A Hybrid Detection and Classification System for Human Motion Analysis

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, UK

Summary:
This paper discusses a hybrid technique for detecting and tracking moving pedestrians in a video sequence. The technique comprises two sub-systems: an active contour model for detecting and tracking moving objects in the visual field, and an MLP neural network for classifying the moving objects being tracked as human or non-human . The axis crossover vector method is used for translating the active contour into a scale-. location-, resolution- and rotation-invariant vector suited for input to a neural network, and we identify the most appropriate level of detail for encoding human shape information. Experiments measuring the neural network s accuracy at classifying unseen computer generated and real moving objects are presented, along with potential applications of the technology. Previous work has accommodated lateral pedestrian movement across the visual field; this paper describes a system which accommodates arbitrary angles of pedestrian movement on the ground plane.

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

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.

Alternative methods exist for determining the class of an object being tracked [1, 2], although these techniques generally rely either upon the object being centred in

the image throughout the process, or upon complex models of the target object being formed on the fly, resulting in either an inability to support multiple objects in the same image, or 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.

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 neural network architectures. 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 computer generated and real human shapes, being tracked by the active contour model, to a high degree of accuracy. 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 Detecting and tracking moving objects

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 [3].

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 [4]. 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.

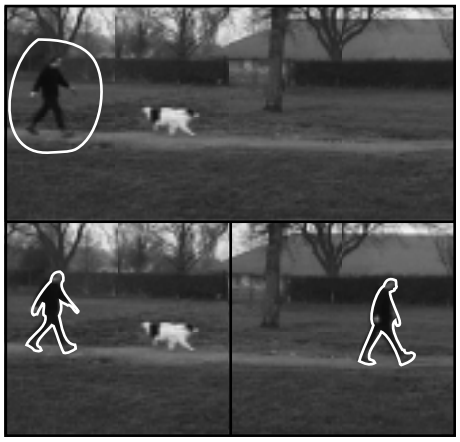

Fig. 1. A human being tracked with a snake. [Top] The snake is initialised manually around the target object. [Bottom left] The snake minimises its energy within a given frame, snapping itself onto the target object. It then moves into the next frame using this relaxed position as its starting state in that frame. [Bottom right] In this way the snake can track a moving object by successively minimising its energy through a sequence of frames

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 [Fig. 1].

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. 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. The energy function is defined by the user and contains a set of mathematical rules which govern the movement of control points. Changing the definition of the energy function, or parameters within the energy function, will result in the contour becoming attracted to different features in 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.

In our work, we have adopted the Fast Snake model [5], 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 [6].

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 [3]. 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:

$$CP\_Energy_i = \alpha \cdot Continuity\_Energy_i + \beta \cdot Curvature\_Energy_i + \gamma \cdot Image\_Energy_i \quad (1)$$

where i is the index of a specific control point in the snake, *CP_Energy* is the total energy for control point i, *Continuity_Energy* is the control point s continuity energy, *Curvature_Energy* is the control point s curvature energy, *Image_Energy* is the control point s image energy, and α, β and γ are the weighting parameters for the continuity, curvature and image energies respectively. By changing these weighting parameters, the precedence of each energy can be increased or decreased, allowing for different behaviours even within the same energy function.

Continuity energy is independent of any features in the image which have been locked onto by the control points, 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 instead control points were to bunch up, then part of the shape would be identified at a high resolution, but other parts would be sparsely represented. 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.

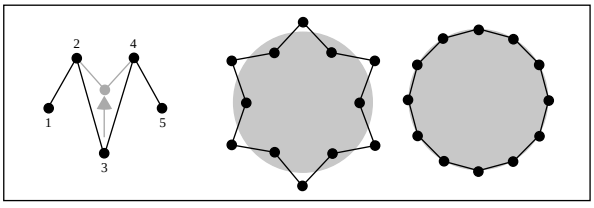

Fig. 2. Curvature energy aims to encourage or discourage corners from forming in the snake, depending upon the value of β in (1). [Left] Curvature energy moves a control point to the central position of neighbouring points (if β is positive). [Middle & Right] When curvature energy is applied to every point on the contour it has the effect of removing jagged corners from the contour

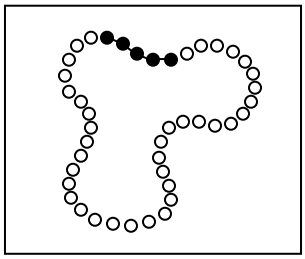

Fig. 3. With sufficient control points, curvature energy need not prevent a snake from forming a complex shape; provided curvature between adjacent control points is low, a snake can still form a complex shape

Curvature energy, like continuity energy, is independent of features in the image. It aims to govern the frequency and sharpness of any corners formed in the contour [Fig. 2]. 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 [6]. 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 [Fig. 3].

Image energy in our implementation is based on a number of image preprocessing algorithms [Fig. 4]. Firstly, areas of movement are detected in the image. Motion detection involves differencing successive frames, and so detects not only moving objects but also the holes they leave behind, that is, their locations in the previous frames. After differencing the resulting motion detection to a continually updated template of the background, only the moving objects remain with a slight padding around their edges. Blobs , that is, regions that are too small to be considered image features, are then masked out, removing most of the noise caused by leaves blowing or water rippling. A Sobel mask [7] is passed over the resultant image, to enhance edges. Finally, all edges still in the image are normalised, to prevent any edges from appearing disproportionately strong due to the narrow 3x3 pixel focus of the Sobel mask [5]. The image energy value for a given pixel is then simply the pixel s colour value in this resultant image [Fig. 4 - right].

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 pixel locations to determine whether any of these locations would offer a lower energy value for the control point than its current location, using the energy function. Having searched its neighbourhood, the control point is then moved to the location in the neighbourhood offering the lowest energy. 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 on 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 locations in their neighbourhood, whereby the snake remains relatively still between iterations. Once a snake ia relaxed, its final position is used as a starting position in the next video frame, thus it is assumed that the movement between frames is sufficiently small that the relaxed snake czn track the object.

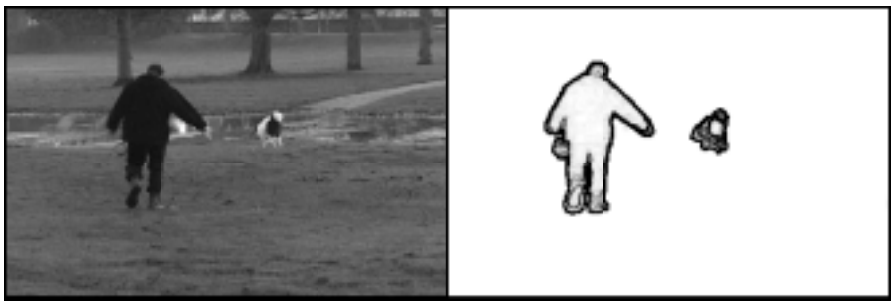

Fig. 4. [Left] The original frame from a movie. [Right] The image enhancement process prior to the introduction of active contour models to the image

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 [Fig. 1].

Fig. 5 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 [3], but the model can be seen to be successfully locating the target human s outlines.

As can be seen from Fig. 5, 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 using active contour models alone.

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.

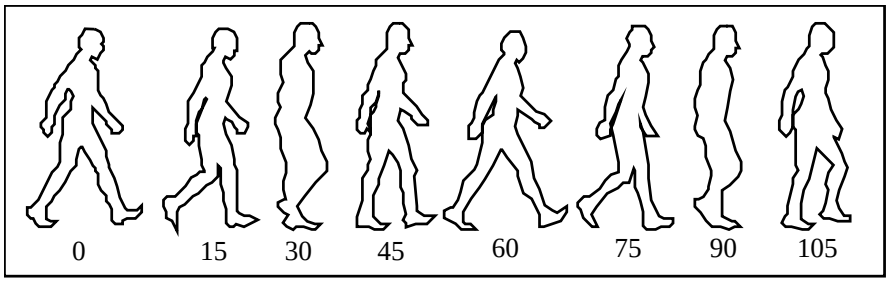

Fig. 5. A target human detected and tracked through a sequence of 105 frames using an active contour. The frame numbers are shown below each frame; intermediate frames have been omitted

3 Active contour vector translation

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 detail 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 section.

The reliance of active contour models upon features in an image brings both advantages and disadvantages to the task of object identification. By locking onto features in an image, an active contour is able to abstract the actual object in the image, rather than an approximation or a prediction. This provides the highest resolution shape possible from the image data alone. Conversely, the control points on the contour have absolute coordinates based in the image coordinate space, as the control points relate to points in the image. This dependence upon image coordinates means that 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 contours are not scale-invariant either. 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 and are thus rotation-variant. All of these factors make the comparison of contours difficult.

In order to use active contours shapes as input data for a neural network performing a shape classification task, these qualities need to be removed. The re-representation therefore must provide for scale-, location-, resolution- and rotation-invariance if it is to be useful as a generic shape description. 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 [8]. 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 [Fig. 6]. The distance from the contour s centre to its furthest edge along that axis is then stored in a vector. The vector length is equal to the number of axes being projected. This allows the vector to be location-invariant, as no image coordinates 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. Similarly, it does not

matter where on the contour the first control point resides, and so is rotation-invariant.

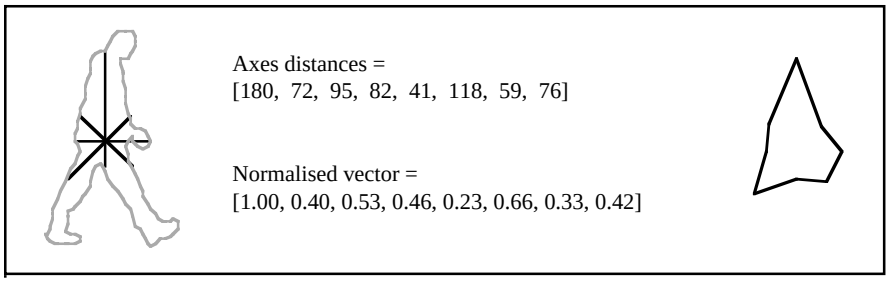

Fig. 6. An 8-axis crossover vector. [Left] A predetermined number of axes are projected from the centre of a contour to its furthest edges at specified angles. The distances of these axes are then stored in a vector and normalised. [Right] A polygonal visualisation of the actual shape information encoded by the axis crossover vector

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.

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 Fig. 6.

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 [Fig. 7].

There are two limitations to using axis crossover representations as input to neural networks. Firstly, a given axis must be projected in the same direction between shapes. For instance the first axis in all of our shapes is projected at 0¡ (vertically). If axes are projected differently between shapes then the task of comparing axis crossover vectors is complicated.

Secondly, the granularity of shape description can be tailored to a specific task by using a larger or smaller number of axes. However, because vector elements map onto input neurons, the number of axes used must be equal in all vectors used to train or test a given neural network.

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.

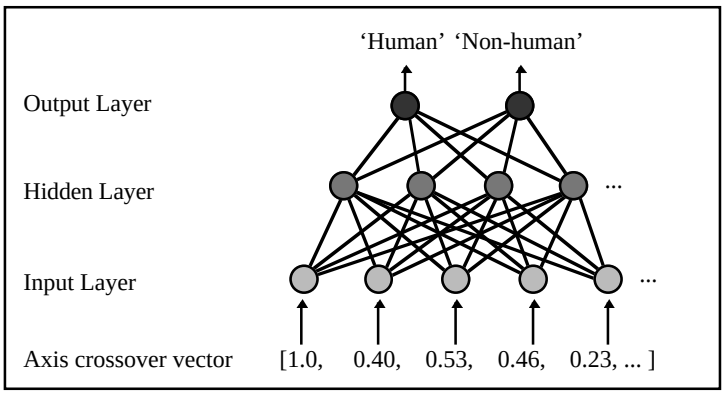

Fig. 7. Axis crossover vectors as input patterns for neural networks. A given vector element maps onto a given input unit in the neural network, thus the size of input layer and axis crossover vector must equate

3.2 Verification of axis crossover vectors as generic shape descriptors for neural networks

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).

In order to classify shapes as human or non-human using a supervised neural network, it was 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 computer generated using 3D modelling and animation software, which allowed for tighter 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.

The same set of computer generated (CG) shapes were used in each neural network s training set, encoded with axis crossover vectors containing the appropriate number of axes, for example 4 axes for the neural network containing 4 input units. The training set contained 150 pedestrian shapes, and 150 non-pedestrian shapes. The non-pedestrian shapes consisted of inanimate outdoor objects, such as cars, trees and lamp-posts.

The test set contained the same set of 10 CG pedestrian and 10 CG non-pedestrian shapes across all neural networks, again encoded using appropriately sized axis crossover vectors for each neural network. All shapes in the test set were unseen, but were of object classes included in the training set.

The networks were trained to within 0.05 error, and when tested, 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 .

Each shape used as a training or test pattern had a snake locked onto it, which was then translated into an axis crossover vector. Axis crossover vectors containing 4, 8, 12, 16, 20 and 24 axes were used, each on 10 identical neural networks containing the corresponding number of input units. Each of the 10 identical neural networks were initialised with a different random weight matrix, to lessen the chances of a network becoming trapped in local minima in the weight space.

All networks contained 2 output units. All training patterns required an output of 1, 0 (human pattern) or 0, 1 (non-human pattern). Using 2 output units allowed the network s confidence value to be identified. The confidence value is simply the difference between the two output units values. This provides a measure of how confident the network considers its classification of a given shape to be; a value of +1.0 denotes maximum confidence that the object is human, a value of -1.0 denotes maximum confidence that the object is non-human, and a value of 0 denotes minimum confidence (maximum uncertainty). Values for shapes which the neural network is not confident in classifying can be analysed to determine why the network is not confident; a value of 0 could mean that the network output 1, 1 , 0, 0 or 0.5, 0.5 , for instance. Each means something different; the network might consider the shape to be both human and non-human, neither human nor non-human, or partly human and partly non-human. If we had used only a single bipolar output unit a value of 0 would only tell us that the network was uncertain, and not why it was uncertain.

Shapes which the network is uncertain at classifying can be incorporated into the training set, so that the training set becomes more representative of the possible shapes which will be encountered, and improves the network s generalisation abilities with these types of shape.

Experiments previously reported in [8] show that the networks which used 16-axis crossover representations, and therefore 16 input units, displayed the best performance. Of these 16 input unit networks, those containing 13 hidden units

were best at distinguishing human from non-human shapes, being able to classify 90% of unseen human shapes and 60% of unseen non-human shapes correctly. Interestingly, using more than 16 axes gave no benefit in performance whilst requiring more epochs during training. The neural network with 16 input units was therefore used henceforth, together with 16-axis crossover vectors.

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 previous experiment to see how confident it was that a pedestrian vector was pedestrian and that a non-pedestrian vector was not.

Fig. 8 shows the average results for the 10 unseen CG human and 10 CG outdoor non-human vectors. The network s output units are split into a human unit and a non-human unit (labelled OU1 and OU2 respectively in Fig. 8) which should ideally fire mutually exclusively of one another, as something cannot be both pedestrian and non-pedestrian. However, as was found in the previous experiment, 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.

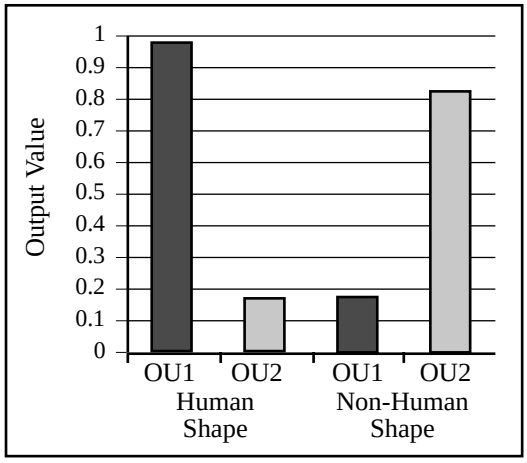

Fig. 8. Mean test set classification results, from experiments on ten identical double output unit neural networks trained using 16-axis CG human and CG outdoor non-human shapes. Each network was initialised with a different initial weight matrix

The average confidence value when classifying an unseen CG 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 an unseen CG 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 Fig. 8 reflect the trend exhibited in the previous experiment, that 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.

4 Shape analysis and classification

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 CG and real object classes. This section uses exclusively a network with 16 input units, 13 hidden units and 2 binary output units, where one output unit was trained to identify human shapes, and the other non-human shapes, as in the previous section.

4.1 CG and real object classification - lateral object movement

The neural network architecture developed in the previous section was re-trained, this time with 400 examples each of CG human and CG non-human shapes in the training set. The non-human shapes consisted of 200 inanimate outdoor objects such as trees, cars and streetlights; and deformable animate objects, namely 100 shapes each of CG dogs and horses, again generated using 3D inverse kinematics software. 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. All animate objects used in the training set (the CG humans, horses and dogs) were moving laterally across the image, either from left to right or right to left. The network was trained to within an error margin of 0.05.

The test set for this categorisation experiment contained 100 unseen CG human shapes, 100 unseen real human shapes, and 100 unseen CG non-human shapes. Within the non-human test patterns, a previously unseen object class was introduced to the network: CG 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. The CG non-human objects consisted of 25 shapes each of CG inanimate objects, CG horses, CG dogs, and CG velociraptors. As with the training patterns, all animate objects were moving laterally across the image from either left to right or right to left. The real human shapes were obtained from complex outdoor scenes, as illustrated in Fig. 1, and included male and female humans of varying heights, weights and ages.

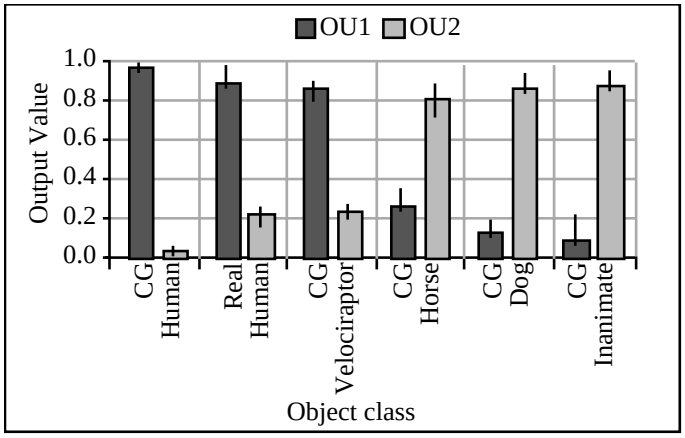

Fig. 9. Mean neural network categorisation values and standard deviations for different types of unseen objects. The network was trained and tested using shapes of objects moving laterally across the image

The results in Fig. 9 show that the network clearly identifies unseen CG humans as human with a high level of confidence, and unseen CG dogs, horses, and inanimate objects as non-human with a high, albeit lesser, level of confidence. The CG velociraptors are classified as more human than non-human, but the network still clearly distinguishes between the CG humans and the CG velociraptors, with mean confidence values of 0.94 and 0.63, respectively. Real humans were not classified as accurately as CG humans, but were nevertheless again categorised as human with more confidence than the CG velociraptors, obtaining a mean confidence value of 0.69. It is highly significant that the network, trained only on CG human / non-human shapes, can then correctly classify real humans. In addition, it is impressive that the network can distinguish the real human shapes from a previously unseen object class.

4.2 CG and real object classification - omni-directional object movement

Despite the results from the previous experiment showing some promise that the neural network was able to distinguish both real and CG humans from CG non-human objects, the only objects used so far, in both training and testing, had all been walking laterally to the camera. It was therefore important to test the same neural network, with no additional training, on objects moving omni-directionally in the ground plane; that is, not just from side to side but in arbitrary directions along the ground plane. Consequently, the test set for this experiment included 320 objects for each object class: CG humans, real humans, CG velociraptors, CG horses, CG dogs, and CG inanimate objects. Each class of object was tested from

16 different viewpoints around the compass: 0¡, 22.5¡, 45¡, 67.5¡, 90¡... 337.5¡. There were thus 20 CG humans moving at 0¡, 20 CG humans moving at 22.5¡, and so on.

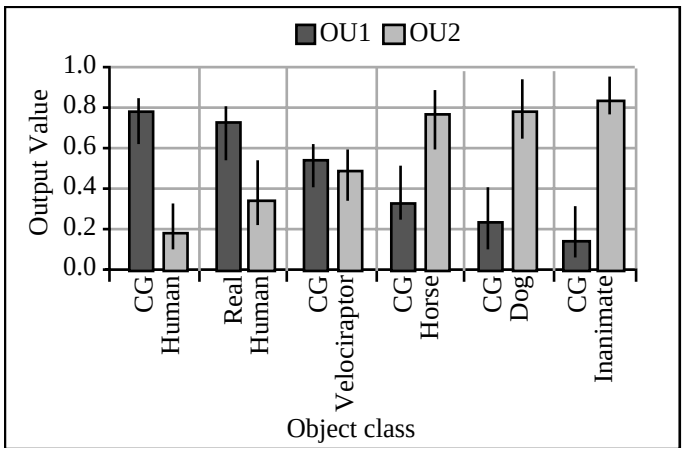

Fig. 10. Mean neural network categorisation values and standard deviations for different types of unseen objects. The network was trained using shapes of objects moving laterally across the image. The objects used in the test set were moving in all directions (0¡ - 360¡) on the ground plane

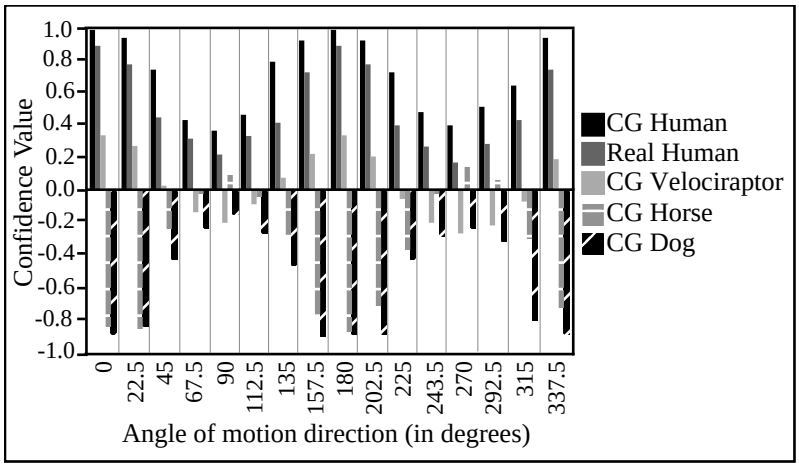

Fig. 11. Mean neural network classification results when tested with objects moving in different directions along the ground plane. Each plot shows the mean classification of 20 examples of an object class moving in a given direction along the ground plane. The neural network was trained with only laterally moving objects

The results in Fig. 10 show that the network is no longer as able to distinguish object types as well when test patterns involve objects moving in different directions. This is unsurprising, as the training set includes no allowance for objects which aren t moving laterally to the camera. In addition to the network s inaccuracy at correctly classifying objects which aren t moving laterally, the standard deviation for the network s classification of each object class, shown on the graph, has also increased [Fig. 10]. A breakdown of how the network performed with each object class at each angle of motion can be seen in Fig. 11.

Of note is that the network is still able to classify the object classes as successfully as before when those objects are moving laterally, that is, when their direction approaches 0¡ or 180¡. However as their movement approaches 90¡ or 270¡, the network s ability to correctly classify them is severely compromised. Nevertheless for the system to be applicable to complex outdoor scenarios, it was necessary to accommodate all movement along the ground plane. Thus it was decided to use a different training set, which incorporated objects moving in different directions along the ground plane.

4.3 CG and real object classification - omni-directional object movement following re-training with omni-directional objects

The network used in the previous section was re-trained to accommodate the classes of object that were in the previous training set, but this time moving in arbitrary directions along the ground plane. Consequently the size of the training set increased; the training set included 600 CG humans, 300 CG horses, 300 CG dogs and 200 CG inanimate objects, totalling 1,400 objects, moving in different directions on the ground plane. Again, no velociraptors were used in the training process. The network shared the same architecture as per previous experiments, and was again trained to within 0.05 error.

The results of classifying unseen omni-directional objects, following training using the new training set, are shown in Fig. 12. The same test set was used as for the previous experiment, and involved unseen omni-directional objects. Of note is that both the network s accuracy and standard deviation at classifying all object classes has improved, including for the unseen real human and CG velociraptor classes. A more detailed breakdown can be seen in Fig. 13, where objects moving in arbitrary directions obtained a consistent classification, irrespective of their direction of movement.

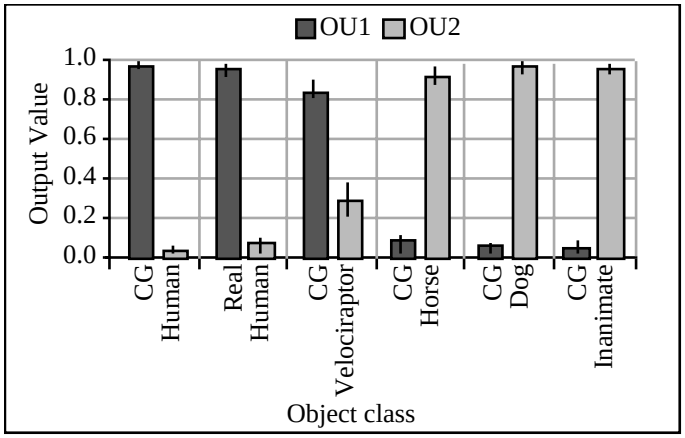

Fig. 12. Mean neural network categorisation values and standard deviations for different types of unseen objects. The network was trained and tested using shapes of objects moving in arbitrary directions (0¡ - 360¡) on the ground plane

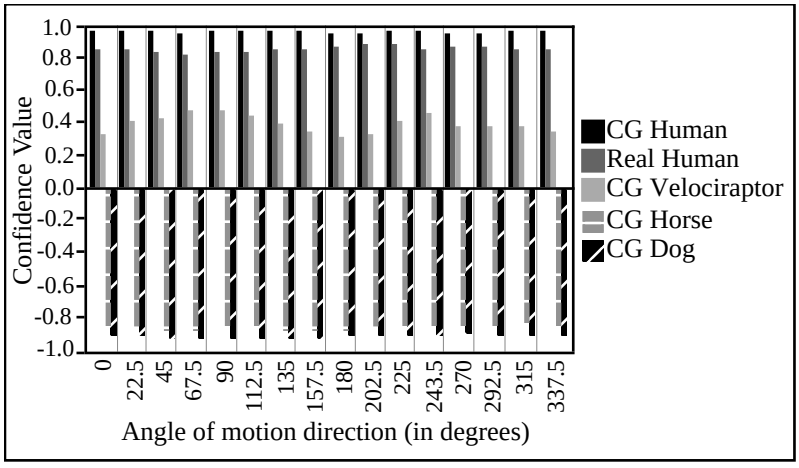

Fig. 13. Mean neural network classification results when tested with objects moving in different directions along the ground plane. Each plot shows the mean classification of 20 examples of an object class moving in a given direction along the ground plane. The neural network was trained with objects moving in arbitrary directions along the ground plane. The network was tested with the same set of shapes as the network in Fig. 11

5 Discussion

It has been clearly demonstrated that the combination of active contour models and neural network classifiers provides a useful tool for the identification of human and non-human forms.

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 CG shapes and unseen real shapes, having been trained using CG shapes only.

When re-training the network with a more representative training set, the network was able to correctly classify objects which were moving in arbitrary directions along the ground plane.

Further work in this research area involves identifying if the neural network s accuracy in classifying real human objects could be improved by training it using real human shapes instead of, or as well as, CG 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 training set from which the neural network can generalise.

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 [9] 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 [10] could be constructed to identify abnormalities in an individual s motion pattern.

Potential uses for this application would be the medical domain or semi-automatic CCTV systems, providing an indication to a human observer that a certain event has occurred.

References

[1] Hayfron-Acquah JB, Nixon MS & Carter JN (2001). Automatic Gait Recognition via the Generalised Symmetry Operator. In Proceedings of the BMVA Workshop on Understanding Visual Behaviour, Jan 24th 2001.

[2] Shimida N, Shirai Y & Kuno Y [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.

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

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

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

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

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

[8] Tabb K, Davey N, George S & Adams R (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.

[9] Cootes TF, Taylor CJ, Cooper DH & Graham J (1992). Training models of shape from sets of examples. In Proceedings of the British Machine Vision Conference 1992, pp. 9-18.

[10] Magee DR & Boyle RD (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.

[11] Tabb K, Davey N, Adams R & George S (2001). The recognition and analysis of animate objects using neural networks and active contour models. In Neurocomputing, Special edition on Engineering Applications of Neural Networks [in press].