Dynamic Time Warp Based Framespace Interpolation for Motion Editing

Golam Ashraf Kok Cheong Wong

Center for Graphics and Imaging Technology
Nanyang Technological University, Singapore

Abstract

Motion capture (MOCAP) data clips can be visualized as a sequence of densely spaced curves, defining the joint angles of the articulated figure, over a specified period of time. Current research has focussed on frequency and time domain techniques to edit these curves, preserving the original qualities of the motion yet making it reusable in different spatio-temporal situations. We refine Guo et. al.'s[6] framespace interpolation algorithm which abstracts motion sequences as 1D signals, and interpolates between them to create higher dimension signals. Our method is more suitable for (though not limited to) editing densely spaced MOCAP data, than the existing algorithm. It achieves consistent motion transition through motion-state based dynamic warping of framespaces and automatic transition timing via framespace frequency interpolation.

Key words: motion editing, framespace interpolation, blending, concatenation, motion correspondence.

1 Introduction

With increasing availability of motion capture devices and high fidelity motion requirements in the entertainment industry, realistic animation has become possible without need for dynamic simulation or laboriously crafted keyframed data. However, *motion reuse* is as relevant as captured motion data, since it is not always feasible to retake motions or record transitions between two desired motions. Simple basis motions are created and then interpolated or extrapolated through various techniques, to yield a large variety of motions. Motion editing broadly encompasses *reshaping, blending, concatenation* and *retargetting* of basis motions. In this paper, we deal with consistent blending and concatenation, both of which are referred to as motion transition problems by several authors[3, 12].

Interactivity in motion editing research has been assigned paramount importance, since it is necessary to avoid clogging up the animators' workflow through undesirably long waits between edits. Ease of editing specification is also very important and requires minimizing the number of control parameters. Framespace interpolation is a time domain motion-transition technique which allows such interactivity through minimal user-specification and simple computation needs. Guo and Roberge[6] use parametric framespace interpolation for transitions between human running and walking, where inter-motion correspondences are developed between key states of the lower half of the body. We reformulate their framespace interpolation technique, specifically keeping motion editing of densely spaced signals in mind, while establishing motion correspondences. The rest of this paper has been organized as follows: a survey of motion editing techniques, an analysis of the existing framespace interpolation algorithm, consistent framespace interpolation via dynamic time warping, results and analysis, and a summary of contributions.

2 Survey of Motion Editing Techniques

Most researchers[3, 5, 6, 16] have treated motion data as a 'blackbox' set of 2D C^2 continuous signals, without differentiating the degrees of freedom (DOF) in terms of relevance to the motion, or hierarchy of structure. Bindiganavale and Badler[2] introduced some heuristics based on end-effector acceleration zero-crossings, to isolate significant events and abstract constraints from an agent's action. This approach cuts down unnecessary constraint checks, hence saving valuable computation. We build on this motion abstraction paradigm and achieve significant savings in pre-blend/pre-concatenation motion-warping.

Motion editing in frequency domain has been proposed by a few researchers. Bruderlin's[3] application of Gaussian and Laplacian pyramids to motion data, provides a way to transform motion by adjusting the gains of the different bands, before reconstruction. Unuma et. al.[14] proposed a Fourier transform of discrete motion signals. From a given set of Fourier coefficients for two motion clips, they achieve *interpolation*, *extrapolation* and *transition* between these motions, by linearly varying the interpolant weights of the Fourier coefficients and phase angles. Though these methods drastically reduce the number of control parameters, a domain transform makes

control less intituitive as a priori knowledge is needed about which frequencies contain the essential motion-characteristics.

Several time domain motion editing techniques have evolved. Guo and Roberge[6, 7] proposed a parametric framespace interpolation paradigm for motion blending and concatenation. By employing a user-specified curve to interpolate entire sequences of articulated figures, animator-effort is minimized. Rose et. al.[11] employ radial basis functions to interpolate and extrapolate actions. Bruderlin[3] used displacement mapping, as a means of editing densely spaced motion curves. The method involves specifying a smooth curve, through a few keyframes (*constraints*), and adding it to the original curve. This way, the original motion characteristics are preserved, yet achieving a change in the motion via a low frequency offset curve. Luo[9] and Gleicher[5] use displacement curves as a constraint specification tool. Witkin[16] uses a similar concept in the form of motion warping.

Witkin and Kass[15] introduced motion synthesis as a constraint optimization problem, where given a set of constraints, the problem is to find a valid motion that best satisfies the goal. Cohen[4] proposed a more interactive system, where the solution is guided by the user. Gleicher[5] simplified the formulation for interactive performance, by using *Sequential Quadratic Programming* (SQP) techniques. SQP solvers perform efficiently because they accept quadratic optimization metrics and only linear constraints. Motion retargetting[5, 8] and concatenation[12] can be viewed as spacetime problems, in that given a length of motion, the problem is to find the best motion which satisfies the constraints and maximizes the goal. Unlike pure IK based re-positioning, the spacetime approach ensures that the constraints affect neighboring frames as well.

3 Analysis of Existing Framespace Algorithm

3.1 Interpolation Algorithm

In simple terms, framespace interpolation is a way of specifying postural blends via an input curve (*interpolant*), drawn within *frames* which enclose a rectangular area or a cubic volume. Basis motions (*primitives*) which are to be interpolated, are each represented by one such frame. Mapping the entire clip of an animated figure with normalized times, on to this frame yields a *1D framespace*. In Eqn.(1) m dimensional point, Q, represents the body posture at a given time instant. Eqn.(2) represents the motion curve of DOF$_i$ as an interpolation function ($\phi_i(s)$) of a time-sequence of a_i points. $F(s)$ abstracts the m motion curves as a 1D framespace, parameterized by arc length s.

$$Q = \{a_1, a_2, \ldots, a_m\}^T \quad (1)$$

$$F(s) = \{\phi_1(s), \phi_2(s), \ldots, \phi_m(s)\}^T, \ 0 \le s \le 1 \quad (2)$$

Combining 2^{n-1} such framespaces, interpolation can be done in *n* dimensional framespace. As a practical interpolation tool, 2D and 3D framespaces are adequate, since user-interaction in higher dimensional interpolation of 1D framespaces has no direct visual mapping. Eqns.(3) & (4) express 2D and 3D linear parametric interpolation, where x (time), y and z (weights) are cartesian coordinates of points, $P(x, y, z)$, of the interpolant. $F_{1,\ldots,4}$ are parametric 1D framespaces, which represent the basis motions.

$$F'(x, y) = (1 - a)\ F_1(x) + a\ F_2(x) \quad (3)$$

$$F'(x, y, z) = (1 - a)(1 - b)\ F_1(x) + a(1 - b)\ F_2(x) + (1 - a)b\ F_3(x) + ab\ F_4(x) \quad (4)$$

where $a = \dfrac{y - y_1}{y_2 - y_1}, \quad b = \dfrac{z - z_1}{z_2 - z_1}$

y_1, y_2, z_1, z_2 are physical bounds of the interface.

Bruderlin[3] points out that arbitrary interpolation between un-correlated motions could give rise to severe inconsistencies in the result. Such correlation has not been established between the primitives in [7]. Guo and Roberge address this limitation, by implementing *consistent 3D framespace interpolation*[6]. Two styles of human walking and running are chosen as primitives and key locomotion postures are chosen as correspondence points (*states*). The 3D interpolation space (see Fig.1) is divided into sub-volumes by *event surfaces* (S_n) constructed between corresponding states from the four primitives ($\{s_n^1 s_n^2 s_n^3 s_n^4\}$ in Eqn.(5)) . These surfaces are non-planar in the general case. Every point on the interpolant, $P(x, y, z)$, is first projected (along the time axis) on all event surfaces to determine whether it lies on a surface. If true, then the four states defining the surface are used for interpolation (as in Eqn.(5)). Otherwise, if P belongs to a sub-volume, v_i then its eight bounding states are used (as in Eqn.(6)).

$$F'(x, y, z) = \sum_{i=1}^{4} a_i F_i(s_n^i) \quad (5)$$

$$F'(x, y, z) = \sum_{i=1}^{4} \left(a_i F_i(s_n^i) + a_{i+4} F_i(s_{n+1}^i)\right) \quad (6)$$

where

s_n^i	:	Correspondence state n of framespace F_i
		$\{s_n^1 s_n^2 s_n^3 s_n^4\}$ define event surface S_n in Fig.1
$a_{1,\ldots,8}$	:	coefficient functions $f\{x, y, z, \sigma_n(z), \sigma_{n+1}(z)\}$
$\sigma_n(z)$	:	$P(x, y, z)$ projected along x axis, on surface S_n

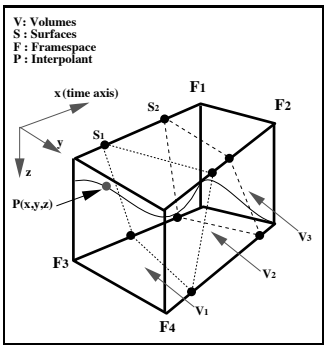


Figure 1: Guo and Roberge's[6] consistent interpolation.

3.2 Drawbacks of Existing Technique

An important difference between Eqns.(3) & (4) and Eqns.(5) & (6) can be observed. While the domain of the former set of equations is the entire framespace $(F_i(x), 0.0 \leq x \leq 1.0)$, the domain of the latter set comprises only of the keyframes $(F_i(x), x \in \{s_0^i, \ldots, s_m^i\})$. Though the coefficients a_i are functions of the entire framespace, it is important to note that these coefficients are merely used to weigh the values of the bounding keyframes, and hence, joint angle values of the entire framespace are *not* used. Thus the algorithm does not exploit the high frequency information in the framespaces constructed from MOCAP data, since only key correspondence points are interpolated. This formulation seems more suited to sparsely placed keyframes rather than dense motion curves, where the premise of keyframes being equivalent to *key events* does not hold.

Secondly, a generic framework has not been proposed for the automatic generation of timing for the resulting motion. The default time mapping proposed is specifically related to human locomotion, and does not cater to general motion transition.

Lastly, the arc length parameterization of the m-D interpolation function in Eqn. (2) overlooks a basic drawback. Referencing a point at arc distance s along $\phi_i(s)$ curves $(1 \leq i \leq m)$, would yield points at different time instances t_i, and not t as we normally expect in parametric keyframe animation. This is because the arc-lengths for the m ϕ_i curves at a given time, will be different due to their characteristic shapes. Besides being an unconventional parameter in keyframe animation, this undesirable feature makes it tedious to validate postures on the fly.

4 Consistent Framespace Interpolation via Dynamic Time Warping

We propose a more efficient algorithm which improves on the above-mentioned drawbacks. A modified framespace interpolation technique is described in this section, which exploits the entire framespace information and provides a generic method of default transition-timing generation. Arc length parameterization is not used for reasons mentioned in the last section.

4.1 Algorithm Overview

The proposed algorithm hinges on ideas drawn from Bruderlin's[3] *dynamic time warping* and correspondence based on high level events[2, 6, 10]. Instead of projecting $P(x, y, z)$ on event surfaces (Sec.3.1), the irregular bounding volumes (Fig.1) are regularized by weighted scaling of inter-state time gaps in the basis motions. The source frames are then extracted from the regularized framespaces and blended using weights drawn from $P(x, y, z)$. Further, these weights are used to interpolate the frequencies of the primitives, to yield smooth timing transition in the resulting curves. Weight coefficients a and b in all equations in these sections, are the same as in Eqns.(3) & (4). Assigning $y_1 = z_1 = 0$ and $y_2 = z_2 = 1$, saves on unnecessary division and yields $a = y$ and $b = z$. The following sections will explain the concepts in greater detail. Fig.2 summarizes the algorithm.

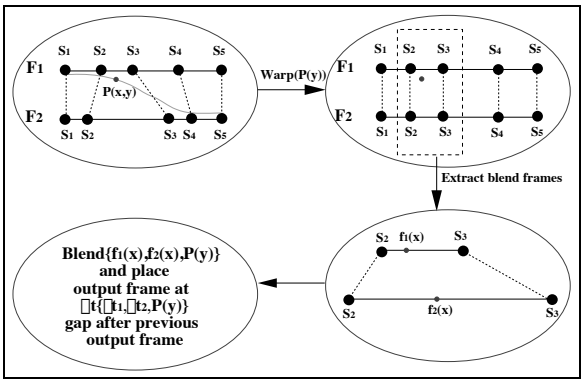


Figure 2: Proposed Consistent Interpolation Framework

4.2 Dynamic Time Warp Preliminaries

Bruderlin[3] uses dynamic time warping to align signals before performing blending. He uses Sederberg's[13] physically based signal correspondence techniques to establish this time warp. The algorithm solves the correspondence problem by exhaustive graph traversal, where the vertices represent all possible combinations of interpoint correspondence between two signals. This is an expensive O(mn) operation (if the two signals have m and n samples each). We simplify the correspondence problem by identifying samples where *key events* occur, and then use these as reference points to apply time warps to entire sections of the signal, instead of warping the samples individually, as in Sederberg's method. Such a motion-abstraction assisted warp method is much more economical than Sederberg's low level warping algorithm.

4.3 Pre-blend Warping of Primitives

In our problem formulation, motion states are pre-classified. The system then automatically identifies corresponding states, based on a best match between event labels of analyzed primitives. What needs to be resolved now is, by how much should each of the primitives be warped to enable consistent blending. Warping the rest of the primitives to one primitive leads to inconsistent results, since the net frequencies of the other primitives are changed completely. Consider a simple 2D interpolation case of transition from *run* to *walk*. If the walk action is warped to match the run, we have a much faster walk parametric space, with the run frequency untouched. When the interpolant meets the *walk frame* (run has no effect), the result will be a funny fast shuffle, which is neither a walk nor a run.

To solve this problem, we propose a regularization warp function for aligning all primitives. The function is driven by $P(x, y, z)$, where y & z provide weights and x makes the function time-variant (dynamic). Let us explain this concept in the 2D interpolation case first. A single cycle of two hypothetical 1D framespaces can be represented as shown in Fig.3. Δt refers to the elapsed time between successive motion states. Then the problem of changing the Δt between successive states so that all the framespaces are *consistently* affected, can be solved by a weighted interpolation of these curves. This can be easily generalized to the 3D interpolation case. Eqns.(7) & (8) define dynamic time warp functions for 2D and 3D framespaces respectively. Δt_n^m represents time elapsed between states n and $n-1$, for framespace F_m, and y & z are interpolant weights. $dT'(n, y)$ and $dT'(n, y, z)$ represent the new Δt values for 2D and 3D interpolation respectively, which will be applied to all the primitives, resulting in regular subspaces.

$$dT'(n,y) = (1-a)\Delta t_n^1 + a\Delta t_n^2 \quad (7)$$

$$dT'(n,y,z) = (1-a)(1-b)\Delta t_n^1 + a(1-b)\Delta t_n^2 + (1-a)b\Delta t_n^3 + ab\Delta t_n^4 \quad (8)$$

The problem of dynamic time warping has thus been elegantly solved by interpolating the Δt curves based on the weights derived from $P(x, y, z)$. In other words, the warp operation tantamounts to stretching and squeezing inter-state times of the framespaces to achieve regular volumes, where the magnitude of transformation depends on $P(y, z)$, at time instance x. So in the special case of $P(x, y, z)$ lying on a primitive frame, $F_n(x)$, the corresponding states in the other primitives ($F_i(x), i \neq n$) are warped to follow the inter-state gaps of $F_n(x)$.

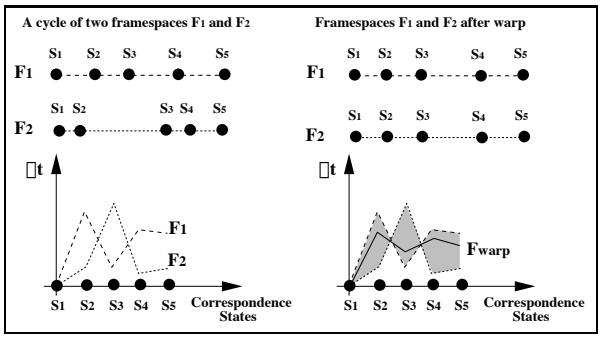


Figure 3: Regularization Warp of Primitives

4.4 Linear Parametric Blending

The previous section describes how pre-blend time warping is done to align the primitives. However, it must be noted that we *do not* actually warp the framespaces for each point $P(x, y, z)$ on the interpolant, since it would incur a lot of meaningless computation. The reason for warping the framespaces is to yield a regular subspace, and extract the *bounding states* for the current interpolant sample $P(x, y, z)$. The bounding state check is done in the warp compensated regular subspaces.

Once the bounding states have been identified, the next step is to extract the reference frames from the primitives. This can be achieved through linear coordinate geometry, as expressed by Eqns.(9) & (10).

$$T'(n) = x - s^*_{n-1} \quad (9)$$

$$f_i(x) = F_i\left(s^i_{n-1} + \frac{\Delta t^i_n}{dT'(n)} T'(n)\right) \quad (10)$$

where

x	:	Current cycle time
$T'(n)$	:	Relative time in warped subspace w.r.t. state n-1
$dT'(n)$	:	Differential time in warped subspace w.r.t. state n-1, from Eqns.(7) & (8)
s^*_n	:	Time of state n after warp compensation
s^i_n	:	Time of state n of framespace F_i, i = 1,2,3,4

$f_i(x)$ are the reference frames from primitives F_i, which are used for the linear weighted blending operation shown in Eqns.(11) & (12). $F'(x, y)$ and $F'(x, y, z)$ are the results of 2D and 3D framespace interpolation.

$$F'(x,y) = (1-a)\, f_1(x) + a\, f_2(x) \quad (11)$$

$$F'(x,y,z) = (1-a)(1-b)\, f_1(x) + a(1-b)\, f_2(x) + (1-a)b\, f_3(x) + ab\, f_4(x) \quad (12)$$

where $f_i(x)$ from Eqn.(10)

4.5 Frequency Transition

Having calculated the value of the result frame, it is equally important to place it at an appropriate distance from the last calculated frame. This is how default frequency calculation of the resulting motion can be described at a micro level. Though the framespace time axis is normalized, the time periods of the original cyclic motions are used to calculate the inter-sample durations of F_i, as shown in Eqn.(14). These are linearly blended with weights derived from $P(x, y, z)$ in Eqn.(13), to yield the gap between the current and previous output samples.

$$\begin{aligned} \delta'(x,y,z) &= (1-a)(1-b)\delta_1 + a(1-b)\delta_2 \\ &\quad +(1-a)b\delta_3 + ab\delta_4 \qquad (13) \\ \delta_i &= \frac{\Pi_i}{n}, \qquad i = 1,\ldots,4 \qquad (14) \end{aligned}$$

where

Π_i : Time period of motion clip i

n : Resolution of interpolation curve samples

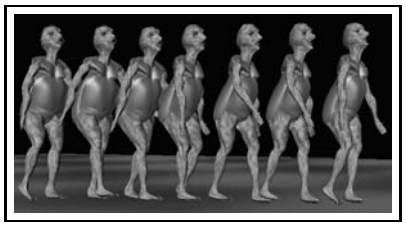
Thus a transition from a low frequency motion to a high frequency motion is accompanied with equivalent changes in joint angular values as well as motion frequency. This simple yet generic method does away with the need for a procedural frequency transition formulation or user defined time warps for natural motion transitions.

4.6 Special Treatment of Pelvis Translatory DOFs

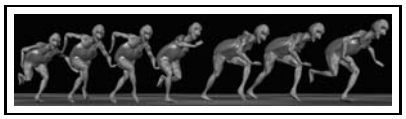
The range of angular DOFs is finite, governed by joint limits. However, the translatory DOFs of the root (pelvis) are not bound by any limits. So interpolating *changes*[12] in translatory motion has two benefits: it does not require the basis motions to start at the same global position, and it eases interpolation across many cycles using only one cycle of each primitive. We build instantaneous velocity curves for each participating pelvis (translatory DOFs) by differencing neighboring samples. The interpolation procedure is the same as the rest of the DOFs, except that the source frames are drawn from the velocity curves, and instantaneous velocities have to be scaled down by δ_i in Eqn.(14) before blending. The interpolated value is then added to the previously calculated sample of the result curve to yield the spatial value.

5 Experimental Results

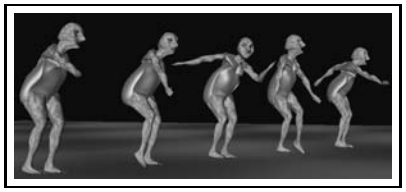
The algorithm has been implemented in Alias/Wavefront Maya^{TM} API and run on Intergraph TDZ2000 (450MHz CPU with 128MB RAM). Though the basis motions are cyclic, we use a single cycle extracted from the MOCAP data. Desired number of cycles are generated through repeated self-concatenation of the primary cycle after smoothing the start and end regions. Our results are obtained from four basis motions, namely, *walking, running, a common dance step (dance1)* and *the "Egyptian rap" (dance2)*. Fig.4 shows the key postures of these basis motions, the time progression being from left to right. We have specifically chosen the dance motions to show that our algorithm can handle motion transitions between widely varying classes of full bodied motions.



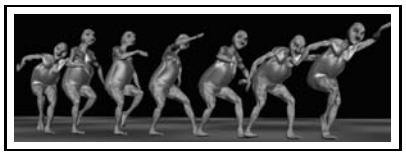
(a) Walk cycle



(b) Run cycle



(c) Dance1 (common dance) cycle



(d) Dance2 (Egyptian rap) cycle

Figure 4: Key postures of basis motions

We present here both 2D and 3D interpolation test cases. Table 1 presents a state classification of the basis motions, based on which correspondences are automatically established as shown in Table 3. Table 2 explains the meanings of the abbreviations used in labelling states. Note that a state can contain more than one characteristic event and is taken care of during state matching. From Table 1, it is evident that the upper and lower body coordination of *dance2* is 180^o out of phase with the rest of the motions. A possible solution to such state-sequence clashes is to perform decoupled blending in the two halves of the body, and is a subject of ongoing research. We have presented some preliminary results of such decoupled blending in sequences involving *dance2*.

Fig. 5 shows examples of concatenation and blend

Run	Walk	Dance1	Dance2
Cycle Time 0.77s	*Cycle Time* 1.24s	*Cycle Time* 2.43s	*Cycle Time* 3.07s
Lowerbody	*Lowerbody*	*Lowerbody*	*Lowerbody*
L_et (0.0)	L_et (0.0)	R.HI_flex (0.0)	L_mt (0.0)
R.HI_flex (0.22)	R.HI_flex (0.22)	R.H_gnd (0.14)	L_et (0.36)
R_mt (0.26)	R_mt (0.26)	L.HI_flex (0.41)	R.HI_flex (0.41)
R_et (0.48)	R_et (0.49)	L.H_gnd (0.69)	R_mt (0.49)
L.HI_flex (0.78)	L.HI_flex (0.88)	R.HI_flex (1.0)	R_et (0.76)
L_mt (0.83)	L_mt (0.9)		L.HI_flex (0.84)
L_et (1.0)	L_et (1.0)		L_mt (1.0)
Upperbody	*Upperbody*	*Upperbody*	*Upperbody*
L.S_ext (0.0) R.S_flex	L.S_ext (0.0) R.S_flex	R.S_ext (0.0) L.S_flex	R.S_ext (0.0) L.S_flex
L.S_msw (0.17) R.S_msw	L.S_msw (0.24) R.S_msw	R.S_msw (0.16) L.S_msw	R.S_msw (0.38) L.S_msw
L.S_flex (0.44) R.S_ext	L.S_flex (0.44) R.S_ext	R.S_flex (0.41) L.S_ext	R.S_flex (0.54) L.S_ext
L.S_msw (0.57) R.S_msw	L.S_msw (0.73) R.S_msw	R.S_msw (0.62) L.S_msw	R.S_msw (0.82) L.S_msw
L.S_ext (1.0) R.S_flex	L.S_ext (1.0) R.S_flex	R.S_ext (1.0) L.S_flex	R.S_ext (1.0) L.S_flex

Table 1: Meta data of basis motions

Code	Description
{LR}.[j]_e	{left or right joint}.[optional joint name]_event name e.g. L.S_flex represents maximum flexion of the left shoulder joint
flex	Local maxima in joint flexion
ext	Local minima in joint flexion
mt	Mid transfer (refer to [6])
et	End transfer (refer to [6])
msw	Mid swing or half way between flexion and extension
gnd	End effector touches ground plane after being in air
joints	S: Shoulder; HI: Hip; H: Heel

Table 2: Index to state label abbreviations

Run, Walk	Dance1, Dance2	Run, Walk, Dance1, Dance2
Lowerbody	*Lowerbody*	*Lowerbody*
{L_et} {0.0, 0.0}	{R.HI_flex} {0.0, 0.41}	{L_et} {0.0, 0.0, 0.69, 0.36}
{R.HI_flex} {0.22, 0.22}	{R.H_gnd} {0.14, 0.76}	{R.HI_flex} {0.22, 0.22, 1.0, 0.41}
{R_mt} {0.26, 0.26}	{L.HI_flex} {0.41, 0.84}	{R_mt} {0.26, 0.26, 0.14, 0.49}
{R_et} {0.48, 0.49}	{L.H_gnd} {0.69, 1.0}	{L.HI_flex} {0.78, 0.88, 0.41, 0.84}
{L.HI_flex} {0.78, 0.88}	{R.HI_flex} {1.0, 0.41}	{L_et} {1.0, 1.0, 0.69, 0.36}
{L_mt} {0.83,0.9}		
{L_et} {1.0,1.0}		
Upperbody	*Upperbody*	*Upperbody*
{L.S_ext, R.S_flex} {0.0, 0.0}	{R.S_ext, L.S_flex} {0.0, 0.0}	{L.S_ext, R.S_flex} {0.0, 0.0, 0.41, 0.54}
{R.S_msw, L.S_msw} {0.17, 0.24}	{R.S_msw, L.S_msw} {0.16, 0.38}	{R.S_msw, L.S_msw} {0.17, 0.24, 0.62, 0.82}
{R.S_ext, L.S_flex} {0.44, 0.44}	{L.S_ext, R.S_flex} {0.41, 0.54}	{R.S_ext, L.S_flex} {0.44, 0.44, 1.0, 1.0}
{R.S_msw, L.S_msw} {0.57, 0.73}	{R.S_msw, L.S_msw} {0.62, 0.82}	{R.S_msw, L.S_msw} {0.57, 0.73, 0.16, 0.38}
{L.S_ext, R.S_flex} {1.0, 1.0}	{R.S_ext, L.S_flex} {1.0, 1.0}	{L.S_ext, R.S_flex} {1.0, 1.0, 0.41, 0.54}

Table 3: Correspondence results yielded by system

Operation(Participants)	Execution Time (secs)	Result Duration (secs)
ζ(Run,Walk)	5.06	12.33
β(Run,Walk)	4.97	11.53
ζ(Dance1,Dance2)	5.22	32.67
β(Dance1,Dance2)	5.08	33.43
ζ(Run,Dance2)	5.42	22.83
β(Run,Dance2)	4.87	25.86
ζ(Run,Walk,Dance1,Dance2)	5.61	21.87
β(Run,Walk,Dance1,Dance2)	5.93	18.33

Table 4: Statistics for 1200 interpolation operations

shapes of the interpolant. The interface allows the animator to manipulate control vertices of the B-Spline interpolant curve to shape the transition function. The curve is automatically constrained to lie within the framespace bounds. The physical length of the framespace time axis is transformed via logical mapping of a user-specified number of cycles onto it, to achieve transitions over variable durations. The resolution of the time axis is made proportional to the number of cycles, to maintain a constant number of samples per cycle and avoid aliasing effects. Fig. 6 shows the results of blending (β) and concatenation (ζ) between the basis motions. Running and walking are somewhat similar in motion characteristics, so the real challenge was in trying to blend different genres of actions like the *Egyptian rap* and running. The concatenations are seamless and yield consistent motion. Blending via 3D framespace interpolation yielded a curious mix of the two different dance styles and jogging, something which looks like a happy jive. Thus framespace interpolation can achieve the same *emotional qualities* as cited in [14] by using appropriate basis motions. For example, a cyclified angry gesticulation blended with walking would yield an angry walk. Performance statistics are presented in Table 4. Concatenation (ζ) and blending (β) operations (decided by the interpolant shape) are both presented for the four chosen combinations of primitives. The experiments use a 63 DOF articulated figure. For identical primitive-combinations and number of operation cycles, the lengths of the results are different for concatenation and blending because of the frequency blending component, which is used to achieve natural transition timing. Fig.7 illustrates the timing transition mechanism and velocity interpolation of translatory DOFs (monotonic curves in Fig.7). The smooth change in cyclic duration is clearly evident in the rotational DOFs. Animation clips of results presented in Table 4 are available at [1].

6 Discussion

Having explained the algorithm and presented the results, we now address some critical questions to compare our methods with related research, and evaluate the role of some of the employed techniques. Pertinent issues about why and how we modify existing techniques and what gains are achieved, are outlined below:

- **Simplified Time Warp vs Physically based Correspondence**: We have simplified Bruderlin's approach of using physically based correspondence techniques[13] to correlate and warp motions. We refer to our warp algorithm as *dynamic* because it constantly changes with time, unlike that in[3, 9].

- **Necessity of framespace warps in identifying reference blend frames**: While it is relatively simple

to determine the bounding region of interpolant P in 2D using coordinate geometry, doing so for irregular 3D sub-spaces is non-trivial and computation-intensive[6]. In our case, regularizing the subspaces via framespace warping, provides a common reference and corresponding blend frames can be easily extracted.

- **Performance considerations**: The performance of our proposed pre-blend framespace warp is highly efficient, since only a few event-reference points (states) are shifted along the time axis, instead of the entire set of signal samples. The warp is used to calculate the blend frames' locations, instead of physically adjusting all the samples at every juncture. Each transition operation involves three linear blending operations, namely, *framespace warp, DOF value interpolation* and *frequency interpolation*.

- **Quality considerations**: By using blend frames from the entire framespace, we exploit high frequency information in the basis motions, unlike[6]. Even dropping some of the correspondence events used in[6] yields no visible degradation, since the angular-blend resolution is not hampered. We use event states from the entire body and not just locomotion states (e.g. unlike[6]), to develop correspondence between actions. This widens the application of our algorithm in interpolating *significantly different* motions (eg. running and dancing). Lastly, though linear blending is being used, a tight coupling between framespace warping, DOF value interpolation and frequency blending, achieves a fluid transition.

Though our algorithm is efficient, it only tackles a subset of motion editing problems. For instance, it does not take care of motion retargetting. Though smooth interpolations are achieved, self collision may occur and additional validation needs to be done. Lastly, if basis motions are drastically different, results might not be acceptable due to the lack of adequate correspondence.

7 Summary and Future Work

We have presented a refined framespace interpolation algorithm, which is better suited (but not limited to) editing MOCAP data. Consistent interpolation is achieved by corresponding labelled states from different basis motions. Weights drawn from the interpolant drive a dynamic regularizing warp function, parametric interpolation and transition timing. Velocity interpolation is performed for pelvic translations. Four basis motions, *run, walk, dance1* and *dance2* have been used to illustrate blending and concatenation results.

Our chief contributions can be summarized as follows: a) Refined existing algorithm to exploit high frequency information of MOCAP data. b) Efficient computation via simplified dynamic framespace warps and linear blending. c) Fluid transition with minimal user specification, achieved via weight-coupled *framespace warp, DOF value interpolation* and *frequency interpolation*. d) Generalization of framespace interpolation to involve different classes of motions.

We are currently investigating a generalized decoupled blending mechanism, and seamless mixing of acyclic and cyclic primitives. Work remains on developing a robust IK assisted cyclification algorithm and motion validation scheme. The results we have achieved so far, are extremely encouraging. The framespace interpolation technique, though based on simple principles, promises to be an efficient, expressive and powerful animation tool, very much feasible for practical use.

8 Acknowledgements

We gratefully acknowledge our GI2000 reviewers, whose valuable feedback has improved the presentation of our paper and demo-clips[1] significantly.

References

[1] Golam Ashraf. Framespace interpolation results. *http://www.cgit.ntu.edu.sg/∼ashraf/framespace/*.

[2] Rama Bindiganavale and Norman I. Badler. Motion abstraction and mapping with spatial constraints. In *Modelling & Motion Capture Techniques for Virtual Environment,CAPTECH'98 Proceedings*, pages 70–82, 1998.

[3] Armin Bruderlin and Lance Williams. Motion signal processing. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 97–104. Addison Wesley, August 1995.

[4] Michael F. Cohen. Interactive spacetime control for animation. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 293–302, July 1992.

[5] Michael Gleicher. Retargeting motion to new characters. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 33–42. Addison Wesley, July 1998.

[6] S. Guo and J. Roberge. A high-level control mechanism for human locomotion based on parametric frame space interpolation. *Eurographics Computer*

Animation and Simulation EGCAS'96, pages 95–107, 1996.

[7] S. Guo, J. Roberge, and T. Grace. Controlling movement using parametric frame space interpolation. In N. Magnenat Thalmann and D. Thalmann, editors, *Models and Techniques in Computer Animation*, pages 216–227. Springer, Tokyo, 1994.

[8] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH '99*, Computer Graphics Proceedings, Annual Conference Series, pages 39–48. Addison Wesley, August 1999.

[9] Yongping Luo. Handling motion processing constraints for articulated figure animation. *Masters Thesis, School of Computer Science, Simon Fraser University*, November 1997.

[10] Roberto Maiocchi. A knowledge-based approach to the synthesis of human motion. In *IFIP Conference on Graphics Modeling*, pages 157–178, 1991.

[11] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5):32–40, September - October 1998.

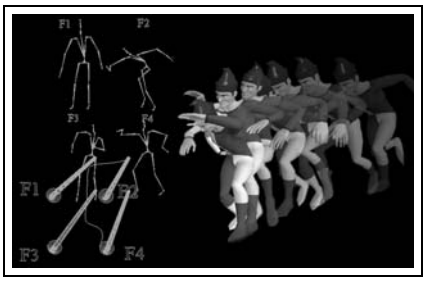
[12] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 147–154. Addison Wesley, August 1996.

[13] Thomas W. Sederberg and Eugene Greenwood. A physically based approach to 2D shape blending. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 25–34, July 1992.

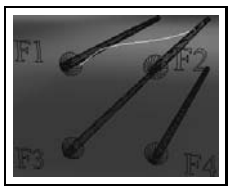
[14] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 91–96. Addison Wesley, August 1995.

[15] Andrew Witkin and Michael Kass. Spacetime constraints. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 159–168, August 1988.

[16] Andrew Witkin and Zoran Popović. Motion warping. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 105–108. Addison Wesley, August 1995.



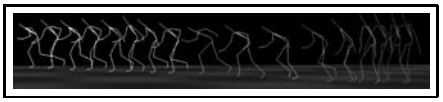
(a) Interface



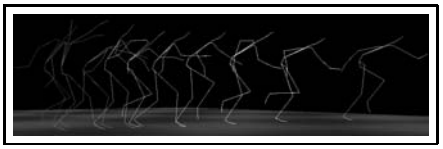
(b) 2D concatn.

(c) 3D blend

Figure 5: Framespace Interpolation interface.



(a) $\beta\{Run, Walk\}$



(b) $\zeta\{Dance2, Run\}$

(c) $\beta\{Run, Walk, Dance1, Dance2\}$

Figure 6: 2D & 3D Interpolation results

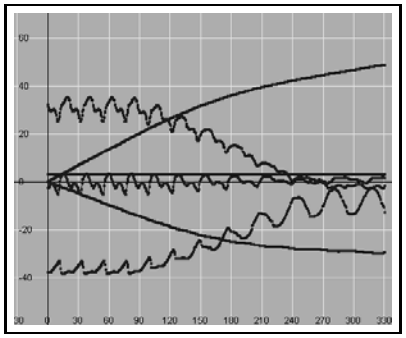
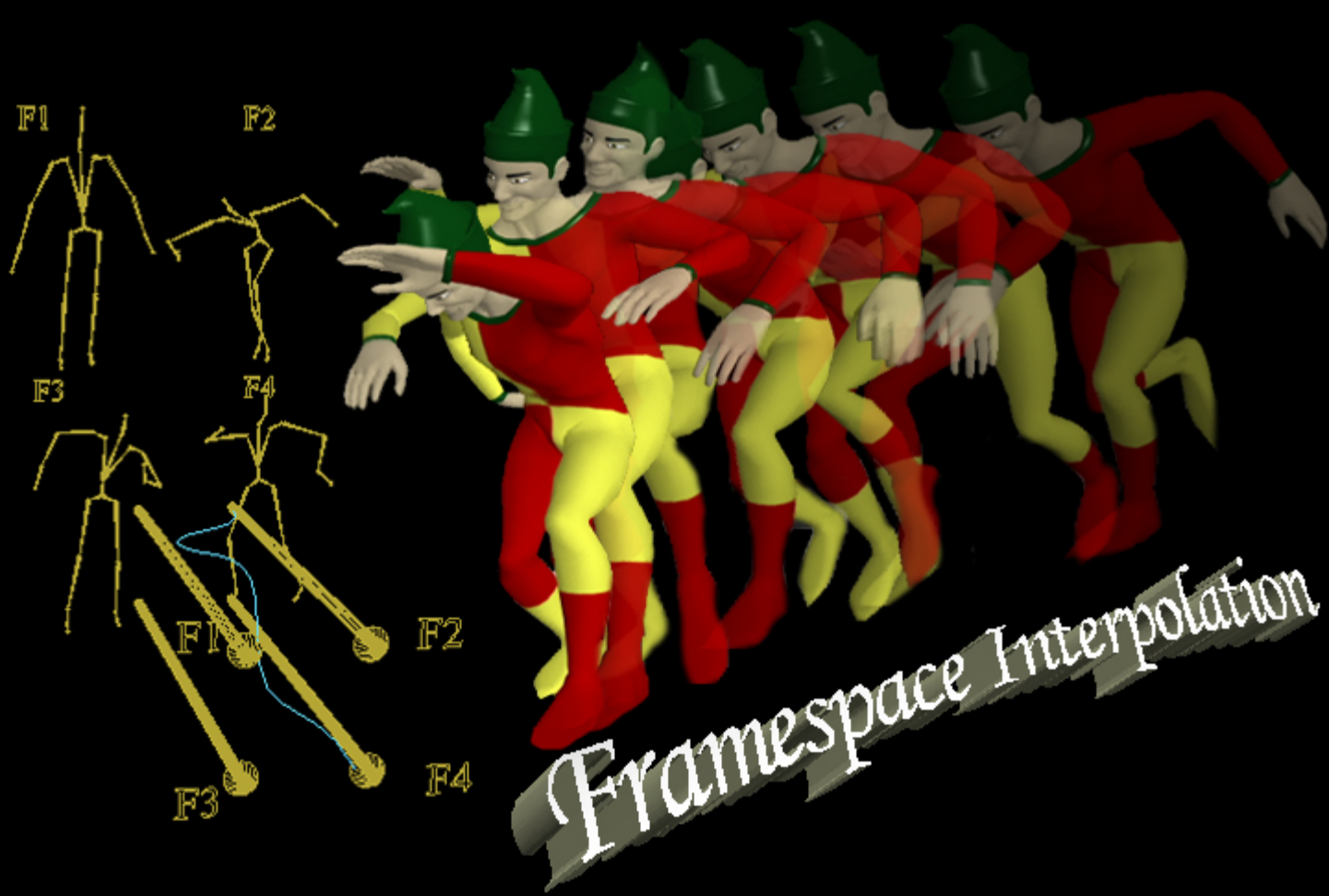


Figure 7: Pelvis DOFs of $\zeta\{w,r\}$


F1
F2
F3
F4
F1
F2
F3
F4
Framespace Interpolation