Learnt inverse kinematics for animation synthesis

Eng-Jon Ong *, Adrian Hilton

Centre for Vision, Speech and Signal Processing, SEPS, University of Surrey, Guildford GU2 7XH, UK

Received 8 February 2006; received in revised form 14 April 2006; accepted 7 July 2006

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 simultaneous modelling of 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 on 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.

Keywords: Learnt inverse kinematics; Animation synthesis; Hierarchical clustering

1. Introduction

The subject of realistic motion synthesis, usually for human bodies, has been a popular and long-standing 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 with the capability of automatically modeling 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 bodies of dogs and humans.

One can also think of the body joint positions as a form of 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 2. The application of the constraint models for performing inverse kinematics and animation synthesis is then provided in Section 3. In Section 4, experimental results are shown before concluding in Section 5.

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 [1] 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 [2], where a system that performed interpolated between specified start and end key-frames was described. The pre-processing 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. [3] presented work where they generated animations of a human figure undergoing different actions, such as climbing and walking on boxes. Similar motion 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. [4] and Arikan et al. [5] 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. [6] used existing motion data and interpolated 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 [7] and Bowden [8] built statistical models through a combination of initial PCA and subsequent clustering on motion-capture data. Each cluster can then be thought of as a physical constraint model joint limits of the human body. An alternative approach to modeling joint limits was proposed by Herda et al. [9] using implicit models. Recently, Grochow et al. [10] used the novel probabilistic model called the Scaled Gaussian Process Latent Variable Model to automatically learn the dynamics of the inverse kinematics process for various motion-styles. They applied this model to various applications ranging from posing from a 2D image to interactive character posing.

Following this, the temporal characteristics of the motions can be modeled 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. [11] and Karlouva et al. [12]. Additionally, Brand and Hertzmann [13] extended the approach by proposing a learning algorithm that identifies different styles in a dataset of animations. Alternatively, Urtasun et al. [14] proposes the approach of modeling the motion capture sequence by means of PCA. To achieve this, a number of motion capture sequences were obtained and ensured to contain the same number of samples. The information in each motion sequence (i.e., joint angles across all frames in the sequence) is then concatenated into a single high-dimensional vector. PCA was then used to model the set of motion-sequence vectors and novel sequences generated via linear combinations of the extracted motion-sequence eigenvectors. It was then shown that this allowed one to extrapolate realistic animations of a person running or walking at different speeds and distances.

Our approach follows the learnt model methodology by proposing a kinematic constraint model learnt from motion-capture data. Such a constraint model can then be utilized to synthesize animations given only a minimal set of constraint positions over time. Following a quick explanation on the chosen notation in the next section, we will describe the learnt kinematic constraint model in Section 2.

2. Learnt constraint model

2.1. Notation

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

2.2. 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 Fig. 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 in a kinematic skeleton model, although sometimes they are also included root positions such as the pelvis position. We also note that the positional 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 \( \mathbf{j} \) of \( M \) concatenated 3-D Euler joint angles. Formally, this can be defined as

\[
\mathbf{q} = (\overline{\mathbf{q}}_1, \ldots, \overline{\mathbf{q}}_M)
\]

\[
\overline{\mathbf{q}}_j = (\theta_{j,1}, \theta_{j,2}, \theta_{j,3})
\]

where \( \overline{\mathbf{q}}_j \) 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 quaternions and exponential maps [15]. However, as can be seen in Section 2.5, our method avoids the problems that arise out of using Euler angles (e.g., wrap-around effects).

Applying the above joint angles \( \mathbf{q} \) to a skeleton model using forward kinematics, we can obtain its 3-D joint positions. The positional constraint vector \( \mathbf{p} \) is then a concatenation of a subset number \( N \)
of these 3-D joint positions. Formally, we denote positional constraints vector as,
\[ \mathbf{p} = (x_i, y_i, z_i : i = 1, \ldots, N) \] (3)
where \((x_i, y_i, z_i)\) is the \(x, y, z\) coordinates of the \(i\)th chosen body part.

2.3. Localised clusters

For the work in this paper, we have chosen clusters with diagonal covariance matrices. Clusters with uniform radius 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 \((\mathbf{\mu})\) and encompasses the sub-space denoted by the diagonal values of the covariance matrix \((\mathbf{d})\). The parameters of the clusters can be seen in Fig. 2.

2.4. Dual layer constraint clusters

The learnt model can then be built using a two-layer hierarchical cluster model (see Fig. 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 \(p\).

![Fig. 2. An illustration of the parameters of a two-dimensional cluster.](image-url)
The $i$th positional constraint cluster in the first layer can be defined as $c_{p,i} = (\bar{p}_{p,i}, d_{p,i})$. The number of clusters on the first layer is defined as $C_N$. The dimensionality ($D_p = 3N$) 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 $(\bar{p}_{p,i})$ and its diagonal covariance vector $(d_{p,i})$ is defined as follows:

$$ c_{p,i} = (\bar{p}_{p,i}, d_{p,i}) $$  \hspace{1cm} (4)

$$ \bar{p}_{p,i} = (\mu_{p,i,1}, \ldots, \mu_{p,i,3N}) $$  \hspace{1cm} (5)

$$ d_{p,i} = (d_{p,i,1}, \ldots, d_{p,i,3N}) $$  \hspace{1cm} (6)

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

For each cluster $(c_{p,i})$ 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 clusters, giving each positional constraint cluster $(c_{p,i})$ its own set of joint angle clusters. The number of angle clusters belonging to the $i$th positional constraint cluster can be defined as $C_{i\mid i} = \{1, \ldots, C_N\}$. The dimensionality ($D_q = 3M$) 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 $q$ 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 $c_{q,i,j} = (\bar{p}_{q,i,j}, d_{q,i,j})$. The elements of its mean vector of the cluster $(\bar{p}_{q,i,j})$ and its diagonal covariance vector $(d_{q,i,j})$ is defined as follows:

$$ c_{q,i,j} = (\bar{p}_{q,i,j}, d_{q,i,j}) $$  \hspace{1cm} (7)

$$ \bar{p}_{q,i,j} = (\mu_{q,i,j,1}, \ldots, \mu_{q,i,j,3M}) $$  \hspace{1cm} (8)

$$ d_{q,i,j} = (d_{q,i,j,1}, \ldots, d_{q,i,j,3M}) $$  \hspace{1cm} (9)

Fig. 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.

Please cite this article as: Eng-Jon Ong, Adrian Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models (2006), doi:10.1016/j.gmod.2006.07.004
where \( i = \{1, \ldots, C_N\} \) and \( j = \{1, \ldots, C_I\} \).

### 2.5. 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 4. 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 is given in [16]. A statistically based clustering method such as Expectation Maximisation (EM) [16] was not chosen since our aim is 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 [17] 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.

### 3. 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 \((\hat{x} = (x_1, \ldots, x_D))\) needs to be found.

---

Fig. 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.

---

Please cite this article as: Eng-Jon Ong, Adrian Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models (2006), doi:10.1016/j.gmod.2006.07.004
Formally, an encompassing cluster \((c_{p,i})\) is required to satisfy the following criteria:

\[ |x_k - \mu_{p,i,k}|, \forall k | k = (1 \ldots D_p) \]  

(10)

For a particular positional constraint cluster \((c_{p,i})\), the set of joint angle clusters \((c_{q,i,j}) j = 1, \ldots, C_i\) 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 \(k\)th joint angle cluster centre \((\mu_{q,i,k})\) that produces the constraint positions \((\tilde{x}_i)\) closest to those given \((\bar{x})\).

\[ w_i = \min_{k_i} \{G||\bar{x} - \tilde{x}_i|| + Fz_i| j = 1, \ldots, C_i\} \]  

(11)

\[ \tilde{x}_{i,j} = Q(\mu_{q,i,j}) \]  

(12)

\(Q(\bar{p})\) 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 2.2) 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, 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

Fig. 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.

Please cite this article as: Eng-Jon Ong, Adrian Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models (2006), doi:10.1016/j.gmod.2006.07.004
can be smoothed by taking a weighted sum between a number of successive frames. We have found that two frames work well: the current \( \bar{\theta}_t \) and previously estimated joint angles \( \bar{\theta}_{t-1} \).

However, we note that it is possible that there will be times where a larger number of frames are necessary, for example in pose spaces where training data is sparse. To perform the smoothing, we firstly convert the two joint angles into the exponential map representation [15], resulting in \( \bar{e}_t \) and \( \bar{e}_{t-1} \), for the previous and current frames’ joint angles, respectively. We can then smooth the weight the exponential maps by:

\[
\bar{e}^\text{new}_t = A \bar{e}_{t-1} + B \bar{e}_t
\]

where \( A \) and \( B \) are the weights for the previous and current frames’ joint angles, respectively. At the moment, we have heuristically defined the values of \( A \) and \( B \) to be 0.6 and 0.4, respectively. Following Eq. (13), we reconvert the result \( \bar{e}^\text{new}_t \) back into the original Euler angle format.

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

1. Locate positional constraint cluster: Get the set of \( P \) number of positional constraint cluster indices \( (I = \{I_1, \ldots, I_P\}) \) for those that satisfy 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. (12), respectively. Therefore, the best joint angle for the current frame can be found by choosing the triplet with the smallest \( D \) value, which is \( k_{q,I,K} \).

3. Smooth the joint angle: Finalise the joint angle by applying a smoothing function in Eq. (13).

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

Please cite this article as: Eng-Jon Ong, Adrian Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models (2006), doi:10.1016/j.gmod.2006.07.004
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.

4. Experiments

This section will present a description of experiments carried out and results on both full body animation reconstructions and synthesis (video sequences of the results are also available by contacting the author). 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 4.2) 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 4.3). 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 4.4).

4.1. Experimental setup

We have chosen to split a body’s learnt model into a number of independent models. Although there are generally correlations between the different limbs for...
a body in motion, better generalisation can be provid-
ed 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.
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 mod-
els, one for each leg. In this work, we have not mod-
elled 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 Fig. 4. The learnt models are produced
using the method proposed in Section 2.
We shall now describe the experiments that were
carried out to test the animation synthesis capability
of the method described in Section 3.

4.2. 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

![Fig. 8. Results of retargeting the original dog walk sequence's footsteps and hand positions to an uneven terrain.](image-url)
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 Fig. 5. From this, we can see that an existing animation can be reconstructed from only a small subset of its original vertices.

4.3. 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 were 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 Fig. 6. From this, we can again see that an existing animation can be reconstructed from only a small subset of its original vertices.

4.4. 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 Figs. 7 and 8. We see from Fig. 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 Fig. 8 showing the reconstruction for the dog, the results show problems caused by the lack of more diverse training examples. The small errors in the foot positions start to show up more, where the foot can be seen to go through the ground plane. However, this can be corrected very quickly and unambiguously using traditional IK methods, as the object configuration is already very close to the correct pose.

5. 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. [6]. 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.

Appendix A. Nearest neighbour clustering

Suppose we wish to cluster a dataset \( \mathcal{T} = \{t_1, \ldots, t_N\} \) of \( N \) number of data-points \((i_a)\). Before the clustering, we need to 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:

**Initialization:**

Make a cluster ($c_1$) with one member, $i_1$.

Set $C = 1$.

**Clustering loop:**

For all the training data examples ($i_n | n = \{1, \ldots, N_T\}$),

Get the cluster ($c_0$) which has a member that is closer to $i_n$ than any other members of all other clusters. The distance between them is this member and $i_n$ is $d$, if the $d < D$,

Make $i_n$ a member of cluster $c_0$.

else,

Add a new cluster $c_{C+1}$, with one member, $i_n$.

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

**Retrieving the cluster parameters**

For each of the clusters found $c_0 | o = \{1, \ldots, C\}$, 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 $H(x, z)$ 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 $(x_r, y_r, z_r)$, 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 $(x_o, y_o, z_o)$ and the off-ground foot is $(x_o, y_o, z_o)$. The displacement height for the entire skeleton is then $h = H(x_o, z_o)$.

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 $y_o + h < H(x_o, z_o)$. If it is, we set $y_o = H(x_o, z_o) + e$, where $e$ is a pre-defined value determining the minimum height an off-ground foot should be above the ground.

References


Please cite this article as: Eng-Jon Ong, Adrian Hilton, Learnt inverse kinematics for animation synthesis, Graphical Models (2006), doi:10.1016/j.gmod.2006.07.004