Spatial Relationship Preserving Character Motion Adaptation

Edmond S.L. Ho[*] Taku Komura[†]
The University of Edinburgh

Chiew-Lan Tai[‡]
Hong Kong University of Science and Technology

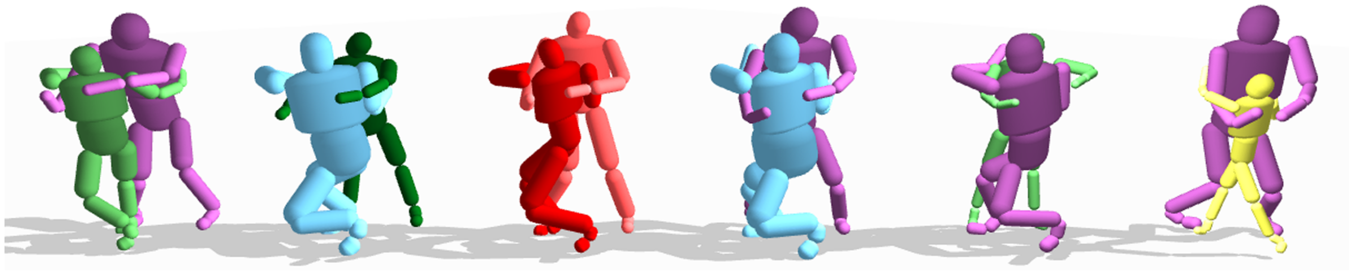

Figure 1: *Our system can retarget motions of close interactions to characters of different morphologies. A judo interaction (red / orange pair) retargeted to characters of different sizes.*

Abstract

This paper presents a new method for editing and retargeting motions that involve close interactions between body parts of single or multiple articulated characters, such as dancing, wrestling, and sword fighting, or between characters and a restricted environment, such as getting into a car. In such motions, the implicit spatial relationships between body parts/objects are important for capturing the scene semantics. We introduce a simple structure called an interaction mesh to represent such spatial relationships. By minimizing the local deformation of the interaction meshes of animation frames, such relationships are preserved during motion editing while reducing the number of inappropriate interpenetrations. The interaction mesh representation is general and applicable to various kinds of close interactions. It also works well for interactions involving contacts and tangles as well as those without any contacts. The method is computationally efficient, allowing real-time character control. We demonstrate its effectiveness and versatility in synthesizing a wide variety of motions with close interactions.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations;

Keywords: Character Animation, Close Interaction, Spatial Relationship, Motion Editing, Motion Retargeting

[*]e-mail: edmond@edho.net
[†]e-mail: tkomura@ed.ac.uk
[‡]e-mail: taicl@cse.ust.hk

1 Introduction

Close interactions, not necessarily with any contacts, between different body parts of single or multiple characters or with the environment are common in computer animation and 3D computer games. Yoga, wrestling, dancing and moving through a constrained environment are some examples. In such motions, the spatial relationships between different body parts of characters are important in capturing the semantics of the scene. When an animator synthesizes or edits such movements, special care is needed to preserve these spatial relationships, for example, "arching back to avoid a punch", "hands extending around each other", "two bodies moving synchronously in close proximity" or "getting into a small car by bending down". However, traditionally, such spatial relationships exist only in the animator's mind and are not digitally embedded into the data. Although humans use spatial relationships to recognize semantics of interactions, their usage has not been considered much in character animation.

Existing scene representations have a fundamental limitation in handling such close interactions. Currently, a motion is typically described in terms of joint angles and kinematic constraints such as contacts. With this representation, automatically computing a valid motion requires randomized exploration and significant computation for collision detection. The animator also needs to shoulder the burden of specifying all the kinematic constraints in advance. From the animator's perspective, this is impractical and not conductive to manual editing. Competitive automatic solutions require an effective representation that allows the extraction of spatial relationships from existing motion data and synthesis of new animations that preserve these relationships. Such a representation will not only allow quantitative evaluation of the way different body parts are interacting, but also facilitate qualitative characterization of scene semantics.

In this paper, we propose a simple representation which we call the *interaction mesh* to represent the spatial relationships between nearby body parts. The interaction mesh is a volumetric mesh defined by the joints of the characters and the vertices of the objects/environment with which the characters are interacting. When editing or retargeting the movements, the motions are automatically adapted by deforming the interaction meshes at all frames with efficient Laplacian deformation techniques [Alexa 2003; Zhou et al. 2005]. The high-level semantics of the interactions are maintained through preserving the local details of the interaction meshes.

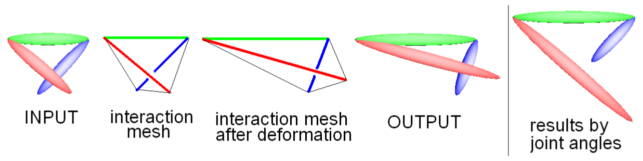

Figure 2: *The posture of an articulated body retargeted to a new morphology with longer red/green and shorter blue segments. Note that a naïve approach by joint angles results in a change of context.*

The interaction mesh representation is general. It provides a unified treatment for interacting body parts of single or multiple characters as well as objects in the environment. As a result, it is applicable to many types of scenarios, such as when single character's actions involve close interactions between different body parts (dancing) or multi-character interactions (wrestling, fighting games). Additionally, the motions may either involve much tangling and contacts (e.g. judo, Fig. 1) or little contact (e.g. Lambada dance). Additionally, it can be applied either per-frame or in the space-time domain according to the complexity of the problem and the available computing resources.

Motion adaptation with the interaction mesh is fully automatic. When the animator changes the size or morphology of the characters or edits parts of the motion, the system automatically deforms the interaction meshes at all the frames using a spacetime optimization and creates a new motion sequence that preserves the original context of the scene. No constraints need to be specified by the animator since they are all encoded in the interaction meshes. If desired, the user may add extra constraints such as anchoring the bodies at the feet. The approach is efficient, allowing real-time control of characters in virtual environments. Specifically, the computational cost increases only linearly in the number of frames and the complexity of the articulated body structures.

The interaction mesh is useful for synthesizing motions for films, computer games and digital mannequin systems. We demonstrate its usefulness in character animation by retargeting captured human motions to characters of very different proportions and volumes, such as a monkey and also by editing the motions of multiple characters while preserving the original context of the scene.

Contributions We introduce a new representation called the interaction mesh for encoding the spatial relationships between closely interacting body parts of articulated characters and objects in environment. We then present an automatic method that uses the interaction mesh for editing or retargeting motions with close interactions. The synthesized motions preserve the spatial relationships, and thus the scene semantics, while reducing the number of inappropriate interpenetrations.

2 Related Work

Most existing motion synthesis methods use kinematic constraints such as positional constraints to enforce a spatial relationship between characters and the environment. A few more recent works on character animation consider implicit spatial relationships.

Constraint-based motion synthesis Since kinematic constraints can usually be represented by single equations, they can be easily embedded into optimization problems for motion synthesis. Such an approach has been adopted for physically-based animation [Popović and Witkin 1999; Komura et al. 2000; Liu and Popović' 2002; Fang and Pollard 2003], motion editing [Gleicher 1997; Callennec and Boulic 2004; Komura et al. 2004; Liu et al. 2006; Shum et al. 2009] and motion retargeting [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000]. One of the early works by Gleicher [1998] handles close interactions of multiple characters. He retargets close dancing motions of two characters to bodies of different sizes while keeping their hands connected using positional constraints. Other methods avoid penetrations of interacting body parts by using inequality constraints [Liu et al. 2006] or a combination of collision detections and equality constraints [Xu et al. 2007; Shi et al. 2007]. These methods produce excellent results for interactions with contacts, however, they are not applicable for maintaining spatial relationships that are less explicit, because representing them as single equations is difficult. For example, in Lambada dances, the dancers twist their bodies around each other without necessarily any body contact. Handling such motions where the interaction conditions are largely implicit is difficult since the context of the scene must be preserved while avoiding penetrations and collisions. Without a good representation of such implicit spatial relationships, the motion synthesis requires complex global path planners involving significant collision detection effort and randomized exploration [LaValle and Kuffner 2001; Yamane et al. 2004; Shapiro et al. 2007], which is difficult for large numbers of degrees of freedom.

Character animation by spatial relationships There have been a few recent works which take into account the implicit spatial relationships of multiple characters when synthesizing new animations. Kwon et al. [2008] handle the spatial relationships between characters in group motions by encoding the neighborhood formations and individual trajectories as Laplacian coordinates. When editing the trajectories, the relative spatial arrangements of characters are preserved by applying Laplacian mesh editing techniques [Alexa 2003; Sorkine et al. 2004]. Our method is similar in that we also employ Laplacian mesh editing technique, but we are addressing a very different problem. The individual characters in their case are 2D particles with no close interactions. In contrast, our method aims to preserve the spatial relationships between the bodies of 3D articulated characters, which requires considering the connections at the joints, rigidity of the body components and penetrations between them.

Ho and Komura [2009b] use Gauss Linking Integral to detect tangled limbs and encode them using rational tangles for motion retrieval. They later proposed a new representation called topology coordinates for representing tangled body parts [Ho and Komura 2009a] and applied it to synthesize character motions in close contact. These methods are adequate for handling motions involving tangles between 1D manifolds such as strands or skeletons. However, extension to motions involving character shapes seems difficult since relationships between rigid bodies or surfaces need to be encoded. Further, these methods cannot handle close interactions without any tangles. The proposed method considers the relationships among rigid body parts and is more general since it can handle motions of close interactions with/without tangles.

A recent work by Zhou et al. [2010] for deformation transfer represents the spatial relationships between multiple components of an object by Euclidean distance and encode them using a minimum spanning tree. Since the spatial relationships are assumed to be fixed (same as rest pose) during deformation, the method is not applicable to motions with time-varying spatial relationships.

3 Overview

We give an overview of our method in this section. First, the data of the original characters and the motion is loaded into our system. Each body segment of the character model is surrounded by a bounding volume which will be used for collision detection. The interaction mesh is then computed for every time frame.

Next, the user edits or retargets the motion by specifying some of the following: the target body sizes, morphology, target positions or new trajectories of some body parts. These constraint parameters are then interpolated and used to morph the original bodies to the target bodies. At every morph-step, the entire motion is adapted towards achieving the target motion. This iterative approach is necessary since the collisions between the body parts/objects need to be carefully monitored and resolved.

At every morph-step, the system adapts the motion by minimizing the Laplacian deformation of the interaction meshes at all the animation frames (or a fixed window of frames at a time, according to available computing resources) and the acceleration of the bodies in these frames (see Fig. 2 for an example of the adapted result of one frame). This spacetime optimization is performed to ensure temporal coherence of the motion. The optimization is subject to various constraints, namely, bone-length constraints, collision constraints and positional constraints. Collisions are then detected between the bounding volumes. If collisions are detected, the penetration depths are evaluated and a new set of collision constraints are defined to resolve the penetrations in the next morph-step.

4 Interaction Mesh

In this section, we describe how we compute the interaction meshes for a given motion. We assume that the mesh characters are rigged with skeletons, and each body segment is bounded by a volume (we use capsules and boxes in our experiments).

The postures of the characters are represented by the positions of the joints, rather than the joint angles. Using the joint positions as parameters has the advantage of making the constraint matrix sparse since the joints are treated as independent particles. In contrast, using the joint angles as parameters makes the Jacobian matrix very dense, as the joints near the root affect the movements of all the joints below in the hierarchy [Shi et al. 2007].

We compute the volumetric interaction mesh for every animation frame. Using the positions of joints and vertices of objects as a point cloud, we apply Delaunay tetrahedralization [Si and Gaertner 2005] (see Figure 2). Note that the spatial relationships which we want to preserve are those between body parts that are in close proximity and are not occluded by other parts. Since the Delaunay tetrahedralization favors connecting such parts with edges, the Laplacian coordinates of vertices which are defined by vertex neighborhood will lead to mutual influence between these body parts. By the nature of Laplacian mesh editing in preserving local details, the spatial relationships of our interest will be maintained.

The orientation of some body segments cannot be computed only from the positions of joints bounding that segment. For example, the joint positions of the elbow and the wrist are insufficient to confirm the rotation around the forearm. In order to compute such orientations, we sample one extra virtual vertex on the surface of each bounding volume, as in [Shi et al. 2007]. These additional virtual vertices are added to the point cloud when defining the interaction mesh. Since each virtual vertex is not rigidly constrained to the body parts, its position is brought back to the original local coordinate frame once the bone's orientation is confirmed. Another possible solution is to use inverse kinematics. The orientation of the body segments can be inferred from the joint positions and the orientation in the original motion [Bodenheimer et al. 1997]. Such an approach can keep the number of vertices in the interaction mesh low.

5 Spacetime Deformation

In this section we present the spacetime optimization problem that we solve to adapt the motion at each morph-step. The spatial relationships of the body parts/objects are preserved by minimizing the Laplacian deformation energy of all the interaction meshes [Alexa 2003; Zhou et al. 2005] subject to constraints derived from the morphed body sizes, detected collision and user-defined position constraints. We also introduce an acceleration energy to reduce jaggedness between frames.

Deformation energy Let m be the number of vertices in the interaction mesh, $\mathbf{p}^i_j (1 \le j \le m)$ be the vertices at frame i, $\mathbf{V}_i$ be a vector of size $3m$ that includes all $\mathbf{p}^i_j$ such that $\mathbf{V}_i = (\mathbf{p}^{i\,\top}_1, \cdots, \mathbf{p}^{i\,\top}_m)$, and $\mathbf{p}^{i\,\prime}_j$ and $\mathbf{V}'_i$ be the updated vectors after the deformation. The deformation energy of the mesh is defined as

$$\begin{aligned} E_L(\mathbf{V}'_i) &= \sum_j \frac{1}{2} \| \delta_j - L(\mathbf{p}^{i\,\prime}_j) \|^2 & (1) \\ &= \frac{1}{2} \mathbf{V}'^{\top}_i \mathbf{M}^{\top}_i \mathbf{M}_i \mathbf{V}'_i - \mathbf{b}^{\top}_i \mathbf{M}_i \mathbf{V}'_i + \frac{1}{2} \mathbf{b}^{\top}_i \mathbf{b}_i & (2) \end{aligned}$$

where L is the operator to compute the Laplacian coordinates from the vertex locations $\mathbf{V}_i$, δ_j is the original Laplacian coordinate, and $\mathbf{M}_i, \mathbf{b}_i$ are the matrix and vector, respectively, computed by expanding Eq.(1). The Laplacian coordinates are calculated by:

$$L(\mathbf{p}_j) = \mathbf{p}_j - \sum_{l \in N_j} w^j_l \mathbf{p}_l \quad (3)$$

where N_j is the one-ring neighborhood of $\mathbf{p}_j$ and w^j_l are the normalized weights which are set as inversely proportional to the distance between the vertices so that farther apart vertices have less influence on each other.

Acceleration energy To reduce jaggy jumps between frames, we introduce an acceleration energy term E_A which imposes temporal relations between corresponding vertices in adjacent frames. Specifically, to reduce sudden acceleration, we minimize the movement of the corresponding vertices in adjacent frames:

$$E_A(\mathbf{V}'_{i-1}, \mathbf{V}'_i, \mathbf{V}'_{i+1}) = \frac{1}{2} \| \mathbf{V}'_{i-1} - 2\mathbf{V}'_i + \mathbf{V}'_{i+1} \|^2 \quad (4)$$

where $\mathbf{V}'_i$ is the set of vertices at frame i.

5.1 Constraints

Here we explain the bone-length constraints, positional constraints and collision constraints imposed in the spacetime deformation.

Bone-length constraints We introduce the bone-length constraints in order to morph the bone lengths (distance between adjacent joints) from the original scales to the target scales. In each morph-step and each animation frame, the target length l_e for each bone e is computed by linearly blending the original and final lengths. Then, a constraint enforcing the target length is imposed as $(\|\mathbf{p}^{1\prime}_e - \mathbf{p}^{2\prime}_e\| - l_e)^2$ where $\mathbf{p}^{1\prime}_e, \mathbf{p}^{2\prime}_e$ are the end vertices of the edge e. Linearizing all the bone-length constraints result in

$$C_B(\mathbf{V}'_i) = \mathbf{B}_i \mathbf{V}'_i - \mathbf{l}, \quad (5)$$

where $\mathbf{B}_i$ is the Jacobian matrix and $\mathbf{l}$ is a vector of constant terms.

Sometimes the bone-length constraints may conflict with the Laplacian deformation energy which means satisfying the bone-length constraints increases the Laplacian deformation energy. This conflict is the main source of slow convergence. We cope with this problem by excluding the vertex $\mathbf{p}_b$ from the neighborhood of vertex $\mathbf{p}_a$ when computing the Laplacian coordinate of $\mathbf{p}_a$ if the edge connecting $\mathbf{p}_a$ and $\mathbf{p}_b$ corresponds to a bone of the body.

Positional constraints The user can add positional constraints by anchoring some joints or a linear combination of their locations. The original trajectories of these body parts are gradually morphed to the given trajectories in each morph-step. We compute the target locations of these parts, $\mathbf{P}_i$, using linear interpolation, and write the positional constraints as:

$$C_K(\mathbf{V}_i{}') = \mathbf{K}_i\mathbf{V}_i' - \mathbf{P}_i \tag{6}$$

where $\mathbf{K}$ is a $3k \times 3m$ weight matrix that defines the influence of each joint in each positional constraint, and k is the number of positional constraints.

Collision constraints The collision constraints prevent penetration between the bounding volumes of the skeleton. We perform collision detection by applying the ODE library [Smith 2005] to the current configuration of the bounding volumes. Specifically, when a penetration is detected, we compute the penetration depth, directions and the point pair penetrating each other the farthest and add the following constraints:

$$C_C(\mathbf{V}_i') = \mathbf{J}_i\mathbf{V}_i' - \mathbf{d}_i \tag{7}$$

where $\mathbf{J}_i$ is the Jacobian of the positions of the colliding parts with respect to the joint positions, and $\mathbf{d}_i$ is the penetration depth multiplied to the normal vectors of the penetrated surface. The Jacobian is computed by finite differencing. The joint vertices are moved and the locations of the penetrating points are recomputed according to the posture. We do not apply collision detection to adjacent body parts along the body tree structure as self-penetrations easily happen when the joints are bent.

Constraint energy We separate the constraints in Eq.(5)-7 into soft and hard constraints:

$$\mathbf{F}_i\mathbf{V}_i' = \mathbf{f}_i \ \text{(soft)}, \quad \mathbf{H}_i\mathbf{V}_i' = \mathbf{h}_i \ \text{(hard)} \tag{8}$$

and define a constraint energy that represents the amount of violation of the soft constraints:

$$E_C(\mathbf{V}_i') = \frac{1}{2}\mathbf{V}_i'^{\top}\mathbf{F}_i{}^{\top}\mathbf{W}\mathbf{F}_i\mathbf{V}_i{}' - \mathbf{f}_i{}^{\top}\mathbf{W}\mathbf{F}_i\mathbf{V}_i' + \frac{1}{2}\mathbf{f}_i{}^{\top}\mathbf{W}\mathbf{f}_i. \tag{9}$$

where $\mathbf{W}$ is a square diagnol matrix that assigns a different weight to each constraint.

By default, we set the bone-length constraints and one positional constraint (supporting foot) hard, and the collision constraints and the rest of positional constraints soft. The bone-length constraints are set hard so that the bodies are correctly scaled to the target values. Soft collision constraints stabilize the motion when there is little open space. It also provides the animator some results when collisions cannot be avoided due to insufficient open space when bodies are enlarged. It is also necessary to set the other positional constraints soft to avoid over constraining. By default, the weights in $\mathbf{W}$ are set 4.0 and 0.4 for the collision and additional positional constraints, respectively. The constraints can be switched between soft and hard according to the desired animation effect. When the number of morph steps is small, we also need to set the bone-length constraints soft, and their weights are set to 2.0.

Algorithm 1 Motion Adaptation by Interaction Mesh

```
Input: Skeleton and input motion sequence
    initial / final body scaling factors: $\mathbf{s}_0, \mathbf{s}_f$
    locations of initial/final positional constraints: $\mathbf{p}_0, \mathbf{p}_f$
Output: Target motion sequence
(Initialization)
- Compute the interaction meshes (vertex positions and connectivity) of all input frames.
(Motion Adaptation)
for $k$=1 to $N$ morph-steps do
    - Update body scale / location of positional constraints:
        $\mathbf{s}_k \leftarrow \frac{N-k}{N}\mathbf{s}_0 + \frac{k}{N}\mathbf{s}_f, \quad \mathbf{p}_j \leftarrow \frac{N-k}{N}\mathbf{p}_0 + \frac{k}{N}\mathbf{p}_f$
    - Update the constraint matrices $\mathbf{F}_i, \mathbf{H}_i$, the target values of the constraints $\mathbf{f}_i, \mathbf{h}_i$ in Eq.(8) and the deformation vector $\mathbf{b}_i$ in Eq.(2) for frames $i = 1, ..., n$.
    - Update the vertex locations by solving Eq.(10).
    - Compute segment orientations and update virtual vertices.
end for
```

5.2 Iterative Morphing

At every morph-step, the body sizes and the positional constraints are updated, and the motions of the characters are adapted by minimizing the sum of the deformation (Eq.1), acceleration (Eq.4) and constraint energy (Eq.9) of all frames subject to the hard constraints. The adapted motion is computed by solving

$$\arg\min_{\mathbf{V}_i', \lambda_i (1 \le i \le n)} \sum_i^n E_L + w_\Delta E_A + E_C + \lambda_i^{\top}(\mathbf{H}_i\mathbf{V}_i' - \mathbf{h}_i) \tag{10}$$

where n is the number of frames, $\mathbf{V}_i'$ is the set of new vertex positions at frame i, λ_i are the Lagrange multipliers and w_Δ is a constant weight (we use 0.2). Note that E_A is not defined for the first and last frames, hence we set them to zero. The spacetime optimization problem in Eq.(10) can be solved by differentiating it with respect to $\mathbf{V}_i'$ and λ_i, and solving the following linear equation:

$$\begin{pmatrix} \mathcal{M}^{\top}\mathcal{M} + w_\Delta \mathcal{A}^{\top}\mathcal{A} + \mathcal{F}^{k\top}\mathcal{W}\mathcal{F}^k & \mathcal{C}^{k\top} \\ \mathcal{C}^k & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathcal{V} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathcal{M}^{\top}\mathcal{B} + \mathcal{F}^{k\top}\mathcal{W}f \\ \mathcal{H} \end{pmatrix} \tag{11}$$

where $\mathcal{B}$, $\mathcal{H}$, f, $\mathcal{V}$ and λ are vectors that include $\mathbf{b}_i$ (in Eq.2), $\mathbf{h}_i$, $\mathbf{f}_i$ (in Eq.8), $\mathbf{V}_i'$ and λ_i for all the frames, respectively, i.e. $\mathcal{B} = (\mathbf{b}_1^{\top}, ...\mathbf{b}_n^{\top})^{\top}$, $\mathcal{H} = (\mathbf{h}_1^{\top}, ...\mathbf{h}_n^{\top})^{\top}$, $f = (\mathbf{f}_1^{\top}, ...\mathbf{f}_n^{\top})^{\top}$, $\mathcal{V} = (\mathbf{V}_1'^{\top}, ...\mathbf{V}_n'^{\top})^{\top}$, $\lambda = (\lambda_1^{\top}, ...\lambda_n^{\top})^{\top}$ and $\mathcal{M}$ is the Laplacian matrix, $\mathcal{F}^k$ is a soft constraint matrix at the k-th morph-step, each of which includes $\mathbf{M}_i$ in Eq.(2), and $\mathbf{F}_i$ in Eq.(8) for all the frames, respectively:

$$\mathcal{M} = \begin{pmatrix} \mathbf{M}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{M}_n \end{pmatrix}, \mathcal{F}^k = \begin{pmatrix} \mathbf{F}_1^k & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{F}_n^k \end{pmatrix},$$

and $\mathcal{A}$ is a matrix that computes the acceleration for all the frames from $\mathcal{V}$, $\mathcal{W} = \text{diag}(\mathbf{W}, ..., \mathbf{W})$, and $\mathcal{C}^k$ is the constraint matrix at the k-th morph-step, which includes $\mathbf{H}_i$ in Eq.(8) for all the frames:

$$\mathcal{A} = \begin{pmatrix} \mathbf{0} & & & & \\ \mathbf{I} & -2\mathbf{I} & \mathbf{I} & & \\ & & \ddots & & \\ & & \mathbf{I} & -2\mathbf{I} & \mathbf{I} \\ & & & & \mathbf{0} \end{pmatrix}, \mathcal{C}^k = \begin{pmatrix} \mathbf{H}_1^k & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{H}_n^k \end{pmatrix}.$$

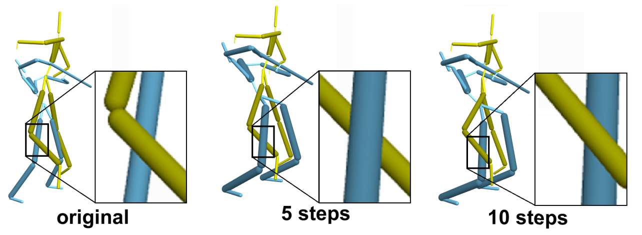

Figure 3: *Flipping of a tetrahedron during a morph-step may lead to a change of spatial context: (left) initial posture, (middle) flipping happens at the left lower leg of the yellow character when using 5 morph-steps, (right) flipping avoided when using 10 morph-steps.*

Our motion adaptation algorithm is summarized in Algorithm 1.

The interaction meshes are defined for the original motion frames and their connectivities are kept unchanged at all the morph-steps, which results in a constant $\mathcal{M}$. Note that re-computing the tetrahedralization at each morph-step would result in gradual drifting of the motion away from the original sequence. Keeping the mesh topology from the original motion helps to preserve the spatial relationships of the components in the original motion.

Possible artifacts and solutions Due to the non-uniform weighting of the edges, drastic updates of the morphing parameters such as the character sizes may result in flipping of the tetrahedra in the interaction meshes. Such flipping, if it occurs in an open space does not cause noticeable artifacts, but may result in a change of context if it happens with a tetrahedron composed of two bones which are nearby but not adjacent to each other. We prevent the flipping by detecting collisions between the body parts and applying the collision constraints (Eq.7) at each morph step. However, the system may fail when the linearization of the collision constraints breaks down, which happens when the body sizes are very narrow and the penetration depth is too large. This results in pushing out the penetrating body in the wrong direction (see Fig. 3). The problem can be avoided by giving enough volume to the body parts and setting small morph steps. We obtained satisfactory results using 10 morph-steps for all the examples.

The soft collision constraints can result in penetrations when the bodies are enlarged too much while there is limited open space. We can alert the user of the lack of space by giving the sum of the squares of penetration depth for reference. The user can then choose to either enlarge the environment or stop scaling.

The system may become unstable when the original motion contains movements where the body parts pass through each other. When this happens, the direction the collision constraint pushes the bodies away from each other will turn opposite at some moment, resulting in a large movement in one frame. This contradicts the minimization of acceleration energy and causes vibrations. Switching off the collision constraints at the frames of these pass throughs can remove such artifacts.

6 Experimental Results

In this section, we show experimental results from applying our motion retargeting method to character animation. We apply it to several types of motions with close interactions, namely, between body parts of a single character or multiple characters, and between a character and its environment. We also demonstrate its usefulness for real-time character control. The heights of the feet are all constrained to the original values by hard constraints by default, which are necessary especially when the characters are standing on the ground. The readers are referred to the supplementary video for further details.

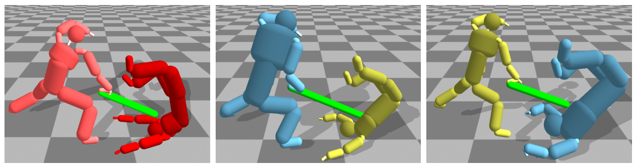

Figure 4: *Snapshots of a sword attacking motion (left) retargeted to characters of different morphologies (middle, right).*

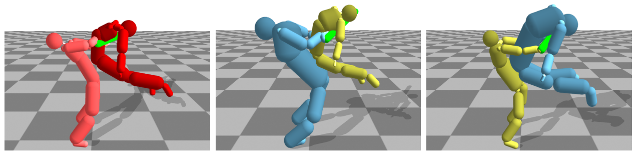

Figure 5: *Snapshots of a back breaking attacking motion (left) retargeted to characters of different morphologies (middle, right).*

6.1 Retargeting Motions of Close Interactions

First we show the results of retargeting motions involving only characters. We use motions involving much tangling and contacts, such as judo, as well as those with few contacts, such as dancing and attackers/defenders in fighting games.

Judo attacks Our first motion example is a judo "Ogoshi throw" in which the attacker holds the arm and the waist of the defender and throws the defender by carrying him/her onto the back. We retarget the motions of the thrower and the defender to characters of various morphologies. We use the body sizes of characters in Allen et al. [2003] as reference (See Fig. 1). Here, although the proportions and the bounding volumes of the new characters are completely different from those of the original characters, our system can still produce the Ogoshi throw. Note that previous motion editing/retargeting approaches are difficult to apply to this kind of close interactions since they only consider the joint angles of the original motion but not the spatial relationships. As a result, the retargeted motion may have a different context (e.g. the arms extend to the other side of the defender's body). They also require the animators to manually specify all the positional constraints in all the frames, such as the attacker's hand holding the defender's body. This can be a tedious task for the animator.

Fighting scenes Our next two motion examples are fighting scenes involving two characters provided by a game company.

The first scene involves a character holding a sword attacking its enemy. The sword penetrates through the enemy character when the enemy is stabbed, and therefore, we turned off the collision constraint in those frames. The second scene is from the same game. The character holding the sword breaks the back of the enemy with both arms and drops it. The sword unintentionally passes through the arm of the defender in some of the frames, which is due to the manual design. Again, we turned off the collision constraint in those frames. Both the attacker's and defender's morphologies are changed and animations of different combination of bodies are created. Snapshots of the original and synthesized motions are shown in Fig. 4 and Fig. 5, respectively. Note that manual editing would

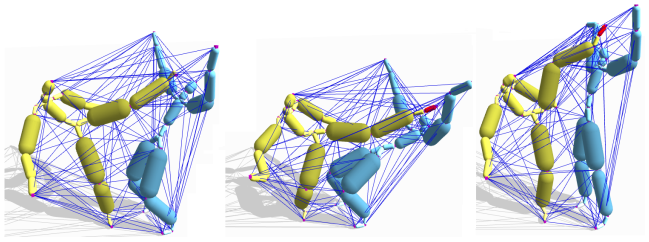

Figure 6: *(left) A posture of a turn kick interaction. (middle, right) The animator drags the left foot of the yellow character by mouse and the other character moves to preserve the spatial relationship.*

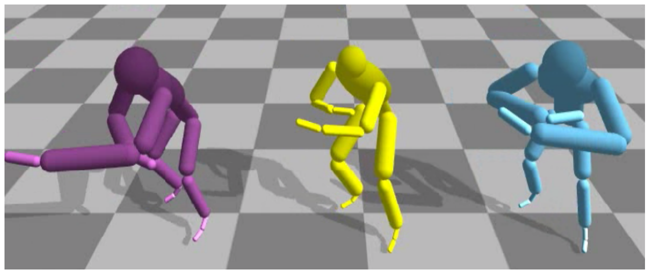

Figure 7: *Original dancing motion (middle) and the retargeted results to a monkey model with long arms using a joint-angle based method (left) and using our method (right).*

require a lot of care to avoid unintentional penetrations of the sword into the body. Our method can automatically produce the motion of passing the sword into the space between the enemy's arm and torso. These examples show that our method can also be used for manually designed motions which are not penetration-free. In such cases, the context of the penetrations (e.g. stabbing the enemy) are preserved in the synthesized results. Feedback from the game company indicates that the quality of our retargeted movements is high enough for games usage.

Interactive character control Next, we show a demo of using the interaction mesh for real-time control of characters. Pausing the animation of interaction at some frame, we can let the user interactively control a body part using inverse kinematics while maintaining its spatial relationships with the other character(s). In such cases, we solve Eq.(11) for a single frame rather than for the entire motion, which provides real-time performance. The other character(s) will follow the movements of the controlled character according to the interaction mesh at that frame. The controlled body part is softly constrained by an additional positional constraint. An example of editing a posture in a turn-kick motion is shown in Fig. 6. The updates in the edited frame can be propagated to the whole motion by iteratively solving Eq.(10) using the edited posture as a constraint.

Single character motions We use a dancing motion in which the character performs an arms cycling motion and retarget it to a character with monkey proportions (Fig. 7). Our method can preserve the context of the motion despite the much longer arms of the monkey character. In contrast, the method of [Lyard and Magnenat-Thalmann 2008] results in a motion with many collisions, causing the movements to appear unstable. Note that since this motion does not involve any tangles, the topology coordinates [Ho and Komura 2009a] are also difficult to apply.

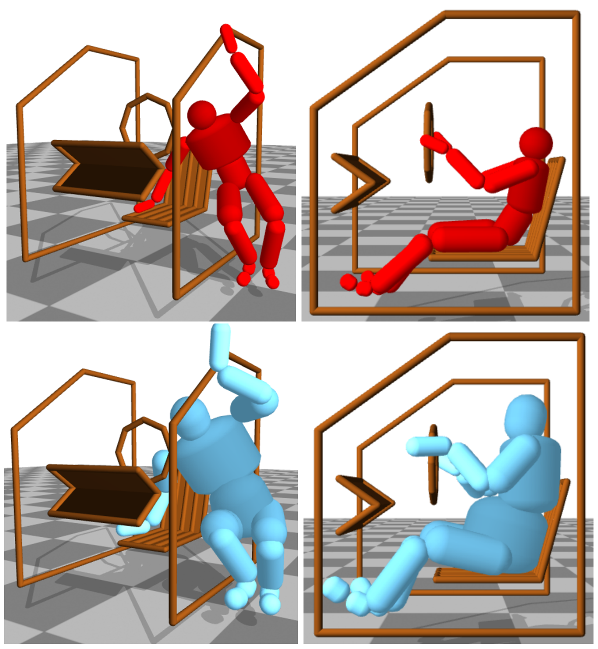

Figure 8: *Snapshots of a character getting into and riding a car model; (top) original character and (bottom) a tall fat character.*

6.2 Motion Adaptation in a Constrained Environment

Motions in a constrained environment, such as getting in and out of a car, involve close maneuvers and collision avoidance. Retargeting such motions to characters of different sizes or adapting the motions to environment with different parameters (e.g. size of car) has a great demand in CAD design and digital mannequins [Badler et al. 1999]. Here we show examples of using the interaction mesh for such a purpose. We captured the motion of a person getting into a car and holding the steering wheel. The environment is composed of simple polylines representing the car doors, ceiling, floor, seats, and steering wheel. The interaction mesh is composed of the vertices of the environment and the skeleton joints and end effectors. Snapshots of the input motion and that retargeted to a scaled character are shown in Fig. 8. Observe that the character's motion is successfully adapted to the new character size. Since some of the interactions of the character with the car, such as ducking and passing through the narrow space, cannot be described only with explicit constraints such as contacts, these motions are difficult to handle for previous methods.

6.3 Computational Costs

The main bottleneck of our method is solving the large linear equation in Eq.(11). We use UMFPACK [Davis 2004] and GotoBLAS [Goto and Van De Geijn 2008]. Since the Laplacian matrix $\mathcal{M}$ and the constraint matrix C^k are both sparse, the computation only increases linearly with respect to the number of vertices in all the interaction meshes. Therefore, the complexity of the problem is $O(m \times n)$, where m is the number of vertices in each mesh and n is the number of frames. For all the retargeting examples shown in this paper, the computation required for each motion is around 1 minute for an animation of 100 frames, using one core of a Core i7 2.67GHz CPU. Since most of the computation, such as the computation of the Laplacian and constraint matrices and solving the large linear system [Bolz et al. 2003] are highly parallelizable, much faster response can be expected with GPU implementations.

7 Discussions

In this section, we compare our approach with previous methods in terms of advantages and limitations.

Simply combining the kinematic constraints and collision avoidance as is done in [Lyard and Magnenat-Thalmann 2008] cannot preserve spatial relationships between bones. Although collision detection can avoid interpenetration of body parts, the movement of one part does not affect the movements of the other parts until collision occurs. Moreover, as shown in Fig. 7, left, the collision does not necessarily repulse the body in the direction that maintains the context of the original scene. As a result, coordination between the bodies can easily be lost. With our method, the interaction mesh moves all the nearby bones together such that their spatial relationships are maintained.

Shi et al. [2007] apply a cascading scheme for motion retargeting to speed up the process and make use of the GPU resources. Such a cascading scheme is inapplicable to the motions of our interest since it may greatly update the posture after one iteration. Large updates may be acceptable for open postures such as standing and reaching out for an object, however for motions with close interactions between body parts, resolving all the collisions after a large update may result in a change of context, for example, the body parts may be flipped to the opposite side. This also explains why we need to gradually morph the morphology of the characters and sizes of objects in the scene.

The concept of the interaction mesh is very general. It is possible to compute the interaction mesh using vertices of freeform meshes and use it for shape deformation. This can be an interesting application of deformation transfer [Sumner and Popovic 2004; Zayer et al. 2005] and cloth animation. For example, animation of a character wrapping an object or wearing clothes.

Using a high resolution volumetric mesh, such as one defined by the surface of a mesh character, may be a good solution for intricate finger motions, such as manipulating a thread or small object, as it can represent the details of the interaction between the fingers' surface and the object. However, applying such a method for spacetime control of a full body character will be very computationally costly due to the huge number of vertices.

Spacetime optimization is known to be an important tool for creating realistic character motions since it allows the characters to prepare for the interactions ahead along the timeline [Liu et al. 2006], such as an early bending of the torso before entering a car. It also removes jaggedness from the movements, which is a drawback of frame-based approaches. Therefore, we argue that our combined use of a volumetric structure defined by joint positions and the spacetime optimization is a good design choice for efficient synthesis of realistic motions of interactions.

The acceleration energy term might unnecessarily smooth the large accelerations from impacts, which is often an important feature of motions. Experimentally, we have found the advantage of using the acceleration term outweigh the smoothing of high impulse movements. One possible solution for maintaining the features of high acceleration is to use inter-frame Laplacian coordinates proposed in Kwon et al. [2008] instead of the acceleration energy.

Limitations Our method may fail when the constraints are drastically different from those in the original motion, such as when one of the interacting characters is scaled too small or too large. In such situations, the body parts may be too far apart to maintain the interactions. However, any methods that use inverse kinematics are not immune to such a problem under extreme scaling. An example of this vulnerability is when a character is in contact with another object at many parts of its body, for example, a full body hold. If the object's size is scaled down, the Laplacian deformation will try to preserve the spatial relationship at all the contact areas, which is physically impossible due to the rigidity of the body parts. As a result, the deformation error will be shared among all the contact areas, resulting in the loss of all contacts. This is related to the fact that our method does not require user-specified contact constraints. A simple possible solution is to prioritize such contacts and impose contact constraints at areas which appear to be important. Another solution may be to recompute the interaction meshes at certain morph-steps when the spatial relationship is difficult to preserve and allow the mesh to drift from the original topology.

8 Conclusion and Future Work

In this paper, we have presented a new method to edit and retarget character animation that involves many close interactions by introducing a new representation called interaction mesh. When updating the motion, the spatial relationships between different body components and objects can be preserved by applying a spacetime optimization to minimize the deformation of the interaction meshes subject to collision constraints as well as bone-length and positional constraints. The method is fully automatic, not requiring manual intervention from the user. We have demonstrated the effectiveness of the proposed method by showing realistic synthesized animations of various types of close interactions, which are difficult to produce using previous methods.

As a future work, we plan to numerically evaluate the topological and geometric features of the interaction meshes, and introduce a metric to compare motions at the level of interaction meshes. Our method can then be extended for use in reinforcement learning, enabling computer-controlled characters to smartly interact with user-controlled characters in closely interacting environments. Another possible research direction is to apply our method for controlling characters in physically-based environments, such as by combining it with the balance keeping techniques in [da Silva et al. 2008; Macchietto et al. 2009]. Tackling such a problem may also lead to solutions for controlling multi-biped robots to cooperatively accomplish tasks such as carrying objects together.

Acknowledgement

We thank the anonymous reviewers for their constructive comments, Oscar Kin-Chung Au and Hubert Pak-Ho Shum for valuable feedback and Adam Barnett for the video narration and suggestions. We also thank NAMCO BANDAI Games Inc. for providing the game motion data. This work was partially supported by grants from EPSRC (EP/H012338/1) and the Hong Kong Research Grant Council (Project No. GRF619908).

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2-3, 105–114.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics 22*, 3 (Jul.), 587–594.

BADLER, N. I., PALMER, M. S., AND BINDIGANAVALE, R. 1999. Animation control for real-time virtual humans. *Communications of the ACM 42*, 8, 64–73.

BODENHEIMER, B., ROSE, C., ROSENTHAL, S., AND PELLA., J. 1997. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation 97*, 318.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖODER, P. 2003. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Transactions on Graphics 22*, 3 (Jul.), 917–924.

CALLENNEC, B. L., AND BOULIC, R. 2004. Interactive motion deformation with prioritized constraints. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 163–171.

CHOI, K.-J., AND KO, H.-S. 2000. Online motion retargeting. *Journal of Visualization and Computer and Animation 11*, 5, 223–235.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27*, 3 (Aug.), 82:1–10.

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

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (Jul.), 417–426.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings of Symposium on Interactive 3D Graphics*, 139–148.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 98*, ACM Press / ACM SIGGRAPH, M. Cohen, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 33–42.

GOTO, K., AND VAN DE GEIJN, R. 2008. High-performance implementation of the level-3 blas. *ACM Transactions on Mathematical Software 35*, 1 (Jul.), 1–14.

HO, E. S. L., AND KOMURA, T. 2009. Character motion synthesis by topology coordinates. *Computer Graphics Forum 28,* 2, 299-308.

HO, E. S. L., AND KOMURA, T. 2009. Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (May), 481–492.

KOMURA, T., SHINAGAWA, Y., AND KUNII, T. L. 2000. Creating and retargetting motion by the musculoskeletal human body model. *The Visual Computer*, 5, 254–270.

KOMURA, T., LEUNG, H., AND KUFFNER, J. 2004. Animating reactive motions for biped locomotion. In *Proceedings of ACM Virtual Reality Software and Technology*, 32–40.

KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. 2008. Group motion editing. *ACM Transactions on Graphics 27*, 3 (Aug.), 80:1–8.

LAVALLE, S., AND KUFFNER, J. 2001. Rapidly-exploring random trees: Progress and prospects. In *Robotics: The Algorithmic Perspective. 4th Int'l Workshop on the Algorithmic Foundations of Robotics*, 293–308.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, ACM Press / ACM SIGGRAPH, A. Rockwood, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 39–48.

LIU, C. K., AND POPOVIĆ', Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (Jul.), 408–416.

LIU, C. K., HERTZMANN, A., AND POPOVIC, Z. 2006. Composition of complex optimal multi-character motions. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 215–222.

LYARD, E., AND MAGNENAT-THALMANN, N. 2008. Motion adaptation based on character shape. *Computer Animation and Virtual Worlds 19*, 3-4, 189–198.

MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. R. 2009. Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (Aug.), 80:1–8.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, ACM Press / ACM SIGGRAPH, A. Rockwood, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 11–20.

SHAPIRO, A., KALLMANN, M., AND FALOUTSOS, P. 2007. Interactive motion correction and object manipulation. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D graphics and Games*,137-144.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Transactions on Graphics 26*, 3 (Jul.), 81:1-9

SHUM, H. P. H., KOMURA, T., AND YADAV, P. 2009. Angular momentum guided motion concatenation. *Computer Animation and Virtual Worlds 20*, 2-3, 385–394.

SI, H., AND GAERTNER, K. 2005. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, 147–163.

SMITH, R. 2005. Open dynamics engine. www.ode.org.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 179–188.

SUMNER, R. W., AND POPOVIC, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (Aug.), 397–403.

XU, W., ZHOU, K., YU, Y., TAN, Q., PENG, Q., AND GUO, B. 2007. Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics*, 26, 3 (Jul.), 84:1–10.

YAMANE, K., KUFFNER, J., AND HODGINS, J. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics 23*, 3 (Aug.), 532–539.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. *Computer Graphics Forum 24*, 3, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Transactions on Graphics 24*, 3 (Jul.), 496–503.

ZHOU, K., XU, W., TONG, Y., AND DESBRUN, M. 2010. Deformation transfer to multi-component objects. *Computer Graphics Forum 29*, 2.