EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Character Motion Synthesis by Topology Coordinates

Edmond S.L. Ho[†] and Taku Komura[‡]

School of Informatics, University of Edinburgh, Scotland, UK

Abstract

In this paper, we propose a new method to efficiently synthesize character motions that involve close contacts such as wearing a T-shirt, passing the arms through the strings of a knapsack, or piggy-back carrying an injured person. We introduce the concept of topology coordinates, in which the topological relationships of the segments are embedded into the attributes. As a result, the computation for collision avoidance can be greatly reduced for complex motions that require tangling the segments of the body. Our method can be combinedly used with other prevalent frame-based optimization techniques such as inverse kinematics.

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

1. Introduction

Despite the huge amount of researches that has been done in the field of robotics and computer animation, synthesizing motions that involve close contacts is still a challenging task. Previous researches suffer from a huge amount of computation for collision avoidance as path-planning is done at the level of generalized coordinates or Cartesian coordinates, which are the lowest level of state representation.

In this paper, we efficiently plan such complex motions by introducing the idea of *topology coordinates* which takes into account the topological relationships of the segments. In topology coordinates, *writhe*, which represents how much the segments twist around each other, is the main attribute of the state space. We can easily avoid collisions of segments by simply moving the segments along the axis of writhes in topology coordinates. For the rest of this article, let us call the manifold represented by the topology coordinates as *topology space*.

Let us think of an example of interpolating the two postures shown in Figure 1 (a) and (c). The main difference of the two postures is that in (a), the left arm is crossing over the right arm, and vice-versa in (c). Although the two postures are similar at the level of generalized coordinates, the arms will penetrate through each other if they are linearly interpolated, as the blue character is doing. On the other hand, the two postures are far apart from each other in topology space. If they are linearly interpolated in topology space, we will obtain a motion in which the character twists its arms around each other, as the red character is doing.

We propose a method to interactively synthesize motions of close contacts by modeling the movements in topology space. The movements in topology coordinates can be easily mapped to / inversely mapped from generalized coordinates. Therefore, we can combine our method with keyframe animation and frame-based optimization.

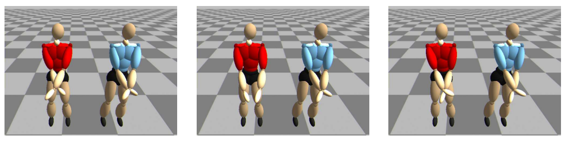

Figure 1: *The snapshots of interpolating the postures in (a) and (c) by topology coordinates (left, red character) and by generalized coordinates (right, blue character). Postures in (b) are the intermediate postures. The two arms twist around each other when they are interpolated by topology coordinates, while they penetrate through each other when they are interpolated by generalized coordinates.*

† e-mail: S.L.Ho@sms.ed.ac.uk
‡ e-mail: tkomura@ed.ac.uk

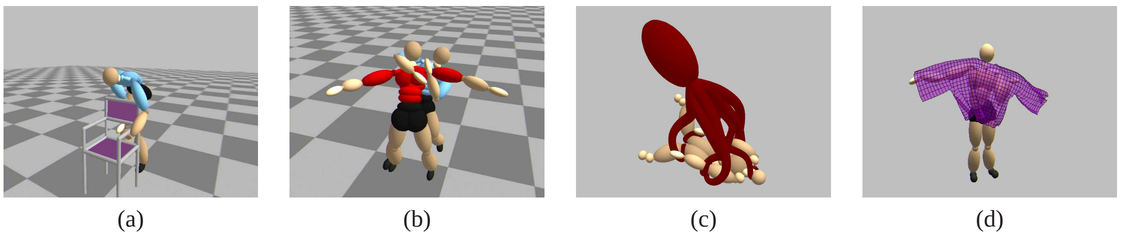

Figure 2: *Motions that involve close contacts are generated by controlling the characters in topology space. The user can interpolate keyframe postures or interactively control the topology coordinates of characters to produce animations such as (a) a character tangling its arms with a bulky furniture to hold it, (b) two characters playing wrestling by switching from one tangled posture to another, (c) an octopus tangling its limbs with a human character while avoiding its limbs getting tangled themselves and (d) a human character wearing a T-shirt.*

Our approach is most suitable for creating motions which involve close contacts between characters / deformable objects. The user can interpolate keyframe postures or interactively control the topology coordinates to produce animations such as a character tangling its arms with a bulky furniture to hold it (Figure 2 (a)), two characters playing wrestling by switching from one tangled posture to another (Figure 2 (b)), an octopus tangling its limbs with a human character while avoiding its limbs getting tangled themselves (Figure 2 (c)) and a human character wearing a T-shirt (Figure 2 (d)).

1.1. Related Work

Simulating the close interactions of multiple characters is an interest of many researchers in character animation. We first review some work in such a field. We then review global path-planning methods which are the only currently available techniques to generate motions that involve tangling.

Character interaction: The simulation of interactions between multiple characters has many applications such as computer games, virtual environments and films. Liu et al. [LHP06] simulate the close dense interactions of two characters by repetitively updating the motion of each character by spacetime constraints. Lee and Lee [LL04] simulate the boxing match by using reinforcement learning. Treuille et al [TLP07] also use reinforcement learning to simulate the pedestrians avoiding each other. Shum et al [SKY07] use min-max search to find the optimal action in a competitive environment. They also propose a real-time approach based on an automatically produced finite state machine [SKY08]. These researches do not handle very close interactions such as holding or wrestling. Ho and Komura [HK07b] generate wrestling motions by finding the topological relationship of characters from the template postures and use PD control to simulate movements where the topological relationship is kept the same. If we want to simulate a scene where the topological relationship of characters changes in time, we cannot apply such a method. A method to dynamically update the postures and the topological relationships is required. [HKar] propose to evaluate the similarity of character postures based on the topological relationships. When equivalent postures are found, the postures are linearly interpolated at the level of generalized coordinates. However, no method to interpolate topologically dissimilar postures is proposed. We propose such an approach in this paper.

Global path-planning: Up to now, if we want to generate scenes of close contacts such as one character holds, carries or wrestles with another, it is necessary to plan the motions by global path-planning / collision avoidance approaches, such as Rapidly-exploring Random Trees (RRT) [LK01] or Probabilistic Road Maps (PRM) [KSLO94]. Hirano et al [HKY05] and Berenson et al. [BDN*07] propose to use RRT to search the state space for grasping objects. As the computational cost of RRT is exponentially proportional to the degrees of freedom (DOF), the DOFs to be controlled must be reduced to the level the RRT can be expanded. Shapiro et al. [SKF07] propose a hybrid approach in which only some of the DOFs are controlled by the RRT and the rest by the captured motion data. Although the DOFs to be planned by the RRT can be reduced, the user needs to manually select which DOFs to be controlled by the RRT.

Another method to reduce the dimensionality of the control is to combine it with methods such as inverse kinematics (IK); we can limit the DOFs of the search to the movement of the end-effector and compute the internal joint angles by IK. Yamane et al. [YKH04] simulate motions to move luggage from one place to another by such a method. Ho and Komura [HK07a] plan the movements of the end effector using RRT for motions such as holding and piggybacking. The problem with such methods is that sometimes specifying the movements of the end-effector is not enough to avoid the body from getting stuck due to the collisions of the internal segments. Although such problems might be solved by collision free IK [Kal08] methods, specifying the trajectories of the end effectors is not enough for motions involving close contacts and tangling, as the internal segments may need to play active roles for the motion.

In summary, if we want to create the whole body motions that involve a lot of close contacts at interactive rates, global

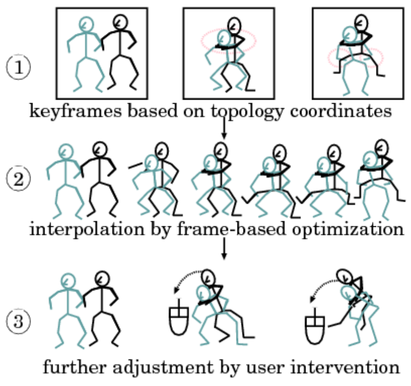
1
keyframes based on topology coordinates
2
interpolation by frame-based optimization
3
further adjustment by user intervention

Figure 3: *Overview: The keyframe postures are given based on the topology coordinates. The keyframes are interpolated by frame-based optimization. The user can further update the motions by dragging or constraining segments.*

path-planning approaches suffer from the intense computational costs. In addition to that, paths created are not consistent as the paths are randomly searched, and resulting motions require further refinements such as smoothing due to their jaggynesses.

Our Approach: Instead of planning the motions in the level of generalized coordinates, we first plan the movements in topology space. The changes in the topology coordinates are mapped to generalized coordinates through optimization based on quadratic programming, which can handle other common constraints such as kinematic or dynamical constraints. Our method can be integrated with other frame-based optimization approaches, so that it can enhance the functions of previous methods.

The outline of our method to synthesize character animation is shown in Figure 3. The details of each step are as follows: (1) The user specifies how the characters tangle their bodies during the animation. The user specifies the segments of the body to be tangled using our user interface, and produces a keyframe posture by changing the topology coordinates. The concept of topology coordinates is explained in Section 2. Our interface to edit keyframe postures is explained in Section 4.1. (2) The animation of the characters is produced by interpolating the keyframe postures in topology space. The basic idea to control multi-segment chains by changing the topology coordinates is explained in Section 3. (3) If the user is not satisfied with the animation, he/she can further add kinematic constraints or control either of the character interactively using inverse kinematics, or even replace the motion of character with captured motion data. The topological relationship between the characters can be kept by using topology coordinates as constraints. Examples of such animations are shown in Section 4.2.

1.2. Contribution

- We propose a new state space called topology space, in which the topological relationship of the body segments are embedded in the coordinate system. As a result, motion synthesis of complex interactions that requires collision avoidance can be done efficiently.
- We propose a method to map the movements in topology coordinates to those in generalized coordinates.

2. Topology Space and Coordinates

In this section, we explain the fundamental ideas of topology space in which the topological relationships of multibody segments are embedded in the coordinate system. We also present their mathematical definitions.

In topology space, we assume the segments are modelled by curves or line segments. If the character has a multibody structure, we control the hierarchical bone structure of the characters in topology space.

2.1. Topology Coordinates

Here the three attributes of topology coordinates are explained. The first attribute is the **writhe**, which counts how much the two curves are twisting around each other. Writhe can be calculated by using Gauss Linking Integral (GLI) [Poh68] by integrating along the two curves γ_1 and γ_2 as:

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

where $\times$ and $\cdot$ are cross product and dot product operators, respectively. The GLI computes the average number of crossings when viewing the tangle from all directions.

Curves can twist around each other in various ways. In order to further specify the status of the two chains, we introduce two other attributes, **center** and **density**. Examples of changing these attributes for a pair of strands are shown in Figure 4. The center, which is composed of two scalar parameters, explains the center location of the twisted area. The density, which is a single scalar parameter, explains how much the twisted area is concentrated at one location along the strands. When the density is zero, the twist is spread out all over the two strands. When the density value is either very large or very small, we can say one strand is playing a major role to compose the twist, as it is twisting around the other strand which is kept relatively straight (Figure 4). When the density turns from negative to positive, or vice-versa, the strand playing the major role switches.

2.2. Mathematical Definition

We represent the bone structure of characters by a set of line segments. Therefore, now we will mathematically define the topology coordinates of serial chains. Let us assume we have two chains S_1 and S_2, each composed of n_1 and

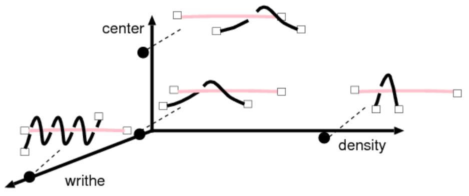

Figure 4: *The three axes in topology space : writhe, center and density. The center, which specifies the central location of the twist, is composed of two scalar parameters although it is represented by a single axis in this figure. The density tells which strand plays the major role to compose the twist.*

n_2 line segments, connected by revolute, universal or gimbal joints (Figure 5). In this case, we can compute the total writhe by summing the writhes by each pair of segments:

$$w = GLI(S_1, S_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} T_{i,j} \quad (2)$$

where w represents the writhe, $T_{i,j}$ is the writhe between segment i on $S1$ and j on $S2$. An analytical solution exists for computing $T_{i,j}$ [KL00]. The details are explained in Appendix 1. Let us define a $n_1 \times n_2$ matrix $\mathbf{T}$ whose (i,j)-th element is $T_{i,j}$, and call this the **writhe matrix**. The writhe matrix explains how much each pair of segments from S_1 and S_2 contributes to the total writhe value. Various twists and the corresponding writhe matrix are shown in Figure 6.

We first define a 2D vector $\mathbf{c}$ as the center calculated by the following equation:

$$\mathbf{c} = (x_g, y_g) = \left(\frac{\sum_i^{n_1} \sum_j^{n_2} i \cdot T_{i,j}}{w} - \frac{n_2}{2}, \frac{\sum_i^{n_1} \sum_j^{n_2} j \cdot T_{i,j}}{w} - \frac{n_1}{2}\right) \quad (3)$$

This explains the center of the twisted chains in topology space. It tells the overall location of the twisted area.

In Figure 6, it can be observed that the elements of large absolute values concentrate along a particular axis across the matrix. This axis tends to be vertical when S1 twists around S2 and horizontal when S2 twists around S1. When both chains twist around each other, the axis lies along the diagonal line. This observation leads us to the definition of the density to be the orientation of the principal axis. As different pair of chains result in different sizes of writhe matrices, we first normalize the data by scaling the writhe matrix to a square area, and then compute the principal axis in this area. We compute the orientation of this axis against the diagonal line and define this as the density (Figure 7).

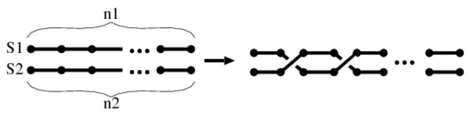

Figure 5: *Twisting a chain of segments around each other*

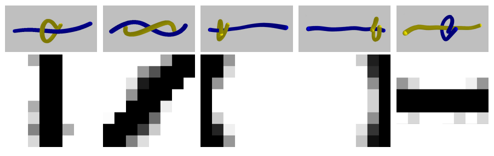

Figure 6: *(upper) Tangles with different density and center, and (lower) the distribution of elements with large absolute values in the corresponding writhe matrix. The darkness represents the amplitude of the absolute value.*

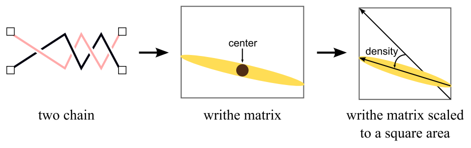

Figure 7: *A twisted chain (left) and the writhe matrix (middle). The center is computed by calculating the center of mass of the writhe matrix's elements. The writhe matrix is scaled to a square area (right) to compute the principal axis. The angle made between the principal axis and the diagonal line is defined as the density.*

We have explained how to compute the topology coordinates from the kinematics of the chains in this section. We can also solve the inverse problem; computing the kinematics from the topology coordinates. The methodology of such a manipulation, which is useful for generating motions that involve twisting, is explained in the following section.

3. Manipulation in Topology Space

In this section, we first explain how to synthesize the motions of a pair of serial chains, composed of n_1 and n_2 segments, by changing their topology coordinates. The topology coordinate of the chains are gradually updated and their movements are mapped to the generalized coordinates (Figure 8). This process is repeated until the topology coordinates reach their target values. At every step, suppose we want to change the topology coordinates from $\mathbf{G} = (w, d, \mathbf{c})$ to $\mathbf{G} + \Delta\mathbf{G} = (w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$. Using the updated topology coordinates, a corresponding writhe matrix is computed (step 1 in Figure 8). This process is explained in Section 3.1. Then the kinematics of the serial chains are adjusted so that their writhe matrix become similar to the one computed from the topology coordinates (step 2 in Figure 8). This process is explained in Section 3.2.

Next, we explain how this methodology can be used for two other manipulations: one is to pass a chain through a loop, and the other is to tangle a chain with a bundle of chains. These methods are explained in Section 3.3.

3.1. Desired Writhe Matrix

Here we explain a method to compute a writhe matrix that corresponds to a given topology coordinate $G = (w^d, d^d, \mathbf{c^d})$. This is done by first preparing a matrix which has all the values concentrated at one or two columns of the matrix, and then rotating and translating this part inside the matrix.

We start from a $n_1 \times n_2$ matrix $\mathbf{I}$, who has values evenly distributed at the $(n_2+1)/2$-th column if n_2 is odd, or at both the $n_2/2$ and $n_2/2+1$-th column if it is even:

$$\mathbf{I} = \begin{pmatrix} 0\cdots,\frac{1}{n_1},\cdots,0 \\ \vdots \\ 0\cdots,\frac{1}{n_1},\cdots,0 \end{pmatrix}, \qquad \begin{pmatrix} 0\cdots,\frac{1}{2n_1},\frac{1}{2n_1},\cdots,0 \\ \vdots \\ 0\cdots,\frac{1}{2n_1},\frac{1}{2n_1},\cdots,0 \end{pmatrix}$$
$$(n_2 \text{ is odd}) \qquad\qquad (n_2 \text{ is even})$$

Note that for this writhe matrix, the corresponding topology coordinates are $w = 1, d = -\frac{\pi}{4}, \mathbf{c} = (0,0)$. In order to obtain a writhe matrix for arbitrary topology coordinates, we rotate and translate the elements of $\mathbf{I}$.

Changing the density by rotation: As the density was defined as the orientation of the main axis of the writhe matrix, rotating $\mathbf{I}$ around the center results in changing the density. We first map the elements of the writhe matrix to a circle, rotate the values around the center until the orientation reaches the target density value d^d, and finally inverse-map the values onto the matrix (Figure 9). The mapping from the matrix to the circle is done by first mapping the matrix into a square area, and then into a circle area. The inverse mapping is done vice-versa. This operation is defined as $\mathbf{M'} = R(\mathbf{I}, d^d)$, where $\mathbf{M'}$ is the output matrix. The details of this operation are explained in Appendix 2. This operation of rotation results in changing the density of the chain-pairs.

Changing the center by translation: For changing the center, we simply translate all the elements according to the difference of the target and current location of the center (Figure 10). As elements with values might be shifted out from the matrix, the translation applied will not bring the center to the target location. The matrix is recursively translated until

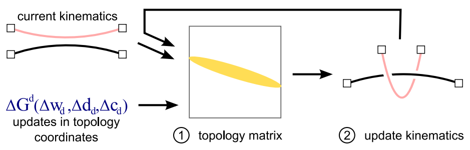
current kinematics
$\Delta G^d(\Delta w_d, \Delta d_d, \Delta c_d)$
updates in topology coordinates
① topology matrix
② update kinematics

Figure 8: *The overview of updating the kinematics of two serial chains by changing their topology coordinates. The increment/decrement of the topology coordinate is given at each step. The writhe matrix that corresponds to the updated topology coordinate is computed (step 1), and then the kinematics of the the chains are updated so that their writhe matrix become similar to the target one (step 2). This process is repeated until the target topology coordinate is reached.*

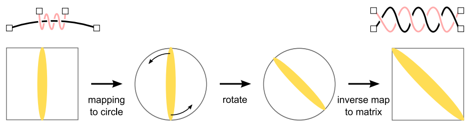

Figure 9: *The elements of the writhe matrix are first mapped to a square area, and then to a circle. The circle is rotated until the axis reaches the desired density value. Finally, the axis is mapped back to the writhe matrix.*

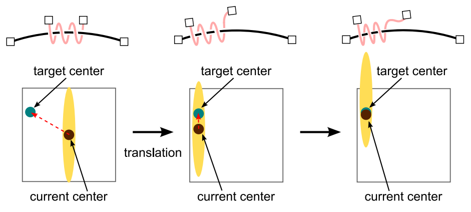

Figure 10: *The elements of the writhe matrix are translated according to the difference of the target and current center location. As elements with values might be shifted out from the matrix, the translation applied will not bring the center to the target location. The process is repeated until the current center comes close enough to the target location.*

the error is smaller than a predefined threshold. As the sum of the elements can be smaller than one after some elements are shifted out from the matrix, the values are normalized at the end. This operation is defined here as $Tr(\mathbf{M}, \mathbf{c^d})$, where $\mathbf{c^d}$ is the target center location and $\mathbf{M}$ is the input matrix.

Calculating the desired writhe matrix: Up to here, the writhe value has been kept to 1. Finally, we multiply w to the whole matrix so that it reaches the desired writhe value. Let us define this operation by $S(\mathbf{M}, w^d)$. We sequentially apply $R()$, $Tr()$ and $S()$ to $\mathbf{I}$ to obtain the writhe matrix of the target topology coordinate. It is to be noted that when these operators are applied to the writhe matrix in this order, each operation will only update either the density, center and or the writhe, respectively, without affecting the other attributes. The writhe matrix $\mathbf{T}$ computed by the following operation represents topology coordinate $(w^d, d^d, \mathbf{c^d})$:

$$\mathbf{T^d} = S(Tr(R(\mathbf{I}, d^d - \frac{\pi}{4}), \mathbf{c^d}), w^d). \tag{4}$$

3.2. Mapping Topology Coordinates to Generalized Coordinates

Here we explain how to update the generalized coordinates according to the updates in of the topology coordinates. Assuming the current generalized coordinates of the two chains are $(\mathbf{q_1}, \mathbf{q_2})$, and the topology coordinates of the two chains are to be updated from $\mathbf{G} = (w, d, \mathbf{c})$ to $\mathbf{G} + \Delta\mathbf{G} = (w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$, the task here is to compute the updates of the generalized coordinates $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$.

We first compute $\mathbf{T^d}$, the desired writhe matrix at $\mathbf{G}+\Delta\mathbf{G}$, using the method explained in Section 3.1, and then update the kinematics of the chains so that the resulting writhe matrix become similar to the desired writhe matrix. Let us assume the writhe matrix of the current configuration is $\mathbf{T}$, and its update after changing the configuration is $\Delta\mathbf{T}$.

$$\mathbf{T}+\Delta\mathbf{T}-\mathbf{T^d}=\mathbf{0} \tag{5}$$

Here we assume the updates are small enough so that the relationships of $\Delta\mathbf{T}$, and $(\Delta\mathbf{q_1},\Delta\mathbf{q_2})$ are linear:

$$\Delta\mathbf{T}=\frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1}+\frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\mathbf{q_2}. \tag{6}$$

When we control the two chains, we need to make sure that the two chains do not penetrate through each other. When two line segments approach to each other, the writhe value increases. At the moment before they cross each other, the absolute value of the writhe approaches to 0.5. Therefore, penetrations can be avoided by limiting the maximum writhe value between arbitrary segments:

$$|T_{i,j}+\Delta T_{i,j}|\leq\sigma \quad (1\leq i\leq n_1, 1\leq j\leq n_2) \tag{7}$$

where $\Delta T_{i,j}$ is the (i,j)th element of $\Delta\mathbf{T}$, σ is a threshold, that is set to 0.2 in our experiments to avoid the segments to approach too close to each other.

We can also add any kinematical constraints which can be linearized with respect to the generalized coordinates when the movement is small, such as the movements of any parts of the body in Cartesian coordinates or the center of mass. Let us summarize such constraints as follows:

$$\mathbf{r}=\mathbf{J_1}\Delta\mathbf{q_1}+\mathbf{J_2}\Delta\mathbf{q_2}, \tag{8}$$

where $\mathbf{J_1},\mathbf{J_2}$ are the Jacobians of this constraint, and $\mathbf{r}$ is the linearized output of this constraint.

The update of the configurations of the chains can be computed by minimizing an objective function that represents the norm of the of movements subject to constraints (6)-(8):

$$\min_{\Delta\mathbf{q_1},\Delta\mathbf{q_2},\delta}||\Delta\mathbf{q_1}||^2+||\Delta\mathbf{q_2}||^2+||\delta||^2 \text{ s.t.} \begin{cases}\Delta\mathbf{T}=\frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1}+\frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\mathbf{q_2}\\ |T_{i,j}+\Delta T_{i,j}|\leq\sigma\\ (1\leq i\leq n_1,1\leq j\leq n_2)\\ \mathbf{T}+\Delta\mathbf{T}-\mathbf{T^d}+\delta=\mathbf{0}\\ \mathbf{r}=\mathbf{J_1}\Delta\mathbf{q_1}+\mathbf{J_2}\Delta\mathbf{q_2}\end{cases} \tag{9}$$

where δ is a vector of parameters introduced to convert Equation (5) into soft constraints. The updated generalized coordinates $(\mathbf{q_1}+\Delta\mathbf{q_1},\mathbf{q_2}+\Delta\mathbf{q_2})$ correspond to the target topology coordinate $(w+\Delta w, d+\Delta d, \mathbf{c}+\Delta\mathbf{c})$. By repetitively solving Equation (9), we can update the configurations by specifying the trajectories of the topology coordinates.

3.3. Additional Manipulations

Here we first introduce a technique to pass a serial chain through a loop (Figure 11 (a)), which is useful for simulating movements such as moving the arms through sleeves or passing a hand through a hole. Next we introduce a technique to tangle a serial chain with a bundle of serial chains (Figure 11 (b)). Such a motion happens when holding the arms and torso of another character at the same time.

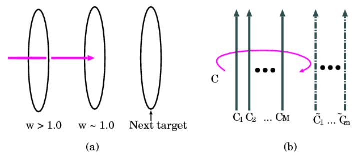

Figure 11: *(a) Passing a chain through loops. The writhe is around 1 when the chain passes through the loop. The chain can be passed through multiple loops by sequentially switching the target loop. (b) Tangling a chain with a bundle of chains while avoiding to get tangled with others.*

Passing a chain through loops: When controlling the serial chain to pass through a loop, we will only use the writhe of the topology coordinates as we do not care near which part of the loop that the serial chain is passing through. The writhe can be computed by Equation (2). The writhe approaches one when the chain passes through the loop, as shown in Figure 11(a), (Recall the Gauss Integral computes the average number of crossings when viewing from all directions). Therefore, we control the serial chain by gradually increasing the writhe value between the loop and the chain and updating the kinematics of the serial chain accordingly:

$$\min_{\Delta\mathbf{q},\delta}||\Delta\mathbf{q}||^2+\delta^2 \;\text{ subject to }\; w+\frac{\partial w}{\partial\mathbf{q}}\Delta\mathbf{q}+\delta=w_d \tag{10}$$

where $\mathbf{q},\Delta\mathbf{q}$ are the generalized coordinates of the serial chain and their updates, δ is a variable to convert the constraint into a soft constraint, w is the current writhe value between the chain and the loop and w_d is the target writhe value to reach in this step. When passing the chain through multiple loops, we switch the target loop from one to another. In our experiments, when the writhe of the chain with the current target loop has exceeded 0.95, we switched the target loop to the next one (Figure 11(a)).

Tangling a chain with a bundle of chains: Here we explain how to tangle a chain C with a bundle of chains $C_1,\ldots,C_M$, while avoiding getting tangled with another set of chains $\bar{C}_1,\ldots,\bar{C}_m$ (Figure 11(b)). The main idea is to specify the average of the writhes between C and $C_1,\ldots,C_M$, while keeping their variance small. The following problem is repetitively solved until the chain twists around a bundle of chains:

$$\min_{\Delta\mathbf{q}}||\Delta\mathbf{q}||^2+\sum_{k=1}^{M}\left(\frac{w+\Delta w}{M}-(w_k+\Delta w_k)\right)^2$$

$$\text{subject to}\begin{cases}\Delta w=\Delta w_1+\ldots+\Delta w_M\\ \Delta w_k=\frac{\partial w_k}{\partial\mathbf{q}}\Delta\mathbf{q}(1\leq k\leq M)\\ T_{i,k_j}+\Delta T_{i,k_j}\leq\sigma(1\leq k_j\leq n_k)\\ 0.5\geq\bar{w}_l+\Delta\bar{w}_l\\ \Delta\bar{w}_l=\frac{\partial\bar{w}_l}{\partial\mathbf{q}}\Delta\mathbf{q}(1\leq l\leq m)\\ T_{i,l_j}+\Delta T_{i,l_j}\leq\sigma(1\leq k_l\leq n_l)\end{cases} \tag{11}$$

where $\Delta\mathbf{q}$ is the increment of C's generalized coordinates, n_k, n_l are the number of segments composing C_k and $\overline{C_l}$, w_k $(1 \le k \le M)$ and $w_l(1 \le l \le m)$ are writhes between C and $(C_k, \overline{C_l})$, respectively. The third and sixth constraints keep the line segments from penetrating through each other. The fourth constraint prevents C from getting tangled with $\overline{C_j}$ by keeping their writhes below 0.5.

Here we again only specified the writhe of the topology coordinate as the chain may not have enough degrees of freedom for the user to specify the center and density, as there are many other chains to get tangled with. However, it is possible to combine this problem with that of Equation (9) by specifying the topology coordinates of the controlled chain and one of the serial chains in the bundle.

4. Experiments

In this section, we show some example motions produced by our system. We first explain about our user interface to produce keyframe postures for character animation. Then, we show the resultant animations produced by interpolating such keyframe postures.

4.1. Controlling Characters

We provide an interface for the user to interactively edit the postures of one or two characters. The user can specify the topological constraints as well as ordinary kinematic constraints, such as the trajectories of the end effectors. Once the keyframe postures are generated, their topology coordinates are available, and we can interpolate the postures in topology space.

Because the techniques explained in Section 3 are only for open serial chains or loops, we need to specify the paths between the character's joints / end effectors which we want to tangle. For instance, suppose we want to generate a posture of a wrestling attack called Full Nelson Hold (Figure 12, right), in which the attacker needs to first pass the arms under those of the other character from behind, and then let the attacker press the other's neck. In this case, the paths "left shoulder - left hand" and "right shoulder - right hand" must be twisted with the paths "head-top - left hand" and "head-top - right hand" of the other character, respectively. Once the paths to be tangled are specified, the user can use the scrollbars to adjust the writhes, density and center of the paths. The values set by the scrollbars are used to compute the desired writhe matrix $\mathbf{T^d}$ in Equation (9).

If there are two characters, we constrain the posture of one character by default, as we found this is easier for the user to generate the desired postures. The user can alternately switch the active character until the postures are satisfactory. The user can also adjust the postures using inverse kinematics. In this case, the posture of the other character is updated so that the topology coordinates are kept the same.

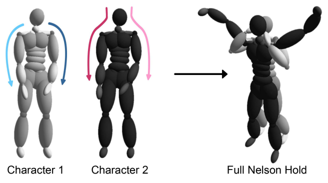

Figure 12: *The user specifies the area of the body to be tangled. The two paths (left shoulder - left hand), (right shoulder - right hand) of Character 1 are to be twisted with (head-tip - left hand), (head-tip - right hand) of Character 2. The posture on the right is the expected final posture.*

The postures created can be used as a keyframe for motion synthesis. We interpolate the keyframes in topology space, and therefore, we can easily twist the segments around each other without using complex path-planning or collision avoidance schemes. The interpolated motions can be further adjusted by the user using inverse kinematics: the user can drag a segment of one of the bodies by the mouse. The postures of the two bodies will be updated so that the kinematic constraints and topology constraints are both satisfied.

4.2. Motion Synthesis

Here, experimental results of controlling characters by the topology coordinates are shown. First, motions of a single / multiple human characters that require close contacts between segments were created. Next, examples of a human character interacting with rigid / deformable objects were created. Finally, animations of an octopus interacting with swimming fishes or a human character were created.

Human character animation : First, a motion of a single character was generated by interpolating keyframe postures shown in Figure 13(a). In these postures, the body of the character is self-tangled. Such postures cannot be interpolated by the generalized coordinates without self-collisions. We can easily produce natural-looking behaviors by interpolating them in topology space.

Next, wrestling movements of two human characters were simulated by interpolating five keyframe postures in Figure 13(b). For the motion of the red character in the front, we imported motion capture data of a single character moving around. The topology coordinates of the postures were linearly interpolated. The attacking character behind had to dynamically adjust its movements so that the topology coordinates between the limbs become the same as the target values. This example shows that our method can be combinedly used with captured motion data.

Finally, a piggyback motion that requires multiple tangles by the limbs was created as shown in Figure 13(c). Four keyframe postures were used to create this motion. Although this motion requires a significant amount of collision

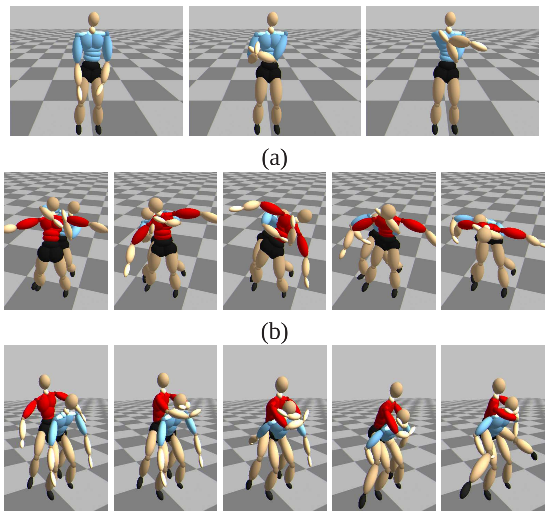

Figure 13: *(a) Three keyframe postures to generate a stretching motion. (b) The wrestling motion in which the red character re-holds the blue character in various ways. (c) A piggyback motion created from four keyframes.*

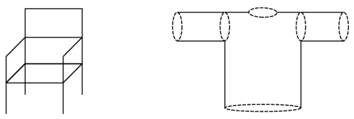

Figure 14: *The topology of the chair model composed of eighteen pipes (left) and the shirt model with six rings (right) used for creating the animations.*

avoidance, we can produce the motion in a very short time. The computational cost for creating this motion is compared with that when using a global path-planning method based on RRT is shown at the end of this section.

Interaction with rigid/deformable objects: First, we created an animation of a character re-holding a chair in various ways. The topological structure of the chair is shown in Figure 14 (left). The five keyframes shown in Figure 15 are used to produce the animation. The keyframes are created by changing the topology coordinates of the character's arm with the pipes of the chair. If these postures are interpolated by generalized coordinates, the body can easily penetrate through the pipes of the chair. When we interpolate the postures in topology space, the arms can avoid the pipes and hold the chair. Next, an example of a character holding a deformable object composed by a bundle of line segments is generated (Figure 16(left)). The writhe values between the path formed by the two arms and the bundle of lines of the object were increased to create an animation of holding the object. The method explained in Section 3.3 was used to simulate such a motion. The character can stably hug the object

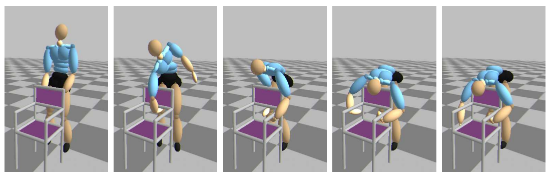

Figure 15: *The five keyframes to produce the animation of re-holding the chair.*

although the shape of the object is dynamically deformed. The character adjusts its posture so that its topological relationship with the object does not change. This example shows our method is extensible to control the topological relationships of 3D shapes.

Finally, an example of a human character wearing a shirt is made. Six virtual rings were prepared inside and at the fringe of the shirt (Figure 14 (right)). The character was guided to pass its arms and necks through the rings by the topological constraints. A snapshot is shown in Figure 2(d). This example shows that our method is also useful for creating animations of characters interacting not only with each other, but also with deformable models. It also shows we can handle topological relationships between the serial bodies and rings.

Octopus Motions: We have prepared an octopus model whose legs are modeled by serial chains of rigid segments. Each leg is composed of twelve segments. Firstly, we show an example in which the octopus catches a number of fishes using different limbs simultaneously (Figure 16,right). The difficulty of generating such motions is that if we do not take into account the tangle constraints, the limbs can get tangled with each other. In our system, this is avoided by adding extra constraints to prevent the limbs to generate tangles whose writhe is larger than 0.5. As a result, the limbs will move away from each other when their distance gets closer. We have also generated an example of tangling the octopus with a human character (Figure 2(c)). The motions of the octopus were created from only two keyframe postures, one each for the initial and final posture.

In summary, although these animations can be produced by existing techniques, they will require careful tuning to avoid artifacts. For example, if they are made by traditional keyframe animation methods, a great number of keyframes need to be inserted to guide the character correctly and avoid penetrations of segments. The readers are referred to the supplementary video for further details.

4.3. Computational Costs

In our demos, we used a human character model of 42 DOF, and an octopus model of 276 DOF. All DOFs are used when controlling the characters. For the human character model,

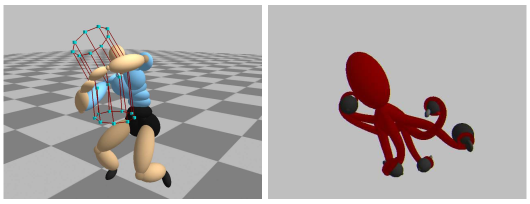

Figure 16: *A character hugging an object composed of a bundle of line segments (left), and an octopus catching multiple fish simultaneously using its limbs (right)*

when one character is controlled, we can obtain an interactive rate of 40 frames per second when using a Pentium IV 3.2 GHz PC. When two characters are simultaneously controlled, we can still obtain a frame rate of 10 frames per second. We use ILOG CPLEX 9.1 [cpl] as our quadratic programming solver.

The bottleneck of the computation is at that of the writhe matrix. When we want to tangle a body of n_1 segments with that of n_2 segments while avoiding it getting tangled with another body of n_3 segments, the cost for computing the writhe matrices becomes $O(n_1 \times n_2 + n_1 \times n_3)$. For tangle avoidance, we can omit computing the matrix elements for those which are far away from each other. Therefore, the complexity becomes $O(n_1 \times n_2)$. The analytical solver of the GLI enables fast computation of the writhe matrix. As the writhe matrix is sparse, once it is computed, the quadratic programming solver can efficiently compute the generalized coordinates. Therefore, we can achieve real-time performance even when the DOFs of the bodies are large.

For the scene of one character piggybacking another, when a global method by Ho and Komura [HK07a] is used, it takes more than 1000 seconds to obtain the results. On the other hand, by using our method, we can obtain the results in less than 2 seconds.

5. Summary and Discussions

In this paper, we have proposed a new method to control characters by using topological constraints. We specify how the segments of the characters should twist around each other over time. Various basic motions such as hugging, wrestling attacks, holding a bulky object, which were difficult to be handled before can be generated. The ability to create complex motions of a character which has a large DOF such as an octopus while avoiding the penetrations from only a few number of keyframes is a great advantage of our method. Existing global path-planning techniques based on RRT will suffer when DOF of the character is large.

There are some shortcomings with our method: First of all, as our method is based on local optimization, in some cases the solver can get caught in local minimum. This happens mostly when there are too many topological constraints to be solved simultaneously. In fact, some topological constraints cannot be physically satisfied at the same time. In such a case, the user needs to decrease the number of topological constraints by limiting the number of chains to get tangled with or restrained from being tangled with. This can be done interactively by the user.

Secondly, in this paper we assume all the models are only composed of line segments. Many objects have area and volume, and sometimes it is difficult to approximate the topological relationships by those objects by line segments. We also did not handle the collisions between the rigid objects. The penetration of line segments can be examined by tracing the writhe value. However, another collision detection scheme is required for rigid bodies. Although we can insert a collision detection stage after the postural updates and add repulsive forces to the segments so that they do not penetrate each other, sometimes the shape of the rigid bodies can inhibit the bodies from getting tangled / untangled. One solution is to model each rigid segment by a mesh of line segments. We can use a multi-resolution approach and represent the segments by line segments at low resolution and by a mesh structure in high resolution. When precise collision avoidance based on the detailed shape is required, we can calculate the motions using the high resolution model.

Finally, the user needs to specify the order to tangle the segments to those of the other. When there are many tangles between the characters, there are many ways to arrive to the goal postures. The appropriate way to evaluate such routes can be application dependent, and usually the user prefers to specify by him/herself. Therefore, in this research, we let the user specify the sequence by him/herself by providing the list of sequence. Planning the sequence of tangles and synthesizing a sequence of motions can be one direction for future research.

Acknowledgement

This project was partly supported by a CERG grant from the Research Grant Council of Hong Kong (Ref:CityU1149/05) and the Wolfson Microelectronics Scholarship.

References

[BDN*07] Berenso D., Diankovage R., Nishiwaki K., Kagami S., Kuffner J.: Grasp planning in complex scenes. *IEEE/RAS Humanoids* (2007).

[cpl] *ILOG Inc,CPLEX http://www.ilog.com.*

[HK07a] Ho E. S. L., Komura T.: Planning tangling motions for humanoids. In *Proc of Humanoids 2007* (2007).

[HK07b] Ho E. S. L., Komura T.: Wrestle alone: Creating tangled motions of multiple avatars from individually captured motions. In *Proc of Pacific Graphics* (2007), pp. 427–430.

[HKar] Ho E. S. L., Komura T.: Indexing and retrieving motions of characters in close contact. *IEEE Trans on Visualization and Computer Graphics* (to appear).

[HKY05] HIRANO Y., KITAHAMA K., YOSHIZAWA S.: Image-based object recognition and dexterous hand/arm motion planning u sing rrts for grasping in cluttered scene. *IEEE/RSJ Int Conf on Intelligent Robots and Systems* (2005).

[Kal08] KALLMANN M.: Analytical inverse kinematics with body posture control. *Computer Animation and Virtual Worlds 19*, 2 (2008), 79–91.

[KL00] KLENIN K., LANGOWSKI J.: Computation of writhe in modeling of supercoiled dna. *Biopolymers 54* (2000), 307–317.

[KSLO94] KAVRAKI L., SVESTKA P., LATOMBE J., OVERMARS M.: *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*. Tech. rep., Stanford, CA, USA, 1994.

[LHP06] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Composition of complex optimal multi-character motions. *ACM SIGGRAPH / Eurographics Symp on Computer Animation* (2006), 215–222.

[LK01] LAVALLE S., KUFFNER J.: Rapidly-exploring random trees: Progress and prospects. *Robotics: The Algorithmic Perspective. 4th Int'l Workshop on the Algorithmic Foundations of Robotics* (2001).

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. *Proc of 2004 ACM SIGGRAPH/Eurographics Symp on Computer Animation* (2004), 79–87.

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

[SKF07] SHAPIRO A., KALLMANN M., FALOUTSOS P.: Interactive motion correction and object manipulation. *Proc of ACM SIGGRAPH Symp on Interactive 3D graphics and Games (I3D)* (2007).

[SKY07] SHUM H. P. H., KOMURA T., YAMAZAKI S.: Simulating competitive interactions using singly captured motions. *Proc of ACM Virtual Reality Software Technology 2007* (2007), 65–72.

[SKY08] SHUM H. P. H., KOMURA T., YAMAZAKI S.: Simulating interactions of avatars in high dimensional state space. *ACM SIGGRAPH Symp on Interactive 3D Graphics (i3D)* (2008), 131–138.

[TLP07] TREUILLE A., LEE Y., POPOVIC' Z.: Near-optimal character animation with continuous control. *ACM Trans on Graphics 26*, 3 (2007), 7:1–7:7.

[YKH04] YAMANE K., KUFFNER J., HODGINS J.: Synthesizing animations of human manipulatioin tasks. *ACM Trans on Graphics 23*, 3 (2004), 532–539.

Appendix 1: Analitically Calculating Writhes of Lines

Klenin and Langowski [KL00] proposes an analytical solution for the Gauss Linking Integrals of two line segments. Assume we have two line segments i and j and their writhe is $T_{i,j}$. Suppose points a,b and c,d are the end points of i and j, respectively. The vectors connecting a-b,a-c,a-d,b-c,b-d,c-d are defined by $r_{ab}, r_{ac}, r_{ad}, r_{bc}, r_{bd}, r_{cd}$, respectively (Figure 17). Using these vectors, the normal vectors of the tetrahedron made by these four points can be calculated by:

$$n_a = \frac{r_{ac}\times r_{ad}}{\|r_{ac}\times r_{ad}\|}, n_b = \frac{r_{ad}\times r_{bd}}{\|r_{ad}\times r_{bd}\|}, n_c = \frac{r_{bd}\times r_{bc}}{\|r_{bd}\times r_{bc}\|}, n_d = \frac{r_{bc}\times r_{ac}}{\|r_{bc}\times r_{ac}\|}.$$

Finally, $T_{i,j}$ is calculated by

$$T_{i,j} = \arcsin(n_a n_b) + \arcsin(n_b n_c) + \arcsin(n_c n_d) + \arcsin(n_d n_a).$$

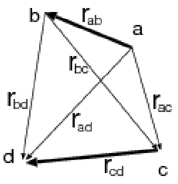

Figure 17: *The tetrahedron composed by $\vec{ab}$ and $\vec{cd}$.*

Appendix 2: Changing the Density of a Writhe Matrix

Here we explain how to change the density of arbitrary writhe matrix. Let us define this transformation of rotation by $R(\mathbf{M},\phi)$, where M is the input writhe matrix, and ϕ is the amount of rotation. Firstly, the elements of the input matrix M are mapped to an area within a square which is centered at the origin of a 2D Cartesian coordinate, and whose edges are over $x = \pm 1$ and $y = \pm 1$. This is done by

$$v_i = \frac{i - \frac{n_1+1}{2}}{\frac{n_1-1}{2}}, v_j = \frac{j - \frac{n_2+1}{2}}{\frac{n_2-1}{2}}.$$

Secondly, (v_i, v_j) are mapped into an area inside a circle whose radius is 1 by the following equation:

$$(v'_i, v'_j) = \frac{\sqrt{2}}{\cos(\theta_0)}$$

where $\theta_0 = \arctan\frac{v_i}{v_j}$. Thirdly, (v''_i, v''_j) is rotated around the origin for the increment of density ϕ.

$$(v''_i, v''_j) = R(\phi)(v'_i, v'_j).$$

Finally, (v''_i, v''_j) is inverse-mapped back into the square area, and the indexes in the new writhe matrix is computed:

$$\begin{aligned}
(v'''_i, v'''_j) &= s_{-1}(v''_i, v''_j) \text{ where} \\
s_{-1} &= \begin{cases} \frac{\sin(\phi-\frac{\pi}{4})}{\sqrt{2}} & (-\frac{\pi}{2} \le \phi \le 0, \frac{\pi}{2} \le \phi \le \pi) \\ \frac{\cos(\phi-\frac{\pi}{4})}{\sqrt{2}} & (\pi \le \phi \le \frac{5}{4}\pi, 0 \le \phi \le \frac{\pi}{2}). \end{cases} \\
i' &= \frac{n+1}{2} + \frac{n-1}{2} v'''_i, j' = \frac{m+1}{2} + \frac{m-1}{2} v'''_j,
\end{aligned}$$

where (v'''_i, v'''_j) is the vertex in the square area and (i', j') are the indices where the elements of $\mathbf{M}$ at indices (i, j) are mapped to. As (i, j) went through transformations of scaling and rotation, (i', j') are no longer integers. Therefore, at the last stage, we will compute the elements of the new writhe matrix $\mathbf{M}'$ by using the weighted sum of entries at (i', j'):

$$\mathbf{M}'(i'', j'') = \frac{\sum_{k=1}^{4} D(i'' - i'_k, j'' - j'_k)\mathbf{M}(i, j)}{\sum_{k=1}^{4} D(i'' - i'_k, j'' - j'_k)}$$

where (i'', j'') are the new integer indices of matrix $\mathbf{M}'$, $(i'_k, j'_k)(k = 1, \ldots, 4)$ are the transformed indices (i', j') that surrounds (i'', j'') and $D()$ represents the Euclidean distance operator.