Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Biomechanically-Inspired Motion Path Editing

Noah Lockwood and Karan Singh

University of Toronto

Abstract

We present a system for interactive kinematic editing of motion paths and timing that employs various biomechanical observations to augment and restrict the edited motion. Realistic path manipulations are enforced by restricting user interaction to handles identified along a motion path using motion extrema. An as-rigid-as-possible deformation technique modified specifically for use on motion paths is used to deform the path to satisfy the user-manipulated handle positions. After all motion poses have been adjusted to satisfy the new path, an automatic timewarping step modifies the timing of the new motion to preserve the timing qualities of the original motion. This timewarp is based on biomechanical heuristics relating velocity to stride length and path curvature, as well as the preservation of acceleration for ballistic motion. We show that our system can be used to quickly and easily modify a variety of locomotive motions, and can accurately reproduce recorded motions that were not used during the editing process.

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

1. Introduction

Motion editing is a quick way of reusing motion data that is particularly useful to novice animators. Current motion editing techniques are often limited to a particular class of motions, either by learning a model based on a number of example input motions, or using analytical expressions or physical models of a particular motion type. We seek to address this through motion path editing for scenarios where a corpus of data for the motion to be edited is not readily available and where the overall motion can exhibit a variety of action that makes it hard to label into a fine-grained motion class. Rather than propose a general strategy that processes path edits in a uniform manner over the entire motion, we base our approach on biomechanical observations and use these to both constrain the scope of editing on a path as well as how the motion is edited to match the user alterations.

We use a modified as-rigid-as-possible deformation technique to modify motion paths. Instead of allowing arbitrary constraints and manipulations of the path, which can result in unrealistic motions, user manipulation of the path is restricted to a sparse set of point handles along the path. These handles are automatically detected by identifying geometric features along the path which correspond to changes in ground contact, enforcing that any significant changes in path are anchored at locations where the character has leverage to make such changes. Once a user has manipulated the handle positions to determine a new path, all poses of the motion are modified accordingly, and an automatic timewarping step modifies the timing along the path to preserve the relationship between the stride length and velocity of the original motion.

In addition, all stages of this method are automatically augmented and restricted by a set of biomechanically and physically-inspired heuristics. We decouple horizontal and vertical motion editing mechanisms in space, to better reflect biomechanical motion rather than orientation invariant geometric relationships. We identify segments without environmental contact, and restrict both the spatial deformation and timewarping of these segments to preserve the shape and timing of ballistic motion. Finally, turning also automatically affects motion timing by using a well-established relationship between path curvature and velocity.

We show that our system can be used to quickly and easily modify a variety of locomotive motions, and can accurately reproduce recorded motions that were not used during the editing process.

Our work makes the following contributions:

- An improved as-rigid-as-possible deformation algorithm which corrects the artifacts introduced when applying the original algorithm to the editing of motion paths.
- A general kinematic path editing framework, which identifies useful manipulation handles based on motion features, and automatically timewarps deformed motion based on the original motion's timing.
- A number of motion components based on biomechanical observations and simple physics which augment and restrict both the spatial and temporal editing procedures to preserve natural motion.

2. Related Work

A path-based approach is not uncommon for a variety of animation tasks. A sketched path can be used to synthesize motion from interpolated motion graph traversals [SH07], individually parsed and parameterized motions [TBvdP04], or deformable models of a large number of example motions [MCC09]. Our path-based editing approach is similar to that of Gleicher [Gle01] and Kim et al. [KHKL09], however, our system can operate on more complicated paths involving vertical motion, and also incorporates an automatic timewarping step to maintain timing similarity.

Some motion editing approaches operate on the spatial components of a motion, incorporating constraints and solving for a new character pose at each frame. Editing with spacetime optimization [Gle97] can incorporate user-specified constraints while maintaining overall motion smoothness, while automated systems can automatically modify motions based on a variety of factors, such as physical plausibility [SKG03].

Some approaches focus solely on temporal transformation; namely, re-timing or timewarping a motion so that the same spatial values are expressed, but with different duration or speed. One possible method of expressing desired timing changes for a motion can be the user's performance with a simple 2D input device [TM04]. Like Hsu et al. [HdSP07], our approach determines a timewarp based on an example motion – however, their system requires a user-specified temporal directive such as duration to determine a discrete (frame-based) timewarp, while ours determines a continuous timewarp automatically.

Finally, some approaches can combine spatial and temporal transformation. The animation principles of anticipation, exaggeration and follow-through can be applied to the path, timing, and deformation of an object [WDAC06]. Alternate motion representations have been developed which allow the modification of motion extrema and timing succession between related joints [CBSG08]. The forces and momentum of a motion can also be directly modified, with both the duration, speed and extent of a motion modified accordingly [SYLH10].

3. Basic Motion Editing Procedure

Our procedure operates on the animation of a single character skeleton, in the form of a standard hierarchy of joint transformations. Character motion is represented as a discrete sequence of *keys*, which describe the rotation of every skeleton joint as well as the translation of the root joint at an associated time; the sequence of 3D root joint positions defines a character's *motion path*. Our algorithm does not utilize the motion curve information present in modern animation software, which allows for the editing of sampled animation from physical simulation or motion capture. Sparsely keyframed animation is also permitted, as keys are not required to be uniformly arranged in time.

Our algorithm operates on an animation and motion path in four stages. In the first stage, which occurs only when an animation is initially loaded, the animation is processed to detect contact constraints and determine path manipulation handles. In the second stage, the user manipulates the handles to deform the path into a new shape. In the third stage, our system automatically adjusts the character poses at every key, while conforming to the user's path modification. Finally, in the fourth stage, an automatic timewarp adjusts the timing of the animation.

3.1. Motion Preprocessing

The goal of the preprocessing stage is to identify points along the motion path to serve as user-manipulable *handles* which constrain the deformed path. While it is possible to allow the user-selected handles [KHKL09], this can easily generate unrealistic motions; for example, such motions could violate the observation that humans are incapable of turning suddenly during a step and instead require appropriate preparation during the step preceding a turn [PPRN91]. Therefore, we seek to automatically determine handles which only enable natural-looking changes in the motion path. We also seek to avoid confusing the user by introducing or removing handles during interaction. Having a constant set of handles implies that handles should occur not only where a motion changes, but where a motion *could* change.

We hypothesize that the most meaningful path changes during locomotion can potentially occur at points which are identified by vertical minima of a character's root position. As we are considering characters who propel themselves by exerting force against a static environment in the presence of gravity, vertical minima naturally correspond to the inflection point at which a character stops traveling downwards with gravity and is actively opposing it – indicating an exertion which could, but does not always, change the path significantly. However, many local minima can occur during a motion due to high-frequency motion characteristics or motion capture noise, not all of which correspond to potential motion changes (Figure 1, top).

To address this, handles are identified from the minima during specific periods of the motion identified from the character's environmental contact. Using the technique of Coleman et al. [CBSG08], contacts are detected for predefined parts of the character skeleton ("end effectors"). Based on the timing of these contacts, time periods of a locally maximum number of contacts are identified; this includes, for example, the double-stance phase of a human walk. Within each period of maximum contact, the vertically minimum path point is retained as a handle. This results in one handle for each step of periodic motions, in addition to handles enforced at the first and last path points (Figure 1).

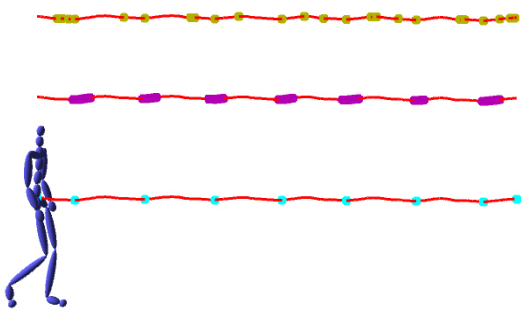

Figure 1: *User-manipulated handles (bottom) are detected by finding local vertical minima along the path (top) during each period of locally maximal contact (middle).*

3.2. Path Manipulation and Deformation

We develop our path manipulation algorithm to fulfill the goals of a direct manipulation system [Shn97]. Chief among these in a motion editing context is that operations should be easily incremental and reversible, which is accomplished by determining each deformation relative to the starting configuration, rather than the previous deformation result.

We utilize a modified version of the as-rigid-as-possible deformation algorithm of Igarashi et al. [IMH05], which was previously used with slight modification by Kim et al. [KHKL09] for motion path editing in two dimensions. We have found our modified version of this algorithm to be more reliable and intuitive for interactive path deformation than other methods, such as splines with displacement maps or blended transformations.

In the algorithm's first stage, a "scale-free deformation" of the initial points $\mathbf{p} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is determined by defining a relative position (u_i, v_i) for each point $\mathbf{p}_i$ in a local coordinate system determined by the neighbouring points:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + u_i(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) + v_i R_{90}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) \quad (1)$$

The "desired" position of each point in the scale-free path $\bar{\mathbf{p}} = \{\bar{\mathbf{p}}_0, \bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_n\}$ is the reconstructed position from these local coordinates:

$$\bar{\mathbf{p}}_i^{desired} = \bar{\mathbf{p}}_{i-1} + u_i(\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_{i-1}) + v_i R_{90}(\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_{i-1}) \quad (2)$$

The scale-free path $\bar{\mathbf{p}}$ minimizes the weighted sum of squared distances between the desired and final positions of each point:

$$\underset{\bar{\mathbf{p}}}{\operatorname{argmin}} \sum \bar{w}_i \|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_i^{desired}\|^2 \quad (3)$$

where $\bar{w}_i = \frac{1}{\|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|}$, and the handle positions are enforced as as hard linear constraints $\mathbf{H}\bar{\mathbf{p}} = \mathbf{h}$.

The second stage of the algorithm generates the final "scale-adjusted" path $\hat{\mathbf{p}} = \{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\}$ based on $\bar{\mathbf{p}}$. Let $\bar{\mathbf{e}}_i = (\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i)\frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\|\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i\|}$ be the edge between two adjacent points $\bar{\mathbf{p}}_{i+1}$ and $\bar{\mathbf{p}}_i$ scaled to the length of the corresponding original edge. Then $\hat{\mathbf{p}}$ minimizes the weighted sum of squared distances between the final and scale-adjusted edges:

$$\underset{\hat{\mathbf{p}}}{\operatorname{argmin}} \sum \hat{w}_i \|\hat{\mathbf{e}}_i - \bar{\mathbf{e}}_i\|^2 \quad (4)$$

where $\hat{\mathbf{e}}_i = \hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i$, $\hat{w}_i = \frac{1}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$, and $\hat{\mathbf{p}}$ is subject to the same hard handle position constraints in the form of $\mathbf{H}\hat{\mathbf{p}} = \mathbf{h}$. The linear systems for both stages are sparse and can be efficiently computed using a sparse LU solver [Dav04].

We have modified this algorithm in two ways. The original deformation algorithm operates on a triangulated mesh, but artificial triangle edges can cause kinks during deformation (Figure 2b). Kim et al. [KHKL09] modified the scale adjustment term to not require additional edges, however we have found our explicit edge difference formulation (Equation 4) to be simpler and better behaved. We have also modified both stages to incorporate inverse edge length weighting ($\bar{w}_i$ and $\hat{w}_i$) to compensate for uneven path sampling. In the second stage in particular, weighting by $\hat{w}_i$ has the desirable effect of modifying deformations with two handles exactly into a nonuniform scale in the handle displacement direction (compare Figure 2c and Figure 2d).

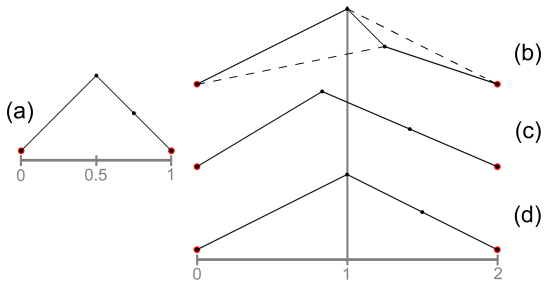

Figure 2: *An initial curve (a) being deformed under three formulations: with an artificial triangulation (b), uniform edge weighting (c), and inverse edge length weighting (d).*

3.2.1. Automatic Scale Factor

A significant disadvantage of this algorithm as utilized by previous work is that as deformations become larger, increasingly large derivative discontinuities are introduced at point constraints, causing a loss of smoothness (Figure 3, top). This behaviour occurs in the original triangulated formation of the deformation algorithm as well, and is not improved by supersampling the initial curve. The discontinuities are due to the scale adjustment error term (Equation 4), which results in the path segments between point constraints being "flattened" to more closely match the original edge lengths.

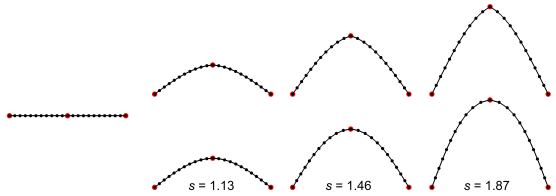

Figure 3: *The original algorithm introduces increasingly large discontinuities in smoothness (top); our automatic scale factor corrects these discontinuities (bottom).*

To correct this, we introduce an extra "scale factor" parameter s which uniformly scales the path prior to deformation, based on the notion that since small deformations introduce smaller discontinuities, large deformations of a particular path could be replaced by small deformations of a larger copy of that path. This results in the scale adjustment step penalizing deviations from a variable scale rather than the original scale, and is expressed by modifying equation 4 to:

$$\underset{\hat{\mathbf{p}}}{\operatorname{argmin}} \sum \hat{w}_i \|\hat{\mathbf{e}}_i - s \cdot \bar{\mathbf{e}}_i\|^2 \qquad (5)$$

The first stage of the algorithm (Equation 3) is scale-independent and is unaffected by any uniform scaling introduced by s.

Therefore, since only the second stage of the algorithm is affected, the final path point positions vary linearly with respect to s. When $s = 0$, the path minimizes the weighted sum of edge lengths, resulting in a path which linearly interpolates the point constraints (center path in Figure 4). Since this "zero-scale path" can be determined in closed form, the linear parameterization of the paths satisfying a particular arrangement of handles can be easily computed by solving the scale adjustment linear system only once.

We have found that discontinuities are almost always eliminated by choosing the scale factor which minimizes the difference between the arclength of the final curve and of the scaled initial curve; i.e., the scale factor which approximately results in only bending and not stretching to deform into the final shape (Figure 3, bottom). However, for

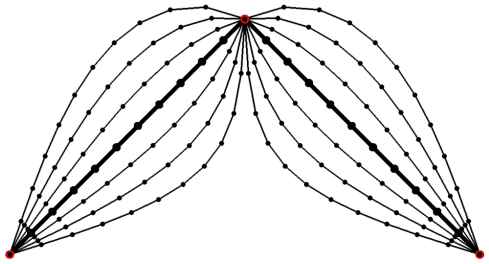

Figure 4: *A deformation at integer scale factors between -3 and 3, with the zero-scale curve (center) in bold.*

asymmetric deformations, a single scale factor can still introduce discontinuities; instead, separate scale factors can be determined independently for each path segment (Figure 5). These final scale factors are determined automatically through a line search which generally converges within 2-5 iterations, and each iteration is simple to compute using the linear scale parameterization.

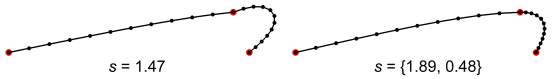

Figure 5: *A single scale factor results in discontinuities in highly asymmetric deformations (left). Independent scale factors for each segment maintain smoothness (right).*

3.3. Pose Transformation

The transformation of the root path determines the transformation of all other aspects of the animation. First, the character's rotation at each key must be adjusted. This is determined from the instantaneous horizontal direction at each key, calculated with central differences from the previous and subsequent key positions. For each key, any rotation of this direction is applied to the character, thus preserving the character's orientation relative to their instantaneous horizontal direction.

Any end effectors of the character which are constrained from environmental contact must also be adjusted to maintain smooth motion without introducing footskate. This is accomplished by applying the same motion path deformation algorithm to the end effector paths, with the contact keys as handles. Gleicher [Gle01] noted that constrained foot positions during ground contact must be rigidly transformed together, and while this transformation should correspond to *some* root key from the constraint period, the most appropriate key is unclear. However, we have observed that human footplants are often nearly parallel with the tangent of the root path at its closest point (Figure 6). Thus, the associated key for a constraint is determined not temporally

but spatially, by choosing the key which minimizes the horizontal distance to the constraint. For any edit, we apply the horizontal transformation of the associated key to all frames of the constraint, which maintains horizontal foot placement relative to the root but avoids ground interpenetration. End effector positions in-between constrained sections are automatically determined by the deformation algorithm, using additional handles to maintain relative foot positions where a swing leg passes a footplant, with new rotations determined using the same method as the root. Given these new end effector transformations, the intervening joint angles are determined using the closed–form inverse kinematics solution of Tolani et al. [TGB00].

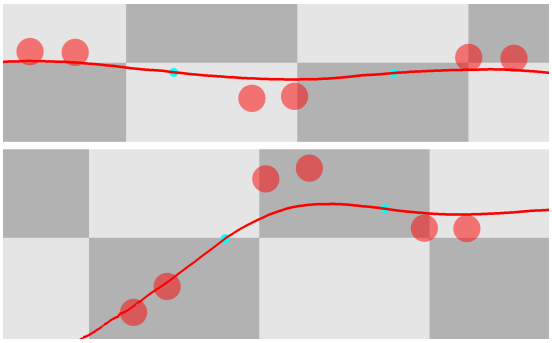

Figure 6: *During locomotion, foot orientation seems to remain roughly normal to the root path at its nearest point.*

3.4. Timewarping

Once a path manipulation is complete, playing the motion with its original timing will often look unnatural. While it is possible for a user to manually manipulate the timing of a deformed motion [KHKL09], we believe that determining a natural timing is a difficult task, and especially undesirable when a user may want a deformed motion to simply "look right" without needing to specify speed or duration.

In addition, the spatial and temporal attributes of a motion are tightly coupled and arguably should not be modified independently. Such a relationship can be described using the Froude number, a quantity used to establish dynamic similarity between phenomena at different scales, which was originally developed to aid in 19th century ship design [Fro74] (Vaughn and O'Malley [VO05] provide a complete historical and contemporary context of the Froude number). The dimensionless Froude number is defined as:

$$Fr = \frac{v^2}{gL} \tag{6}$$

where v is velocity, g is gravity, and L is a characteristic length. Alexander [Ale76] used the Froude number to describe a consistent relationship in many different animals, including humans, between velocity and stride length λ, where L is leg length:

$$\frac{\lambda}{L} = 2.3\left(\frac{v^2}{gL}\right)^{0.3} \tag{7}$$

Our system utilizes this relationship between stride length and velocity to automatically timewarp a motion. Equation 7 can be rearranged to describe the ratio between new and old velocities caused by a change in stride length:

$$\frac{v_{new}}{v_{old}} = \left(\frac{\lambda_{new}}{\lambda_{old}}\right)^{\frac{5}{3}} \tag{8}$$

Since only a stride length *ratio* is required for this calculation, actual stride length need not be measured from the motion at all. Instead, we approximate changes in stride length at any key by the proportional change in path edge length at that key. However, since the scale factors described in Section 3.2.1 are computed independently between each pair of adjacent path handles, the ratios of new-to-old path edge length can be discontinuous. To obtain a continuous and smooth edge length ratio, the segment scale factors are assigned to the arclength midpoint of each segment, and the intervening values are computed through cubic spline interpolation.

Using this continuous scale factor as an approximation of the stride length ratio, a new velocity can be calculated for each key. Since the starting time of the animation remains at time 0, these target velocities overconstrain the new key times $\mathbf{t} = \{t_0, t_1, \ldots, t_n\}$, which are calculated by minimizing the squared error between the target and new velocities (calculated through central differences):

$$\underset{\mathbf{t}}{\operatorname{argmin}} \sum \left(v_{new_i} - \frac{\|\hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_{i-1}\|}{t_{i+1} - t_{i-1}}\right)^2 \tag{9}$$

4. Motion Components

The basic approach in the previous section treats all parts of the motion uniformly. For anything but the simplest motions, however, this approach is unsuitable because different parts of a motion should not be freely deformed or timewarped. For example, Kim et al. [KHKL09] identified interactions between multiple characters and constrained any path edits to maintain the relative character positions during interaction. We extend this notion to a variety of other *motion components* for single characters, which are easily identified automatically and can occur at any time as well as simultaneously.

4.1. Vertical Motion

Previous path deformation approaches [Gle01, KHKL09] as well as our basic algorithm operate solely horizontally, in two dimensions, which greatly restricts the user's editing

capability. While it may be possible to adapt as-rigid-as-possible deformation to work in three dimensions simultaneously, this would not generate realistic deformations resulting from vertical manipulations. This is due to the rotation-invariant local coordinate system (Equation 1): while motions in a horizontal plane can be easily rotated around a vertical axis, rotating motions in a vertical plane around a horizontal axis is not realistic (Figure 7).

Figure 7: *Side views of a path during a forward and upward step. Actual vertical adjustments of a motion path do not induce rotation of the path, but a different transformation.*

Instead of a rotation, a more appropriate transformation of a motion in a vertical plane is a shear, which retains the vertical orientation of a motion. This is accomplished by applying the as-rigid-as-possible algorithm a second time, to the horizontal and vertical components of a motion path, where the horizontal coordinates are fully constrained after being determined from the ground-plane-parallel two-dimensional deformation. In this vertical deformation, a different local coordinate system can be defined which uses a local semi-horizontal axis, and the global vertical axis:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + u_i(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) + v_i(0,1) \qquad (10)$$

This formulation results in local shears instead of rotations, as intended (Figure 8). However, the blending of local shears can result in tangent changes around path handles, which is undesirable since the handles were selected due to their local vertical minimality. This can be fixed by adding a single hard tangent constraint to the least squares system, which maintains path shape at handles.

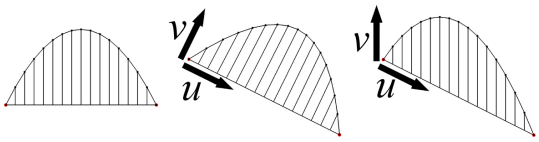

Figure 8: *Deforming an initial curve (left) using a fully local coordinate system produces rotations (middle). A mixed local/global coordinate system produces shears (right), which is more appropriate for gravity-oriented vertical motion.*

4.2. Ballistic Motion

Applying a deformation algorithm uniformly to a motion path can result in unrealistic deformations to non-contact or airborne segments of motion (Figure 9, left). To address this, our system considers all motion without any ground contact as ballistic motion, which should result in restricting a character's centre of gravity to a parabolic arc. We have found that a character's root provides a sufficient approximation to the centre of gravity without requiring the dynamics calculations to compute the character's mass distribution (e.g., as in Shin et al. [SKG03]). However, as an approximation, the root does not necessarily follow a parabolic arc. To address this, we restrict the transformation of ballistic path segments to those affine transformations which *would* preserve parabolas. During horizontal deformation (Section 3.2), simple hard constraints restrict the ballistic segment to a similarity transform (Figure 9, right). Since ballistic segments generally contain no side-to-side motion, this is roughly equivalent to allowing these segments to stretch and rotate, but not bend. During vertical deformation (Section 4.1), the entire ballistic segment is restricted to a single shear.

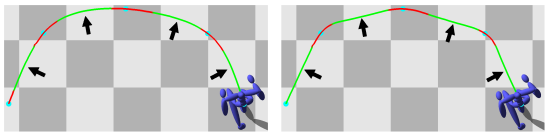

Figure 9: *Without appropriate constraints, ballistic motion can be deformed into impossible shapes (left). Constraining the deformation (right) allows the path to bend during contact but only scale/stretch during ballistic motion.*

While realistic ballistic motions should not be deformable horizontally, vertical deformations should be permitted: specifically, modifying the height of a jump without modifying the vertical positions of its enclosing path handles. This is accomplished by adding a vertical-only handle which uniformly scales the vertical displacement of each constrained key from the plane connecting the first and last unconstrained keys.

Following a constrained deformation of a ballistic path segment, the pose adjustment stage of our system (Section 3.3) works without modification. However, the velocity-based timewarping produces an unrealistic result on deformed ballistic segments, which is unsurprising as it is based on the behaviour of stride-based ground motion. Because ballistic segments are affected primarily by gravity, our system determines a timewarp for these segments which preserves the magnitude of each key's acceleration (i.e., gravity) rather than velocity. However, the same least squares formulation is not adaptable because this condition cannot be expressed in terms which are linear with respect to key times, and we have found that nonlinear solutions for the timewarp may not operate at interactive rates.

Therefore, we determine a full timewarp by combining local uniform timewarps centered at each ballistic key. We utilize a central differences approximation of the acceleration

at key i:

$$accel_i = \frac{1}{\Delta t_{i-1} + \Delta t_i} \cdot \left(\frac{1}{\Delta t_i} \cdot \Delta \hat{\mathbf{p}}_i - \frac{1}{\Delta t_{i-1}} \cdot \Delta \hat{\mathbf{p}}_{i-1} \right) \quad (11)$$

where $\Delta t_i = t_{i+1} - t_i$ and $\Delta \hat{\mathbf{p}}_i = \hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i$. For each key, our system determines the local uniform timewarp of magnitude m which minimizes the squared distance between the new timewarped acceleration and the original acceleration $\mathbf{a}_i$:

$$\underset{m}{\operatorname{argmin}} \left\| \frac{1}{m^2} accel_i - \mathbf{a}_i \right\|^2 \quad (12)$$

Applying this timewarp to ballistic keys preserves accelerations correctly, but retaining the velocity-based timewarp for all contact keys leads to velocity discontinuities at the contact/ballistic transition. Therefore, we adapt the as-rigid-as-possible formulation to determine the final timewarp, similar to Kim et al. [KHKL09], though our process is fully automatic. All keys in a fully in-contact path segment between two handles have their relative times determined by velocity-based timewarping, as before. However, we do not constrain the times of "transition keys" preceding and following a ballistic segment up to the nearest handle; therefore, a one-dimensional application of the as-rigid-as-possible algorithm determines a smooth timing transition between the velocity-based contact timewarps and the acceleration-based ballistic timewarps.

4.3. Turns

While the stride length-based timewarping described in Section 3.4 is effective in many situations, it can yield unrealistically fast motion during turns. To address this, we incorporate an additional timewarping criteria based on the "one-third power law", a commonly-used model in biomechanics which relates instantaneous velocity v and path curvature κ:

$$v = c \cdot \left(\frac{1}{\kappa} \right)^{\frac{1}{3}} \quad (13)$$

where c is often referred to as an empirically-determined "velocity gain factor". This law intuitively results in an inverse relationship between path curvature and velocity, i.e., sharper turns occur slower. To evaluate path curvature in practice, this law is generally applied not to the high-frequency actual path of the kinematic root or centre of mass, but to an ideal path which is being actively followed.

To approximate this, we seek to determine an abstraction of the root path, an "overall path" which incorporates only the lower-frequency aspects of the root's motion. Directly accomplishing this by frequency filtering the root path would require parameter tuning which may need to be motion-specific, and as such is undesirable. As well, we seek to determine an overall path that is curvature continuous since recent biomechanics research indicates that human walking paths are well-described by curvature continuous paths [ALHB08].

We have found a simple but effective solution to be a natural C2 cubic spline which interpolates only the path manipulation handles. This results in a path which ignores the higher-frequency oscillations like hip sway between handles but still describes the complete root path well, is curvature continuous, and is efficient to compute (Figure 10). We follow the common approach in one-third power law literature and determine curvature at a key by evaluating the curvature at the nearest point on the overall path.

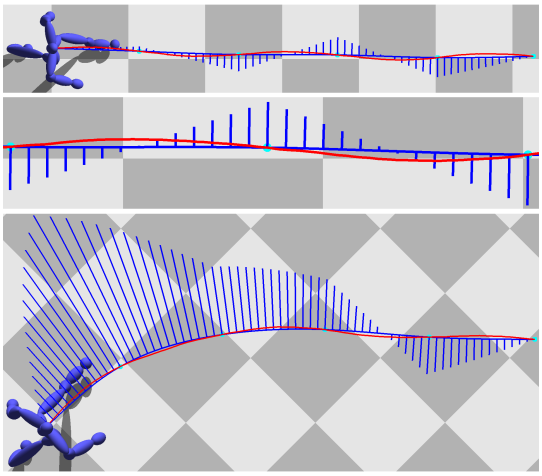

Figure 10: *Top: The overall path (blue) fits the actual root path (red) well and has low curvature (blue normal lines) for forward motion. Middle: close up of higher-frequency hip sway ignored by the overall path. Bottom: overall path and curvature for a turning walk.*

The one-third power law is generally applied in a piecewise fashion to ranges of curvatures, and clearly contains a singularity at $\kappa = 0$ which prevents the law from applying in all situations. However, by approximating κ with $\varepsilon + \kappa$ where $\varepsilon > 0$, and assuming that path deformations do not affect the velocity gain factor, we can determine the ratio between old and new velocities as a function of curvature with no singularities:

$$\frac{v_{new}}{v_{old}} = \left(\frac{\varepsilon + \kappa_{old}}{\varepsilon + \kappa_{new}} \right)^{\frac{1}{3}} \quad (14)$$

The two power laws presented in Equations 8 and 14 are generally mutually exclusive, as the Froude number-based and curvature-based velocity laws describe straight-ahead and turning movements, respectively. However, recent work by Bennequin et al. [BFBF09] presented a theory of movement timing which combines multiple velocity predictions based on different geometric properties. Therefore, we formulate our two velocity rules as follows:

$$v_{Fr} = \left(\frac{\lambda_{new}}{\lambda_{old}} \right)^{\frac{5}{3}} \cdot v_{old} \quad (15)$$

$$v_\kappa = \left(\frac{\varepsilon + \kappa_{old}}{\varepsilon + \kappa_{new}} \right)^{\frac{1}{3}} \cdot v_{old} \tag{16}$$

and combine both of these velocity predictions with the following product:

$$v_{new} = (v_{Fr})^{\beta_{Fr}} \cdot (v_\kappa)^{\beta_\kappa} \tag{17}$$

where $\beta_{Fr} + \beta_\kappa = 1$ and $\beta_{Fr}, \beta_\kappa \geq 0$. Bennequin et al. present a similar multiplicative form of velocity combination with a weighted combination through exponents which sum to 1, rather than an additive form. While their model's weights vary continuously according to geometric properties, we leave the adaptation of this derivation to future work, as we have found that a constant assignment of $\beta_{Fr} = \beta_\kappa = 0.5$ works well and generates realistic results that are also very similar to ground truth comparisons (see Section 5).

5. Results

We have tested our motion editing system on a variety of human locomotive motions recorded by optical motion capture at 120 frames per second. Most motions are loaded in under one second, an interactive framerate is maintained during editing without subsampling motion, and the pose adjustment and timewarping are performed quickly enough to be imperceptible. All of the following edits were performed interactively in single sessions of less than 30 seconds each.

Our system can easily edit motions which contain a mix of motion components. Figure 11 shows a motion containing a sequence of running, a jump, standing, and running, modified in a number of ways: the path has deformed into a zigzag pattern, the jump has been made shorter and higher, and the landing has been lowered.

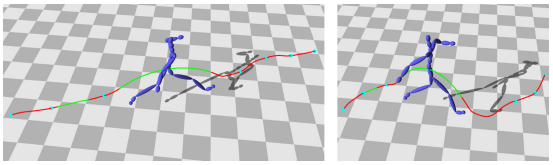

Figure 11: *A mixed motion of running, jumping, and standing, before (left) and after (right) a variety of path edits.*

In most cases, end effector constraints are constrained to horizontal translation to prevent ground interpenetration, and rotation about a vertical axis to prevent tilting or rolling the foot into an unfeasible position. However, removing the constraints on translation allows footplants to be automatically elevated along with a motion path. Figure 12 shows a level walking motion modified into a stair climbing motion.

Significantly different variations on a single motion can be quickly generated with our system. Figure 13 shows a single jumping motion which has been modified in both distance and height, with additional modifications to the crouching height before and after the jump to properly anticipate and follow-through on the exaggerated jump. The acceleration-based timewarping speeds up the lengthened ballistic motion but slows down the heightened motion to maintain gravity. The original motion took 3.45 seconds; the longer jump requires a higher forward velocity which results in a shorter duration of 3.14 seconds, while the higher jump necessarily requires more "hang time" and is 3.88 seconds in duration.

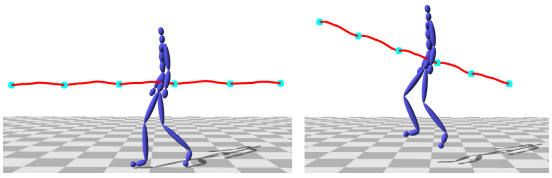

Figure 12: *Vertical translation of end effector constraints allows a walk (left) to be modified into a stair climb (right).*

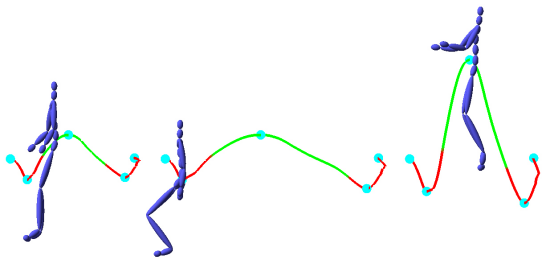

Figure 13: *A jumping motion shown unedited (left), lengthened (middle), and heightened (right). All motions are shown 2.3 seconds from their start.*

Non-human character motion can also be edited using our system, which requires only small modifications to the criteria for identifying manipulation handles on the motion path. For a forward walk cycle of a horse (Figure 14, left), double-stance periods of solely the rear legs are used to identify handles, since the skeleton root is located at the hips. Once these handles are identified, no further modifications to our system are necessary to edit the motion (Figure 14, right).

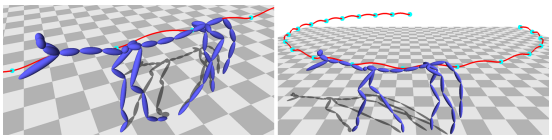

Figure 14: *A forward walk cycle of a horse (left) edited into a circular walk (right).*

Editing completely non-biomechanical character motion is also possible with our system. Motion sampled from a

simulated bouncing ball can be edited by identifying manipulation handles at all local minima without considering contact, i.e., at each impact. Once this is completed, each bounce of the ball is treated as a separate ballistic motion period (Section 4.2). Figure 15 shows a bouncing ball before and after a number of edits to the height and length of each bounce.

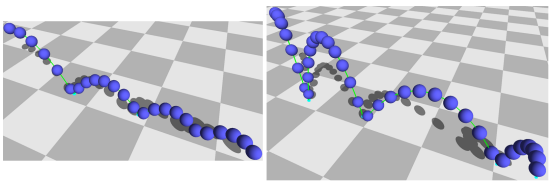

Figure 15: *A bouncing ball animation before (left) and after (right) several edits. Each motion is "strobed" at 0.1 second intervals; the original motion is 2.32 seconds in duration, and the edited motion is 3.55 seconds.*

Finally, our path deformation algorithm is able to accurately reproduce motion paths with very low error by modifying another motion from the same subject. Figure 16 shows two motions manually truncated to contain the same number of path handles, with the same foot forward at each corresponding handle. An automated edit was used to align the path handles of the source motion, a forward walk, with the corresponding handles of the target motion, a turn. Using the real-world scale from our motion capture data, we have calculated that the average distance between each key in the modified path and the nearest position along the target path is 0.5cm, and the maximum distance is 1.5cm.

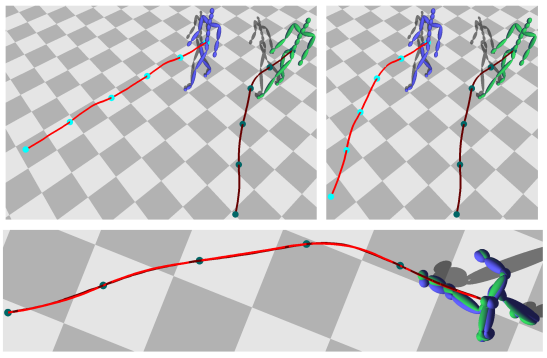

Figure 16: *A forward walk and turn compared before (top left) and after (top right) an automated alignment edit. The modified and target motion are overlaid (bottom) to compare their paths.*

6. Conclusion

We have presented a path-based kinematic motion editing system which extends the path deformation capabilities of previous work by identifying meaningful handles to restrict path deformations realistically and modifying the as-rigid-as-possible deformation algorithm to deform motion paths more realistically. Our method uses a timewarping technique which allows the user to concentrate on meaningful spatial edits, while motion timing is automatically modified to preserve the biomechanical relationship between path shape and velocity. Our system is extended by a number of motion components which are automatically identified and augment or restrict all stages of the editing process to maintain gravity-relative motion during vertical manipulations, preserve the shape and timing qualities of ballistic motion, and adjust turning speeds in a biomechanically correct way.

There are some limitations to our system. The velocity-based timewarping strictly enforces a general model for natural timing, but in reality, there is a range of plausible velocities for any path, with more extreme variations possible with deliberate effort. Allowing the user to modify timing within this feasible range would allow meaningful timing control without resorting to specifying absolute timing.

As well, there most likely are limits to how aspects of motion can be edited without incorporating dynamics calculations. More complicated acrobatic maneuvers would require a centre of mass to maintain realism. We have been unsuccessful at modeling foot placement during turns kinematically using path curvature and velocity, while simple dynamics calculations can modify foot placement accurately to maintain balance [SKG03].

This approach cannot edit aspects of a motion which are not significantly expressed by the root motion path. For example, the same motion in a variety of styles would differ in performance aspects such as extreme poses and limb timing, but would have similar motion paths. As well, static motions containing gestural components do not have a root motion path that can be meaningfully edited; however, perhaps our approach could be adapted to limb paths as well.

Our system also does not currently modify joints within the character skeleton except to satisfy end effector constraints. Allowing the upper body of a character to be adjusted would allow, for example, a walking character to swing their arms further as their strides lengthened, or a horse to bend its spine to lean towards the direction of a turn.

Acknowledgements

We thank the members and alumni of the Dynamic Graphics Project lab for valuable discussions, feedback, and technical assistance. We are also grateful to the anonymous reviewers for their comments. All human motion capture clips were from the Carnegie Mellon University motion capture database, and the horse motion capture data was provided by Centroid Motion Capture. Software for offline motion capture processing and the bouncing ball simulation was provided by Autodesk.

References

[Ale76] ALEXANDER R. M. N.: Estimates of speeds of dinosaurs. *Nature 261* (1976), 129–130. 5

[ALHB08] ARECHAVALETA G., LAUMOND J.-P., HICHEUR H., BERTHOZ A.: An optimality principle governing human walking. *IEEE Transactions on Robotics 24*, 1 (2008), 5–14. 7

[BFBF09] BENNEQUIN D., FUCHS R., BERTHOZ A., FLASH T.: Movement timing and invariance arise from several geometries. *PLoS Computational Biology 5*, 7 (July 2009). 7

[CBSG08] COLEMAN P., BIBLIOWICZ J., SINGH K., GLEICHER M.: Staggered poses: A character motion representation for detail-preserving editing of pose and coordinated timing. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2008). 2, 3

[Dav04] DAVIS T.: Algorithm 832: Umfpack v4.3 - an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw. 30* (June 2004), 196–199. 3

[Fro74] FROUDE W.: On useful displacement as limited by weight of structure and of propulsive power. *Transactions of the Royal Institution of Naval Architects 15* (1874), 148–155. 5

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *I3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics* (1997). 2

[Gle01] GLEICHER M.: Motion path editing. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (2001). 2, 4, 5

[HdSP07] HSU E., DA SILVA M., POPOVIĆ J.: Guided time warping for motion editing. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007). 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005). 3

[KHKL09] KIM M., HYUN K., KIM J., LEE J.: Synchronized multi-character motion editing. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (2009). 2, 3, 5, 7

[MCC09] MIN J., CHEN Y.-L., CHAI J.: Interactive generation of human animation with deformable motion models. *ACM Trans. Graph. 29* (December 2009). 2

[PPRN91] PATLA A. E., PRENTICE S. D., ROBINSON C., NEUFELD J.: Visual control of locomotion: Strategies for changing direction and for going over obstacles. *Journal of Experimental Psychology: Human Perception and Performance 17*, 3 (1991). 2

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007). 2

[Shn97] SHNEIDERMAN B.: Direct manipulation for comprehensible, predictable and controllable user interfaces. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces* (1997). 3

[SKG03] SHIN H. J., KOVAR L., GLEICHER M.: Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003). 2, 6, 9

[SYLH10] SOK K. W., YAMANE K., LEE J., HODGINS J.: Editing dynamic human motions via momentum and force. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), SCA '10. 2

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (2004). 2

[TGB00] TOLANI D., GOSWAMI A., BADLER N. I.: Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models 62*, 5 (2000). 5

[TM04] TERRA S. C. L., METOYER R. A.: Performance timing for keyframe animation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004). 2

[VO05] VAUGHAN C., O'MALLEY M.: Froude and the contribution of naval architecture to our understanding of bipedal locomotion. *Gait & Posture 21*, 3 (2005), 350–362. 5

[WDAC06] WANG J., DRUCKER S. M., AGRAWALA M., COHEN M. F.: The cartoon animation filter. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006). 2