Synchronized Multi-Character Motion Editing

Manmyung Kim Kyunglyul Hyun Jongmin Kim Jehee Lee

Seoul National University

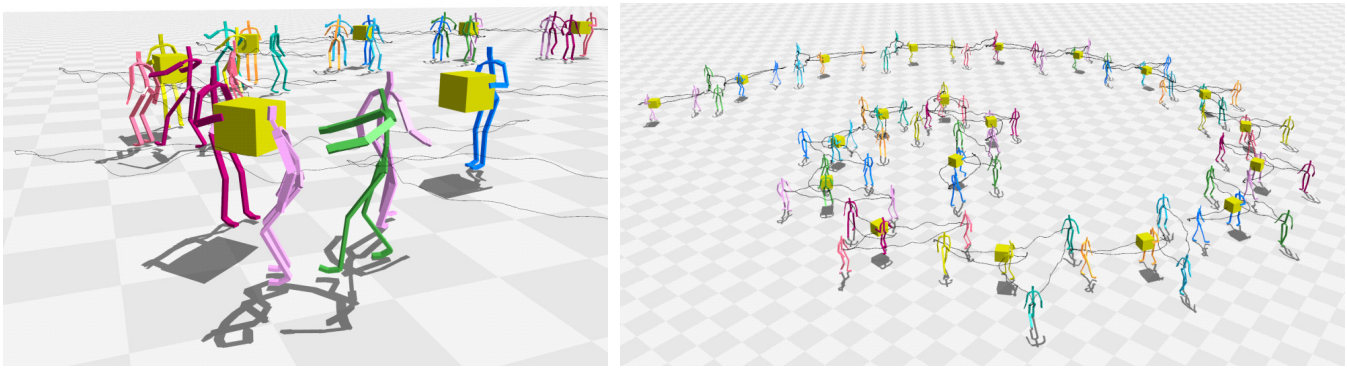

Figure 1: 78 characters carry boxes in relay. We interactively manipulated the synchronized multi-character motion to shape a spiral.

Abstract

The ability to interactively edit human motion data is essential for character animation. We present a novel motion editing technique that allows the user to manipulate synchronized multiple character motions interactively. Our Laplacian motion editing method formulates the interaction among multiple characters as a collection of linear constraints and enforces the constraints, while the user directly manipulates the motion of characters in both spatial and temporal domains. Various types of manipulation handles are provided to specify absolute/relative spatial location, direction, time, duration, and synchronization of multiple characters. The capability of non-sequential discrete editing is incorporated into our motion editing interfaces, so continuous and discrete editing is performed simultaneously and seamlessly. We demonstrate that the synchronized multiple character motions are synthesized and manipulated at interactive rates using spatiotemporal constraints.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Animation—Virtual reality

Keywords: Character Animation, Interactive Motion Editing, Human Motion, Motion Capture, Multi-Character Interaction

1 Introduction

The ability to interactively edit human motion data is a vital tool for character animation. The common goal of motion editing is to make a desired change to existing motion data while preserving the essential quality and features of the original motion as much as possible. Interactive editing techniques typically allow the user to manipulate motion data by specifying pinning constraints and dragging its moving trajectory directly.

Collaborative and adversarial interaction among multiple characters usually requires synchronization in both space and time. Most existing motion editing techniques allow the motion of each individual character to be manipulated independently. Editing synchronized multi-character motions can be cumbersome if the animator has to maintain interpersonal constraints manually for synchronization. We present a novel interactive motion editing technique that allows the animator to specify the interaction among characters as a collection of linear constraints and manipulate synchronized multiple character motions simultaneously, while maintaining the constraints. Various types of manipulation handles and interpersonal constraints are provided to specify absolute/relative spatial location, direction, temporal duration, and timing of interaction. The user can stretch and squeeze synchronized motion data in both spatial and temporal domains through intuitive, easy-to-use user interfaces.

Our motion editing method is based on a one-dimensional version of Laplacian mesh editing [Igarashi et al. 2005; Sorkine et al. 2004]. This method allows a polygonal curve to be deformed in an as-rigid-as-possible manner. We generalize this idea to deal with synchronized motion paths. To do so, we discuss how to incorporate spatial/temporal constraints into the Laplacian formulation and how to cope with degenerate cases. We employ this path editing algorithm for both spatial path editing and interactive time warping, which requires smooth time-warp curve editing. Therefore, the spatial and temporal aspects of motion data can be handled uniformly.

The Laplacian editing method makes smooth, continuous changes to the original motion by warping its moving trajectory. The underlying structure of the motion remains intact for mild stretching, squashing, and bending. However, discrete, structural transformations of motion (for example, adding or removing walking steps) should be made to accommodate large edits in the motion path. Such discrete motion editing can be implemented by segmenting motion data into short clips and splicing them in a novel order [Lee et al. 2002; Kovar et al. 2002]. Our system incorporates non-sequential discrete editing capabilities seamlessly into motion

editing interfaces, so discrete changes can ensue as fluidly as continuous path editing during interactive manipulation. Discrete optimization of a motion path is notorious for its exponential computation complexity with respect to the number of potential discrete transformations. General discrete optimization using a collection of rich connected motions is impractical for interactive applications. Fortunately, interactive editing is inherently incremental and thus no sudden jump is permitted in manipulation. This gradual change assumption allows us to achieve interactive performance by taking an efficient, incremental strategy.

The continuous and discrete editing capabilities of our system are demonstrated with examples that require precise spatiotemporal control of multiple characters interacting with each other and environment objects. Our specific contributions are threefold.

- A unified formulation that can deal with spatial, temporal, and inter-personal manipulation
- A framework that seamlessly combines continuous and discrete motion editing
- An user interface that allows for intuitive manipulation of synchronized multi-character motion

2 Related Work

Researchers have studied ways to generate a continuous span of human motion by either warping a single motion segment or interpolating multiple motion examples. Given input motion data, motion warping techniques make a smooth change to the motion data to satisfy user-specified constraints. Uni-/multi-level motion displacement mapping [Gleicher 1998; Lee and Shin 1999] has often been employed to preserve the detailed features of the original motion data. The techniques based on motion displacement mapping are versatile and adequate for editing joint trajectories. However, their applicability is limited if the moving path of the character's root should be warped considerably during interactive manipulation. Gleicher [2001] considered a two-dimensional motion path as an abstraction of the positional movement of a character and allowed the motion path to be directly manipulated for motion editing. Manipulation objectives and handles have also been derived from either statistical models [Chai and Hodgins 2007] or physical properties, such as linear/angular momentum [Liu and Popović 2002] and balancing [Sulejmanpasic and Popović 2005; Tak and Ko 2005; Zordan and Hodgins 2002]. These ideas have further been extended to deal with varied morphologies [Hecker et al. 2008].

Parametric motion blending requires a set of motion examples that are structurally similar. Kovar and Gleicher [2004] presented a method that identifies a set of potentially blendable motions from motion data sets. Mukai and Kuriyama [2005] explored a motion interpolation method that provides precise control over interpolated motions. Time warping is an essential ingredient of motion interpolation to establish correspondence between example motions. Hsu et al. [2007] presented a time warping technique for modifying the duration of motions while preserving the natural timing of input motion. McCann et al. [2006] employed physically based objective functions and constraints for interactive motion retiming.

A typical process of non-linear, structural motion editing begins with a set of motion segments and splicing them to synthesize a novel motion sequence. The set of motion segments are often maintained in a graph representation that stores the possibility of transition from one segment to another [Lee et al. 2002; Kovar et al. 2002]. Provided with the graph, motion synthesis and editing can be posed as various forms of combinatorial optimization problems addressed by a state space search [Lee et al. 2002; Kovar et al. 2002], dynamic programming [Arikan et al. 2003; Hsu et al. 2004; Pullen and Bregler 2002], path planning [Safonova and Hodgins 2007], and reinforcement learning [Lee and Lee 2004; McCann and Pollard 2007; Treuille et al. 2007; Lo and Zwicker 2008].

Several researchers explored the convergence of motion graphs and parametric motion interpolation [Shin and Oh 2006; Heck and Gleicher 2007]. The implementation of this convergence concept typically requires segmentation of motion data, clustering similar motion segments into groups, and parameterization in each group. The graph thus obtained allows parametric interpolation of motion while traversing though the graph. Alternatively, Shin and Lee [2006] projected a motion graph into a low-dimensional space, in which the transitioning and interpolation between motions can be parameterized seamlessly. Safonova and Hodgins [2007] represented the motion as an interpolation of two time-scaled paths through a motion graph and performed discrete optimization to satisfy user-specified constraints.

Research on multiple character interaction has recently emerged. Liu et al. [2006] explored a method to combine a small collection of motion clips to synthesize multiple character interactions with physically based objectives and constraints. Kwon et al. [2008a] modeled competitive interaction between two Taekwondo players as a dynamic Bayesian network. Motion patches by Lee et al. [2006] were originally developed to animate characters in a complex virtual environment by identifying a collection of building blocks of the environment annotated with motion data. Shum et al. [2008] generalized the idea further to encapsulate short-term interaction between two characters into "interaction patches" and combined them to synthesize larger-scale interaction among many characters. Ho and Komura [2009] synthesized multiple characters interacting in close physical contacts by analyzing their topology coordinates.

Our work is similar to group motion editing by Kwon et al. [2008b] in the sense that both addressed interactive editing of multiple character motions. They represent the group locomotion of pedestrians as a three-dimensional mesh and allows the animator to directly manipulate the group motion, while maintaining its group formation and individual moving trajectories in the original animation as much as possible. The limitation of group motion editing is that it can handle only the locomotion of pedestrians. Our work extends the scope of motion and interaction to cope with an arbitrary type of character motions and a wider range of inter-personal interaction in our interactive editing system.

3 Multiple Character Interaction

Animated characters may interact with environments and other characters synchronized both in time and space. Such character-environment and character-character relations are described in terms of their relative spatial locations, spatial directions, and timing of interaction. We will consider multiple character interactions that can be specified as linear constraints and will formulate the motion editing problem as an over-determined system of linear equations with linear equality constraints.

The motion of an articulated character is represented by its time-varying position and orientation of the root body segment in the reference coordinate system together with its joint angles also varying over time. The root trajectory projected onto the ground forms a motion path. Our system allows us to specify constraints either on a motion path or on any end-effectors to edit a full-body motion. In this paper, we focus on editing multiple motion paths synchronized by interpersonal constraints. Whenever each path is modified, we refine the full-body motion immediately to enforce end-effector constraints and remove foot sliding artifacts.

Symbol	Meaning
subscript i, j	frame indexes
superscript k, α, β	path indexes
subscript s	spatial
subscript t	temporal
$\mathbf{p}_i \in \mathbb{R}^2$	position of the root at frame i
$\theta_i \in \mathbb{R}$	orientation of the root at frame i
$\mathbf{t}_i \in \mathbb{R}^2$	tangent vector at frame i
$\mathbf{n}_i \in \mathbb{R}^2$	normal vector at frame i
$t_i \in \mathbb{R}$	time-warp at frame i
$\mathbf{P}^k$	k-th motion path
$\mathbb{P} = \{\mathbf{P}^k\}$	a set of motion paths
$\mathbb{C}$	a set of constraints

Figure 2: Mathematical notation

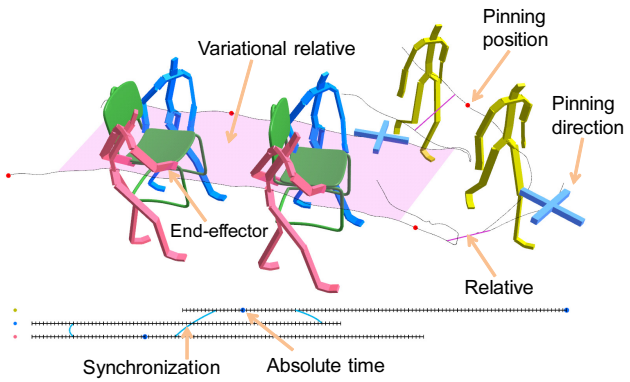

Figure 3: Interactive path editing with spatial and temporal constraints.

3.1 Spatial Constraints

Let $\{(\mathbf{p}_i, \theta_i) \in \mathbb{R}^2 \times \mathbb{R}\}$ be a motion path that describes a time series of body (pelvis) locations and directions on the horizontal plane. For the convenience in implementation, we represent position $\mathbf{p}_i$ in a global reference coordinate system and direction θ_i with respect to a local coordinate system defined by tangent and normal vectors of the motion path. We estimate the tangent vector at point $\mathbf{p}_i$ by finite difference, $\mathbf{t}_i = \mathbf{p}_{i+1} - \mathbf{p}_{i-1}$, and the normal vector by $\mathbf{n}_i = R\mathbf{t}_i$, where R is a 2×2 rotation matrix of angle 90°. Since direction θ_i is bound to the local coordinate system, the direction of a character is determined automatically with respect to the deformation of its translational moving path. Henceforth, we will manipulate only the position components in motion paths and allow the direction components to be determined accordingly.

The tangent vector at a point may degenerate for a stationary motion such that $\|\mathbf{t}_i\| < \epsilon$ for minuscule ϵ. To cope with this situation, we first identify all degenerate points by checking the length of tangent vectors. We then determine new tangent vectors at those points by linearly interpolating nearby non-degenerate tangent vectors.

Our system provides three types of spatial constraints for interactive manipulation (Figure 3). Pinning and relative constraints are imposed on motion paths, while end-effector constraints can be imposed on any part of the character's body.

- **Pinning:** The position and direction of a character at any frame i may be constrained with respect to a global, reference coordinate system. Given a position constraint $(c_x, c_y) \in \mathbb{R}^2$, the constraint equation $\mathbf{p}_i = (c_x, c_y)$ is trivially linear with respect to a series of $\mathbf{p}_i$'s. Similarly, the tangent direction of a motion path can be constrained linearly such that $\mathbf{p}_{i+1} - \mathbf{p}_{i-1} = (d_x, d_y)$ for any direction $(d_x, d_y) \in \mathbb{R}^2$.
- **Relative:** The position and direction of character α at frame i can also be constrained with respect to a body, local coordinate system of another character β at frame j. Given relative position $(c_x, c_y) \in \mathbb{R}^2$ and relative direction $(d_x, d_y) \in \mathbb{R}^2$, constraint equations are

$$\mathbf{p}_i^\alpha = \mathbf{p}_j^\beta + c_x(\mathbf{p}_{j+1}^\beta - \mathbf{p}_{j-1}^\beta) + c_y R(\mathbf{p}_{j+1}^\beta - \mathbf{p}_{j-1}^\beta) \quad (1)$$

$$\mathbf{p}_{i+1}^\alpha - \mathbf{p}_{i-1}^\alpha = d_x(\mathbf{p}_{j+1}^\beta - \mathbf{p}_{j-1}^\beta) + d_y R(\mathbf{p}_{j+1}^\beta - \mathbf{p}_{j-1}^\beta). \quad (2)$$

- **End-effector:** This category includes any type of constraints that can be handled by a standard inverse kinematics solver. Typically, the position and orientation of any body part of a character at any frame may be constrained with respect to either a reference coordinate system or a local coordinate system of another character/object.

A constraint of any type can be imposed either on a single time instance (called *instant constraints*) or on a time interval (called *variational constraints*). For example, consider two characters walking towards a chair, picking it up from opposite sides, and together carrying the chair (Figure 3). To edit this collaborative, synchronized motion, we specify relative constraints for the characters to stay at the correct locations relative to the chair and end-effector constraints for their hands to remain on the chair while carrying the chair. In the example, both relative and end-effector constraints must be variational.

Our system allows the user to decide how precisely each constraint must be satisfied. Soft constraints are maintained in a least squares sense, while hard constraints are enforced precisely. A variational constraint is approximated by a series of soft constraints. The series of coordinate values of multiple paths concatenate to form a long vector $\mathbf{p} = (\mathbf{p}_0^\alpha\ \mathbf{p}_1^\alpha \cdots \mathbf{p}_0^\beta\ \mathbf{p}_1^\beta \cdots)^\top$. A collection of path constraints poses a linear system of equations in terms of vector $\mathbf{p}$. Let $B_s\mathbf{p} = \mathbf{b}_s$ and $H_s\mathbf{p} = \mathbf{h}_s$ be soft and hard spatial constraints, respectively.

3.2 Temporal Constraints

The motion clip is a sequence of motion frames usually sampled at fixed frame rate. A collection of motion clips are aligned along multiple time tracks and strung together to synthesize multiple character interaction. Each time track governs the timing of a single character motion. The timing in each motion clip is locally parameterized by frame index i that maps to $t_i \in \mathbb{R}$ in synchronized reference time. Three types of temporal constraints are provided to time-warp motions in a single time track and synchronize multiple character motions across time tracks.

- **Absolute time:** The timing of a character motion at any frame i may be constrained with respect to reference time. Given an absolute time $c_t \in \mathbb{R}$, the constraint equation is $t_i = c_t$.
- **Duration:** The duration of a motion sequence between frame i and frame j can be specified by a constraint, which is formulated as $t_j - t_i = d_t$. Duration $d_t \in \mathbb{R}$ is a scalar value.
- **Synchronization:** The timing of two characters can be synchronized. The constraint equation is $t_i = t_k + e_t$, where t_i is the timing of a character at frame i, t_k is the timing of another character at frame k, and time discrepancy e_t is a constant scalar value. If $e_t = 0$, two character motions must be synchronized exactly at frame i and frame k in the reference time.

Temporal constraints can also be specified as either soft or hard. Let $B_t\mathbf{t} = \mathbf{b}_t$ and $H_t\mathbf{t} = \mathbf{h}_t$ be soft and hard temporal constraints, respectively.

4 Laplacian Motion Editing

4.1 Spatial Path Editing

In this section, we incorporate user-defined constraints into the as-rigid-as-possible curve editing formulation [Igarashi et al. 2005] and discuss how to deal with degenerate cases. Given a collection of synchronized motion paths, we want to modify the paths to meet user-specified constraints by solving two least squares problems sequentially. The solution of the first problem generates an intermediate result that satisfies user constraints while preserving the detailed features of the original paths as much as possible. The result is obtained by solving the second least squares problem that compensates for undesirable scaling artifacts in the intermediate result.

Shape-Preserving Manipulation. A point $\mathbf{p}_i$ on a motion path can be represented with respect to its neighbors such that

$$\mathbf{p}_i = \mathbf{p}_{i-1} + x_i(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) + y_i R(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}), \qquad (3)$$

where (x_i, y_i) are the local coordinates of $\mathbf{p}_i$. The first phase of the algorithm intends to minimize the distortion in local coordinates while satisfying user constraints. To do so, we expect that equation 3 holds for a deformed path $\bar{\mathbf{p}} = \{\bar{\mathbf{p}}_i\}$ with the same local coordinates at every point. This condition can be formulated as a linear system $A^k\bar{\mathbf{p}}^k = \mathbf{0}$ for each motion path k. Note that the right hand side of the equation is zero, because equation 3 does not have a constant term. Concatenating linear systems for multiple motion paths and user-defined soft constraints forms an over-determined linear system $M_s\bar{\mathbf{p}} = \mathbf{m}_s$, that is,

$$M_s\bar{\mathbf{p}} = \left(\begin{array}{cccc} A^0 & 0 & \cdots & 0 \\ 0 & A^1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \cdots & A^k \\ \hline & & B_s & \end{array}\right)\bar{\mathbf{p}} = \left(\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \mathbf{b}_s \end{array}\right) = \mathbf{m}_s \qquad (4)$$

subject to user-defined hard constraints $H_s\bar{\mathbf{p}} = \mathbf{h}_s$. Using the pseudo-inverse of M_s and Lagrange multipliers λ_s, the objective of shape-preserving manipulation is minimizing energy function

$$E_{shape} = \min_{\bar{\mathbf{p}}} \|\frac{1}{2}\bar{\mathbf{p}}^\top M_s^\top M_s\bar{\mathbf{p}} - \bar{\mathbf{p}}^\top M_s^\top \mathbf{m}_s + \left(H_s\bar{\mathbf{p}} - \mathbf{h}_s\right)^\top \lambda_s\| \qquad (5)$$

Differentiating with respect to $\bar{\mathbf{p}}$ then λ_s leads to the augmented system

$$\left(\begin{array}{cc} M_s^\top M_s & H_s^\top \\ H_s & 0 \end{array}\right)\left(\begin{array}{c} \bar{\mathbf{p}} \\ \lambda_s \end{array}\right) = \left(\begin{array}{c} M_s^\top \mathbf{m}_s \\ \mathbf{h}_s \end{array}\right). \qquad (6)$$

The result $\bar{\mathbf{p}}$ at the first phase is obtained by solving the unconstrained system, which is sparse and thus can be solved efficiently using a sparse LU solver.

Scale Compensation. The second phase adjusts the intermediate result $\bar{\mathbf{p}}$ to remove scaling artifacts. Let $\bar{\mathbf{v}}_i = (\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i)\frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\|\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i\|}$ be a vector between two successive points $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{p}}_{i+1}$ scaled to its original length in the undeformed path. At the second phase, the scale-preserving path $\{\hat{\mathbf{p}}_i\}$ bends or flattens to satisfy user constraints. Therefore, the scale-preservation objective leads to a linear equation

$$\hat{\mathbf{v}}_i - \hat{\mathbf{v}}_{i-1} = \bar{\mathbf{v}}_i - \bar{\mathbf{v}}_{i-1}, \qquad (7)$$

where $\hat{\mathbf{v}}_i = \hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i$ (see [Igarashi et al. 2005] for details). This objective forms a collection of linear systems $\hat{A}^k\hat{\mathbf{p}}^k = \hat{\mathbf{a}}^k$ for path k. Concatenating the linear systems for multiple paths and the user-defined soft constraints forms an over-determined linear system $\hat{M}_s\hat{\mathbf{p}} = \hat{\mathbf{m}}_s$ subject to hard constraints. Replacing M_s with $\hat{M}_s$ in equations 5 and 6, we can derive the objective function E_{scale} for scale compensation and its corresponding augmented system.

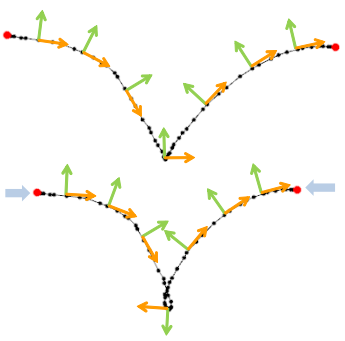

Figure 4: Tangent flipping. Gentle squeezing of a motion path (top) could cause the tangent direction to flip at the middle of the path (bottom).

4.2 Handling Degenerate Cases

We have to deal with two types of degenerate cases. The first type of degeneracy is a stationary path. Some human motion data look natural after large deformation, while some others are fragile even for small deformation. For example, walking motions are usually resilient to bending of moving paths, so can be deformed in a wide range of curvature angles. Conversely, manipulating the path of a stationary motion could make the character hover around unnaturally with slippery feet. A sub-path (a part of a motion path) is *stationary* if the average velocity on the path is within a threshold or both feet are firmly planted on the ground. Our system treats stationary sub-paths as rigid segments in such a way that the sub-paths are not deformed during interactive manipulation. We can do that simply by putting the corresponding shape-preserving constraints in equation 3 into a collection of hard constraints.

The second type of degeneracy is tangent flipping (Figure 4). Small deformation of a motion path could cause a tangent direction to flip. We identify tangent flipping by checking angles between successive tangent vectors. The sudden change of tangent direction indicates tangent flipping. Precisely, we consider tangent vectors $\mathbf{t}_i$ and $\hat{\mathbf{t}}_i$ of undeformed and deformed paths. The angle $\phi_i = \angle(\mathbf{t}_i, \hat{\mathbf{t}}_i)$ between them provides a useful measure. Our system reports tangent flipping, if $\|\phi_{i+1} - \phi_i\|$ is beyond a user-provided threshold for any i. We determine a new tangent vector at that singular point by linearly interpolating nearby tangent vectors.

4.3 Full-Body Refinement

The character's full-body motion along the deformed path requires a final touch-up to enforce user-defined end-effector constraints and avoid foot-sliding artifacts. a foot-ground contact is identified automatically if the foot is sufficiently close to the ground and its veloc-

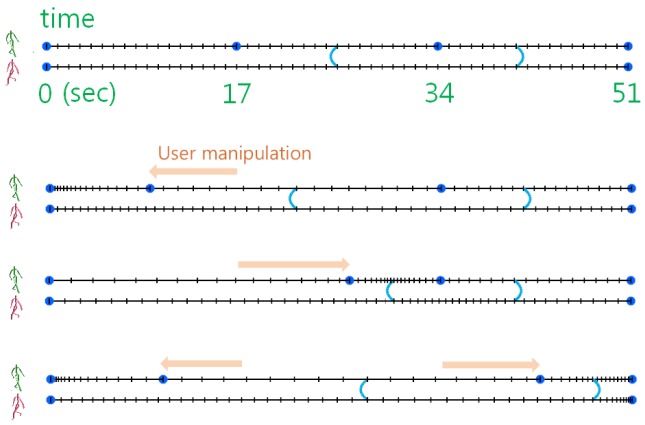

Figure 5: Interactive time warping of two synchronized time tracks. The original uniform time sequences at the top are interactively manipulated to generate smooth, synchronized time-warps. The user manipulates only one track and the other track is warped accordingly.

ity is below a threshold. The ground contact constraints and end-effector constraints feed into a hybrid inverse kinematics solver and hierarchical displacement mapping [Lee and Shin 1999] to adapt full-body motions.

While root path editing can be formulated as a very efficient linear formulation, end-effector constraints involve non-linear equations and thus require an iterative inverse kinematics solver. The combination of root path editing and IK-based clean up is a pragmatic solution that makes our system fast enough for practical applications. To make this combination work, end-effector constraints are usually supplemented with root constraints. For example, to create a character carrying a box, we first constrain the root to stay near the box (using a relative position/orientation constraint) and then constrain the character's hands to touch the box (using end-effector constraints). The root-box constraint keeps the box move together with the character while the character moving, and the end-effector constraints refine the character's pose at every time frame to keep the hands stay the box. It is a natural combination to use root constraints to specify the global location/orientation of the character and end-effector constraints to adjust the character's pose in small scales.

4.4 Interactive Time Warping

We also use the Laplacian curve manipulation method for interactive time warping. The user of our system is allowed to specify arbitrary temporal constraints. Then, the system computes a one-dimensional smooth time-warp to meet user-defined constraints (Figure 5). Input motion data are uniformly sampled such that $t_{i+1} - t_i = h$, where h is the sampling time step. Time warping should be carried out, while preserving as much uniformity as possible. This condition can be formulated as a linear system $T^k \bar{\mathbf{t}}^k = \mathbf{h}$ for each motion path k, where $\bar{\mathbf{t}}^k = (t_i^k)$ is the time-warp of the k-th motion path. Matrix T^k is banded and only two diagonals have non-zero values. $\mathbf{h} = (h \cdots h)^\top$ is a constant vector. Concatenating linear systems for multiple motion paths and soft temporal constraints $B_t \bar{\mathbf{t}} = \mathbf{b}_t$ forms an over-determined linear system $M_t \bar{\mathbf{t}} = \mathbf{m}_t$, that is,

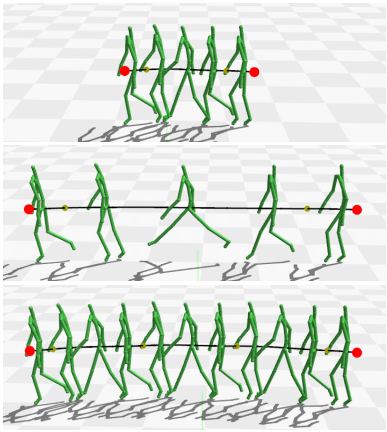

Figure 6: Discrete motion editing. (Top) input walking motion. (Middle) Laplacian path editing necessarily entails longer strides. (Bottom) Discrete motion editing introduces extra steps to cope with excessive stretching.

$$M_t \bar{\mathbf{t}} = \left(\begin{array}{cccc} T^0 & 0 & \cdots & 0 \\ 0 & T^1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \cdots & T^k \\ \hline & & B_t & \end{array} \right) \bar{\mathbf{t}} = \left(\begin{array}{c} h \\ h \\ \vdots \\ h \\ \mathbf{b}_t \end{array} \right) = \mathbf{m}_t \quad (8)$$

subject to hard temporal constraints $H_t \bar{\mathbf{t}} = \mathbf{h}_t$. Replacing M_s with M_t in equations 5 and 6, we can derive the objective function E_{time} for time warping and its corresponding augmented system.

$$\left(\begin{array}{cc} M_t^\top M_t & H_t^\top \\ H_t & 0 \end{array} \right) \left(\begin{array}{c} \bar{\mathbf{t}} \\ \lambda_t \end{array} \right) = \left(\begin{array}{c} M_t^\top \mathbf{m}_t \\ \mathbf{h}_t \end{array} \right). \quad (9)$$

Time warping requires only the first phase of the path editing algorithm. The scaling artifact occurs usually in a direction orthogonal to the manipulator's dragging direction. Therefore, scale compensation is not required for one-dimensional curves. The second phase of the algorithm is rather harmful. It causes the frames in the middle of motion data to scale more rapidly than the frames at the beginning and end of the data.

Time never goes backward and thus time-warp should increase monotonically. Undue time warping could result in time flipping or excessive slowdown of motion sequences (Figure 5). We address this problem by imposing inequality constraints $h_{min} < t_{i+1} - t_i < h_{max}$ for each i. The linear system does not easily incorporate inequality constraints. Instead, we impose the inequality constraints in an iterative manner. We first solve the linear system in equation 9 without inequality constraints and then insert equality constraints ($t_{i+1} - t_i = h_{min}$ if $t_{i+1} - t_i < h_{min}$ for any i, and $t_{i+1} - t_i = h_{max}$ if $t_{i+1} - t_i > h_{max}$) where the inequality constraints are violated. We repeat this process until all inequality constraints are satisfied.

5 Discrete Motion Editing

The goal of our discrete motion editing is to allow discrete transforms to occur appropriately and seamlessly, while manipulating a motion path. The example in Figure 6 motivates the use of discrete transforms in interactive motion editing. Without discrete

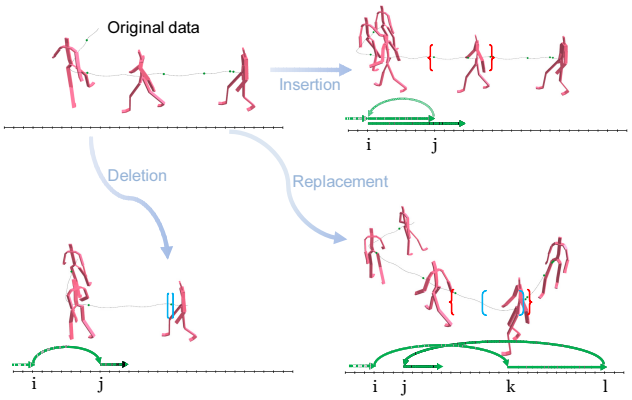

Figure 7: Three types of discrete transformations.

transformations, stretching a walking motion (using two pinning constraints) results in unnaturally longer strides. Our discrete optimization procedure automatically introduces extra steps to avoid making excessively long strides.

Predictability and controllability are key design concepts of interactive user interfaces. We avoid re-synthesizing the entire path using path planning or state-space search techniques during interactive manipulation. A totally new re-planned path suddenly suggested by a smart planner can be often confusing for the user. Instead, our user interface is intended to make small, predictable discrete changes at each moment so that discrete editing can be seamlessly incorporated into continuous path editing. Furthermore, our incremental update approach is computationally more efficient than re-synthesizing the path from scratch.

The key to our discrete path editing is allowing such structural changes to happen seamlessly during Laplacian path editing. To do so, we discuss how to identify potential discrete transformations, how to evaluate transformed paths, and how to incorporate discrete optimization into interactive motion editing interfaces while retaining the interactive performance.

5.1 Discrete Transformations

We build motion graphs [Lee et al. 2002] of input motion data to identify plausible structural changes to motion data efficiently. The nodes of a motion graph correspond to motion frames and its edges correspond to connecting transitions between frames. The motion graph can be automatically constructed from input motion data by identifying similar frames and creating transitions between similar frames. We use a similarity measure presented by Lee and Lee [2006] to decide if two motion frames are sufficiently similar. The measure considers various features of motion. These include joint angles, root height, root orientation, the velocities of joints, root translation, and root rotation. Motion frames are temporally coherent and thus many connecting transitions have nearby start and end frames. We favor the best likely transitions among many similar transitions by selecting local maxima in the transition matrix. All the other transitions are pruned to produce a sparse set of connecting transitions.

We consider three types of discrete transformations: insertion, deletion, and replacement (Figure 7). Let frame i and frame j be similar frames that allow connecting transition from one frame to the other, where $i < j$. Traversing the forward transition from i to $j+1$ results in the deletion of frames between $i+1$ and j, while traversing the backward transition from j to $i+1$ leads to duplicating frames between $i+1$ and j. Duplication is a special type of insertion. In general, insertion requires two transitions from i to $j+1$ and then from k to $i+1$, for $j < k$, which leads to copying frames from $j+1$ to k and inserting these frames between frame i and $i+1$. Similarly, replacement using two transitions from i to $k+1$ and then from l to $j+1$, for $i < j$ and $k < l$, leads to replacing frames between $i+1$ and j with frames between $k+1$ and l.

Algorithm 1: Discrete Path Editing

$\mathbb{P}$: A set of motion paths;
$\mathbb{P}'$: A set of deformed, time-warped paths;
$\mathbb{C}$: A set of spatial and temporal constraints;

```
1  while motions are manipulated do
2      if ℂ is updated then
3          ℙ_new ← ℙ;
4          ℙ'_new ← LaplacianEditing ( ℙ, ℂ );
5          E_new ← DeformationEnergy ( ℙ_new, ℙ'_new, ℂ );
6          foreach ℙ_c ∈ DiscreteTransform ( ℙ ) do
7              ℙ'_c ← LaplacianEditing ( ℙ_c, ℂ );
8              E_c ← DeformationEnergy ( ℙ_c, ℙ'_c, ℂ );
9              if E_c < E_new then
10                 ℙ_new ← ℙ_c;
11                 ℙ'_new ← ℙ'_c;
12                 E_new ← E_c;
13         ℙ ← ℙ_new;
14         ℙ' ← ℙ'_new;
```

5.2 Discrete Optimization

Our system applies an appropriate discrete transformation, if needed, to a motion path in immediate response to user manipulation. We allow only one motion path to be transformed for each update of user constraints. Only one discrete transformation is allowed to occur at any time instance to avoid abrupt structural changes. Once user (either spatial or temporal) constraints are updated, our system enumerates all possible discrete transformations and evaluates each transformation with respect to user constraints. Then, the best transformation is selected and deformed to meet user constraints precisely using Laplacian editing techniques (Algorithm 1). This greedy approach is appropriate for interactive manipulation because its responses are immediate and predictable.

Deformation Energy. We need to determine how well each transformed path fits user-defined constraints (lines 6–8). To do so, we run Laplacian (both path and time-warp) editing on the transformed path to meet user constraints and then evaluate its deformation energy

$$E = E_{shape} + E_{scale} + E_{time} + c_s|L_s(\mathbf{P}') - L_s(\mathbf{P})| + c_t|L_t(\mathbf{P}') - L_t(\mathbf{P})| \quad (10)$$

where $\mathbf{P}$ is the original motion path and $\mathbf{P}'$ is a transformed, deformed, time-warped path. E_{shape}, E_{scale} and E_{time} are the deformation energy of shape-preservation, scale-compensation and time-warping, respectively. L_s is spatial path length and L_t is temporal duration. The first three terms penalize path deformation by Laplacian motion editing and the last two terms penalize lengthening and shortening of motion paths by discrete transformation. c_s and c_t weigh the path deformation against discrete transformation.

Pruning Discrete Transformations. We often have too many potential discrete transformations to enumerate and evaluate comprehensively at interactive rates. It is important to prune as many unsuitable discrete transformations as possible before applying Laplacian editing. Our pruning rules take account of the interactivity of motion editing and user constraints:

- **Duration.** We prune a transformation if the interval of motion frames to be deleted/inserted by the transformation is either too short or too long. Discrete transformations of very short interval tend to occur too frequently, while transformations of very long interval cause abrupt visual changes in motion manipulation. In our experiments, transformations of 0.3 to 2.0 seconds are accepted.

- **Enclosing.** One transformation encloses the other transformation if the interval of motion frames to be deleted/inserted by the former transformation includes the interval of motion frames to be affected by the latter transformation. We prune a transformation if it encloses another transformation. The rationale for this pruning rule comes from the interactivity of manipulation. We assume that user constraints are updated continuously as the user drags a manipulator. Therefore, the enclosed transformation will be invoked if needed and the enclosing transformation can never occur.

- **Constraints.** A transformation should not remove any user-defined constraints. Specifically, transformations that delete or replace any frame curbed by instant constraints are pruned. Motion frames curbed by variational constraints can be deleted, replaced with other frames, and allow other frames to be inserted by discrete transformations as long as the boundary frames where variational constraints initiate and terminate remain intact.

- **Locality.** Discrete changes could occur unexpectedly far away from the point/time the user manipulates. We addressed this problem by providing the user with simple user interfaces to enable and disable discrete transformations. The user can selectively enable specific transformation types, characters, time zones, spatial regions that permit discrete changes.

Acceleration Technique. The performance bottleneck of discrete path editing is the evaluation of potential discrete transformations (lines 7–8). The evaluation of each transformation demands solving three large, sparse linear systems for continuous Laplacian editing. The size of linear systems depends on the total number of frames in motion paths. We can trade off the accuracy of evaluation for further performance gain. To do so, we subsample motion paths to construct smaller linear systems. When subsampling, user-defined constraints and potential discrete transformations should be retained. This requirement leads to non-uniform subsampling of motion frames. Laplacian editing with subsampled paths gives a quite reasonable approximation of the deformation energy of the original, full-frame paths. Evaluating the deformation energy does not require end-effector constraints to be satisfied. Hierarchical displacement mapping is applied once at the end of the algorithm.

6 Results

The motion data used in our experiments were captured from a Vicon optical system with sixteen cameras at the rate of 120 frames/second and then sub-sampled to 30 frames/second.

Our system is fast enough to manipulate dozens of characters at interactive rates. Table 1 shows performance statistics. Discrete editing is performed with subsampled paths. The subsampling ratio is $1/25$. The subsampled path includes extra frames to retain user-defined constraints and potential discrete transformations. The extra frames include frames with instant constraints, frames that permit in-coming/out-going transitions for discrete transformation, and their immediate neighboring frames. The last column of the table shows the computation time for Laplacian editing without subsampling. Roughly speaking, Laplacian editing of one thousand synchronized frames takes less than one hundred milliseconds. The computation includes solving three sparse linear systems and enforcing end-effector constraints.

Example	Motion data	# of frames	Path constraints		Discrete changes	Computation time (ms)	
			Soft	Hard		Discrete	Laplacian
High five	high five 1	142 (105)	10	30	7 (4)	45.8	58.9
	high five 2	105 (105)	11	30	4 (4)		
	high five 3	253 (105)	10	30	16 (4)		
	high five 4	105 (105)	10	30	4 (4)		
Chair	carry side 1	322 (210)	27	98	10 (4)	68.8	80.2
	carry side 2	322 (210)	22	49	10 (4)		
	sit down	241 (206)	8	105	7 (4)		
Adventure	duck walk 1	356 (356)	10	35	5 (6)	123.1	332.4
	push 1	140 (140)	9	53	4 (4)		
	dodge 1	370 (370)	9	53	7 (7)		
	shrink back 1	251 (251)	10	55	7 (7)		
	duck walk2	140 (140)	9	35	10 (6)		
	push2	140 (140)	9	53	4 (4)		
	dodge2	370 (370)	9	53	7 (7)		
	shrink back 2	251 (251)	10	55	7 (7)		
Stacking	pick up 1	368 (257)	12	87	9 (4)	112.7	282.5
	pick up 2	220 (257)	10	87	2 (4)		
	carry 3	250 (250)	9	82	3 (4)		
	carry 4	210 (250)	9	82	2 (4)		
	put down 5	298 (226)	8	110	7 (4)		
	put down 6	262 (226)	8	110	7 (4)		
	push move 1	216 (144)	11	57	13 (4)		
	push move 2	168 (144)	13	57	7 (4)		
	carry side 1	307 (179)	11	70	9 (4)		
	carry side 2	211 (179)	10	70	5 (4)		
Relay	relay box 1	217 (178)	17	70	4 (4)	82.8	206.1
	relay box 2	208 (171)	9	51	4 (4)		
	relay box 3	217 (178)	19	70	3 (3)		
	relay box 4	171 (171)	10	51	4 (4)		
	relay box 5	217 (178)	14	70	7 (4)		
	relay box 6	171 (171)	9	51	4 (4)		
	side step 1	178 (178)	14	38	3 (3)		
	jump 1	221 (221)	10	49	5 (3)		
	side step 2	178 (178)	14	38	3 (3)		
	jump 2	176 (176)	10	49	3 (3)		

Table 1: Performance statistics. The sixth column shows the number of plausible discrete transformations after pruning. The numbers in parenthesis are counted at the initial state before the motions are edited. The other numbers are counted or measured after the editing is finished. The seventh column shows the computation time required to enumerate and evaluate all plausible discrete transformations. The last column is the computation time for Laplacian motion editing. All computation time was measured by averaging over 40 iterations.

Our system specifies hard constraints automatically to rigidify stationary sub-paths in motion data. All user-defined path constraints are soft. The variational path constraint is approximated by a series of soft constraints that are sampled every tenth frame over the constrained interval.

Our discrete optimization deals with spatial manipulation and time warping in a uniform way. Consider motion data for which a subject walked, stopped, idled, and walked again. We can pick both ends of the motion to lengthen its spatial path. By the stretching of the motion path, our discrete optimization would select a cycle of ambulatory motion and duplicate the cycle. On the other hand, we can use two time manipulators to stretch the motion in time. Then, our system would select an idling motion to duplicate. This choice allows the duration of the motion to be stretched, while the length of its spatial trajectory is unchanged.

The continuous and discrete editing capabilities of our system is demonstrated with five examples (Figure 3 and Figure 8). The high five example shows manipulating four characters simultaneously. Their path lengths change during manipulation, but the timing of slapping is precisely synchronized. In the adventure example, we have eight characters interacting sequentially. The pink character

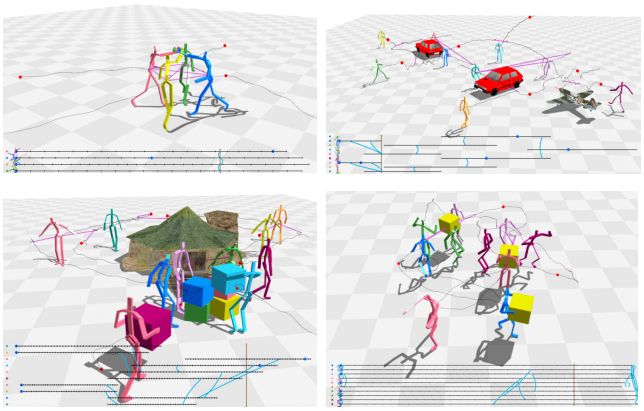

Figure 8: Synchronized multi-character motion editing. (Top to bottom, left to right) High five, adventure, stacking boxes, and carrying boxes in relay.

dodges, walks, and pushes the yellow character, who then jumps over the green character. The entire scene can be manipulated and stretched, while maintaining the interaction between characters. The box stacking examples shows ten characters dragging, pulling, carrying boxes to make a stack of boxes. The relay example shows a group of four characters. Two characters carry boxes in relay and the other two characters just hang around to disturb. We duplicated the group of four characters and splice them to create a long relay of carrying boxes shown in Figure 1.

The editing process for creating each example is as follows. For the box stacking example, what we actually did is:

- Load motion clips that recorded our subject carrying/pushing/pulling boxes in various ways.
- Annotate each motion clip with the timing of touching the box and the relative location/orientation of the root, the left hand, and the right hand with respect to the coordinate system of the box at the moment. We used an inverse kinematics solver and a simple user interface to specify the relative locations and orientations in three-dimensional space.
- Specify the timing of passing boxes between characters. Manipulate the motion paths and adjust the height of boxes to make a stack of boxes.

All examples in the video except the spiral relay example were created through similar steps and took less than 10 minutes to reproduce from scratch given a set of motion clips and annotations. For the relay example in Figure 1, we hard corded a procedure to automatically duplicate characters/constraints and computed the result off-line.

7 Discussion

In this paper, we explored multiple aspects (continuous, discrete, space, and time) of interactive motion editing in a unified framework. Continuous motion editing and non-sequential discrete editing are combined seamlessly in an interactive system. This deals with spatial path editing and time warping uniformly. Our method is simple, intuitive, and versatile. We demonstrated how multiple character motions are manipulated, spatially-aligned, and temporally-synchronized in an interactive editing system.

Our motion editing method focused on manipulating two-dimensional motion paths. It is not that generalizing the method for three-dimensional paths is difficult. It is just that we were unable to acquire motion data that exhibit large height variations because of the limited space of our motion capture studio. Small jumping and falling down can be captured. However, such motions are governed by gravity and thus manipulating them in the gravity direction would result in unnatural, physics-violating motion. Small height variations that does not involve free-fall under gravity can be handled by using hierarchical displacement mapping [Lee and Shin 1999]. Three-dimensional path editing would attract attention if we can capture swimming or flying motion.

Motion synthesis based on motion graphs sometimes suffers from exponentially-growing computational costs for searching through richly-connected motion graphs. Continuous nature of interactive dragging allows us to take our incremental update method instead of time-consuming re-synthesis approaches. With appropriate pruning strategies, the number of potential discrete transformations we have to examine at every time instance scales linearly with respect to the size of motion data to be manipulated.

Consider that the user manipulates motion data by dragging a handle and then drags the handle backwards to put it back to its start location. The path manipulation is *reversible* if the motions trace back to its original state. Similarly, the path manipulation is *trajectory-independent* if the results depend only on the final destination of mouse dragging regardless of its trajectory to the destination. The reversibility and trajectory-independence are closely related to the optimality of path manipulation. The globally optimal strategy is supposed to satisfy both properties. Our discrete editing method selects the locally best choice incrementally and, in general, not globally optimal. We trade off the optimality against interactive runtime performance and better frame-to-frame coherency. Therefore, our method is neither reversible nor trajectory-independent.

Our system employs linear Laplacian formulation for modeling multi-character interaction. The linear formulation has a big advantage of allowing interactive performance. However, it also prevents us from using general forms of constraints. For example, collision avoidance necessitates inequality constraints. Modeling sophisticated social/personal relations and close physical contacts require non-linear equations. Though we chose to use the Laplacian curve deformation method for its simplicity and efficiency, our system can take any other similar techniques developed for as-rigid-as-possible deformation. Probably, a more flexible non-linear optimization method would allow us to model high-level social behavior and inter-personal interaction.

Even with efficient linear formulation and acceleration techniques, our motion editing system can deal with only dozens of characters at interactive rates. Recent feature films and video games require the capability to synthesize and manipulate the motion of many more characters for crowd simulation. The computational bottleneck of our system is its linear system solver. Though the solver [UMFPACK] we used is highly-optimized for solving sparse linear systems, parallel solvers running on GPU or multi-core CPU might achieve further improvements.

Acknowledge

We would like to thank all the members of the Movement Research Laboratory for their help. We are especially grateful to Taesoo Kwon for his valuable advice and sharing his Laplacian solver, and Kyungyong Yang for editing the accompanying video. This work was supported by the IT R&D program of MKE/MCST/IITA 2008-F-033-02.

References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3, 402–408.

CHAI, J.-X., AND HODGINS, J. K. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics (SIGGRAPH 2008) 26*, 3.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proceedings of SIGGRAPH 98*, 33–42.

GLEICHER, M. 2001. Motion path editing. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, 195–202.

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proceedings of Symposium on Interactive 3D Graphics and Games*, 129–136.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3.

HO, E. S., AND KOMURA, T. 2009. Character motion synthesis by topology coordinates. *Computer Graphics Forum (Eurographics 2009*.

HSU, E., GENTRY, S., AND POPOVIĆ, J. 2004. Example-based control of human motion. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, 69–77.

HSU, E., DA SILVA, M., AND POPOVIĆ, J. 2007. Guided time warping for motion editing. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 45–52.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1134–1141.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 473–482.

KWON, T., CHO, K.-S., PARK, S. I., AND SHIN, S. Y. 2008. Two-character motion analysis and synthesis. *IEEE Transactions on Visualization and Computer Graphics 14*, 3, 707–720.

KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. 2008. Group motion editing. *ACM Transactions on Graphics (SIGGRAPH 2008) 27*, 3.

LEE, J., AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, 79–87.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, 39–48.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 491–500.

LEE, K. H., CHOI, M. G., AND LEE, J. 2006. Motion patches: Building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (SIGGRAPH 2006) 26*, 3, 898–906.

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

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

LO, W.-Y., AND ZWICKER, M. 2008. Real-time planning for parameterized human motion. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*.

MCCANN, J., AND POLLARD, N. S. 2007. Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3.

MCCANN, J., POLLARD, N. S., AND SRINIVASA, S. 2006. Physics-based motion retiming. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1062–1070.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3, 501–508.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3, 106.

SHIN, H. J., AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds (CASA 2006) 17*, 3-4, 219–227.

SHIN, H. J., AND OH, H. S. 2006. Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 291–298.

SHUM, H., KOMURA, T., SHIRAISH, M., AND YAMAZAKI, S. 2008. Interaction patches for multi-character animation. *ACM Transactions on Graphics (SIGGRAPH Asia 2008) 27*, 6.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., AND ALEXA, M. 2004. Laplacian surface editing. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Geometry Processing*, 175–184.

SULEJMANPASIC, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 1, 165–179.

TAK, S., AND KO, H.-S. 2005. A physically based motion retargeting filter. *ACM Transactions on Graphics 24*, 1, 98–117.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous user control. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3.

UMFPACK. Library for solving unsymmetric sparse linear systems, http://www.cise.ufl.edu/research/sparse/umfpack/.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, 89–96.