Interactive Animation of 4D Performance Capture

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, *Member*, *IEEE*, and Adrian Hilton, *Member*, *IEEE*

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 nonlinear solutions with efficient online computation. Transitions between different parametric motion spaces are evaluated in real time based on surface shape and motion similarity. Four-dimensional parametric motion graphs allow real-time interactive character animation while preserving the natural dynamics of the captured performance.

Index Terms—Character animation, 3D video, real-time animation, multiview reconstruction, video-based animation, 4D modeling, 4D performance capture

1 Introduction

Current interactive character authoring pipelines commonly consist of two steps: modeling 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 [1], [2], [3] and parameterized skeletal motion spaces [4], [5] 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.

Recent advances in performance capture [6], [7], [8], [9] 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 postproduction while maintaining photo realism [10]. Captured sequences have subsequently been exploited for retargeting surface motion to other characters [11] and analysis of cloth motion to simulate novel animations through manipulation of skeletal motion and simulation of secondary cloth movement [12]. Recent work exploited multiview skeletal tracking to index a video database enabling realistic offline character animation [9].

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 that enable real-time animation from a database of mesh sequences while preserving the captured dynamics. Our main contributions are:

- Parameterization of 4D models for interactive control: high-level motion parameterization is introduced by real-time blending of multiple captured mesh sequences with hybrid nonlinear deformation. This leads to parametric control of mesh sequences analogous to approaches for skeletal motion capture [5].
- 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 parameterized motions, and edges transitions between motions, which preserve surface shape and nonrigid motion. This representation allows real-time interactive control analogous to skeletal move trees or parametric motion graphs [4].
- Accurate real-time approximation of similarity measurement between parameterized meshes, allowing online shape similarity evaluation of 4D models.
- Online path optimization 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 while allowing continuous interactive motion control. This demonstrates the potential of the approach as a methodology for authoring novel interactive 3D characters.

• *The authors are with Centre for Vision Speech and Signal Processing, Electronic Engineering, University of Surrey, GU2 7XH Guildford, United Kingdom. E-mail: d.casasguix@surrey.ac.uk.*

Manuscript received 31 May 2012; revised 15 Oct. 2012; accepted 21 Nov. 2012; published online 29 Nov. 2012.
Recommended for acceptance by M. Garland and R. Wang
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2012-05-0097.
Digital Object Identifier no. 10.1109/TVCG.2012.314.

This paper extends the original paper [13] presented in *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2012*. A new approach for real-time similarity measurement of parameterized meshes is introduced and evaluated. Other additions include a description of both the nonlinear mesh interpolation method used and the approach to temporally align mesh sequences together with further details of the implementation.

2 Related Work

Four-dimensional video-based performance reconstruction. Kanade and Rander [14] pioneered the reconstruction of 3D mesh sequences of human performance for free-viewpoint replay with the Virtualized Reality system using a $5\ \mathrm{m}$ 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 [6], [8], [7], [10]. 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 [15], [6], [7]; and surface-tracking or scene flow approaches, which do not assume prior knowledge of the surface structure [16], [17], [18]. 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 nonrigid deformations, which occur with rapid movements requiring manual correction; and sequential approaches are limited to alignment across single sequences. Nonsequential alignment approaches [19], [20], [21] 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 [22], [23]. This approach is analogous to previous example-based approaches to concatenative synthesis used for 2D video [24], [25], [26] and motion graphs used for skeletal motion capture [2], [1]. Recently, example-based approaches through resampling video sequences have been extended to body motion [9], [26] 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 [27], [28]. 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.

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 1990s a range of techniques to support editing and reuse have been developed. Space-time editing techniques [29], [30], [31], [32] provide powerful tools for interactive motion editing via key-frame manipulation. Brundelin and Williams [33] introduced parametric motion control by interpolating pairs of skeletal motions. Parametric motion synthesis was extended to blending multiple examples to create a parameteried skeletal motion space [34], [5], [35], [36]. This allows continuous interactive motion control through high-level parameters such as velocity for walking or hand position for reaching. Skeletal motion graphs [2], [1], [3], [37] 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 [4] introduced parametric motion graphs combining skeletal motion parameterization 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 parameterization 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 parametreized motion space controlled by high-level parameters (for example, walk speed/direction or jump height/length). Parameterized motions are combined in a motion graph to allow transitions between different motions. For both motion parametrization and transitions based on 4D mesh sequences it is necessary to introduce techniques that preserve the surface motion while 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 runtime from a 4D performance capture database. Parameterized sets of motions form the graph nodes each representing a distinct set of motions, which the character can perform with associated user-controlled

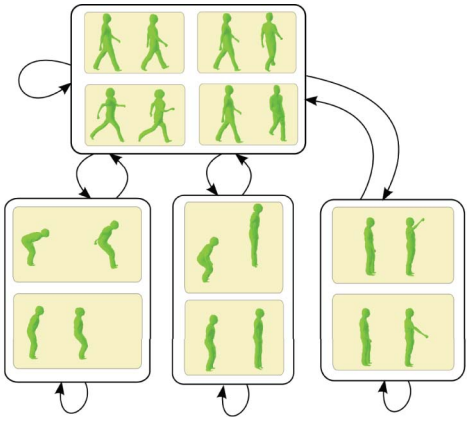
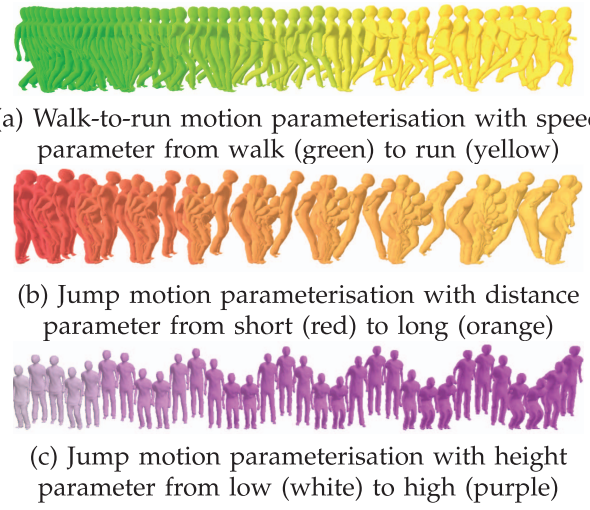


Fig. 1. Illustration of a 4D parametric motion graph showing four nodes with parameterized motion spaces: walk (top) parameterized for speed and direction; long-jump (bottom left) parameterized for length; jump-up (bottom middle) parameterized for height; and reach (bottom right) parameterized for hand location.



(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)

Fig. 2. Examples of parameterized motions between two motion sequences with continuous parameter variation (every fifth frame).

animation parameters. The problem is then to define the graph edges, which allow smooth transition between motions at runtime while maintaining real-time interactive control of the character motion. The parameter space for each node is continuous and we cannot therefore precompute all possible optimal transitions between parameterized motions. We, therefore, introduce a real-time approach to evaluate the optimal motion transitions with low latency. Fig. 1 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 Parametrization

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 parameterization of skeletal motion capture have previously been introduced [35], [5], [36] 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. Nonlinear blending of meshes produces superior deformation [27], [38], [39], [28] 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; nonlinear mesh blending of the time-warped sequences; and mapping from low-level blending weights to high-level parameters (speed, direction, and so on.). In this section, we focus on real-time nonlinear mesh blending, which is the novel contribution of this work. As in previous work on skeletal motion parameterization we assume that individual mesh sequences $M_i(t)$ are temporally aligned by a continuous time-warp function $t = f(t_i)$ [33], [40] which aligns corresponding poses prior to blending such that $t \in [0, 1]$ for all sequences. Mesh sequences are also assumed to be centered at the origin point. Original rotation and displacement of each mesh are used to spatially locate the character in the scenario.

3.1.1 4D Performance Capture

Actor performance is captured in a multiple camera studio and 3D mesh sequences are reconstructed using the multiple view stereo approach [8]. Meshes are reconstructed independently at each frame resulting in an unstructured mesh sequence with no temporal correspondence.

Throughout this work, we assume that all mesh sequences have been temporally aligned following the approach introduced in [19], [21]. Sparse correspondences between an unaligned frame and a reference mesh with the desired topology is estimated by SIFT feature and geometric patch matching [41]. Subsequently, these matches are incorporated as constraints to deform the reference mesh within a Laplacian framework [38], obtaining temporally aligned meshes as a result. Alignment of all frames across multiple sequences of a performance is achieved using pairwise alignment across the minimum spanning tree representation using the approach presented in [21].

3.1.2 Real-Time Hybrid Nonlinear Mesh Sequence Blending

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

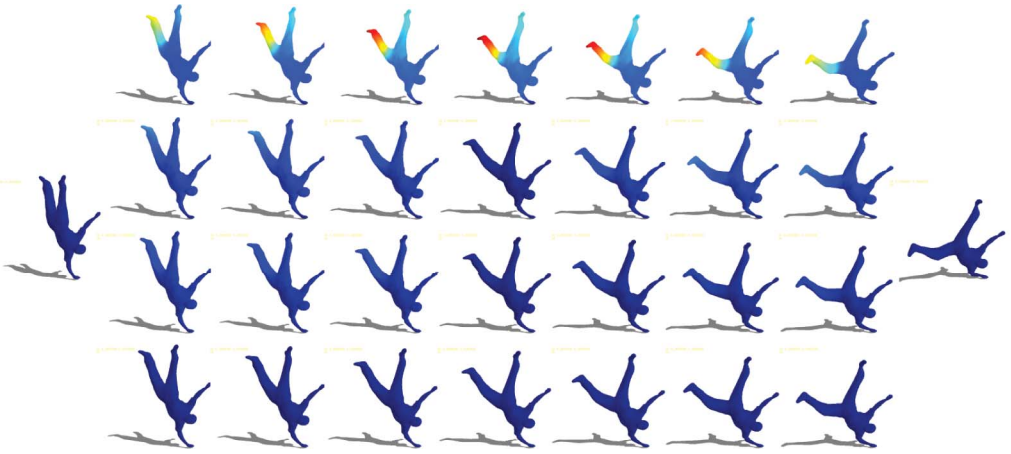


Fig. 3. Comparison of blending between two meshes (left, right) using linear (top row), nonlinear (bottom row) and the proposed real-time hybrid solution to nonlinear mesh blending with one reference (second row) and three references (third row). Heat maps show errors versus nonlinear blending from dark blue (zero) to red (maximum). The second row shows that our hybrid method with one reference gives a reasonable approximation to the nonlinear result.

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 $M_{NL}(t, \mathbf{w})$. Computation of triangle rotations q^k and scale/shear transformations S^k is performed using *slerp* and linear interpolation, respectively, as shown in:

$$q_{ij}^k = slerp\big(q_i^k, q_j^k, w_{ij}\big), \tag{1}$$

$$S_{ij}^k = S_i^k + w_{ij}\big(S_j^k - S_i^k\big), \tag{2}$$

where k denotes the index of each triangle, i and j refer to the pair of meshes $(M_i(t), M_j(t))$ being blended, and w_{ij} is the relevant weight. Applying the transformations q^k and S^k results in a set of new triangles than can be linked back together using existing approaches [27], [38], [39], [28]. In particular, we employ a Laplacian deformation framework [38], [42]. For $N > 2$, function $b()$ performs the described nonlinear interpolation iteratively for each pair of frames.

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 nonlinear 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 nonlinear deformation for blend weights $\mathbf{w}$ can then be recovered by linear interpolation together with a nonlinear 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 nonlinear deformation $D_{NL}(t, \mathbf{w}) > \epsilon$.

To accurately approximate the nonlinear deformation for blending a set of N reference meshes $\mathbf{M}$ with arbitrary weights $\mathbf{w}$ we precompute the nonlinear 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 nonlinear correction:

$$M(t, \mathbf{w}) = \sum_{j=1}^{N+N_{NL}} g(\mathbf{w}, \mathbf{w}_j)(M_L(t, \mathbf{w}_j) + D_{NL}(t, \mathbf{w}_j)), \tag{3}$$

where $g(\mathbf{w}, \mathbf{w}_j)$ is a weight function giving a linear blend of the nearest pair of reference meshes for each weight $w_i \in \mathbf{w}$ and zero for all others. Equation (3) gives an exact solution at the reference meshes and an interpolation of the nearest reference meshes elsewhere. Recursive bisection subdivision (a new subdivision is created per subdivision) of the weight space $\mathbf{w}$ is performed to evaluate a set of nonlinearly 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.

3.1.3 Evaluation of Hybrid Nonlinear Blending

Hybrid nonlinear mesh blending allows accurate approximation of nonlinear mesh deformation while maintaining the computational performance of linear blending to allow real-time interactive animation. Fig. 3 presents a comparison of errors for linear blending with the proposed hybrid nonlinear 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 nonlinear mesh blending approach achieves accurate approximation even with a single intermediate nonlinear displacement map (second row), whereas linear blending results in large errors (top).

Fig. 4 characterizes the representation error and storage cost against number of subdivisions for different error thresholds ϵ. Fig. 4b shows that there is a relatively small error reduction for thresholds below 5 mm while there the memory usage increases significantly. This is caused by the $5\ \mathrm{mm}$ resolution of the original database such that details below this level are not reconstructed.

TABLE 1
Maximum Vertex Displacement Error with Respect to Nonlinear Blending as a Percentage of Model Size for in Meshes in Fig. 3

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*

3.1.4 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 parameterization 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 $\mathbf{p}$ [35].

In this work, an inverse mapping function is obtained by evaluation of the parameter value for a weights w in the range $[0, 1]$ with step size $\Delta w = 0.05$ and fitting a low-order polynomial to parameterise the inverse relationship [43]. In practice, a quadratic polynomial has been found to accurately parameterise the relationship between user controlled high-level motion parameters (speed, distance, direction, and so on.) and the weights.

3.2 Motion Transitions

Transitions between parameterized 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 nonrigid 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.

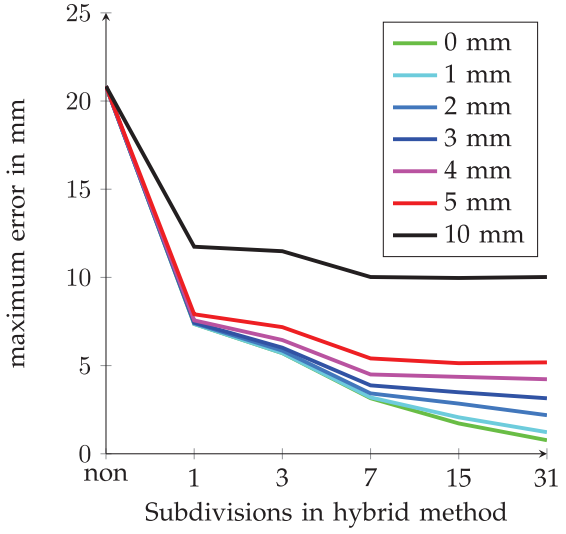
Previous parametric motion graphs [4] based on skeletal motion sequences precomputed a discrete set of good transitions by evaluating the similarity in pose and motion between pairs of source and target points. To limit the memory required a fixed number of good transitions are stored and interpolated at runtime. Precomputation of a fixed set of transitions can result in a relatively high latency due to the delay between the current state and next precomputed good transition.

We introduce an alternative approach, which does not require the precomputation of a discrete set of transitions. Optimal transitions are computed at runtime 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 precomputed fixed set allowing the best transition for a given starting point to be evaluated at runtime; and transition points can be evaluated on the fly to minimize latency while ensuring a smooth transition.

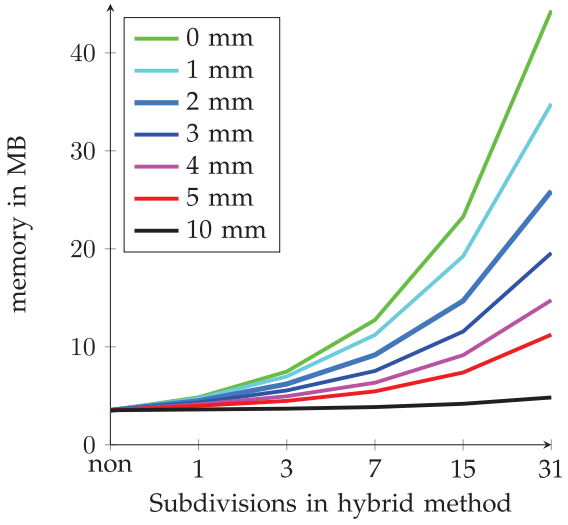
A parameterized motion is defined as a mesh sequence $M^r(t^r, \mathbf{p}^r)$ where $\mathbf{p}^r \in \Re^{N^r}$ is the set of user controlled parameters for the rth motion class. Each parameterized motion $M^r(t^r, \mathbf{p}^r)$ is computed from a set of 4D mesh sequences $\{M_i^r(t)\}_{i=1}^{N^r}$ according to (3). 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 runtime. To evaluate the best transition path P_{opt} online we optimize a cost function representing the tradeoff 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 $M^t(t^t, \mathbf{p}^t)$:

$$P_{opt} = \arg\min_{P \in \Omega}(E_S(P) + \lambda E_L(P)), \tag{4}$$

where λ defines the tradeoff between transition similarity and latency ($\lambda = 5$ throughout this work). The transition path P is optimized over a trellis of frames starting at the



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



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

Fig. 4. Evaluation of the maximum error and memory usage of the hybrid blending method, for the walk-run parameterized motion.

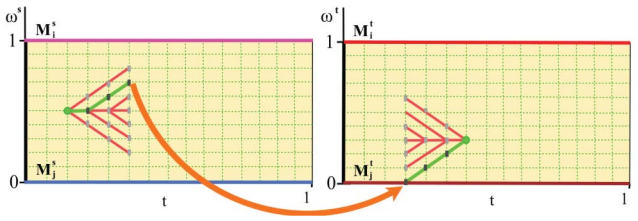


Fig. 5. Illustration of the transition between a parameterized motion space $source$ created by the sequences M_i^s (pink) and M_j^s (blue), and a parameterized motion space $target$ created by sequences M_i^t (red) and M_j^t (brown). A trellis linking the current location on the $source$ parametric space and the $target$ location is built, shown in red, with the candidate transition frames, shown in gray. The green path represents the optimal path P_{opt}. Orange arrow shows the actual transition. Motions used to create a parametric space are depicted in Fig. 1.

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 Fig. 5. 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 give 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 normalizing 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 precomputed similarity between the mesh sequences used to build the source and target motion spaces.

To achieve real-time evaluation of the optimal transition with accurate approximation of the true similarity we introduce a nonlinear bisection approach analogous to the hybrid mesh blending presented in Section 3.1.2. Source and target motion parameter spaces are subdivided with offline nonlinear interpolation of intermediate meshes. Real-time online computation of the similarity between motion spaces is then performed by linear interpolation of the similarity. Bisection subdivision allows accurate approximation of the true nonlinear similarity. In practice, a single bisection gives an accurate approximation for path optimization as discussed in Section 3.2.4. As in the hybrid mesh blending approach, the price paid is a small precomputation step and a modest increase of memory usage. The proposed approach enables the computation of mesh similarities with significantly lower errors than using the linear method, while maintaining the online performance.

3.2.2 Optimal Transition Path Evaluation

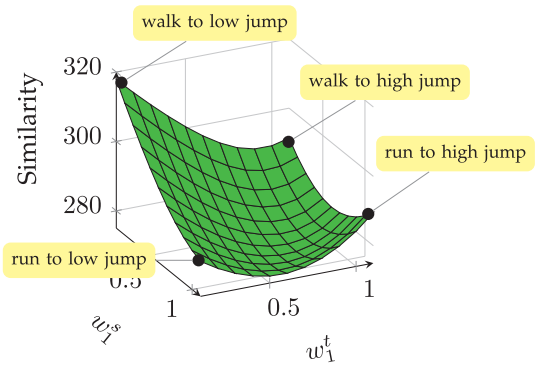
As explained in Section 3.1.2, from a set of captured mesh sequences $\{\mathbf{M}_p(t)\}_{p=1}^N$ defining a parametric motion space we can precompute interpolated mesh sequences $M_{NL}(t, \mathbf{w}) = b(\mathbf{M}, \mathbf{w})$, where $b()$ is a nonlinear blend function and $\mathbf{w}$ the blending weights. Bisection subdivision of the parametric motion space gives a set of reference mesh sequences $\{M_R(t, w_p)\}_{p=1}^{N_R}$ composed of the set of captured mesh sequences $\{M_p(t)\}_{p=1}^N$ and a nonlinear set $\{M_{NL}(t, w_p)\}_{p=1}^{N_{NL}}$ using a blending weight set $\{\mathbf{w}_p\}_{p=1}^{N_{NL}}$ where $N_R = N + N_{NL}$.

To evaluate the transition cost between a source and target parametric motion space, we precompute the shape and motion similarity between each pair of source and target reference mesh sequences. For each pair of source $M_R^s(t_u, \mathbf{w_p})$ and target $M_R^t(t_v, \mathbf{w_q})$ mesh sequences, we evaluate the shape and motion similarity $s(M_R^s(t_u, \mathbf{w_p}), M_R^t(t_v, \mathbf{w_q}))$ for all frames $t_u \in [0, T_s], t_v \in [0, T_t]$ giving a matrix $\mathbf{S}_{pq}$ of size $T_s \times T_t$. Precomputing the similarity between all pairs of source and target mesh sequences gives a similarity matrix $\mathbf{S}$ of size $N_R^s T_s \times N_R^t T_t$, where N_R^s and N_R^t are the number of mesh sequences in the source and target motion space, respectively.

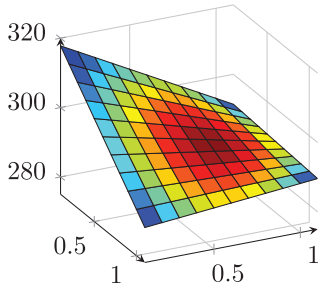
Online real-time computation of the shape similarity between the mesh in the source $M^s(t^s, \mathbf{p}^s)$ and target $M^t(t^t, \mathbf{p}^t)$ spaces for any source $\mathbf{p}^s$ and target $\mathbf{p}^t$ parameter value can then be evaluated as a weighted sum of the similarity of the corresponding pairs of reference meshes in source $M_R^s(t_u, \mathbf{w}_p^s)$ and target $M_R^t(t_v, \mathbf{w}_q^t)$ where $\mathbf{w} = h^{-1}(\mathbf{p}) \in [0, 1]$. For convex weights it can be shown that the following inequality holds (see supplementary material for proof, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.314):

$$\begin{aligned} & s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t)) \\ & \quad \geq \sum_{p=1}^{N^s} \sum_{q=1}^{N^t} \mathbf{w}_p^s \mathbf{w}_q^t s(M_R^s(t^s, \mathbf{w}_p), M_R^t(t^t, \mathbf{w}_q)) \\ & \quad \geq \mathbf{w}^s \mathbf{S}^\top (\mathbf{w}^t)^\top. \end{aligned} \tag{5}$$

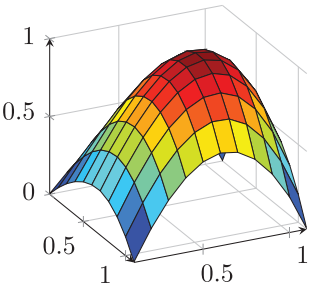
Thus, a lower bound on the similarity between any two frames in the parameterized source and target motion space can be evaluated by a sum of the weighted similarity between the source and target mesh sequences. As the pairwise sequence similarities can be precomputed, evaluation of the maximum lower bound on similarity between any point in the two parameterized spaces can be efficiently computed at runtime. If $\mathbf{w}^s$ and $\mathbf{w}^t$ are the source and target weight vectors, then we can evaluate the maximum similarity according to (5). Bisection subdivision of the source and target parametric motions spaces to give a set of nonlinearly interpolated reference meshes results in a piecewise linear approximation of the nonlinear similarity, which can be efficiently evaluated at runtime.



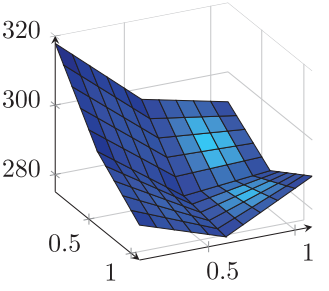
(a) Non-linear similarities



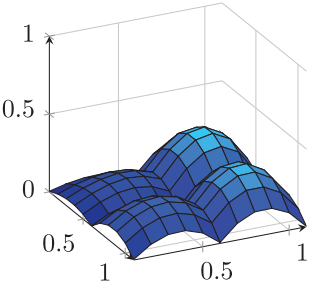
(b) Approximation of the similarities using a linear approach



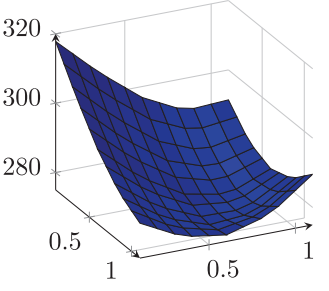
(c) Normalised geometric representation of the linear errors



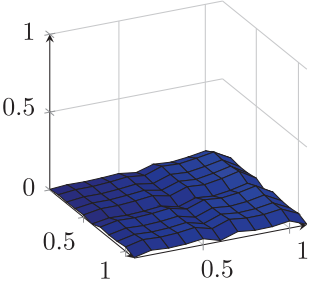
(d) Approximation using the proposed approach with $\mathbf{w} = \{0, 0.5, 1\}$



(e) Normalised geometric representation of the errors of (d)



(f) Approximation using the proposed approach with $\mathbf{w} = \{0, 0.25, 0.5, 0.75, 1\}$



(g) Normalised geometric representation of the errors of (f)

Fig. 6. Influence of the number of subdivisions in the weight space $\mathbf{w}$ on the error in similarity computation. The two mesh parametric spaces being compared have been built from a walk and a run motions, and from a high jump and low jump motions. $t^s = 0.05$ and $t^t = 0.54$, $t \in [0, 1]$. Errors and computational time are presented in Table 3.

The optimal transition path P_{opt} for a given source frame (current state) and target frame (end state) is evaluated according to (4) at runtime by evaluating the cost for all paths in the trellis defining possible transitions between the source and target motion spaces. The similarity cost $E_S(P)$ for path P is defined as the similarity at the transition frames between the source and target motion spaces evaluated according to (5): $E_S(P) = s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t))$. The optimal path P_{opt} can be evaluated with low latency at runtime because computation of the similarity cost by interpolation, (5), is computationally efficient. Furthermore, to speed up the computation of P_{opt}, Dijkstra's shortest path algorithm is used, approximately halving the required time to find P_{opt}.

3.2.3 Mesh Similarity Evaluation

As stated in Section 3.2.1, a matrix $\mathbf{S}$ containing the similarities $s(M_R^s(t_u, \mathbf{w_p}), M_R^t(t_v, \mathbf{w_q}))$ is precomputed offline and used at runtime to approximate any similarity $s(M^s(t^s, \mathbf{p}^s), M^t(t^t, \mathbf{p}^t))$. The approximation error depends on the weights in $\mathbf{w_p}$ and $\mathbf{w_q}$, which are based on a recursive bisection subdivision of the weight space.

Fig. 6 shows the influence of the number of subdivisions in the weight space $\mathbf{w}$ on the final performance of the proposed approach. Fig. 6a characterizes the ground-truth linear similarities at time $t^s = 0.08$ and $t^t = 0.54$ between a parametric space built from walk and run motions and a parametric space built from low jump and high jump

TABLE 2
Comparison of Transition Cost, Latency and Computation Time for Precomputed Fixed Transitions with the Proposed Online Computation($\lambda = 5$, Frame Rate $= 0.04$ s)

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

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.

motions. Fig. 6b is the result of approximating these similarities using a linear approximation, colored as a heat map to highlight the differences with respect to the nonlinear approach. Figs. 6d and 6f characterize the similarities obtained using one subdivision and two subdivisions of the weight space, respectively. Figs. 6c, 6e, and 6g show the error (difference between hybrid linear approximation and ground-truth nonlinear similarity) normalized to the range [0,1]. Note that even with a single bisection, Fig. 6e, the maximum error in similarity computation is <33 percent and with two bisections, Fig. 6g, the maximum error in similarity is <10 percent.

Table 3 shows the computational time and quantitative errors for figures of Fig. 6. Notice how the error with respect to the nonlinear case decreases significantly using the proposed approach with one subdivision ($\mathbf{w} = \{0, 0.5, 1\}$). Using two recursive subdivisions ($\mathbf{w} = \{0, 0.25, 0.5, 0.75, 1\}$) the error is not significant in the computation of the best transition. Real-time online computational performance is maintained with the evaluation of all similarities for a trellis of depth 10 taking less than 9 ms for all subdivision levels. These quantitative results demonstrate that the proposed approach gives accurate approximation of the nonlinear similarity between parametric motion spaces while allowing computationally efficient evaluation for real-time motion control. The price paid is precomputation of the reference meshes and evaluation of pairwise similarities that need to be stored in memory. Throughout this work we have used two subdivisions to accurately approximate the similarities between the different nodes of a 4D Parametric Motion Graph.

3.2.4 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 optimization achieves lower latency (delay in transition) for a similar transition cost, while maintaining real-time performance for evaluation of transitions (<0.2 ms/transition). Precomputation of fixed transitions gives a marginally lower similarity as it selects the best transition points, but results in significantly higher latency.

Fig. 7 shows a qualitative comparison of the same transitions using six different configurations (results are also presented in the supplementary[1] video). Figs. 7a and 7b show results with one and five precomputed transitions, respectively. This demonstrates the relatively high latency with a large number of transition frames (pink). Figs. 7c, 7d, 7e, and 7f show the performance of the proposed online path optimization 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 optimization with a trellis depth of 12 and 10 as shown in Figs. 7c and 7d). Figs. 7e and 7f 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 optimization 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

Data sets used in this work are reconstructed from multiple view video capture of actor performance with eight HD cameras equally spaced in a circle and capture volume $5\ \mathrm{m}^2 \times 2\ \mathrm{m}$. Reconstruction is performed using multiview stereo [8] followed by temporal alignment [19], [21] 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 nonlinear interpolation which gives accurate approximation. Four-dimensional parametric motion graphs are constructed using 10 sequences (walk, run, left turn, right turn, reach low/high, jump low/high, and jump short/long) of 20-80 frames each with a 3,000 vertex mesh. Total size of the input mesh database is 52 MB.

Examples of interactive character animation from 4D performance capture are shown in Figs. 8a, 8b, and 8e). Meshes are colored 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 >50 Hz with all parameter control and transitions evaluated on a single CPU. Results demonstrate that the 4D parametric motion graph

TABLE 3
Computational Time (Offline and Online) and Errors of the Proposed Approach Presented in Section 3.2.3 to Approximate the Computation of Mesh Similarities for All Possible Transition with a Trellis Depth $= 10$

Method	offline time	online time	ave. error	max. error
Non-linear	0.000 sec	160.583 sec	0.000	0.000
Linear	39.698 sec	0.009 sec	10.05	17.61
1 subdivision	78.087 sec	0.009 sec	2.55	5.88
2 subdivisions	238.023 sec	0.009 sec	0.64	1.74

1. Link: http://cvssp.org/projects/4d/parametricmotiongraphs/tvcg2012/.

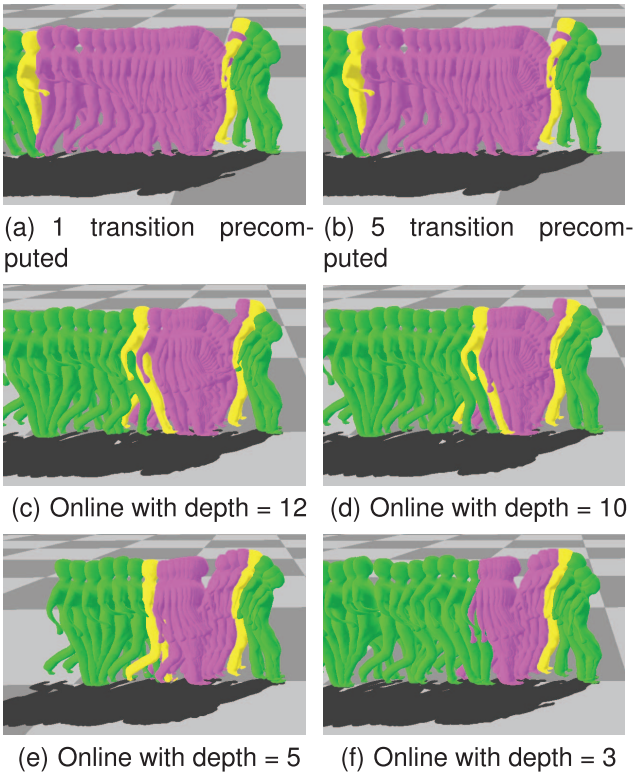


Fig. 7. Comparison of the same transition between a run motion $\alpha = 0.9$ of the parametric space showed in Fig. 2a to a jump motion $alpha = 0.5$ of the parametric space showed in Fig. 2b. In yellow, source $M^s(t^s, \mathbf{p}^s)$ and target $M^t(t^t, \mathbf{p}^t)$ meshes. In pink, the meshes computed to transition.

enables interactive control with smooth variation in character motion in response to changes in high-level parameters. Fig. 8c shows another application for real-time character animation where the user interactively sets a path defined by a set of contiguous Bezier 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 parameterization and the parametric surface motion graph representation enable real-time interaction with high-level movement control. Limitations of the current implementation include:

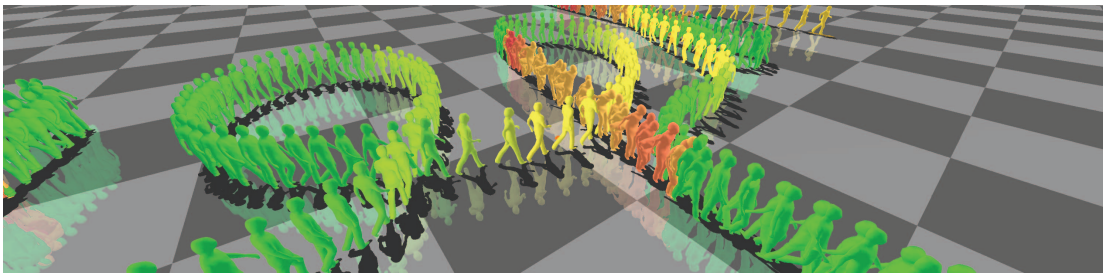
- four-dimensional performance capture: Although not addressed in this paper current reconstruction of actor performance and subsequent temporal alignment into a database of sequences achieves realistic results but contains errors. Average errors are in the order of 1-2 cm rms for reconstruction and alignment but may be larger for rapid movement where large interframe 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 to improve the reconstruction and alignment to allow animation without visible artefacts.
- Mesh sequence parameterization: A hybrid approach to approximate nonlinear blending while maintaining real-time performance is proposed for parameterization of multiple sequences of related motions. This combines the realistic mesh deformation of nonlinear approaches [27], [38], [39], [28] with the efficiency of linear blending and is demonstrated to give an accurate approximation with real-time performance (< 10 ms/frame). This approach requires a small number of additional intermediate nonlinear displacement maps to be precomputed for each parameterized motion.
- Transitions in high-dimensional parametric spaces: Online computation of transitions between motion spaces with up to three parameters can be performed in real-time (< 0.2 ms). Online optimization 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 precomputation of transitions costs could be employed at the expense of increased storage and fixed transitions locations leading larger latency.

6 Conclusion

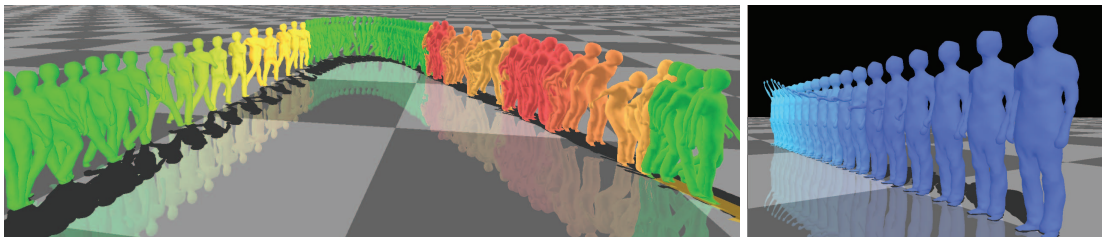
A system for real-time interactive character animation from 4D actor performance capture has been presented. The approach is based on a database of temporally aligned mesh sequence reconstructions of an actor performing multiple motions. Four-dimensional parametric motion graphs are introduced enabling 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 nonlinear blending [27], [38], [39], [28] with the real-time performance of linear blending. Shape similarity between parameterized meshes is computed through an approach that approximates the nonlinear similarity values but maintains the real-time performance. Natural transitions between different parametric motion spaces are evaluated in real time through a transition path optimization based on shape, nonrigid motion similarity, and latency. Online transition optimization from the current source state is demonstrated to reduce latency while maintaining real-time performance. This representation allows interactive animation from mesh sequences, which is analogous to approaches previously introduced skeletal data [4], [5], [2], [1]. Results for a CPU-based implementation of the parametric control and motion transitions demonstrate a frame-rate >50 Hz for a mesh of 3,000 vertices. Four-dimensional 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.

Acknowledgments

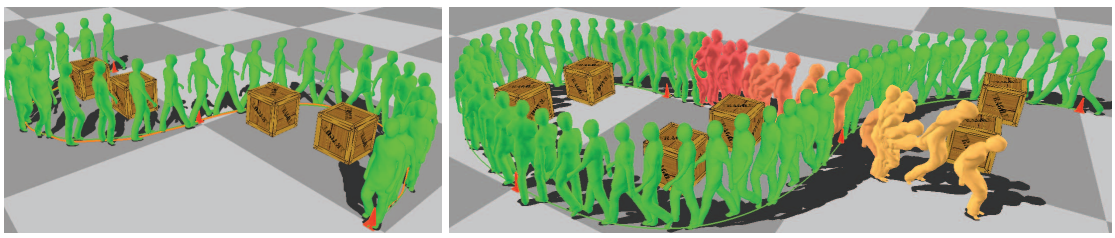
This research was supported by EU IST Project RE@CT and United Kingdom EPSRC Visual Media Platform Grant.



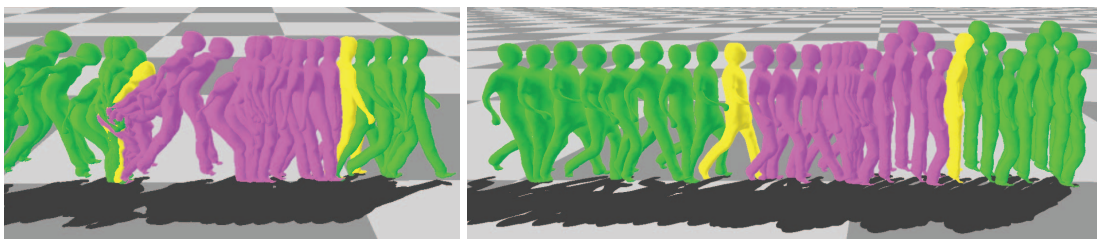
(a) Real-time interactive character animation with a 4D parametric motion graph



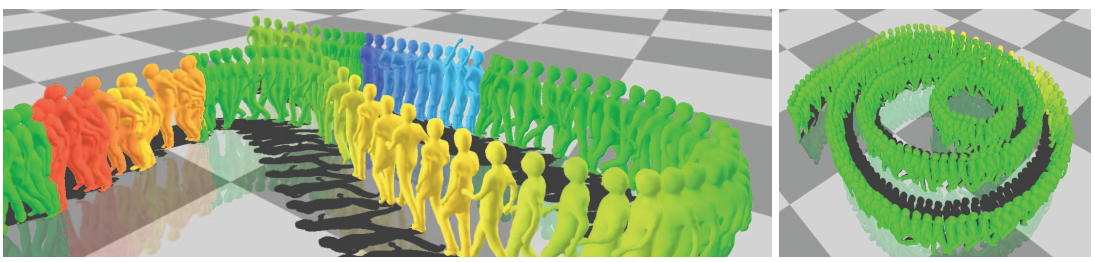
(b) Interactive parametric control of jump and reach motions



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



(d) Left: Transition between a long jump motion and a walk motion. Right: Transition between a run motion and a low jump. In yellow, the source and target poses requested by the user. In pink the meshes automatically generated to generate the transition.



(e) Left: A character interactively controlled combines walk, run, reach and jump. Right: Any turn is possible!

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

REFERENCES

[1] O. Arikan and D. Forsyth, "Synthesizing Constrained Motions from Examples," *Proc. ACM SIGGRAPH '02*, 2002.

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion Graphs," *Proc. ACM SIGGRAPH '02*, pp. 473-482, 2002.

[3] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive Control of Avatars Animated with Human Motion Data," *Proc. ACM SIGGRAPH '02*, pp. 491-500, 2002.

[4] R. Heck and M. Gleicher, "Parametric Motion Graphs," *Proc. ACM Symp. Interactive 3D Graphics*, 2007.

[5] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and Adverbs: Multidimensional Motion Interpolation," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32-40, Sept. 1998.

[6] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance Capture from Sparse Multi-View Video," *Proc. ACM SIGGRAPH '08*, vol. 27, no. 3, 2008.

[7] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated Mesh Animation from Multi-View Silhouettes," *Proc. ACM SIGGRAPH '08,* 2008.
[8] J. Starck and A. Hilton, "Surface Capture for Performance-Based Animation," *IEEE Computer Graphics and Applications,* vol. 27, no. 3, pp. 21-31, May/June 2007.
[9] F. Xu, Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.-P. Seidel, J. Kautz, and C. Theobalt, "Video-Based Characters - Creating New Human Performances from a Multi-View Video Database," *Proc. ACM SIGGRAPH '11,* 2011.
[10] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-Quality Video View Interpolation Using a Layered Representation," *Proc. ACM SIGGRAPH '04,* 2004.
[11] I. Baran, D. Vlasic, E. Grinspun, and J. Popovic, "Semantic Deformation Transfer," *Proc. ACM SIGGRAPH '09,* 2009.
[12] C. Stoll, J. Gall, E. de Aguiar, S. Thrun, and C. Theobalt, "Video-Based Reconstruction of Animatable Human Characters," *Proc. ACM SIGGRAPH ASIA '10,* 2010.
[13] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton, "4d Parametric Motion Graphs for Interactive Animation," *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games (I3D '12),* pp. 103-110, 2012.
[14] T. Kanade and P. Rander, "Virtualized Reality: Constructing Virtual Worlds from Real Scenes," *IEEE MultiMedia,* vol. 4, no. 1, pp. 34-47, Jan.-Mar. 1997.
[15] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel, "Free-Viewpoint Video of Human Actors," *Proc. ACM SIGGRAPH '03,* pp. 565-577, 2003.
[16] C. Cagniart, E. Boyer, and S. Ilic, "Free-Form Mesh Tracking: A Patch-Based Approach," *Proc. IEEE Conf. Computer Vision and Pattern Recognition '10,* pp. 1339-1346, 2010.
[17] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-Dimensional Scene Flow," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 3, pp. 475-480, Mar. 2005.
[18] M. Wand, B. Adams, M. Ovsianikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling, "Efficient Reconstruction of Non-Rigid Shape and Motion from Real-Time 3D Scanner Data," *ACM Trans. Graphics,* vol. 28, no. 2, article 15, 2009.
[19] P. Huang, C. Budd, and A. Hilton, "Global Temporal Registration of Multiple Non-Rigid Surface Sequences," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition '11,* 2011.
[20] T. Tung and T. Matsuyama, "Dynamic Surface Matching by Geodesic Mapping for Animation Transfer," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition '10,* 2010.
[21] C. Budd, P. Huang, M. Klaudiny, and A. Hilton, "Global Non-Rigid Alignment of Surface Sequences," *Int'l J. Computer Vision,* pp. 1-15, http://dx.doi.org/10.1007/s11263-012-0553-4, 2012.
[22] P. Huang, A. Hilton, and J. Starck, "Human Motion Synthesis from 3d Video," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition '09,* 2009.
[23] J. Starck, G. Miller, and A. Hilton, "Video-Based Character Animation," *Proc. ACM Symp. Computer Animation,* 2005.
[24] C. Bregler, M. Covell, and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," *Proc. ACM SIGGRAPH '97,* pp. 1-8, 1997.
[25] A. Schodl, D. Szeliski, R. amd Salesin, and I. Essa, "Video Textures," *Proc. ACM SIGGRAPH '00,* 2000.
[26] M. Flagg, A. Nakazawa, Q. Zhang, S.-B. Kang, Y. Ryu, I. Essa, and J. Rehg, "Human Video Textures," *Proc. ACM Symp. Interactive 3D Graphics,* 2009.
[27] S. Kircher and M. Garland, "Free-Form Motion Processing," *ACM Trans. Graphics,* vol. 27, no. 2, pp. 1-13, 2008.
[28] W. Xu, K. Zhou, Y. Yu, Q. Peng, and B. Guo, "Gradient Domain Editing of Deforming Mesh Sequences," *Proc. ACM SIGGRAPH '07,* vol. 26, no. 3, 2007.
[29] M. Gleicher, "Motion Editing with Spacetime Constraints," *Proc. ACM Symp. Interactive 3D Graphics,* 1997.
[30] J. Lee and S. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures," *Proc. ACM SIGGRAPH '99,* pp. 39-48, 1999.
[31] Z. Popovic and A. Witkin, "Physically Based Motion Transformation," *Proc. ACM SIGGRAPH '99,* 1999.
[32] J. Min, Y.-L. Chen, and J. Chai, "Interactive Generation of Human Animation with Deformable Motion Models," *ACM Trans. Graphics,* vol. 29, no. 1, pp. 9:1-9:12, Dec. 2009.
[33] A. Brundelin and L. Williams, "Motion Signal Processing," *Proc. ACM SIGGRAPH '95,* pp. 97-104, 1995.
[34] D. Wiley and J. Hahn, "Interpolation Synthesis for Articulated Figure Motion," *Proc. IEEE Virtual Reality Int'l Symp.,* pp. 157-160, 1997.
[35] L. Kovar and M. Gleicher, "Automated Extraction and Parameterization of Motions in Large Date Sets," *Proc. ACM SIGGRAPH '04,* vol. 23, no. 3, pp. 559-568, 2004.
[36] T. Mukai and S. Kuriyama, "Geostatistical Motion Interpolation," *Proc. ACM SIGGRAPH '05,* 2005.
[37] L. Zhao and A. Safonova, "Achieving Good Connectivity in Motion Graphs," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA'08),* pp. 127-136, 2008.
[38] O. Sorkine, "Differential Representations for Mesh Processing," *Computer Graphics Forum,* vol. 25, no. 4, pp. 789-806, 2006.
[39] R. Sumner and J. Popovic, "Deformation Transfer for Triangle Meshes," *Proc. ACM SIGGRAPH '04,* vol. 23, no. 3, pp. 399-405, 2004.
[40] A. Witkin and Z. Popovic, "Motion Warping," *Proc. ACM SIGGRAPH '95,* 1995.
[41] C. Budd and A. Hilton, "Temporal Alignment of 3d Video Sequences Using Shape and Appearance," *Proc. European Conf. Visual Media Production (CVMP '10),* pp. 114-122, 2010.
[42] M. Tejera and A. Hilton, "Space-Time Editing of 3d Video Sequences," *Proc. Conf. Visual Media Production,* pp. 148-157, http://dx.doi.org/10.1109/CVMP.2011.23, 2011.
[43] A. Ahmed, F. Mokhtarian, and A. Hilton, "Parametric Motion Blending through Wavelet Analysis," *Proc. Eurographics '01 Short Paper,* pp. 347-353, Sept. 2001.

Dan Casas received the MEng degree in computer science from Universitat Autònoma de Barcelona, Spain, in 2009, and is working toward the PhD degree at the Centre for Vision, Speech and Signal Processing, University of Surrey, United Kingdom. His research interests include character animation, motion synthesis, and free-viewpoint video.

Margara Tejera received the master's degree in telecommunication engineering from the University of Seville, Spain, and is working toward the PhD degree at the Centre for Vision, Speech, and Signal Processing, University of Surrey, United Kingdom. Her current research work aims to generate algorithms and tools that allow the synthesis of novel 3D video sequences.

Jean-Yves Guillemaut received the MEng degree from the Ecole Centrale de Nantes, France, in 2001, and the PhD degree from the University of Surrey, United Kingdom, in 2005. He is currently a lecturer at the Centre for Vision, Speech and Signal Processing, University of Surrey, United Kingdom. His research interests include free-viewpoint video and 3D TV, image/video-based scene reconstruction and rendering, image/video segmentation and matting, camera calibration, and active appearance models for face recognition. He is a member of the IEEE.

Adrian Hilton received the BSc (hons.), DPhil, and CEng degrees. He is a professor of Computer Vision and the director of the Centre for Vision, Speech and Signal Processing at the University of Surrey, United Kingdom. His research interests include robust computer vision to model and understand real-world scenes. His contributions include technologies for the first hand-held 3D scanner, modeling of people from images and 3D video for games, broadcast, and film. He currently leads research investigating the use of computer vision for applications in entertainment content production, visual interaction, and clinical analysis. He is a member of the IEEE.