Pacific Graphics 2013
B. Levy, X. Tong, and KK. Yin
(Guest Editors)

Volume 32 (2013), Number 7

Topology Aware Data-Driven Inverse Kinematics

Edmond S. L. Ho[†1], Hubert P. H. Shum[2], Yiu-ming Cheung[1] and P. C. Yuen[1]

[1]Department of Computer Science, Hong Kong Baptist University
[2]Northumbria University, United Kingdom

Abstract

Creating realistic human movement is a time consuming and labour intensive task. The major difficulty is that the user has to edit individual joints while maintaining an overall realistic and collision free posture. Previous research suggests the use of data-driven inverse kinematics, such that one can focus on the control of a few joints, while the system automatically composes a natural posture. However, as a common problem of kinematics synthesis, penetration of body parts is difficult to avoid in complex movements. In this paper, we propose a new data-driven inverse kinematics framework that conserves the topology of the synthesizing postures. Our system monitors and regulates the topology changes using the Gauss Linking Integral (GLI), such that penetration can be efficiently prevented. As a result, complex motions with tight body movements, as well as those involving interaction with external objects, can be simulated with minimal manual intervention. Experimental results show that using our system, the user can create high quality human motion in real-time by controlling a few joints using a mouse or a multi-touch screen. The movement generated is both realistic and penetration free. Our system is best applied for interactive motion design in computer animations and games.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Applications—

1. Introduction

Human movement design is one of the most labour intensive tasks in computer graphic and animation applications. While captured motion data is now widely available, it is usually very difficult to adjust existing movement to satisfy requirements based on design purposes and environmental constraints. The animator has to coordinate all joints individually to create a natural posture, even if he/she is only interested in adjusting a few joints. Body penetrations have to be avoided and resolved throughout the editing process. Constraints on human physiology such as range of movement, as well as the logical meaning of the posture, have to be manually maintained. These processes are time consuming even for highly skilled animators.

Techniques based on inverse kinematics (IK) are proposed to ease the pain of motion editing, such that when the user designs the target locations of specific body parts of a virtual character, which are typically the end effectors, kinematics parameters such as joint angles in the skeleton hierarchy can be computed accordingly. However, it is difficult to guarantee the high level naturalness of the postures created by IK. That is, even if the posture obeys joint limits, it may still be unnatural because of the intrinsic human behaviour such as balancing and body parts correlation. To solve this problem, data-driven IK algorithms such as [GMHP04,CH05] are proposed. Using motion captured from a real human, a latent space can be created with dimensionality reduction techniques. Posture optimization is performed in such a space considering the user inputs and predefined energy functions to synthesize the required posture. Because of the use of real human motion data, the created postures are natural.

The major problem of existing data-driven IK is the difficulty to avoid body part penetration during the editing process. This is because the latent space is created by analyzing kinematic features such as joint position and orientation, instead of the topology of the motion. As a result, kinematically similar but topologically different postures, such as a posture with the right arm lying on the left one and another vice versa (Figure 2), are placed in the same latent space. When optimizing for a target posture in such a space, the posture can easily switch from one topology to another, re-

† e-mail: edmond@comp.hkbu.edu.hk

sulting in body penetration. While one can design energy terms or constraints to avoid penetration during optimization, it is highly inefficient as such evaluations introduce non-linearity into the space. In the worst case, optimization could get trapped in local minima and fail to produce the optimal posture. To remedy this, it is essential to construct a more representative latent space, instead of simply adding optimization constraints.

In this paper, we propose a topology aware data-driven IK algorithm that creates realistic human movement based on user controls, while intelligently conserving the topology of the movement and avoiding penetration. We adapt the lazy learning framework and construct a low dimensional space from a subset of sample postures in a motion database during run-time. We then apply online frame-based optimization to synthesize the postures required. Unlike previous approaches, our algorithm analyzes the topology of the postures using Gauss Linking Integral when constructing the latent space. With the space consisting of postures with similar topology, optimization terms related to penetration avoidance can perform consistently.

With our proposed framework, it becomes possible to efficiently design high dimensional human movements with low dimensional input devices such as mice and multi-touch screens. The animator can focus on adjusting the key parts of the body, while the system automatically controls the rest of the joints to produce logically correct and penetration free postures. Experimental results show the creation of complex movements such as a character climbing a ladder and sitting on a chair, in which the joints have to be precisely managed in order to maintain the movement topology and avoid penetration. Our algorithm is best applied for interactive posture and movement creation in design and animation applications.

The rest of the paper is organized as follow. We first review related research in Section 2. We then give an overview of the system and highlight our major contributions in Section 3. The core of the system involves preparing a motion database (Section 4), constructing the latent space (Section 5) and optimizing for the target posture (Section 6). Experimental results and system analysis are presented in Section 7. Finally, we discuss and conclude the proposed method in Section 8.

2. Related Work

In this section, we discuss research that is closely related to this paper. We will first give a general overview on tradition IK algorithms, focusing on the different methods of IK. Readers are referred to [AL09] for a more comprehensive review of IK. Second, we review research to create natural movement using data-driven algorithms, and point out their major weaknesses. Finally, we discuss how penetration-free motion can be synthesized, and point how topology can enhance real-time motion synthesis.

2.1. Traditional Inverse Kinematics

Here, we review some of the important algorithms in traditional IK, and point out their respective weakness.

IK is used to calculate the joint parameters in the skeleton hierarchy based on a desired location of the end effector. There are several methods to estimate these parameters. A popular one is to apply numerical solvers such as the least square method [Whi69] and the singularity robust inverse method [NH86], which usually require higher computational cost and are not suitable for real-time applications. To achieve better performance, Lee et al. proposed an analytical IK solver to calculate the body posture with the given positions of the hands and the feet [LS99]. Shin et al. proposed an intelligent inverse kinematics solver that priorities multiple constraints for real-time application [SLSG01]. To deal with the temporal inconsistency across frame when applying IK, Gleicher suggested the use of spacetime constraints to synthesize high quality movement [Gle97]. A general problem of traditional IK algorithms is the difficulty to ensure the naturalness of the synthesized motion. This is because natural human motion involves a lot of subtle behaviours such as balancing and correlation of body parts, which are difficult to be modelled mathematically.

2.2. Data-Driven Inverse Kinematics

The idea of data-driven IK is to make use of captured motion data to help synthesize the required posture, such that natural movement can be created using a low dimensional control signal. Here, we review algorithms that analyze the whole motion database as an offline process, followed by those using online modelling.

To generate natural movements, one can create a natural movement space based on captured motion. Any movement synthesized in the space is then assumed to be natural. Grochow et al. created the space using scaled Gaussian process latent variable model [GMHP04]. The user used the mouse to control a few joints, and the system searched for the appropriate posture that satisfies the constraints in the latent space. Wu et al. further proposed the concept of representative postures, which are a subset of distinctive postures in the database, to deal with large motion database [WTR11]. Wei and Chai solved the same problem by constructing a mixture of factor analysis [WC11]. The algorithm segments the motion database into local regions and models each of them individually. Nevertheless, the training cost and system complexity increases with the amount of source data, and the effectiveness of dimensional reduction reduces with the increase of motion data variety.

As opposed to offline training approaches, online modelling has shown to be effective for real-time application with large motion dataset. The idea is to select a small subset of posture based on run-time information to synthesize the required posture. A naive method is to search for a best

match posture during run-time based on the user input to synthesize the character movement [SH12]. A better approach is proposed by Chai and Hodgins applies local Principal Component Analysis to construct a latent space during run-time [CH05]. A set of postures that are similar to the current one is used to construct the space, such that natural full body motion can be synthesized with a small number of positional constraints. Liu et al. extended the idea by using the maximum a posteriori framework to reconstruct the motion, which enhanced the consistency of the movement in the temporal domain [LWC*11]. The general problem of these methods is that it is difficult to ensure the set of extracted postures to be logically similar as kinematics metric is used. Postures of different topology in the space may result in invalid synthesis with self-penetration. We also apply online modelling in this paper. However, we propose a GLI metric [HK09a] to extract logically similar postures for creating the space, ensuring penetration-free movements.

2.3. Penetration-free Movement Synthesis

One of the major sources of visual artifacts in real-time motion editing system is body penetration. Here, we discuss why traditional approaches fail to address the problem, as well as how the topology-based motion synthesis is applied.

Collision detection and avoidance in the joint space is sub-optimal. While applying path planning algorithms such as Rapidly-exploring Random Trees (RRT) [LK00] in the joint space is a possible solution [SKF07], the process is computationally inefficient and tends to generate unnatural movements. This is because the joint space provides very limited guidance on how the collision could be effectively avoided, making the RRT search a trial and error process. Ho et al. proposed a spatial representation to model the relationship between a character and its surrounding environment as a mesh structure [HKT10,HS13,HCKL13]. However, because the representation is inconsistent across frames, it can only be used for adjusting an existing posture, but not synthesizing new postures that satisfy user defined constraints.

The Gauss Linking Integral (GLI) is a value describing how much two curves wrap around each other [Poh68]. Ho and Komura proposed to model a human character as a set of polygon curves, and calculate the GLI values to index postures involving two characters in a motion database [HK09b]. They further proposed the topology coordinates based on GLI for motion synthesis by interpolating topologically similar postures [HK09a]. Such a coordinate system has been shown effective in creating penetration-free movement in close interaction between two characters such as wrestling [HK11]. However, the framework cannot guarantee motion naturalness. Also, it is only suitable for movement that involves tangling of body parts/limbs, which hugely limits the variety of motion that can be synthesized. For motion that does not involve tangling, such as reaching motions, the topological constraints become less effective to control the characters. This is because the GLI values of such postures are close to zero and does not reflect the posture well. This paper proposes a new framework that combines the advantage of traditional GLI and data-driven approaches, such that different classes of natural movement can be produced.

3. System Overview

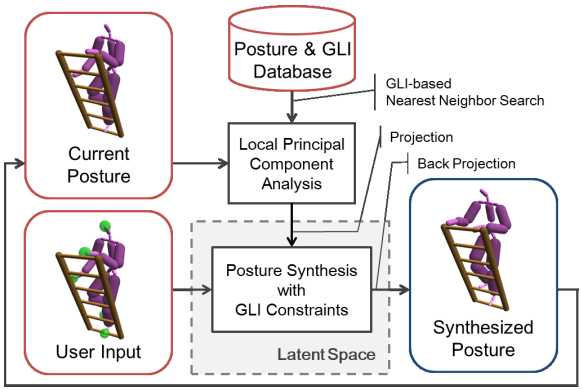

Figure 1: *The overview of the proposed framework.*

The overview of our motion creation framework, which is a continuous posture synthesis algorithm, is shown at Figure 1. We first prepare a database that includes postures in both GLI and joint angle representation. Given a current posture, we search for a set of similar postures with a GLI-based distance metric. This ensures the obtained neighbour postures to be penetration-free and logically similar to the current one. We perform local Principal Component Analysis to construct a latent space, and optimize for a resultant posture with the user inputs and GLI constraints. Because the latent space is created with captured motion, the generated posture will follow the natural human behaviour. Finally, the optimized posture is back projected to the full dimensional space and becomes the new current posture. The process is repeated to create a continuous motion.

3.1. Contributions

We have two major contributions in this paper:

- We propose a new topology aware data-driven IK algorithm that can ensure natural and penetration free synthesis for complex movements such as tight body movements and interaction with objects. Comparing to existing approaches, our real-time system has similar computational cost but more consistent simulation quality.
- We propose a framework to combine the benefits of traditional kinematic-based and topology-based motion synthesis. Our system can produce topology consistent movements. In addition, it can create a much wider variety

Posture type	Number of postures	
	Original	After filtering
Basic Movement	4539	1400
Ladder	360	170
Chair	1034	331

Table 1: *Number of postures in our motion database.*

of postures comparing to existing topology-based approaches including non-tangle movements.

4. Database Creation

In this section, we explain how we construct the motion database, which is used to enhance the naturalness of the synthesized postures in later stages.

Each record of our motion database consists of three components: (1) a normalized posture in the joint angle representation, (2) the corresponding GLI representation of the posture, and (3) an optional GLI representation of the interaction between the character and an object. We encode (2) and (3) in GLI such that our system can prevent self-penetration and penetration with objects respectively during motion synthesis.

4.1. Posture in Joint Angles

Here, we explain how we prepare the postures in joint angle representation, and how we filter the database for similar postures.

We capture human motion from traditional motion capture system, and retarget the motion into standard body size according to [Arm88] using commercial software. We then create normalized postures by removing the translation and rotation along the vertical axis of the root joint. This allows us to represent the motion database as a collection of independent postures. Each posture is encoded as a series of Euler angles that represents the orientation of each joint.

We filter the motion database for similar postures. We compare every pair of postures in the motion database, and calculate the sum of square of joint position differences obtained by Forward Kinematics. If the value is smaller than a threshold, we remove one of them. This operation has two major advantages. First, it hugely reduces the size of the motion database and enhances run-time efficiency. Second, by removing similar postures, we ensure that when searching for neighbour postures during run-time, we can have a set of postures with reasonable variation. A summary of the number of postures in our motion database before and after the filtering process is shown in Table 1.

Notice that we opt to filter the database in the joint position space, instead of the GLI representation that will be explained in the next section. This ensures posture samples to be evenly distributed in the joint position domain, and thus covers the possible target end-effector locations specified by the user.

4.2. Posture in GLI Representation

Here, we describe the process to calculate the corresponding GLI representation for each posture in the motion database.

The major problem of the joint angle representation is the incapability of representing the logical meaning of a motion. Figure 2 shows an example. The two postures are very similar in joint angle but different in topology. Synthesizing with a set of kinematically similar but topologically different postures may result in penetration. As a solution, we adapt the topological relationship based on GLI [Poh68] that represents on the logical meaning of the movement, such as whether an arm is on top of or below another [HK09a]. GLI describes how two curves twisted around each other. A large absolute GLI value means that the two curves are tangled, while a small one indicates them being nearly parallel. The sign of the value changes when two curves penetrate each other, making it a perfect choice to construct a space for penetration avoidance. Figure 3 shows the GLI values of two pairs of curves with different configurations.

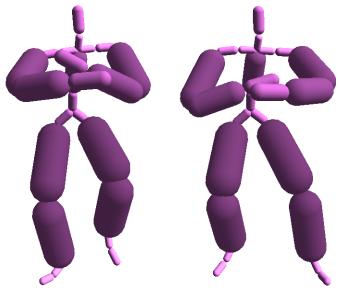

Figure 2: *A pair of postures that are kinematically similar but topologically different.*

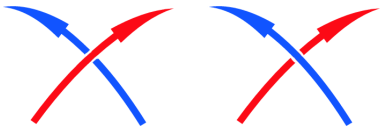

Figure 3: *Examples of curves and the corresponding GLI values.*

We adapt the algorithm from Ho and Komura [HK09b] to construct GLI representation for each posture. Here, we outline the method and highlight the key steps. The readers are referred to the mentioned paper for the details. In [HK09b], the tree structure of a character is represented using 10 polygon curves. However, we found that the representation using such an extensive number of polygon curves is more

detailed then required, making the system very sensitive to small changes of movement from the character. In this paper, we represent the major body parts of a character using 5 polygon curves as shown in Figure 4 left. They are the *head and torso*, *right arm*, *left arm*, *right leg* and *left leg* respectively. The GLI representation of a posture is defined as the set of all possible combinations of polygon curve pairs. For each pair of curves γ_1 and γ_2, the GLI value is calculated as:

$$GLI(\gamma_1,\gamma_2) = \frac{1}{4\pi}\int_{\gamma_1}\int_{\gamma_2}\frac{d\gamma_1 \times d\gamma_2 \cdot (\gamma_1-\gamma_2)}{\|\gamma_1-\gamma_2\|^3} \quad (1)$$

where $\times$ and $\cdot$ represent cross and dot product respectively.

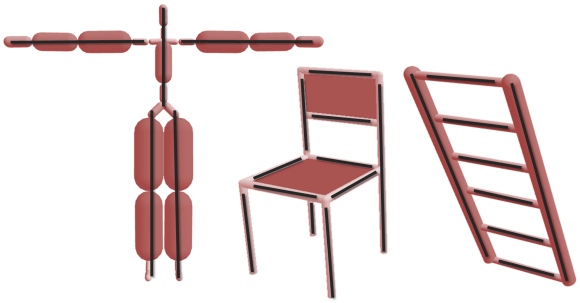

Figure 4: *The polygon curves assignment to compute GLI for the character and objects.*

Notice that as we use 5 polygon curves to represent the structure of a character, the number of possible polygon curve pairs is 10. Thus, a posture is represented as a GLI vector with 10 degrees of freedom. We implement the analytical solution proposed by [KL00] for calculating the GLI for efficient computation.

4.3. Object Interaction in GLI Representation

Here, we illustrate how the interaction between a character and an object is represented by GLI and stored in the database accordingly.

The object interaction is an optional component in our motion database. It is only used for the motion involving interaction with external objects. During motion capture, we capture both movements from the character and the object. Then, we manually design the polygon curve representation of the object as we do for the character. The interaction is encoded as the set of GLI values consisting of all pairs from the polygon curves of the character to that of the object.

We model two objects in this work. Figure 4 middle shows how a chair is modelled with 10 polygon curves, and Figure 4 right shows how a ladder is modelled with 8 polygon curves. The number of dimension of the GLI vector that represents the interaction is calculated as the multiplication of the character and the object GLI dimensions. For example, for a character climbing a ladder as shown in Figure 5, the interaction is computed by pairing the 5 character polygon curves and the 8 ladder polygon curves, resulting in a vector of $5 \times 8 = 40$ GLI dimensions.

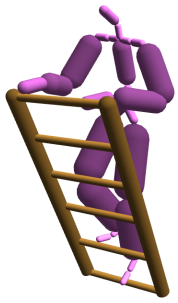

Figure 5: *The interaction between the character and the ladder is encoded with GLI values.*

Notice that assigning polygon curves to represent an object is more about art than science. However, there are two major principles to guide the process. First, the polygon curves should illustrate the major outline of the object, such that we can capture the way the character interacts with it. Second, we should use the minimum number of polygon curves to reduce run-time cost since using excessive curves to represent character and objects will not improve the quality of the resultant postures.

5. Latent Space Creation

In this section, we explain the run-time process to construct a latent space using local Principal Component Analysis (lPCA). We first detail the process assuming there is no object in the scene. Then, we explain how the system handles the object.

5.1. Local Principal Component Analysis

Here, we describe the process to search for the K nearest neighbours from the current posture, and apply PCA to reduce the dimension of the postures.

Given the current posture of a character, the first step of lPCA is to search for the K nearest neighbour in the motion database. The important issue here is that we should search for postures with similar topology. Otherwise, the latent space created will blend postures of different topology, resulting in potential penetrations. Therefore, unlike previous works such as [CH05] that search for postures solely based on the closeness in kinematics, we search for postures based on their GLI representation to obtain topologically similar postures before comparing the kinematics as in [HK09b]. We calculate the GLI representation of the current posture using Equation 1. The different between the cur-

rent posture p^c, and one of those in the database, p^d, is calculated as:

$$D(p^c, p^d) = \sum_{i=1}^{i_{total}} (GLI_i^c - GLI_i^d)^2 \quad (2)$$

where GLI_i^c and GLI_i^d are the i^{th} dimension of the GLI vectors of p^c and p^d respectively, i_{total} is the total number of dimension of the GLI vectors, which is 10 in our setup as explained in Section 4.2. Using Equation 2, we obtain a set of topologically similar postures. We further select the K nearest neighbours, which is set to 50 in our system, by comparing the similarity of joint angles between the current posture and those from the subset. We apply brute force search to extract the K postures because of the small size of the database. If a much larger database is required, KD-tree [KTWZ10] can be used to index the GLI vectors for efficient query.

We then construct the latent space with the extracted postures. Notice that we utilize the joint angle representation of the extracted posture to construct the space here. This is because GLI representation is limited to how two curves wrapping around each other, as shown in [HK09b]. If we use the GLI representation to construct the latent space, it will be difficult to synthesize movement that increases or decreases the distance of two curves, as well as movement that is parallel to the curve directions. The dimensionality of the joint angle vectors is reduced using PCA [Bis96] to create the latent space, which is set to 30 dimensions in our system. Since the latent space is created using human postures, it represents the natural style of human movement. Moreover, because the source postures are of similar GLI values, postures synthesized in the space follow the same topology.

5.2. Object Handling

Here, we explain how the system takes the object into consideration, if there is any, during the latent space construction process.

If interaction with an object is required, the process of Section 5.1 is updated to include the object. We apply the distance function in Equation 2 to compute $D(I^c, I^d)$ and $D(p^c, p^d)$, where I^c and I^d are the GLI vector for the interaction between the character and the object for the current posture and the database posture respectively. The final distance is calculated as:

$$D(I^c, I^d) + D(p^c, p^d) \quad (3)$$

With the combined distance function, we extract postures that interact with the object in a similar way. This helps to prevent penetration with objects during the motion synthesis.

When creating the latent space, we only need to consider the joint angle representation of the posture, even if there is an object in the scene. This is because we assume the object does not move by itself, and hence cannot be directly controlled.

6. Posture Synthesis

In this section, we discuss how a posture is synthesized in the latent space using posture optimization. We first detail the individual energy term to evaluate the quality of a posture. Then, we define the constraints that represent the user input. Finally, we explain the overall optimization process.

6.1. Energy Terms

Here, we explain the three energy terms designed in our system, namely the topology term, the continuity term and the style term.

The *topology term* minimizes the change of the posture topology to prevent penetration. It is defined as the change in GLI given a posture in the latent space:

$$J_{GLI} = \begin{pmatrix} \frac{\partial G_{y,1}}{\partial l_{y,1}} & \cdots & \frac{\partial G_{y,1}}{\partial l_{y,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial G_{y,a}}{\partial l_{y,1}} & \cdots & \frac{\partial G_{y,a}}{\partial l_{y,n}} \end{pmatrix} \quad (4)$$

$$E_t = \left|\left| J_{GLI} \cdot \dot{l_y} \right|\right| \quad (5)$$

where l_y is a vector defining a point at the latent space, $\dot{l_y}$ is its derivatives, J_{GLI} is the Jacobian of the GLI derivatives with respect to $\dot{l_y}$ computed by finite differencing, a and n are the dimensionality of the GLI and latent representations respectively, $G_{y,i}$ represents the i^{th} pairwise GLI value and $1 \leq i \leq a$, $J_{GLI} \cdot \dot{l_y}$ returns a vector consisting of the GLI values in the full dimensional space.

Notice that while the latent space is created using posture samples with similar topology, the space itself is infinitely large. Postures synthesized at region far away from the posture samples may still be invalid. That is why we implement the topology term to guide the optimization process. Also, it is advisable to implement E_t as an energy term, instead of a constraint term. This is because the positional constraints to be explained in Section 6.2 may conflict with the topology constraints. By implement E_t as an energy term, we allow small topology changes such that the character can achieve the required target postures, while penalizing heavily on penetration when the sign of the GLI values is changed, as explained in Section 4.2.

The *continuity term* ensures smooth movement to be synthesized. It is defined as the positional difference between the synthesized posture and the current posture:

$$J_{pos} = \begin{pmatrix} \frac{\partial p_{y,1}}{\partial l_{y,1}} & \cdots & \frac{\partial p_{y,1}}{\partial l_{y,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{y,m}}{\partial l_{y,1}} & \cdots & \frac{\partial p_{y,m}}{\partial l_{y,n}} \end{pmatrix} \quad (6)$$

$$\dot{p_y} = J_{pos} \dot{l_y} \quad (7)$$

$$E_m = ||\dot{p_y}|| \quad (8)$$

where J_{pos} is the Jacobian of the joint position derivatives

with respect to $\dot{l}_y$ computed by finite differencing, m and n are the number of joints and latent variables respectively, $\dot{p}_y$ represents the difference between the input posture and the edited posture with respect to $\dot{l}_y$.

The *style term* encourages the optimization to be performed at the center of the latent space, where there are more sample postures, such that the style of the sample postures can be best maintained. It is defined as:

$$E_s = ||\dot{l}_y|| \tag{9}$$

6.2. Constraints

Here, we define the constraints designed in our system, which are used to achieve the requirements given by the user.

We setup positional constraints to control the posture of the characters. It is defined as:

$$J_{jpos,j} = \left(\frac{\partial p_{j,y}}{\partial l_{y,1}} \quad \cdots \quad \frac{\partial p_{j,y}}{\partial l_{y,n}} \right) \tag{10}$$

$$target_j - p_{j,y} = J_{jpos,j} \cdot \dot{l}_y \tag{11}$$

where $J_{jpos,j}$ is the Jacobian of the position derivatives of joint j with respect to $\dot{l}_y$ computed by finite differencing to obtain the locally linear relationship between $J_{jpos,j}$ and $\dot{l}_y$, n is the dimensionality of the latent representation, $p_{j,y}$ is the position of joint j for a posture y, $target_j$ is the target 3D position of the joint j defined by the user.

Notice that the user can control multiple joints at the same time. In such a case, multiple constraints are defined. The optimization engine will then try to optimize for a posture that fit into all constraints.

While it is possible for the user to over-constrain the system, it rarely happens during run-time since a few control points can usually achieve the desired posture. In fact, the beauty of our system is that the user can create natural movement without controlling a large number of joints.

6.3. Optimization

Here, we explain how we synthesize the final posture using optimization.

We optimize the new posture by minimizing the energy terms while satisfying the positional constraints. For the ease of explanation, all of the hard constraints explained in Section 6.2 are represented by $h = \mathcal{H}l_y$ and the optimization problem is defined as:

$$\begin{aligned} \underset{\dot{l}_y}{\text{minimize}} \quad & w_t E_t + w_m E_m + w_s E_s \\ \text{subject to} \quad & h = \mathcal{H}\dot{l}_y \end{aligned} \tag{12}$$

where w_t, w_m and w_s are the weights for the respective terms, and are set as 1.0, 0.2 and 1.0 respectively. The optimization is then solved by solving the following linear equation:

$$\begin{pmatrix} M^\intercal M & \mathcal{H}^\intercal \\ \mathcal{H} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \dot{l}_y \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ h \end{pmatrix} \tag{13}$$

Experiment	Database Size	Our Time	[CH05] Time
Basic Movement	1400	20ms	18ms
Ladder	170	17ms	16ms
Chair (Setup 1)	331	29ms	27ms
Chair (Setup 2)	331	32ms	31ms

Table 2: *Details on the experiment conducted.*

in which

$$M = \begin{pmatrix} w_t J_{GLI} \\ w_s A \\ w_m J_{pos} \end{pmatrix} \tag{14}$$

where A is a diagonal matrix for representing $\dot{l}_y$ in Equation 9. The linear equation can be solved efficiently.

To enhance the optimization efficiency, the optimized posture will be considered as the initial posture of the next iteration. Since the algorithm is proposed for real-time applications, there are two conditions in which the optimization terminates: (1) an optimal posture is found, and (2) the number of optimization iteration exceeds a predefined value, which is set to 3 in the experiment. This allows us to trade-off synthesis quality and computation time.

7. Experimental Results

In this section, the experimental results are presented. The experiments run on a computer with an Intel Core i7-2600 Processor 3.40 GHz and 8GB of RAM. The system is implemented on Windows with Visual C++. We use UMFPACK [Dav04] as the linear solver to solve Equation 13.

For all the experiment, the value of K in the K nearest neighbour search is set to 50, the dimension of the latent space is set to 30, w_t, w_m and w_s are set as 1.0, 0.2 and 1.0 respectively.

To evaluate the effectiveness of the proposed method, we quantitatively compare the results in terms of motion quality and computational cost with traditional data-driven IK, which is a traditional data-driven IK approach. In the experiments, the purple character is synthesized by our method, and the green circles indicate the user control gesture. We duplicated the gesture to control the yellow character, which is synthesized by the method proposed in [CH05]. Body parts colored in red indicate detected collisions. The readers are referred to the attachment video for the resultant motions.

7.1. Basic Character Movements

In this experiment, we edit the posture of a single character by controlling 3 joints of the character, as shown in Figure 6. The user constrains the feet location to avoid foot sliding while dragging a body part of the character to edit the

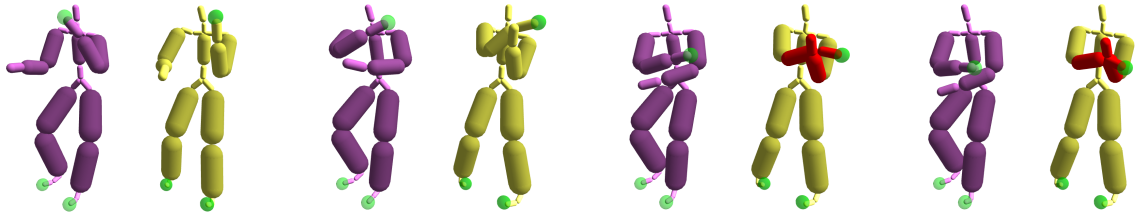

Figure 6: *Editing the posture of a single character using our method (purple) and traditional data-driven IK (yellow). Colliding joints are highlighted in red.*

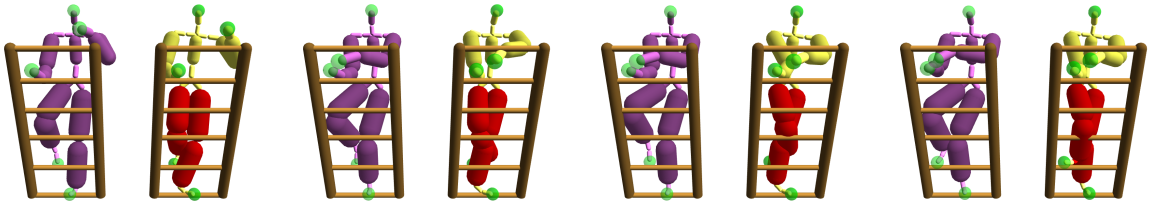

Figure 7: *Editing the posture of a character interacts with a ladder.*

posture interactively. Our method creates collision-free postures throughout the experiment. However, postures synthesized by [CH05] contain penetrations between the body segments, especially when the body parts move tightly around each other. Table 2 detailed the size of the filtered database in frames, as well as the frame time for both our method and [CH05]. The results show that the computation cost does not increase significantly with the GLI computation, but the motion quality is enhanced significantly.

7.2. Climbing a Ladder

In this experiment, the posture of a character that interacts with a ladder is edited. The user drags one of the five end-effectors to edit the posture while constraining the rest. The results are shown in Figure 7. Notice that there are significant collision between the lower body and the ladder with traditional data-driven IK.

7.3. Sitting on a Chair

In this experiment, the posture of a character who sits on a chair is edited. In the first setup, the user drags one of the body parts of the character and the rest of the body is unconstrained. The results are shown in Figure 8. We can see that the legs collide frequently with the chair. In the second setup, the user constrains the feet while dragging the upper body to see if this could improve the quality of the postures synthesized by traditional method. However, collision remains serious. This is likely because the latent space contains postures of different topologies. Thus, collision avoidance is inefficient and the optimization would easily get trapped in local optimal solution, resulting in unnatural and colliding postures.

8. Conclusion and Discussion

In this paper, we propose a new topology aware data-driven IK algorithm that can synthesize natural and penetration-free movements. Our system combines the benefits of traditional kinematics-based and topology-based motion editing. It can synthesize different variety of postures, as well as preserving the topology of the postures. We conducted experiments with complex movements that involve body parts moving tightly with each other, as well as movements that involve interaction with external objects. We show that our method outperforms traditional data-driven IK algorithm with minimal increase of computational cost.

While the framework is presented as an interactive IK system, it can be applied to real-time motion synthesis applications. In particular, the experimental results show that our method can create collision free movements in real-time. This is particular suitable for applications such as console games and VR systems to control virtual characters.

We did not take the volume and the shape of the body parts into account in the topological representation. Under extreme situations where the character has a notably different size from that of the captured motion, collision could occur. However, our system can still ensure penetration-free movement as it conserves the topology of the movement. In that case, collision can be resolved easily using motion retargetting algorithms.

If the character to be edited has a different skeletal structure from those in the motion database, such as different segment lengths and topologies, the positional constraints specified by the user may not be satisfied. This is because the final posture is synthesized by interpolating the database poses in

Figure 8: *Editing the posture of a character sitting on a chair with two set of control gestures.*

the latent space. With a different segment length, the controlled body parts may not be able to reach the target positions. The same problem would arise if the object to be interacted has a different structure from those in the database. We argue that the aforementioned issue is a motion retargeting problem. One possible solution is to synthesize the target posture with the database skeletal and object structures, and retarget the synthesized posture to the required structure using existing methods [Gle98, HKT10].

We iteratively tune the system parameters, such as the size of K in nearest neighbour search, the dimensionality of the latent space, and the number of maximum iterations in the optimization, by considering the trade-off between performance and synthesis quality. Specifically, we select a set of postures from the database randomly, and consider the positions of their end-effector as constraints. We exclude those postures from the database, and apply our algorithm to synthesize them. We tune the parameters one by one to minimize the difference between the synthesized postures and the original ones in terms of Euclidean distance of the joint positions, while maintaining real-time performance. This process is repeated until the result cannot be improved further.

Similar to every data-driven algorithm, our system may fail if the user tries to create a posture that is significantly different from every posture in the database. Figure 9 shows a failure case in which the user tries to drag the left hand. Similar posture cannot be found in the database, and hence the synthesized posture is not natural.

Since there could be multiple Euler angles representations for the same posture, interpolating postures in the Euler angles space could be problematic. However, we did not suffer from major issues with such a representation due to our system design. When constructing the latent space, we select the K topologically similar nearest neighbours from the motion database with similar Euler angles. Thus, during posture synthesis, the optimized posture based on intrinsic interpolation is always consistent. The potential drawback is that similar postures with different Euler representations will not be retrieved during the KNN search. This would reduce the number of usable postures and degrade synthesis quality. One possible solution is to implement Quaternion-based posture representations and system designs.

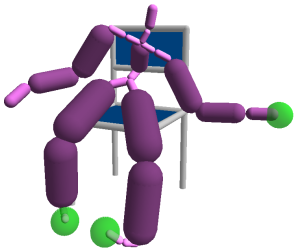

Figure 9: *A failure case when the posture to create is not similar to any posture in the database.*

As a future direction, we would like to simulate the movement where multiple characters interact with each other. The opponent characters can be considered as dynamic objects, and we can analyze the topology change during individual movement. The key problem to be solved is to model the dynamic opponents effectively, such that we can maintain a reasonable computational cost with the increase of characters.

Acknowledgement

The authors would like to thank anonymous reviewers for their constructive suggestions. This work was supported in part by Hong Kong Baptist University under the Science

Faculty Research Grant FRG1/12-13/055 and FRG2/12-13/078, and in part by Northumbria University under Academic Staff Support Grant 100700-IO114845. The basic character movement and climbing ladder motion data used in this project were obtained from mocap.cs.cmu.edu and www.animazoo.com respectively.

References

[AL09] ARISTIDOU A., LASENBY J.: *Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver*. Tech. Rep. CUEDF-INFENG, TR-632, Department of Information Engineering, University of Cambridge, September 2009. 2

[Arm88] ARMSTRONG H. G.: Anthropometry and mass distribution for human analogues. volume 1. military male aviators, 1988. 4

[Bis96] BISHOP C. M.: *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996. 6

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 686–696. 1, 3, 5, 7, 8

[Dav04] DAVIS T. A.: Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software 30*, 2 (Jun.) (2004), 196–199. 7

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics* (New York, NY, USA, 1997), ACM, pp. 139–ff. 2

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 33–42. 9

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 522–531. 1, 2

[HCKL13] HO E. S. L., CHAN J. C. P., KOMURA T., LEUNG H.: Interactive partner control in close interactions for real-time applications. *ACM Trans. Multimedia Comput. Commun. Appl. 9*, 3 (July 2013), 21:1–21:19. 3

[HK09a] HO E. S. L., KOMURA T.: Character motion synthesis by topology coordinates. *Computer Graphics Forum 28*, 2 (2009), 299–308. 3, 4

[HK09b] HO E. S. L., KOMURA T.: Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 481–492. 3, 4, 5, 6

[HK11] HO E. S. L., KOMURA T.: A finite state machine based on topology coordinates for wrestling games. *Comput. Animat. Virtual Worlds 22*, 5 (Sept. 2011), 435–443. 3

[HKT10] HO E. S. L., KOMURA T., TAI C.-L.: Spatial relationship preserving character motion adaptation. *ACM Trans. Graph. 29*, 4 (July 2010), 33:1–33:8. 3, 9

[HS13] HO E. S. L., SHUM H. P. H.: Motion adaptation for humanoid robots in constrained environments. In *ICRA '13: Proceedings of the 2013 IEEE International Conference on Robotics and Automation* (2013), IEEE. 3

[KL00] KLENIN K., LANGOWSKI J.: Computation of writhe in modeling of supercoiled dna. *Biopolymers 54*, 5 (2000), 307–17. 5

[KTWZ10] KRÜGER B., TAUTGES J., WEBER A., ZINKE A.: Fast local and global similarity searches in large motion capture databases. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2010), SCA '10, Eurographics Association, pp. 1–10. 6

[LK00] LAVALLE S., KUFFNER J.: Rapidly-exploring random trees: Progress and prospects, 2000. In Workshop on the Algorithmic Foundations of Robotics. 3

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 39–48. 2

[LWC*11] LIU H., WEI X., CHAI J., HA I., RHEE T.: Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 133–140. 3

[NH86] NAKAMURA Y., HANAFUSA H.: Inverse kinematic solutions with singularity robustness for robot manipulator control. *J. Dyn. Sys. Meas. Control 108*, 3 (Sep 1986), 163–171. 2

[Poh68] POHL W.: The self-linking number of a closed space curve. *Journal of Mathematics and Mechanics 17* (1968), 975–985. 3, 4

[SH12] SHUM H. P. H., HO E. S. L.: Real-time physical modelling of character movements with microsoft kinect. In *VRST '12: Proceedings of the 2012 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2012), ACM. 3

[SKF07] SHAPIRO A., KALLMANN M., FALOUTSOS P.: Interactive motion correction and object manipulation. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 137–144. 3

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 2 (Apr. 2001), 67–94. 2

[WC11] WEI X. K., CHAI J.: Intuitive interactive human-character posing with millions of example poses. *IEEE Computer Graphics and Applications 31* (2011), 78–88. 2

[Whi69] WHITNEY D.: Resolved motion rate control of manipulators and human prostheses. *Man-Machine Systems, IEEE Transactions on 10*, 2 (1969), 47–53. 2

[WTR11] WU X., TOURNIER M., REVERET L.: Natural character posing from a large motion database. *Computer Graphics and Applications, IEEE 31*, 3 (may-june 2011), 69 –77. 2