To Appear at The 2001 ACM Symposium on Interactive 3D Graphics

Motion Path Editing

Michael Gleicher *

Computer Sciences Department, University of Wisconsin, Madison

Abstract

In this paper we provide methods that allow for path-based editing of existing motion data. We begin by exploring the concept of path as an abstraction of motion, and show how many of the common motion editing tools fail to provide proper control over this useful property. We provide a simple extension to displacement mapping methods that provide better control over the path in a manner that is easy to implement in current systems. We then extend this simple method to avoid violation of geometric constraints such as footskate.

Keywords: Animation, Animation with Constraints, Interaction Techniques

1 Introduction

Motion capture provides a method for obtaining motion for character animation that is physically realistic and conveys the personality of the performer. Unfortunately, while a captured motion can record the specific nuances and details of a performance, it also specifically records the performance. It encodes a specific performer, performing a specific action, in a specific way. Should any part of the motion not meet the users needs, either a different motion must be captured, or the existing motion must be transformed to meet the those needs. Recording new motion to "fill in the gaps" is difficult and expensive. Creating convincing transformations is difficult because they must retain the desired qualities of the motions while making changes to undesired aspects.

In this paper, we consider the problem of altering a previously captured motion to follow a different path. We introduce methods for editing of the path of a motion. For example, the motion of a character walking in a straight line can be transformed to walk along a curved path in a manner that preserves as much of the detail and nuance of the original motion as possible. Such transformations are important in a variety of applications, such as using motions in new environments (walking around obstacles) or in dynamic applications (walking to a goal location). The most basic form of motion path editing extends current techniques to provide interactive manipulation of a motion. Motion path editing can be enhanced, like the methods it extends, to to preserve essential properties of the original motion by applying constraint-based techniques. We demonstrate how path transformations work with constraint-based approaches to provide an interactive method for altering the path of a motion. This leads to several useful applications.

A path is an abstraction of the positional movement of a character. The path encodes the direction of motion, which is different from, but related to, the orientation of the character as it moves along the path. This abstraction leads to the idea of representing a motion relative to the path, allowing the path to be altered and the motion to be adjusted accordingly. The methods we present maintain the relationship between the motion and the path.

This paper is organized as follows. We begin by describing an example of what our techniques are capable of and useful for (Section 2). This discussion both motivates our methods as well as discusses their relationship to existing techniques. We then introduce the abstraction of a path for a motion (Section 3) including methods for automatically creating it.

The most basic form of path transformation, presented in Section 4, can create a new motion that follows an arbitrary path and orients the character appropriately. However, this transformation may damage some of the fine details in the motion such as the crispness of footplants. Better results can be obtained by using constraint processing to explicitly maintain details, as described in Section 5. The motion is continuously updated as the user drags portions of the path. Even the most sophisticated of the methods presented works interactively in all of our examples. We conclude by discussing experiments with our prototype implementation.

2 Overview and Example

Consider the problem of creating an animation of a fashion model strutting down a curved catwalk. This walk has a distinctive character to it that is difficult to describe mathematically, but is readily recorded using motion capture. In fact, a model walking in a straight line is a standard example in a motion capture library since it shows how something as basic as a walk can require a range of specific examples. Unfortunately, the libraries typically contain only the model walking in a straight line, not along the curved path we require.

Had we known that we required a curved path during the capture, we could have captured the exact motion that we needed. Unfortunately, this would have required the foresight to know the path that would be needed, and in our circumstances would be impossible as we are limited to what is provided by the motion library. We are therefore faced with the task of transforming the straight-line motion to a curved path. The challenge is to create a new motion that keeps the distinctive character of the original walk, yet moves along the correct path.

The tools described in this paper can transform the model's walk to a curved path. The system defines a "path" – a curve that represents the abstraction of her motion. This path can be edited using the same interactive tools used to edit any other curve. In Figure 1, we represent the path as a cubic B-Spline with 6 knots and use a direct

*gleicher@cs.wisc.edu, http://www.cs.wisc.edu/graphics

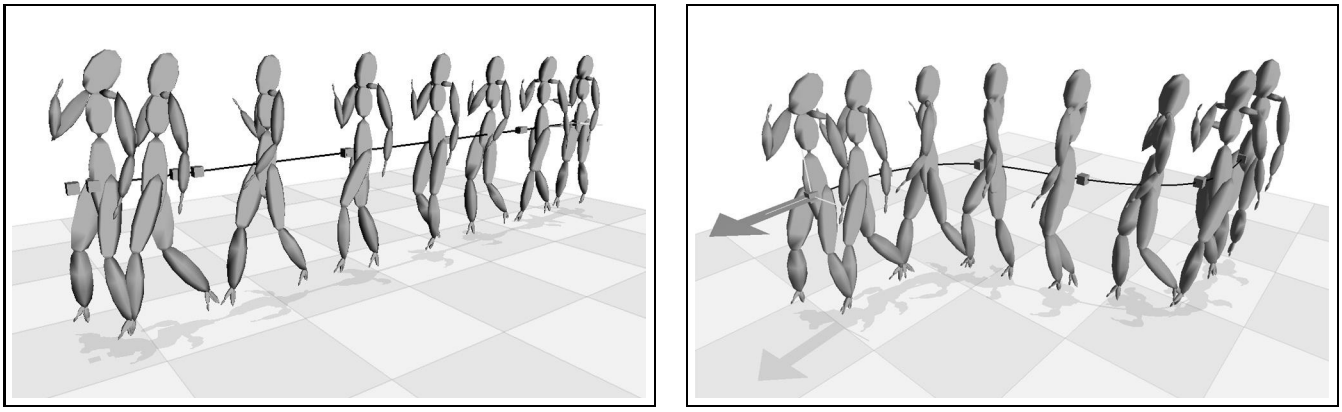

Figure 1: Editing the path of a walking motion. The left image shows the original motion, the right shows the edited result.

manipulation dragging technique to reposition the knots as the user drags points on the curve. As the path is changed, the motion is updated interactively.

In order to apply path editing, no information beyond motion capture data is required. Better results can be obtained by using a constraint solver to maintain features of the motion during dragging, or to clean up artifacts as a post-process. This requires constraints to be identified. Automatic techniques, such as the one presented by Bindiganavale and Badler [1] exist, but are imperfect. In practice, we identify constraints semi-automatically, manually refining the results of an automatic constraint detector[2].

Figure 2 shows a different example: the original performer danced in a circle, while we prefer a straight line. We transform the captured motion to that of a cartoon character (using the methods of [13] and then use the methods of this paper to straighten the path.

An advantage and limitation of the methods we describe are that they alter the path or an existing motion, rather than generate a new motion that better suits the path. In the catwalk example, the initial motion is six seconds long and consists of 8 steps. Because we are only transforming the existing motion, the resulting motion will have the same number of frames as the original motion. If the path is stretched, more steps will not be generated, although, we consider such generation in Section 7.2. While the method knows little of the mechanics of walking motion, and has no explicit description of the important properties of the motion, the results are usually reasonable, given a reasonable request. However, they are not necessarily what the performer would do in the situation. Better results could be obtained by providing more information to the constraint solver, however, our focus is on creating methods that are fast and interactive providing the opportunity to make adjustments afterwards[1].

2.1 Alternative Approaches

The ability to define motions along a path is a central feature of procedural and synthetic motion methods. Tools for generating a variety of locomotion methods along arbitrary paths have not only been demonstrated in research [3] [15] [17] (see [20] for a survey), but have been proven in products such as Life Forms [7] and Character Studio [16]. Unfortunately, these synthesis methods lack some of the qualities of captured motion: the ability to present variations such as personality and style without having to find mathematical methods for describing them. For example, while sophisticated synthesis techniques can generate realistic walking motions, it unlikely that they could create a convincing strut corresponding to a specific fashion model.

There are two strategies for creating a range of motions using motion capture data. One is to capture all desired motions and to use this large library in a production. The alternative is to have editing tools that can alter a base motion to meet the desired needs. Each strategy is limited: motion libraries cannot possibly predict every possible required motion (especially when there is a continuous range of motions that may be needed). These libraries are expensive to create and can become unwieldy and difficult to maintain as they expand to cover all needs. Despite recent innovations, there is still a limited range of editing that can be done that preserves the desired characteristics of motions. In practice, productions use a combination of approaches: a large library of motions provides examples that are close to what is needed, making the required edits easier.

Neither current editing techniques nor expanded libraries can adequately address the path alteration problem. Because there are an infinite variety of paths we might want to have a character move along, we cannot possibly capture all of them in a library. Current editing tools, including those based on inverse kinematics, signal-processing (such as motion blending [21], warping [30], and displacement mapping [4]) and constrained optimization (including the spacetime [12] [18] and physical methods [22]), also fail to address the problem of path transformation. Inverse-kinematics methods, by computing each frame individually, cannot affect properties over a long time scale, such as a path. The signal processing methods treat each individual signal separately and therefore cannot maintain the relationship between path direction and orientation over an entire character's motion. The optimization-based approaches need this relationship expressed in terms of constraints.

Since the earliest productions[19], animators have created characters by defining locomotion cycles and looping these cycles while the character moves along. Motion capture data can be converted to this form and applied along a path, however, this only works in simple, repetitive, straight-line motions. For example, the method used by Rose et al. [23] can apply a single cycle of a walk and run motion as a character is displaced along a path. The method provides no way to alter an existing complex path, as in the lantern example of Section 7.1.

A related approach would build longer motions of characters moving along desired paths from smaller pieces. Such an approach requires not only a library of the character making various turns but also methods for transitioning between the various different direc-

[1] For the examples in this paper, the results shown are the results of our methods. (No manual tweaking or post-processing is done unless explicitly stated.)

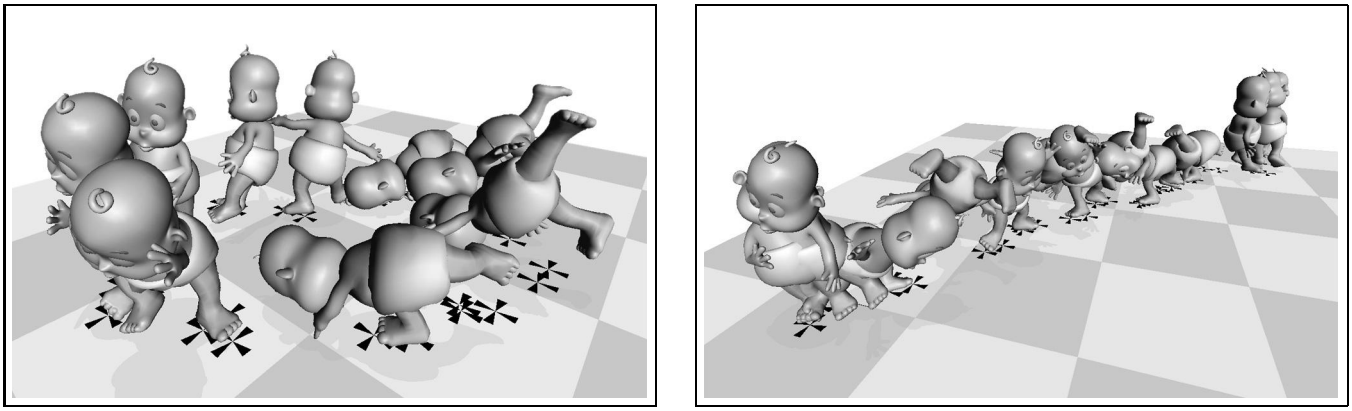

Figure 2: (also Color Figure) Editing the path of a dance motion. The original performance danced in a circle. This motion was applied to a cartoon character using retargeting methods (left). Path editing transforms this motion from a curved to a straight line. Dark marks are footplant constraints.

tional elements. The path control provided by the techniques in this paper makes such an approach more practical by providing a way to adapt the individual pieces to fit, as we will show in Section 7.2.

3 Paths

In practice, the user of our method sees the path as a curve generated by the system that provides a handle for control over the motion a character moves over. In this section, however, we provide a more precise and general definition of path, allowing the methods for path editing to be defined easily. Section 3.2 presents our specific technique for automatically computing the path.

The path is an abstraction of the positional aspects of the movements. The path does not contain the details of the characters motion, simplifying the character to a single point. We also prefer to ignore the smaller details of the movement, so that the path better fits an idealized description. In the previous example, the model's true motion is complex, however we prefer to think of it as strutting in an almost straight line. The path is intended as a conceptual tool for the user, not to encode the details of the motion.

Path is not an intrinsic property of a motion, but rather, a function of how a motion is interpreted. The same motion might be viewed differently in different circumstances: a person's movement might be interpreted as walking along a zig-zag, or zig-zagging in a straight-line. Abstraction in this way is often related to resolution or scale: depending on circumstance, we may or may not wish to consider the small details of a movement. The formalisms of multi-resolution [5] or scale space [28] consider a function at different levels of detail. However, because path is an abstraction to be interpreted by a user, we do not limit the path to be a filtered version of the original motion, but rather, allow it to be an arbitrary curve that relates to the motion. The important attribute of a path is that it is meaningful for the user.

A path is a time-varying space curve whose value is related to the position of the character at a given time. We will denote the path as the vector valued function $\mathbf{p}(t)$, and for convenience consider the corresponding translational transformation matrix $\mathbf{P}(t)$.

In the terminology of multi-resolution curve editing [8], the relationship between the path and the translational motion is the detail of the motion.

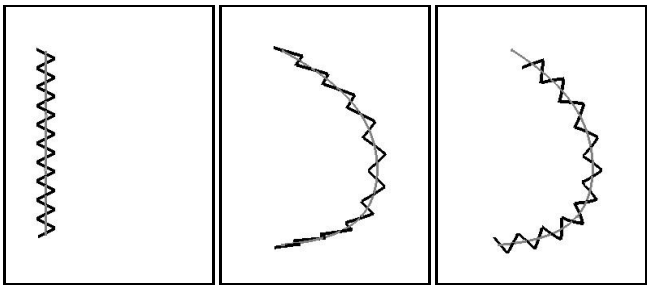

Figure 3: A character zig-zags relative to its path (left). If the details of the motion are represented in an absolute sense (e.g. left and right on the page), they may not apply if the direction of travel is changed (center). By representing the details relative to its path, the details can be preserved in different paths (right).

3.1 Direction of Path

Work on multi-resolution surfaces [9] and curves [8] discusses the importance of local detail relative to the geometry itself, rather that in some absolute coordinate system. Both works represent the details in a coordinate system defined differentially from the coarser version. We apply the same concept to paths.

At any instant a path has a direction of motion. Trivially, this is the direction of the tangent of time derivative of the path curve. This is neither the orientation of the character nor the instantaneous motion of the character. However, it is often convenient to discuss each of these details of the motion in terms of the path direction: we typically prefer to describe a character as zig-zagging side to side as they walk forward, rather than describing the path perturbations in an absolute way (e.g. north to south). This distinction becomes even more significant as we consider editing the path, as we would like to maintain the description of detail even if the direction of motion changes, as illustrated in Figure 3.

The preference for relative descriptions of detail is not absolute. While a character that periodically jumps eastward no matter what direction they are traveling (perhaps because they are called to a pilgrimage) seems contrived, the dominance of gravity on our experience causes the vertical direction to have specific meaning.

The articulations of the character are also best expressed in a convenient coordinate system. Typically, we prefer to describe the orientation relative to the direction of motion. Even if the character is not facing the direction of movement, it is the relative, not the absolute, orientation that is likely to be important. A character walking sideways is rotated with respect to their direction of motion, whether

their path is north-south or east-west.

A path, therefore, defines a coordinate system that moves in time. The coordinate system is centered at the position $\mathbf{p}(t)$, and oriented such that the details are represented conveniently. As we prefer a right-handed Y-up convention, we define the Y vector of the coordinate system to be upwards, the positive Z-direction to be the projection of the path's tangent on the horizontal, and the X axis to be their cross product. We will denote this orientation by the time varying matrix $\mathbf{R}(t)$. The path's coordinate system is given by $\mathbf{P}(t)\mathbf{R}(t)$, and the transformation from global to local coordinates is given by $\mathbf{R}(t)^{-1}\mathbf{P}(t)^{-1}$.

Note that $\mathbf{P}(t)$ and $\mathbf{R}(t)$ could have been independent curves. By relating them, we gain the ability to control both by simply altering one. We orient $\mathbf{R}$ with respect to the tangent of $\mathbf{P}$ and gravity. Alternatively, we might choose to define $\mathbf{R}$ by the Frenet frame [25] of $\mathbf{p}(t)$, a coordinate system defined using only differential properties. We consider the absolute-Y coordinate system to be superior for a number of reasons. First, it is defined whenever the character is in motion, while the existence of a Frenet frame requires the path to have non-vanishing curvature. Secondly, orientation is most typically determined relative to gravity: a character stands up while walking north, south, or up or down a hill. While this precludes banking into turns, such effects are more typically related to the physics of the situation and cannot be recreated by the geometry we consider. Finally, by using the rotation about a fixed axis, issues in representing rotations are trivialized.

Note: our Y-up system is not the same as simply assuming a level path and using the Frenet frame. In this latter case, the vertical axis would flip up or down depending on how the character turned.

3.2 Automatic Path Abstraction

Typically, we use the path as a handle to manipulate motion. For such an application, the goal of a path is to provide a curve that relates closely to the motion, yet allows control at the correct level of detail. An approximation at a level of detail convenient for manipulation can be created by filtering the translational motion of the character to produce a path. We usually compute paths from the trajectory of the global offset of the articulated figure. Alternatively, we might use the character's center of mass, or another point on the character. When dealing with two characters moving together (such as dancing), we choose the average position of the two characters.

The motion capture data is filtered to create the initial path. We implement this by computing a least-squares fit of a piecewise polynomial curve. In all of the examples of this paper, we choose uniform cubic B-Splines with a number of knots chosen for ease of manipulation. In practice, it is best to space the knots equally in arc-length, rather than in time (parameter-space). If the arc-length spacing is not used, the paths often have ill-defined derivatives in regions where the character stops, causing difficulties in manipulation.

Because errors in the fitting process will be "kept" as detail in the motion, it is not essential that the path fits the motion well. It is important that the relationship between the path and the motion is clear to the user. Unfortunately, artifacts of the scale tangent computation may sometimes leads to non-intuitive results. For example, if a character pauses, this may cause the tangent to spin around, which can cause unusual effects when the path is edited.

4 Path Editing

Given the path corresponding to an initial motion, we can factor the motion into the path and a residual by placing the motion in the moving coordinate system defined by the path. We can then replace the path curve to produce a motion that follows the new path. We will denote the initial path with a 0 subscript.

It is common practice in animation systems to allow a character to be placed in a moving coordinate system, inserting transformations "above" the character in a kinematic hierarchy. In commercial systems this can be done by placing the character in a group whose transformation can be animated.

Because we define the initial motion in the coordinate system of the initial path, the coordinate system for the character is given by:

$$\mathbf{P}(t)\mathbf{R}(t)\mathbf{R_0}(t)^{-1}\mathbf{P_0}(t)^{-1}. \quad (1)$$

If the method is applied to sampled motions, such as motion capture data, $\mathbf{R_0}(t)^{-1}$ and $\mathbf{P_0}(t)^{-1}$ can be stored as a table of samples. This enables a simple implementation that can be realized in the scripting languages of commercial animation systems. The four transformations can be applied to the character and animated accordingly. As the path $\mathbf{P}(t)$ is updated, the corresponding $\mathbf{R}(t)$ must be computed.

If we do not use the rotation of the frame, our method is exactly equivalent to using a motion displacement map [4] (also known as a motion warp [30]) on the translation of the character. The path serves as the displacement map.

Alternatively, one might view our method as the application of multi-resolution curve editing [8] to the root translation of a character. The difference is that rather than simply filtering out the detail of the curve, we also "filter" the detail of the character's movements to be added back in later. We use the curve to define not just a point position, but a coordinate system in which the details are represented. Also, we do not limit ourselves to only using low-pass filtering to create the abstracted curves.

Any methods can be used to alter the path curve, or define a new one. We provide the user with direct control of the path's control points as well as direct manipulation editing of the curve itself using a least-squares technique[10] [27].

4.1 Arc-Length Reparameterization

A path is a temporal entity as well as a geometric one. Each time determines a position and orientation. By parameterizing the path by time, altering the path allows control not only of the position but of the velocity as well, as shown in Figure 4. The beginning of the path corresponds to the beginning of the motion. Likewise the character reaches the end of the path when the motion ends.

Unfortunately, the temporal control is not always desirable. As the user performs geometric operations on the path, they may inadvertently alter velocities they want to preserve. Or they may break the motion by stretching parts of the motion such that the character cannot reasonably step from one footplant to another. The problem is even more pronounced when the path is replaced by a completely different curve. In such a situation, the parameterization of the new curve may not match the parameterization of the original path, especially if the new path is created using a purely geometric method.

The user interface provided for manipulating curves could assist the user in maintaining the desired velocities. Instead, we offer the option of arc-length parameterizing the new curve such that the velocity along the new curve is identical to the original.

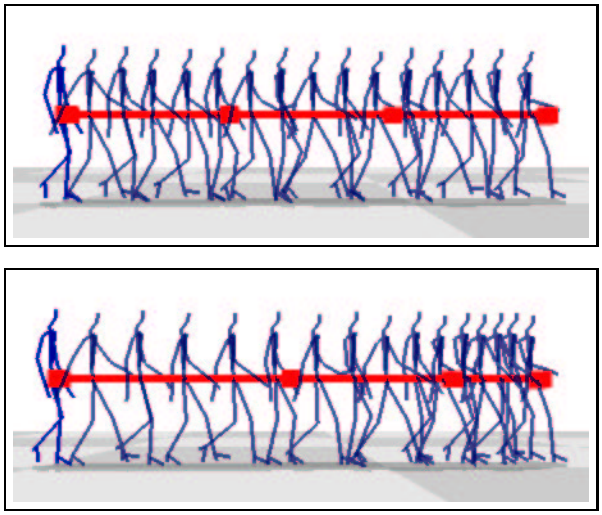

Figure 4: Path editing is used to alter the timing of a motion. The evenly spaced knots of the original B-Spline path are repositioned in the lower image.

If we map the character's position on the path by arc-length instead of time, we can provide a different interface which preserves the distance a character moves through time as we edit the path. Using this parameterization we can better maintain the dynamics of the motion. The velocity of the character matches the kinematics which produced the motion, and foot skate introduced by path transformations is reduced. Using this parameterization also allows the motion to be slid along the path by altering where on the curve it should begin. To efficiently approximate arc-length parameterizations, we use Euler step integration using distances between motion samples to approximate the tangent vector. This approximation is sufficient in practice because the path curves are usually smooth.

5 Constraints

The path transformation methods described preserve certain aspects of motion. For example, the joint angles are unaffected by the process, but the resulting positions of the end effectors are moved. Often the details of the end-effector movement are more significant to defining the motion, and therefore must be explicitly maintained during any transformations.

More problematically, path transformations change the trajectory of end effectors. While we would expect that a character's foot would be in a different place when its path is changed, the relative motions may be important – for example, the foot should remain stationary when planted. Because a path transformation may affect different times differently, points in the original motion that are stationary over a period of time may move along a curve as each instant during the duration is transformed differently. When the methods of the preceding section are applied to a motion, slipping along the floor is typically evident, especially when the path of the character is lengthened.

These problems are not specific to path transformation: any technique that transforms motion must consider what aspects are to be changed and what will be maintained. A common strategy is to identify specific geometric features of motions as constraints that must be maintained during the transformation. This strategy has been applied to a number of motion transformation problems, including interactive editing [12, 18], retargeting to new characters [13], and generation of transitions [24]. The methods that have been applied to these other problems may be applied to path transformations as well. In this section we survey some of the details specific to using constraint-based methods with path transformations.

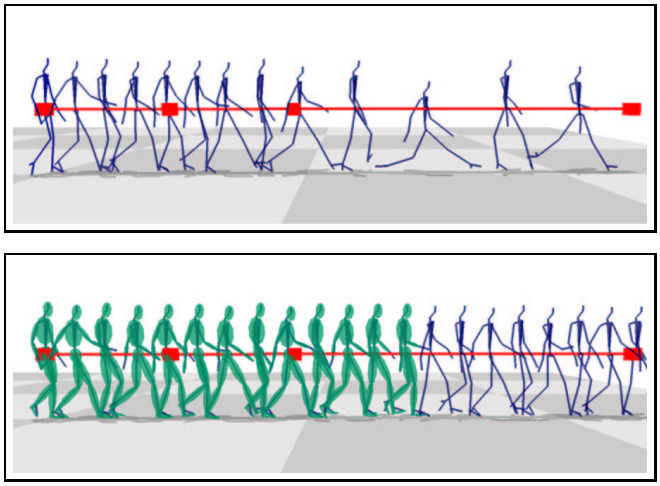

Figure 5: The path of a walking motion is lengthened. The last of 4 B-Spline knots is moved along the line. (upper) As the path lengthens the footplants separate until it is physically impossible for the character to step from one to the next. (lower) As the path lengthens the original motion (full body) is little changed. Additional frames (line figure) are added so that the motion arrives at the end of the path by repeating the cyclic motion.

We apply constraint processing in a fashion similar to [13]. That is, the path transformation is applied first to "mess up" the motion, and then the motion itself is altered to re-establish the constraints. Constraint processing does not change the path, just the details of the motion.

Geometric constraints on end-effectors may specify either absolute or relative positions. Absolute positions maintain the relationship between the character and some other object. For these constraints, the specific positions must be maintained. Relative position constraints maintain features relative to the character's motion and must be updated as the path is transformed. For example, the specific position of a character's footsteps may be moved, providing that the foot remains planted. This category of constraints must be handled specially when applying path transformations: a new position for the constraint must be determined that fits with the path.

For a relative constraint at a single instant of time, a new position can be computed just as it was for the character. However, since different instants in time may be transformed differently, there is no guarantee that a constraint that exists over a duration of time will preserve any properties. One particularly common problem is that stationary footplants tend to be spread over their duration. The simple cure for this is to choose a single time for each constraint, and transform the entire duration of the constraint by the transformation at this one instant.

Different choices in time lead to different constraint positions which, in turn, lead to different characteristics of the motions. The three most obvious choices are the beginning, middle, and end of the duration of the constraints. Figure 6 shows how this choice affects a walking motion. The differences are subtle, and illustrate the challenge of motion transformation: there is no clearly "correct" answer given the limited information that we have. We are left with either building more sophisticated models to determine the movement, relying on heuristics (or user input) to select amongst the myriad of small choices, or reconciling ourselves to never having the perfect answer.

One specific caveat: entire footplants must be dealt with at a sin-

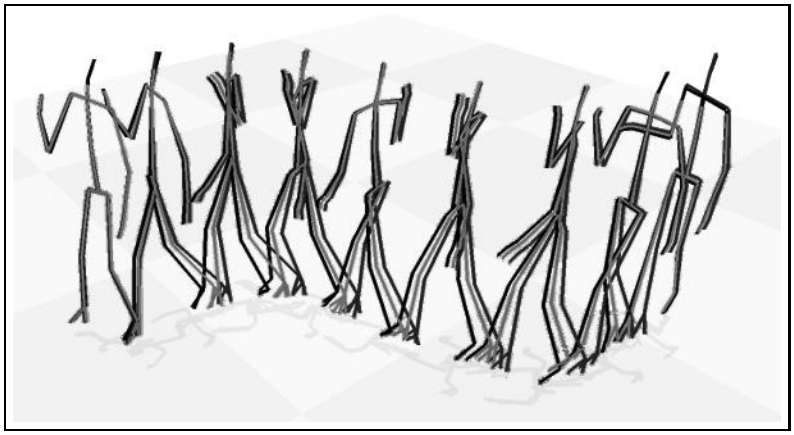

Figure 6: The example of Figure 1 is adapted using different methods for choosing the new footplant positions. The different colors represent the motion obtained by moving footplants to their beginning, middle or end of their duration.

gle instant, the heel and toe strikes cannot be given separate times. If they are transformed at the same instant, their positions will be transformed rigidly. Otherwise, the heel and toe strike positions may be transformed differently, changing the distance between them which is impossible if the character is a rigid skeleton.

An alternative to computing new locations for constraints is to express the constraints in a relative manner. For example, rather than specifying that a footplant is at a particular location, the constraints merely express that the position is constant over the duration of time. Such a formulation has the advantage that the solver can adjust the locations based on other criteria. The disadvantage is that by coupling different times, simpler solution methods that consider individual instants in time independently are not applicable; see [14] for details.

5.1 Constraint Solution Methods

With the constraint positions determined, we are left with the task of computing a new motion that meets the constraints. A variety of approaches have been presented in the literature, and are surveyed in [14].

The simplest approach is to solve the constraints at each instant in time individually. Such an approach has the advantage that it solves a series of small kinematic problems. A wide range of techniques including direct solution [26] and iterative numerical methods [11] have been applied to these inverse kinematics problems, and solutions are widely available. The problem is that solving each problem frame as an independent problem is that there is no way to maintain consistency across the subproblems.

Many of the consistency issues come from constraint switching: the instant before a constraint (or after the constraint) the solver has no way to prepare for the future. For example, if a constraint requires a footplant to be in a particular place, unless the frames leading up to this need are suitably altered, the foot would jump to its goal location at the beginning of the constraint. Other consistency issues stem from the fact that there may be multiple solutions to the constraint problems or that consistent answers may not be computed in response to subtly different inputs.

Inconsistency among individual per-frame constraint solutions leads to high-frequency artifacts in the resulting motion. In practice, we find that such an approach unacceptable as we see limbs "snapping" to "hit" footplants, and we see jitter introduced as solver imprecision gives slightly different answers on each frame. A method for dealing with interframe consistency is a requirement.

Approaches to interframe consistency that do not allow for constraint switching, such as that used in the online motion retargeting system of Choi and Ko [6] or Bindiganavale and Badler [1], do not apply to path transformations. Such approaches rely on known continuous paths for the end effectors. Using this with path transformations by path-transforming the initial end-effector positions would constrain the motion to have foot skate.

Spacetime constraint methods consider larger spans of the motions and therefore can provide solutions that have interframe consistency. While the approach was originally introduced for physical motion generation [29], more recent work has applied the techniques to motion editing tasks. The original spacetime approach of defining a single constrained optimization that computes the new motion was employed by Rose et al.[24] to generate transitions between motions, by Gleicher for interactive editing of motions [12] and retargeting [13], and by Popovic and Witkin [22] to create physically realistic motion transformations. Lee and Shin [18] introduced an alternative solution method that computes a constraint solution on a per-frame basis and then uses a B-Spline fitting process to combine these into a consistent result.

We have implemented both spacetime and per-frame ik plus filtering (PFIK+F) methods in our system. The tradeoffs between the methods are described in [14]. The PFIK+F methods are simpler to implement, but do not offer the generality of the spacetime methods.

6 Recipe

To summarize, the steps of the path transformation process are:

1. Augment the motion with constraints that maintain any essential geometric features. Typically, this is done when motions are placed into our library such that the effort can be amortized across many uses of the motion.

2. Define the initial path $\mathbf{P_0}(t)$, either by filtering the character's motion, or by user specification. Determine $\mathbf{R_0}(t)$, from $\mathbf{P_0}(t)$.

3. Create a new path, most typically by using curve editing tools on the original. The corresponding orientations must also be determined.

4. Apply the transformation of Equation 1 to the initial motion.

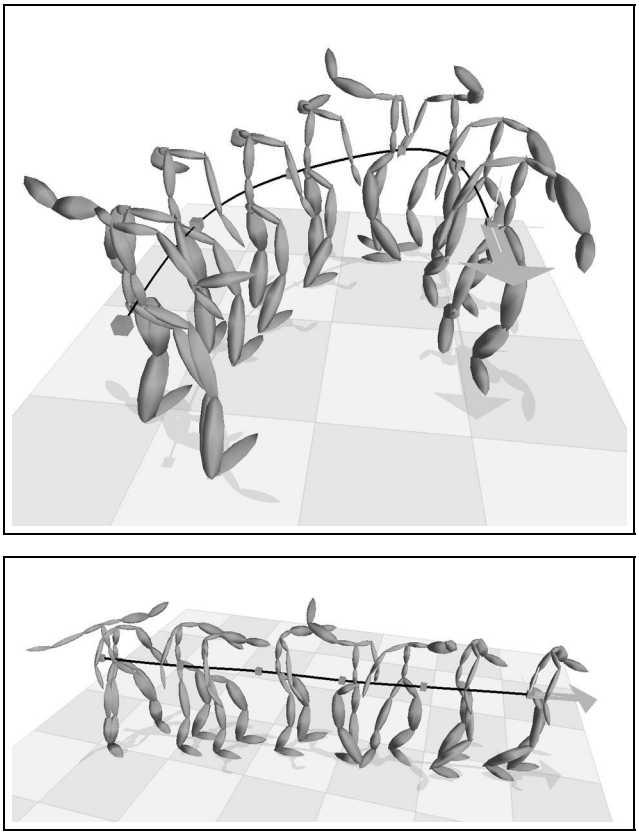

Figure 7: The top image shows a motion where the character searches a room while carrying a lantern. The lower image shows the results of transforming the path such that the character searches a straight corridor.

5. Apply a constraint solution technique to re-establish the geometric features of the motion. This step modifies the motion, not the path.

With fast constraint processing, steps 3, 4 and 5 can be interleaved in iteration with drawing to create interactive motion manipulation.

7 Examples and Applications

All examples described in this paper are run using our motion editing testbed running on personal computers under Windows NT. All example motion in this paper was originally created using optical motion capture.

7.1 Interactive Path Editing

To date, the primary use of our tools has been for interactive motion editing. The illustrations of this paper show some of the examples that we have used with our system.

Most of the examples presented transform a straight motion into a curved path. In Figure 7, a motion of a character searching around is adapted to work in a straight hallway. The turning of the character to look around is maintained, despite the straightening of the direction of travel.

The searching motion is among the longer of the experiments we have tried, consisting of 550 frames covering 18 of the characters steps. Over 3700 scalar constraints are used for the footplants. Our prototype can achieve a 30Hz refresh rate during interactive dragging without constraint processing (but including arc-length adaptation), providing a motion that can be "cleaned-up" afterwards. To achieve interactive performance (5-10 Hz) with our real-time solver we use sparse constraint sampling and large tolerances as described in [12]. The results are acceptable for low-resolution applications and can be cleaned up as a post-process. All other examples shown run faster.

The motion of Figure 2 also changes a curved motion to a straight line. With only 250 frames in the motion, achieving interactive rates during constrained-dragging is not a problem with our current implementation. Because of the highly dynamic nature of the leaping and spinning dance, one would expect that our disregard for Newton's laws would cause many artifacts. Momentum is most likely not preserved when the path changes as the ballistic trajectory during leaps will be altered. However, the physics of the cartoon character performing this motion are so stylized to begin with, it is difficult to say which is less realistic.

7.2 Transitions

Long sequences of motions are typically created by connecting shorter segments. Making a realistic transition between two motions can be difficult if the motions are quite different. However, if the end of the first motion is similar to the beginning of the second, simple mechanisms can be applied to make the transitions. If the motions connect extremely well, they may be simply spliced together, or small changes may be accounted for by blending. Motion libraries are typically designed to support this by providing moves that connect well to one another.

The overall position and direction of a motion is important to preserve over a transition. Often, motion libraries contain families of "turning" moves to allow a character to change direction. Path editing reduces the need for such redundancy by permitting a single motion to serve a variety of needs. To connect two motions, we use a path transformation to insure continuity of the positional movement of the character. After bringing the overall movement together, blending is often sufficient to make effective transitions. We use constraint processing techniques to clean up geometric details, such as footplants after blending.

A cyclic motion is designed to transition to itself, permitting the generation of arbitrarily long motions. With an arbitrary path, a cyclic motion can be applied repeatedly until the entire path has been filled with the motion. Using this technique we can create arbitrarily complex paths for any given cyclic motion, or create motions whose velocities are not accelerated when the path is stretched, as shown in Figure 5.

8 Conclusions

Path transformations provide an important tool in enhancing the utility of motion libraries by permitting a single motion to be applied in a wide variety of settings. The basic methods are simple to implement and can provide an interactive tool for motion editing. When these methods are combined with constraint techniques, they can still provide for interactive performance while maintaining key features of the original motion.

The methods described in this paper are purely geometric: there is no notion of physics. This means that the methods might fail for motions that are highly kinetic, such as jumping, or even mak-

ing sharp turns when running. Because we rarely know the precise physical properties of animated characters or the physical laws of their worlds, fast, simplified methods, combined with the possibility of tweaking, seem to offer the most practical and useful tools.

The ability to factor the path from a motion may provide the opportunity to perform operations on motions independent of the path, for example, blending motions that had different directions. With the ability to create a range of movements decoupled from their range of paths, an approach to animation based on assembling pieces of motion as needed becomes practical.

Acknowledgements

Andrew Prock played an important role in the development of path transformations, implemented the initial version of arc-length parameterizations and transitions, and assisted with the initial draft of the paper. Alex Mohr and Andrew Gardner helped with the Timelines system and producing illustrations, Rob Iverson assisted with paper production. The baby model of Figure 2 was provided by Hou Soo Ming.

This research is supported by an NSF Career Award "Motion Transformations for Computer Animation" CCR-9984506, support from Microsoft, equipment donations from IBM and Intel, and software donations from Microsoft, Intel, Alias/Wavefront, SoftImage, Autodesk and Pixar. The motion used for the examples in this paper was generously donated by Digital Domain, House of Moves, and Mainframe Studios.

References

[1] R. Bindiganavale and N. Badler. Motion abstraction and mapping with spatial constraints. In Frank Crow and Stephen M. Pizer, editors, *Modeling and Motion Capture Techniques for Virtual Environments*, pages 70–82, November 1998.

[2] Peter Bodik. Automatic footplant detection inside flmoview. Student Summer Project Report Web Page.

[3] Armin Bruderlin and Thomas W. Calvert. Goal-directed, dynamic animation of human walking. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 233–242, July 1989.

[4] Armin Bruderlin and Lance Williams. Motion signal processing. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, Addison Wesley, August 1995.

[5] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Communications*, 31:532–540, 1983.

[6] Kwang-Jin Choi and Hyeong-Seok Ko. On-line motion retargetting. *Pacific Graphics '99*, October 1999. Held in Seoul, Korea.

[7] Credo Interactive. Life forms. Computer Program, 1999.

[8] Adam Finkelstein and David H. Salesin. Multiresolution curves. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 261–268. ACM SIGGRAPH, ACM Press, July 1994.

[9] David R. Forsey and Richard H. Bartels. Hierarchical B-spline refinement. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 205–212, August 1988.

[10] Barry Fowler and Richard Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics & Applications*, 13(5):43–49, September 1993.

[11] Michael Girard and Anthony A. Maciejewski. Computational modeling for the computer animation of legged figures. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 263–270, July 1985.

[12] Michael Gleicher. Motion editing with spacetime constraints. In Michael Cohen and David Zeltzer, editors, *Proceedings 1997 Symposium on Interactive 3D Graphics*, pages 139–148, apr 1997.

[13] Michael Gleicher. Retargeting motion to new characters. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 33–42. ACM SIGGRAPH, Addison Wesley, July 1998.

[14] Michael Gleicher. Comparative analysis of constraint-based motion editing. In Hyeong-Seok Ko and Normal Badler, editors, *Proceedings of the Symposium on Human Modeling and Animation*, June 2000.

[15] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 71–78. ACM SIGGRAPH, Addison Wesley, August 1995.

[16] Kinetix Division of Autodesk Inc. Character studio. Computer Program, 1997.

[17] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996.

[18] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of SIGGRAPH 99*, pages 39–48, August 1999.

[19] E. G. Lutz. *Animated Cartoons: How They are Made, Their Origin and Development*. Charles Scribner's Sons, 1920. Reprinted by Applewood Books, 1998.

[20] Franck Multon, Laure France, Marie-Paule Cani-Gascuel, and Giles Debunne. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation*, 10(1):39–54, January - March 1999. ISSN 1049-8907.

[21] Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, March 1995.

[22] Zoran Popovic and Andrew Witkin. Physically based motion transformation. *Proceedings of SIGGRAPH 99*, pages 11–20, August 1999.

[23] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5), September – October 1998. ISSN 0272-1716.

[24] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 147–154, August 1996.

[25] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish Press, 1990.

[26] Deepak Tolan, Ambarish Goswami, and Norman Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *submitted for publication*, 2000.

[27] William Welch, Michael Gleicher, and Andrew Witkin. Manipulating surfaces differentially. In *Proceedings, Compugraphics '91*, September 1991.

[28] Andrew Witkin. Scale space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, 1983.

[29] Andrew Witkin and Michael Kass. Spacetime constraints. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 159–168, August 1988.

[30] Andrew Witkin and Zoran Popović. Motion warping. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 105–108. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

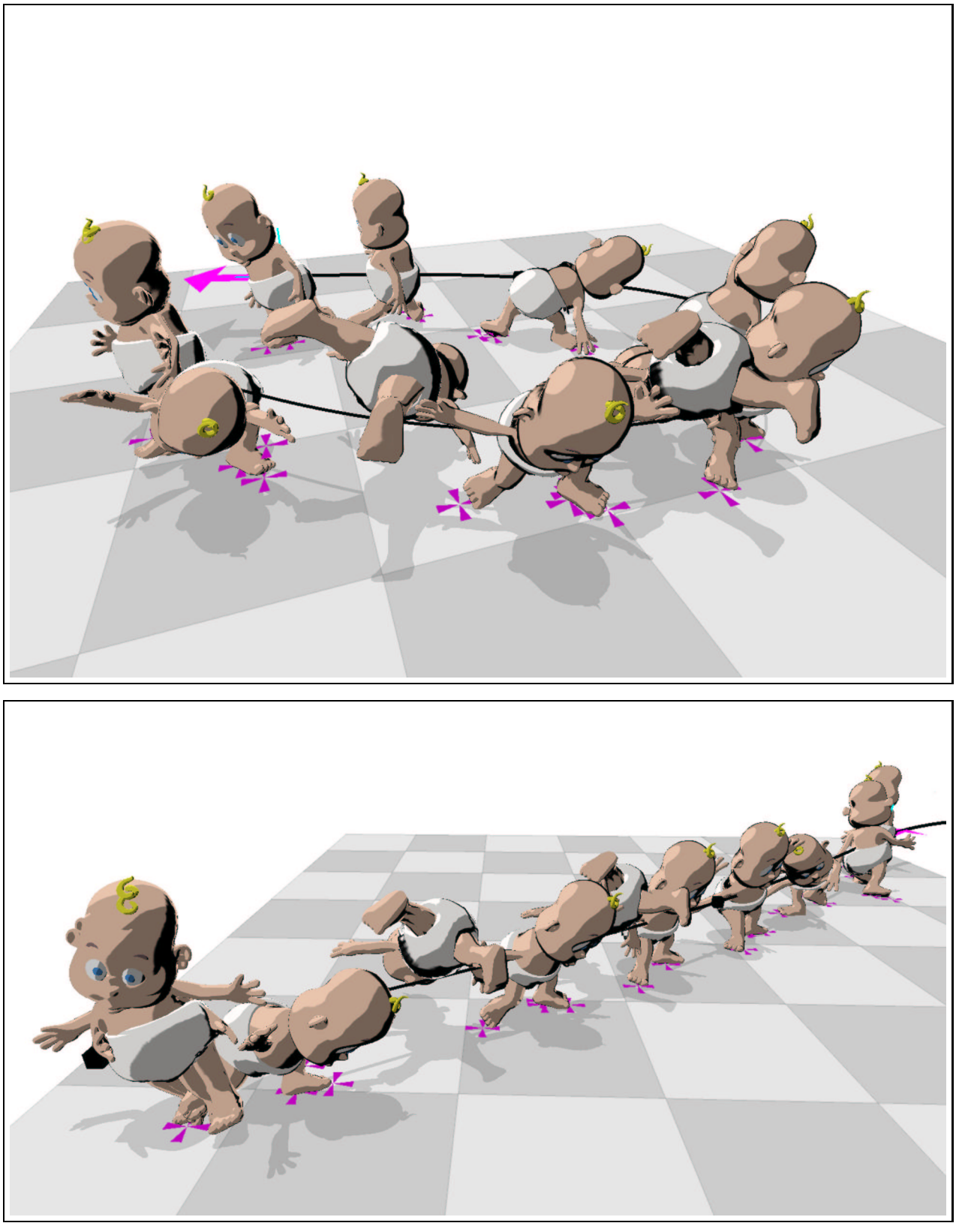

Figure 8: **Color Plate:** Editing the path of a dance motion. The original performance danced in a circle. This motion was applied to a cartoon character using retargeting methods (left). Path editing transforms this motion from a curved to a straight line. Dark marks are footplant constraints. The figure shows the interactive toon shading of our animation testbed.