VVG (2005)
(Editors)

Learnt Inverse Kinematics for Animation Synthesis

Anonymous

Abstract

Existing work on animation synthesis can be roughly split into two approaches, those that combine segments of motion capture data, and those that perform inverse kinematics. In this paper, we present a method for performing animation synthesis of an articulated object (e.g. human body and a dog) from a minimal set of body joint positions, following the approach of inverse kinematics.

We tackle this problem from a learning perspective. Firstly, we address the need for knowledge on the physical constraints of the articulated body, so as to avoid the generation of a physically impossible poses. A common solution is to heuristically specify the kinematic constraints for the skeleton model. In this paper however, the physical constraints of the articulated body are represented using a hierarchical cluster model learnt from a motion capture database. Additionally, we shall show that the learnt model automatically captures the correlation between different joints through the simultaneous modelling their angles.

We then show how this model can be utilised to perform inverse kinematics in a simple and efficient manner. Crucially, we describe how IK is carried out from a minimal set of end-effector positions. Following this, we show how this "learnt inverse kinematics" framework can be used to perform animation syntheses of different types of articulated structures. To this end, the results presented include the retargeting of a flat surface walking animation to various uneven terrains to demonstrate the synthesis of a full human body motion from the positions of only the hands, feet and torso. Additionally, we show how the same method can be applied to the animation synthesis of a dog using only its feet and torso positions.

1. Introduction

The subject of realistic motion synthesis, usually for human bodies, has been a popular and longstanding research topic, due to its numerous practical applications in areas such as video games, movies and biomechanics. To have a realistic appearance, a body structure needs to assume a physically correct and natural pose. Two issues arise out of this: the requirement for some form of knowledge on the physical constraints of the body joints (e.g. a human body) and a way of using such a knowledge base to achieve a realistic body animation synthesis.

One solution for tackling the first issue is to heuristically specify the physical limitations of different body joints. These can take the form of hard constraints on the range of allowable angles on a particular joint. Following this, one can then incorporate such constraints directly into the kinematic model. However, the constraints of a joint may be different depending on the configuration of its parent and child joints.

In recent years, there has been an increase in the availability of motion-captured data, due to its growing usage and support in both hardware and software. Additionally, such motion capture databases usually span many different types of movement example. Therefore, it would be advantageous if one could exploit the availability of such information to obtain the needed constraint information.

With this in mind, we have developed a system that addresses the above issues. At its heart lies a learning-based statistical model for capturing the physical or kinematic constraints of an articulated structure. The use of a learning-based approach allows us to automatically model the physical movement limitations of the body structure of interest. The learnt model takes the form of a dual-layer hierarchical cluster model that captures the correspondences between body end-effector positions (e.g. hands and feet positions for a human body) and the respective joint angles of the skeleton hierarchy. The use of a learning framework also provides us capability of automatically modelling the kinematics constraints of totally different skeletal structures using the same method. As we will show later in the result section, we have used the same method for learning the joint angle constraints for both the body of dogs and humans.

One can also think of the body joint positions as a form of a *positional constraint information* for the possible joint angles. The learnt cluster model then allows us to efficiently index from a minimal set of endpoint or constraint positions to the required joint angles. We also present two types of clustering method used to learn the parameters of the clusters from available motion-captured data.

We then propose a novel algorithm that uses the cluster model for performing a form of "learning-based inverse kinematics". Additionally, we show how such an algorithm can be applied to the synthesis of motions of a particular body-type from only a small set of given constraint positions (e.g. feet and hand positions).

The rest of the paper is presented as follows: After a discussion on related work in the next section, we will describe the proposed learnt constraint model and its training methods in Section 3. The application of the constraint models for performing inverse kinematics and animation synthesis is then provided in Section 4. In Section 5, experimental results are shown before concluding in Section 6.

1.1. Related Work

In this section, we will provide a review of related research that focus on the synthesis of human body animations.

Following the increasing availability of motion capture data, a lot of research has been carried out into methods for generating animations through the combination of existing motion capture segments. Early work by Lamouret and Panne [11] created new animations by cutting-and-pasting together motion segments of an existing animation. There, motion segments are selected based on how well they fit into a desired motion and subsequently tailored for a more precise fit. This was applied to the Pixar Luxo lamp, where novel animations of the lamp jumping along uneven terrain were generated.

Recently, there has been research carried out that combines both motion-capture datasets with learnt models. This approach was originally proposed by Molina and Hilton[12], where a system that performed interpolated between specified start and end key frames was described. The preprocessing of data is performed using Principal Component Analysis (PCA) and clustering. Clusters were used to partition the motion dataset into groups of similar motions. The user-specified key-frames are located within a particular start and end clusters. A connecting path through the start and end clusters is then found through dynamic programming, and the most probable sequence of motion segments passing through these clusters are used to generate the required motion.

Along similar lines, Lee et al. [5] presented work where they generated animations of a human figure undergoing different actions, such as climbing and walking on boxes. Similar motions segments from different sequences are initially clustered together. Following this, possible transitions between motion frames to other motion frames are captured using a data structure called a cluster tree, which subsequently is used for generating new animations. Work by Kovar et al.[10] and Arikan et al.[1] similarly generated motion by initially generating a motion graph from a database of motion-capture data. The motion graph encodes how clips of captured data can be re-assembled in different ways. Therefore, new animations can be generated by selecting sequences of nodes on the motion graph. Rose et al.[14] used existing motion data and interpolates them using radial basis functions to create new motions, with the objective of solving a given inverse-kinematics constraint. However, all of these approaches require the original motion-capture database to be kept.

As an alternative, learnt models can be built and used for generating new human animation sequences, thereby completely replacing the dataset. Work by Ong and Gong[13] and Bowden[3] built statistical models through a combination of initial PCA and subsequent clustering on motion capture data. Following this, the temporal characteristics of the motions were modelled as Markov chains. A more sophisticated model for the dynamics of the body motions can be captured using hidden Markov models, as given in work by Galata et al.[6] and Karlouva et al.[9]. Additionally, Brand and Hertzmann[4] extended the approach by proposing a learning algorithm that identifies different styles in a dataset of animations.

Our approach follows the learnt model approach by proposing a kinematic constraint model learnt from motion capture data. Such a constraint model can then be utilised to synthesise animations given only a minimal set of constraint positions. Following a quick explanation on the chosen notation in the next section, we will describe the learnt kinematic constraint model in Section 3.

2. Notation

- functions are denoted by bold capital letters, $\mathbf{Q}$.
- constant values are denoted by capital normal letters, D.
- scalars variables are denoted by normal lowercases, x, y, z.
- vectors are denoted with a bar above, $\bar{p}, \bar{q}$.
- compound objects (e.g. clusters) are denoted by bold lowercases, $\mathbf{c}$.

3. Learnt Constraint Model

3.1. Representation

We now introduce the definitions for the representation of the human body. This consists of two types of information, the positional constraints and joint angles (see Figure 1). Positional constraints are a set of 3-D positions of certain body parts. They are usually the end-effector positions (e.g. hand or feet positions) produced by applying a set of joint angles

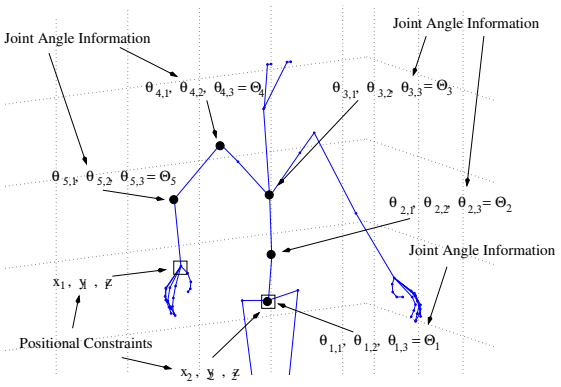

Figure 1: *An illustration of the positional constraints and its associated joint angles for the example of modelling the torso and right arm. The vertices that were selected to act as positional constraints are highlighted with boxes, while filled circles show the vertices that will provide the required joint angles.*

in a kinematic skeleton model, although sometimes they are can also include root positions such as the pelvis position. We also note that the postional constraints are usually made relative to some reference point (e.g. the feet and hands are relative to the pelvis position).

The joint angles can be represented as a high dimensional vector (j) of M concatenated 3-D Euler joint angles. Formally, this can be defined as,

$$\bar{q} = (\bar{\Theta}_1, ...\bar{\Theta}_M) \quad (1)$$

$$\bar{\Theta}_j = (\theta_{j,1}, \theta_{j,2}, \theta_{j,3}) \quad (2)$$

where $\bar{\Theta}_j$ is the Euler joint angles triplet for the j^{th} body joint. Therefore, in total, the dimensionality of the joint angle vector is $3M$. Alternative representations for the joint angles are quaternians and exponential maps[7]. However, as can be seen in Section 3.4, our methods avoid the problems that arise out of using Euler angles (e.g. wrap-around effects).

Applying the above joint angles ($\bar{p}$) to a skeleton model using forward kinematics, we can obtain its 3-D joint positions. The positional constraint vector ($\bar{p}$) is then a concatenation of a subset number (N) of these 3-D joint positions. Formally, we denote positional constraints vector as,

$$\bar{p} = (x_i, y_i, z_i : i = \{1...N\}) \quad (3)$$

where (x_i, y_i, z_i) is the x,y,z coordinates of the i^{th} chosen body part.

3.2. Localised Cluster

For the work in this paper, we have chosen clusters with diagonal covariance matrices. Clusters with uniform radius

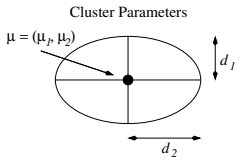

Figure 2: *An illustration of the parameters of a two-dimensional cluster.*

were not chosen since the dimensionality of the data modelled can be fairly large. In such situations, the radii of a spherical cluster can become very large, causing the cluster to capture many invalid body configurations. Alternatively, one can use clusters with full covariance matrices. However, the number of parameters of the clusters can dramatically increase. We have found that for our experiments, diagonal covariance matrix clusters can provide a good trade-off between the flexibility offered by full covariance clusters and the simplicity of spherical clusters.

In general, a cluster ($\mathbf{c}$) in a D dimensional space, provides a localised constraint centred at a particular location ($\bar{\mu}$) and covers the area denoted by the diagonal values of the covariance matrix ($\bar{d}$). The parameters of the clusters can be seen in Figure 2.

3.3. Dual Layer Constraint Clusters

The learnt model can then be built using a two-layer hierarchical cluster model (see Figure 3).

The first layer is responsible for modelling the positional constraints. This is achieved using a cluster model. The cluster model effectively partitions the valid positional-constraints-space into a number of localised regions. To differentiate the clusters of the first and second layer, a subscript is added. For identifying clusters that belong to the first layer, we use the subscript $\mathbf{p}$.

The i^{th} positional constraint cluster in the first layer can be defined as $\mathbf{c}_{\mathbf{p},i} = (\bar{\mu}_{\mathbf{p},i}, \bar{d}_{\mathbf{p},i})$. The number of clusters on the first layer is defined as C_N. The dimensionality ($D_\mathbf{p} = 3N$) of the clusters is the same as the dimensionality of the positional constraint vector, which was defined in Eq.3. The elements of the mean vector of the cluster ($\bar{\mu}_{\mathbf{p},i}$) and its diagonal covariance vector ($\bar{d}_{\mathbf{p},i}$) is defined as follows:

$$\mathbf{c}_{\mathbf{p},i} = (\bar{\mu}_{\mathbf{p},i}, \bar{d}_{\mathbf{p},i}) \quad (4)$$

$$\bar{\mu}_{\mathbf{p},i} = (\mu_{\mathbf{p},i,1}, ..., \mu_{\mathbf{p},i,D_\mathbf{p}}) \quad (5)$$

$$\bar{d}_{\mathbf{p},i} = (d_{\mathbf{p},i,1}, ..., d_{\mathbf{p},i,D_\mathbf{p}}) \quad (6)$$

where $i = \{1...C_N\}$.

For each cluster ($c_{\mathbf{p},i}$) on the first layer, using the original training data, we can locate the joint angle data that produced the positional constraints that the cluster covers. These joint angles can in turn be modelled using another set of *angle*

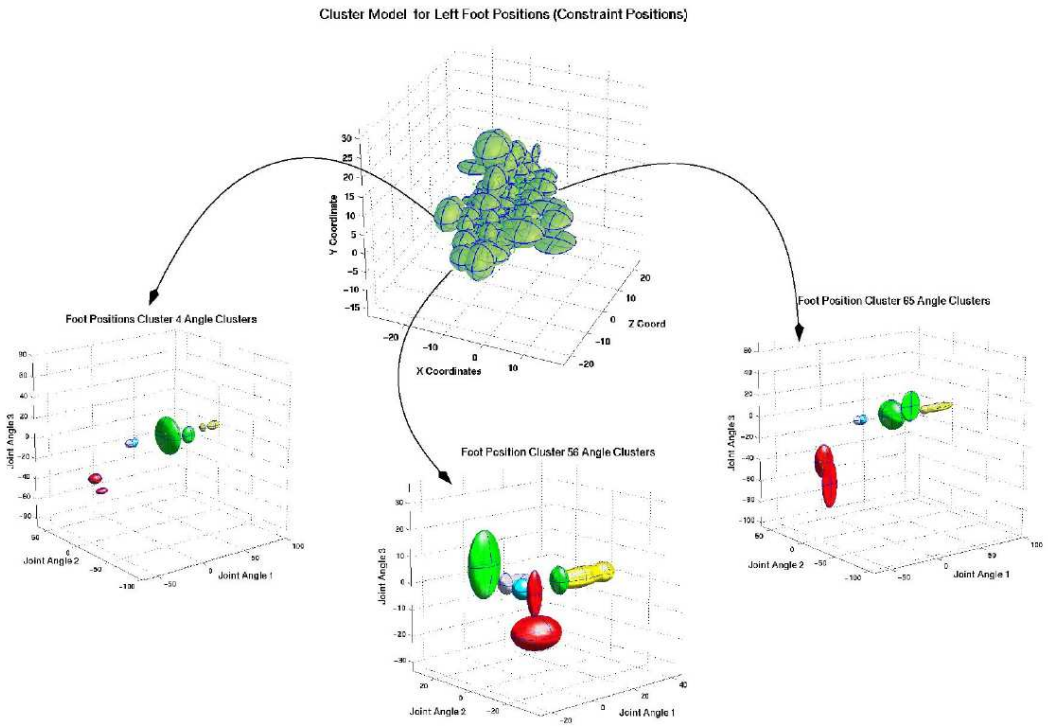

Figure 3: *An illustration of the dual layer hierarchical cluster model. Shown is the cluster model for the foot positions that are also the positional constraints. The angle clusters are shown for three different positional clusters. The angles triplets for the different joint angles are shown in different colours. However, we note that for a given constraint position cluster, we only have a small range of joint angles associated with it.*

clusters, giving each positional constraint cluster ($c_{\mathbf{p},i}$) a its own set of joint angle clusters. The number of angle clusters belonging to the i^{th} positional constraint cluster can be defined as $C_i|i = \{1...C_N\}$. The dimensionality ($D_\mathbf{q} = 3M$) of the clusters is the same as the dimensionality of the joint angle vector, which was defined in Eq.2.

Formally, the angle clusters can be defined in a similar way to its parent clusters. We use the subscript $\mathbf{q}$ to differentiate them from its parent clusters in the first layer.

The j^{th} angle cluster of the i^{th} positional constraint cluster can be defined as $\mathbf{c}_{\mathbf{q},i,j} = (\bar{\mu}_{\mathbf{q},i,j}, \mathbf{d}_{\mathbf{q},i,j})$. The elements of its mean vector of the cluster ($\bar{\mu}_{\mathbf{p},i,j}$) and its diagonal covariance vector ($\mathbf{d}_{\mathbf{p},i,j}$) is defined as follows:

$$\mathbf{c}_{\mathbf{q},i,j} = (\bar{\mu}_{\mathbf{q},i,j}, \bar{d}_{\mathbf{q},i,j}) \quad (7)$$

$$\bar{\mu}_{\mathbf{q},i,j} = (\mu_{\mathbf{q},i,j,1}, ..., \mu_{\mathbf{q},i,j,D_\mathbf{q}}) \quad (8)$$

$$\bar{d}_{\mathbf{q},i,j} = (d_{\mathbf{q},i,j,1}, ..., d_{\mathbf{q},i,j,D_\mathbf{q}}) \quad (9)$$

where $i = \{1...C_N\}$ and $j = \{1...C_i\}$.

3.4. Training the Model

In order to determine the values of the cluster parameters, a training set of a number of varying body poses are used. The specifics of the training dataset that was used can be found in Section 5. However, the elements of the training dataset take the form of pairs of high dimensional vectors (one vector for the constraint positions and another for the joint angles).

To obtain the parameters of the positional constraint cluster model, we have chosen the well-known K-means clustering method. The details of the algorithm can be found a book by Bishop[2]. We have not chosen a statistically based clustering method such as Expectation Maximisation (EM)[2] since we wish to partition the space into a number of separate regions. EM however treats each cluster as a Gaussian distribution model, instead of a localised space. As a result, models resulting from applying EM can sometimes result in clusters contained within other clusters, in order to produce the required statistical distribution indicated by the training data.

However, it was found that certain positional constraint clusters were associated with joint angle data that formed

separated clusters. This was due to the wrap-around nature of the Euler angles. We have also found that the number of separated clusters can be different, depending on which positional constraint cluster was chosen. This factor can present a problem when K-means is used for clustering, since the numbers of clusters needs to be specified beforehand. Specifying an insufficient number can result in angle clusters spanning spaces that should be modelled by two or more clusters. As a result, it captures invalid joint angles as well.

To address this, the nearest neighbour clustering algorithm [8] was chosen for building the cluster model for the joint angles. The details of this method can be found in Appendix A. One advantage in this method is that it automatically determines the required number of clusters.

4. Learnt Inverse Kinematics

With the constraint model described in the previous Section, we can now utilise it to perform inverse kinematics. Inverse kinematics can be performed in three steps: Locating the positional constraint cluster, locating the angle cluster, smoothing the output angle.

The first step uses the top-level cluster set of positional constraints. There, the clusters that encompasses a new set of constraint positions ($\bar{x} = (x_1, ..., x_{D_\mathbf{p}})$) needs to be found. Formally, an encompassing cluster ($\mathbf{c}_{\mathbf{p},i}$) is required to satisfy the following criteria:

$$|x_k - \mu_{\mathbf{p},i,k}|, \forall k | k = (1...D_\mathbf{p}) \tag{10}$$

For a particular positional constraint cluster ($\mathbf{c}_{\mathbf{p},i}$) , the set of joint angle clusters ($\mathbf{c}_{\mathbf{q},i,j} | j = 1...C_i$) under it provides the set of joint angles hypotheses. Subsequently, a more accurate joint angle can then be found between these joint angle clusters. For this, we select joint angles given by the k_i^{th} joint angle cluster centre ($\mu_{\mathbf{q},i,k_i}$) that produces the constraint positions ($\tilde{x}_{k_i}$) closest to those given ($\bar{x}$).

$$w_i = \min_{k_i} \{G\|\bar{x} - \tilde{x}_{i,j}\| + Fz_i | j = 1...C_i\} \tag{11}$$

$$\tilde{x_{i,j}} = Q(\mu_{\mathbf{q},i,j}) \tag{12}$$

$\mathbf{Q}(\bar{p})$ is the standard forward kinematics function that transforms a set of joint angles ($\bar{p}$) into a set of 3-D positions and subsequently selects only the N (Section 3.1) number of subset of 3-D positions required for the positional constraints. Additionally, we require that the joint angle produce a skeleton that is as similar to that estimated in the previous frames. In order to achieve this, we introduce another variable, z_i, which is the difference in the skeleton vertex *positions* of the previous frame and that produced by using the joint angle of a cluster. Finally, to weight the two measurements are weighted by the pre-defined constants G and F. In this paper, the two constants were determined heuristically.

Finally, to overcome jitter in the approximations between different frames, the joint angles can be smoothed by taking a weighted sum between the current ($\bar{q}_t$) and previously estimated joint angles ($\bar{q}_{t-1}$). To achieve this, we firstly convert the two joint angles into the exponential map representation [7], resulting in $e_t^-{}_1$ and $\bar{e}_t$ for the previous and current frames' joint angles respectively. We can then smooth the weight the exponential maps by:

$$\bar{e}_t^{new} = A\bar{e}_{t-1} + B\bar{e}_t \tag{13}$$

where A and B are the weights for the previous and current frames' joint angles respectively. Following Eq.13, we reconvert the result ($\bar{e}_t^{new}$) back into the original Euler angle format.

The procedure for performing inverse kinematics using the two-layer constraint cluster model can be summarised as follows:

1. Locate positional constraint cluster: Get the set of P number of positional constraint cluster indices ($\bar{I} = \{I_1, ..., I_P\}$) for those that satisfies Eq.10.
2. Find the angle cluster: For each of the positional constraint clusters found in the previous step, find the one joint angle cluster that is closest to the given constraint position using Eq.12. Group the results into triplets (I, K, D), where I is the index of the positional cluster. K and D is the joint angle cluster index (k_i) and distance d_i given in Eq.12respectively. Therefore, the best joint angle for the current frame can be found by choosing the triplet with the smallest D value, which is $\mu_{\mathbf{q},I,K}$.
3. Smooth the joint angle: Finalise the joint angle by applying a smoothing function in Eq.13.

Having described the learnt constraint model and the method in which we can use it for performing inverse kinematics, the next section will provide results on its various applications.

5. Experiments

This section will present a description experiments carried out and its results on both full body animation reconstructions and synthesis. Firstly, we provide a description of the training data and learnt model that we have used for the experiments.

Following this, we show two reconstruction experiments. The first experiment (Section 5.2) presents results on the process reconstructing "on-the-spot" motions using only a small set of end-effector positions and a root pelvis position. These include the motion of a dog jumping on the spot and a man hurling an object. Next, we extend the movement to a walking animation of the entire body (Section 5.3). Finally, we will show how the end effector positions of both the dog and human body can be modified in a simple way to retarget the animation to an uneven terrain scenario (Section 5.4).

5.1. Experimental Setup

We have chosen to split a body's learnt model into a number of independent models. Although there are generally cor-

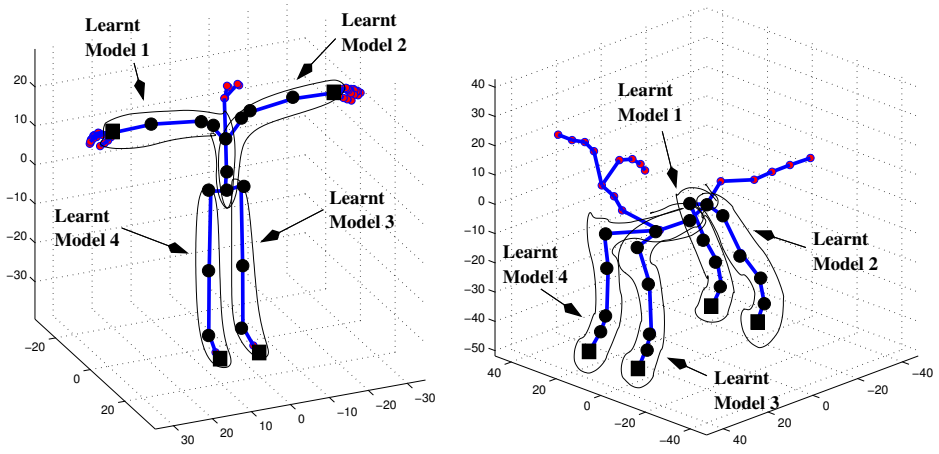

Figure 4: *This figure shows the different skeleton structure for both the dog and human body. Also shown are the vertices (joint and end-effectors) that were selected for the parameters of the 4 learnt models for both the dog and human body that were used in the experiments. The constraint positions are indicated using filled squares, while the joint angles are indicated using filled circles.*

relations between the different limbs for a body in motion, better generalisation can be provided through independent learnt models. This is caused by the requirement for the availability of training data covering all the combinations of the different limb configurations. Consequently, the required number of training data would increase dramatically, more than what that is available to us. Therefore, for both the dog and human body, one learnt model was chosen for each limb. For the human body, this gives us a total of 4 learnt models, corresponding to one for each leg and one for each arm. For the dog, we again chose to use 4 learnt models, one for each leg. In this work, we have not modelled the constraints for the tail.

To provide the training data, the commercial motion capture library, Muybridge, was used to provide 28 sequences of varied lengths of motion-captured human body walking animation sequences. In total, all the sequences provided 2790 examples of human body poses undergoing various styles of walking motions. For the dog, various animation sequences from the animal motion capture library from Credo Software was used. A total of 15 sequences were chosen, resulting in a database of 893 examples.

The skeleton structure of the human body and the dog consisted of 67 vertices and 39 vertices respectively. The parameters (constraint positions and joint angles) for each of the learnt models can be seen in Figure 4. The learnt models are produced using the method proposed in Section 3.

We shall now describe the experiments that were carried out to test the animation synthesis capability of the method described in Section 4.

5.2. Reconstruction I: On-the-spot Motions

The first experiment was carried out to test the learnt models' reconstructive ability from an existing animation of minimal constraint positions. In this experiment, a jumping-on-the-spot motion sequence was selected for the dog, while a throwing motion was selected for the human body. However, only the positions of the pelvis and end-effector positions were kept. For the human body, the end-effector positions included the hand and feet positions. For the dog, the end-effector positions were the positions of the four feet. These positions corresponded directly with the original animation's results. Additionally, these then provided the constraint positions for the individual learnt models. The results of the animation reconstructions using the learnt models can be seen in Figure 5. From this, we can see that an existing animation can be reconstructed from only a small subset of its original vertices.

5.3. Reconstruction II: Walking Motions

The second experiment was carried out to further test the learnt models' reconstructive ability from an existing animation of constraint positions. For this experiment, a walk sequence for both the human and dog was chosen. The information that was kept from the walk sequences is the same as that of the previous section (i.e. end-effector and pelvis positions). The results of the animation reconstructions can be seen in Figure 6. From this, we can again see that an existing animation can be reconstructed from only a small subset of its original vertices.

5.4. Retargeting: Uneven Terrain Walking

Finally, we carried out experiments that retargeted an existing walking animation to an uneven terrain surface. All of the previous experiments were carried out on an even surface, as the original walk sequence was captured on a flat surface. Therefore, the footsteps and hand positions for all the walk sequences used in the previous experiment were used and modified to an uneven terrain. The details of this method is described in Appendix B. The results of the retargeted animations can be see in Figure 7 and 8. We see from Figure 7 that we still have a fairly reasonable reconstruction of a person walking across an uneven terrain. There are a small number of instances where the feet position was estimated incorrectly, resulting in the foot going through the ground. However, in Figure 8 showing the reconstruction for the dog, the results show problems caused by the lack of more diverse training examples. The errors in the foot positions start to show up more, where the foot can been seen to go through the ground plane.

6. Conclusions

In this work, a learning-based approach was used to tackle the problem of human motion synthesis. To this end, we proposed to model the kinematic constraints of an articulated body structure, using a dual-layer hierarchical cluster model. We have then shown how this cluster model can be used to perform a form of "learning-based kinematics". This allows one to easily generate animations from a limited set of end-effector and root joint positions.

The main feature of this work lies in the simplicity of using a learnt cluster model to infer joint angles from position constraints, or inverse kinematics. However, it has to be noted that this differs from the typical approach to inverse kinematics. In effect, we are reconstructing poses from end-effector positions based on real poses, since the cluster models were constructed from motion-captured data.

Additionally, we show that the kinematic constraints of both the body of a dog and a human can be automatically modelled using clustering methods. For the issue where the estimations were incorrect, one solution could be to incorporate blending approaches like work by Rose et al.[14]. In such a situation, a cluster of joint angles could generate a number of examples that can subsequently be blended to produce a more accurate result.

Future work can be applied to exploring the possible advantage that this learning based approach is not restricted to only the human body structure. It can instead be used on other articulated objects (e.g. animals), provided there is adequate training information available. Another area that warrants further research is in the use of explicit dynamics. However, it is not clear at present how such information can be incorporated into the cluster model.

References

[1] O. Arikan and D. Forsyth. Interactive motion generation from examples. In *Proceedings of ACM SIGGRAPH 2002*, pages 483–490, July 2002. 2

[2] C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995. 4, 4

[3] R. Bowden. Learning statistical models of human motion. In *IEEE Workshop on Human Modelling, Analysis and Synthesis, CVPR2000*, 2000. 2

[4] M. Brand and A. Hertzmann. Style machinese. In *Proceedings of ACM SIGGRAPH 2000*, pages 183–192, July 2000. 2

[5] J. Lee et al. Interactive control of avatars animated with human motion data. In *Proceedings of ACM SIGGRAPH 2002*, pages 491–500, July 2002. 2

[6] A. Galata, N. Johnson, and D. Hogg. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding Journal*, 81(3):398–413, March 2001. 2

[7] F. Grassia. Practical parameterization of rotations using the exponential map. *The Journal of Graphics Tools*, 3(3), 1998. 3, 5

[8] A. Jain and R.Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988. 5

[9] I. Karalouva, P. Hall, and A. Marshall. A hierarchical model of dynamics for tracking people with a single video camera. In *Proceedings of British Machine Vision Conference 2000*, 2000. 2

[10] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of ACM SIGGRAPH 2002*, pages 473–482, July 2002. 2

[11] Alexis Lamouret and Michiel van de Panne. Motion synthesis by example. In *Computer Animation and Simulation '96*, pages 199–212, 1996. 2

[12] L. Molina-Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Workshop on Human Motion (HUMO'00)*, pages 137–142, 2000. 2

[13] E. Ong and S. Gong. The dynamics of linear combinations. *Image and Vision Computing*, 20(5–6):397–414, 2002. 2

[14] C. Rose, P. Sloan, and M. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. In *Proceedings of EUROGRAPHICS 2001*, 2001. 2, 7

Appendix A: Nearest Neighbour Clustering

Suppose we wish to cluster a dataset $\bar{T} = \{\bar{t}_1, ..., \bar{t}_{N_T}\}$ of N_T number of datapoints ($\bar{t_n}$). Before the clustering, we need to

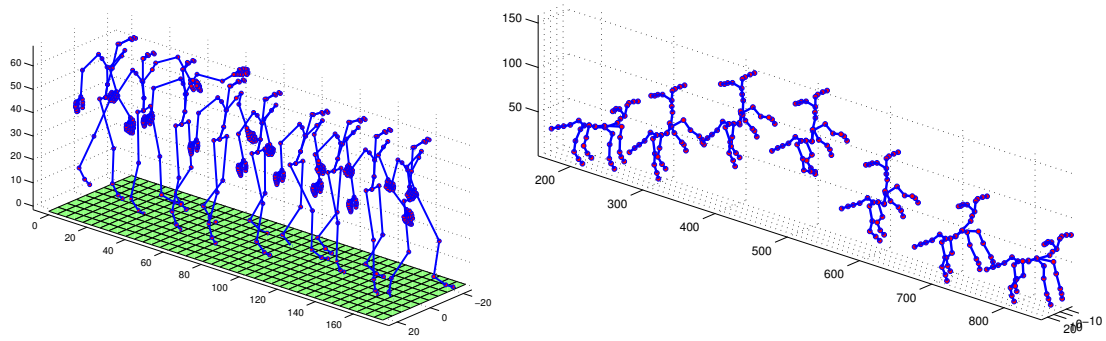

Figure 5: *Synthesising existing on-the-spot motions using just the end-effector and pelvis positions. In the figure, the frames shown are spanned across the z axes for visual clarity. The actual motion does not involve movement across the z axis.*

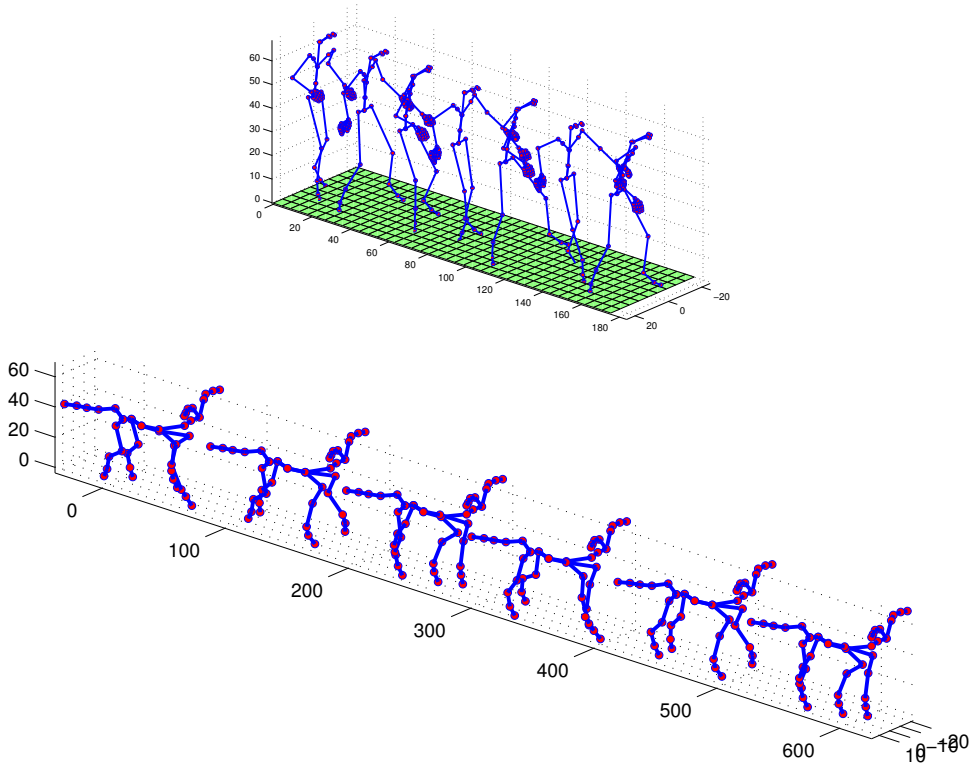

Figure 6: *Synthesising an existing walking animation for the human body and dog from just its constraint positions (end-effector and pelvis positions).*

pre-define a distance-to-cluster tolerance value (D) and an initial cluster size (s).

During clustering, we define a cluster as a set of training points. The number of clusters is defined as C. The nearest cluster algorithm can be defined as follows:

Initialisation:

Make a cluster ($\mathbf{c}_1$) with one member, $\bar{t}_1$.
Set $C = 1$.

Clustering Loop:

For all the training data examples ($\bar{t}_n | n = \{1..., N_T\}$),
Get the cluster ($\mathbf{c}_o$) which has a member that is closer to $\bar{t}_n$ than any other members of all other clusters. The distance between them is this member and $\bar{t}_n$ is d.
if the $d < D$,
Make $\bar{t}_n$ a member of cluster $\mathbf{c}_o$.
else,
Add a new cluster $\mathbf{c}_{C+1}$, with one member, $\bar{t}_n$.

Increase the number of clusters, $C = C + 1$.

Retrieving the Cluster Parameters

For each of the clusters found $\mathbf{c}_o | o = \{1...C\}$,

The mean of the cluster is the average of its members.

The diagonal covariances is the standard deviation of the elements of its members.

Appendix B: Uneven Terrain Footstep Modification Algorithm

We assume that the terrain is modelled by a height function $\mathbf{H}(x,z)$ where (x,z) is the point on the ground. The function would then provide the elevation of the terrain at that point.

We assume that we are given the original positions of the left and right feet, (x_l, y_l, z_l) and (x_r, y_r, z_l) respectively and a pelvis position (x_p, y_p, z_p). We next need to detect the foot that is on the ground. This is done by detecting which foot has a y value below a pre-defined "on-ground" value (G). Suppose the 3-D coordinates of the on-ground foot is (x_g, y_g, z_g) and the off-ground foot is (x_o, y_o, z_o). The displacement height for the entire skeleton is then $h = \mathbf{H}(x_g, z_g)$.

However, this can still result in the off-ground foot stepping inside the ground, especially if the ground ahead or behind is raised. Therefore, we check to see if $y_o + h < \mathbf{H}(x_o, z_o)$. If it is, we set $y_o = \mathbf{H}(x_o, z_o) + e$, where e is a pre-defined value determining the minimum height an off-ground foot should be above the ground.

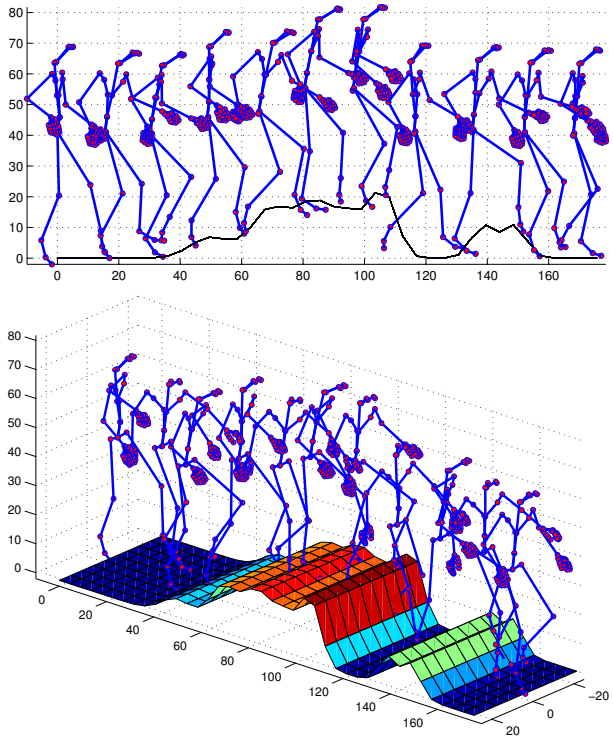

Figure 7: *Results of retargeting the original human walk sequence's footsteps and hand positions to an uneven terrain.*

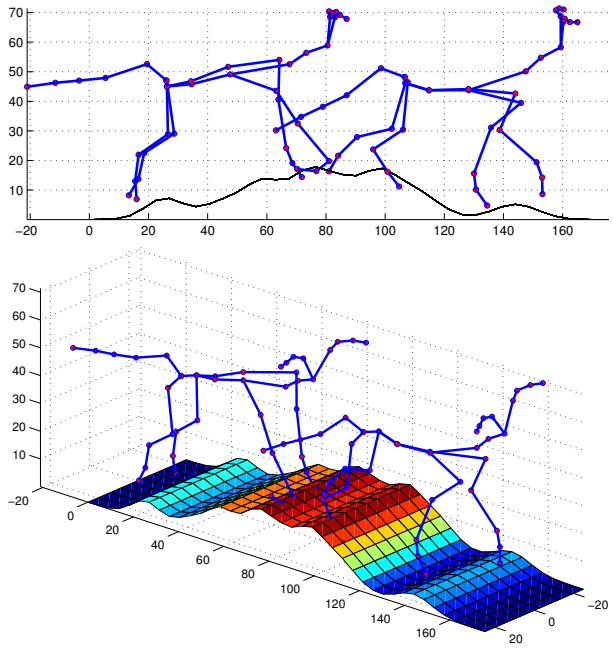

Figure 8: *Results of retargeting the original dog walk sequence's footsteps and hand positions to an uneven terrain*