# Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions

Charles Rose
chuckr@microsoft.com

Bobby Bodenheimer
bobbyb@cc.gatech.edu

Michael F. Cohen
mcohen@microsoft.com

Microsoft Research

**Abstract**

This paper describes methods and data structures used to leverage motion sequences of complex linked figures. We present a technique for interpolating between example motions derived from live motion capture or produced through traditional animation tools. These motions can be characterized by emotional expressiveness or control behaviors such as turning or going uphill or downhill. We call such parameterized motions "verbs" and the parameters that control them "adverbs." Verbs can be combined with other verbs to form a "verb graph," with smooth transitions between them, allowing an animated figure to exhibit a substantial repertoire of expressive behaviors.

A combination of radial basis functions and low order polynomials is used to create the interpolation space between example motions. Inverse kinematic constraints are used to augment the interpolations in order to avoid, for example, the feet slipping on the floor during a support phase of a walk cycle.

Once the verbs and verb graph have been constructed, adverbs can be modified in real-time providing interactive or programmatic control over the characters' actions. This allows the creation of autonomous characters in a virtual environment that exhibit complex and subtle behavior.

## 1 Introduction

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

Research into controllable human figure animation can be divided into three major groupings: procedural, simulated, and interpolated. Procedural animation uses code fragments to derive the degree of freedom (DOF) values at a particular time. The procedures can be as sophisticated as needed in order to provide different motion styles or to 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 is bounded by how accurately human motion is understood and modeled. Both these methods have the disadvantage that they can alienate classically trained animators and that they do not easily make use

of motion capture technology. This is important since animators and motion capture systems each produce compelling results. To leverage their qualities, a system must use what these resources provide.

Interpolated animation is the third major grouping. This method uses sets of example motion together with an interpolation scheme to construct new motions. The primary problems of this approach are in providing a set of meaningful, high level control knobs to the animator or runtime system, maintaining the aesthetic of the source motions in the interpolated motions, and motion extrapolation. Another consideration is the difficulty of acquiring the examples. Each is precious. Additionally, for an interpolated motion scheme to be used in a run-time environment rather than earlier in a production pipeline, it must be efficient. For these reasons, an approximation scheme using radial basis functions was chosen.

This paper describes a system for real-time animation by example that addresses some of these problems. Through the creation of 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 is able to show emotions such as happiness and sadness, and demonstrate subtle variations due to walking uphill or downhill while turning to the left and right.

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

# 2 Related Work

The idea of altering existing animation to produce different characteristics is not new. Unuma *et al.* [16] use Fourier techniques to interpolate and extrapolate motion data. Amaya *et al.* [1] alter existing animation by extracting an "emotional transform" from example motions which is then applied to other motions. For example, "anger" from an angry walk is applied to a run to generate an "angry" run. Our work does not follow this approach in that 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 frequency domain and thus can handle non-periodic motions which earlier methods fail to capture.

Bruderlin and Williams [6] use multitarget interpolation with dynamic timewarping to blend between motions, and displacement mappings to alter motions such as grasps. Witkin and Popović [19] present a similar system for editing motion capture clips. The former work is in the same spirit as ours, and addresses many of the same difficulties, specifically the necessity of selecting appropriate key times for interpolation 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 solution representation from interpolation mechanism. Perlin [11] has approached this problem in a very different way by using noise functions to simulate personality and emotion in existing animation.

Both Wiley and Hahn [18] and Guo and Robergé [8] 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 $d \geq 3$.

The work of Wiley and Hahn is closely related to our own, the additional difference being that their time scaling is uniform, whereas ours is non-uniform and based on key events. While uniform time-scaling obviates the need for an animator to select structurally similar poses during motions, it assumed that the repertoire of motions being interpolated between must be very similar in time. When this assumption is violated, oddities in the motion can result. Wiley and Hahn also reparameterize and sample their motions on a multidimensional grid and then 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 the blending of subsets of examples, which would arise when new examples are placed "between" old examples as a designer refines the interpolated motion space. Our technique, on the other hand, is refined by more examples as required. As we use an approximation method based upon radial B-splines with compact support to perform the interpolation, our examples have limited effect over the space of animations, thus ensuring that subsets of the examples are used at runtime as appropriate.

Hodgins and Pollard [9] 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 the physical characteristics of the figure from a child to an adult or from a male to a female character.

An important distinction in our work is that the interpolation is performed 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 are also applicable 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 [12] system, and the work proposed by Blumberg and Galyean [4]. The high level control structures found in such applications are capable of selecting 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.

# 3 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 and for putting 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 the invocation of verbs and the evaluation of 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 Table 1 for the symbols used in the text.

The simulated figures, or "creatures," discussed here are assumed to be well represented as a hierarchy of rigid links connected by joints. Each joint may contain one or more rotational degrees of freedom (DOF). The root of the hierarchy has six special DOFs which represent the position and orientation of the figure 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 DOFs, one can render the creature at any point in time.

Traditionally hand-crafted or motion captured animation segments have DOF functions of 1 variable,

| Object | Variable | Subscript | subscript meaning | subscript range |
|---|---|---|---|---|
| Motion example | $M$ | $i$ | Motion example number | $1..NumExamples$ |
| DOF | $\theta$ | $i$ | Motion example number | $1..NumExamples$ |
| | | $j$ | DOF index | $1..NumDOF$ |
| B-spline | $B$ | $k$ | B-spline index | $1..NumCP$ |
| B-S 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 $p_i$ | |
| Time | Variable | range | | |
| Clock time | $\tau$ | | | |
| Keytime time | $T$ | $0..K_{NumKeyTimes}$ | | |
| Generic time | $t$ | $0..1$ | | |

Table 1: Terminology

time. The DOF functions for a verb are never explicitly represented, but rather evolve at runtime through the interpolation of example motions weighted by changing interpolation parameters or adverbs. These adverbs may represent emotional axes such as happy-sad, knowledgeable-clueless, and physical parameters such as whether, in the case of a walk verb, the figure is walking uphill, or downhill and whether the motion is curving to the left or right. The set of adverb axes define a multidimensional adverb space of all possible variations for a particular verb.

Verbs are constructed from sets of similar, but distinct example motions. These examples can be obtained 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 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.* [5] present methods to ensure consistent DOF values for motion captured data.

Each example motion is annotated by hand with a set of adverb values, placing the example somewhere in the adverb space. Additionally, a set of "keytimes", instants when important structural elements such as a foot-down occurs, must be specified. These keytimes are used to preprocess the examples into a canonical timeline for interpolation and for later synthesis of the phrasing for the interpolated motions. Finally, examples are defined by a set of intermittent constraints (e.g., that the foot should not slip while in contact with the floor).

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

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

where the $\theta_{ij}(T)$ are the DOF functions for the $i^{th}$ example $M_i$, $\mathbf{p}_i$ the location of the example in the adverb space, and $K$ the set of keytimes which describe the phrasing (relative timing of the structural elements) of the example. The time parameter, $T$, represents clocktime but with the clock starting at 0 at the beginning of the motion example.

## 3.1 Example Motions

Each example motion $M_i$ is defined by of a number of DOF functions, denoted by $\theta_{ij}(T)$. Each $\theta_{ij}$ (i.e., the $j^{th}$ DOF for the $i^{th}$ 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 $i^{th}$ example $M_i$. Bartels *et al.* [3] contains 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.

## 3.2 Time Warping

The keytimes are used to 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 to be 0. The last keytime, $K_{NumKeyTimes}\}$ marks the duration of the verb. 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 $K_{NumKeyTimes}$

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

for the largest $m$ such that $T > K_m$ and keeping in mind that $t(0) = 0$. This mapping is illustrated in Figure 1. In other words, at each keytime, the generic time $t(K_m) = \frac{m-1}{NumKeyTimes-1}$. At the third of four keytimes, $t = \frac{2}{3}$ as the keytimes will be at 0, $\frac{1}{3}$, $\frac{2}{3}$, and 1. Between keytimes, $t$ is linearly interpolated.

Once all the examples have had their time reparameterized to generic time $t$, they are said to be in a canonical form. For a given $t$, all examples will be at the same structural point of the motion, regardless of phrasing or overall duration. Examples are interpolated within this generic timeframe. The keytime equations 1 themselves are also interpolated between examples. The inverse of this interpolated timing function is then applied to untimewarp the interpolated motion to recapture phrasing and duration.

## 3.3 Inverse Kinematic Constraints

In addition to being used to timewarp the motions, keytimes also specify periods during which kinematic constraints need to be enforced. For example, keytimes of a walk 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 (e.g., the foot) should maintain are not set when they are designed, but are set at playback when a keytime that triggers a constraint is

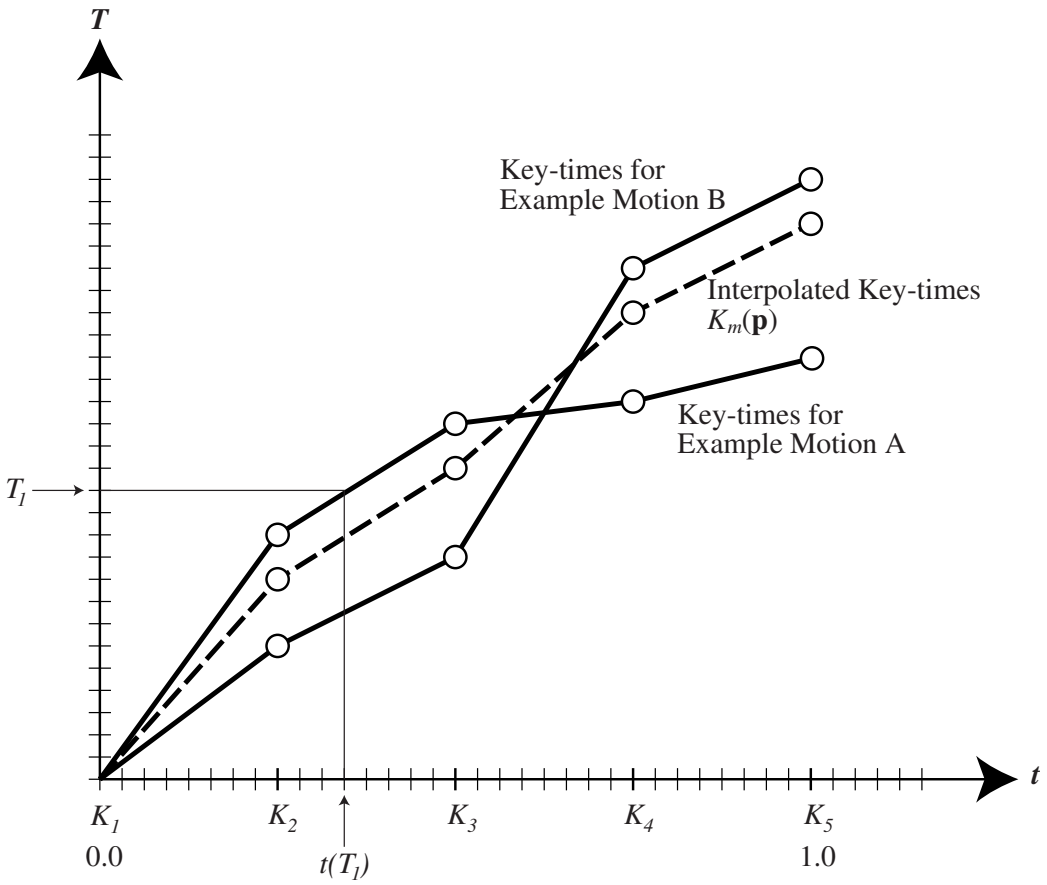


Figure 1: Mapping Between Generic Time and Keytimes

crossed. If, for example, when a keytime is crossed triggering a foot constraint, the foot's horizontal location is at $(x, z)$ and the height of the floor at that point is $y = floor(x, z)$, then the constraint location is set to $(x, y, z)$. This constraint position is maintained until another keytime releasing the constraint is crossed.

Constraints are enforced at runtime with a fast inverse kinematics optimization. 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. They make the subtle changes needed to exactly satisfy the constraint, we will only consider modifying DOFs between the root and the end effector. The changes in these DOFs needed to satisfy the constraint are found at each frame at runtime by solving the linear system (typically of size 5 or 6) of the form

$$\mathbf{J}\,\boldsymbol{\Delta\theta} \;=\; \boldsymbol{\Delta}\mathbf{x} \tag{2}$$

where $\mathbf{J}$ is the Jacobian of the DOF with respect to the motion of the end effector, $\boldsymbol{\Delta}\mathbf{x}$ is the vector from where the end effector was placed by the verb to the constraint point, and $\boldsymbol{\Delta\theta}$ is the amount the DOF needs to be perturbed to hold the end effector in place. See Girard [7] or Watt [17] for details on such inverse kinematic problems.

# 4 Verb Construction

Once the example motions have been specified for a verb with their associated keytimes and adverb settings, the next step is to construct a continuous "space" of motions parameterized by the adverbs. The dimension of this space is the number of adverbs, $NumAdverbs$. A point in this space is defined by the specific values of the individual adverbs. 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 downhill. The goal is to produce at any point $\mathbf{p}$ in the adverb space a new motion $M(\mathbf{p}, t)$ derived through interpolation of the example motions. When $\mathbf{p}$ is equal to 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 B-spline control point for each DOF 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 allows us to treat the example motion interpolation as a separate problem for each control point and each keytime (i.e., $NumCP \times NumDOF + NumKeyTimes$ individual interpolation problems).

The standard problem of multivariable interpolation is as follows: 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, a combination of radial basis functions and low order (linear) polynomials was selected 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:

$$R_i\left(d_i(\mathbf{p})\right)$$

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 [10] and in the survey article by Powell [13]. Radial basis functions have been used in computer graphics for image warping by Ruprecht and Müller [15], and Arad *et al.* [2].

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

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

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

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

For each verb there are $(NumCP \times NumDOF)$ control point interpolations (eq. 3) and $NumKeyTimes$ keytime interpolations (eq. 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 first solving for the linear coefficients and then for the radial basis coefficients.

## 4.1 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 $l^{th}$ 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 $i^{th}$ 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

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

It is the job of the radial basis to interpolate these residuals.

## 4.2 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 $\mathbf{p}_i$ (i.e., where

$d = 0$). We would also like the radial bases to have compact support (i.e., have value zero beyond some distance) to limit each example motion's influence to a local region of adverb space, thus allowing for local refinement of the verbs. There are a number of choices for the specific shape of the radial basis. For its combination of simplicity and $C^2$ continuity, in this work we chose to use a radial basis with a cross section of a dilated cubic B-spline, $B(d/\alpha)$. The dilation factor, $1/\alpha$, is chosen 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 $\alpha$ 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 $\mathbf{D}$ is a square matrix with terms equal to the value of the radial basis function centered on motion $i_1$ at the location of motion $i_2$. Thus

$$\mathbf{D}_{i_1,i_2} = B_{i_1}\left(\frac{R_{i_1}(\mathbf{p}_{i_2})}{\alpha_{i_1}}\right).$$

# 5 Verb Graphs

In addition to the construction of 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 there are multiple transition arcs leaving a verb, each transition may be given a "likelihood" of transitioning, 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 uphill/downhill may have no meaning to a "sit" verb. The verb graph is static, that is, fixed at authoring time.

## 5.1 Transitions

Given corresponding time intervals (as set by the designer) in two verbs, transitions smoothly move control 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 is a mapping of 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 these times are expressed as generic times since the verbs $A$ and $B$ may have different durations based upon their current adverb settings. In other words, the transitions are designed based on the similarity of the structure of the motions (generic time) and so will work even though the verbs themselves are changing at runtime.

The duration of the transition is calculated by taking the average of the lengths of the two blending regions

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

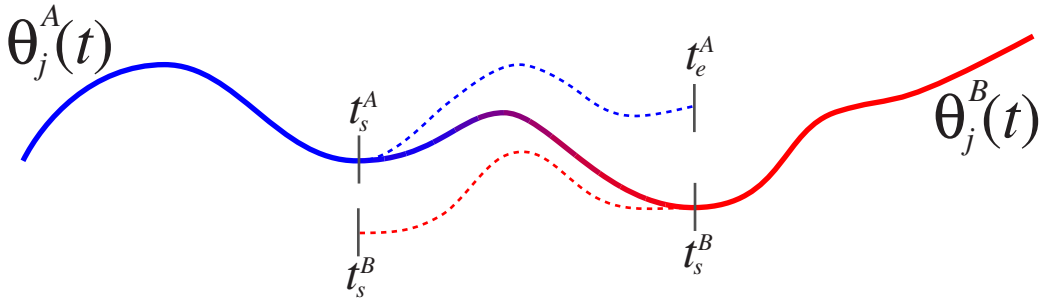


Figure 2: Transition Blending During Correspondence Region

where the $T$ function maps generic time to real time (keytime time) for the verb's current adverb settings. Transitions, as 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.

Spacetime transitioning mechanisms were discussed by the authors in [14]. Generation of spacetime transitions between two motions is still an offline process. Since the specific motions the verbs generate are not known until runtime, this method cannot be used here. Thus, in this work we will rely on a simpler formulation to allow runtime calculation of the transitions between verbs.

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

Over the transition duration, $\beta$ moves linearly from 0 to 1 representing the fraction of the way through the transition intervals in each verb. The transitional motion is determined by linearly interpolating the verbs $A$ and $B$ with weights $\alpha$ and $1-\alpha$ respectively. The internal DOFs are found by interpolating the joint positions, as shown in Figure 2, which shows the DOF function $\theta_j$ as a combination of the two DOF functions $\theta_j^A$ and $\theta_j^B$ during the transition region. The path the joint takes is defined by $\theta_j^A$ before the transition, $\theta_j^{A\rightsquigarrow B}$ during the transition, and $\theta_j^B$ after the transition. Smooth motion for the root DOFs is achieved by interpolating the velocities (rather than positions) of the two verbs and then integrating the results through the transition.

## 5.2 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, a search is executed to determine a sequence of transitions and verbs that will reach the desired action from the one currently playing. The shortest path, where shortness is defined as the number of needed transitions, is chosen. This path through the verb graph is represented as a queue of transition, verb, transition, verb, and so on.

Some bookkeeping is required to keep the verb graph in a useful state. The motion of the root of the creature must be stored and maintained at each time step in order 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 where the creature was 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 allowed to empty or the creature will come to a stop. For the verb graph to continue, it must be able to transition from its current action before that action has completed. When only the currently active verb remains on the queue, a transition is automatically selected from the set of available transitions away from that verb. The selection is made stochastically according to weights the designer specified during verb-graph construction. In cyclic verbs such as walking the default (with probability one) transition is usually the one leading back into itself.

# 6 Runtime Verb Evaluation

Given (a) the authored verbs parameterized by adverbs and time warped by keytimes, and (b) a verb graph to organize the verbs and transitions, we are now ready to see how these structures operate at runtime. The goal is to allow continuous variation of the adverb settings, and to have these changes reflected in the subtle motion of the simulated creatures. In other words, the user or application defines the path that the point $\mathbf{p}$ will take on the fly, and the runtime system must respond accordingly. Given the time and setting of $\mathbf{p}$, the evaluation of the pose of the creature must be fast enough to allow interactive frame rates.

The main loop of the runtime system is a discrete event simulator. 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. Events are inserted in the event queue in time stamp order and are processed by invoking the callback function with the timestamp as a parameter. Events may be of one of three types: $normal$, $sync$, or $optional$. Normal events are processed as they are reached on the queue, independent of the relative values of the timestamp and the clock. Sync events wait for the clock to catch up to the timestamp if the clocktime is less than the timestamp; they execute immediately otherwise. Optional events are skipped if the clock has passed the timestamp; otherwise, optional events act like normal events.

The most common events are "render" and "display" events. A render event evaluates the DOFs at the time indicated by the timestamp to set the pose of the creature and 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 is reached on the queue. A display event with the same timestamp but with a sync flag waits for the clock to reach the timestamp and displays the rendered image. The render event also measures the amount of clocktime between frames and estimates the best timestamp for the next render/display events and inserts them on the event queue. In this way the frame rate dynamically adjusts to the computational load.

Other events are scheduled to induce transitions between verbs or to turn on or turn off constraints.

## 6.1 Evaluating DOF

The render event callback requests all DOFs to be evaluated at a given time, $\tau$. The currently active verb (or verbs when a transition is occurring) is evaluated at time $\tau - \tau_{offset}$ with the current adverb settings, $\mathbf{p}$. $\tau_{offset}$ is the clock time when the current verb or transition came 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 = InterpolateKey(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 = InterpolateBSCoeff(j, k, p)   // Eqn. 3
12     Next
13     θ_i = Σ_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 need to be evaluated at each frame time. The values computed at the other steps are cached. The interpolations, lines 4 and 11, only change when the parameters change or a new verb is invoked. In addition, in line 11, only 4 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 new one is added as the knot value is passed. Thus on average, less than one of the interpolated coefficients per DOF need be evaluated per frame if $\mathbf{p}$ is unchanging. If $\mathbf{p}$ changes from frame to frame, 4 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.

# 7 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, as well as walks at different inclinations and radius 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 degree-of-freedom sensors positioned on the body. The raw data must be preprocessed to fit to a rigid-body model with hierarchical joint rotations and fewer degrees of freedom that correspond to the limitations of human joints. The methods described by Bodenheimer *et al.* [5] ensure that the motion capture data makes consistent use of joint angles for redundant DOFs. The final model has 40 joint degrees of freedom in addition to six degrees of freedom of the root, located between the hips, for global positioning and orientation.

The authors constructed a parameterized walk along two emotional axes–happy-sad and knowledgeable-clueless–as well as physical characteristics such uphill/downhill, and turning. A sampling of this walk across the two emotional axes is shown in Figure 3. A reaching verb was parameterized by the 3 positional values representing the goal of the reach (see Figure 4). Various emotional idle motions were constructed as was a jog with a turning adverb.

Parameterizing a space based on turning and on changing elevation gives us a great degree of control over the creature. Both the jogging and walking turn adverbs were created from two motion captured example motions each, a forward motion and a motion to the right. A third example motion to the left was created by mirroring the motion to the right. The interpolation gives convincing control of the radius of curvature of

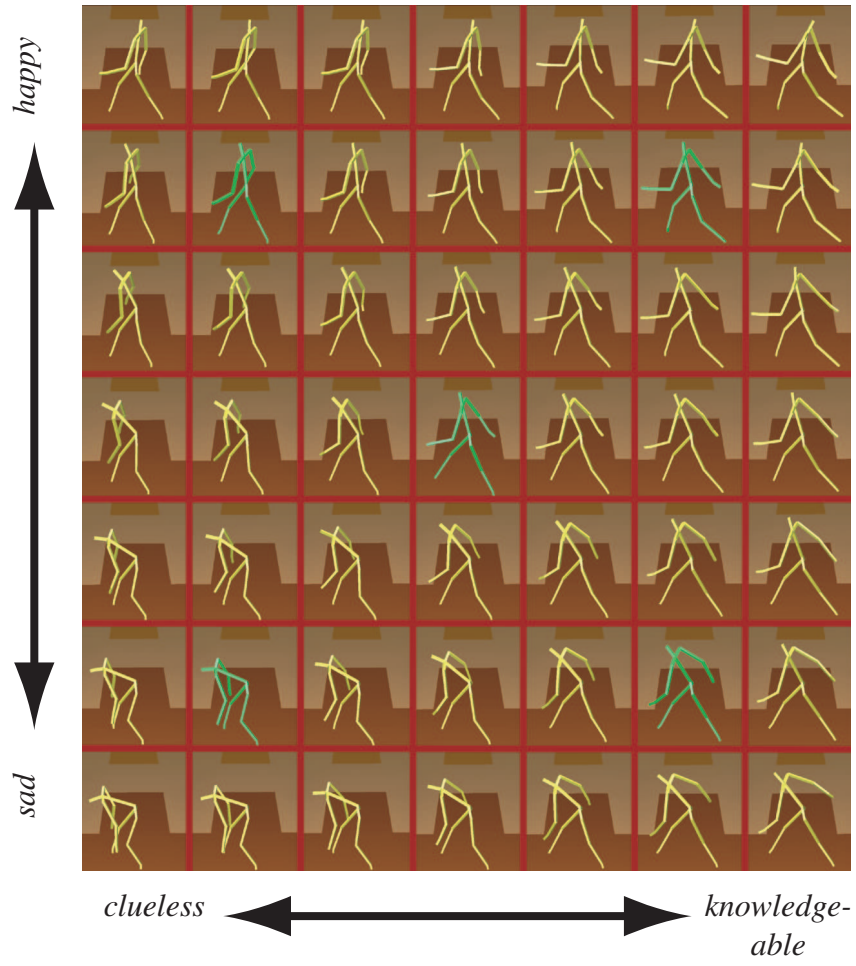


Figure 3: A walk sampled across two emotional axes. The green figures are the example motions. The rest are created through the verb/adverb mechanism.

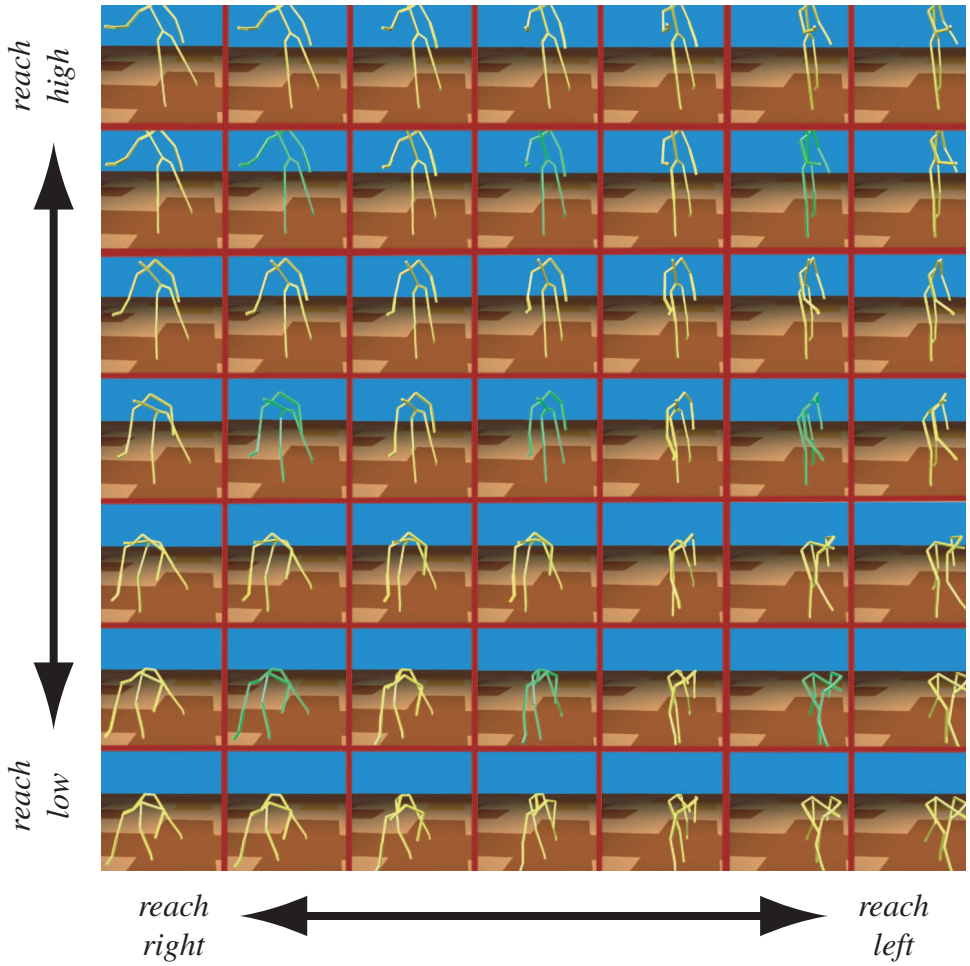


Figure 4: A reach sampled across two axes of the goal position for the hand. The green figures are the example motions. The rest are created through the Verb/Adverb mechanism.

turns, and allows convincing navigation.

Solving for the coefficients of the interpolation took about 2 minutes on a 200Mhz PentiumPro processor for the most complex verbs. Recall that this offline computation need only be carried out once per verb. At runtime, the evaluation of the position of the character as described in the pseudocode above can be carried out at approximately 200 Hz., or about 5 milliseconds, thus barely impacting the overall framerate. This timing was taken with a constantly changing $\mathbf{p}$, thus requiring interpolating the full 4 coefficients per DOF per frame plus the $m$ keytimes.

We are able to transition through the adverb space and through the verb graph to exhibit a variety of degrees of emotions and subtleties of movement. Finally, 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.

# 8 Conclusions

This paper has demonstrated a method to reuse and modify motion clips of complex anthropomorphic linked figures. A variety of motion is derived by building "verbs" parameterized by "adverbs." The interpolations of verbs are built through the use of radial basis functions and low order polynomials. Similar methods are used to time warp each example motion into a canonical time frame.

The interpolation method is general and scales well with the dimensionality of the problem (number of adverbs). The resulting animations show richer content than existing techniques of modifying animation, or procedural or physics-based approaches.

Additionally, through the creation of verb graphs, we demonstrated the ability to create characters capable of a wide variety of behaviors and expressiveness. 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. Our goal is the eventual creation of 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, that being the talent and inspiration of the animator constructing the example motions, and/or 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.

More information about this work, video segments in particular, can be found at the Human Figure Animation Project's website at `http://www.research.microsoft.com/research/graphics/hfap`.

# Acknowledgments

The authors 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 paper.

# References


[1] Amaya, K., Bruderlin, A., and Calvert, T. Emotion from motion. In *Graphics Interface '96* (May 1996), W. A. Davis and R. Bartels, Eds., pp. 222–229.

[2] Arad, N., Dyn, N., Reisfeld, D., and Yeshurun, Y. Image warping by radial basis functions: Application to facial expressions. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing 56*, 2 (Mar. 1994), 161–172.

[3] Bartels, R. H., Beatty, J. C., and Barsky, B. A. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[4] Blumberg, B. M., and Galyean, T. A. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics* (Aug. 1995), pp. 47–54. Proceedings of SIGGRAPH 95.

[5] Bodenheimer, B., Rose, C., Rosenthal, S., and Pella, J. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, D. Thalmann and M. van de Panne, Eds. Springer NY, Sept. 1997, pp. 3–18. Eurographics Animation Workshop.

[6] Bruderlin, A., and Williams, L. Motion signal processing. In *Computer Graphics* (Aug. 1995), pp. 97–104. Proceedings of SIGGRAPH 95.

[7] Girard, M., and Maciejewski, A. A. Computational modeling for the computer animation of legged figures. In *Computer Graphics* (July 1985), pp. 263–270. Proceedings of SIGGRAPH 85.

[8] Guo, S., and Roberge, J. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Computer Animation and Simulation '96*, R. Boulic and G. Hégron, Eds. Springer NY, Aug. 1996, pp. 95–107. Eurographics Animation Workshop.

[9] Hodgins, J. K., and Pollard, N. S. Adapting simulated behaviors for new characters. In *Computer Graphics* (Aug. 1997), pp. 153–162. Proceedings of SIGGRAPH 97.

[10] Micchelli, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation 2* (1986).

[11] Perlin, K. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (Mar. 1995), 5–15.

[12] Perlin, K., and Goldberg, A. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics* (Aug. 1996), pp. 205–216. Proceedings of SIGGRAPH 96.

[13] Powell, M. J. D. Radial basis functions for multivariable interpolation: A review. In *Algorithms for Approximation*, J. C. Mason and M. G. Cox, Eds. Oxford University Press, Oxford, UK, 1987, pp. 143–167.

[14] Rose, C. F., Guenter, B., Bodenheimer, B., and Cohen, M. Efficient generation of motion transitions using spacetime constraints. In *Computer Graphics* (Aug. 1996), pp. 147–154. Proceedings of SIGGRAPH 96.

[15] Ruprecht, D., and Muller, H. Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications 15*, 2 (Mar. 1995), 37–43.

[16] 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.

[17] Watt, A., and Watt, M. *Advanced Animation and Rendering Techniques: Theory and Practice*. ACM Press, 1992.

[18] Wiley, D. J., and Hahn, J. K. Interpolation synthesis for articulated figure motion. In *Proceedings of the Virtual Reality Annual International Symposium* (Mar. 1997), IEEE Computer Society Press, pp. 157–160.

[19] Witkin, A., and Popovic, Z. Motion warping. In *Computer Graphics* (Aug. 1995), pp. 105–108. Proceedings of SIGGRAPH 95.