Interpolation Synthesis of Articulated Figure Motion

Douglas J. Wiley
Naval Research Laboratory

James K. Hahn
The George Washington University

Most conventional media depend on engaging and appealing characters. Empty spaces and buildings would not fare well as television or movie programming, yet virtual reality usually offers up such spaces. The problem lies in the difficulty of creating computer-generated characters that display real-time, engaging interaction and realistic motion.

Articulated figure motion for real-time computer graphics offers one solution to this problem. A common approach stores a set of motions and lets you choose one particular motion at a time. This article describes a process that greatly expands the range of possible motions. Mixing motions selected from a database lets you create a new motion to exact specifications. The synthesized motion retains the original motions' subtle qualities, such as the realism of motion capture or the expressive, exaggerated qualities of artistic animation. Our method provides a new way to achieve inverse kinematics capability—for example, placing the hands or feet of an articulated figure in specific positions. It proves useful for both real-time graphics and prerendered animation production.

Interpolation synthesis based on motion capture data provides real-time character motion for interactive entertainment or avatars in virtual worlds.

Character motion for virtual reality

Video production in computer-based and traditional animation relies largely on the labor-intensive process of keyframing.[1] This does not permit creating real-time characters with intriguingly rich varieties of motion. Motion capture allows more rapid production of character motion, but still suffers from the inherent lack of control of prerecorded motion data. Physical simulation approaches offer promise, but are hampered by lack of control, difficulty of use, instabilities, and computational cost that usually precludes real-time operation.[2]

The predominant approach to real-time motion synthesis, selection of one stored motion at runtime, restricts computation to the stored motion's display. Unfortunately, a limited number of stored motions can result in repetitive or incorrect motions.

The repertoire of possible motions expands greatly—at the cost of additional computation—with interpolation synthesis. This technique combines a set of motions similar to that desired to form an exactly specified motion. A small set of example motions then yields a continuous multidimensional space of possible motions. The example motions can come from keyframing, physical simulations, or motion capture. Because the interpolation process generally preserves the motion's subtle qualities, we can create reusable libraries of motion data from a single step of laborious keyframing, iterative estimation of initial conditions, or correction of errors and dropouts, respectively.

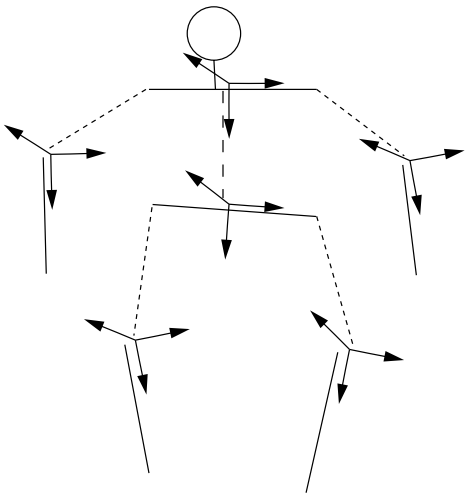
This article focuses on real-time interpolation synthesis of motion based on motion capture data. This approach works well for real-time character motion in interactive entertainment or for avatars in multiplayer networked games or virtual worlds. After transmission of a motion database, network traffic would consist merely of a motion specification indicating parameters of the interpolation.

Related work

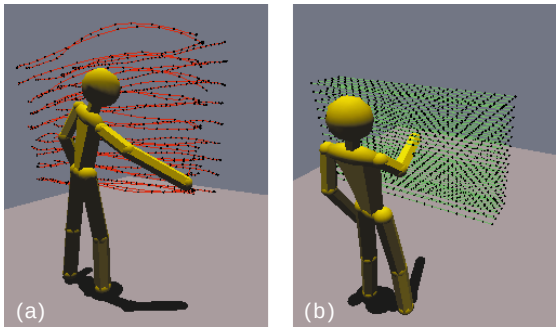
Traditional hand-drawn animation follows principles derived from extensive study of film and live subject motion, and uses motion data in a nonmathematical way.[1] Many motion and human animation models feature adjustable parameters. Some facial models use anatomical and empirical data on muscle structure[3] and perception of expression.[4] Parameterized gait models employ observation-based physical simulation and kinematic principles,[5,6] and some recent efforts focus on manipulation and reuse of motion data.

Previous authors have described pairwise motion interpolation and explored the use of a stored database to create motion.[7-10] One study demonstrated parameterized gait motion synthesis.[9] One technique of combining keyframed data with motion capture data allows insertion of a specific figure pose into a recorded sequence.[7,10] These authors used joint angle interpolation as well as frequency-based representations. Using

0272-1716/97/$10.00 © 1997 IEEE



1 Position/ quaternion pose representation corresponding to magnetic tracker data.



2 (a) Initial and (b) resampled hand position data.

nonuniform time scaling required aligning the starting and ending times of motions before interpolation.[7] Another work described sequencing different stored motions over time with random inputs.[8]

Using intuitive parameters, we can invoke real-time locomotion from a stored library of motions.[11] Other work has addressed the problem of storage and retrieval of real-time motion with multiple levels of detail.[6] Recent efforts have demonstrated real-time inverse kinematics for a portion of an articulated figure body using modified robotics formulations.[12]

The stability of the particular figure representation used here allows a greater degree of interpolation than shown previously. We can combine motions of greater dissimilarity and subsequently increase the range of interpolation synthesis results. We illustrate this point by using interpolation to achieve inverse kinematics capability, generating joint angles to place the hands or feet of an articulated figure at any specific point in space. Our method does this by direct computation, unlike the more common robotics formulations, and without defaulting redundant degrees of freedom arbitrarily.

Articulated figure representation

An articulated figure has rigid limbs and joints with one, two, or three degrees of freedom. Some motion models use Euler angles to specify a figure's pose, defining each limb's position relative to its attachment point in a hierarchy. Instead, we use a hybrid position and orientation representation corresponding to data from magnetic trackers. The center of rotation of the lower limbs, chest, and pelvis is specified in absolute terms (trihedral origins), as Figure 1 illustrates. A rigid lower limb, pelvis, or chest segment is attached at the tracker positions (solid lines), while a line between shoulder or pelvis points and lower limb origins (dashed lines) defines upper limb and torso sections. A quaternion specifies the absolute orientation of each segment.[13]

A simple conversion between this hybrid representation and joint Euler angle forms allows the use of interpolation synthesis within conventional character animation software environments. While the hybrid representation does not ensure constant upper limb or torso length, we can add a step of conversion to fixed limb lengths and orientations when such accuracy is desired. The lower arm, for example, can be adjusted along the line between the elbow and shoulder point to the correct distance.

An articulated figure skeleton often forms part of a more complicated character with additional muscle and skin layers. Because the skeleton's motion controls the entire character, consideration of the skeletal motion alone suffices in these cases.[14]

Interpolation synthesis of hand position

Interpolation synthesis produces new motions, not directly recorded, from a mixture of data motions. Given a desired motion's specification, we simply find a subset of the data motions that are most similar to the desired motion and that "surround" the desired motion in the parameter space.

We can illustrate these two concepts with the example of single-hand inverse kinematics capability. We generate a single static pose that places one hand at a desired point in space from a mixture of static poses that place the hand in a variety of positions. We specify a desired pose using three parameters: the hand's position in three dimensions. Interpolation synthesis of the desired pose requires finding a subset of the data poses with hand positions close to the desired point that form a volume containing that point.

A lack of precision often arises from the collection of specific motion capture data. This makes it difficult to select the required data subset for interpolation synthesis, and the best solution is often exhaustive search of the data. Searching on one dimension at a time does not work for data that does not lie on a regular grid (see Figure 2a). An exhaustive search might suffice for small data sets, but is inefficient for the hundreds of data poses in this example.

To address this, we resample the data set to a regular grid of the parameter space (see Figure 2b). We can do this on one dimension at a time or apply the entire interpolation synthesis process to a set of positions on a regular grid. Subsequent interpolation synthesis uses these resampled data motions. Efficient selection of the required data subset then doesn't require a search, just a division by grid point spacing in each dimension.

Now that we have found the required subset of initial or resampled data, we form the correct mixture by multiple interpolation. Figure 3 depicts data hand positions surrounding a desired hand position point t. We first

apply a binary-tree progression of interpolations on each successive dimension. We derive the interpolation coefficients (mixture proportions) from the hand's geometrical position relative to the data hand positions. Each stage of the progression independently interpolates each position and orientation component of the pose representation.

We use linear interpolation for each vector position component of the representation (a, b, and result c):

$$c = (1-u)*a + u*b, \quad u \text{ in } [0,1] \tag{1}$$

We use spherical linear interpolation[13] for each quaternion orientation component (p, q, and result r):

$$r = q*\frac{\sin((1-u)W)}{\sin(W)} + p*\frac{\sin(uW)}{\sin(W)},$$
$$u \text{ in } [0,1],\ W = \arccos\ q \bullet p \tag{2}$$

In general, we obtain interpolation coefficient u by assuming linearity of the inputs with respect to the outputs in each dimension. Given target t lying between grid points l and h,

$$u = \frac{t-l}{h-l} \tag{3}$$

We use the same interpolation coefficient for the corresponding quaternion interpolations.

We first perform four interpolations along the x axis. We use interpolation coefficient $u00$ (Equation 4, below) to interpolate each of the six position and orientation components of data poses $d000$ and $d100$, resulting in intermediate pose $i00$. The points shown in Figure 3 denote hand position in space corresponding to a particular data or intermediate interpolation result pose.

$$u00 = \frac{t_x - d000_x}{d100_x - d000_x} \tag{4}$$

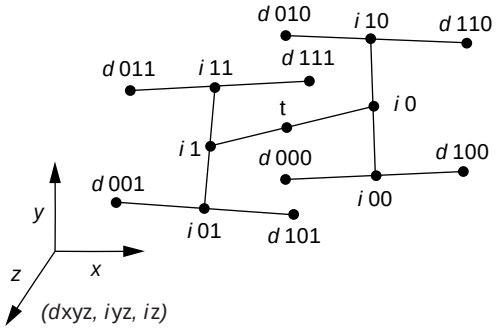
We obtain Poses $i01$, $i10$, and $i11$ similarly. We then interpolate these intermediate poses based on the target y-axis position using

$$u0 = \frac{t_y - i00_y}{i10_y - i00_y} \tag{5}$$

We make a final interpolation of the poses $i0$ and $i1$ based on the target z-axis position.

This direct computation results in a pose that, given sufficiently dense data, closely approximates the target goal. We can enforce limb length if necessary and can accommodate higher accuracy or sparse data using iterative optimizations.

Three degrees of freedom suffice for the simple character representation used here. Six degrees of freedom would merely require sufficient data varying in both position and orientation, and a higher dimensional interpolation procedure.



3 Trilinear interpolation binary tree: four interpolations in the *x* direction, two in the *y* direction, and a final interpolation in the *z* direction.

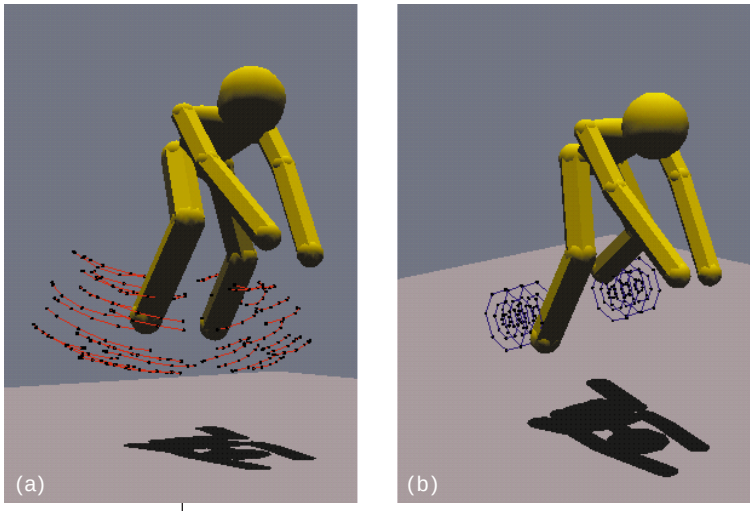
Motion Capture Technology

The most widely used full-body motion capture technologies, wireless optical and magnetic tracking systems, are available with and without wires running to the performer. A third technology, less widespread, involves mechanical sensing of joint angles directly using an exoskeleton.

Optical systems have their roots in medical gait analysis systems. Most systems use passive reflective markers, as opposed to active markers (small light sources). Numerous small spherical reflectors are attached to the performer, and multiple cameras record how they reflect infrared light-emitting diode strobe lights mounted on the cameras. The 2D images are combined to form 3D data by triangulation of the angle-of-arrival position data. These systems can track more than 100 points simultaneously but require that at least two cameras maintain an unobstructed view of each marker at all times to avoid data dropouts. A required postprocessing calculation and/or manual point identification step prevents real-time operation.

Magnetic systems are "nonblocking," that is, they do not require a clear line of sight. A single transmitter broadcasts electromagnetic signals to many small receivers that report position as well as orientation. A cable connects the receiver to the computer or to a small electronics enclosure on the performer's body containing a transmitter or recording device. Magnetic systems typically have up to 30 receivers and deliver real-time 3D data, allowing interactive "puppeteering" applications. They are, however, sensitive to metal, magnets in audio speakers, and monitor radiation in the capture volume.

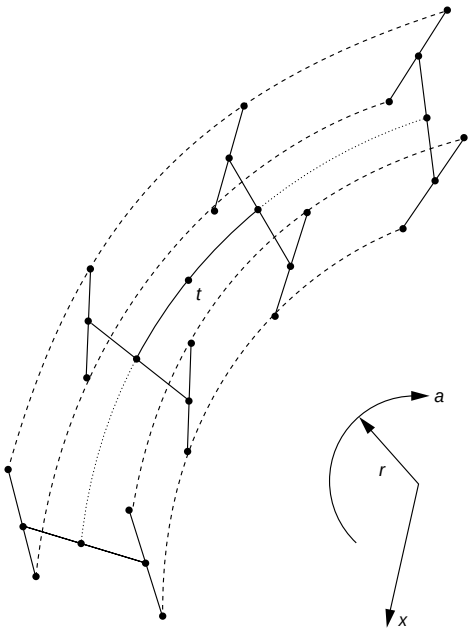
Exoskeletons, goniometers, and bend sensors represent the first type of motion capture technology. Relatively inexpensive mechanical devices and a computer interface form a complete system, but designing an exoskeleton that works for performers of widely varying body types presents a challenging problem. Available angle sensors include rotary optical encoders, potentiometers, and electrical resistive and optical fiber bend sensors. Real-time 3D data are delivered as joint angles, the same data format used in most animation software environments. This does not determine the performer's position and overall orientation, however, which necessitates some other method.



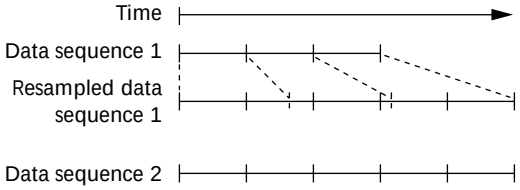
4 (a) Initial and (b) resampled foot position data.

Interpolation synthesis of foot periodic motion

The previous example constructed a pose placing one hand in a specific position described in Cartesian coordinates; we also resampled the data to a regular grid in Cartesian coordinates. We now demonstrate inverse kinematics capability for foot position in a cylindrical coordinate system, resampling the pose data to a regular grid in cylindrical coordinates. The foot position periodic angle parameter maps naturally to a bicycle riding motion.



5 Data subset for foot inverse kinematics with cubic spline interpolation of angle parameter.



6 Resampling an interpolated sequence to rescale to a uniform number of samples (longest of set is sequence 2).

We gathered data from a motion capture actor performing outward motions of both feet on a stationary bicycle, shown in Figure 4a. We first resampled the pose data to a regular grid in cylindrical coordinates by the previously described Cartesian interpolation synthesis using exhaustive search (see Figure 4b). We then performed real-time pose synthesis by directly selecting resampled pose data.

Because the periodic angle parameter is nonlinear, this example uses a higher order (cubic spline) interpolation, whereas the previous linear and spherical linear interpolation suffices for the other two parameters. Catmull-Rom cubic splines[13] interpolate between the middle pair of a four-data-point window; this requires extending the previous notion of data points that "surround" the desired foot position. We must find three adjacent volumes where the middle volume contains the desired foot position t (see Figure 5). We perform eight linear and spherical linear interpolations along the cylindrical axis x and four along the radial axis r on the four pairs of intermediate pose results; we carry out a final cubic interpolation in angle a on the four results of the previous step. Interpolation coefficients result as before from desired foot position relative to the data foot positions. Carried out independently for each foot, this process creates a pedaling motion using an increasing value for angle and constant values for the other two parameters (with an angle difference of π between the two feet).

Motion interpolation synthesis

The previous example synthesized static pose instantaneously from a specification varied in real time. The interpolation synthesis technique also extends to synthesis of entire motions of finite duration or finite period; here we consider a motion as a time sequence of poses. We illustrate by generalizing the principles discussed above to synthesis of a walk motion on a continuously variable slope.

We recorded a motion capture actor walking up and down five different actual slopes. Blending the data at the ends of the walk sequence formed a seamless loop. A single linear interpolation of the correct pair of walk data motions generated motion. The data was taken on a regular grid of ground slope angles, which made resampling to a regular grid in the parameter space unnecessary. However, the walk cycles took more time with increasing slope, necessitating resampling in time. This process benefits general interpolation of motions of different lengths or periods.

Interpolation between poses at different points in time lets us generate a consistent pose at any time. This permits resampling a motion to a greater or lesser number of poses. We can resample each walk cycle to a uniform (longest of set) number of samples (as shown in Figure 6), and then interpolate between two walk cycles pose by pose, at each point in (adjusted) time. We also calculate an interpolated time duration to preserve dynamical realism, resampling the interpolation of two walk cycles to this interpolated duration. Walk cycle interpolation using an interpolation coefficient of zero or one then recovers the data's original durations, and in practice

appears correct at intermediate values as well.

We obtain the interpolation coefficient from the desired slope relative to the data slope values just above and below the desired slope. A walk cycle for any desired slope in the range of the data slopes results, with consistent duration.

Reach motion interpolation synthesis

Interpolation synthesis can be specified by a value that occurs over the entire motion (for example, walk) or at a specific point in time. That point in time may also vary between data motions, as in the next example of a reach motion. The figure stands at rest, arms at its sides, then reaches out to a target point, then returns to the original position. Given a target point, we wish to generate a sequence of poses that produce the appropriate reach.

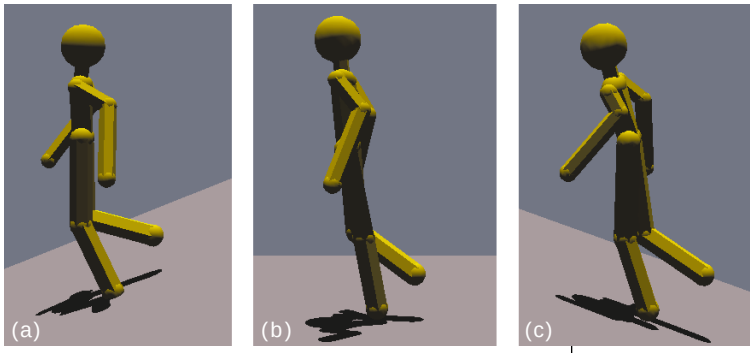
We divide the reach motion into two segments: motion from rest to the target point, and return to rest. This ensures phase alignment of the data for interpolation and accounts for the two segments' data-dependent time durations. Each segment is handled independently according to the motion interpolation procedure (of different durations) described earlier.

The reach synthesis problem is very similar to single-hand inverse kinematics static pose synthesis. A single pose forms the boundary of the two segments of each data motion to control the process. We obtain the required data subset and interpolation coefficients from the hand position (point that was "reached to") at these boundary poses. We apply the same multiple interpolation at every other time point of the time-resampled uniform duration motions. Finally, we resample the uniform duration result segments in time to their respective interpolated durations.

Implementation specifics

All examples ran in real time on a 100-MHz R4400 processor workstation. We entered interpolation synthesis parameters interactively. The inverse kinematics examples produced smooth, continuous motion in response to continually varying parameters.

The single-hand inverse kinematics example employed four vertical planes of data consisting of nine lines traced out by the motion capture actor. We first resampled the data along each line in the figure's left-to-right direction on a regular grid, then connected the results vertically and resampled them at a regular grid in height, yielding four planes of data on a regular grid in width and height but not depth. Finally, we formed and resampled lines connecting grid points in each plane at a regular grid in depth. We then demonstrated real-time figure positioning by trilinear interpolation with interactive goal point input. Convincing subtleties such as knee bending, back bending, and motion of the other arm occur during operation, increasing the realism of the figure's motion. We observed no apparent inaccuracy in positioning in this demonstration because we used approximately 100 data points in each plane. The resulting motion looks convincingly like direct motion capture due to the data's motion capture origin. Synthesis using expressive and exaggerated motion from keyframing would conceivably retain these qualities as well.



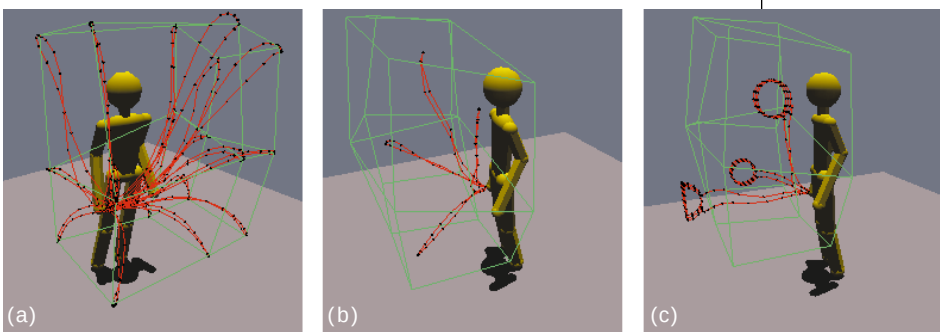
7 Variable slope walk cycle: (a) downhill, (b) level, and (c) uphill.

We resampled foot position data to three rows of three concentric rings for each foot, with eight samples in each ring. We used cubic interpolation to follow a circular path. No additional delay in real-time performance resulted in spite of roughly twice as much computation compared to linear interpolation.

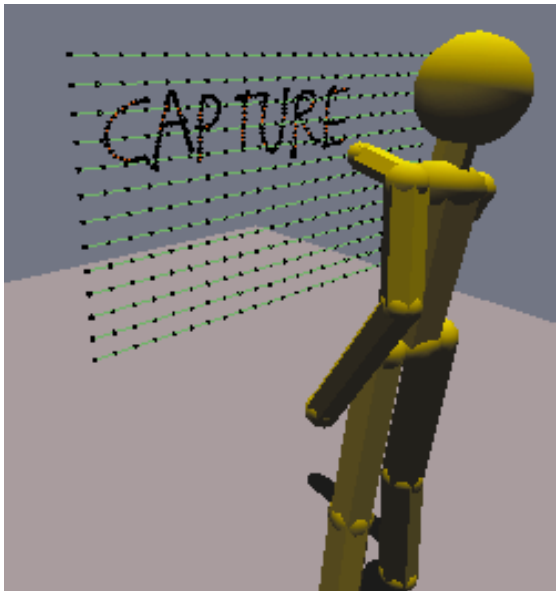
The variable slope walk example used a single step of motion interpolation. We altered slope interactively, and both gait and gait cycle duration changed continuously in real time (see Figure 7). The resulting realism would be difficult to generate by keyframing or conventional robotics inverse kinematic techniques.

We also synthesized real-time reach with interactive goal point positioning by recording reach motion data for two planes of nine goal points (Figure 8a) and storing uniform duration reach sequences as the data for multiple interpolation. We used no spatial resampling since only four possible interpolation windows existed. We generated reach motion using limb length enforcement and iterative accuracy optimization (Figure 8b).

Using the reach point for inverse kinematics, we created another demonstration, this time specifying a helical path with a reach to the initial position and return to rest from the end position. Using limb length enforcement, we deformed the helix to a "D" shape, as shown in Figure 8c. Chalkboard writing uses a variation of static pose generation. A motion capture actor wrote the entire alphabet in a single plane of space at a single position, and we recorded only the hand position. We used the alphabet hand trajectory to drive the figure pose, pro-



8 (a) Initial reach data, (b) reach to specific points in space, and (c) helical path insertion.



9 Chalkboard writing using interpolation inverse kinematics.

ducing any sequence of letters starting at any position in space upon interactive keyboard entry (Figure 9).

Analysis and discussion

Interpolation synthesis is limited to small numbers of parameters, on the order of ten. The examples presented here involved one or three parameters. The amount of data required at least doubles for every additional parameter, since the entire previous data set is repeated for at least two values of the additional parameter. In general, for p parameters, $2^p - 1$ interpolations are performed with a window of 2^p data samples. The interpolations are performed for every time step of the motion and for each component of the character (12 in the representation used here). Motion segment durations are also interpolated $2^p - 1$ times. A final time rescaling occurs for each component of the result. The interpolation window and blending coefficients occur only once in synthesis of an entire motion, as the same window and coefficients are used at each time step. Preprocessing (performed only once) consists of resampling the data to uniform length to a regular grid in the synthesis parameters.

All examples presented here involved interpolation of the pose or the entire body's motion to achieve maximum realism. Allowing interpolation synthesis over a subset of the body's degrees of freedom would allow one or more separate interpolation synthesis processes to operate at once, affording many more synthesis parameters than described here.

Storage, interpolation requirements, and final time rescaling calculations scale linearly with both the character's number of degrees of freedom and the data's time sample rate. This is important for extending the work to more detailed representations and more complicated figures, a straightforward process that could use a combination of fixed segments with implicitly defined segments constrained by length enforcement.

A trade-off exists between density of the data in the parameter space and storage requirements. Higher data density leads to greater accuracy, stability, and continuity between adjacent parameter set values. The data can be resampled at varying grid spacing as well, again trading off accuracy for storage.

The interpolation process is not guaranteed to produce physically possible motions because certain regions of the parameter space may not represent valid motions. If necessary, those regions can be identified and avoided for intermediate or final interpolation results. In general, interpolation between points close together in the parameter space is less likely to produce impossible results. We can apply explicit checks and enforcement (for example, upper limb length enforcement, described earlier) to any constraints that may potentially be violated. Joint angle limits could also be enforced if necessary, and angular and translational velocities of joints and limbs could be enforced to physically possible values. The application will dictate the necessity of explicit constraint enforcement—a fast-moving game character presents completely different requirements than an ergonomic manufacturing assembly study simulation.

The benefits of the two-stage process outweigh the drawbacks. Motion capture data that requires extensive manual correction of errors and dropouts can be corrected just once, with subsequent interpolation synthesis based on the corrected data. The trial and error involved in keyframing could also be reused by interpolation synthesis from keyframed data. The process also greatly decreases the accuracy requirements of motion capture recording sessions or simulation runs since just a representative sample set is needed, as opposed to the exact motion that will be used.

Conclusions

The interpolation synthesis process using position/quaternion representation yields stable multiple interpolation with surprisingly sparse data, as the reach

Applications of Interpolation Synthesis

Interpolation synthesis creates a new motion from a mixture of prerecorded motions. This method can be useful for interactive real-time applications and prerendered animation.

Interactive real-time applications

Interpolation synthesis permits generating motion in real time in response to the situation or user. Specific sets of motion data can be prerecorded for the kinds of motions that typify that application.

Markets include computer and console games, CD-ROM titles, virtual set characters, 3D Internet avatars, and location-based and theme park entertainment.

Prerendered animation

Interpolation synthesis also proves useful for postprocessing of motion data. Recording multiple "takes" of a desired motion gives a director the freedom to choose a motion not recorded directly but which can be constructed from the data motions.

Markets include feature films, television and video special effects, and character animation.

example shows. Iterative optimization can improve accuracy as needed, but adequately dense data with direct computation should suffice for many interactive entertainment applications. The spatial ordering and resampling techniques prove useful even if motions are used exactly as stored because they provide a direct search and choice of sampling density for stored motion. The emergence of low-cost, powerful processors and 3D graphics hardware should make interpolation calculations more feasible in future consumer graphics and networked virtual reality, however.

This approach offers a unique combination of realism and controllability for real-time motion synthesis. While limited to a small number of parameters, the technique provides a far richer range of motions than the use of prestored motions. The combination of motion capture and interpolation synthesis has the potential to animate the characters needed in currently empty virtual reality environments and add a rich variety of motion to avatars in networked virtual worlds. ■

Acknowledgments

Support for this work was provided by the Naval Research Laboratory in Washington, D.C. under contract N00014-94-2-C002 and grant N00014-94-K-2009 to George Washington University. Special thanks to House of Moves, Digital Domain, and ElektraShock of Venice Beach, Calif. for discussions about motion capture in the entertainment industry.

References

1. J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," *Computer Graphics* (Proc. Siggraph 87), Vol. 21, No. 4, July 1987, pp. 35-43.
2. A. Witkin and M. Kass, "Spacetime Constraints," *Computer Graphics* (Proc. Siggraph 88), Vol. 22, No. 4, July 1988, pp. 159-168.
3. K. Waters, "A Muscle Model for Animating Three-Dimensional Facial Expression," *Computer Graphics* (Proc. Siggraph 87), Vol. 21, No. 4, July 1987, pp. 17-24.
4. F. Parke, "A Parameterized Model for Facial Animation," *IEEE Computer Graphics & Applications*, Vol. 2, No. 9, Nov. 1982, pp. 61-68.
5. A. Bruderlin, and T. Calvert, "Goal-Directed, Dynamic Animation of Human Walking," *Computer Graphics* (Proc. Siggraph 89), Vol. 23, No. 3, July 1989, pp. 233-242.
6. J. Granieri, J. Crabtree, and N. Badler, "Off-Line Production and Real-Time Playback of Human Figure Motion for 3D Virtual Environments," *Proc. VRAIS 95*, IEEE Computer Society Press, Los Alamitos, Calif., 1995.
7. A. Bruderlin and L. Williams, "Motion Signal Processing," *Computer Graphics* (Proc. Siggraph 95), Vol. 29, No. 4, Aug. 1995, pp. 97-104.
8. K. Perlin, "Real-Time Responsive Animation with Personality," *IEEE Trans. on Visualization and Computer Graphics*, Vol. 1, No. 1, Mar. 1995, pp. 5-15.
9. M. Unuma, K. Anjyo, and R. Takeuchi, "Fourier Principles for Emotion-Based Human Figure Animation," *Computer Graphics* (Proc. Siggraph 95), Vol. 29, No. 4, Aug. 1995, pp. 91-96.
10. A. Witkin and Z. Popovic, "Motion Warping," *Computer Graphics* (Proc. Siggraph 95), Vol. 29, No. 4, Aug. 1995, pp. 105-108.
11. H. Ko and J. Cremer, "VRLoco: Real-Time Human Locomotion from Positional Input Streams," *Presence*, Vol. 5, No. 4, 1996, pp. 367-380.
12. D. Tolani and N. Badler, "Real-Time Inverse Kinematics of the Human Arm," *Presence*, Vol. 5, No. 4, 1996, pp. 393-401.
13. A. Watt and M. Watt, *Advanced Animation and Rendering Techniques*, Addison-Wesley, Wokingham, England, 1993.
14. J.E. Chadwick, D.R. Haumann, and R.E. Parent, "Layered Construction for Deformable Animated Characters," *Computer Graphics* (Proc. Siggraph 89), Vol. 23, No. 3, July 1989, pp. 293-302.

Douglas Wiley *just completed a DSc in computer science at George Washington University while working at the Naval Research Laboratory in Washington, D.C. His interests include motion capture data manipulation, real time graphics and virtual reality, and procedural modeling. He previously conducted research in optical signal processing at the Army Research Laboratory in White Oak, Maryland, after completing a PhD in Electrical Engineering at the University of Southern California. He holds BS degrees in electrical engineering, mathematics, and computer science from the University of Maryland in College Park, Maryland. He is currently working as an independent consultant and contractor.*

James Hahn *is an associate professor in the Department of Electrical Engineering and Computer Science at the George Washington University. He is the founding director of the George Washington University Institute for Computer Graphics and the Laboratory for Advanced Computer Applications in Medicine. His research interests include motion control, virtual reality, image rendering, and sound. He received BS degrees from the University of South Carolina in physics, mathematics, and computer science and an MS in physics from University of California, Los Angeles. He received an MS and a PhD in computer and information science from the Ohio State University in 1989.*

Readers may contact Wiley at 9211 Bardon Rd., Bethesda, MD 20814, e-mail doug-wiley@juno.com or dwiley@seas.gwu.edu.