Verbs and Adverbs: Multidimensional Motion Interpolation

Charles Rose and Michael F. Cohen
Microsoft Research

Bobby Bodenheimer
Georgia Institute of Technology

We describe methods to leverage existing motion capture or animated sequences that allow real-time interpolation based on the settings of adverbs such as happy, sad, and angry.

Creating believable animated human figures proves difficult, even with the most sophisticated software available. Once an acceptable animation segment has been created, either by an animator or through motion capture, the results remain difficult to reuse. The additional work to modify the animation may take almost as much time as creating the original motion. Furthermore, the exact style or structure needed for a particular motion may not be known until runtime. Interactive applications, such as a 3D video game or a virtual environment require a real-time controllable animation system.

Research into controllable human figure animation divides into three major groupings: procedural, simulated, and interpolated. Procedural animation uses code fragments to derive the degrees of freedom (DOF) values at a particular time. The procedures can be as sophisticated as needed to provide different motion styles or react to different conditions of the simulated environment.

Dynamically simulated figure animation uses controllers together with a simulated human to generate motion. The degree to which this method succeeds relies on how accurately we can understand and model human motion. Unfortunately, both these methods can alienate classically trained animators and use motion-capture technology ineffectively. This proves important since animators and motion-capture systems each produce compelling results. To leverage their qualities, a system must use what these resources provide.

The third major grouping, interpolated animation, uses sets of example motion with an interpolation scheme to construct new motions. The primary problems to solve with this approach are to provide a set of meaningful, high-level control knobs to the animator or runtime system, maintain the aesthetic of the source motions in the interpolated motions, and extrapolate motion. Plus, it is difficult to acquire the examples—each is precious. Additionally, motion interpolation must be efficient for use in a run-time environment rather than earlier in a production pipeline. For these reasons, we chose an interpolation scheme using radial basis functions.

This article describes a system for real-time interpolated animation that addresses some of these problems. Through creating parameterized motions—which we call "verbs" parameterized by "adverbs"—a single authored verb produces a continuous range of subtle variations of a given motion at real-time rates. As a result, simulated figures alter their actions based on their momentary mood or in response to changes in their goals or environmental stimuli. For example, we demonstrate a "walk" verb that can show emotions such as happiness and sadness, and demonstrate subtle variations due to walking up or down hill while turning to the left and right.

We also describe "verb graphs," which act as the glue to assemble verbs and their adverbs into a runtime data structure. Verb graphs provide the means for seamless transition from verb to verb for the simulated figures within an interactive runtime system. Finally, we briefly discuss the discrete event simulator that handles the runtime main loop.

Related work

The idea of altering existing animation to produce different characteristics is not new. Unuma et al.[1] use Fourier techniques to interpolate and extrapolate motion data. Amaya et al.[2] alter existing animation by extracting an "emotional transform" from example motions, which they then apply to other motions. For example, "anger" from an angry walk can be applied to a run to generate an "angry" run. Our work does not follow this approach—it does not apply characteristics of one motion to another, but instead assumes that the initial library of motions contains these emotions. Unlike these two techniques, our method is not based in the fre-

0272-1716/98/$10.00 © 1998 IEEE

Table 1. Terminology for objects.

Object	Variable	Subscript	Subscript Meaning	Subscript Range
Motion example	M	i	Motion example number	1..*NumExamples*
DOF	θ	i	Motion example number	1..*NumExamples*
		j	DOF index	1..*NumDOF*
B-spline	B	k	B-spline index	1..*NumCP*
B-spline control point	b	i,j,k		
Point in adverb space	$\mathbf{p}$	i		
Keytime	K	m	Keytime index	1..*NumKeyTimes*
Radial basis	R	i	Basis associated with M_i	
Radial coefficient	r	i,j,k		
Linear basis	A	l	Adverb index	1..*NumAdverbs*
Linear coefficient	a	j,k,l		
Distance	d	i	Distance to p_i	

quency domain and thus can handle nonperiodic motions that earlier methods failed to capture.

Bruderlin and Williams[3] use multi-target interpolation with dynamic time warping to blend between motions and displacement mappings to alter motions such as grasps. Witkin and Popovic[4] have a similar system for editing motion-capture clips. The former work—done in the same spirit as ours—addresses many of the same difficulties, specifically the need for selecting appropriate key times and the consequent need for time warping. One difference between the two approaches lies in the choice of interpolation techniques: Bruderlin and Williams use multiresolution filtering of joint angles in the frequency domain, whereas our technique decouples the solution representation from the interpolation mechanism. Perlin[5] approaches this problem in a very different way by using noise functions to simulate personality and emotion in existing animation.

Both Wiley and Hahn[6] and Guo and Robergé[7] produce new motions using linear interpolation on a set of example motions. Both techniques require $O(2^d)$ examples, where d is the dimensionality of the control space. Our technique, using radial B-splines, requires $O(n)$ examples to establish the baseline approximation and $O(n^3)$ to compute the resulting answer. To compare, a Delaunay triangulation of the data would require $O(n^{\text{ceil}\,(d/2)})$ to compute, when $d \geq 3$.

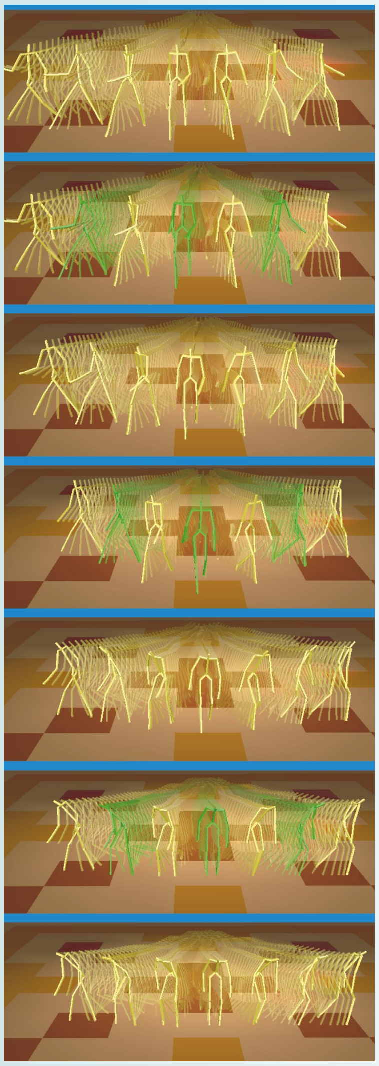
Wiley and Hahn's work closely relates to our own, with the additional difference that their time scaling is uniform, whereas ours is nonuniform and based on key events. While uniform time scaling obviates the need for an animator to select structurally similar poses during motions, it assumes that the motions being interpolated are very similar in time. When you violate this assumption, oddities in the motion can result. Wiley and Hahn also reparameterize and sample their motions on a multidimensional grid. Then they perform simple interpolations at runtime. This requires computation and storage exponential in the number of parameters.

Additionally, neither Wiley and Hahn nor Guo and Robergé discuss blending subsets of examples, which would arise when a designer places new examples "between" old examples when refining the interpolated motion space. Our technique, on the other hand, becomes refined with more examples as required. Because we use an approximation method based on radial B-splines with compact support to perform the interpolation, our examples have limited effect over the space of animations, thus ensuring the use of subsets of

To Find Out More About This Research

The Human Figure Animation Project is working to make better and more realistic animations of humans for computer graphics. Much of our current work involves motion capture reuse. Past work includes torque-minimal transitioning between motion captured segments and robust motion capture analysis.

More information about this work, video segments in particular, can be found at the Human Figure Animation Project's Web site at http://www.research.microsoft.com/research/graphics/hfap. See Figure A for an example of verbs and adverbs in use.



A The character depicted walking here expresses different emotional states through its demeanor while turning in various directions. You can see increasing happiness in the walk from bottom to top.

Table 2. Terminology for time.

Time	Variable	Range
Clock	τ	
Keytime	T	$0..K_{NumKeyTimes}$
Generic	t	$0..1$

the examples at runtime as appropriate.

Hodgins and Pollard[8] interpolate over the space of control laws as opposed to joint trajectories. The control functions define the gains in the joints within a physical dynamics framework. By interpolating the control for an action such as running, they can alter a figure's physical characteristics from a child to an adult or from a male to a female character.

An important distinction in our work is that we perform the interpolation simultaneously in real time over multiple dimensions, such as emotional content and physical characteristics. Although we apply the techniques to interpolating trajectories characterized by coefficients of spline curves, the methods presented here also apply to coefficients in the Fourier and other domains. It may also be possible to apply similar ideas to control the parameters of a physically based model.

In the context of autonomous agents, our work presents a back end for applications such as games, the Improv[9] system, and the work proposed by Blumberg and Galyean.[10] The high-level control structures found in such applications can select verbs and adverbs, while the work we present here provides the low-level animation itself. Thus, we create the motion in real time for "directable" creatures as discussed by Blumberg and Galyean.

Verbs and adverbs

The figure animation system presented here consists of two main parts. An offline authoring system provides the tools for a designer to construct controllable motions—verbs—from sets of examples. In addition, the authoring system provides tools for constructing transitions between verbs. It also puts the verbs and transitions together in a verb graph. This structure becomes the controlled object for the runtime portion of the verbs and adverbs system. The runtime system controls invoking the verbs and evaluating the figure's pose at each frame of the animation. We begin with a discussion of the authoring system and then move on to the runtime system. Refer to Tables 1 and 2 for the symbols used in the text.

We assume that the simulated figures, or "creatures," discussed here are represented well as a hierarchy of rigid links connected by joints. Each joint may contain one or more rotational DOF. The root of the hierarchy has six special DOF representing the figure's position and orientation in the global coordinate frame. In particular, we use a 46 DOF human figure. Each DOF's motion is represented as a function through time $\theta_j(T)$, $j = 1\ldots NumDOF$, where T represents clock time from [0..duration of the motion], which will shortly be defined as keytime time. Given these motions of the DOF, you can render the creature at any point in time.

Traditionally, hand-crafted or motion-captured animation segments have DOF functions of one variable—time. The DOF functions for a verb are never explicitly represented, but rather evolve at runtime through interpolating example motions weighted by changing interpolation parameters or adverbs. These adverbs may represent emotional axes such as happy-sad, knowledgeable-clueless, and so on. They may also represent physical parameters such as whether, in the case of a walk verb, the figure walks up or down hill and whether the motion curves to the left or right. The set of adverb axes define a multidimensional adverb space of all possible variations for a particular verb.

We construct verbs from sets of similar but distinct example motions. These examples can be obtained by keyframing or with a motion-capture system. In either case, certain restrictions on the set of examples apply. The primary restriction is that all examples for a verb must be structurally similar. A set of example walks, for instance, must all start out on the same foot, take the same number of steps, have the same arm swing phase, and have no spurious motions such as a head scratch. The other primary restriction is consistent use of joint angles. Anatomically implausible settings of the DOF values can yield the same overall effect as a plausible setting due to the redundant nature of two- and three-DOF joints. Bodenheimer et al.[11] present methods to ensure consistent DOF values for motion captured data.

We annotate each example motion by hand with a set of adverb values, placing the example somewhere in the adverb space. Additionally, a set of "keytimes"—instances when important structural elements such as a foot-down occur—must be specified. We use these keytimes to preprocess the examples into a canonical timeline for interpolation and for later synthesis of the phrasing for the interpolated motions. Finally, we annotate examples with a set of intermittent constraints (for example, the foot should not slip while in contact with the floor).

A verb M is therefore defined by a set of example motions M_i,

$$\begin{aligned} M_i = \{&\theta_{ij}(T), \mathbf{p}_i, K_m : i = 1\ldots NumExamples, \\ &j = 1\ldots NumDOF, m = 0\ldots NumKeyTimes\} \end{aligned}$$

where the $\theta_{ij}(T)$ are the DOF functions for the ith example M_i, $\mathbf{p}_i$ the location of the example in the adverb space, and K the set of keytimes that describe the example's phrasing (relative timing of the structural elements). The time parameter, T, represents clocktime with the clock starting at 0 at the beginning of the motion example.

Example motions

Each example motion M_i is defined by a number of DOF functions, denoted by $\theta_{ij}(T)$. Each θ_{ij} (that is, the jth DOF for the ith example) is represented as a uniform cubic B-spline curve specified by $NumCP$ control points:

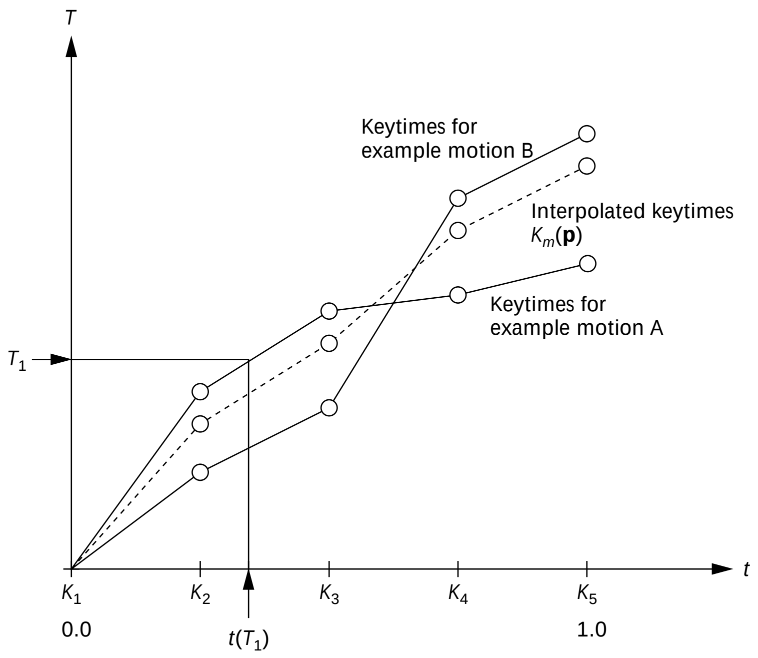
$$\theta_{ij}(T) = \sum_{k=1}^{NumCP} b_{ijk} B_k(T)$$

where the $B_k(T)$ are the B-splines and the b_{ijk} are the scalar B-spline coefficients or control points for the ith example M_i. Bartels et al.[12] offer an extensive discussion of B-splines and their properties.

The interpolation technique presented here decouples the interpolation scheme from the representation of the examples. The examples, therefore, could be encoded using other methods, such as a wavelet or Fourier decomposition.

Time warping

The keytimes define a piecewise linear mapping from $T \in \{0 \ldots K_{NumKeyTimes}\}$ to a generic time $t \in \{0 \ldots 1\}$. The first keytime of the verb, K_1, is defined as 0. The last keytime, $K_{NumKeyTimes}$, marks the verb's duration. In the case of cyclic verbs, such as walking, it must also mark the same event as the first. More generally, given a T between 0 and $K_{NumKeyTimes}$,



1 Mapping between generic time and keytimes.

$$t(T)=\left((m-1)+\frac{T-K_m}{K_{m+1}-K_m}\right)\frac{1}{NumKeyTimes-1} \quad \textbf{(1)}$$

for the largest m such that $T > K_m$ and keeping in mind that $t(0) = 0$. Figure 1 shows this mapping. In other words, at each keytime, the generic time

$$t(K_m)=\frac{m-1}{NumKevTimes-1}$$

At the third of four keytimes, $t = 2/3$ as the keytimes will be at 0, 1/3, 2/3, and 1. Between keytimes, t is linearly interpolated.

Once all the examples have had their time reparameterized to generic time t, they are in what is called canonical form. For a given t, all examples will lie at the same structural point of the motion, regardless of phrasing or overall duration. The system interpolates the examples within this generic timeframe. The keytime equations themselves are also interpolated between examples. The system then applies the inverse of this interpolated timing function to "untimewarp" the interpolated motion to recapture phrasing and duration.

Inverse kinematic constraints

In addition to time warping the motions, keytimes also specify periods during which kinematic constraints must be enforced. For example, a walk's keytimes might be heel-strike and toe-off. Between these keytimes, the foot must remain stationary.

The values for specific constraint conditions such as the location the end effector (for example, the foot) should maintain, are set not when they're designed, but at playback when a keytime that triggers a constraint is crossed. If, for example, a keytime is crossed triggering a foot constraint when the foot's horizontal location is (x, z) and the floor's height at that point is $y = floor(x, z)$, then the constraint location is set to (x, y, z). The system maintains this constraint position until another keytime releasing the constraint is crossed.

A fast inverse kinematics optimization enforces constraints at runtime. At a given generic time t, a creature is first positioned independent of any constraints by evaluating the active verb(s) at that time. In general, this will place the end effector close to the constraint point. We need only consider modifying the DOF between the root and the end effector when making the subtle changes needed to exactly satisfy the constraint. You can find the changes in these DOF needed to satisfy the constraint at each frame at runtime by solving the linear system (typically of size 5 or 6) of the form

$$\mathbf{J}\,\Delta\theta = \Delta\mathbf{x} \quad \textbf{(2)}$$

where $\mathbf{J}$ is the Jacobian of the DOF with respect to the end effector's motion, $\Delta\mathbf{x}$ is the vector from where the verb placed the end effector to the constraint point, and $\Delta\theta$ is the amount the DOF must be perturbed to hold the end effector in place. See Girard[13] or Watt[14] for details on such inverse kinematic problems.

Verb construction

Once the example motions have been specified for a verb with their associated keytimes and adverb settings, the system then constructs a continuous "space" of motions parameterized by the adverbs. The dimension of this space equals the number of adverbs, *NumAdverbs*. The specific values of the individual adverbs define a point in this space. The point may move from moment to moment during the interactive animation if, for example, the character's mood changes or the character begins to walk up or down hill. The goal is to produce at any point $\mathbf{p}$ in the adverb space a new motion $M(\mathbf{p}, t)$ derived by interpolating the example motions. When $\mathbf{p}$ equals the adverb settings for a particular example motion i, then $M(\mathbf{p}, t)$ should equal $M_i(t)$.

Each example motion has one free variable for each DOF's B-spline control point and one free variable for each keytime. The time warping described above ensures that corresponding control points in each example motion specify similar moments in each motion, even if the overall lengths of the example motions differ. This lets us treat the example motion interpolation as a separate problem for each control point and each keytime (that is, $NumCP \times NumDOF + NumKeyTimes$ individual interpolation problems).

Here is the standard problem of multivariable interpolation: given N distinct points $\mathbf{p}_i$ in $\mathbf{R}^n$ and N values v_i in $\mathbf{R}$, find a function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ such that for all i, $f(\mathbf{p}_i) = v_i$ and such that f does not oscillate badly between values. The high dimensionality of the space defined by the adverbs, coupled with the desire to require few example motions (perhaps only two to three times the number of adverbs), presents difficulties for many interpolation methods. Given these difficulties, we selected a combination of radial basis functions and low-order (linear) polynomials for this problem. The polynomial function provides an overall approximation to the space defined by the example motions. It also allows for extrapolation outside the convex hull of the locations of the example motions. The radial bases then locally adjust the polynomial to interpolate the example motions themselves.

Radial basis functions have the form

$$R_i\,(d_i(\mathbf{p}))$$

where R_i is the radial basis associated with M_i and $d_i(\mathbf{p})$ is a measure of the distance between $\mathbf{p}$ and $\mathbf{p}_i$, most often the Euclidean norm $\|\mathbf{p}-\mathbf{p}_i\|$. Because sums of radial bases cannot represent an affine or polynomial function, radial basis sets are often augmented by adding a polynomial of fixed degree.

Details of the mathematics for this type of interpolation can be found in the seminal work of Micchelli[15] and in the survey article by Powell.[16] Radial basis functions have been used in computer graphics for image warping by Ruprecht and Müller[17] and Arad et al.[18]

The value of each interpolated DOF curve control point in this space, $b_{jk}(\mathbf{p})$, is defined as

$$b_{jk}(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{ijk} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{jkl} A_l(\mathbf{p}) \qquad \mathbf{(3)}$$

where the r_{ijk} and R_i are the radial basis function weights and radial basis functions themselves, and the a_{jkl} and A_l the linear coefficients and linear bases as explained below. Interpolated keytimes are similarly defined as

$$K_m(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{im} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{lm} A_l(\mathbf{p}) \qquad \mathbf{(4)}$$

For each verb there are $(NumCP \times NumDOF)$ control point interpolations (Equation 3) and $NumKeyTimes$ keytime interpolations (Equation 4).

This leaves us with the problem of choosing the specific shape of the radial bases and determining the linear and radial coefficients. We solve these problems in two steps, by solving first for the linear coefficients and then for the radial basis coefficients.

Linear approximation

In the first step, we solve for the linear coefficients by finding the hyperplane through the adverb space that best fits the variation across the example motions of the selected control point or keytime. The linear basis functions are simply $A_l(\mathbf{p}) = \mathbf{p}_l$, the lth component of $\mathbf{p}$, and $A_0(\mathbf{p}) = 1$. An ordinary least squares solution determines the $NumAdverbs + 1$ linear coefficients, a_{jkl}, that minimize the sum of squared errors between

$$\tilde{b}_{ijk}(\mathbf{p}_i) = \sum_{l=0}^{NumAdverbs} a_{jkl} A_l(\mathbf{p}_i)$$

and b_{ijk}, the actual B-spline control point (or keytime) being interpolated, where $\mathbf{p}_i$ is the adverb setting for the ith example motion. Letting $\mathbf{b}_{jk}$ and $\tilde{\mathbf{b}}_{jk}$ denote vectors of each $b_{ijk}(\mathbf{p}_j)$ and $\tilde{b}_{ijk}(\mathbf{p}_j)$ for a fixed j and k, the linear approximation leaves the residuals

$$\bar{\mathbf{b}}_{jk} = \mathbf{b}_{jk} - \tilde{\mathbf{b}}_{jk}$$

The radial basis interpolates these residuals.

Radial basis functions

We define one radial basis function for each example motion. The radial basis functions are solely a function of the distance, $d_i(\mathbf{p}) = \|\mathbf{p}-\mathbf{p}_i\|$ between a point in the adverb space, $\mathbf{p}$, and the point in the adverb space corresponding to example motion i, $\mathbf{p}_i$. The radial basis itself, $R_i(\mathbf{p})$, has its maximum at $\mathbf{p}_i$ (that is, where $d = 0$). We would also like the radial bases to have compact support (that is, have value zero beyond some distance) to limit each example motion's influence to a local region of adverb space.

Several choices exist for the specific shape of the radial basis. For its combination of simplicity and C^2 continuity, we chose a radial basis with a cross section of a dilated cubic B-spline, $B(d/\alpha)$. We chose the dilation factor, $1/\alpha$, for each example motion to create a support radius for the B-spline equal to twice the Euclidean distance to the nearest other example motion. For $\alpha = 1$, the cubic B-spline has a radius of 2.0, thus α is simply the minimum separation to the nearest other example in the adverb space. Given this definition, it is clear that the example motions must be well separated.

The coefficients, r_{ijk}, can now be found for each DOF B-spline control point and keytime by solving the linear system,

$$\mathbf{D}\mathbf{r}_{jk} = \bar{\mathbf{b}}_{jk}$$

where $\mathbf{r}_{jk}$ is a vector of the r_{ijk} terms for a fixed j and k, and **D** a square matrix with terms $D_{i1,\, i2}$ equal to the value of the radial basis function centered on motion i_1 at the location of motion i_2. Thus

$$\mathbf{D}_{i_1,i_2} = R_{i_1}(\mathbf{P}_{i_2}) = B\left(\frac{d_{i_1}(\mathbf{p}_{i_2})}{\alpha_{i_1}}\right)$$

Verb graphs

In addition to constructing individual verbs, it is important to combine them so that smooth transitions occur between verbs. To do so, an author builds a directed graph of verbs, or "verb graph," in which the nodes represent the verbs and the arcs represent transitions between verbs. If multiple transition arcs leave a verb, each transition may be given a "likelihood," used to stochastically choose at runtime between multiple possible transitions if none have been explicitly specified.

The adverbs are shared across verbs although each verb may or may not respond to each adverb. For example, an adverb for up hill or down hill may have no meaning to a "sit" verb. The verb graph remains static, that is, fixed at authoring time.

Transitions

Given corresponding time intervals (as set by the designer) in two verbs, transitions move control smoothly from one verb to the next. For example, in transitioning from walking to running, the time intervals may both be set to span the right foot placement.

A transition maps similar segments between two verbs *A* and *B*. The correspondence region is determined by the four variables, t_s^A, t_e^A, t_s^B, and t_e^B the start and stop times of the region in each of the two verbs. Note that we express these times as generic times since the verbs *A* and *B* may have different durations based on their current adverb settings. In other words, the transitions are designed based on the similarity of the motions' structure (generic time) and so will work even though the verbs themselves change at runtime.

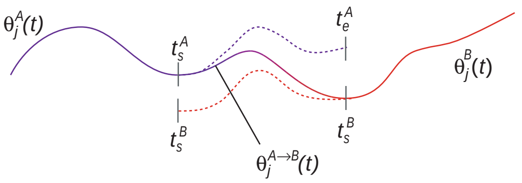
We calculate the transition's duration by taking the average of the two blending regions' lengths:

$$\frac{(T(t_e^A)-T(t_s^A))+(T(t_e^B)-T(t_s^B))}{2}$$

where the *T* function maps generic time to real time (keytime time) for the verb's current adverb settings. Transitions, since they are made up of verbs, are affected by the verb's adverb settings and therefore take on the mood, duration, and phrasing of their constituent verbs.

Rose et al.[19] discuss space-time transitioning mechanisms. Generating space-time transitions between two motions remains an offline process. Since the specific motions the verbs generate are not known until runtime, we cannot use this method here. Thus, in this work we rely on a simpler formulation to calculate the transitions between verbs at runtime.

Verbs *A* and *B* blend together by fading one out while fading the other in. A monotonically decreasing blending function with a range and domain of [0,1] determines the relative contribution of the two verbs. We use a sigmoid-like function, $\alpha = 0.5 \cos(\beta\pi) + 0.5$, in this work.



2 Transition blending during correspondence region.

Over the transition duration, β moves linearly from 0 to 1, representing the fraction of the way through the transition intervals in each verb. We determine the transitional motion by linearly interpolating the verbs *A* and *B* with weights α and $1 - \alpha$, respectively. You can find the internal DOF by interpolating the joint positions as in Figure 2, which shows the DOF function θ_j as a combination of the two DOF functions θ_j^A and θ_j^B during the transition region. The path the joint takes is defined by θ_j^A before the transition, $\theta_j^{A\to B}$ during the transition, and θ_j^B after the transition. To achieve smooth motion for the root DOF, we interpolate the velocities (rather than positions) of the two verbs and then integrate the results through the transition.

Verb graph at runtime

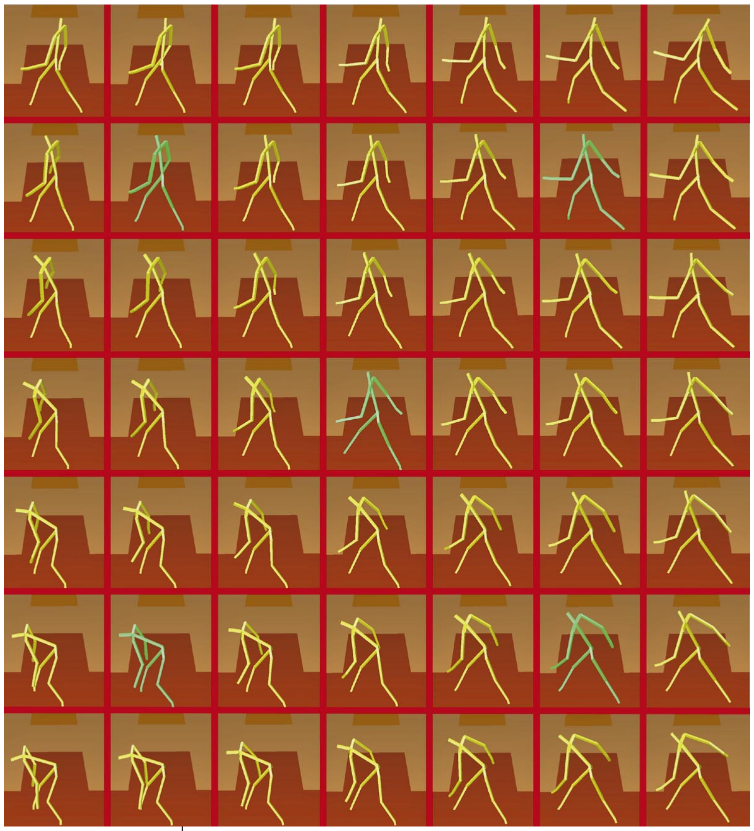
Issuing commands to the creature controls the movement from verb to verb in the verb graph. When directed to perform a particular verb, the system executes a search to determine a sequence of transitions and verbs that will reach the desired action from the one currently playing. The system chooses the shortest path, where shortness represents the number of needed transitions. We represent this path through the verb graph as a queue consisting of transition, verb, transition, verb, and so on.

To keep the verb graph in a useful state requires some bookkeeping. The root of the creature's motion must be stored and maintained at each time step to reorient and reposition the base verbs for concatenation onto the current action. For example, in a walk transitioning to itself, the horizontal position specified by the verb indicates an offset from the creature's position when the verb was invoked. Thus, the creature continues to walk forward and does not jump back to the origin on each stride. Also, the set of current active constraints must be updated by discarding expired constraints and creating new ones as the animation progresses.

The verb graph's execution queue cannot be empty or the creature will come to a stop. For the verb graph to continue, it must transition from its current action before that action has completed. When only the currently active verb remains in the queue, the system automatically selects a transition from the set of available transitions away from that verb. The selection is made stochastically according to weights the designer specified during the verb graph construction. In cyclic verbs such as walking, the default (with a probability of one) transition is usually the one leading back into itself.

Runtime verb evaluation

Given the authored verbs parameterized by adverbs and time warped by keytimes, and a verb graph to orga-



3 A walk sampled across two emotional axes. The green figures are the example motions. The rest were created through the verb-adverb mechanism. Happiness increases from bottom to top and knowledge from left to

nize the verbs and transitions, we are now ready to see how these structures operate at runtime. The goal is to continuously vary the adverb settings and have these changes reflected in the simulated creatures' subtle motion. In other words, the user or application defines the path that the point **p** will take on the fly, and the runtime system must respond accordingly. Given the time and setting of **p**, the evaluation of the creature's pose must be fast enough to allow interactive frame rates.

A discrete event simulator serves as the runtime system's main loop. This system tracks the clock and sequentially processes events placed on its event queue. Each event has a time stamp and an associated callback function. The system inserts events in the event queue (in time stamp order) and processes them by invoking the callback function with the time stamp as a parameter. Events may be one of three types: normal, sync, or optional. The system processes normal events as they are reached in the queue, independent of the relative values of the time stamp and clock. Sync events wait for the clock to catch up to the time stamp if the clock time is less than the time stamp; they execute immediately otherwise. The system skips optional events if the clock has passed the time stamp; otherwise, optional events act like normal events.

The most common events are "render" and "display" events. A render event evaluates the DOF at the time indicated by the time stamp to set the creature's pose, then renders (but does not display) an image. The render event has the "normal" flag and thus creates an image as soon as the event reaches the queue. A display event with the same time stamp but with a sync flag waits for the clock to reach the time stamp and displays the rendered image. The render event also measures the amount of clock time between frames and estimates the best time stamp for the next render or display events and inserts them in the event queue. This way, the frame rate dynamically adjusts to the computational load.

The system schedules other events to induce transitions between verbs or to turn on or off constraints.

Evaluating DOF

The render event callback requests that all DOF be evaluated at a given time, τ. The currently active verb (or verbs when a transition occurs) is evaluated at time $\tau - \tau_{offset}$ with the current adverb settings, **p**. τ_{offset} is the clock time when the current verb or transition comes to the fore of the queue. The following pseudocode summarizes this process:

```
 1 T = τ − τ_offset
 2
 3 For each keytime m
 4   K_m = InterKey(m, p)  // Eqn.  4
 5 Next
 6
 7 t = GenericTime(T, K)  // Eqn.  1
 8
 9 For each DOF j
10   For each B-spline coefficient k
11      b_jk = InterBSCoeff(j, k, p)
        // Eqn. 3
12   Next
13   q_i   =   Â_k   b_jk   B_k(t)
14 Next
15 For each kinematic constraint c
16   EnforceConstraint(c)  //  Eqn. 2
17 Next
```

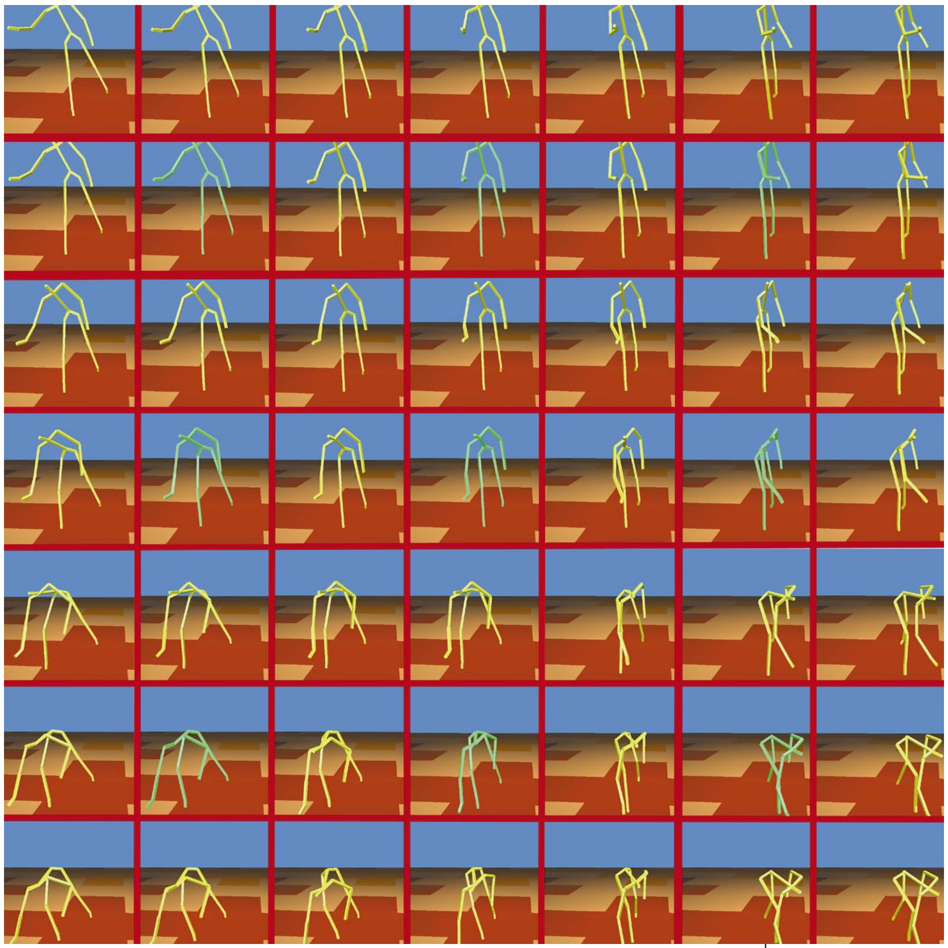
Note that only lines 1, 7, 13, and 16 must be evaluated at each frame time. The system caches the values computed at the other steps. The interpolations, lines 4 and 11, only change when the parameters change or the system invokes a new verb. In addition, in line 11 only four of the B-spline coefficients are required for a given *t*. As *t* progresses past a B-spline knot value, one coefficient drops off the beginning of the set of four and the system adds a new one as the knot value passes. Thus, on average, less than one of the interpolated coefficients per DOF need be evaluated per frame if **p** does not change. If **p** changes from frame to frame, four coefficients per DOF must be calculated as must the *m* interpolated keytimes. The entire DOF evaluation process takes only a small fraction of the time to compute compared to the polygon rendering for the frame given the pose.

Results

Our library of motion capture contains a repertoire of example motions for a variety of parameterized verbs—walking, jogging, reaching, and idling. Some verbs, such as walk, have a large number of examples representing different emotions such as happy, sad, angry, afraid, clueless, tired, delirious, determined, frenzied, ashamed, bored, goofy, and grief stricken. We also have walks at different inclinations and radii of turn.

The data for the example motions we discuss here were motion captured with an Ascension Motion Star system sampled at 120 Hz with 15 six-DOF sensors positioned on the body. The raw data must be preprocessed to fit a rigid-body model with hierarchical joint rotations and fewer DOF corresponding to the limitations of human joints. The methods described by Bodenheimer et al.[11] ensure that the motion-capture data use joint angles consistently for redundant DOF. The final model has 40 joint DOF in addition to six DOF of the root, located between the hips, for global positioning and orientation.

We constructed a parameterized walk along two emotional axes—happy-sad and knowledgeable-clueless—as well as physical characteristics such as up hill or down hill and turning. Figure 3 shows a sampling of this walk across the two emotional axes. A reaching verb was parameterized by the three positional values representing the goal of the reach (see Figure 4). We constructed various emotional idle motions, plus a jog with a turning adverb.


4 A reach sampled across two axes of the goal position for the hand. The green figures are the example motions. The rest were created through the verb-adverb mechanism. Reach height ranges along the vertical axis and reach direction along the horizontal axis.

Parameterizing a space based on turning and on changing elevation gives us great control over the creature. We created the jogging and walking turn adverbs from two motion-captured example motions each, a forward motion and a motion to the right. To create a third example motion to the left, we mirrored the motion to the right. The interpolation gives convincing control of the radii of the turn's curvature and allows convincing navigation.

Solving for the interpolation coefficients took about two minutes on a 200-Mhz PentiumPro processor for the most complex verbs. Recall that this offline computation need only be carried out once per verb. At runtime, you can evaluate a character's position as described in the pseudocode above at approximately 200 Hz, or about 5 milliseconds, thus barely affecting the overall frame rate. This timing was taken with a constantly changing **p**, thus requiring interpolating the full four coefficients per DOF per frame plus the *m* keytimes.

We can move through the adverb space and verb graph to exhibit several degrees of emotions and subtleties of movement. We have shown the construction of a large verb graph consisting of these parameterized verbs in addition to various unparameterized motions such as stretching, standing, and looking around, with associated transitions between them. The transitions are generated in real time and obey inverse kinematic constraints.

Conclusions

Currently our system only accepts variations in expressions and changes of behavior through user input, but future work will explore putting much of this input under the control of state machines executing under our discrete-event simulator. We plan to eventually create complex autonomous agents in virtual environments.

We expect to continue to enhance the authoring system for verb construction to allow fast modification of the interpolation space the verb defines. The verb-adverb construction described here should thus provide a means to leverage what is perhaps the most valuable aspect of any animation system—the talent and inspiration of the animator constructing the example motions and the capture of real motion from expensive motion-capture systems. In this way, the artist's skills can be leveraged within a real-time setting. ■

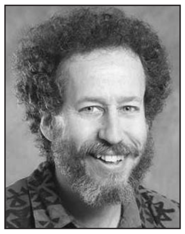
Acknowledgments

We would like to thank Brian Guenter for his input on all aspects of this work. We would also like to thank the Interactive Media Productions group at Microsoft for contributing their time, equipment, and motion-capture expertise. Finally, we would like to thank the reviewers and Jessica Hodgins, whose suggestions helped strengthen this article.

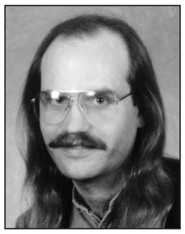
References

1. M. Unuma, K. Anjyo, and R. Tekeuchi, "Fourier Principles for Emotion-Based Human Figure Animation," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, Aug. 1995, pp. 91-96.
2. K. Amaya, A. Bruderlin, and T. Calvert, "Emotion from Motion," *Graphics Interface 96,* W.A. Davis and R. Bartels, eds., Morgan Kaufmann, San Francisco, May 1996, pp. 222-229.
3. A. Bruderlin and L. Williams, Motion Signal Processing," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, Aug. 1995, pp. 97-104.
4. A. Witkin and Z. Popovc, "Motion Warping," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, Aug. 1995, pp. 105-108.
5. K. Perlin, "Real-Time Responsive Animation with Personality," *IEEE Trans. on Visualization and Computer Graphics,* Vol. 1, No.1, Mar. 1995, pp. 5-15.
6. D.J. Wiley and J.K. Hahn, "Interpolation Synthesis for Articulated Figure Motion," *Proc. Virtual Reality Annual Int'l Symp.,* IEEE CS Press, Mar. 1997, pp. 157-160.
7. S. Guo, and J. Robergé, "A High-Level Control Mechanism for Human Locomotion Based on Parametric Frame Space Interpolation," *Proc. Computer Animation and Simulation 96*, Eurographics Animation Workshop, R. Boulic and G. Hégron eds., Springer, New York, Aug. 1996, pp. 95-107.
8. J.K. Hodgins and N.S. Pollard, "Adapting Simulated Behaviors for New Characters," *Computer Graphics* (Proc. Siggraph 97), ACM Press, New York, Aug. 1997, pp. 153-162.
9. K. Perlin, and A. Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds," *Computer Graphics* (Proc. Siggraph 96), ACM Press, New York, Aug. 1996, pp. 205-216.
10. B.M. Blumberg and T.A. Galyean, "Multilevel Direction of Autonomous Creatures for Real-Time Virtual Environments," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, Aug. 1995, pp. 47-54.
11. B. Bodenheimer et al., "The Process of Motion Capture: Dealing with the Data," *Computer Animation and Simulation 97,* Eurographics Animation Workshop, D. Thalmann and M. van de Panne, eds., Springer, New York, Sept. 1997, pp. 3-18.
12. R.H. Bartels, J.C. Beatty, and B.A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, San Fransicso, 1987.
13. M. Girard and A.A. Maciejewski, "Computational Modeling for the Computer Animation of Legged Figures," *Computer Graphics* (Proc. Siggraph 85), ACM Press, New York, July 1985, pp. 263-270.
14. A Watt and M. Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, ACM Press, New York, 1992.
15. C.A. Micchelli, "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions," *Constructive Approximation 2,* 1986.
16. M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review," *Algorithms for Approximation*, J.C. Mason and M.G. Cox, eds., Oxford University Press, Oxford, UK, 1987, pp. 143-167.
17. R. Ruprecht and H. Müller, "Image Warping with Scattered Data Interpolation," *IEEE Computer Graphics and Applications,* Vol. 15, No. 2, Mar. 1995, pp. 37-43.
18. N. Arad et al., "Image Warping by Radial Basis Functions: Application to Facial Expressions," *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing,* Vol. 56, No. 2, Mar. 1994, pp. 161-172.
19. C.F. Rose et al., "Efficient Generation of Motion Transitions using Space-Time Constraints," *Computer Graphics* (Proc. Siggraph 96), Aug. 1996, pp. 147-154.

__Charles Rose__ is a research software design engineer in the graphics group of Microsoft Research. His work has primarily concerned with deriving controllable animation from example motion. He is also interested in autonomous figures and artist-directed skinning techniques. He received a BS in computer science at Trenton State College in Ewing, New Jersey in 1992 and an MA in computer science from Princeton University in 1995. He is currently finishing his doctoral work at Princeton. He is a member of the Computer Society, ACM, and Siggraph.



__Michael F. Cohen__ is a senior researcher and manager of the graphics group of Microsoft Research. He received his PhD at the University of Utah, an MS from Cornell, a BS from Rutgers, and a BA from Beloit in Wisconsin. His interests include realistic image synthesis, image-based rendering, automatic camera control, and linked figure animation. He received the 1998 Siggraph Achievement Award.



__Bobby Bodenheimer__ is currently a postdoctoral fellow in the College of Computing at Georgia Institute of Technology in Atlanta. He received his BS in electrical engineering and mathematics from the University of Tennessee and his PhD in electrical engineering from the California Institute of Technology. He spent two years as a postdoctoral fellow in the graphics group at Microsoft Research. His interests include generating realistic animations using dynamic simulation and motion-capture techniques. He is a member of the IEEE and ACM.

Readers may contact Rose at Microsoft Research, One Microsoft Way, Redmond, WA 98052, e-mail chuckr@microsoft.com.