Efficient Generation of Motion Transitions using Spacetime Constraints

Charles Rose
Princeton University
(currently at Microsoft Research)

Brian Guenter
Microsoft Research

Bobby Bodenheimer
Microsoft Research

Michael F. Cohen
Microsoft Research

Abstract

This paper describes the application of space time constraints to creating transitions between segments of human body motion. The motion transition generation uses a combination of spacetime constraints and inverse kinematic constraints to generate seamless and dynamically plausible transitions between motion segments. We use a fast recursive dynamics formulation which makes it possible to use spacetime constraints on systems with many degrees of freedom, such as human figures. The system uses an interpreter of a motion expression language to allow the user to manipulate motion data, break it into pieces, and reassemble it into new, more complex, motions. We have successfully used the system to create basis motions, cyclic data, and seamless motion transitions on a human body model with 44 degrees of freedom.

Additional Keywords and Phrases: computer animation, inverse kinematics, motion capture, motion control, human figure animation, cyclification.

CR Categories and Subject Descriptions: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism: Animation; I.6.3 [Simulation and Modeling]: Applications; G.1.6 [Constrained Optimization]; I.3.5 [Physically-Based Modeling].

1 Introduction

Existing 3D animation tools primarily provide support for creating a single linear stream of animation where the entire motion is planned in advance and computed off-line. Interactive 3D character animation, however, is characterized by a degree of uncertainty that is not present in animation for film or television. Characters are under the control of the user and must be able to change the way they move at any time. Crafting animation for an interactive application presents a new set of problems and requires a different set of specialized tools.

One solution for these problems is to generate a set of high quality motions, called basis motions in the remainder of the paper, and then create transitions between these motions so they can be strung together into animations of unlimited length and great variety. Basis motions are typically short and can be combined into motions representative of the type of actions that the body has to perform. For example, basis motions might be walk cycles, arm waves, karate kicks, and so forth. Basis motions need not specify all the degrees of freedom in the body; they can specify just those of one limb or even a part of a limb. Using motion capture techniques, it is relatively easy to make high quality basis motions, but generating high-quality transitions among those basis motions is still difficult and involves significant manual labor. The techniques presented work well with motion capture data, but would work equally well with hand-animated basis motions.

Email: {t-chuckr, briangu, bobbyb, mcohen}@microsoft.com

We have developed an algorithm for generating these transitions semi-automatically, greatly reducing the time spent and the number of parameters an animator must specify. The system provides two semi-automatic mechanisms for generating motion: motion transition generation and cyclification. Motion transition generation uses a combination of spacetime constraints [10] and inverse kinematic constraints [12] to generate transitions between basis motions. A fast dynamics formulation makes it practical to use spacetime transition generation on high degree of freedom systems. With this dynamics formulation, the algorithm achieves the lower bound time complexity for spacetime algorithms that use gradient based optimization techniques.

The motion transitions satisfy both dynamic and kinematic constraints. This differs from the work described in [2, 11, 9]. These papers described various mechanisms, such as dynamic time warping, Fourier interpolation, and multi-resolution signal processing, for transforming existing motion data. Transitions between motion clips were achieved using linear combinations of the two motions being transitioned. This can result in motion which does not have realistic dynamic qualities and which may not necessarily satisfy kinematic or anthropomorphic constraints. The transition mechanism described here generates motion which minimizes the torque required to transition from one motion to another while maintaining joint angle constraints. Inverse kinematics are used to ensure that kinematic constraints are satisfied.

Additionally, we have defined a motion expression language to allow the user to manipulate motion data, break it into pieces, and reassemble it into new, more complex motions. We have successfully used the system to create basis motions, cyclic data, and motion transitions on our human body model which has 44 degrees of freedom.

Section 2 of the paper describes the human body model and how motion capture data is processed before it gets into the system. Section 3 describes the semi-automatic spacetime and inverse kinematic transition mechanism. This section also explains the fast dynamic formulation which allows spacetime constraint optimization to run quickly on systems with many degrees of freedom. Section 3 also describes our method of motion cyclification. Section 4 explains how body motion is internally represented and describes the motion expression language. Section 5 presents results of transitions generated with the system. Section 6 concludes the paper. Appendix A contains an explanation of the notation used in the dynamics formulation, the dynamics equations, and their partial derivatives as they are used in the spacetime optimization.

2 Human Body Model

Before motion data can be edited by the system it usually needs to be preprocessed to remove degrees of freedom which are not actually present in humans. Most motion capture data, for example, has three rotational degrees of freedom at each joint. A handful of human joints actually have three degrees of freedom but most have one or two. Anatomically extraneous degrees of freedom cause trouble when generating motion transitions since the synthetic body may move in ways that are impossible for a real human. We use an optimization procedure [12] which minimizes the angular and positional deviation between our internal model of a human and the motion capture data. Limb lengths are automatically extracted from the motion data and used to scale the internal model appropriately. Our human body model has 38 joint degrees of freedom and six degrees of freedom at the root, located between the hips, for positioning and orienting the entire body. As with most other animation work on human body models we assume that human joints can be accurately modeled as revolute joints. While this is not precisely the case, especially for the knee and the shoulder, for most joints the errors introduced by making this assumption are typically small.

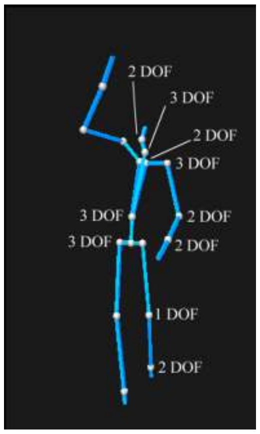

Figure 1: Human body model illustrating degrees of freedom.

3 Semi-Automatic Motion Generation

Two primary semi-automatic motion generation capabilities are provided. The first is motion transition generation and the second is motion cyclification. Motion transition generation creates new motion to span undefined regions between two basis motions. Motion cyclification transforms a motion which may not be perfectly cyclic into one which is.

3.1 Function Representation

Representations used in the past for the joint angle function,

$$\mathbf{q}(t) = (q_1(t), \ldots, q_n(t))$$

where $q_i(t)$ is the angle of joint i at time t, include piecewise constant [10], B-splines [4], and B-spline wavelets [8]. B-spline wavelets show good convergence properties for spacetime optimization when the number of basis functions in a single degree of freedom is large, e.g., more than 20 or 30. Since the transitions we are generating are generally short, on the order of 1 second or less, good paths can be represented with 5 to 10 B-spline coefficients. Our experience has been that very few iterations are required to achieve convergence with a B-spline basis, so the extra complexity and computation of the B-spline wavelet basis was not justified. For these reasons we use cubic B-splines as the basis functions for $\mathbf{q}(t)$.

3.2 Motion Transitions

Motion transitions are generated using a combination of kinematics and dynamics. The motion of the root of the body is determined kinematically while the motion of all the limbs which are not supporting the body is determined using spacetime constraints. Limbs which support the body during the transition are controlled using an optimization procedure to solve the inverse kinematics problem over the entire transition time interval instead of just at one point in time. A support limb is defined as the kinematic chain from the support point (e.g., a foot on the floor) back up the kinematic tree to the root.

3.2.1 Root Motion

The x-z plane is defined to be coincident with the floor with the y axis pointing upward. The x and z components of the root position are interpolated based on either the velocities or accelerations available at the beginning and end of the transition. The y component of the root translation is linearly interpolated from the end of the first motion (at time t_1) to the beginning of the second motion (at time t_2). The root position in the x-z plane, $\mathbf{p}(t)$, during the transition time is

$$\mathbf{p}(t) = \mathbf{p}(t_1) + \int_{t_1}^{t} \left\{ \mathbf{v}_1 \left(1 - \frac{\alpha - t_1}{t_2 - t_1}\right) + \mathbf{v}_2 \frac{\alpha - t_1}{t_2 - t_1} \right\} d\alpha$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the vector velocities in the x-z plane of the root at time t_1 and t_2. This expression can be easily evaluated analytically and provides a C^1 path for the root. A C^2 path is more desirable and could be achieved by double integration of the accelerations of the root. However due to limitations in the motion capture process estimates of the acceleration are poor especially at the beginning and end of a motion capture data stream.

3.2.2 Inverse Kinematics

Support limbs are controlled kinematically. The system attempts to locate support points automatically by finding coordinate frames with motion that remains within a small bounding box over an extended period of time. The animator has the option of overriding the system's guess, manually specifying a joint coordinate frame as being a support frame. During the transition this coordinate frame will be held fixed using inverse kinematics constraints. Enforcing kinematic constraints is done using an extension of the techniques presented in [12], optimizing for coefficients influencing a range of time. The inverse kinematics constraint is enforced by minimizing the deviation

$$r_k(t) = \|\mathbf{p}_k(t) - \hat{\mathbf{p}}_k(t)\|_2$$

of the constrained joint coordinate frame k from its desired position, where $\mathbf{p}_k$ is the actual position of the constrained coordinate frame and $\hat{\mathbf{p}}_k$ is the desired position. The total error, R, is given by the sum over all K constrained frames and integrated across the constrained time interval

$$R = \int_{t_1}^{t_2} \sum_{k=1}^{K} r_k(t)\, dt$$

Since a finite and usually small number of B-spline coefficients are used to represent the motion curves, minimizing this objective does not result in rapid oscillations. These can occur when inverse kinematic constraints are maintained independently at each frame time.

R is a function of the joint angles in the body

$$R = f\left(q_1(t), \ldots, q_n(t)\right)$$

which are themselves functions of the B-spline control points defining each joint angle function

$$q_i(t) = g(b_{i,1}, \ldots, b_{i,m})$$

where the $b_{i,j}$ are the control points of the B-spline curve for joint angle function $q_i(t)$. We minimize R using the BFGS optimization algorithm described in more detail in Section 3.2.3. For the purposes of the present discussion the only relevant part of the BFGS algorithm is that it requires the gradient of R at each iteration in the optimization

$$\nabla R = \left[\frac{\partial R}{\partial b_{i,1}}, \cdots, \frac{\partial R}{\partial b_{n,m}}\right]$$

$$\frac{\partial R}{\partial b_{i,j}} = \int_{t_1}^{t_2} \sum_{k=1}^{K} \frac{\partial r_k}{\partial b_{i,j}}\, dt$$

$$\frac{\partial r_k}{\partial b_{i,j}} = 2\left(\mathbf{p}_k(t) - \hat{\mathbf{p}}_k(t)\right)\left(\mathbf{u}_i \times \mathbf{d}_{ki}\right)\frac{\partial q_i}{\partial b_{i,j}}$$

where $\mathbf{u}_i$ is the axis of rotation of joint i and d_{ki} is the vector from joint i to the constrained frame k. Figure 2 shows the effect using the inverse kinematic constraint to fix the feet of the body on the ground during a motion transition period. The left leg is constrained to be on the ground during the entire transition interval while the right leg is constrained to be on the ground only at the end of the interval. In the image on the left the inverse kinematic constraints were turned off; the left leg drifts from its desired position and the right leg fails to meet its desired position. In the image on the right the inverse kinematic constraints are satisfied. The left leg remains fixed during the transition period and the right leg touches down at the end of the interval.

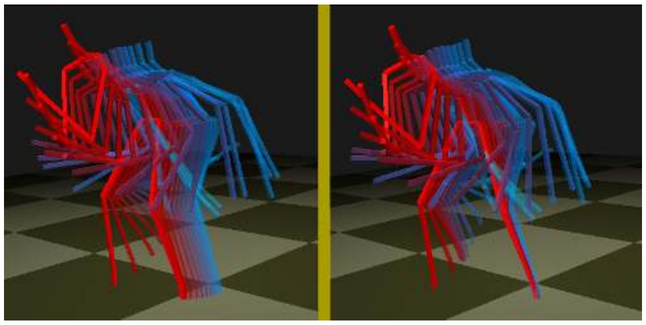
Figure 2: Effect of inverse kinematics constraint on placement of feet.

3.2.3 Spacetime Dynamics Formulation

The energy required for a human to move along a path is actually a complex non-linear function of the body motion since energy can be stored in muscle and tendon in one part of the motion and released later on. As shown in [3], joint torques are a reasonable predictor of metabolic energy, so minimizing torque over time should be a reasonable approximation to minimizing metabolic energy. Experience has shown that motion that minimizes energy looks natural. This leads to the minimization problem:

$$\text{minimize } e = \int_{t_1}^{t_2} \sum_i \tau_i^2(t)\, dt.$$

We use the BFGS optimization algorithm [5] to find a minimum of this integral equation. BFGS belongs to the class of quasi-Newton algorithms which progress toward a solution by using the gradient of the objective function

$$\mathbf{g} = \nabla e.$$

The gradient is used to incrementally update a matrix decomposition of a pseudo-Hessian matrix, $\mathbf{H}$, and to compute a new step direction

$$\mathbf{d} = -\mathbf{H}^{-1}\mathbf{g}.$$

The relative amount of computation for each subtask required at every iteration of the algorithm is common to several quasi-Newton algorithms: gradient computation, pseudo-Hessian update, and computation of the step direction.

Since each of the τ_i is potentially a function of all the q_i, $\dot{q}_i$, $\ddot{q}_i$ the gradient requires the evaluation of $\mathbf{O}(n^2)$ partial derivatives where n is the number of degrees of freedom of the body. This is in fact a lower bound for the asymptotic time complexity of space time algorithms which use gradient-based optimization techniques.

If m is the number of B-spline coefficients used to define the time function of each degree of freedom then the pseudo-Hessian is of size nm by nm. The update of the pseudo-Hessian and computation of the step direction are both $\mathbf{O}((nm)^2)$. For m small, less than 20, and n large, more than 30, the time required to compute g dominates all other computation thus an efficient formulation for g will pay the greatest dividends in reducing computation.

Computing g requires finding the joint torques and a variety of subsidiary quantities, such as angular velocity and acceleration. This is the inverse dynamics problem which has been extensively studied in the robotics literature. See [1] for a good overview of many of these algorithms. Many inverse dynamics formulations have been proposed in the robotics literature ranging from $\mathbf{O}(n^4)$ non-recursive to $\mathbf{O}(n)$ recursive algorithms. The inverse dynamics formulation we use is due to Balafoutis [1]. This is an $\mathbf{O}(n)$ recursive formulation which requires $96n - 77$ multiplications and $84n - 77$ additions to solve the inverse dynamics problem for a robot manipulator with n joints. This is faster than the $\mathbf{O}(n)$ Lagrangian recursive formulation developed by Hollerbach [6] and used in [7], which requires $412n - 277$ multiplications and $320n - 201$ additions.

The efficiency of the Balafoutis algorithm derives from the computational efficiency of Cartesian tensors and from the recursive nature of the computations. These efficiencies carry over to the computation of the gradient terms.

The Balafoutis algorithm proceeds in two steps. In the first step velocities, accelerations, net torques, and forces at each joint are computed starting from the root node and working out to the tips of all the chains in the tree. In the second step the joint torques are computed starting from the tips of the chains back to the root node. See the appendix for details. These recursive equations can be differentiated directly to compute g or one can use Cartesian tensor identities to compute the derivatives. Since the differentiation is tedious and somewhat involved we have included some of the partial derivatives of the recursive equations in the appendix as an aid to those attempting to reproduce our results.

3.3 Motion Cyclification

If cyclic motions, such as walking and running, come from motion capture data they will not be precisely cyclic due to measurement errors and normal human variation in movement. The discontinuities in motion will likely be small enough that the full power of a spacetime transition will not be necessary in order to splice a motion back onto itself smoothly. In this case we use a much simpler and faster algorithm for generating seamless cycles.

The cyclification algorithm proceeds in two steps. First the user marks the approximate beginning and end of a cycle. We create two time regions I_s and I_f centered about the markers the user has chosen. The time regions are set to be one-fifth the length of the time between markers. The system then finds one time point in each interval that minimizes the difference between position, velocity, and acceleration of the body:

$$\min_{t_1 \in I_s,\, t_2 \in I_f} \|\mathbf{a} - \mathbf{b}\|_2$$

where $\mathbf{a} = [q(t_1), \dot{q}(t_1), \ddot{q}(t_1)]^T$ and $\mathbf{b} = [q(t_2), \dot{q}(t_2), \ddot{q}(t_2)]^T$.

For most motions there will still be a discontinuity at the time where the motion cycles. We distribute this discontinuity error over the entire time interval so that the end points of the cycle match exactly by adding a linear offset to the entire time interval. We then construct a C^2 motion curve by fitting a least squares cyclic B-spline approximation to the modified motion.

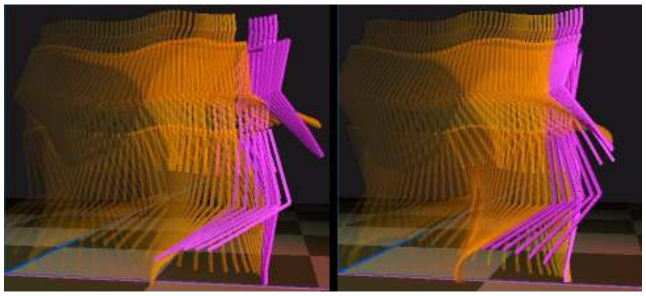

Figure 3: Results of cyclification on a walk

4 Motion Representation

To minimize the complexity of working with motions which involve many degrees of freedom we have developed a flexible functional expression language, and an interactive interpreter for the language, for representing and manipulating motions. Using this language it is a simple matter to interactively type in or procedurally generate complex composite motions from simpler constituent motions. We include a small example to demonstrate the simplicity of using the language.

Motions are represented as a hierarchy of motion expressions. Motion exressions can be one of three types of objects: intervals; degrees of freedom (DOF); and motion units (MU). These primitives are described in a pseudo-BNF notation below:

$$\begin{array}{rcl} \mathit{interval} & \rightarrow & (f_1, \ldots, f_n, t_s, t_f) \,|\, \epsilon \\ \mathit{DOF} & \rightarrow & \mathit{interval} \,|\, \mathit{DOF}, \mathit{interval} \,|\, \epsilon \\ \text{MU} & \rightarrow & \mathit{array}\, 1 \ldots n\, [\mathit{DOF}] \end{array}$$

An interval is a list of scalar functions of time plus a start time t_s and a stop time t_f. A DOF is a list of intervals. A DOF defines the value over time of one of the angular or translational degrees of freedom of a body. An MU is an array of DOFs which defines the value over time of some, but not necessarily all, of the degrees of freedom of the body.

There are three kinds of operations defined on these primitives: set operations, function operations, and insert and delete operations.

The set operations are intersection, undefine, and composition, denoted $\wedge$, $-$, and $+$ respectively. They are defined on intervals as follows (without loss of generality assume $t_3 > t_1$):

$$\begin{array}{l} I_1 = (f_1, \ldots, f_n, t_1, t_2) \\ I_2 = (g_1, \ldots, g_n, t_3, t_4) \end{array}$$

$$I_1 \wedge I_2 = \begin{cases} \epsilon & t_1 \leq t_2 \\ (f_1, \ldots, f_n, g_1, \ldots, g_n, t_3, t_2) & t_3 < t_2, t_2 < t_4 \\ (f_1, \ldots, f_n, g_1, \ldots, g_n, t_3, t_4) & t_3 < t_2, t_4 < t_2 \end{cases}$$

$$I_1 - I_2 = \begin{cases} (f_1, \ldots, f_n, t_1, t_2)(g_1, \ldots, g_n, t_1, t_2) & t_2 \leq t_3 \\ (f_1, \ldots, f_n, t_1, t_3)(g_1, \ldots, g_n, t_2, t_4) & t_3 < t_2, t_2 < t_4 \\ (f_1, \ldots, f_n, t_1, t_3)(f_1, \ldots, f_n, t_4, t_2) & t_3 < t_2, t_4 < t_2 \end{cases}$$

$$I_1 + I_2 = I_1 - I_2, I_1 \wedge I_2$$

where the comma denotes list concatenation.

The intersection operation takes two intervals as arguments and returns an interval. The undefine operator takes as arguments two intervals I_1 and I_2 and returns a DOF containing two intervals A and B. A diagrammatic representation of this operation is shown in Figure 4. The effect of the undefine operator is to undefine any portions of I_1 which overlap with I_2.

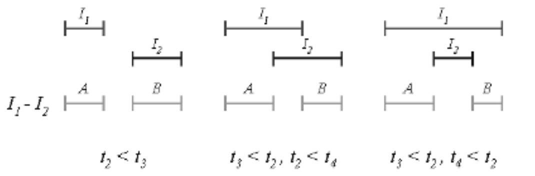

Figure 4: Interval undefine operation.

The set addition operator takes as arguments two intervals I_1 and I_2 and returns a DOF containing two intervals, A and B, if the intersection of I_1 and I_2 is empty, or three intervals, A, B, and C, if the intersection of I_1 and I_2 is not empty. A diagrammatic representation of this operator is shown in Figure 5.

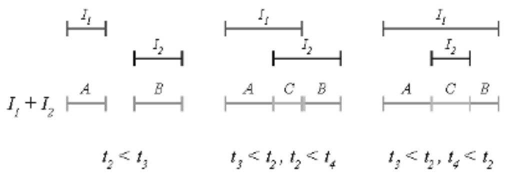

Figure 5: Interval addition operation.

The effect of this operator is to replace the region of I_1 that was removed by the set undefine operator with a new interval C. Set operations are useful for finding all the time regions over which two intervals, DOFs, or MUs are defined or those time regions where they are multiply defined, i.e., where composed motions conflict.

The function operations perform functional composition on the elements of an interval: functions and times. For example, one of the functional operators is $\mathit{affine}(t_s, d, I)$. This operator scales and translates the time components of an interval, and implicitly the time value at which the functions in that interval are evaluated. Other functional operations we have implemented include $\mathit{clip}(t_s, t_f, I)$ which clips out a portion of an interval and translates the beginning of the clip back to time zero, *clip-in-place*(t_s, t_f, I) which performs the same operation as clip except that it leaves the clip time unchanged, and $\mathit{concatenate}(I_1, I_2)$ which puts the two intervals in time sequence.

Both the set and function operations can be applied to any interval, DOF, or MU. The operations are pushed down to the level of intervals at which point the primitive interval operations are invoked. For example if we intersect MU_1 and MU_2, first we intersect all the DOFs of MU_1 and MU_2 and then we intersect all the intervals in all the DOFs.

Complex motions can be easily created by functional composition of simple motions using the set and function operations defined above. Figure 6 shows an example of a spacetime transition from a walk arm motion to a wave arm motion and then back to a walk arm motion. The arm wave MU defines only the DOFs of the arm and the walk MU defines all the DOFs of the body. First we perform an affine transformation on the arm wave MU and undefine this from the walk MU. This will undefine the arm degrees of freedom of the walk MU during a time that is slightly longer than the arm wave MU. When we add a time shifted version of the arm wave MU to the resulting MU there will be two undefined regions surrounding it which the spacetime operator will fill in with spacetime transition motion. The result will be a smooth transition from a walking arm motion to an arm wave motion and back to a walking arm motion. This operation is shown in both a time line view and an operator tree view in Figure 6. Letting SP denote the spacetime optimization, the algebraic representation of this motion is

$$SP\left(affine_2(\text{wave}) + (\text{walk} - affine_1(\text{wave}))\right).$$

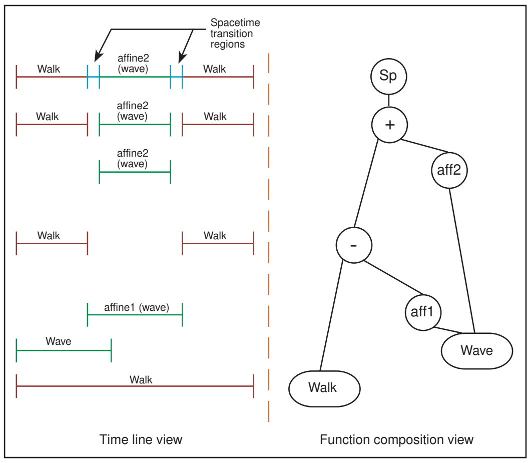

Figure 6: Spacetime composition of motions.

5 Results

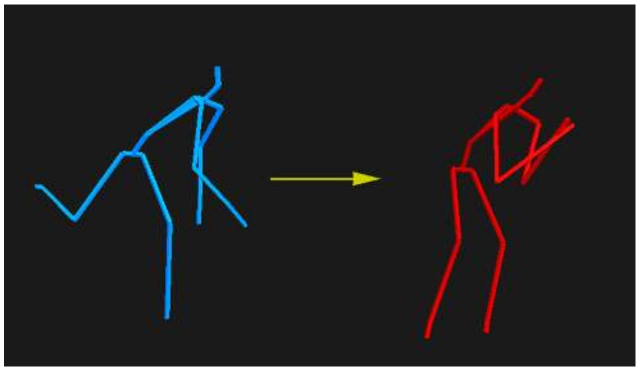

Figure 7: End position of motion 1 and beginning position of motion 2 for a motion transition

We have successfully applied the motion transition algorithm on many motions. For this example the transition time was set to .6, and the number of B-spline coefficients to 5. The resulting transition is shown in Figure 8. Our experience has been that successful transitions are quite short, usually in the range of .3 to .6 seconds. Without a biomechanical model to guide a large motion, our minimal energy model will often prove insufficient.

The beginning of the transition is colored blue and the end is colored red with intermediate times a linear blend of the two colors. This motion is one transition from a longer animation which has 5 transitions between 6 motions.

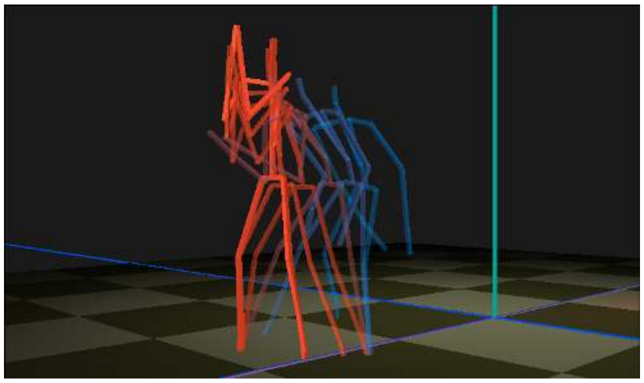

Figure 8: Multiple time exposure of transition generated from the motions in Figure 7

Figure 9 shows an example of a motion transition which affects only the arm degrees of freedom of the motion. This sequence actually consists of two space time transitions: one from a walking arm motion to the salute motion and another back to the walking arm motion. Each transition motion is .3 seconds long.

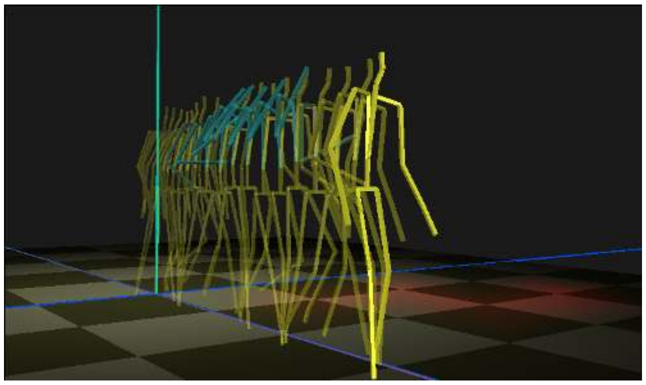

Figure 9: Arm walk motion transitioning to salute motion and back to walk motion. Arm degrees of freedom affected by the transition are colored green.

Computation times for transitions are strongly dependent on the number of degrees of freedom involved since the spacetime formulation we use is $\mathbf{O}(n^2)$ in the number of degrees of freedom. For the transition of Figure 8 generating the spacetime transition motion took 72 seconds. This transition involved 44 degrees of freedom. For the transition of Figure 9 generating the spacetime transition took 20 seconds. All timings were performed on a 100 MHz Pentium processor.

Spacetime transitions are more costly to generate than joint angle interpolation techniques, but they often produce more realistic motion. One type of motion that demonstrates this superiority is motion that has identical joint space beginning and ending conditions on some of the degrees of freedom of the figure. An example of this type of motion is shown in Figure 10. This motion begins with the forearm nearly vertical, held close to the shoulder with zero initial

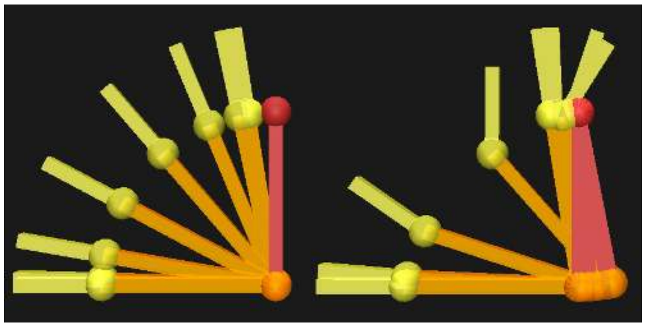

Figure 10: Joint angle interpolation versus spacetime optimization.

velocity. The motion ends with the forearm held horizontal also with zero velocity. Because the upper arm and the wrist have identical joint space starting and ending conditions any simple interpolation technique, which would include linear interpolation, polynomial interpolation, and most other types of interpolation which simply take a weighted sum of the two endpoint conditions will yield a motion such as that shown on the left in Figure 10. This is an unnatural motion since there is no joint space motion at the shoulder or the wrist. The spacetime motion, however, has motion at every joint and looks much more like the kind of motion a person might make.

6 Conclusion

This paper has presented a powerful animation system for manipulating motion data using a motion expression interpreter. Data can be positioned in space and time, and complete control of the degrees of freedom in the system allows motions to be spliced and mixed in arbitrary manners.

The system is capable of generating seamless transitions between segments of animations using spacetime constraints and inverse kinematics. Using a new, fast, dynamics formulation we can apply spacetime constraints to systems having a large number of degrees of freedom and still have reasonable computation time. An additional capability of the system is the generation of arbitrary length periodic motions, such as walking, by cyclifying segments of motion data which are approximately periodic.

The results of using our system to generate animations starting from a base library of soccer motions are quite good. Cyclification of a segment of such motions as a walk produces a quite realistic walk of arbitrary length. The spacetime constraint and inverse kinematic optimization produce transitions between diverse motions which are seamless and invisible to the viewer. While the optimization cannot be done in real time, it is relatively fast, and quite usable by an animator designing motions.

We plan to extend our motion model to more accurately model the dynamics of the human body model. The current approximation used for computing root motion works reasonably well for the class of transitions we have worked with but is not accurate for free body motion.

Acknowledgements

The authors thank Jessica Hodgins, College of Computing, Georgia Institute of Technology, for providing inertia matrices for the human body model. The Microsoft Consumer Division generously shared motion capture data with us for use in this project.

A Appendix: Dynamics Details

A.1 List of Symbols

$\mathbf{o}_i$	origin of the *i*-th link coordinate frame.
$\mathbf{c}_i$	center of mass of the *i*-th link.
$\boldsymbol{\omega}_i^i$	angular velocity of the *i*-th link.
$\mathbf{z}_i^i$	joint axis of the *i*-thlink expressed in the *i*-thcoordinate frame.
$\mathbf{s}_{i,j}^i$	vector from $\mathbf{o}_i$ to $\mathbf{o}_j$ expressed in the *i*-th coordinate frame.
$\mathbf{r}_{i,j}^i$	vector from $\mathbf{o}_i$ to $\mathbf{c}_j$ expressed in the *i*-th coordinate frame.
$\mathbf{A}_i$	3x3 coordinate (or 4x4 homogeneous) transformation relating the *i*-th coordinate frame to the $(i-1)$-th frame.
$\mathbf{I}_{c_i}^k$	inertia tensor of the *i*-th link about $\mathbf{c}_i$ expressed in the *k*-th coordinate frame.
$\mathbf{J}_{c_i}^k$	Euler's inertia tensor of the *i*-th frame about $\mathbf{c}_i$ expressed in the *k*-th coordinate frame.
$\boldsymbol{\Omega}_i^i$	angular acceleration tensor of the *i*-th link expressed in the *i*-th coordinate frame.
$\mathbf{F}_{c_i}^i$	force vector acting on $\mathbf{c}_i$ expressed in the *i*-th coordinate frame.
$\mathbf{M}_{c_i}^i$	moment vector about $\mathbf{c}_i$ expressed in the *i*-th coordinate frame.
$\mathbf{f}_i^i$	force vector exerted on link *i* by link $(i-1)$.
η_i^i	moment vector exerted on link *i* by link $(i-1)$.
τ_i	torque at joint *i*.
$\mathbf{g}$	gravity.
m_i	mass of the *i*-th link.

In the above, the subscript indicates the coordinate frame being represented and superscript the coordinate frame in which it is represented.

We use $+$ and $-$ on index variables to denote relative placement in the joint hierarchy. Thus, $i-$ is the predecessor of i which is the predecessor of $i+$. For example, in the equation $\boldsymbol{\omega}_{i+}^{i+} = \mathbf{A}_{i+}^T \boldsymbol{\omega}_i^i + \mathbf{z}_{i+}^{i+} \dot{q}_{i+}$, the variable $\boldsymbol{\omega}_i^i$ is the angular velocity in the coordinate frame which precedes the coordinate frame of $\boldsymbol{\omega}_{i+}^{i+}$. In other words, coordinate frame *i* is closer to the root coordinate frame than is frame $i+$. Note that there is no guarantee of a uniquely defined successor.

$$\mathbf{g} = [0.0, -9.80655, 0.0]^T$$

$$\mathbf{J}_{c_i}^i = \frac{1}{2}\,\mathrm{trace}\left(\mathbf{I}_{c_i}^i\right) 1 - \mathbf{I}_{c_i}^i$$

$$\mathrm{dual}(\mathbf{v}) = \tilde{\mathbf{v}} = \begin{bmatrix} 0 & -\mathbf{v}_3 & \mathbf{v}_2 \\ \mathbf{v}_3 & 0 & -\mathbf{v}_1 \\ -\mathbf{v}_2 & \mathbf{v}_1 & 0 \end{bmatrix}$$

$$\mathrm{dual}(\tilde{\mathbf{v}}) = \mathbf{v}$$

A.2 Forward Dynamics Equations

Base conditions at the root of the creature:

$$\begin{aligned}
\boldsymbol{\omega}_0^0 &= \mathbf{z}_0^0 \dot{q}_0 \\
\dot{\boldsymbol{\omega}}_0^0 &= \mathbf{z}_0^0 \ddot{q}_0 \\
\ddot{\mathbf{s}}_{0,0}^0 &= \mathbf{A}_0^T \mathbf{g}
\end{aligned}$$

Recursive forward dynamics equations:

$$\begin{aligned}
\boldsymbol{\omega}_{i+}^{i+} &= \mathbf{A}_{i+}^T \boldsymbol{\omega}_i^i + \mathbf{z}_{i+}^{i+} \dot{q}_{i+} \\
\dot{\boldsymbol{\omega}}_{i+}^{i+} &= \mathbf{A}_{i+}^T \dot{\boldsymbol{\omega}}_i^i + \tilde{\boldsymbol{\omega}}_i^{i+} \mathbf{z}_{i+}^{i+} \dot{q}_{i+} + \mathbf{z}_{i+}^{i+} \ddot{q}_{i+} \\
\boldsymbol{\Omega}_{i+}^{i+} &= \dot{\tilde{\boldsymbol{\omega}}}_{i+}^{i+} + \tilde{\boldsymbol{\omega}}_{i+}^{i+} \tilde{\boldsymbol{\omega}}_{i+}^{i+} \\
\ddot{\mathbf{s}}_{0,i+}^{i+} &= \mathbf{A}_{i+}^T \left[\ddot{\mathbf{s}}_{0,i}^i + \boldsymbol{\Omega}_i^i \mathbf{s}_{i,i+}^i \right] \\
\ddot{\mathbf{r}}_{0,i+}^{i+} &= \boldsymbol{\Omega}_{i+}^{i+} \mathbf{r}_{i+,i+}^{i+} + \ddot{\mathbf{s}}_{0,i+}^{i+} \\
\mathbf{F}_{c_{i+}}^{i+} &= m_{i+} \ddot{\mathbf{r}}_{0,i+}^{i+} \\
\tilde{\mathbf{M}}_{c_{i+}}^{i+} &= \left(\boldsymbol{\Omega}_{i+}^{i+} \mathbf{J}_{c_{i+}}^{i+} \right) - \left(\boldsymbol{\Omega}_{i+}^{i+} \mathbf{J}_{c_{i+}}^{i+} \right)^T
\end{aligned}$$

A.3 Backward Recursive Equations (Torque Equations)

At a joint controlling an end-effector:

$$\begin{aligned}
\mathbf{f}_i^i &= \mathbf{F}_{c_i}^i \\
\eta_i^i &= \tilde{\mathbf{r}}_{i,i}^i \mathbf{F}_{c_i}^i + \mathbf{M}_{c_i}^i \\
\tau_i &= \eta_i^i \cdot \mathbf{z}_i^i
\end{aligned}$$

At an internal joint:

$$\begin{aligned}
\mathbf{f}_i^i &= \mathbf{F}_{c_i}^i + \sum_{i+} \left[\mathbf{A}_{i+} \mathbf{f}_{i+}^{i+} \right] \\
\eta_i^i &= \tilde{\mathbf{r}}_{i,i}^i \mathbf{F}_{c_i}^i + \mathbf{M}_{c_i}^i + \sum_{i+} \left[\mathbf{A}_{i+} \eta_{i+}^{i+} + \tilde{\mathbf{s}}_{i,i+}^i \mathbf{f}_{i+}^i \right] \\
\tau_i &= \eta_i^i \cdot \mathbf{z}_i^i
\end{aligned}$$

A.4 The Energy Function and its Partial

$$e = \int_t \sum_i \tau_i^2 \, dt$$

$$\frac{\partial e}{\partial q_j} = \frac{\partial \int_{t_1}^{t_2} \sum_i \tau_i^2 \, dt}{\partial q_j} = 2 \int_{t_1}^{t_2} \sum \tau_i \frac{\partial \tau_i}{\partial q_j} \, dt$$

A.5 Forward Partials with Initial Conditions

$$\begin{aligned}
\frac{\delta \boldsymbol{\omega}_{i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta \boldsymbol{\omega}_i^i}{\delta q_j} \\
\frac{\delta \boldsymbol{\omega}_{i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta \mathbf{A}_j^T}{\delta q_j} \boldsymbol{\omega}_{j-}^{j-}
\end{aligned}$$

$$\begin{aligned}
\frac{\delta \dot{\boldsymbol{\omega}}_{i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta \dot{\boldsymbol{\omega}}_i^i}{\delta q_j} + \widetilde{\frac{\delta \boldsymbol{\omega}_i^{i+}}{\delta q_j}} \mathbf{z}_{i+}^{i+} \dot{q}_{i+} \\
\frac{\delta \dot{\boldsymbol{\omega}}_{i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta \mathbf{A}_j^T}{\delta q_j} \dot{\boldsymbol{\omega}}_{j-}^{j-} + \left(\widetilde{\frac{\delta \mathbf{A}_j^T}{\delta q_j} \dot{\boldsymbol{\omega}}_{j-}^{j-}} \right) \mathbf{z}_j^j \dot{q}_j
\end{aligned}$$

$$\frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} = \widetilde{\frac{\delta \dot{\boldsymbol{\omega}}_{i+}^{i+}}{\delta q_j}} + \widetilde{\frac{\delta \boldsymbol{\omega}_{i+}^{i+}}{\delta q_j}} \tilde{\boldsymbol{\omega}}_{i+}^{i+} + \left(\widetilde{\frac{\delta \boldsymbol{\omega}_{i+}^{i+}}{\delta q_j}} \tilde{\boldsymbol{\omega}}_{i+}^{i+} \right)^T$$

$$\begin{aligned}
\frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \left[\frac{\delta \ddot{\mathbf{s}}_{0,i}^i}{\delta q_j} + \frac{\delta \boldsymbol{\Omega}_i^i}{\delta q_j} \mathbf{s}_{i,i+}^i \right] \\
\frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta \mathbf{A}_j^T}{\delta q_j} \left[\ddot{\mathbf{s}}_{0,j-}^{j-} + \boldsymbol{\Omega}_{j-}^{j-} \mathbf{s}_{j-,j}^{j-} \right]
\end{aligned}$$

$$\frac{\delta \mathbf{M}_{c_{i+}}^{i+}}{\delta q_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} \mathbf{J}_{c_{i+}}^{i+} - \left(\frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} \mathbf{J}_{c_{i+}}^{i+} \right)^T$$

$$\frac{\delta \mathbf{F}_{c_{i+}}^{i+}}{\delta q_j} = m_{i+} \frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta q_j}$$

A.6 Reverse Partials with Initial Conditions

$$\begin{aligned}
\frac{\delta \mathbf{f}_i^i}{\delta q_j} &\underset{\exists i+}{=} \frac{\delta \mathbf{F}_{c_i}^i}{\delta q_j} + \sum_{i+} \left[\frac{\delta \mathbf{A}_{i+}}{\delta q_j} \mathbf{f}_{i+}^{i+} + \mathbf{A}_{i+} \frac{\delta \mathbf{f}_i^i}{\delta q_j} \right] \\
\frac{\delta \mathbf{f}_i^i}{\delta q_j} &\underset{\nexists i+}{=} \frac{\delta \mathbf{F}_{c_i}^i}{\delta q_j}
\end{aligned}$$

$$\begin{aligned}
\frac{\delta \eta_i^i}{\delta q_j} &\underset{\exists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{c_i}^i}{\delta q_j} + \widetilde{\frac{\delta \mathbf{M}_{c_i}^i}{\delta q_j}} + \sum_{i+} \left[\frac{\delta \mathbf{A}_{i+}}{\delta q_j} \eta_{i+}^{i+} + \right. \\
&\qquad \left. \mathbf{A}_{i+} \frac{\delta \eta_{i+}^{i+}}{\delta q_j} + \widetilde{\mathbf{s}_{i,i+}^i} \frac{\delta \mathbf{A}_{i+}}{\delta q_j} \mathbf{f}_{i+}^{i+} + \widetilde{\mathbf{s}_{i,i+}^i} \mathbf{A}_{i+} \frac{\delta \mathbf{f}_{i+}^{i+}}{\delta q_j} \right] \\
\frac{\delta \eta_i^i}{\delta q_j} &\underset{\nexists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{c_i}^i}{\delta q_j} + \widetilde{\frac{\delta \mathbf{M}_{c_i}^i}{\delta q_j}}
\end{aligned}$$

$$\frac{\delta \tau_i}{\delta q_j} = \frac{\delta \eta_i^i}{\delta q_j} \cdot \mathbf{z}_i^i$$

References

[1] BALAFOUTIS, C. A., AND PATEL, R. V. *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach.* Kluwer Academic Publishers, 1991.

[2] BRUDERLIN, A., AND WILLIAMS, L. Motion signal processing. In *Computer Graphics* (Aug. 1995), pp. 97–104. Proceedings of SIGGRAPH 95.

[3] BURDETT, R. G., SKRINAR, G. S., AND SIMON, S. R. Comparison of mechanical work and metabolic energy consumption during normal gait. *Journal of Orhopaedic Research 1*, 1 (1983), 63–72.

[4] COHEN, M. F. Interactive spacetime control for animation. In *Computer Graphics* (July 1992), pp. 293–302. Proceedings of SIGGRAPH 92.

[5] GILL, P. E., MURRAY, W., AND WRIGHT, M. H. *Practical Optimization.* Academic Press, 1981.

[6] HOLLERBACH, J. M. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics SMC-10*, 11 (Nov. 1980).

[7] LIU, Z., AND COHEN, M. F. An efficient symbolic interface to constraint based animation systems. In *Proceedings of the 5th EuroGraphics Workshop on Animation and Simulation* (Sept. 1994).

[8] LIU, Z., GORTLER, S. J., AND COHEN, M. F. Hierarchical spacetime control. In *Computer Graphics* (July 1994), pp. 35–42. Proceedings of SIGGRAPH 94.

[9] UNUMA, M., ANJYO, K., AND TEKEUCHI, R. Fourier principles for emotion-based human figure animation. In *Computer Graphics* (Aug. 1995), pp. 91–96. Proceedings of SIGGRAPH 95.

[10] WITKIN, A., AND KASS, M. Spacetime constraints. In *Computer Graphics* (Aug. 1988), pp. 159–168. Proceedings of SIGGRAPH 88.

[11] WITKIN, A., AND POPOVÍC, Z. Motion warping. In *Computer Graphics* (Aug. 1995), pp. 105–108. Proceedings of SIGGRAPH 95.

[12] ZHAO, J., AND BADLER, N. I. Inverse kinematics positioning using non-linear programming for highly articulated figures. *ACM Transactions on Graphics 13*, 4 (Oct. 1994), 313–336.