Computers & Graphics 47 (2015) 59–67

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Motion retiming by using bilateral time control surfaces ☆

Innfarn Yoo *, Michel Abdul Massih, Illia Ziamtsov, Raymond Hassan, Bedrich Benes

Purdue University, West Lafayette, IN, United States

ARTICLE INFO

Article history:
Received 31 July 2014
Received in revised form
4 October 2014
Accepted 9 November 2014
Available online 28 November 2014

Keywords:
Motion
Motion retiming
Motion editing
Bilateral time control surface
BTCS
Animation interface

ABSTRACT

Motion retiming is an important tool used to edit character animations. It consists of changing the time at which an action occurs during an animation. One example might be editing an animation of a person playing football to change the precise time at which the ball is kicked. It is, nevertheless, a non-trivial task to retime the motion of a set of joints, since spatio-temporal correlation exists among them. It is especially difficult in the case of motion capture, when there are forward kinematic keys on every frame, to define the motion. In this paper, we present a novel approach to motion retiming that exploits the proximity of joints to preserve the motion coherence when a retiming operation is performed. We introduce the bilateral time control surface (BTCS), a framework that allows users to intuitively and interactively retime motion. The BTCS is a free-form surface, located on the timeline, that can be interactively deformed to move the action of a particular joint to a certain time, while preserving the coherency and smoothness of surrounding joints. The animation is retimed by manipulating successive BTCSs, and the final animation is generated by resampling the original motion by time spans defined by the BTCSs.

1. Introduction

Targeting an action or movement of an existing character animation to a particular time can be achieved by *motion retiming*. Retiming animation in an intuitive way poses a difficult problem because of the need to maintain the spatial and temporal coherence of the different parts of the animated character. For example, if the hand should move to a certain part of the scene at a specific moment, retiming the hand motion requires one to take into account the complex relationship of the hand with the elbow, the shoulder, and possibly other parts of the body.

The importance of this problem is demonstrated through a vast body of previous works (see Section 2). A common strategy used in character animation for the preservation of joint relationships is to employ inverse kinematic controls. Such controls can ease the retiming process by allowing multiple joints to be manipulated or retimed by keying a single controller. The retiming has also been addressed through the use of time warping, sketching, actor performances, and via physics-based simulations. However, simultaneously providing good user control and, at the same time, maintaining coherence between independently retimed joints and the rest of the body are still an open problem because a strong spatiotemporal correlation of motion between individual joints of an animated character exists.

We observe that the spatial proximity of joints translates into temporal coherence of their motion. When retiming the motion of a joint, it is possible to enhance keyframing with global coherence by constraining the other joints to be within a certain distance from the edited point. This way, we could make sure that a certain joint is exactly in a certain location at the given time, but we could also maintain the coherence and the global characteristics of the dependencies of the rest of the joints.

We present a novel motion retiming approach that exploits the spatial proximity of joints to keep the temporal relationship between them when applying retiming operations. Our framework can be applied for retiming any motion sequence, it preserves globality and motion coherence, it preserves smoothness of the joint curves, and it is simple to control. To our knowledge, no other work exists that approaches the retiming or editing of motion by using the smoothness of a surface that intersects motion paths to quickly manipulate the time edits. The editing is based on manipulating smooth surfaces along the joint motion paths and joint topology to control the timing of motion. We call each of these surfaces a bilateral time control surface (BTCS). The retiming results from the deformation of the surface and its intersection with the motion paths of individual joints. We make use of the spatial continuity of the surface to ensure that the changes in the timing of a joint smoothly affect the timing of other joints in its topological proximity. Our key contributions are given as follows:

1. Introducing a new formulation for retiming motion.
2. An interactive and intuitive retiming interface without the need for modifying motion curves independently.

☆This article was recommended for publication by KangKang Yin.
* Corresponding author.

http://dx.doi.org/10.1016/j.cag.2014.11.001
0097-8493/

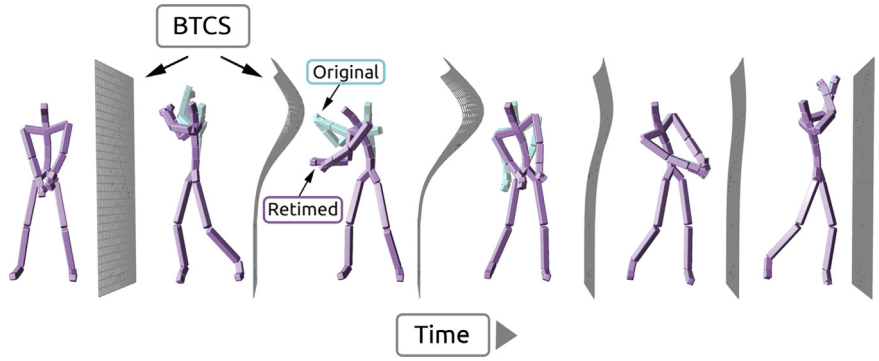

Fig. 1. Retimed golfing motion. The cyan character represents the original motion and the purple character represents the resulting motion. The surfaces shown in front of the characters represent the BTCS's for each of the frames. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

3. Bilateral filtering for ensuring that edits are applied within a joint's topological distance.
4. A new visualization technique of spatio-temporal motion information.

Fig. 1 shows six non-consecutive frames of a golfing motion retimed with our method. The character displayed in cyan color represents the original motion and the one displayed in purple represents the retimed motion. The surfaces shown in front of the characters are the BTCS's for each frame. The retimed animation can be edited in 90 s, and the final sequence is generated interactively.

2. Previous work

In this section, we discuss relevant literature describing motion retiming and animation interfaces.

Motion retiming provides the ability to change the time at which events happen during the animation or the whole duration of the motion. Most animation editing and synthesis methods perform motion retiming through time warping. Bruderlin and Williams apply signal processing to motion curves in [1]. Heck et al. [2] use time warping to synchronize motions from different sources, combined in one animation, and Rose et al. [3] use time warping to handle the motions in a canonical space. Witkin and Popovic [4] edit motions through warping and blending, and Lee and Shin [5] use time warping for making transitions between motion clips. McCann et al. [6] introduce physics-based motion retiming to achieve physically plausible motion after changing the motion duration. Hsu et al. [7] develop a method for changing the duration of a motion by using different amounts of warping along the motion depending on user provided constraints. Neff and Fiume [8] implement an animation system in which poses are defined by subsets of DOFs, and allow for intuitive time variation, or succession. Coleman et al. [9] also provide a method that allows to achieve successive activation of DOFs through their staggered poses. Our work is related to [8] and [9] in the sense that we allow for intuitive motion retiming of different joints simultaneously. Contrary to the previous work, our approach eases motion retiming by providing the user with the ability to concurrently edit the timing of different joints by transforming and deforming a control surface.

Intuitive *interfaces* are important for animators to edit input character motion. An intuitive way of generating and editing motion is through sketching interfaces. Thorne et al. [10] generate motion by letting the user sketch a character by selecting its armature and the path of the character's motion, and Guay et al. [11] allow the user to generate character poses for motion synthesis by drawing the line of action, mimicking the method that cartoonists follow for sketching the initial pose of their characters. Recently, Yoo et al. [12] stitch animation clips found from user sketches by searching in a motion database. Other methods for the intuitive design of animation interfaces include the work of Dontcheva et al. [13] that allows the users to act out the motion of subsets of joints as layers of an animation and then adds the layers together to obtain a final animation. Igarashi et al. [14] allow the user to generate character poses that are linked to positions on the 3D space of the character. For motion synthesis, the user moves a point around this space and the system interpolates between poses to yield a final animation. Kim et al. [15] introduce a new approach to controlling interactions of multiple characters via seamlessly connected motion patches. Wang et al. [16] recently provide a way to generate keyframes in between two poses based on an energy graph. Ho and Komura [17] tackle indexing and retrieving problems of two closely interacting motions using a topology-based method. Neff and Fiume [18] combine scripting, sketching, and direct user edits for quick prototyping and changes in a cyclic process of refinement until the animation is finalized. Lockwood and Singh [19] present an interface for synthesizing locomotion by following the “finger walking” of users.

Similar to our goal is the work of Terra and Metoyer [20] that allows the user to describe the motion timing by gesturing. Mukai and Kuriyama [21] present to the user an animation timeline that shows controls, such as icons, that can be dragged and dropped, as well as dropping a source motion on the timeline for editing the timing of the animation. Kim et al. [22] present an interface for manipulating space and time properties of multiple characters by dragging motion paths and timelines simultaneously.

Our approach differs from the works presented above with respect to the tool for manipulating the motion retiming. Instead of directly manipulating individual motion curves or exploring a reduced dimension space, we provide the user with a surface that represents an individual frame, which can be deformed and transformed to span many frames so that subsets of joints can be manipulated at the same time while keeping coherence on retiming operations.

3. Method

Fig. 2 shows the overview of our method. The input of our retiming framework is a sequence of motion data of a character, keyframes (a time-ordered set of 3D poses), and the output is the retimed animation (Fig. 3).

The user is first shown the path of each joint for the duration of the animation alongside the original motion (Fig. 2b). Then, the joint paths are projected onto planes whose normal is a pre-determined time direction. In our system, the time direction is initially set as the motion direction, and the user can manually select another direction.

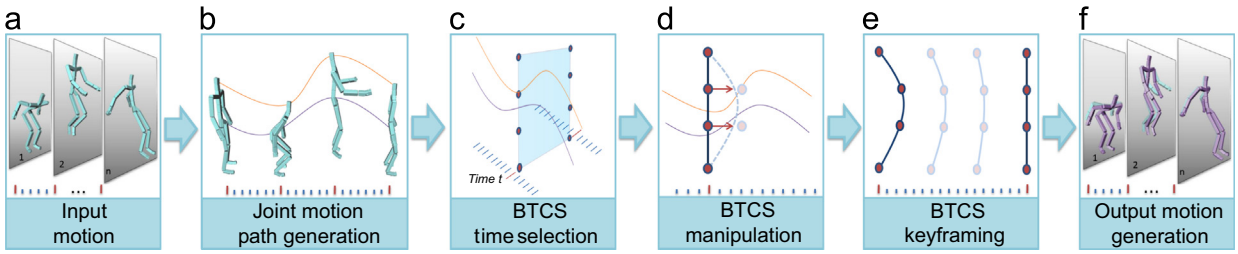

Fig. 2. Overview. The input of the framework is a character motion (a) from which the joint motion projected path curves are generated (b). The user sets a time control surface at a specific time for retiming (c) and adjusts it by manipulating control points (d). The user creates keyframes of time control surfaces at different times (e) and the system generates a new motion (f).

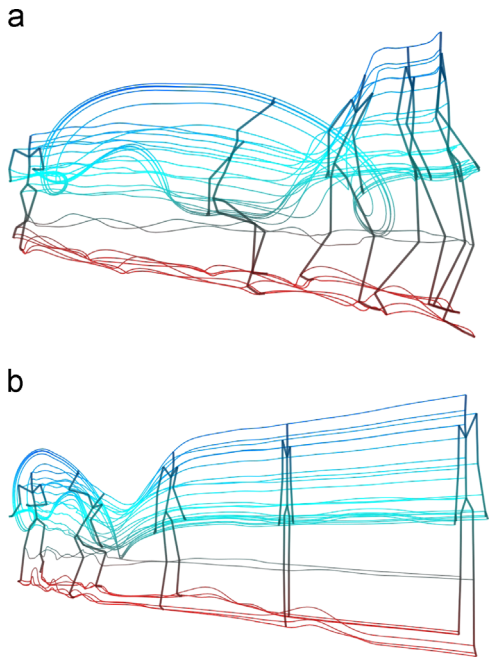

Fig. 3. Joint paths generated from the input 3D poses. Figure (a) represents the original joint curves, and (b) shows projected and time direction aligned joint curves.

A bilateral time control surface (BTCS), initially a quadrilateral perpendicular to the time direction, is selected by the user at time t where the motion retiming will be applied, as seen in Fig. 2c and discussed in Section 3.2. The BTCS at time t will be represented as BTCS_t throughout this paper, and it intuitively represents the motion's spatiotemporal relationship.

After selecting the BTCS_t, the user creates and moves its control points to manipulate the surface (Fig. 2d and Section 3.3). This process is repeated for each frame that needs retiming (Fig. 2e and Section 3.4). All the user-defined keyframe BTCSs are then interpolated to generate the final motion (Fig. 2f and Section 3.5).

3.1. Projected joint path generation

The first step in the process of motion retiming is to create joint paths from existing motion data. Following the notation of [5], the motion of a character composed of n joints is initially defined as a function $\mathbf{m}(t) = (\mathbf{p}(t), \mathbf{q}_1(t), \ldots, \mathbf{q}_n(t))$, where $\mathbf{p}(t)$ is the position of the root joint in $\mathbb{R}^3$, $\mathbf{q}_1(t)$ is the rotation of the root joint, and $\mathbf{q}_i(t)$, with $2 < i < n$, are the rotations of the rest of the joints at time $t \in [t_0, t_m]$. The skeleton is formed by a directed graph structure, $X = (J, E)$ where J are joints and E is a set of joint-index pairs. The position and rotation at time t of a joint i are computed by evaluating $\mathbf{q}_i(t)$, and recursively concatenating its parent joint's position and rotation. The evaluation returns a 4×4 matrix. Only the translation elements are used. We denote the 3×1 translation vector as $\mathbf{J}_i(t)$. By evaluating the entire time step, the motion equation can be reformulated as a matrix in the following equation:

$$\mathbf{M} = \begin{pmatrix} J_1(t_0) & J_1(t_1) & \cdots & J_1(t_m) \\ J_2(t_0) & J_2(t_1) & \cdots & J_2(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ J_n(t_0) & J_n(t_1) & \cdots & J_n(t_m) \end{pmatrix} \tag{1}$$

The design matrix M has $3n \times m$ dimensions, where n is the number of joints and m is the number of time steps. Rows represent each joint's path, and columns are positional elements of joints at a specific time step t. The path is displayed by connecting all the points with line segments, and it shows the overall motion of the joints throughout the animation.

Since the generated joint paths have 4 dimensions (x, y, z, and t), the visualization and intuitive control of joints' timing are not a trivial task. To tackle this problem, we reduce dimensions of the motion using a projection onto the time direction. The time direction, $\vec{d}$, is initially defined as the root's direction at time t_0, and the user can change $\vec{d}$ manually. Then we project the joints' positions onto a plane at time t that has a normal vector $\vec{d}$ and passes through $J_1(t)$, the root position. The plane equation at time t is thus $P_t(\vec{v}) = \vec{d} \cdot \vec{v} - \vec{d} \cdot J_1(t) = \vec{d} \cdot (\vec{v} - J_1(t))$. The projection can be done using the plane equation, $PJ_i(t) = J_i(t) - P_t(J_i(t))\,\vec{d}$. Finally, we apply the projection to the design matrix,

$$\mathbf{M_p} = \begin{pmatrix} PJ_1(t_0) & PJ_1(t_1) & \cdots & PJ_1(t_m) \\ PJ_2(t_0) & PJ_2(t_1) & \cdots & PJ_2(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ PJ_n(t_0) & PJ_n(t_1) & \cdots & PJ_n(t_m) \end{pmatrix} \tag{2}$$

An important advantage of the projected joint paths is that they allow the simultaneous visualization of *motion and time* at the same time. Each time step can be represented as a flat surface. Although we lose some information because of the projection, the user can choose the best $\vec{d}$ from her viewing perspective. The design matrix allows us to easily extract motion at time t or the joint path at index i, since the columns and the rows of the matrix represent keyframes and joint paths respectively.

3.2. BTCS definition

The projected motion paths generated in the previous section can convey spatial and timing information at the same time. In this section, we introduce BTCS that allows one to manipulate the timing of each joint with respect to the joints' topological information.

Initially, a BTCS is a bounded and conceptually flat surface that represents a specific time step t on the projected spatial domain (x and y). The boundary of BTCS is given by $x \in (x_{min}, x_{max})$ and $y \in (y_{min}, y_{max})$ where x_{min}, x_{max}, y_{min}, and y_{max} are the boundaries of projected joint paths M_p. We display the projected motion paths

along with the BTCS, so that users are able to see where the current time step is in the spatial domain. Our method allows users to manipulate the BTCS as a free-form deformation, and it can be understood as time being bent in the spatial domain. We apply the bilateral filtering algorithm [23] to combine joints' topological distance on the surface. The BTCS is a triangle mesh whose vertices form a $p \times q$ rectangular grid that we will call V. This planar mesh is positioned at every time step, it is perpendicular to the time direction $\overrightarrow{d}$, and it intersects the joint paths.

3.3. BTCS manipulation

The motion retiming is achieved by allowing the user to manipulate the shape of the BTCS, as illustrated in Fig. 4. The key idea of our approach is that by retiming a single joint, the joints in its periphery, bounded by topological distance, are also continuously affected and should undergo retiming as well. This way, smoothness of the retiming operation is achieved.

Let us recall that the purpose of the time manipulation is to define the precise time at which a certain action should occur, and also that the joints have been projected onto the surface, so by manipulating any part of the surface, the joints on the affected area will be retimed. In order to carry out a retiming operation, the user clicks on a point $\mathbf{p} = (p_x, p_y)$ on the BTCS domain and drags it to the desired time location along the direction of time $\mathbf{n}$ by α units of time. We denote $N_{p,t}$ as the nearest joint from $\mathbf{p}$ at time t:

$$\mathbf{N}_{p,t} = \arg\min_{J_i(t)} \|p - J_i(t)\|. \tag{3}$$

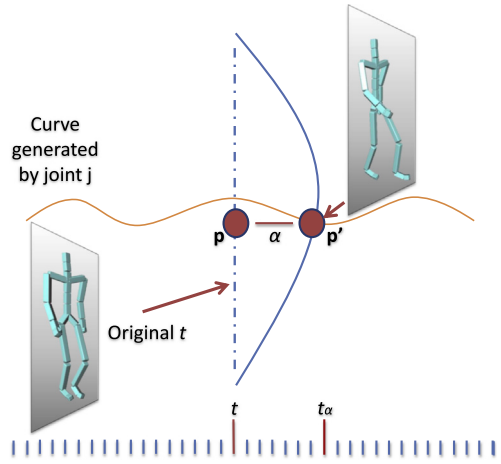

Fig. 4. BTCS editing. The user moves the control points of the surface. New motion parameters are sampled from the intersection between the surface and the joint path.

The underlying structure of the BTCS is a set of vertices that form a grid V. Each vertex v_i of V has two components $v_i = [x_i, y_i]$, where $i \in [0, m \cdot n)$. The new shape of the grid V, which we will call V', is achieved by evaluating a 2D interpolation or approximation function at each vertex v_i of V. We support 2D Gaussian and polynomial interpolation or approximation functions, $S : \mathbb{R}^2 \to \mathbb{R}$, and provide the user with the ability to switch between them. However, it can be any smooth surface such as a 2D spline or a user defined NURBS surface. It is desirable for the surface to interpolate the control points defined by the user. We could use an arbitrary interpolation scheme as long as it provides at least C^2 continuity.

3.3.1. 2D Gaussian $S(x, y)$

We used a 2D Gaussian function S to deform the surface:

$$S(x,y) = \exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)\right) \tag{4}$$

where σ_x and σ_y are the standard deviations through x and y directions, respectively. We intentionally remove the α, the normalization factor, since Eq. (8) takes into account the scale factor. A 2D Gaussian function provides smooth retimed t per joint. For x_0 and y_0, we provide the user clicked position $\mathbf{p} = (p_x, p_y)$, so that BTCS is deformed at the center $\mathbf{p}$ with the sigmas σ_x and σ_y. Our application also allows users to change σ_x and σ_y, so the shape of the deformation can be both isotropic ($\sigma_x = \sigma_y$) or anisotropic ($\sigma_x \neq \sigma_y$).

3.3.2. 2D Polynomial $S(x, y)$

We formulate an optimized 2D polynomial equation using Least Squared Fitting. As can be seen in Eq. (5), the 2D polynomial S is initially a power m polynomial equation:

$$\begin{aligned} S(x,y) = {} & (a_1 x^m + a_2 x^{m-1} + \cdots + a_m x) \\ & + (b_1 y^m + b_2 y^{m-1} + \cdots + b_m y) \\ & + (c_1 x^{m-1} y + c_2 x^{m-2} y^2 + \cdots + c_m x y^{m-1}) \end{aligned} \tag{5}$$

Then we need to determine $a_1, \ldots, a_m, b_1, \ldots, b_m, c_1, \ldots, c_m$. If we have n control points, then we can generate a matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} x_1^m & \cdots & x_1 & y_1^m & \cdots & y_1 & x_1^{m-1}y_1 & \cdots & x_1 y_1^{m-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_n^m & \cdots & x_n & y_n^m & \cdots & y_n & x_n^{m-1}y_n & \cdots & x_n y_n^{m-1} \end{pmatrix} \tag{6}$$

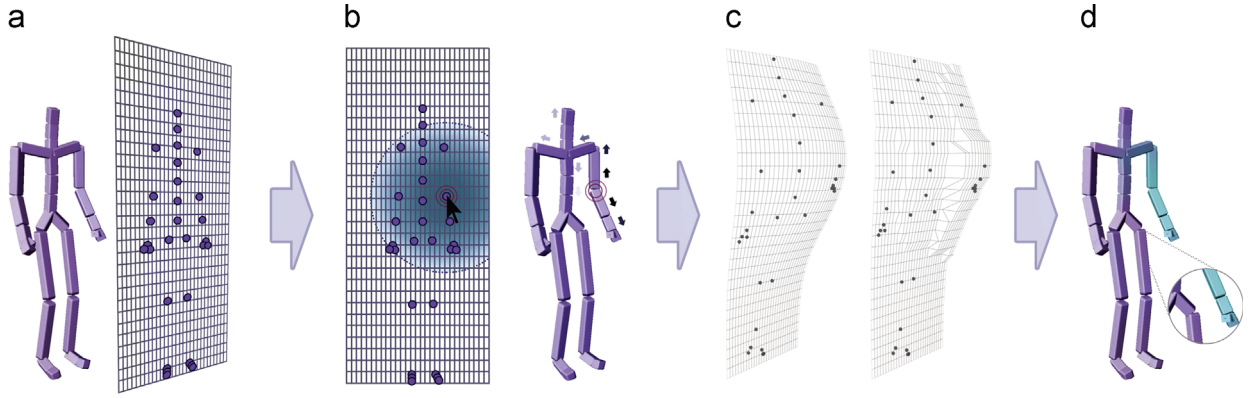

Fig. 5. Topological distance. (a) Projected joints on the BTCS. (b) The user clicks on the elbow joint, and the topological distance from the elbow to the rest of the joints is computed. (c) The topological distances are combined with the time control surface, and formed BTCSs. (d) Weight distribution is based on the topological distance represented by the cyan gradient. The inset shows that even if the leg and the hand are close in space, the operation will not affect the leg due to the high topological distance.

We multiply the matrix A with the vector $x=(a_1,\ldots,a_m, b_1,\ldots,b_m,c_1,\ldots,c_m)^T$, so that $Ax=b$ where b is a known displacement from the surface. Since the designed matrix is an over-determined system and possibly singular, we apply Moore–Penrose pseudoinverse (denoted as A^+) to find the best fit x by solving $x=(A^TA)^+A^Tb$. A large m may result in fluctuation of the surface (Runge's phenomenon), and may lead to difficulties with user control, so we generally use $m\leq 3$.

3.3.3. BTCS surface editing

We also take into account the topological distance between $N_{p,t}$ and grid vertices v_i (see Fig. 5). Let us define $E_s(i,j)\in X$ as the edges of the shortest path from joint index i to j. Then the topological distance, $T(N_{p,t},N_{v_i,t})$, can be calculated:

$$T(N_{p,t},N_{v_i,t})=\sum_{i,j\in E}\|J_i-J_j\| \tag{7}$$

where $E=E_s(N_{p,t},N_{v_i,t})$. Then, we can deform the BTCS by calculating s_i for every v_i in V:

$$s_i=\alpha\cdot S(x,y)\cdot G_{\sigma_r}(T(N_{p,t},N_{v_i,t})) \tag{8}$$

where G_{σ_r} is the 1D Gaussian function with standard deviation σ_r, α is the normalization factor, and s_i is used to compute the configuration of the new v_i' is defined by $v_i'=v_i+s_i\mathbf{n}$. The bilateral step is represented in Eq. (8), where $S(x,y)$ is the domain filter and $G_{\sigma_r}(T(N_{p,t},N_{v_i,t}))$ is the range filter. The domain filter smoothly deforms the surface, and the range filter applies topological distance to the smooth surface. A retriangulation of the surface is not necessary because we are not changing the mesh topology.

In Section 3.5, we describe how the new timing of joints is calculated using the BTCS deformation.

3.4. BTCS keyframing

We described how to manipulate a BTCS at a single time t to achieve retiming. Sudden change of a BTCS_t at time t would cause discontinuities in the motion, so we smoothly propagate and deform BTCS_t to $\pm\gamma$ consecutive time. The parameter γ could be either set explicitly by the user or calculated by the smooth deformation. If BTCS_t is modified by the user, the surfaces at times around t are C^2 smoothly deformed into the shape of BTCS_t.

Let us define δ_l and δ_u, as bounds of $t\in BTCS_t$

$$\begin{aligned}\delta_l&=\inf\{t|\forall i,\quad v_i=(x,y,z,t)\in BTCS_t\}\\ \delta_u&=\sup\{t|\forall i,\quad v_i=(x,y,z,t)\in BTCS_t\}\end{aligned} \tag{9}$$

In our framework, our initial $\delta_l=\delta_u=t\in BTCS_t$, so BTCS_t is a flat plane. However, after the manipulation, the BTCS_t spans (δ_l,δ_u). To preserve smoothness of time propagation, we provide additional $\pm\gamma$ to the time range, so that interpolation is applied on $(\delta_l-\gamma,\delta_u+\gamma)$. Let d_t^i be the displacement of the i-th vertex of BTCS_t, and let d_τ^i be the displacement of the i-th vertex of BTCS_t where $\delta_l-\gamma<\tau<\delta_u+\gamma$. We generate the displacement values for these vertices in such a way that the vertices from surfaces furthest from t have a small displacement. The displacement increases as the surfaces are closer to t, and interpolates the value t. The smooth variation is achieved by computing $d_\tau^i=\alpha d_t^i$, where α is obtained from a user defined smooth interpolation function. To avoid complex and counterintuitive behavior, we prohibit the user from setting the BTCS keyframes anywhere inside of other BTCS time ranges.

3.5. Output motion generation

In the previous section, we calculated the deformation of the BTCS for each time step of the animation. Using this data, we can now generate the output animation that corresponds to the user-defined actions and smoothly interpolates the BTCSs.

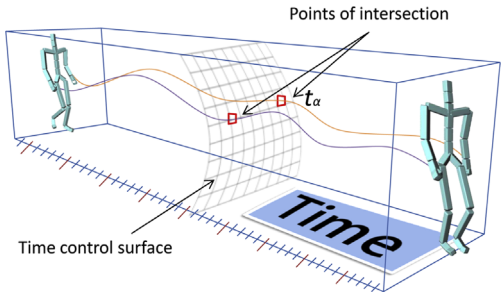

Fig. 6. Intersection points between the BTCS and the joint paths.

The shape of the BTCS at each time step determines the new character pose $m'(t)$. The new time t_j for each joint j can be determined from the intersections between BTCS_t and the projected joint paths. Thus, we obtain $m'(t)$ by computing the intersections of BTCS_t with all the projected joint paths and then re-evaluate each joint's rotational and positional values.

Once the intersections of the joint paths with BTCS_t have been computed, the new joint positions can be generated. The location of the intersection of the joint curve c_i with BTCS_t takes place at the time step t_α (Fig. 6). The new value for joint i at time t is $\mathbf{q}'(t)=\mathbf{q}(t_\alpha)$. This means that we are replacing the rotation of the joint at time t with the rotation of the joint at time t_α, effectively applying a retiming operation to the joint.

The tessellation of the BTCS determines its smoothness, which is directly linked to the accuracy of the intersection point. In the case of this work, a 30×30 vertex mesh yields good results.

4. Results

In this section we show several applications of our framework. Namely we describe retiming on a golf swing, a yawn, a kick, a throw, and a baseball swing. Additional animations can be found in the accompanying video.

Our framework allows for an interactive motion retiming. It has been implemented in C++, uses CUDA for collision detection, and OpenGL with GLSL for visualization. For the Moore–Penrose pseudoinverse calculation, we use the Eigen math library, and calculate the pseudoinverse using Singular Value Decomposition (SVD). We tested the framework on a computer equipped with an Intel i7 CPU clocked at 2.4 GHz, with 16 GB of memory, and an NVIDIA Tesla K40c.

Our application allows users to choose and deform specific BTCS_t by simply clicking and dragging on the surface so that users could quickly change the surface (see Fig. 5, example B). Other parameters, such as γ, polynomial surface's power m, Gaussian surface's σ_x and σ_y, can be selected and changed by users in the GUI of our application. The default values of the parameters are $\alpha=1$, $\gamma=0.3$, $m=3$, $\sigma_x=1.0$, and $\sigma_y=1.0$. Changing the parameter's values has a great effect on the motion and might even create some distortions. The dimensions of the rectangular grid of the BTCSs can be adjusted. We use a 30×30 rectangular grid. We also provide an option to change only the rotations of joints. In many of our examples, we do not need to change the positional value of root joint. Instead we apply only rotations to avoid footskating.

Fig. 8 shows several examples of retimed motions, with around five to six frames of the motion shown. These frames are not contiguous in the original motion, but are sampled several frames apart. Note that there is an accompanying video showing the examples.

Fig. 8a shows an example of a retimed yawning motion. The objective is to make both arms stretch faster than the original motion, and also to make the right arm stretch faster than the left arm. The third BTCS in the figure has been manipulated by the user to perform the retiming. Note that the purple character's right arm is ahead in the motion. The frames surrounding the manipulated BTCS are smoothly interpolated until the last frames (right) are completely in sync.

Fig. 8b shows a retimed right hand throw in which the hand of the retimed character is raised earlier. Fig. 8c shows a retimed baseball swing where the purple character starts the swing before the cyan character. The motions are aligned by the last frame in the figure.

The last example in Fig. 8d shows a retimed kick. The third frame in the figure was manipulated by the user to make the right leg move faster than the original. Note again how the rest of the BTCS's are smoothly deformed to avoid abrupt changes in the motion around the modified $BTCS_t$.

To verify that the variation applied to each joint when retiming is smooth, we record each of the joint's retimed t, at each time step, using a sampling of 120 frames per second. Fig. 7 shows an example of this, with golf swings that have been slowed down. The duration of the reference golf swing motion is approximately 3 s, and we deform a BTCS near $t=2.35619$ and $\alpha=0.597557$ for the slowed down swing motion.

The inset in Fig. 7b shows the retimed region where $t \in (t-\alpha, t+\alpha)$. The slowed down motion in Fig. 7b shows that the time curves decrease in the area where $t \in (t-\alpha, t+\alpha)$.

Comparison with other methods is difficult and can vary depending on quality of the results, time consumption, intuitiveness, and diversity of editable motions. However, to informally verify our method, we asked two trained professionals to retime several motions, and to record the time spent and parameters used. Table 1 summarizes the parameters and shows the time required for retiming the motions. The table provides the following data: the name of motions, δ_l, δ_u, γ, and the time to create the retimed animation in seconds. The average time required for a retiming operation is 69.6 s, with standard deviation 9.7 s, and the average duration of the retiming is 0.76 s. Using commercial software packages, the retiming process requires more time, since the motion editing is done with a graph editor. A graph editor is a 2D interface that represents joint motion with three different curves, one for each axis of motion. It often requires multiple iterations of playback on retimed motions to verify the accuracy of the change. Using our system, when editing a baseball pitch (see Fig. 7, example C), trained users follow a three step process. First, they find the joint of most influence. In the case of a baseball pitch, it will be the wrist of a right hand. Then they place BTSC at the appropriate time position where the motion is to be edited. Finally, they deform the BTCS so that the joint of most influence will be in the center of the deformed BTCS. The rest of the joints will adjust according to the corresponding the surface on the BTCSs. Note, there may be cases when the user might prefer to not place the joint of most influence in the center. One example of this is if the joint is located close to the edge of the BTCS.

The results provided above show retiming operations performed directly on motion capture data that contains forward kinematic keys on every frame. This sort of motion edit would require the creation of inverse kinematic controls, as well as an experienced animator, to achieve the same goal as what can be achieved in seconds (Fig. 9).

Table 1
Parameters time for the motion retiming of examples from this paper. Headings indicate name of the motion, their lower and upper bounds on t of the BTCS, γ, and editing time.

Motion	δ_l	δ_u	γ	*Editing time* (s)
Soccer kick	3.1	4.8	0.63	58
Bowling	1.6	1.9	0.43	78
Cartwheeling	0.4	1.0	0.65	81
Tai chi	1.7	2.6	0.45	73
Stretching	0.8	1.5	0.46	81
Ballet	1.5	1.9	0.36	63
Dance 1	7.3	8.0	0.30	52
Dance 2	5.2	5.8	0.47	73
Arm swing	9.9	10.6	0.3	67
Pole swing	3.0	4.0	0.46	70

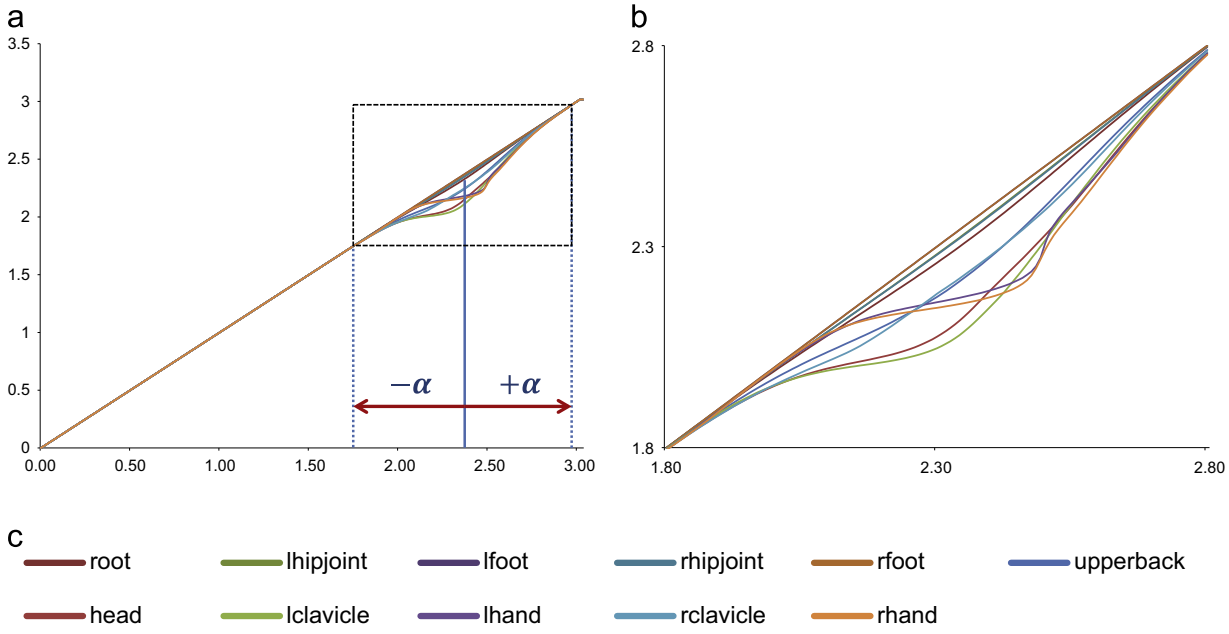

Fig. 7. The graphs show the retimed t for joints of a golf swing motion at each time step (a) and (b). (c) shows the joint color information. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

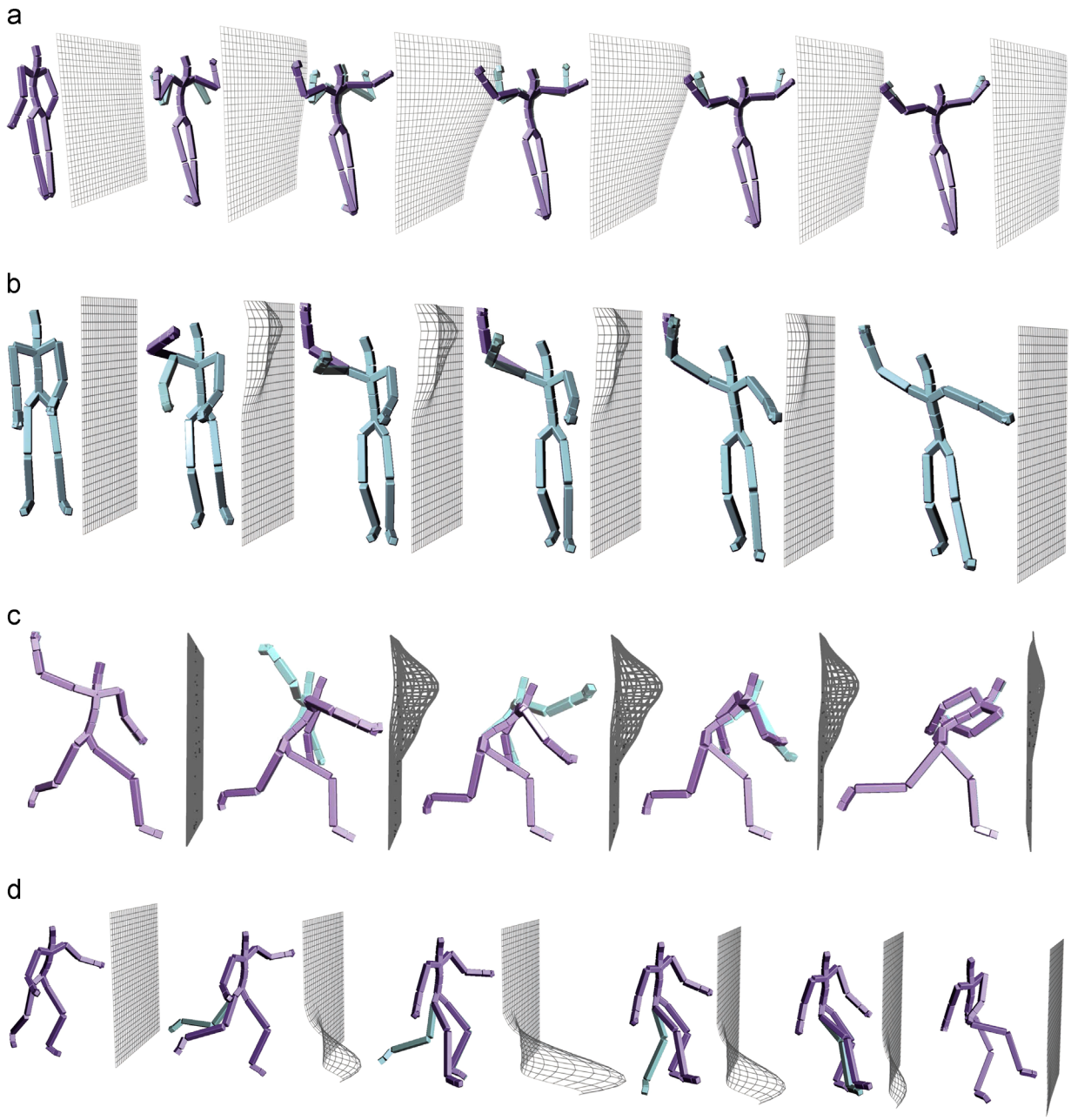

Fig. 8. Examples of retiming. The character in cyan color represents the original motion and the purple one is the resulting motion. The surfaces shown in front of the characters are the BTCS's for each frame. The examples show (a) retimed yawning motion, (b) throw, (c) baseball pitch, and (d) soccer kick. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

5. Conclusions

In this paper, we introduce an interactive framework for retiming character animations. The animations contain keys in every frame, as is common with motion capture, and they would be otherwise cumbersome to edit, given the amount of pre-processing required on the data to achieve the same goal. Our approach takes advantage of the spatial proximity of joints to preserve spatio-temporal coherence of the motion through the use of newly introduced bilateral time control surfaces. The strength of our framework is that by simply manipulating a surface at the desired time steps, it is possible to perform retiming operations in a matter of seconds, without the need of inverse kinematics controls to keep the coherence and coordination between joints that are close spatially or topologically. The proposed methodology also retimes joints in a successive order, which is a property that enhances the sense of flow in an animation, as described by Neff and Fiume in [8]. Our approach is intuitive, fast to use, and provides interactive results.

Our approach also has some limitations. When the velocity and acceleration of the root joint are changed, or any parts that contribute to locomotion are retimed, it creates the danger of footskating, or may alter the style of the motion. Although our framework allows users to retime only the rotations of each joint, it does not provide a complete solution for this artifact. When the artifact is not too severe, it can be solved by methods like [24]. It is also difficult to manipulate BTCSs that would overlap, as a certain distance between two successive frames is necessary for an efficient action. This could cause problems if a retiming of very fast action is required. The projected joint paths sometimes can suffer a loss of information. When time direction is nearly perpendicular to any keyframe root direction, retiming can be

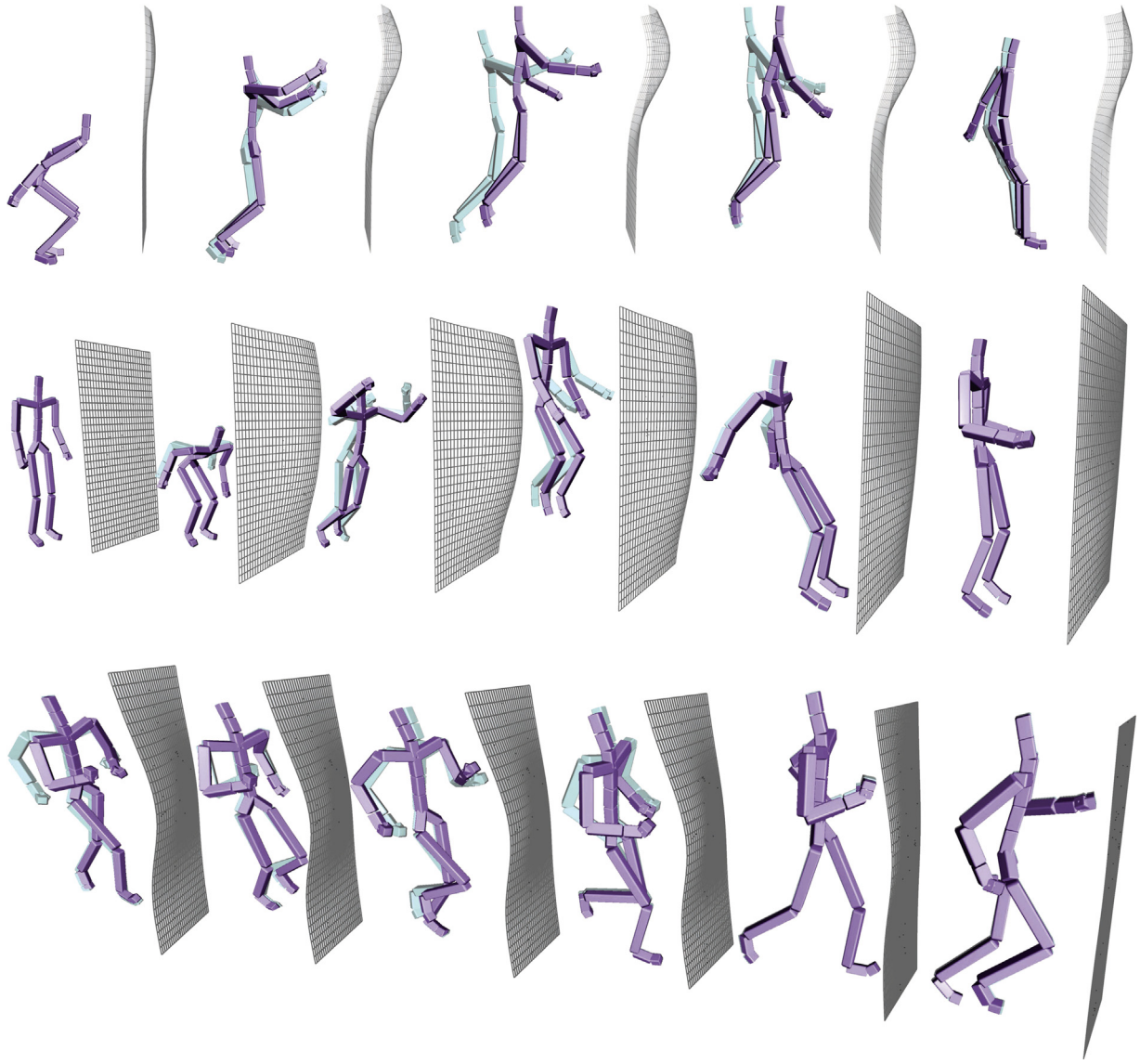

Fig. 9. Another examples of retimed motions: jumping and running. The character in cyan color depicts the original motion and the purple one is the retimed one. The surfaces shown in front of the characters show the BTCS's for each frame. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

imprecise, but it can be alleviated by manual manipulation of time direction. Moreover, a motion that is moving backwards can cause a counter-intuitive effect in our framework. Similar to the other retiming methods, some properties of motion captured data, such as balancing and constraints of specific limbs, could be broken during the retiming process.

As future work, we would like to extend our method to allow for generation of diverse motion by automatically computing deformations of the BTCSs. Another future direction is generating cartoonish exaggerated motion by using the BTCSs for sampling joint positions instead of rotations, in order to achieve a squash and stretch effect.

Acknowledgments

The motion data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. We also thank NVIDIA for providing graphics hardware.

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2014.11.001.

References

[1] Bruderlin A, Williams L. Motion signal processing. In: Proceedings of the SIGGRAPH'95; 1995. p. 97–104.
[2] Heck R, Kovar L, Gleicher M. Splicing upper-body actions with locomotion. In: Computer graphics forum, vol. 25; 2006. p. 459–66.
[3] Rose C, Cohen MF, Bodenheimer B. Verbs and adverbs: multidimensional motion interpolation. Comput Graph Appl 1998;18(5):32–40.
[4] Witkin A, Popovic Z. Motion warping. In: SIGGRAPH'95; 1995. p. 105–8.
[5] Lee J, Shin SY. A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of the SIGGRAPH'99; 1999. p. 39–48.
[6] McCann J, Pollard NS, Srinivasa S. Physics-based motion retiming. In: Proceedings of the SCA; 2006. p. 205–14.
[7] Hsu E, da Silva M, Popović J. Guided time warping for motion editing. In: Proceedings of the SCA; 2007. p. 45–52.
[8] Neff M, Fiume E. Aesthetic edits for character animation. In: Proceedings of the SCA; 2003. p. 239–44.

[9] Coleman P, Bibliowicz J, Singh K, Gleicher M. Staggered poses: a character motion representation for detail-preserving editing of pose and coordinated timing. In: Proceedings of the SCA; 2008. p. 137–46.
[10] Thorne M, Burke D, van de Panne M. Motion doodles: an interface for sketching character motion. In: ACM SIGGRAPH 2007 courses; 2007. p. 24.
[11] Guay M, Cani M, Ronfard R. The line of action: an intuitive interface for expressive character posing. ACM Trans Graph 2013;32(6):205.
[12] Yoo I, Vanek J, Nizovtseva M, Adamo-Villani N, Benes B. Sketching human character animations by composing sequences from large motion database. Vis Comput 2014;30(2):213–27.
[13] Dontcheva M, Yngve G, Popović Z. Layered acting for character animation. ACM Trans Graph 2003;22(3):409–16.
[14] Igarashi T, Moscovich T, Hughes JF. Spatial keyframing for performance-driven animation. In: Proceedings of the SCA; 2005. p. 107–15.
[15] Kim M, Hwang Y, Hyun K, Lee J. Tiling motion patches. In: Proceedings of the 11th ACM SIGGRAPH/eurographics conference on computer animation, EUROSCA'12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 978-3-905674-37-8; 2012, p. 117–26.
[16] Wang H, Ho E, Komura T. An energy-driven motion planning method for two distant postures. IEEE Trans Vis Comput Graph 2014. p. 18–30.
[17] Ho E, Komura T. Indexing and retrieving motions of characters in close contact. IEEE Trans Vis Comput Graph 2009;15(3):481–92.
[18] Neff M, Fiume E. Aer: aesthetic exploration and refinement for expressive character animation. In: Proceedings of the SCA; 2005, p. 161–70.
[19] Lockwood N, Singh K. Fingerwalking: motion editing with contact-based hand performance. In: Proceedings of the SCA; 2012. p. 43–52.
[20] Terra SCL, Metoyer RA. Performance timing for keyframe animation. In: Proceedings of the SCA; 2004. p. 253–8.
[21] Mukai T, Kuriyama S. Pose-timeline for propagating motion edits. In: Proceedings of the SCA; 2009. p. 113–22.
[22] Kim M, Hyun K, Kim J, Lee J. Synchronized multi-character motion editing. ACM transactions on graphics (TOG), vol. 28. ACM, New Orleans, Louisiana, New York, NY, USA; 2009. p. 79.
[23] Tomasi C, Manduchi R. Bilateral filtering for gray and color images. In: Sixth international conference on computer vision; 1998, p. 839–46.
[24] Kovar L, Schreiner J, Gleicher M. Footskate cleanup for motion capture editing. In: Proceedings of the 2002 ACM SIGGRAPH/eurographics symposium on computer animation. ACM, San Antonio, Texas, New York, NY, USA; 2002, p. 97–104.