4D Parametric Motion Graphs for Interactive Animation

Dan Casas* University of Surrey
Margara Tejera* University of Surrey
Jean-Yves Guillemaut* University of Surrey
Adrian Hilton* University of Surrey

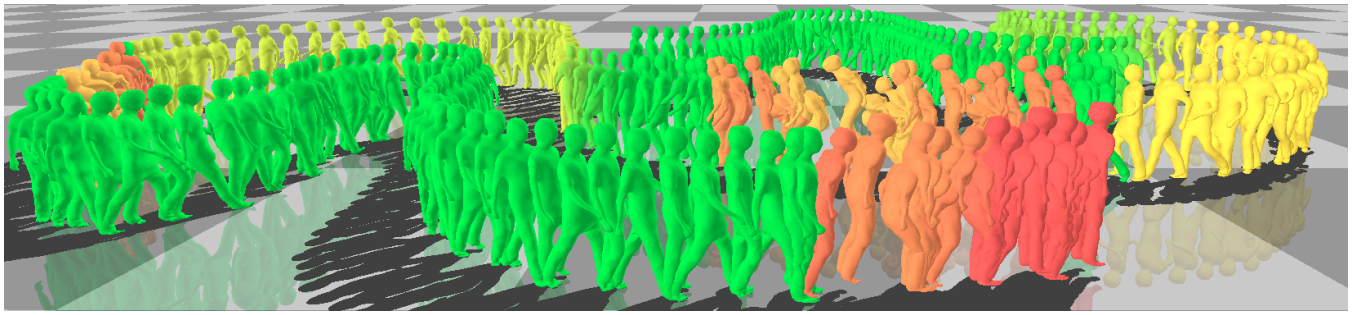


Figure 1: *Interactive character animation with a 4D parametric motion graph*

Abstract

A 4D parametric motion graph representation is presented for interactive animation from actor performance capture in a multiple camera studio. The representation is based on a 4D model database of temporally aligned mesh sequence reconstructions for multiple motions. High-level movement controls such as speed and direction are achieved by blending multiple mesh sequences of related motions. A real-time mesh sequence blending approach is introduced which combines the realistic deformation of previous non-linear solutions with efficient online computation. Transitions between different parametric motion spaces are evaluated in real-time based on surface shape and motion similarity. 4D parametric motion graphs allow real-time interactive character animation while preserving the natural dynamics of the captured performance.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—;

Keywords: character animation, 3D video, real-time animation

1 Introduction

Current interactive character authoring pipelines commonly consist of two steps: modelling and rigging of the character model which may be based on photographic reference or high-resolution 3D laser scans; and move-trees based on a database of skeletal motion capture together with inverse dynamic and kinematic solvers for secondary motion. Motion graphs [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002] and parameterised skeletal motion spaces [Heck and Gleicher 2007; Rose et al. 1998] enable representation and real-time interactive control of character movement from motion capture data. Authoring interactive characters requires a high-level of manual editing to achieve acceptable realism of appearance and movement.

*{d.casasguix, m.tejerapadilla, j.guillemaut, a.hilton}@surrey.ac.uk

Recent advances in performance capture [de Aguiar et al. 2008; Vlasic et al. 2008; Starck and Hilton 2007; Xu et al. 2011] have demonstrated highly realistic reconstruction and 4D representation as temporally coherent mesh sequences. This allows replay of the captured performance with free-viewpoint rendering and compositing of performance in post-production whilst maintaining photo-realism [Zitnick et al. 2004]. Captured sequences have subsequently been exploited for retargeting surface motion to other characters [Baran et al. 2009] and analysis of cloth motion to simulate novel animations through manipulation of skeletal motion and simulation of secondary cloth movement [Stoll et al. 2010]. Recent work exploited multiview skeletal tracking to index a video database enabling realistic offline character animation [Xu et al. 2011].

In this paper we present a 4D parametric motion graph representation for real-time interactive animation from a database of captured 4D mesh sequences. The principal contribution is the introduction of methods for high-level parametric control and transition between multiple motions which enable real-time animation from a database of mesh sequences whilst preserving the captured dynamics. Our main contributions are:

- Parameterisation of 4D models for interactive control: high-level motion parameterisation is introduced by real-time blending of multiple captured mesh sequences with hybrid non-linear deformation. This leads to parametric control of mesh sequences analogous to approaches for skeletal motion capture [Rose et al. 1998].
- Parametric motion graph representation of a database of 4D mesh sequences for interactive animation: a database of mesh sequences is represented in a graph-structure with nodes representing parameterised motions, and edges transitions between motions which preserve surface shape and non-rigid motion. This representation allows real-time interactive control analogous to skeletal move-trees or parametric motion graphs [Heck and Gleicher 2007].
- Online path optimisation for transition between parametric motion spaces with low-latency. This allows responsive interactive character control without delays in transitioning between motions.

Results are presented for interactive animation of character models constructed from databases of captured performance sequences for

a variety of movements. Character animations preserve the dynamics of the captured movement whilst allowing continuous interactive motion control. This demonstrates the potential of the approach as a methodology for authoring novel interactive 3D characters.

2 Related work

4D video-based performance reconstruction: Kanade et al. [Kanade and Rander 1997] pioneered the reconstruction of 3D mesh sequences of human performance for free-viewpoint replay with the Virtualized Reality^{TM} system using a $5m$ dome with 51 cameras. Multiple view video reconstruction results in an unstructured mesh sequence with an independent mesh at each frame. Advances in performance capture from video have enabled reconstruction of mesh sequences for human performance capturing the detailed deformation of clothing and hair [de Aguiar et al. 2008; Starck and Hilton 2007; Vlasic et al. 2008; Zitnick et al. 2004]. These approaches achieve a free-viewpoint rendering quality comparable to the captured video but are limited to performance replay. A critical step for editing and reuse of 4D performance capture is the temporal alignment of captured mesh sequences to obtain a consistent mesh structure with surface correspondence over time. A number of approaches have been proposed for temporal alignment of mesh sequences based on sequential frame-to-frame surface tracking. These can be categorised into two methodologies: model-based approaches which align a prior model of the surface with successive frames [Carranza et al. 2003; de Aguiar et al. 2008; Vlasic et al. 2008]; and surface-tracking or scene flow approaches which do not assume prior knowledge of the surface structure [Cagniart et al. 2010; Vedula et al. 2005; Wand et al. 2009]. Sequential alignment approaches have three inherent limitations: accumulation of errors in frame-to-frame alignment resulting in drift in correspondence over time; gross-errors for large non-rigid deformations which occur with rapid movements requiring manual correction; and sequential approaches are limited to alignment across single sequences. Non-sequential alignment approaches [Huang et al. 2011; Tung and Matsuyama 2010] have recently been introduced to overcome these limitations and allow alignment of databases of mesh sequences into a coherent mesh structure. This allows the construction of 4D models of temporally coherent mesh sequences from multiple view video performance capture, as used in this work.

Reuse and editing of 4D performance capture: To date the primary focus for research in 4D performance capture has been free-viewpoint video replay without modification or editing of the content. The lack of temporal coherence in the mesh sequence has prohibited the development of simple methods for manipulation. Animation from databases of mesh sequences of actor performance has been demonstrated by concatenating segments of captured sequences [Huang P. 2009; Starck et al. 2005]. This approach is analogous to previous example-based approaches to concatenative synthesis used for 2D video [Bregler et al. 1997; Schodl et al. 2000; Flagg et al. 2009] and motion graphs used for skeletal motion capture [Kovar et al. 2002; Arikan and Forsyth 2002]. Recently, example-based approaches through resampling video sequences have been extended to body motion [Xu et al. 2011; Flagg et al. 2009] allowing offline animation via key-frame or skeletal motion. These approaches preserve the realism of the captured sequences in rendering but are limited to replaying segments of the captured motion examples and do not allow the flexibility of conventional animation. More general free-form mesh sequence editing frameworks have been proposed [Kircher and Garland 2008; Xu et al. 2007]. However, the application of this approach to captured mesh sequences has been limited due to the requirement for a temporally coherent mesh structure. A central objective of this work is to allow real-time interactive control available in conventional animation with captured 4D mesh sequences.

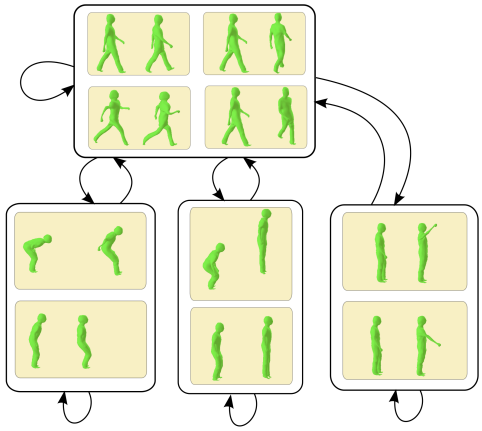
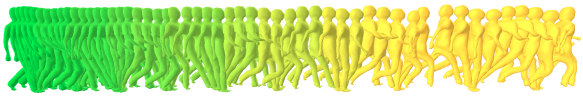


Figure 2: *Illustration of a 4D parametric motion graph showing four nodes with parameterised motion spaces: walk (top) parameterised for speed and direction; long-jump (bottom-left) parameterised for length; jump-up (bottom-middle) parameterised for height; and reach (bottom-right) parameterised for hand location*

Reuse and editing of skeletal performance capture: Marker-based technologies are widely used for skeletal performance capture. Since the introduction of skeletal performance capture to the entertainment industry in the early 90's a range of techniques to support editing and reuse have been developed. Space-time editing techniques [Gleicher 1997; Lee and Shin 1999; Popovic and Witkin 1999] provide powerful tools for interactive motion editing via key-frame manipulation. [Brundelin and Williams 1995] introduced parametric motion control by interpolating pairs of skeletal motions. Parametric motion synthesis was extended to blending multiple examples to create a parameterised skeletal motion space [Wiley and Hahn 1997; Rose et al. 1998; Kovar and Gleicher 2004; Mukai and Kuriyama 2005]. This allows continuous interactive motion control through high-level parameters such as velocity for walking or hand position for reaching. Skeletal motion graphs [Kovar et al. 2002; Arikan and Forsyth 2002; Lee et al. 2002] synthesise novel animations from a database of examples by concatenating segments of skeletal motion capture allowing transitions between a wide range of motions. [Heck and Gleicher 2007] introduced parametric motion graphs combining skeletal motion parameterisation with motion graphs to allow interactive character animation for a wide-range of motions with high-level continuous control. These approaches have enabled flexible editing and reuse of skeletal motion capture for character animation. This paper introduces analogous parameterisation for 4D performance capture to allow real-time interactive animation control.

3 4D Parametric Motion Graph

In this section we introduce a 4D parametric motion graph representation for interactive animation from a database of 4D movement sequences. Given a 4D performance database of mesh sequences for different movements with a consistent mesh structure at every frame we first achieve parametric control by combining multiple sequences of related motions. This gives a parametrierised motion space controlled by high-level parameters (for example walk speed/direction or jump height/length). Parameterised motions are combined in a motion graph to allow transitions between different motions. For both motion parametrisation and transitions based on



(a) Walk-to-run motion parameterisation with speed parameter from walk (green) to run (yellow)

(b) Jump motion parameterisation with distance parameter from short (red) to long (orange)

(c) Jump motion parameterisation with height parameter from low (white) to high (purple)

Figure 3: *Examples of parameterised motions between two motion sequences with continuous parameter variation (every 5^{th} frame)*

4D mesh sequences it is necessary to introduce techniques which preserve the surface motion whilst allowing real-time interaction. The 4D parametric motion graph representation allows interactive real-time control of character animation from a database of captured mesh sequences.

The 4D parametric motion graph represents the set of character motions and transitions which can be controlled interactively at run-time from a 4D performance capture database. Parameterised sets of motions form the graph nodes each representing a distinct set of motions which the character can perform with associated user-controlled animation parameters. The problem is then to define the graph edges which allow smooth transition between motions at run-time whilst maintaining real-time interactive control of the character motion. The parameter space for each node is continuous and we cannot therefore pre-compute all possible optimal transitions between parameterised motions. We therefore introduce a real-time approach to evaluate the optimal motion transitions with low-latency. Figure 2 shows a simple 4D parametric motion graph with nodes for four motions: walk with parameters for speed and direction; long-jump with parameters for length; jump-up with parameters for height; and reach with parameters for hand position. The arrows between nodes indicate possible transitions and the arrows to the same node indicate loops for cyclic motion.

3.1 Mesh Sequence Parametrisation

Interactive animation requires the combination of multiple captured mesh sequences to allow continuous real-time control of movement with intuitive high-level parameters such as speed and direction for walking or height and distance for jumping. Methods for parameterisation of skeletal motion capture have previously been introduced [Kovar and Gleicher 2004; Rose et al. 1998; Mukai and Kuriyama 2005] based on linear interpolation of joint angles. Linear blending of meshes is computationally efficient but may result in unrealistic deformation or mesh collapse if there are significant differences in shape. Non-linear blending of meshes produces superior deformation [Kircher and Garland 2008; Sorkine 2006; Sumner and Popovic 2004; Xu et al. 2007] but commonly requires least-squares solution of a system of equations which is prohibitive for real-time interaction.

Three steps are required to achieve high-level parametric control from mesh sequences: time-warping to align the mesh sequences; non-linear mesh blending of the time-warped sequences; and mapping from low level blending weights to high-level parameters (speed, direction, etc.). In this section we focus on real-time non-linear mesh blending which is the novel contribution of this work. As in previous work on skeletal motion parameterisation we assume that individual mesh sequences $M_i(t)$ are temporally aligned by a continuous time-warp function $t = f(t_i)$ [Brundelin and Williams 1995; Witkin and Popovic 1995] which aligns corresponding poses prior to blending such that $t \in [0, 1]$ for all sequences.

3.1.1 Real-Time Hybrid Non-Linear Mesh Sequence Blending

In this work we introduce a real-time approach to mesh blending which exploits offline pre-computation of non-linear deformation for a small set of intermediate parameter values. Our approach, referred to as hybrid non-linear blending, is a piece-wise linear interpolation of non-linear interpolated intermediate meshes. Differences between the linear and non-linear mesh deformation are pre-computed and used to correct errors in linear deformation. This approach allows approximation of the non-linear deformation to within a user-specified tolerance whilst allowing real-time computation with a similar cost to linear blending. The price paid is a modest increase in memory required to store intermediate non-linear mesh displacements for blending.

Given a set of N temporally aligned mesh sequences $\mathbf{M} = \{M_i(t)\}_{i=1}^{N}$ of the same or similar motions (e.g. walk and run) we want to compute a blended mesh deformation according to a set of weights $\mathbf{w} = \{w_i\}_{i=1}^{N}$: $M_{NL}(t, \mathbf{w}) = b(\mathbf{M}, \mathbf{w})$, where $b()$ is a non-linear blend function which interpolates the rotation and change in shape independently for each element on the mesh according to the weights $\mathbf{w}$ and performs a least-squares solution to obtain the resulting mesh $M_{NL}(t, \mathbf{w})$. This non-linear operation can be performed offline using existing approaches [Kircher and Garland 2008; Sorkine 2006; Sumner and Popovic 2004; Xu et al. 2007], throughout this work we employ a volumetric Laplacian deformation framework based on [Sorkine 2006].

Linear vertex blending gives an approximate mesh $M_L(t, \mathbf{w})$ as a weighted sum of vertex positions: $M_L(t) = \frac{1}{\sum w_i} \sum w_i M_i(t)$, where $w_i M_i(t)$ denotes the product of the mesh vertex positions with weight w_i. Given the non-linear mesh deformation $M_{NL}(t, \mathbf{w})$ and linear approximation $M_L(t, \mathbf{w})$ we can evaluate a displacement field: $D_{NL}(t, \mathbf{w}) = M_{NL}(t, \mathbf{w}) - M_L(t, \mathbf{w})$. The exact non-linear deformation for blend weights $\mathbf{w}$ can then be recovered by linear interpolation together with a non-linear correction: $M_{NL}(t, \mathbf{w}) = M_L(t, \mathbf{w}) + D_{NL}(t, \mathbf{w})$. Note that for blending between mesh sequences of similar motions linear blending will give a reasonable approximation for large parts of the surface $D_{NL}(t, \mathbf{w}) \approx 0$, this allows efficient representation of regions with significant non-linear deformation $D_{NL}(t, \mathbf{w}) > \epsilon$.

To accurately approximate the non-linear deformation for blending a set of N reference meshes $\mathbf{M}$ with arbitrary weights $\mathbf{w}$ we pre-compute the non-linear displacement field $D_{NL}(t, \mathbf{w}_j)$ at a discrete set of intermediate weight values $\mathbf{w}_j$ to give an additional set of N_{NL} reference meshes for interpolation: $M_j(t, \mathbf{w}_j) = M_L(t, \mathbf{w}_j) + D_{NL}(t, \mathbf{w}_j)$. Real-time online interpolation is then performed using a linear vertex blending with the non-linear correction:

$$M(t, \mathbf{w}) = \sum_{j=1}^{N+N_{NL}} g(\mathbf{w}, \mathbf{w}_j)(M_L(t, \mathbf{w}_j) + D_{NL}(t, \mathbf{w}_j)) \quad (1)$$

where $g(\mathbf{w}, \mathbf{w}_j)$ is a weight function giving a linear blend of

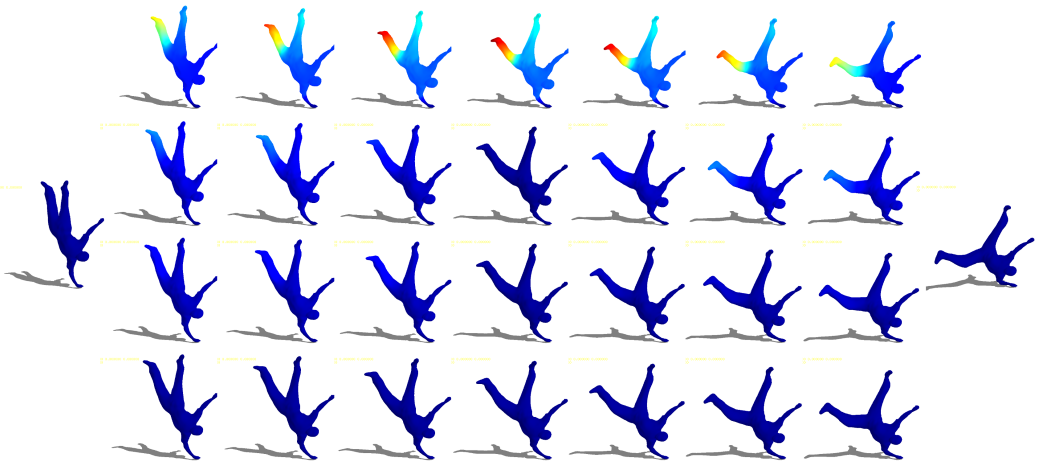


Figure 4: *Comparison of blending between two meshes (left, right) using linear (top row), non-linear (bottom row) and the proposed real-time hybrid solution to non-linear mesh blending with one reference (2nd row) and three references (3rd row). Heat-maps show errors vs. non-linear blending from dark-blue (zero) to red (maximum). The 2nd row shows that our hybrid method with one reference gives a reasonable approximation to the non-linear result.*

Method	Max. Error	Time
Linear	14.38 %	0.008 *sec / frame*
Hybrid 1 reference	3.67 %	0.015 *sec / frame*
Hybrid 3 references	1.60 %	0.017 *sec / frame*
Non-linear	0.00 %	0.749 *sec / frame*

Table 1: *Maximum vertex displacement error with respect to non-linear blending as a percentage of model size for in meshes in Figure 4.*

the nearest pair of reference meshes for each weight $w_i \in \mathbf{w}$ and zero for all others. Equation 1 gives an exact solution at the reference meshes and an interpolation of the nearest reference meshes elsewhere. Recursive bisection subdivision of the weight space $\mathbf{w}$ is performed to evaluate a set of non-linearly interpolated reference meshes such that for all $\mathbf{w}$ the approximation error $|M_{NL}(t, \mathbf{w}) - M(t, \mathbf{w})| < \epsilon$. Typically for interpolation of mesh sequences representing related motions only a single subdivision is required.

3.1.2 Evaluation of Hybrid Non-Linear Blending

Hybrid non-linear mesh blending allows accurate approximation of non-linear mesh deformation whilst maintaining the computational performance of linear blending to allow real-time interactive animation. Figure 4 presents a comparison of errors for linear blending with the proposed hybrid non-linear approach with one and three interpolated reference mesh. Table 1 presents quantitative results for error and CPU-time. This shows that the proposed real-time hybrid non-linear mesh blending approach achieves accurate approximation even with a single intermediate non-linear displacement map (2nd row) whereas linear blending results in large errors (top).

Figure 5 characterises the representation error and storage cost against number of subdivisions for different error thresholds ϵ. Figure 5(b) shows that there is a relatively small error reduction for thresholds below 5mm while there the memory usage increases significantly. This is caused by the $5mm$ resolution of the original database such that details below this level are not reconstructed.

3.1.3 High-Level Parametric Control

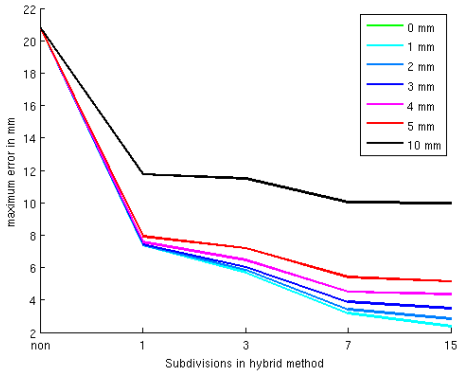
High-level parametric control is achieved by learning a mapping function $h(\mathbf{w})$ between the blend weights and user specified motion parameters $\mathbf{p}$. As in skeletal motion blending the blend weights do not provide an intuitive parameterisation of the motion. We therefore learn a mapping $\mathbf{w} = h^{-1}(\mathbf{p})$ from the user-specified parameter to the corresponding blend weights required to generate the desired motion. Motion parameters $\mathbf{p}$ are high-level user specified controls for a particular class of motions such as speed and direction for walk or run, end-effector position for reaching, or height and distance for a jump. The inverse mapping function $h^{-1}()$ from parameters to weights can be constructed by a discrete sampling of the weight space $\mathbf{w}$ and evaluation of the corresponding motion parameters $\mathbf{p}$ [Kovar and Gleicher 2004].

3.2 Motion Transitions

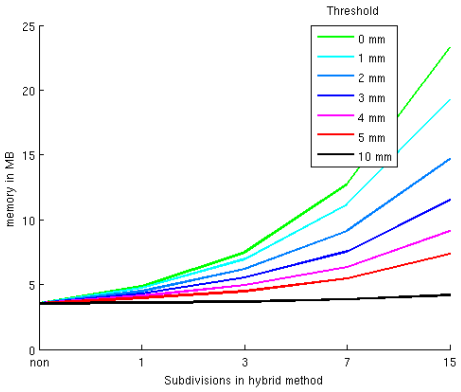
Transitions between parameterised spaces for different motions in the 4D motion graph are required to allow interactive character animation with multiple motions. Natural transitions require a similar shape and non-rigid motion in both spaces. In addition, for responsive interaction in real-time character control it is important to ensure low latency between the user input and motion transition.

Previous parametric motion graphs [Heck and Gleicher 2007] based on skeletal motion sequences pre-computed a discrete set of good transitions by evaluating the similarity in pose and motion between paris of source and target points. To limit the memory required a fixed number of good transitions are stored and interpolated at run-time. Pre-computation of a fixed set of transitions can result in a relatively high latency due to the delay between the current state and next pre-computed good transition.

We introduce an alternative approach which does not require the pre-computation of a discrete set of transitions. Optimal transitions are computed at run-time based on the current state in the source space and desired state in the target motion space. This approach has two advantages: transitions are not limited to a pre-computed fixed set allowing the best transition for a given starting point to be evaluated at run-time; and transition points can be evaluated on the fly to minimise latency whilst ensuring a smooth transition.



(a) Maximum displacement error of the hybrid method, depending on the number of subdivisions and threshold ϵ



(b) Memory in MB of the precomputed matrix D_{NL}, depending on number of subdivisions and threshold ϵ

Figure 5: *Evaluation of the maximum error and memory usage of the hybrid blending method.*

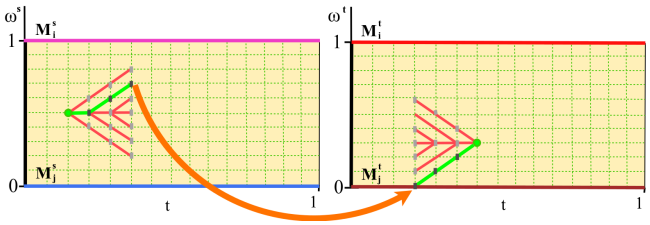


Figure 6: *Illustration of the transition between two parameterised motion space showing the trellis of candidate transition frames*

A parameterised motion is defined as a mesh sequence $M^r(t^r, \mathbf{p}^r)$ where $\mathbf{p}^r \in \Re^{N^r}$ is the set of user controlled parameters for the r^{th} motion class. Each parameterised motion $M^r(t^r, \mathbf{p}^r)$ is computed from a set of 4D mesh sequences $\{M_i^r(t)\}_{i=1}^{N^r}$ according to Equation (1). Given a current source state $M^s(t^s, \mathbf{p}^s)$ and a target state $M^t(t^t, \mathbf{p}^t)$ the problem is then to find the optimal transition path at run-time. To evaluate the best transition path P_{opt} online we optimise a cost function representing the trade-off between similarity in mesh shape and motion at transition, $E_S(P)$, and the latency, $E_L(P)$, or delay in transition for a path P between the source state $M^s(t^s, \mathbf{p}^s)$ and target state $M^t(t^t, \mathbf{p}^t)$:

$$P_{opt} = \arg\min_{P \in \Omega} \left(E_S(P) + \lambda E_L(P)\right) \tag{2}$$

where λ defines the trade-off between transition similarity and latency ($\lambda = 5$ throughout this work). The transition path P is optimised over a trellis of frames starting at the current frame $M^s(t^s, \mathbf{p}^s)$ in the source motion space and a trellis ending at the target frame $M^t(t^t, \mathbf{p}^t)$ in the target motion space as illustrated in Figure 6. The trellis is sampled forward in time at discrete intervals in time Δt and parameters $\Delta \mathbf{p}$ up to a maximum depth l_{max} in the source space. Similarly from the target frame a trellis is constructed going backward in time. This defines a set of candidate paths $P \in \Omega$ with transition points between each pair of frames in the source and target trellis. For a path P the latency cost $E_L(P)$ is measured as the number of frames in the path P between the source and target frames. Transition similarity cost $E_S(P)$ is measured as the similarity in mesh shape and motion at the transition point between the source and target motion space for the path P.

3.2.1 Transition Similarity Cost

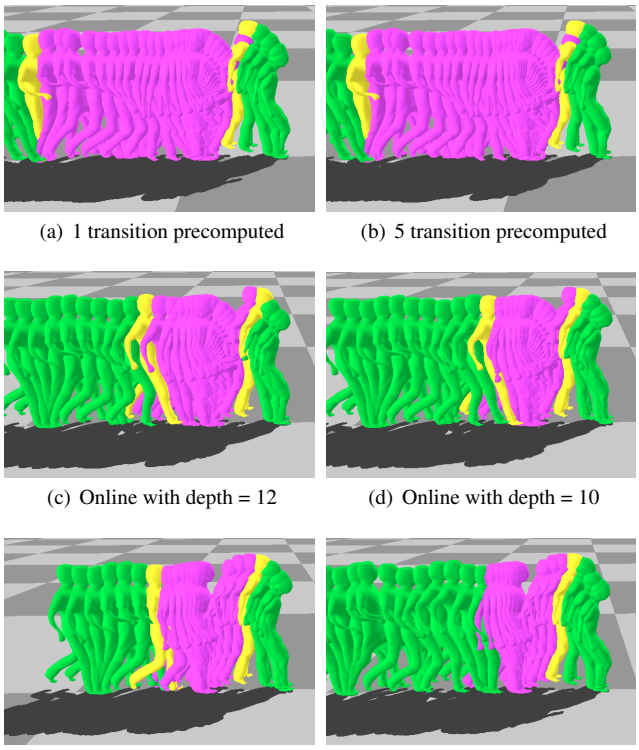
The mesh and motion similarity between any two source and target frames $s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t))$ is defined as follows. As the vertex correspondence between the source and target meshes are known we can compute the shape and motion similarity between any pair of meshes. The Euclidean distances between their vertex positions and velocities gives a distance: $d(M_i, M_j) = \frac{1}{N_v}(\|X_i - X_j\|^2 + \lambda\|V_i - V_j\|^2)$ where vertex velocity $V_i(t) = X_i(t) - X_i(t-1)$. Similarity is then computed by normalising by the maximum distance: $s(M_i, M_j) = 1 - \frac{d(M_i, M_j)}{max(d(M_i, M_j))}$.

Evaluation of this similarity $s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t))$ between pairs of frames in a continuous parametric motion space is prohibitively expensive for online computation. Real-time computation can be achieved by approximating the similarity using an interpolation of the corresponding pre-computed similarity between the mesh sequences used to build the source and target motion spaces.

For each pair of source $M_i^s(t_u)$ and target $M_j^t(t_v)$ input mesh sequences we evaluate the shape and motion similarity $s(M_i^s(t_u), M_j^t(t_v))$ for all frames $t_u \in [0, T_i], t_v \in [0, T_j]$ giving a shape similarity matrix $\mathbf{S}_{ij}^{st}(u, v)$. Online real-time computation of the shape similarity for the source and target mesh is then evaluated as a weighted sum of the similarity for all source and target mesh pairs based on the motion parameters $\mathbf{p}$. For convex weights $\mathbf{w} = h^{-1}(\mathbf{p}) \in [0, 1]$ it can be shown that the following inequality holds (see supplementary material for proof):

$$\begin{aligned} s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t)) &\geq \sum_{i=1}^{N^s}\sum_{j=1}^{N^t} w_i^s w_j^t s(M_i^s(t^s), M_j^t(t^t)) \\ &\geq \mathbf{w}^s \mathbf{S}^\top (\mathbf{w}^t)^\top \end{aligned} \tag{3}$$

Thus a lower-bound on the similarity between any two frames in the parameterised source and target motion space can be evaluated by a sum of the weighted similarity between the source and target mesh sequences. As the pairwise sequence similarities can be pre-computed, evaluation of the maximum lower-bound on similarity between any point in the two parameterised spaces can be efficiently computed at run-time. If $\mathbf{w}^s$ and $\mathbf{w}^t$ are the source and target weight vectors then we can evaluate the maximum similarity



(a) 1 transition precomputed (b) 5 transition precomputed

(c) Online with depth = 12 (d) Online with depth = 10

(e) Online with depth = 5 (f) Online with depth = 3

Figure 7: *Comparison of the same transition between a run motion $\alpha = 0.9$ of the parametric space showed in Figure 3(a) to a jump motion alpha $= 0.5$ of the parametric space showed in Figure 3(b). In yellow, source $M^s(t^s, \mathbf{p}^s)$ and target $M^t(t^t, \mathbf{p}^t)$ meshes. In pink the meshes computed to transition.*

according to equation (3). $\mathbf{S}$ is the $N^s \times N^t$ matrix of similarities between N^s source and N^t target mesh sequences, which comprises pre-computed similarities from $\mathbf{S}^{st}_{ij}$.

The optimal transition path P_{opt} for a given source frame (current state) and target frame (end state) is evaluated according to equation 2 at run-time by evaluating the cost for all paths in the trellis defining possible transition between the source and target motion spaces. The similarity cost $E_S(P)$ for path P is defined as the simililarity at the transition frames between the source and target motion spaces evaluated according to equation 3: $E_S(P) = s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t))$. The optimal path P_{opt} can be evaluated with low-latency at run-time because computation of the similarity cost by interpolation, equation 3, is computationally efficient.

3.2.2 Transition Performance Evaluation

Performance of the proposed online approach has been evaluated against an offline method in which a fixed set of the n best transitions between two parametric spaces was precomputed and stored. Table 2 presents quantitative results for transition cost, latency and computation time averaged over 50 transitions with random start and end points in the source and target motion space. Results demonstrate that the proposed online path optimisation achieves lower latency (delay in transition) for a similar transition cost, whilst maintaining real-time performance for evaluation of transitions (<0.2ms/transition). Pre-computation of fixed transitions gives a marginally lower similarity as it selects the best transition points, but results in significantly higher latency.

Figure 7 shows a qualitative comparison of the same transitions using 6 different configurations (results are also presented in the supplementary[1] video). Figures 7(a,b) show results with 1 and 5 precomputed transitions respectively. This demonstrates the relatively high-latency with a large number of transition frames (pink). Figures 7(c-f) shows the performance of the proposed online path optimisation approach using trellis depths (l_{max}) of 12, 10, 5 and 3. A smooth transition with reduced latency (smaller number of transition frames) is produced for online path optimisation with a trellis depth of 12 and 10 as shown in Figures 7(c,d)). Figures 7(e,f) show that as the trellis depth is reduced to 5 and 3 unnatural jumps in the motion appear at the transition, this is due to the restricted path optimisation not finding a good transition point between the motion spaces within the trellis.

In this work we use time step size $\Delta t = \alpha T$ and parameter step size $\Delta \mathbf{p} = \alpha(\mathbf{p}_{max} - \mathbf{p}_{min})$ with $\alpha = 0.1$, where the parameter range between the interpolated motion samples is $[\mathbf{p}_{min}, \mathbf{p}_{max}]$.

4 Results and Applications

Datasets used in this work are reconstructed from multiple view video capture of actor performance with 8 HD cameras equally spaced in a circle and capture volume $5m^2 \times 2m$. Reconstruction is performed using multi-view stereo [Starck and Hilton 2007] followed by temporal alignment of all frames to have a consistent mesh structure. Parametric motion spaces are then constructed from related motions. In this work we use a single intermediate mesh for hybrid non-linear interpolation which gives accurate approximate. 4D parametric motion graphs are constructed using 10 motions (walk, run, left turn, right turn, reach low/high, jump low/high and jump short/long) of 20-80 frames each with a 3000 vertex mesh.

Examples of interactive character animation from 4D performance capture are shown in Figures 1 and 8(a,b). Meshes are coloured to show the motion space and parameter change. The accompanying video shows the real-time interactive animation control and motion transitions. Character animation runs at interactive rates $>50Hz$ with all parameter control and transitions evaluated on a single CPU. Results demonstrate that the 4D parametric motion graph enables interactive control with smooth variation in character motion in response to changes in high-level parameters. Figure 8(c) shows another application for real-time character animation where the user interactively sets a path defined by a set of contiguous Bézier curves, which the character automatically follows by finding the path in the graph that best satisfies user constraints.

5 Discussion

The results presented in this paper demonstrate the potential for interactive character animation from 4D performance capture. Mesh-sequence parameterisation and the parametric surface motion graph representation enable real-time interaction with high-level movement control. Limitations of the current implementation include:

- 4D performance capture: Although not addressed in this paper current reconstruction of actor performance and subsequent temporal alignment into a database of sequences achieves realistic results but contains errors. Average errors are in the order of $1-2cm$ rms for reconstruction and alignment but may be larger for rapid movement where large inter-frame deformations occur requiring manual correction. This level of error can result in loss of surface detail and surface noise which is visible in the resulting animated models unless textured with the observed surface appearance. Further research is required

[1] Supplementary video: `http://vimeo.com/33399136`

Method	$\mathbf{E_S(P_{opt})}$	$\lambda\mathbf{E_L(P_{opt})}$	Latency(s)	online CPU time (s)	offline CPU time (s)
1 Precomputed	251.10	207.00	1.656	0.000005	93.32
5 Precomputed	236.13	191.59	1.533	0.000007	93.32
10 Precomputed	237.28	208.38	1.444	0.000010	93.32
Online depth = 12	254.41	166.99	1.336	0.000190	12.56
Online depth = 10	276.30	116.63	0.933	0.000085	12.56

Table 2: *Comparison of transition cost, latency and computation time for pre-computed fixed transitions with the proposed online computation($\lambda = 5$, frame-rate$= 0.04s$). Cost and computation times are average values of 50 randomly generated transitions for each method between walk/run and long/short jump parametric motion spaces.*

to improve algorithms for reconstruction and alignment to allow animation without visible artefacts.

- Mesh sequence parameterisation: A hybrid approach to approximate non-linear blending whilst maintaining real-time performance is proposed for parameterisation of multiple sequences of related motions. This combines the realistic mesh deformation of non-linear approaches [Kircher and Garland 2008; Sorkine 2006; Sumner and Popovic 2004; Xu et al. 2007] with the efficiency of linear blending and is demonstrated to give an accurate approximation with real-time performance ($<10ms$/frame). This approach requires a small number of additional intermediate non-linear displacement maps to be pre-computed for each parameterised motion.
- Transitions in high-dimensional parametric spaces: Online computation of transitions between motion spaces with up to 3 parameters can be performed in real-time ($< 0.2ms$). Online optimisation of transition paths is shown to give reduced latency compared to fixed precomputed transitions. This allows optimal transitions to be computed with respect to the current state. Higher-dimensional spaces require an exponentially increasing number of candidate transitions to be evaluated. GPU implementation and pre-computation of transitions costs could be employed at the expense of increased storage and fixed transitions locations leading to increased latency.

6 Conclusion

A system for real-time interactive character animation from 4D actor performance capture has been presented. The approach is based on a database of temporally aligned mesh sequence reconstructions of an actor performing multiple motions. 4D parametric motion graphs are introduced enabling movement control and transition between different motions. Real-time continuous high-level parametric motion control is achieved by blending multiple mesh sequences of related motions. A hybrid mesh sequence blending approach is introduced which combines the realistic deformation achieved with non-linear blending [Kircher and Garland 2008; Sorkine 2006; Sumner and Popovic 2004; Xu et al. 2007] with the real-time performance of linear blending. Natural transitions between different parametric motion spaces are evaluated in real-time through a transition path optimisation based on shape, non-rigid motion similarity and latency. Online transition optimisation from the current source state is demonstrated to reduce latency whilst maintaining real-time performance. This representation allows interactive animation from mesh sequences which is analogous to approaches previously introduced skeletal data [Heck and Gleicher 2007; Rose et al. 1998; Kovar et al. 2002; Arikan and Forsyth 2002]. Results for a CPU based implementation of the parametric control and motion transitions demonstrate a frame-rate $>50Hz$ for a mesh of 3000 vertices. 4D parametric motion graphs provide an approach to real-time interactive animation from a database of mesh sequences of reconstructed actor performance while preserving the captured dynamics and appearance. Further research is required to improve the resolution of both reconstruction and temporal mesh sequence alignment.

Acknowledgements: EU IST Project RE@CT.

References

ARIKAN, O., AND FORSYTH, D. 2002. Synthesizing constrained motions from examples. In *Proc. ACM SIGGRAPH 2002*.

BARAN, I., VLASIC, D., GRINSPUN, E., AND POPOVIC, J. 2009. Semantic deformation transfer. In *Proc. ACM SIGGRAPH 2009*.

BREGLER, C., COVELL, M., AND SLANEY, M. 1997. Video rewrite: Driving visual speech with audio. In *Proc. ACM SIGGRAPH 1997*, 1—8.

BRUNDELIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proc. ACM SIGGRAPH 1995*, 97–104.

CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Free-Form Mesh Tracking: a Patch-Based Approach. In *Conference on Computer Vision and Pattern Recognition 2010*, 1339–1346.

CARRANZA, J., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2003. Free-viewpoint video of human actors. In *Proc. ACM SIGGRAPH 2003*, 565—577.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance Capture from Sparse Multi-view Video. *Proc. ACM SIGGRAPH 2008 27*, 3.

FLAGG, M., NAKAZAWA, A., ZHANG, Q., KANG, S.-B., RYU, Y., ESSA, I., AND REHG, J. 2009. Human Video Textures. In *ACM Symposium on Interactive 3D Graphics*.

GLEICHER, M. 1997. Motion Editing with Spacetime Constraints. In *ACM Symposium on Interactive 3D Graphics*.

HECK, R., AND GLEICHER, M. 2007. Parametric Motion Graphs. In *ACM Symposium on Interactive 3D Graphics*.

HUANG, P., BUDD, C., AND HILTON, A. 2011. Global temporal registration of multiple non-rigid surface sequences. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition 2011*.

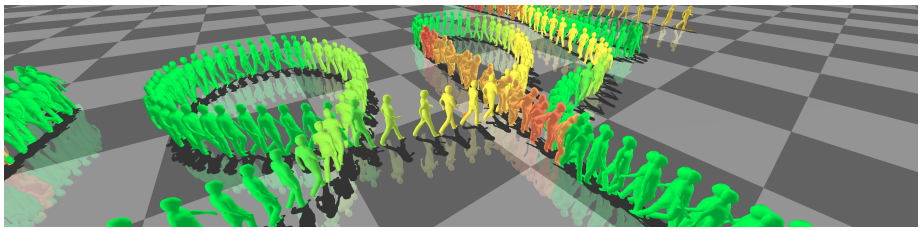
HUANG P., HILTON A., S. J. 2009. Human motion synthesis from 3d video. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition 2009*.

KANADE, T., AND RANDER, P. 1997. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*.

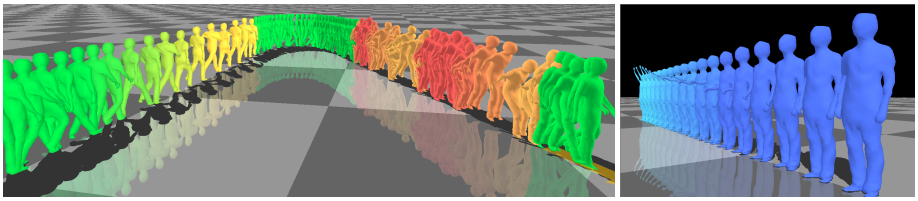
KIRCHER, S., AND GARLAND, M. 2008. Free-Form Motion Processing. *ACM Transactions on Graphics 27*, 2, 1–13.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large date sets. In *Proc. ACM SIGGRAPH 2004*, 23(3):559—568.

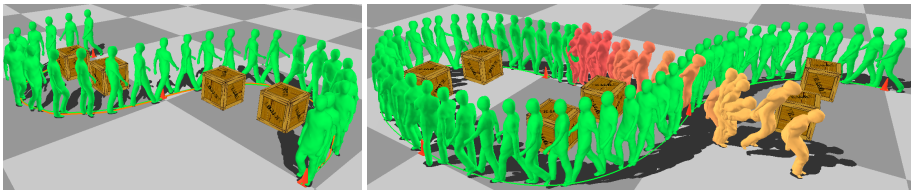
KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proc. ACM SIGGRAPH 2002*, 473–482.



(a) Real-time interactive character animation with a 4D parametric motion graph



(b) Interactive parametric control of jump and reach motions



(c) A character automatically follows a path set by the user to reach the red cones, avoiding the obstacles.

Figure 8: *Real-time interactive animation from performance capture using a 4D parametric motion graph. Meshes rendered every 10th frame. Colour represents changes in parameter values and motion space. See supplementary video to view results.*

LEE, J., AND SHIN, S. 1999. A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures. In *Proc. ACM SIGGRAPH 1999*, 39–48.

LEE, J., CHAI, J., REITSMA, P., HODGINS, J., AND POLLARD, N. 2002. Interactive control of avatars animated with human motion data. In *Proc. ACM SIGGRAPH 2002*, 491—500.

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical Motion Interpolation. In *Proc. ACM SIGGRAPH 2005*.

POPOVIC, Z., AND WITKIN, A. 1999. Physically Based Motion Transformation. In *Proc. ACM SIGGRAPH 1999*.

ROSE, C., COHEN, M., AND BODENHEIMER, B. 1998. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

SCHODL, A., SZELISKI, R. AMD SALESIN, D., AND ESSA, I. 2000. Video textures. In *Proc. ACM SIGGRAPH 2000*.

SORKINE, O. 2006. Differential Representations for Mesh Processing. *Computer Graphics Forum 25*, 4.

STARCK, J., AND HILTON, A. 2007. Surface capture for performance-based animation. *IEEE Computer Graphics and Application 27*, 21–31.

STARCK, J., MILLER, G., AND HILTON, A. 2005. Video-based character animation. In *ACM Symp. on Computer Animation*.

STOLL, C., GALL, J., DE AGUIAR, E., THRUN, S., AND THEOBALT, C. 2010. Video-based Reconstruction of Animatable Human Characters. In *ACM SIGGRAPH ASIA 2010*.

SUMNER, R., AND POPOVIC, J. 2004. Deformation Transfer for Triangle Meshes. *Proc. ACM SIGGRAPH 2004 23*, 3, 399–405.

TUNG, T., AND MATSUYAMA, T. 2010. Dynamic Surface Matching by Geodesic Mapping for Animation Transfer. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition 2010*.

VEDULA, S., BAKER, S., RANDER, P., COLLINS, R., AND KANADE, T. 2005. Three-Dimensional Scene Flow. *IEEE Trans. Pattern Analysis and Machine Intelligence 27*, 3.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. In *Proc. ACM SIGGRAPH 2008*.

WAND, M., ADAMS, B., OVSIANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P., AND SCHILLING, A. 2009. Efficient Reconstruction of Non-rigid Shape and Motion from Real-Time 3D Scanner Data. *ACM Trans.Graphics 28*, 2.

WILEY, D., AND HAHN, J. 1997. Interpolation synthesis for articulated figure motion. In *IEEE Virtual Reality International Symposium*, 157—160.

WITKIN, A., AND POPOVIC, Z. 1995. Motion Warping. *Proc. ACM SIGGRAPH 1995*.

XU, W., ZHOU, K., YU, Y., PENG, Q., AND GUO, B. 2007. Gradient domain editing of deforming mesh sequences. *Proc. ACM SIGGRAPH 2007 26*, 3.

XU, F., LIU, Y., STOLL, C., TOMPKIN, J., BHARAJ, G., DAI, Q., SEIDEL, H.-P., KAUTZ, J., AND THEOBALT, C. 2011. Video-based Characters - Creating New Human Performances from a Multi-view Video Database. *Proc. ACM SIGGRAPH 2011*.

ZITNICK, C., KANG, S., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. In *Proc. ACM SIGGRAPH 2004*.