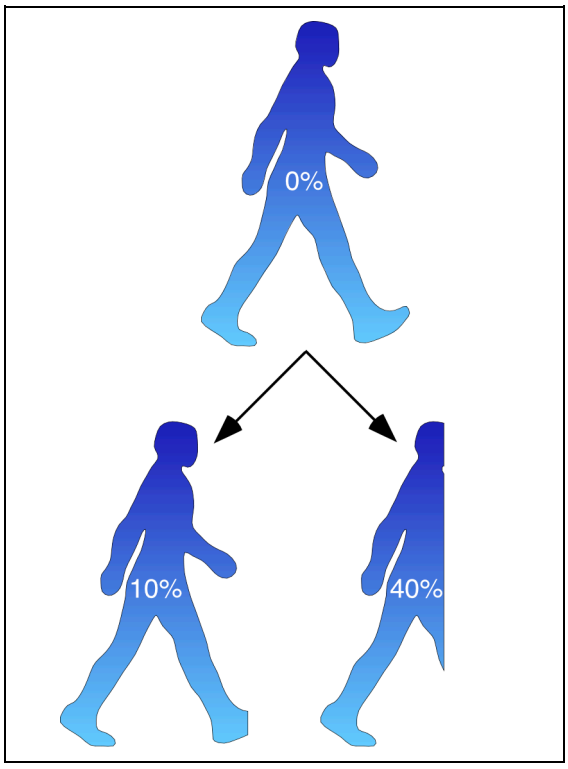

Figure 21: Generating partially occluded simulated human shapes. The original simulated human shape is clipped on its leading edge by a percentage. That partial shape is then abstracted using a snake and an axis crossover vector, as before

The results for 5 individuals occluded from the leading edge are shown in [Figure 22]. From the results it is evident that some poses are much more robust to occlusion than others. Specifically the upright poses are almost completely resilient; the further into the pace, and therefore the more outstretched the legs, the less resilient to occlusion the shapes become. This is perhaps because the upright, 'closed' poses remain similar in shape even when one side of the shape is removed; they are still upright and closed in nature. With the more outstretched poses, however, the removal of one side of the shape can result in the removal of a projected leg [as in Figure 21], changing the shape significantly and testing the neural network's content addressable memory to its limit.

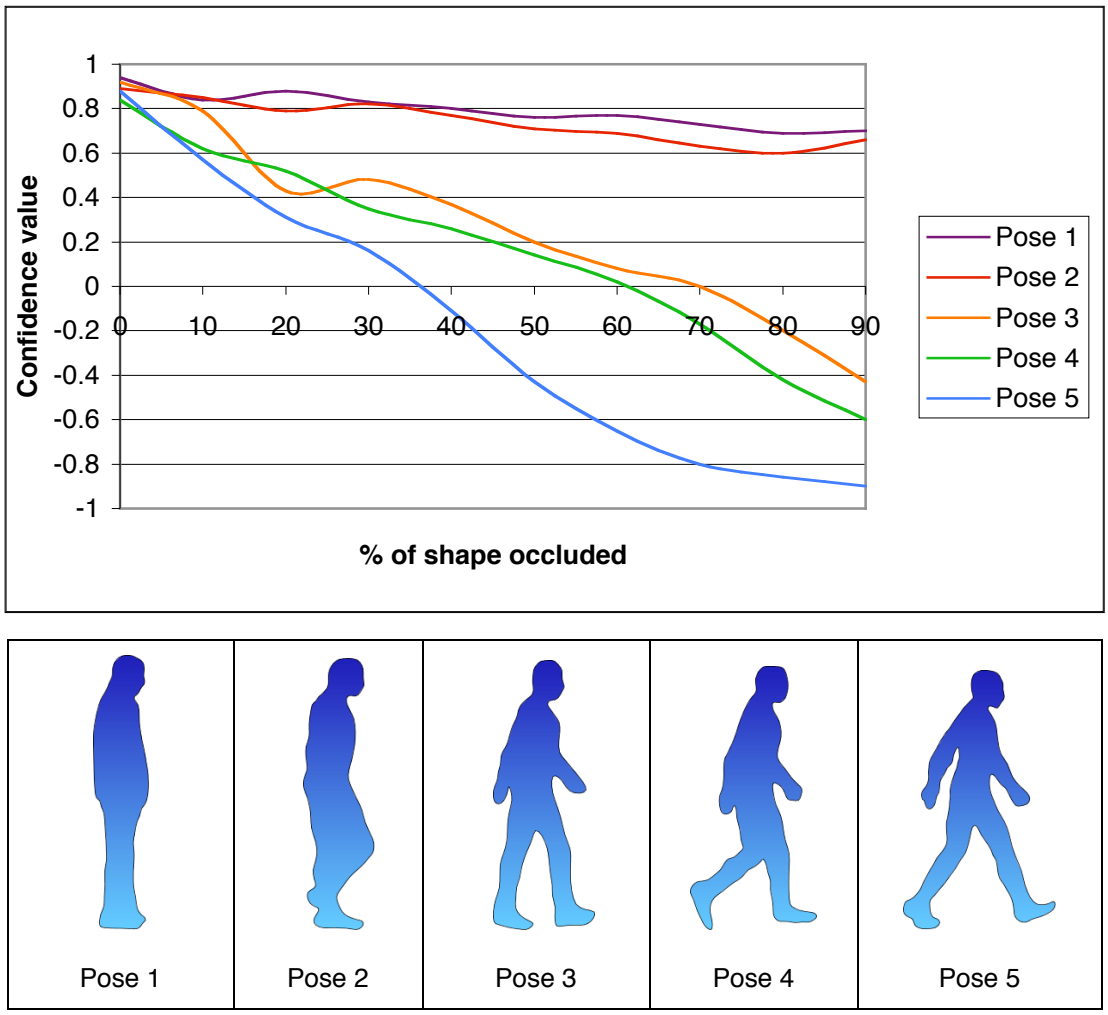

Figure 22: [Top] Neural network classification of 5 simulated human shapes as those shapes become increasingly partially occluded. **[Bottom]** The poses which were subjected to occlusion in the graph. The neural network can be seen to be more resilient to occlusion in some shapes than others; a correlation can be seen between the accuracy of categorisation during occlusion and the distance between the feet of the shape

5 Classifying Simulated and Real Target Objects During Motion

A trained neural network, when presented with a given axis crossover vector, produces a scalar value measuring the confidence of the network that the shape is human. As the object moves, the network's confidence value changes, reflecting the fact that the typicality of the shape of a walking human varies over time. In this section, we investigate the following:

• How an individual's motion pattern is periodic
• How that periodic pattern is different from one object to another within the same class
• How different object classes may exhibit different periodic patterns

The neural network was trained with the same training set described in [Figure 18]; no further training took place to allow for motion analysis. The test set used consisted of the same object classes used in section 4.1, with the exception of inanimate objects; as this experiment involved motion analysis, inanimate objects were not tested. 65 QuickTime movies, each of 120 frames in duration, were used as the test patterns [Figure 23]. For each movie, each frame had a snake introduced to find the object, and a 16-axis crossover vector was then generated from the snake's contour, forming the test pattern. Simulated object movies were created in the same 3D modelling and animation software used to create static images of the simulated objects. The real human movies consisted of recording humans walking in complex outdoor environments, using a DV camera, and exporting that DV movie as a QuickTime movie. Whereas the framerate:pace ratio of the simulated object movies could be finely controlled, allowing for paces to occur every 60 frames, the real humans could not be so easily manipulated. Therefore, the start of each pace in the real human movies was tagged manually, and an even stream of 60 frames was exported automatically from the segment of movie containing the pace, allowing a pace to take 60 frames in both the simulated- and real-object movies, irrespective of the object class.

The first set of results, relating to 4 moving real humans, are shown in [Figure 24]. In each graph 2 consecutive paces are superimposed for the same individual. The output of the network is periodic, repeating once per pace; note that there is some variation between the 2 paces for an individual but that the pattern is very similar, showing that the basic cyclic pattern of each pace is identified. The network's repetitive output could be used by further processes to identify the single paces of an object in motion, so that the object's speed of motion can be gauged. Moreover, individual differences between humans are apparent and these signatures could be used as identification tags.

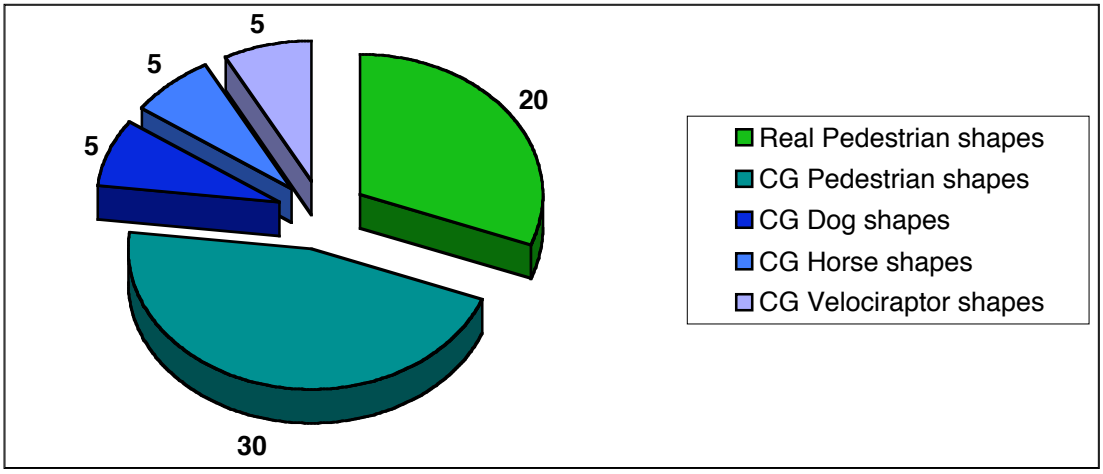

Figure 23: Test set for the motion classification experiment. Each object used in the test set consisted of a 120-frame QuickTime movie, covering 2 consecutive paces of the object's movement. Each object moved from one side of the visual field to the other

Figure 25 shows objects from all classes being analysed over 2 consecutive paces. As with the real humans in [Figure 24], each object displays similar movement from one pace to the next, irrespective of its object class. Furthermore, when these objects' motion patterns are plotted together, it is evident that different object classes display different motion patterns.

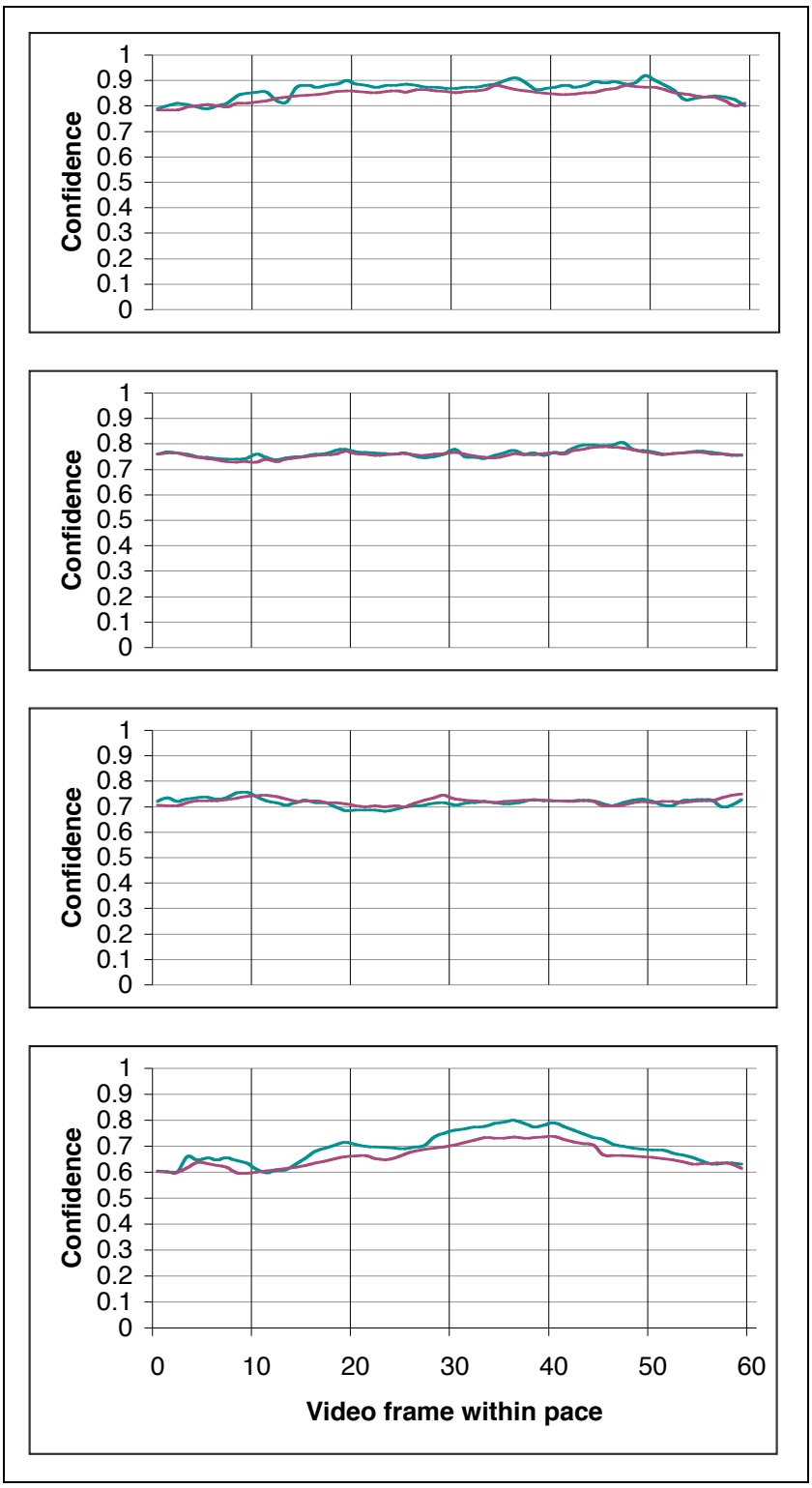

Figure 24: Motion analysis of 4 real humans. In each graph, 2 consecutive paces for the given human have been superimposed, showing each human's repetitive yet distinctive motion pattern

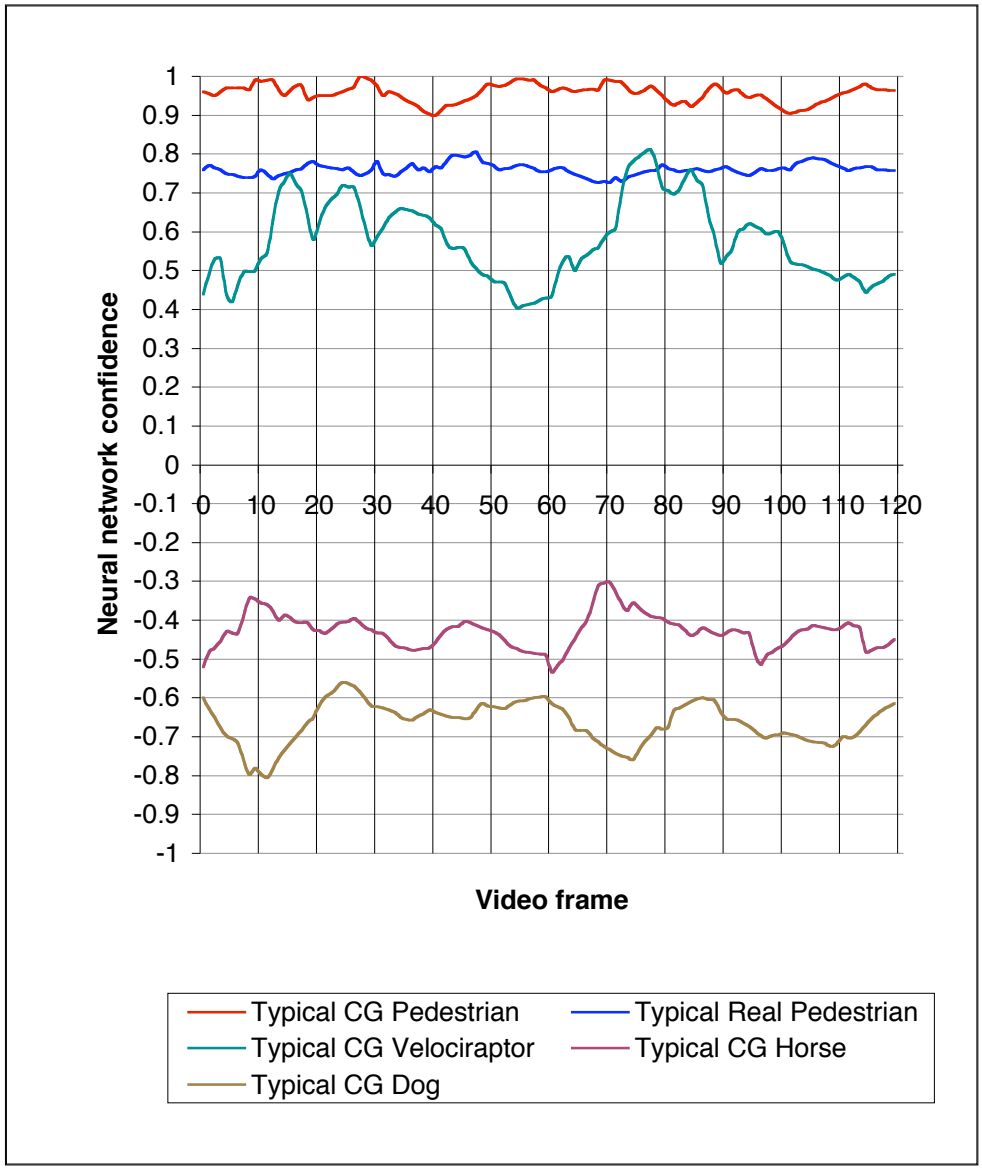

Figure 25: Comparison of neural network confidence values during animate object motion. Each object has been tracked for two paces, where the second pace starts at frame 61. Differences between different object classes' motion signatures are evident, with some overlapping between the previously unseen simulated velociraptor class and the previously unseen real human class

6 Discussion

The main finding of this paper is that the combination of active contour models and neural network classifiers provides a useful tool for the identification of static human and non-human forms, and for analysing the motion of animate objects.

Snakes based upon the Fast Snake model have been used to successfully track moving humans around the visual field. The snakes' contours have then been re-represented in a scale-, location-, resolution- and control point rotation-invariant axis crossover vector. We have found that using 16 axes when generating the vector provides an effective level of detail for encoding human shape information. These vectors have been used to train a neural network to differentiate human shapes from non-human shapes. The resultant neural network achieves a high level of success when presented with both unseen simulated shapes and unseen real shapes, having been trained using simulated shapes only. Furthermore the network's content addressable memory allows it to continue to successfully categorise shapes when those objects are becoming partially occluded.

The network's output values can be analysed over time to provide a motion signature for a given object. By comparing different objects' motion patterns in this way, we have shown how the network can be used to identify the periodic nature of an individual's motion, the subtle individual differences of objects within a class, and the greater class-specific differences in motion.

The system presented does not attempt to address all of the tasks required for an autonomous surveillance system; in particular, the snakes are initialised manually around regions of interest. Alternative methods of contour initialisation exist, typically involving the use of optic flow [3, 2, 1] to determine regions of interest in the image, around which contours are then automatically initialised [1]. Whilst the emphasis of our research is not on the automatic generation of initial contour positions, the abovementioned initialisation method would be compatible with our system .

Further work in this research involves identifying if the neural network's accuracy at classifying real human objects could be improved by training it using real human shapes instead of, or as well as, simulated human shapes. Clearly this would make the task of generating the training set more complex, as there would be less control over the variety and movement of the real human subjects than currently exists in the 3D modelling and animation software, making it more cumbersome to obtain a representative data set from which the neural network can generalise. In a similar vein, the system presented can only at present deal with objects which move across the field of view. It would be more useful if it could classify objects moving more generally in any direction, or changing directions, within the field of view.

It would be interesting to investigate the use of the motion signatures provided by the neural network as identification tags for individuals. Potential uses for this application would be surveillance systems and any system where unknown individuals should be treated with caution. In addition to identifying strangers, the speed of an individual's motion could be gauged by analysing the repetitions in the neural network's output; when the neural network's output repeats, the user has taken another pace.

One weakness of the method presented in this paper is that, even if an object being tracked by the snake is classified 'human', there is still no information available regarding that human's behaviour, for instance its past and current postures. Both active contour models and the axis crossover vector only form a silhouette representation of an object, with no attention paid to the microstructure of the object. For instance, neither model can incorporate an object's self-occlusion in the representation; if one arm obscures part of the torso, it does not affect the outline of the object, and so is not identifiable from either the snake's contour or the axis crossover vector. It would be useful, once the neural network has classified the object being tracked as human, to be able to determine a human's posture. This could potentially be achieved by using a landmark based analysis, for example the Point Distribution Model [4] in concert with the snakes. This would allow particular points on the body to be identified, thus enabling those landmarks' relative positions to be identified. Alternatively a History Space Classifier [7] could be constructed to identify abnormalities in an individual's motion pattern. Applications of such a technology might be the medical domain or semi-automatic CCTV systems, providing an indication to a human observer that a certain event has occurred.

References

[1] A. Blake & M. Isard (1998). Active Contours. Springer-Verlag

[2] M. Sonka, V. Hlavac & R. Boyle (1994). Image Processing, Analysis and Machine Vision. Chapman & Hall

[3] A. Watt & F. Policarpo (1999). The Computer Image. Addison-Wesley

[4] T.F. Cootes, C.J. Taylor, D.H. Cooper & J. Graham (1992). *Training models of shape from sets of examples.* In Proceedings of the British Machine Vision Conference 1992, pp. 9-18

[5] J.B. Hayfron-Acquah, M.S. Nixon & J.N. Carter (2001). Automatic Gait Recognition via the Generalised Symmetry Operator. In Proceedings of the BMVA Workshop on Understanding Visual Behaviour, Jan 24th 2001

[6] M. Kass, A. Witkin & D. Terzopoulos (1988). *Snakes: active contour models.* In International Journal of Computer Vision (1988), pp. 321-331

[7] D.R. Magee & R.D. Boyle (2000). *Spatio-temporal modeling in the farmyard domain.* In Proceedings of the First International Workshop, Articulated Motion and Deformable Objects, Mallorca, September 2000, pp. 83-95

[8] N. Shimida, Y. Shirai & Y. Kuno [2000]. *Model adaptation and posture estimation of moving articulated object using monocular camera.* In Proceedings of the First International Workshop, Articulated Motion and Deformable Objects, Mallorca, September 2000, pp. 158-172

[9] K. Tabb, N. Davey, S. George & R. Adams (1999). *Detecting partial occlusion of humans using snakes and neural networks.* In Proceedings of the 5th International Conference on Engineering Applications of Neural Networks, 13-15 September 1999, Warsaw, pp.34-39

[10] D.J. Williams & M. Shah (1992). *A fast algorithm for active contours and curvature estimation.* In CVGIP - Image Understanding 55, pp. 14-26

[11] K. Tabb & S. George (1998). *Snakes and their influence on visual processing.* Internal Technical Report, Department of Computer Science, University of Hertfordshire