EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Using the Discrete Fourier Transform for Character Motion Blending and Manipulation - a Streamlined Approach

M. R. L. Molnos[1], S. D. Laycock[1], A. M. Day[1]

[1]School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK

Abstract

Motion capture data allows natural-looking motion to be bestowed upon simulated characters. Research has sought ways of extending the range of motions it can reproduce. One such method involves blending between captured sequences in the frequency domain. This paper streamlines the approach taken by similar previous work. Higher efficiency is obtained both by shifting computations from runtime to pre-processing and by using a simpler technique, which is also more flexible allowing the method to be used for a greater range of motions. Furthermore, the already-known use of a triangular network defining a continuous blending space is instead presented as an adjustable interface element which is both intuitive and more flexible than applied to earlier work.
As before input data may be sparse yet still allows the creation of a continuous spectrum of subtly varying motions, enabling characters to integrate well in their environment. Weighting calculation, blending and Fourier synthesis of realistic-looking motion using five harmonics requires 0.39 µs per degree of freedom for each frame in the created sequence - a one-off cost incurred only when blending ratios change. This figure can be improved further using the proposed level-of-detail adjustments, which, combined with its small memory footprint, makes the method particularly suitable for the simulation of crowds.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Individual characters in a virtual environment should behave differently, walking at different speeds, with different styles. Moreover, precise control is required in the animations to enable characters to interact convincingly with each other and their surroundings, enacting, eg, the most subtle of turns or gradual changes in speed and associated gait geometry.

Achieving this by storing a motion database comprising many thousands of locomotion styles and actions is not practical, so this paper instead considers the *synthesis* of new motions from a smaller amount of input data (Figure 1). It provides an alternative to previous work on the discrete Fourier transform (DFT) and Fourier synthesis to generate such an uninterrupted spectrum of motions, yet presents a simpler, more efficient and more flexible approach. The streamlining measures are put firmly in context, by including an implementation-level description for the creation of a Fourier-based blending system which incorporates them.

2. Previous Work

The literature divides motion synthesis from motion capture databases into two categories: pose rearrangement and motion interpolation. This work focusses on the latter. Previous approaches investigated techniques employing scattered data interpolation, [WH97, RCB98, PSS02]. Rose et al, [RCB98], pioneered the use of radial basis functions and Wiley and Hahn, [WH97], employed tri-linear interpolation. However, these approaches require a dense sampling of motion clips, leading to high memory requirements for varied animations. Recently, Lau et al [LBJK09] generated a continuous range of varied output motions from a sparse set of similar input motions by learning a dynamic Bayesian network model. This created subtle variations in walk cycles but required a set of similar input motions.

Glardon et al, [GBT04a, GBT04b], investigated the use of principal component analysis (PCA) to reduce the data to a set of coefficients; a character's speed could be interpolated

© The Eurographics Association 2010.

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/207-214

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

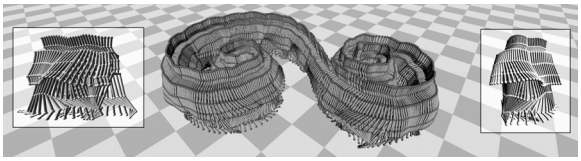

Figure 1: *A continuous range of turns from sharp left, to less sharp, straight, right, then sharp right, created in real time from two short pre-processed input sequences (inset).*

or extrapolated from the input data based upon the character's height. In order for an avatar to convincingly navigate obstacles it is important to be able to control the degree of turn and the style of their movement. Troje, [Tro02], considered the latter using PCA in a similar vein to Glardon et al, but adjusting the style was not independent of the speed, removing significant control from the animator.

Fitting BSplines to weighted combinations of multiple motion clips, [AW01], facilitates the creation of new motion from a sparse set of clips, but also has the potential to remove subtleties from recorded motion. These comprise unwanted noise potentially created during acquisition, but also desired high frequency content indicating the style of the motion.

Treating the motion sequences as a collection of one-dimensional discrete signals, where the joint angles vary over time, permits new motion to be synthesised using signal processing techniques, [UAT95, BW95, PL06]. Bruderlin and Williams, [BW95], present several techniques, of which the most relevant to this paper is multitarget interpolation where angle sequences for the degrees of freedom (DOF) of two or more skeletons are converted to the frequency domain using a method akin to multiresolution image filtering, yielding values in discrete frequency bands which can be adjusted and/or blended to create a new motion when converted back to the time domain. Unuma et al, [UAT95], used Fourier expansion for similar work in the frequency domain, and famously extracted qualities such as tiredness and briskness from a pair of motions and superimposed them onto another. Pettre and Laumond, [PL06], used the discrete Fourier transform (DFT) to obtain Fourier series expansion coefficients prior to blending between sets of three cyclic motions. Input motions are represented as points in a two-dimensional space, with coordinates giving the linear and angular velocities of the root node. The points are joined by a Delaunay triangulation, whose coverage indicates all the velocities available via motion interpolation, providing useful information for the motion-planning context their work focussed on.

This paper uses Fourier expansion as did Pettre and Laumond, [PL06], but expressed in a more compact form leading to a lower expense at runtime. They also required input motions to be resampled so that all comprise the same number of frames, effectively giving them different timesteps, which in turn generates the need for a procedure they term "extracting postures" when adding frames to the synthesised motion - another runtime cost avoided by our method.

Pettre and Laumond, [PL06], employed automatic positioning of triangle apexes, but this imposes significant limitations. Being based on root velocities, it cannot be used for motions where these velocities are not significant, such as standing near-motionless, gently dancing on the spot or standing around while fidgeting. This limitation is avoided with manual apex placement as proposed in this paper, which also allows triangles to be stretched or squashed relative to each other allowing an adjustment of the blending point sensitivity. Furthermore, with manual positioning the triangle network can be displayed as an interface element, proportioned for best user-friendliness, which is both more intuitive and easier to control than the high-level editing tools suggested in [BW95].

It is [PL06] which our work is most similar to and is primarily compared with, showing advantages in certain contexts such as interactive simulations and games.

A high-level view is now given, of the motion synthesis system to which our streamlining measures -described *elsewhere* in this paper- are applied. Some similarity with previous work is evident at this level - the view serving only to provide a context for streamlining. Input motion capture data is pre-processed as described in Section 4. It is first converted to smooth periodic motion sequences and subsequently represented in the frequency domain via the DFT. A continuous space is defined by considering input motion data to be positioned at the vertices of a triangle network. At runtime the location of a blending point in this space specifies an interpolation to be performed on the frequency domain data. Synthesized motion is obtained by then returning the blended data to the time domain using Fourier synthesis. The runtime procedures are described in Section 5.

3. Fourier Series Representation and Phase Angle Blending

Several parts of this paper make reference to our chosen representation for Fourier series expansions, and the associated issue of blending phase angles. A clarification is thus due.

The Fourier series expansion used in [PL06] is

$$m(t) = \frac{\alpha_0}{2} + \sum_{k=1}^{N} \alpha_k cos(\frac{k\pi t}{T}) + \beta_k sin(\frac{k\pi t}{T}) \qquad (1)$$

where α_0, α_k and β_k are magnitude coefficients obtained via the DFT. The formula is shown only to illustrate how it involves twice as many trigonometric functions as that used with our representation (equation 8). The other terms are not relevant to this paper and hence not further explained. By comparison, Equation 8, used in our work, is just as able to synthesize waveforms despite having only half as many

© The Eurographics Association 2010.

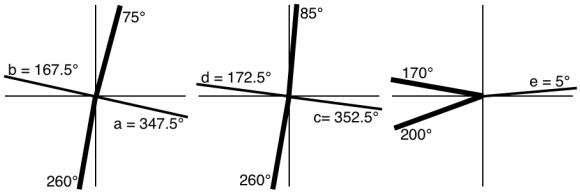

Figure 2: *Potential phase angles, a, b, c and d, from blending those shown in bold. Angle e -see text- is easily refuted.*

trigonometric functions. With this equation the coefficients from the DFT are given by m_0, m_k and p_k, representing both magnitudes and phase angles, which in turn creates the need to blend phase angles. A comprehensive description of the equation and its symbols – is given later in Section 5.2.

Blending is explained in Section 5.1 but in brief it consists of taking a weighted average of the coefficients from each motion in the blend. While blending α_0, α_k and β_k, or m_0 and m_k in this manner is problem-free, this is not the case for the phase angles, p_k.

A phase angle represents a time-shift in one of the sinusoidal components (harmonics) inherent in a single DOF's motion. While naively blending them may succeed in some cases, there are others in which it generates uncoordinated-looking motion. Figure 2, left, gives a simple case where two phase angles, shown by thick lines, are blended using a 50/50 weighting. Two possible blended angles are suggested, shown by thinner lines. Angle *a* may seem like the logical choice as it lies closer to the original angles being blended between. However, the middle diagram shows this approach to fail. Here, only one of the original angles has changed and did so by only 10°, yet the blended phase angle closest to the original angles is now *d*; it thus has jumped by around 180°. Clearly, a small change in the motions being blended should not generate a large jump in the nature of the output motion. The alternative would be to select blended angle *c*, but for consistency this approach should also be used in the right-hand diagram. This time, however, it is the alternative approach which is seen to fail. Phase angles change the style of the motion, so blending between two angles close to 180° should not give *e*, a very different angle close to 0° degrees. The situation becomes still more complex when blending between three motions with variable weightings.

Our solution is to blend $sin(p_k)$ from each motion, and do the same with $cos(p_k)$, the blended angle being given by

$$p_{k,blended} = arctan2(sin(p_k)_{blended}, cos(p_k)_{blended}) \quad (2)$$

where *arctan2* is the two-argument version of the arctangent function. The cost of *arctan2* can be reduced by replacing it with an algorithm based on the following approximation of the standard $arctan(x)$ function, valid for positive x.

$$arctan(x) \simeq p + \frac{q}{x+r} \quad (3)$$

where p, q and r are 1.597, 1.992 and 1.237 respectively for $0 \leq x \leq 0.5$, and 1.583, 1.259 and 0.609 for $x > 0.5$. Any angle calculated with the *arctan2* algorithm has an error below 1°. While not required, higher accuracy is easily achievable by subdividing the domain into further bands.

The above provides an unambiguous way of calculating blended phase angles. It is equivalent to using Equation 1 for blending and subsequently deriving the blended angles from it, but without the cost. The efficiency lies firstly in that the sines and cosines of phase angles are calculated in pre-processing. Secondly, the calculated *arctan2* approximation involves minimal cost. This is mainly because it remains valid for the *entire* output sequence, while making redundant half the trigonometric functions used in Equation 1, which need to be calculated on a *per-frame* basis. This advantage remains if lookup table are used instead of trigonemtric functions. Section 7 (Results) quantifies the associated savings.

4. Pre-processing Input Motions

The steps discussed in this section prepare the input data for blending and only need be performed once for any number of subsequent animation sessions.

4.1. Selection

As blending is performed *between* input motion cycles, the number required is relatively small -and can be minimal- making it a trivial task to manually select them by browsing motion capture files. The main requirement is that they contain a complete cycle of data with start and end frames depicting similar poses. Automating the process by using a distance function as in [KGP02] to select cycles with well-matching endpoints is possible but not seen as an advantage as the aim is not to acquire a large database of motions but instead to select a relatively small number of the most visually appealing ones - a task best left to human judgement.

A principal difference here compared to [PL06] is not only that their work was limited to walking motions, but their requirement that all samples be in phase, eg that they all commence with a left foot-strike. Motion synchronization (Section 4.6) makes this unnecessary and instead allows a post-selection adjustment to synchronize motions, thereby allowing the input motion cycles to be selected solely on the basis of quality without the restriction of enforcing a particular start frame - especially important when motion capture data is sparse. Furthermore, detailed examination of the input data is not required at this or any later stage, completely obviating the difficulty mentioned by Unuma, [UAT95], in estimating the period from measured joint angle data.

© The Eurographics Association 2010.

4.2. Root Rotation Angle Re-sequencing

The position of each node in the skeleton relative to its parent is given by the concatenation of a translation (fixed bone length) and three rotations (of the parent node), $R_a R_b R_c T$.

The position of the end node of a three-bone branch is

$$\begin{aligned} P_{end\ node} = & R_{root\ a}\, R_{root\ b}\, R_{root\ c}\, T_{root} \\ & + R_{mid\ a}\, R_{mid\ b}\, R_{mid\ c}\, T_{mid} \\ & + R_{end\ a}\, R_{end\ b}\, R_{end\ c}\, T_{end} \end{aligned} \quad (4)$$

with the chain of transformations being applied from right to left and ending with the three root rotations.

At runtime the pre-calculated angle increments discussed in Section 4.4 are blended and accumulated frame by frame to create a continuous stream of *new* y-axis rotations for the root node, allowing any turning content in the motion to be endlessly enacted and turns to be built up. This occurs *instead of* blending and applying y-axis values taken straight from the motion capture file, and gives the skeleton its required world-coordinate orientation, consistent with the path so far followed, with no need for any additional transformation to correctly orient the character. Runtime cost reduction is significant, especially in comparison with [KG03] where frames have to be positioned and oriented prior to blending.

The above requires the final (leftmost) matrix rotation in Equation 4 to be the root's y-axis rotation. This ensures the accumulated y-rotations merely rotate the skeleton about the vertical axis. The root angles are thus pre-processed to use YXZ ordering while representing the same 3D rotation.

4.3. Cyclification

Cyclification, as, eg, in [AMH03], modifies motion cycles so their boundaries become compatible, allowing motion concatenation analogous to the tiling of graphic images.

In our approach, having selected and resequenced an input cycle, the angular motion data for the first and last frames of each DOF are offset by equal and opposite amounts so they acquire the same values. Intermediate frames are adjusted accordingly by linear interpolation. The duplicate final frame (describing a pose identical to the first) is then discarded. This simple procedure creates an input motion cycle which can be repeated without any perceived discontinuity.

Cyclification ignores root y-rotations, as making them match at cycle endpoints would prevent characters from accumulating rotations during successive cycles and walking in a circle.

4.4. Root Y-axis Angle Increments

Any discontinuity between the endpoints of root y-rotation data is unaffected by cyclification (lower plot in Figure 3). Such data, taken from three input motions, is to be combined in the frequency domain and rebuilt into a single hybrid angle sequence using Fourier synthesis. However, in order to reduce cost the number of harmonics used during synthesis is restricted, which, if left unaddressed, would result in distortions appearing in the regions of any discontinuity in the input motion. Furthermore, the process of blending and concatenating motion is far simpler with continuous data. The discontinuity is therefore removed by replacing the y-axis data for the root with angle increments as shown in Figure 3.

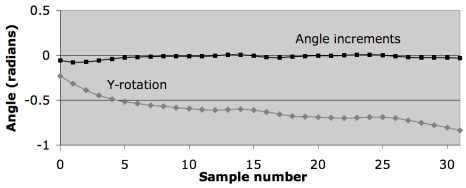

Figure 3: *Example of the elimination of the sharp discontinuity in the root node y-rotations at the ends of the motion sequence by converting them to angle increments.*

The increments are given by

$$i_n = \begin{cases} a_n - a_{n-1} & n \geq 1 \\ (a_{N-1} - a_{N-2} + a_1 - a_0)/2 & n = 0 \end{cases} \quad (5)$$

where a is the angle value and N the number of samples in the motion cycle.

During animation it is the angle increments, instead of the angles themselves, which are blended, and the new increments thus obtained are accumulated frame by frame creating the y-axis orientation used in the synthesised motion.

4.5. Limp Correction

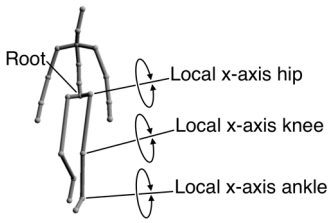

Figure 4: *Limp-removal by returning the root node to its original elevation, if affected by cyclification.*

While cyclification always generates motions which can be repeated without discontinuity, it can, on occasion, generate a limp. This is removed by storing the vertical position of the root node (pelvis) for each frame prior to cyclification, and later adjusting the x-axis angles in the ankles, knees, and hips (Figure 4) to re-establish the same elevations of the pelvis as were present before.

© The Eurographics Association 2010.

4.6. Motion Synchronization

In order for three motions to blend, they must first be aligned with each other. The process is simple, however, entailing none of the complexity of the dynamic time warping used in [BW95]. Pettre and Laumond, [PL06], ensured alignment by requiring that input sequences start with the same part of their cycle, which can limit input motion selection as discussed in Section 4.1.

Our method, instead, aligns one sequence to another by shifting the angle values for each DOF of one motion relative to those of the other, with frames pushed out of a sequence being re-inserted at the opposite end. The angle increments calculated for root y-rotations are similarly rotated. Synchronization is performed by blending 50% of one sequence with the same of another, using real-time visual feedback of the resulting motion to quickly determine the best rotation value. It is performed on all DOFs simultaneously and need only be done once for each pair of motions in a blend. To blend between three motions A, B and C, aligning A to B and B to C is sufficient - the process will also have synchronized motion A to motion C. Here, too, a distance measure as in [KGP02] could be used to automate the process, but the benefit would be insignificant in practice, as it takes only seconds to perform the procedure manually and save the results, which remain available for any number of animation sessions to be run in future.

4.7. Discrete Fourier Transform

The DFT is then performed on the smoothed (i.e. post-cyclification) motion data for every angular DOF and in the case of the root y-rotations, on the angle increments. This calculates, for each DOF, two sequences in the frequency-domain (one real and one imaginary) of length N equal to that of the original sequence. From these two, further sequences are calculated, also of length N, which give the amplitudes and phase angles of the input waveform's component sinusoids. Amplitude and phase angle sequences are chosen for reasons of efficiency, and are not the same coefficients calculated by [PL06] (Section 3).

DFT formulae vary slightly and any could have been used - the selected one being given by [Med00]

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\Pi nk}{N}} \tag{6}$$

where X_k is a complex value comprising the k^{th} element of the real and imaginary sequences mentioned above and x_n is the n^{th} value of the time-domain sequence on which the DFT is being performed.

The DFT is performed in pre-processing on relatively short data sequences. A fast Fourier transform could have been used but is not required.

4.8. Phase Angle Blending Precalculation

As explained in Section 3, in order to enhance Fourier synthesis efficiency while maintaining unambiguous blending of phase angles, the sines and cosines of each phase angle are calculated, which is done at this pre-processing stage.

5. Runtime Processing of Output Motion

The following blending and Fourier synthesis steps are required each time the weighting of the input motions needs to change.

5.1. Blending

As blending is done between three motions at a time, it occurs in a triangular blending space. Phase angle blending is covered in Section 3. Magnitude blending calculates a weighted average of the amplitude sequences which the DFT produced for each of the three motions. This is done for each DOF, creating a new hybrid set of Fourier coefficients. The weightings used are given by areal barycentric coordinates, thus

$$w_a + w_b + w_c = 1, \tag{7}$$

where w_a, w_b and w_c are the weightings for the three input motions. Extrapolation is possible by placing the blending point outside the triangular area, in which case one or two of the weightings will be negative.

5.2. Fourier synthesis

Fourier synthesis builds the output waveform from the blended Fourier coefficients. It exists in a number of variants. The version shown by Equation 8 is matched to the DFT formula given above.

$$R_n = \frac{m_0}{N} + \frac{2}{N}\sum_{k=1}^{H} m_k cos(\frac{2\pi kn}{\mathcal{N}} + p_k) \tag{8}$$

where N is the length of the blended Fourier coefficient sequences, $\mathcal{N}$ is the desired output sequence length and H is the highest harmonic used (indexing such that 1 is the fundamental). m, p and R hold the k^{th} blended Fourier magnitude and phase angle coefficients, and the n^{th} time domain output value respectively, where n varies from 0 to $\mathcal{N}-1$.

Unlike Equation 1 used by [PL06] ours includes no time parameter. Instead n specifies the frame in the output motion cycle for which time domain values are being calculated.

The use of $\mathcal{N}$ for the output length in addition to N for the input allows the synthesized motion to have a dynamically varying sequence length. It is chosen to be the weighted average of the lengths of the input motion data. Thus, the more

© The Eurographics Association 2010.

the output sequence is based on a given input motion, the more its length will resemble that motion's. The adjustable output length also provides a convenient way to resample input motions, by applying the DFT followed by synthesis, to create a sequence comprising any desired number of frames.

Input sequence lengths require consideration too, as input motions may comprise different numbers of frames, and consequently sequences of Fourier coefficients of varying lengths. If these lengths are K, N and M, with M the shortest, the generation of a weighted average can only take place for the first M coefficients in each sequence. There are thus at most M hybrid amplitudes and phase angles available for Fourier synthesis, and the highest number of harmonics which can be used to create the time-domain output sequence is $M-1$. In practice this has never been a problem however, as even an input motion with a very short sequence of 10 frames would allow 9 harmonics to be used during synthesis providing a very high quality output motion.

Using all the DFT-generated frequency domain data for a given input motion, and replacing $\frac{2}{N}$ by $\frac{1}{N}$ in Equation 8, would allow an exact copy of that motion to be rebuilt, akin to using the inverse-DFT. However, this would be costly, and Equation 8 is preferred which, unlike the aforementioned alternative, generates output waveforms of the desired amplitude when using small numbers of harmonics.

5.3. Root Translation

Our method makes no use of any fixed root offset or time-varying root translation in the motion capture data. Foot constraints are employed instead.

When a foot which was previously higher than the other becomes the lower one, the position of that foot is stored and known as the 'anchor point'. In subsequent frames the skeleton is translated such that the appropriate foot is placed at the anchor point which propels the skeleton forward while avoiding any footskate of the anchored foot.

Before ground contact some unwanted sliding of the foot may, occasionally, be visible. However, as applied with each of [UAT95], [BW95] and [PL06], we do not include constraint satisfaction and footskate reduction in the scope of our work and consider it the subject of post-processing addressed by other research. The elementary approach described above is merely intended to allow demonstration of the proposed Fourier blending method.

6. Blending Triangles

6.1. Triangle Networks as Interfaces

Individual blending triangles can be joined to create larger blending areas. Figure 5 shows two similar versions of a four-triangle area. The top left-hand network has different diagonals from the top right-hand one, which means (almost)

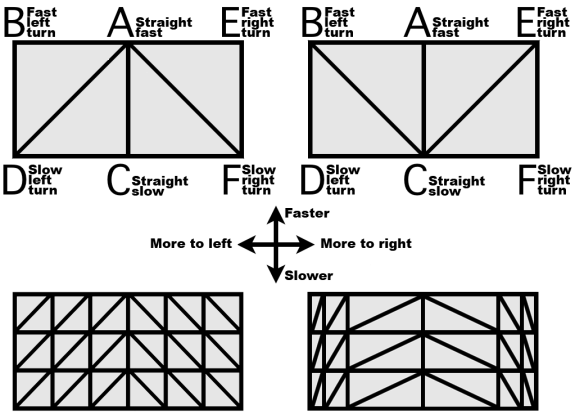

Figure 5: *Diagonal directions, top left and right, have little effect. The denser triangle networks, bottom, give higher quality output and have different blending point sensitivities.*

any point within the blending area will generate a blend from different motions depending on which network is used. In practice, however, little difference is noticed, if any, and moving the blending point, say, vertically from the midpoint of edge DC to that of edge BA creates a slow-paced moderate left turn, accelerating to a faster-paced one, whichever the diagonal direction.

The blending space in the figure spans from slow to fast, and from right to left. If suitable motion capture clips are available, the same area could be covered by a denser network resulting in a higher quality synthesized output (bottom left). However, even a sparse network, as shown top left and right, can produce natural looking motion if the input sequences are of high quality.

Triangle networks also form interface elements, either between the user and an interactive program, or between modules in the program itself. Showing the network and blending point on-screen provides an intuitive way for the user to control a character. Hidden networks are useful too; triangles can be stretched or squashed to modify the sensitivity of a user-controlled input device, or that of an algorithm autonomously driving the skeleton by controlling the blending point. Assuming the same input motions are used as in the bottom left diagram, the network on the right hand side would aid delicate tracking of straight lines and slight curves, while still allowing turns to be as sharp as before.

Networks are not confined to these simple two-dimensional examples. The input motions at triangle apexes can themselves be controlled by blending triangles which have their own blending points. Thus the quality of input motions could -occasionally and hence efficiently- be automatically adjusted, eg, between being tired, hurried or relaxed, in response to the character's current virtual circumstances. In comparison, the superposition of qualities such as tired-

© The Eurographics Association 2010.

ness, as introduced by Unuma et al, [UAT95], would need to be performed continuously to maintain a tired walk, thus involving greater cost.

6.2. Substituting for Missing Input Motion

The small triangle network shown top left in Figure 5 can be viewed in the movie accompanying this paper. The movie furthermore demonstrates the variety of motion that can be synthesised from a very small number of input motions, having used only three short captured sequences (*A* with 30 frames, *B* with 34 and *C* having 42) to create all the motion available in the rectangular blending area. While *E* and *F* were created by reflection, *D* was created using an extension of the above-mentioned superposition technique.

Instead of adding a quality such as briskness to an existing motion, a very different geometry was created. The straight fast walk (*A*) was subtracted from the fast left turn (*B*) resulting in a motion which is not a walkcycle but could be described as "turning-left-ness" and this was then added to the straight slow walk (*C*) to generate a slow left turn (*D*) where previously none existed. Thus not only does Fourier blending function well with little input data, but it is sometimes able to create input sequences where these are not available.

7. Results

Pre-processing of each input motion was performed in seconds with the result being saved in BVH format. Animation was then immediately available upon program launch.

A video can be downloaded from `http://www.crowdsimulationgroup.co.uk/fouBlend.mp4`. It demonstrates how blending is possible even with very few input motions. It also shows a simple triangle network being used as an interface device to assist user-interaction with the skeleton. Furthermore, a section is included showing interpolation of non-walking motions, which could not be blended with the automatically created networks of [PL06]. As applied to [UAT95], [BW95] and [PL06], we note that synthesized motion may require final touching up which we, as they did, consider beyond the scope of our work.

Harmonics	1	2	3	4	5	10
Blend. (μs)	0.14	0.24	0.37	0.49	0.59	1.18
Synth. (μs)	2.68	5.01	7.44	9.84	12.30	24.23

Table 1: *Fourier synthesis and blending times per DOF for a 33-frame output sequence using various numbers of harmonics. Times refer to sequence creation, not merely frames.*

Measured times required for blending and Fourier synthesis using various numbers of harmonics are given in Table 1. They were obtained on an Apple MacBook Pro laptop with Intel Core 2 Duo CPU running at 2.53 GHz with 4GB RAM. The values refer to the creation of an entire 33-sample output sequence for a single DOF, created by blending sequences of length 30, 34 and 42. Blending times shown *include* the associated triangle selection and weighting calculations.

A direct comparison with [PL06] is difficult due to the older hardware they used (in 2006). They reported 0.92*ms* for their equivalent steps to compute a single posture on a 62 DOF skeleton using Fourier coefficients up to the eighth rank. From Table 1, we deduce that a single posture for a 62 DOF skeleton using the same number of Fourier coefficients as did [PL06] would be computed in 38μs. While admittedly a simplistic comparison, the result does appear favourable.

It should be emphasized that blending and synthesis need only be performed when a change in the currently playing motion sequence is desired and that at all other times the cost is nil.

[PL06] state the triangle selection process to be logarithmically dependent on the number of motion captures. However, due to coherence of position of the blending point from frame to frame, this can be reduced to $\mathcal{O}(1)$ by using an appropriate data structure whereby blending triangles store information about their immediate neighbours.

Harmonics	1	2	3	4	5	10
t_{rel} PL06	1.0	1.0	1.0	1.0	1.0	1.0
t_{rel} our work	0.807	0.726	0.718	0.691	0.702	0.689

Table 2: *Synthesis times relative to the method of [PL06].*

The cost of Fourier synthesis can be objectively compared with that of [PL06]. Table 2 gives the times for our synthesis formula *relative* to theirs, obtained by running an efficiently coded version of each on the same hardware. While our compact formula requires the *arctan2* approximation for phase angle blending, this overhead is trivial, as described in Section 3 and shown in Table 1 (whose blending times include more than just phase angle blending). Table 2 shows our formula to be significantly cheaper, and slightly more so for higher numbers of harmonics. A further cost saving of our method beyond that tabulated above, is that synthesis generates a sequence of poses in a simple and direct manner, without the expense of "extracting postures" used by [PL06].

Unuma et al, [UAT95], stated that 3 to 7 harmonics are required for realistic motion which our tests confirm, while [PL06] neglected Fourier coefficients above the eighth rank. Figure 6 illustrates walk cycles using 1, 3 and 25 harmonics. Using a single harmonic looks overly smooth and exhibits a slight bounce due to the exaggerated vertical reach of the step. Merely 3 harmonics are seen to create a walk cycle very similar to the near-perfect motion constructed using 25.

The quality of motion achievable using different harmonics is more objectively evaluated by comparing the node coordinates of two skeletons. Table 3 gives the average error between the node positions of a reference skeleton playing back unaltered motion capture data, and another playing

© The Eurographics Association 2010.

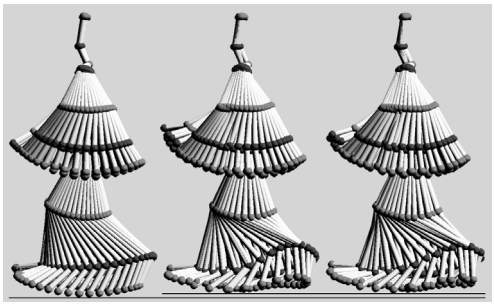

Figure 6: *Walkcycles using (left to right) 1, 3 and 25 harmonics. Horizontal lines show the vertical reach of the step.*

Harmonics	1	2	3	4	5	10
Er_{avg} (cm)	2.22	0.83	0.46	0.38	0.19	0.08

Table 3: *Average node position error during synthesis.*

back a synthesized version of the same motion, using various numbers of harmonics. It is seen that using 5 harmonics the average error is below $2mm$, which represents 0.11% of the skeleton's $1.673m$ height. In the context of a walking human this error is almost imperceptible. Even with three harmonics the average error is below $5mm$. Neither [UAT95], [BW95] nor [PL06] measured the accuracy obtainable using various numbers of harmonics, yet firmly establishing that 3 to 5 harmonics suffice for realistic foreground views avoids unnecessary expense. It also lowers memory requirements, with input motion stored as shorter Fourier coefficient sequences.

8. Conclusion

We have compared our Fourier blending method to previous work, especially that of [PL06], which is the most similar. Our simpler approach is more efficient, as confirmed by measurements and explanations of how runtime expense is reduced. The simplicity aids implementation, and, as demonstrated in the accompanying movie, makes it possible to blend motions for which [PL06] would not be able to create triangle networks. Our networks allow flexibility in apex positioning, with advantages such as adjustable blending point sensitivity and their use as intuitive user-interface devices. The streamlined approach is well suited to real time interactive simulations and games, and while not intended for the motion planning context [PL06] focussed on, it can automatically follow a path as shown in the video.

Level of detail (LOD) adjustments for this, and previous work, include limiting the number of DOFs in distant characters, and reducing the number of harmonics in their animation. Our method, with its inherent resampling ability, makes it easy for distant characters to be animated at a lower frame rate, while still traversing the terrain at the same speed.

Combining such LOD measures with our lightweight Fourier blending is ideal for the simulation of crowds. An avenue for further research would be to map such character animation to the GPU, since both Fourier synthesis and blending are well suited to parallel processing, enabling this method to be further exploited in large crowd simulations.

References

[AMH03] AHMED A., MOKHTARIAN F., HILTON A.: Cyclification of human motion for animation synthesis. In *Short Paper Proceedings of Eurographics 2003* (2003).

[AW01] ASHRAF G., WONG K.: Constrained framespace interpolation. In *Comp. Anim. 2001* (South Korea, 2001), pp. 61–72.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *SIGGRAPH '95* (NY, USA, 1995), ACM, pp. 97–104.

[GBT04a] GLARDON P., BOULIC R., THALMANN D.: A coherent locomotion engine extrapolating beyond experimental data. In *CASA* (2004), pp. 73–84.

[GBT04b] GLARDON P., BOULIC R., THALMANN D.: Pca-based walking engine using motion capture data. In *CGI '04* (USA, 2004), IEEE Comp. Soc., pp. 292–298.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *SCA '03* (Switzerland, 2003), EG Association, pp. 214–224.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Trans. Graph. 21*, 3 (2002), 473–482.

[LBJK09] LAU M., BAR-JOSEPH Z., KUFFNER J.: Modeling spatial and temporal variation in motion data. *ACM Trans. Graph. (SIGGRAPH ASIA 2009) 28*, 5 (2009).

[Med00] MEDDINS R.: *Introduction to Digital Signal Processing*. Newnes, 2000.

[PL06] PETTRE J., LAUMOND J.-P.: A motion capture-based control-space approach for walking mannequins: Research articles. *Comput. Animat. Virtual Worlds 17*, 2 (2006), 109–126.

[PSS02] PARK S., SHIN H. J., SHIN S.: On-line locomotion generation based on motion blending. In *SCA '02* (NY, USA, 2002), ACM, pp. 105–111.

[RCB98] ROSE C., COHEN M., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5 (1998), 32–40.

[Tro02] TROJE N.: Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision 2* (2002), 371–387.

[UAT95] UNUMA M., ANJYO K., TAKEUCHI R.: Fourier principles for emotion-based human figure animation. In *SIGGRAPH '95* (NY, USA, 1995), ACM, pp. 91–96.

[WH97] WILEY D., HAHN J.: Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl. 17*, 6 (1997), 39–45.

© The Eurographics Association 2010.