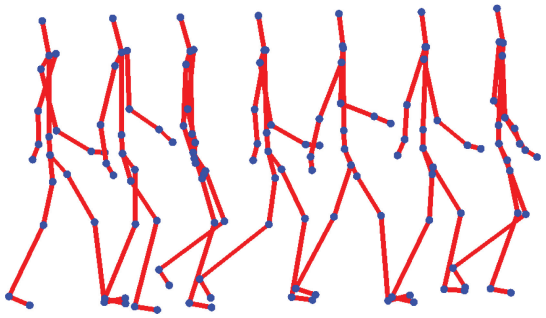
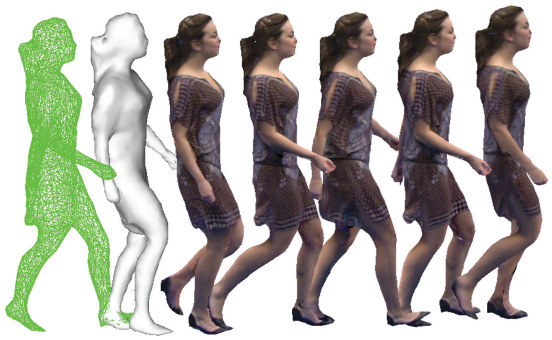
Hybrid Skeletal-Surface Motion Graphs for Character Animation from 4D Performance Capture

PENG HUANG, MARGARA TEJERA, JOHN COLLOMOSSE, and ADRIAN HILTON
University of Surrey





We present a novel hybrid representation for character animation from 4D Performance Capture (4DPC) data which combines skeletal control with surface motion graphs. 4DPC data are temporally aligned 3D mesh sequence reconstructions of the dynamic surface shape and associated appearance from multiple-view video. The hybrid representation supports the production of novel surface sequences which satisfy constraints from user-specified key-frames or a target skeletal motion. Motion graph path optimisation concatenates fragments of 4DPC data to satisfy the constraints while maintaining plausible surface motion at transitions between sequences. Space-time editing of the mesh sequence using a learned part-based Laplacian surface deformation model is performed to match the target skeletal motion and transition between sequences. The approach is quantitatively evaluated for three 4DPC datasets with a variety of clothing styles. Results for key-frame animation demonstrate production of novel sequences that satisfy constraints on timing and position of less than 1% of the sequence duration and path length. Evaluation of motion-capture-driven animation over a corpus of 130 sequences shows that the synthesised motion accurately matches the target skeletal motion. The combination of skeletal control with the surface motion graph extends the range and style of motion which can be produced while maintaining the natural dynamics of shape and appearance from the captured performance.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

General Terms: Algorithms

Additional Key Words and Phrases: Human motion synthesis, motion graphs, surface motion capture, 3D video, 4D performance capture, example-based animation, video-based rendering

ACM Reference Format:

Peng Huang, Margara Tejera, John Collomosse, and Adrian Hilton. 2015. Hybrid skeletal-surface motion graphs for character animation from 4d performance capture. ACM Trans. Graph. 34, 2. Article 17 (February 2015), 14 pages.
DOI: http://dx.doi.org/10.1145/2699643

1. INTRODUCTION

Motion Capture (MoCap) has become a fundamental tool for authoring of 3D character animation for games and film. MoCap technology provides a convenient way to retarget human skeletal motion to characters, conveying a sense of movement realism that is both difficult and labour-intensive to animate manually. However, skeletal MoCap does not capture surface dynamics; it cannot capture wrinkles in clothing or the motion of hair. These performance details are critical to realism, but are still created manually within character models by highly skilled artists and animators.

Recently, 4D Performance Capture (4DPC) has been introduced to capture shape, appearance, and motion of the human body from multiview video. The outcome of 4DPC is a sequence of reconstructed 3D meshes capturing detailed surface dynamics plus associated video that can be projected onto the mesh to achieve video-quality realism. Furthermore, 4DPC meshes have temporally consistent vertices and topology, making them compatible with traditional animation pipelines. By simultaneously capturing both motion and appearance, 4DPC offers unique advantages over skeletal MoCap for the capture of 3D character models. However, there is limited tool support and limited reusable capture data available to enable the use of 4DPC models in character animation.

This article addresses these limitations, presenting a novel data-driven approach for animating 4DPC character models. We make the following technical contributions.

This research was supported by the EU-FP7 project RE@CT and the EPSRC Platform Grant EP/F02728X.
Authors' addresses: P. Huang (corresponding author), M. Tejera, J. Collomosse, and A. Hilton, University of Surrey, Guildford, Surrey GU2 7XH, UK; email: p.huang@surrey.ac.uk.

DOI: http://dx.doi.org/10.1145/2699643

First, we describe Surface Motion Graphs (SMGs): a representation and optimisation algorithm for creating novel animations from 4DPC. Animations are formed through concatenative synthesis: the seamless joining of captured performance fragments to produce new movements. The concept is analogous to Kovar et al.'s [2002] skeletal motion graphs that enable new movements to be sequenced from skeletal MoCap. However, a surface motion graph considers consistency of both surface shape as well as, implicitly, the pose. We describe both the construction of the surface motion graph and an optimisation algorithm that can incorporate high-level user constraints on timing, position, and motion to synthesise high-quality animated sequences. This extends earlier research on surface motion graphs [Starck et al. 2005; Huang et al. 2009] which used unaligned mesh sequences and required manual graph construction. In this article, 3D mesh sequences are first nonrigidly aligned [Budd et al. 2012] to produce a temporally coherent 4D sequence. This allows surface motion graph representation with smooth transitions between sequences and modification of the original motion using a learned deformation space.

Second, we describe a new algorithm to enable the use of skeletal MoCap data to drive the creation of animations under our surface motion graph framework. As an established technology, skeletal MoCap data is cheap and easy to obtain, whereas 4DPC data is currently scarce and reconstruction times are lengthy. We describe a pose retrieval technique for matching MoCap data to skeletal pose inferred from 4DPC. Spatio-temporal editing is applied to retrieved mesh sequences to accurately match the target motion using a learned part-based Laplacian surface deformation model which preserves the shape and motion characteristics of the 4DPC data. The ability to drive a high-fidelity 4DPC character model with readily available libraries of skeletal MoCap data extends the range of motion which can be produced and promises to accelerate the uptake of 4DPC for animation production.

A supplementary video, showcasing the animations produced by our system, accompanies this submission.

2. RELATED WORK

Our concatenative approach to character animation falls squarely within the domain of example-based synthesis (EBS). EBS was introduced in speech to allow reproduction of natural speech from a corpus of recorded spoken audio fragments [Hunt and Black 1996]. Subsequently EBS was exploited in computer graphics to reuse and modify video sequences. Bregler et al. [1997] introduced *video rewrite* to create a novel video of a person speaking by retrieving and concatenating mouth images from a training set via audio cues. Similarly, Ezzat and Geiger [2002] presented an audio-driven visual-speech animation system which also parametrises the mouth images, enabling generalisation beyond captured video frames using a morphable model.

Schödl et al.'s [2000] video textures extended video EBS to be driven for the first time by visual cues. They demonstrated the synthesis of perpetually realistic videos by copying and rearranging frames from a single source video. The video is modelled as a Markov process with each state corresponding to a single frame and the probabilities corresponding to the likelihood of transitions from one frame to another. These likelihoods are computed as frame-to-frame image similarities over a short temporal window. Later, Schödl and Essa [2002] extended this work to create character animations of a moving object, or *video sprite*, cropped from the video. In their work, a cost function was introduced to afford the user a degree of control over the final animation. However, complex or highly structured phenomena, such as 3D articulated full-body human motion, cannot be synthesised convincingly using a purely 2D video-based concatenative synthesis approach.

Kovar et al. [2002] construct a directed graph on skeletal motion capture sequences, referred to as a *motion graph*, where edges correspond to segments of motion and nodes identify connections between them. Motion segments include original motions and transition motions generated by blending segments together. Distances between pairs of frames are computed in pose space to determine if a transition is possible using a fixed similarity threshold. Synthesis is performed by finding an optimal graph walk that satisfies user-defined constraints. Gleicher et al. [2003] enhanced this approach for game applications by allowing a designer to interactively edit the graph structure.

Lee et al. [2002] extended the motion graph representation to a two-layer graph structure encoding possible links between different frames, which allows efficient search and interactive control. The similarity metric is again based on skeletal pose, but in terms of joint angles and velocities rather than spatial position of limbs as in motion graphs [Kovar et al. 2002]. Choice-based, sketch-based, and performance-based interfaces are demonstrated for interactive control of character motion. Similarly, Arikan and Forsyth [2002] employed a directed graph to connect motion segments where each node corresponds to a motion sequence and each edge a transition. Their similarity metric measures the distance between two frames in terms of the differences between corresponding joint positions and velocities, plus whole body velocities and accelerations. Hard and soft constraints, such as motion duration, or body position and orientation at a particular frame, are defined and a hierarchical randomised search, is used to generate motions. In Arikan et al. [2003] interactive motion annotation allows the user to include or exclude motion portions along the time line.

These approaches synthesise skeletal human motion successfully but, since skeletal MoCap does not capture nonrigid surface motion, they cannot reproduce detailed surface dynamics with the realism achievable by 4DPC. Our system extends the concept of a directed graph over skeletal MoCap to 4DPC surface capture data. Similarity is determined using 3D volumetric shape descriptors which were previously shown to achieve good performance for retrieval on 3D performance capture data [Huang et al. 2010]. To ensure accurate matching of both shape and appearance, the volumetric descriptor is extended in this work to include surface colour information. The use of motion graphs for animation from surface sequences was first introduced in Starck et al. [2005] but required manual graph construction. Huang et al. [2009] addressed automatic SMG construction based on shape similarity, allowing animation synthesis by concatenation of unaligned mesh sequences without video textures. In this work we present a complete framework for SMG representation and animation synthesis of shape and appearance from temporally aligned 4DPC data. Novel algorithms are proposed for graph construction and path optimisation for motion planning according to user-specified key-frame constraints or matching a target skeletal motion. This greatly extends the flexibility of animation, range of motion, and visual quality over previous SMG-based animation.

A further contribution of our system is content synthesis in one domain (4DPC) driven by motion from another (skeletal MoCap). In prior EBS work, transfer between the skeletal MoCap and 2D video domains has been explored. Hornung and Dekkers [2007] presented a method to animate photos of 2D characters using skeletal MoCap. Given a single image of a person or essentially human-like subject, the motion of a 3D skeleton is transferred onto the subject's 2D shape in the image space, generating a realistic movement. They reconstruct a projective camera model and a 3D model pose that best matches the given 2D image. A 2D shape template is

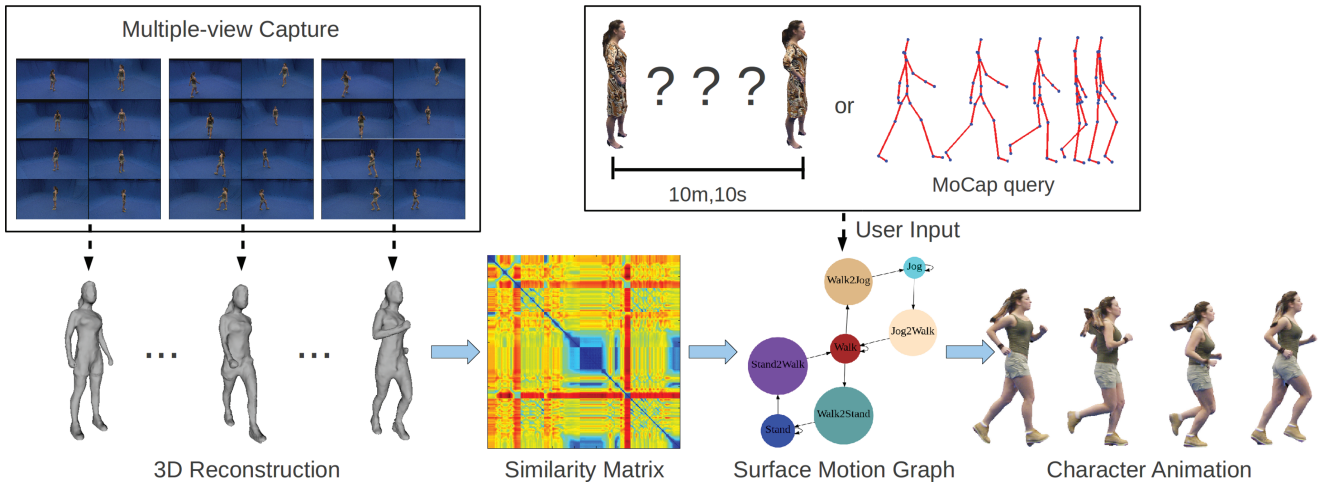


Fig. 1. Overview of the proposed 4DPC character animation pipeline (Section 3).

then fitted onto the character in the input image to drive the deformation according to projected 3D motion data. Flagg et al. [2009] also exploited the combination of 2D video and skeletal MoCap to generate controllable animations of human performance. 2D video and MoCap data are first synchronised and a video graph is then constructed to represent transitions between motion segments. The synthesis is performed via a random walk on the graph. The skeleton associated with each frame is used to blend images to create seamless transitions. Since this approach requires skeletal MoCap, the synthesis contains markers and is limited to human motions in tight clothing. James et al. [2007] proposed *mesh ensemble motion graphs* for interactive data-driven animation of dynamic mesh ensembles. This approach is focused on motion graph techniques for animating collections of objects. Integer programming is used to optimise asynchronous transitions to avoid mesh inter-penetration. In this article we also exploit integer programming for graph path optimisation, allowing optimisation of the number of loops for cyclic motions.

Xu et al. [2011] recently introduced an approach to synthesise 2D video sequences of human motion under user-defined motion and viewpoint constraints by sampling from a 4DPC database. They first exploit video-based skeletal performance capture [Gall et al. 2009] to acquire a database of an actor performing different motions, then retrieve similar frames in a database using pose similarity, and finally adopt a warping strategy in the 2D rendered view to synthesise video of novel motions. In this article, we present a framework to synthesise full 3D character animation from captured 4DPC data allowing free-viewpoint rendering and flexible user control of the character motion.

Previous research has also addressed example-based parametric motion control to increase the range of motion. Heck and Gleicher [2007] introduced *parametric motion graphs* allowing real-time interactive character control based on parameterisation of skeletal motion. Skeletal motion sequences of related motions are parameterised by interpolating between the sequences in joint angle space for example, parameterisation of a walk and run sequence according to speed and direction. The parametric motion graph representation enables transition between multiple parameterised motion spaces to synthesise novel animation sequences. Recent research [Casas et al. 2013] has investigated parametric control from example sequences of 4DPC data. Interactive character animation with multiple motion classes is demonstrated using a *4D parametric motion graph*. This approach allows animation from 4DPC but is limited to interpolation between example surface motion sequences. The approach presented in this article extends the range of motion outside the range of captured motions by using skeletal MoCap sequences to drive the animation while maintaining plausible surface deformation using a learned model. Previous work on space-time editing of mesh sequences [Tejera et al. 2013] used a learned global model to represent the surface shape which was restricted to reproducing poses similar to those in the captured motion. This article extends the range of motion by introducing a learned part-based model to represent the surface deformation, which enables flexible animation of novel poses and motions outside the 4DPC data.

3. CHARACTER ANIMATION PRODUCTION

Our character animation system processes 4D performance capture (4DPC) data through the pipeline presented in Figure 1.

4DPC Reconstruction. The first stage is data capture, in which synchronised HD video from multiple views is first captured with eight calibrated cameras. The cameras are spaced equally around a circle of 8 meters in diameter, at a height of 2.4 meters above the studio floor. A volumetric visual hull [Laurentini 1994] is computed from the silhouette of the performer in each view, obtained via chroma-key. Geometric detail is improved in the resulting mesh through stereo surface refinement — full details of these 3D reconstruction steps are reported in Starck and Hilton [2007]. The process yields an unstructured 3D mesh sequence in which meshes are not temporally consistent. Finally, global temporal registration [Budd et al. 2012] is used to obtain a single deformable mesh model for the captured performance, thus establishing temporal correspondence of vertices between meshes. The global alignment automatically aligns across a database of multiple sequences. This collection of aligned sequences is referred to as the 4DPC database. Our system is able to combine frames from a 4DPC database to synthesise new character animations.

Skeleton Extraction. To facilitate later retargeting of skeletal MoCap to the 4DPC sequence, we must estimate a skeletal pose for each frame. On a single frame, mesh vertices are manually marked to indicate the body parts to which they belong. Since after global

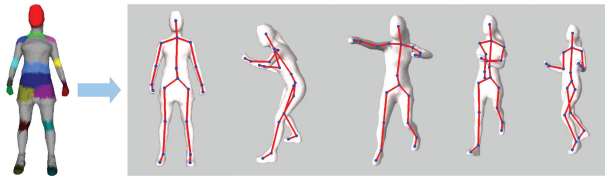


Fig. 2. Skeleton extraction. The user manually identifies different body parts on a single mesh. Vertex correspondence propagates markup to other frames, enabling extraction of a skeletal pose.

temporal registration all meshes share the same connectivity, this markup is propagated to all other frames. For each mesh, the centre of mass of each labelled vertex set is computed as a joint position and aggregated to form a skeleton. Figure 2 shows an example of a marked reference mesh, and the extracted skeletal pose for the reference and other frames.

Character Animation Pipeline. A frame-to-frame similarity matrix is first computed between all frames in the 4DPC database. For each frame, a spherical coordinate system is established around the mesh, and the space quantised into bins. The occupancy of each bin is used to compute a shape histogram that can be directly compared to those of other frames to score similarity. Adaptive temporal filtering is performed on the similarity to establish potential transition points between frames. A surface motion graph is then constructed using these transitions (Section 4). Once the graph structure is computed, character animation is produced according to user input such as key-frames (skeleton or surface pose), global timing, and distance constraints. MoCap sequences can also be used to retrieve similar motions from the surface motion graph. These motion planning algorithms are described in Section 5. Finally, view-dependent 4D video texture rendering [Casas et al. 2014] is used to combine the mesh sequence with captured multiview video, resulting in a video-realistic character animation as the final result.

4. SURFACE MOTION GRAPHS

A surface motion graph for a 4DPC database represents possible inter- and intra-sequence transitions, analogous to motion graphs [Kovar et al. 2002] for skeletal motion capture sequences.

4.1 6D Shape-Colour Similarity

To identify potential transitions, we must measure the similarity of each frame within the 4DPC database. To make sure transitions are seamless in terms of both geometry and appearance we use a similarity metric considering both 3D shape and associated texture colour. Volumetric shape histograms have previously been shown to achieve good performance for matching 3D surface shape and motion of performance capture sequences [Huang et al. 2010]. We extend this to a 6D shape-colour histogram $H(M)$ such that each bin represents both the spatial occupancy and colour appearance distribution for a given mesh M. We define a measure of shape-colour dissimilarity between two meshes M_r and M_s by optimising for the maximum overlap between their corresponding radial bins with respect to rotation about the vertical axis.

$$c(M_r, M_s) = \min_{\phi} \|H(M_r) - H(M_s, \phi)\|. \quad (1)$$

The shape-colour histogram $H(M)$ partitions the space that contains a 3D object into disjoint cells and counts the number of occupied voxels falling into each bin, together with their RGB colour distribution, to construct a 6D histogram as a signature for the object. The space is represented in a spherical coordinate system augmented with an RGB colour space $(r, \theta, \phi, R, G, B)$ located around the object's centre of mass. The number of bins for each dimension is chosen as (5, 10, 20, 10, 10, 10) respectively throughout this work. The Earth Mover's Distance (EMD) [Rubner et al. 1998] is used to compute the distance between the sparse 6D shape-colour histograms due to its suitability for use in high-dimensional sparsely occupied spaces. Eq. (1) is minimised by extending the computationally efficient approach for comparing 3D shape histograms [Huang et al. 2010] to the 6D space: a fine histogram is generated initially at an order of magnitude higher resolution than the desired vertical bin size; the fine histogram is then shifted by the fine bin size and rebinned to a coarse histogram for comparison. An illustration of the 6D shape-colour histogram is shown in Figure 3.

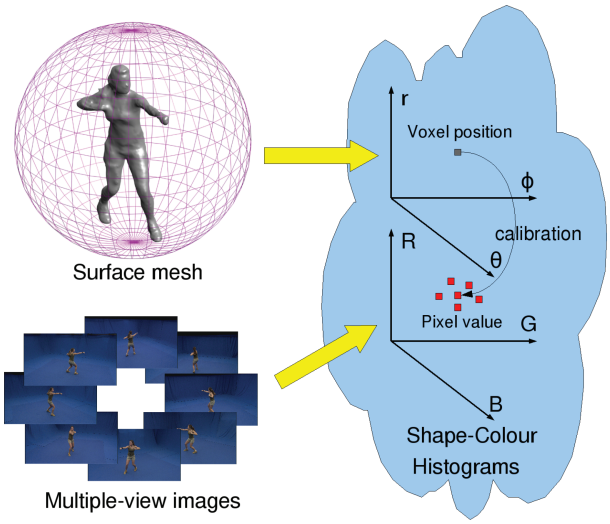


Fig. 3. An illustration of the shape-colour histograms.

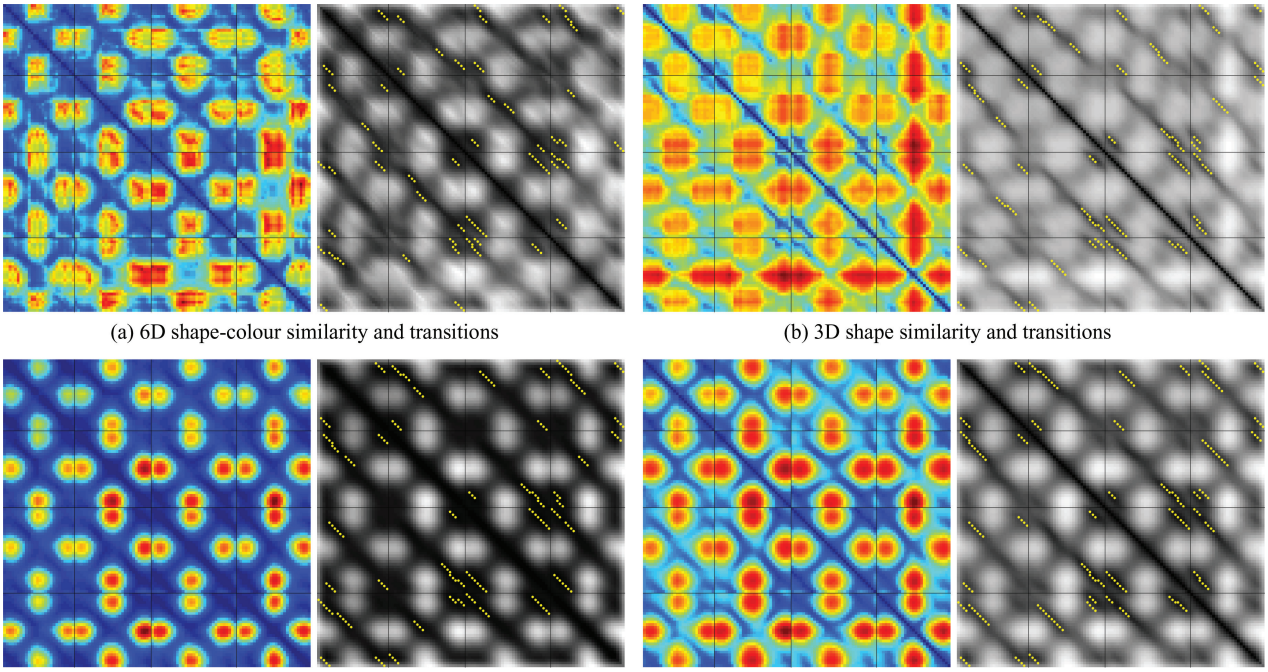
4.2 Transitions

A transition of the surface motion graph $S_{i\to j} = \{M_{i\to j}(t)\}$ from a 4DPC video sequence $S_i = \{M_i(t)\}$ to another 4DPC video sequence $S_j = \{M_j(t)\}$ is defined as an overlap of S_i and S_j. If we denote m, n as the central indices for the overlap, the length of overlap as $2L + 1$, the blending weight for the k^{th} transition frame is computed as $\alpha(k) = \frac{k+L}{2L}$, $k \in [-L, L]$. The k^{th} transition frame $M_{i\to j}(t_k) = G(M_i(t_{m+k}), M_j(t_{n+k}), \alpha(k))$ will be generated by a nonlinear blend described in Section 4.4.

Since the quality of a transition is determined by the distortion of blended to the original sequences, an optimal transition can be identified as the transtion which minimises this distortion. However, blended frames $M_{i\to j}(t)$ cannot be generated before transitions are identified, that is, $M_{i\to j}(t)$ is unknown. We approximate the distortion as the weighted 6D shape-colour dissimilarity (Eq. (1)) between frame $M_i(t)$ and $M_j(t)$, which are known. The distortion measure of a transition frame $M_{i\to j}(t_k)$ is defined as follows,

$$d(M_{i\to j}(t_k)) = \alpha'(k) \cdot c(M_i(t_{m+k}), M_j(t_{n+k})), \quad (2)$$

where m, n are the central indices for the overlap and $\alpha'(k) = \min(1 - \alpha(k), \alpha(k))$. The total distortion for a transition sequence $S_{i\to j}$ is then computed as the sum of the distortion of all transition



(a) 6D shape-colour similarity and transitions

(b) 3D shape similarity and transitions

(c) Euclidean vertex distance similarity and transitions

(d) skeleton similarity and transitions

Fig. 4. An example of static similarity matrix and identified transitions on temporal filtered similarity matrix for Character1between 4DPC mesh sequences of four motions "Jog", "Jog2Walk", "Walk" and "Walk2Jog".

frames,

$$D(S_{i\to j}) = \sum_{k=-L}^{L} d(M_{i\to j}(t_k)). \tag{3}$$

An optimal transition $S^{opt}_{i\to j}$ is then defined to minimise the distortion cost,

$$S^{opt}_{i\to j} = \arg\min_{S_{i\to j}} D(S_{i\to j}). \tag{4}$$

Adaptive Temporal Filtering. Each transition $S_{i\to j}$ is determined by a tuple (m, n, L). A frame-to-frame static similarity matrix between S_i and S_j can be precomputed: the shape similarity between r^{th} frame $M_i(t_r)$ from sequence S_i and s^{th} frame $M_j(t_s)$ from S_j is measured according to Eq. (1); this precomputation is performed only once through all frames and across all motion sequences in the 4DPC database. For simplicity, we denote the similarity matrix between S_i and S_j as $C = [c_{r,s}]_{N_r\times N_s}$, where $c_{r,s} = c(M_i(t_r), M_j(t_s))$, N_r, and N_s are the total number of frames for S_i and S_j, respectively. The global optimisation is then performed by testing all possible tuples (m, n, L) and so finding optimal arguments for the minimum,

$$(m^{opt}, n^{opt}, L^{opt}) = \arg\min_{m,n,L} \sum_{k=-L}^{L} \alpha'(k)\cdot c_{m+k,n+k}. \tag{5}$$

This equates to performing an adaptive temporal filtering with window size $2L+1$ and weighting $\alpha'(k)$ on the static similarity matrix C. In practice, candidates (pairs of m and n) are limited to all local minima of the similarity matrix and the window size is constrained by $L < 25$ (overlap is limited to be less than 1 second). To obtain multiple transitions between a pair of sequences, the top N best transitions are preserved. N is predetermined by the user. The process is efficient and the actual process time negligible (<100ms) with $N = 4$ for a pair of sequences.

This automatic transition identification overcomes the limitation of the approach presented in Huang et al. [2009] that required switching between sequences at the central frame due to the absence of temporal alignment. Their approach is also limited in allowing only one transition between each pair of motion sequences. In this work, the concatenation is performed by nonlinearly blending 3D mesh sequences that can produce smooth transitions. Multiple transitions (top N best transitions) are allowed between each pair of motion sequences, and the optimal window size L is computed for each top N best transitions.

Figure 4 shows an example of the frame-to-frame similarity matrix and associated transition frames (marked in yellow) evaluated using a temporal window $L = 4$ and $N = 4$ for four different similarity metrics: 6D shape-colour histogram; 3D shape-only histogram; Euclidean mesh vertex distance; and skeletal pose similarity. The 6D shape-colour histogram gives the best identification of transitions due to the use of both shape and colour information, as illustrated by the compact clusters of transition points and absence of transitions with large changes in appearance for similar shape. Shape only, Euclidean vertex distance, and skeletal pose similarity result in some lower-quality transitions with changes in appearance that may cause visible artefacts in the rendered animation. Further comparative evaluation of 3D shape descriptors for similarity computation is presented in Huang et al. [2010].

4.3 Motion Graph Construction

A surface motion graph representation is defined as a directed graph, where each node denotes a 4DPC video sequence and each edge denotes one or multiple possible transitions. Since transitions are

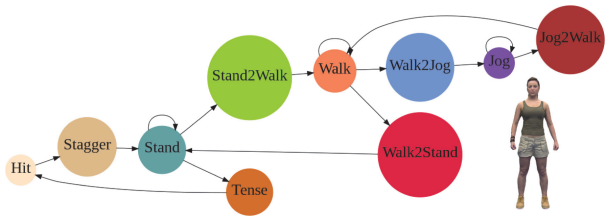


Fig. 5. An example of surface motion graph for a game character. Here, the user tries to create a surface motion graph for a game character, and other possible transitions, such as from "Stand2Walk" to "Walk2Stand", are removed.

automatically identified in Section 4.2, the graph construction is straightforward.

(1) *Initialization*. Insert all 4DPC video sequences as nodes into an empty graph.
(2) *Insert transitions*. If there is at least one transition from the source 4DPC video sequence to the target, then insert a directed edge from the corresponding source node to the target node.
(3) *Modification (optional)*. In general, between each pair of motion sequences, if they contain similar frames, an edge exists, but this edge may not be useful according to the user's need. Although the path optimisation will avoid them, the user can also edit the graph to remove unwanted transition edges to refine the graph.

Figure 5 shows an example of a surface motion graph constructed for a 4DPC of an actor dressed as a game character.

4.4 Nonlinear Blending

Previous concatenative motion synthesis/animation methods link different motion sequences together without any smoothing at transitions [Huang et al. 2009] as the surface correspondence is unknown. This may result in artefacts such as sudden change in surface shape and appearance. After global alignment, all frames share the same 3D mesh structure, that is, vertex correspondences are known. This section leverages this alignment to obtain smooth transitions by nonlinear blending of temporally aligned 4DPC frames both in geometry and appearance. This significantly reduces transition artefacts.

Blending Geometry and Skeletal Pose. Blending for a transition 3D mesh and skeleton pair sequence $S_b = \{(M_b(t_k), K_b(t_k))\}$ from a source sequence $S_s = \{(M_s(t_k), K_s(t_k))\}$ to a target sequence $S_t = \{(M_t(t_k), K_t(t_k))\}$ is performed over the optimal window length L^{opt} obtained from the transition optimisation (Eq. (5)), that is, $k \in [-L^{opt}, L^{opt}]$. The blended mesh $M_b(t_k) = (B, X_b(t_k))$, the source mesh $M_s(t_k) = (B, X_s(t_k))$, and the target mesh $M_t(t_k) = (B, X_t(t_k))$ share the same mesh connectivity B. The blended skeleton $K_b(t_k) = (B', J_b(t_k))$, the source skeleton $K_s(t_k) = (B', J_s(t_k))$, and the target skeleton $K_t(t_k) = (B', J_t(t_k))$ share the same joint connectivity B',

$$M_b(t_k) = G(M_s(t_k), M_t(t_k), \alpha(k)), \tag{6}$$

where $\alpha(k) = \frac{k+L^{opt}}{2L^{opt}}$ denotes the blending weight. $G(\cdot)$ denotes a nonlinear blend function that interpolates triangle transformations, applies them to the source mesh, and finally uses a Laplacian editing framework to link the transformed triangles back together, obtaining a nonlinear interpolated mesh [Tejera and Hilton 2013].

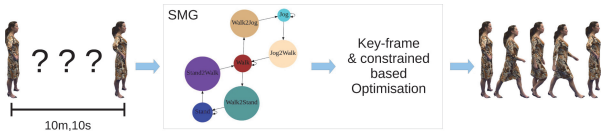


Fig. 6. Key-frame- and constraints-based graph optimisation.

The skeleton $K_b(t_k)$ is then extracted from $M_b(t_k)$ as described in Section 3.

Blending Appearance. To better preserve the visual quality, view-dependent rendering is implemented. For blended 3D meshes, there are no original images, so view-dependent rendering cannot be used directly. A solution is to first perform view-dependent texture mapping for the source and target meshes that have the original images, respectively, and then compute optical flows between them. Blending is then performed based on the computed flows to obtain view-dependent texture for the blended mesh [Casas et al. 2014].

5. GRAPH OPTIMISATION

Graph optimisation is performed to find the path through the surface motion graph that best satisfies the required animation constraints. Constraints can be specified in two ways: as user-defined key-frames together with timing and distance (Section 5.1); or as a query MoCap sequence, for which a matching animation is synthesised via an optimisation framework (Section 5.2).

5.1 Key-Frame-Based Graph Optimisation

Intermediate key-frames selected by the user provide hard constraints defining the desired movement. Alternatively, a series of key-frames can be specified with the path optimisation used to evaluate the best path between each pair of key-frames together with the start and end positons. Start and end key-frame locations specify the target traverse distance d_V and the target traverse time t_V chosen by the user. Both target traverse distance and time are used as soft constraints that define global constraints on the animation. Figure 6 illustrates key-frame-based graph optimisation. We now describe the path cost function to be optimised, before describing the optimisation process itself. The cost function for graph path optimisation to satisfy the constraints is described as follows.

Combined Cost. The cost function for a path P through the surface motion graph between a pair of key-frames is formulated as the combination of three costs: C_{tran} representing cost of transition between motions, soft constraints on distance C_{dist}, and time C_{time},

$$C(P) = C_{tran}(P) + w_{dist} \cdot C_{dist}(P) + w_{time} \cdot C_{time}(P), \tag{7}$$

where w_{dist} and w_{time} are weights for distance and time constraints, respectively. Throughout this work we set $w_{dist} = 1/0.3$ and $w_{time} = 1/10$ that equates the penalty for an error of 30cm in distance with an error of 10 frames in time [Arikan and Forsyth 2002].

Transition Cost. $C_{tran}(P)$ for a path P is defined as the sum of distortion for all transitions between concatenated 4DPC segments. If we denote the index for concatenated 4DPC segments as $\{f_i\}$, $i = 0, \ldots, N_{f-1}$, the total transition cost $C_{tran}(P)$ is computed as

$$C_{tran}(P) = \sum_{i=0}^{N_P-2} D(S_{f_i \to f_{i+1}}), \tag{8}$$

where N_P denotes the total number of transitions on path P and $D(S_{f_i \to f_{i+1}})$ (Section 4.2) the distortion for transition from motion sequence S_{f_i} to motion sequence $S_{f_{i+1}}$.

Distance Cost. $C_{dist}(P)$ for a path P with N_f frames on the surface motion graph is computed as the absolute difference between the user-specified target distance d_V and the total travelled distance $dist(P)$, given the 3D frames on the path of P is $\{M(t_f)\}, f = [0, N_f - 1]$,

$$C_{dist}(P) = |dist(P) - d_V|, \tag{9}$$

$$dist(P) = \sum_{f=0}^{N_f-2} |centre(M(t_{f+1})) - centre(M(t_f))|, \tag{10}$$

where function $centre()$ computes the projection of the centroid of the mesh onto the ground. If the target distance d_V is set to zero, the optimisation will find the shortest path in terms of distance on the graph.

Timing Cost. $C_{time}(P)$ for a path P with N_f frames is evaluated as the absolute difference between the user-specified target time t_V and the total travelled time $time(P)$,

$$C_{time}(P) = |time(P) - t_V|, \tag{11}$$

$$time(P) = N_f \cdot \Delta t, \tag{12}$$

where Δt denotes the frame rate (25 frames per second for our 4DPC data). If the target time t_V is set to zero, the optimisation will find the shortest temporal path on the graph.

5.1.1 *Path Optimisation.* We adopt an efficient approach to search for the optimal path that best satisfies the user-defined soft constraints. The optimal path P^{opt} minimises the combined cost $C(P)$ as defined in Eq. (7):

$$P^{opt} = \arg\min_P C(P). \tag{13}$$

First, the path needs to satisfy the hard constraints, that is, user-defined key-frames. Since there may be cycles in the graph for repetitive motion such as walking, the number of paths between key-frames may be infinite. Each path can be considered as a composition of a path without loops, plus attached loops. The optimisation can then be performed in two steps: we enumerate all paths without loops that satisfy key-frames and also store all loops attaching them; we compare all paths with their own optimal number of loops to find the globally optimal path.

Given a path P, it can be represented as a composition of a path without loops l_0 and loops $l_1, \ldots, l_{N_L}$. For simplicity, we can put the paths without loops and loops together $\mathbf{L} = [l_i]_{N_L}$ and denote the number of repetitions as $\mathbf{n} = [n_i]_{N_L}$, where, $n_0 = 1$ for the path without loops. We can then denote P as follows,

$$P = \mathbf{n} \cdot \mathbf{L}, \tag{14}$$

where $\cdot$ denotes the inner product of the number of loop repetitions n_i with the corresponding loop l_i to give the full path with concatenated loop repetitions.

The optimisation then becomes

$$P^{opt} = \mathbf{n}^{opt} \cdot \mathbf{L}^{opt} = \arg\min_{\mathbf{n},\mathbf{L}} \{C(\mathbf{n} \cdot \mathbf{L})\}. \tag{15}$$

The combined cost defined in Eq. (7) becomes

$$C(\mathbf{n}\cdot\mathbf{L}) = C_{tran}(\mathbf{n}\cdot\mathbf{L}) + w_{dist} \cdot C_{dist}(\mathbf{n}\cdot\mathbf{L}) + w_{time} \cdot C_{time}(\mathbf{n}\cdot\mathbf{L}). \tag{16}$$

In essence, a path without loops (or a single loop) is a graph path that represents a sequence of frames. The computation of transition, distance, and time costs has no influence, therefore, the transition, distance, and time cost for $\mathbf{n} \cdot \mathbf{L}$ can be computed as follows,

$$C_{tran}(\mathbf{n} \cdot \mathbf{L}) = \mathbf{n} \cdot C_{tran}(\mathbf{L}), \tag{17}$$

$$C_{dist}(\mathbf{n} \cdot \mathbf{L}) = |\mathbf{n} \cdot dist(\mathbf{L}) - d_V|, \tag{18}$$

$$C_{time}(\mathbf{n} \cdot \mathbf{L}) = |\mathbf{n} \cdot time(\mathbf{L}) - t_V|. \tag{19}$$

Once the surface motion graph is constructed and key-frames are selected, $\mathbf{L}$ is determined and the goal becomes to optimise $\mathbf{n}$ by minimising $C(\mathbf{n} \cdot \mathbf{L})$ defined in Eq. (16). A two-step optimisation is performed. Let N_k denote the number of walks, for the kth walk $l_{k,0}$, $\mathbf{L}_k$ is determined and the objective is to find the corresponding optimal repetitions of loops $\mathbf{n}_k^{opt}$ according to Eq. (15),

$$\mathbf{n}_k^{opt} = \arg\min_{\mathbf{n}_k} \{C(\mathbf{n}_k \cdot \mathbf{L}_k)\}. \tag{20}$$

Let k^{opt} denote the index of the global optimal walk, we enumerate all walks, compare their optimal cost, and find the minimal

$$k^{opt} = \arg\min_{k=1,\ldots,N_k} \left\{C\left(\mathbf{n}_k^{opt} \cdot \mathbf{L}_k\right)\right\}. \tag{21}$$

Finally, the optimal path P^{opt} is a composition of the optimal walk and loops $\mathbf{L}^{opt} = \mathbf{L}_{k^{opt}}$ together with optimal repetitions for each loop $\mathbf{n}^{opt} = \mathbf{n}_{k^{opt}}^{opt}$,

$$P^{opt} = \mathbf{n}_{k^{opt}}^{opt} \cdot \mathbf{L}_{k^{opt}}. \tag{22}$$

Depth-first search is exploited to decompose all possible paths (constrained by key-frames) to walks and loops $\mathbf{L}$.

5.1.2 *Integer Linear Programming Solver.* An integer linear programming solver is used to solve the optimisation of Eq. (20). Although the equation is nonlinear due to modulus computation in Eqs. (18) and (19), it can be converted into constrained Integer Linear Programming (ILP) subproblems. For a particular walk with loops $\mathbf{L}$, the corresponding number of repetitions $\mathbf{n}$ is optimised as four independent ILP subproblems, $\rho = 0, 1, 2, 3$.

$$\begin{aligned} minimise \quad & \mathbf{C}_\rho \cdot \mathbf{n}_\rho \\ subject\ to \quad & n_{0,\rho} = 1 \\ & 0 \le n_i \le +\infty, \quad int \\ & \mathbf{C}_{dist,\rho} \cdot \mathbf{n}_\rho \ge 0 \\ & \mathbf{C}_{time,\rho} \cdot \mathbf{n}_\rho \ge 0. \end{aligned} \tag{23}$$

$\mathbf{C}_\rho$, $\mathbf{C}_{tran,\rho}$, $\mathbf{C}_{dist,\rho}$, and $\mathbf{C}_{time,\rho}$ are vectors representing the combined costs, total transition, distance, and time costs for each subproblem ρ, whose elements are computed as

$$\begin{aligned} C_{tran,\rho,i} &= C_{tran}(l_i) \\ C_{dist,\rho,i} &= sign_d(\rho) \cdot w_d \cdot (dist(l_i) - d_V) \\ C_{time,\rho,i} &= sign_t(\rho) \cdot w_t \cdot (time(l_i) - t_V) \\ C_{\rho,i} &= C_{d,\rho,i} + C_{t,\rho,i} + C_{s,\rho,i}, \end{aligned} \tag{24}$$

where $sign_d = \{1, 1, -1, -1\}$ and $sign_t = \{1, -1, 1, -1\}$. For each subproblem, $\mathbf{n}_\rho^{opt}$ is solved efficiently by a standard ILP solver. The optimal repeat time of loops $\mathbf{n}$ for a particular walk with loops $\mathbf{L}$ then computed as the one that achieves the minimum combined cost,

$$\mathbf{n}^{opt} = \arg\min_{\rho=0,1,2,3} \left\{\mathbf{C}_\rho \cdot \mathbf{n}_\rho^{opt}\right\}. \tag{25}$$

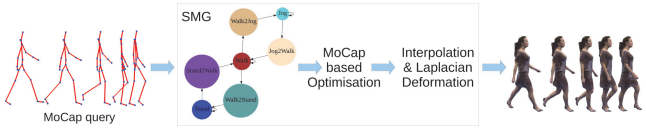


Fig. 7. Skeletal MoCap-based graph optimisation.

5.2 Skeletal MoCap-Based Optimisation

The aim is to enable skeletal MoCap-driven animation production using 4DPC data. Similar to traditional skeletal MoCap-driven animation, the input is skeletal MoCap data and the output is an animation from a 4DPC database performing the same actions as the input. To achieve this, a skeletal MoCap retrieval is first performed to find the closest path through the surface motion graph to the input MoCap sequence. Due to the scale limits of the available 4DPC database, the retrieved results do not exactly match the input MoCap sequence in terms of timing and styling. Additional steps of temporal interpolation and Laplacian deformation are then applied. Figure 7 illustrates skeletal MoCap-based graph optimisation.

Given a query skeletal MoCap sequence $Q = \{K_Q(t_k)\}$, $k \in [0, N_Q - 1]$ and the skeletons extracted from the surface motion graph $G = \{K_G(t_i)\}, i \in [0, N_G - 1]$, a retrieval path $P = \{K_G(t_{r_k})\}$ produces a cost, where N_Q and N_G are the total number of frames for query/retrieval and the surface motion graph, respectively. Index r_k is introduced to denote the matching between query and retrieval sequences in timing. The objective is to minimise the difference between the retrieved path P and the query sequence Q while minimising the number of transitions, where the cost functions are defined as follows.

Combined Cost. The combined cost function is defined as a combination of the retrieval cost and the transition cost,

$$C'(P) = C_{tran}(P) + w_{retr} \cdot C_{retr}(P), \tag{26}$$

where w_{retr} is set as $w_{retr} = 1$ to equate retrieval cost and transition cost. Since the transition cost is defined in the same way in Section 5.1, here we only need to introduce the retrieval cost.

Retrieval Cost. $C_{retr}(P)$ for a path P is defined as the sum of dissimilarity for all pairs of skeletons between query and retrieval,

$$C_{retr}(P) = \sum_{k=0}^{N_Q-1} c_{skel}(K_Q(t_k), K_G(t_{r_k})). \tag{27}$$

where $c_{skel}(K_Q(t_k), K_G(t_{r_k}))$ computes the skeletal dissimilarity between $K_Q(t_k)$ from the query and $K_G(t_{r_k})$ from the surface motion graph. Skeleton similarity between a source skeleton $K_s = (B', J_s)$, $J_s = \{x_i^s\}$, $i \in [0, N_J - 1]$ and a target skeleton $K_t = (B', J_t)$, $J_t = \{x_i^t\}$ is computed as follows,

$$c_{skel}(K_s, K_t) = \sum_{i=0}^{N_J-1} w_i \cdot \left|\left|x_i^s - x_i^t\right|\right|, \tag{28}$$

where x_i^s and x_i^t are the i^{th} joint 3D position vector for the source and the target, respectively, and N_J the total number of joints. Moreover, $||\cdot||$ computes Euclidean distance, and w_i is set as 1 to set equal weights for all joint positions. Note that K_s and K_t are prealigned in both translation and orientation.

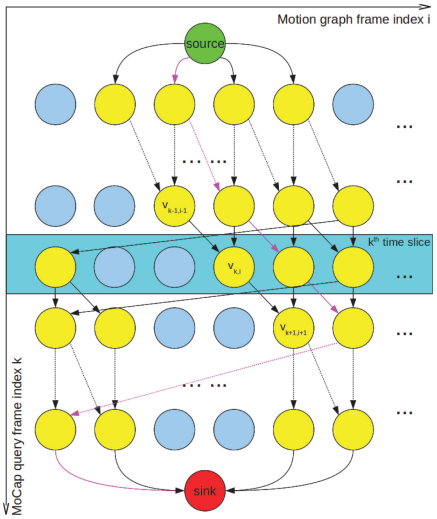


Fig. 8. Illustration of skeleton retrieval trellis graph. Each node denotes a pair of a MoCap query frame and a frame from the surface motion graph. Yellow nodes are candidates which mean skeleton dissimilarity below a given threshold. Edges show possible connections in the surface motion graph. Source and sink nodes are virtual nodes inserted and only used to find the shortest path. Colour light magenta denotes the shortest path from source to sink. The shortest path gives the desired retrieval result.

5.2.1 *Path Optimisation.* The optimal path P^{opt} is then found as the path which minimises the combined cost $C'(P)$ as defined in Eq. (26).

$$P^{opt} = \arg\min_{P} C'(P) \tag{29}$$

This optimisation problem can be formulated as a shortest path search problem. First, we create a trellis graph as illustrated in Figure 8. Each node of the trellis represents a pair of frames from the skeletal query sequence and surface motion graph, respectively. Given a query sequence indexed by k, $k \in [0, N_Q - 1]$ and an adjacency matrix $A_{i,j}$ for the surface motion graph where $i, j \in [0, N_G - 1]$, a trellis node $v_{k,i}$ is created for each pair of a query frame and a surface motion graph frame. To reduce the number of candidates and so manage the search complexity (given the $O(n^2)$ nature of the shortest path search), a maximum skeleton dissimilarity threshold T_{skel} is introduced.

For the k^{th} frame, only those frames from the surface motion graph below a skeleton dissimilarity threshold T_{skel} are treated as candidates (yellow circle shown in Figure 8). The connections from candidate trellis nodes in the $k-1^{th}$ row to the k^{th} row are decided by the adjacency matrix $A_{i,j}$ of the surface motion graph. If there is a directed edge from node i to node j in the surface motion graph, that is, $A_{i,j} = true$, then there is a directed edge from $v_{k-1,i}$ to $v_{k,j}$. Since the timing of query and surface motion graph may not match perfectly, we allow the repetition of 4DPC frames. This introduces additional edges from $v_{k-1,i}$ to $v_{k,i}$ even $A_{i,i} = false$. The node cost for a node $v_{k,i}$ is defined as

$$c_{node}(v_{k,i}) = w_{retr} \cdot c_{skel}(K_Q(t_k), K_G(t_i)). \tag{30}$$

The edge cost for a directed edge from $v_{k-1,i}$ to $v_{k,j}$ is defined as

$$c_{edge}(v_{k-1,i}, v_{k,j}) = c(M(t_i), M(t_j)). \tag{31}$$

Finally, we insert two virtual nodes into the trellis: a source and a sink. The source connects all candidates in the $0th$ row and the sink

connects by all candidates in the $N_k - 1^{th}$ row. All node and edge costs for the source and sink are set to 0. Dijkstra's algorithm is then applied to find the shortest path on the trellis graph from source to sink, $P'^{opt} = \{v_{k,r_k^{opt}}\}$. $P^{opt} = \{r_k^{opt}\}$ is then found as the optimal path through the surface motion graph.

5.2.2 *Interpolation.* The raw retrieved mesh sequence may appear like a "stop motion" due to the optimisation process allowing repetition of the same frame multiple times to match the timing between query and retrieval. A nonlinear temporal interpolation is used to synthesise intermediate frames over time to produce a plausible animation. For a retrieved 3D mesh sequence $\{M(t_{r_i})\}$, resulting from a MoCap-driven path optimisation, if $M(t_{r_i})$ starts to repeat n times, that is, $M(t_{r_i}) = M(t_{r_{i+1}}) = \cdots = M(t_{r_{i+n-1}})$, $M(t_{r_i}) \neq M(t_{r_{i-1}})$ and $M(t_{r_i}) \neq M(t_{r_{i+n}})$, $M(t_{r_{i+j}})$, $j \in [0, n-1]$. The j^{th} frame in the sequence, $M(t_{r_{i+j}})$, can be replaced with a blended mesh between $M(t_{r_i})$ and $M(t_{r_{i+n}})$, $M(t_{r_{i+j}}) = G(M(t_{r_i}), M(t_{r_{i+n}}), \alpha(j))$, where $\alpha(j) = \frac{j}{n}$. Nonlinear blending using Laplacian mesh editing to interpolate the mesh as presented in Section 4.4 is used to perform the temporal interpolation.

5.2.3 *Part-Based Laplacian Deformation.* Interpolation leads to smooth synthesised motion but is restricted to those body shapes represented within the 4DPC mesh data. This may, for example, be a generic walk or run that we wish to tailor to a more characteristic gait or action in the MoCap data. Although standard linear skinning techniques such as linear blend skinning (LBS) could be used to generate a skeleton-driven animation that best matches the motion and style defined by the query MoCap sequence, this would result in the following undesired effects: loss of surface dynamics due to the rigid deformation of a single mesh model; introduction of artefacts associated with linear vertex-based deformation; and the quality of the result being dominated by the particular vertex weights employed. Instead, we propose to synthesise skeleton-driven animations within a Laplacian deformation framework [Sorkine et al. 2004; Botsch and Sorkine 2008] constrained to a plausible deformation space learned from the 4DPC examples. The properties of the Laplacian coordinates and the learned deformation space ensure the preservation of both the surface detail and the inherent structure of the meshes, respectively. This approach, described in detail in the remainder of this section, combines previous space-time mesh sequence editing [Xu et al. 2007; Kircher and Garland 2008] with statistical models of mesh deformation [Summer et al. 2005].

Each source mesh M_s from the interpolated synthesised motion, with vertex positions $\mathbf{x}_s$, is deformed to match a set of positional soft constraints $\mathbf{x}_c$ extracted from the LBS result. These constraints, which are interactively selected by the user once for each character, need to avoid areas exhibiting artefacts and should constrain the pose of the character. The resulting mesh, denoted as $\tilde{M}_s$ with vertex positions $\tilde{\mathbf{x}}_\mathbf{s}$, is computed by solving the least-squares minimisation [Tejera et al. 2013]:

$$\tilde{\mathbf{r}}_\mathbf{s}, \tilde{\mathbf{x}}_s = \arg\min_{\mathbf{r},\mathbf{x}}(\|\mathbf{L}\mathbf{x} - \boldsymbol{\delta}(\mathbf{r})\|^2 + \|\mathbf{W}_c(\mathbf{x} - \mathbf{x}_c)\|^2), \tag{32}$$

where $\mathbf{W}_c$ is a diagonal weight matrix that allows for control of the importance of each positional constraint and $\boldsymbol{\delta}(\mathbf{r})$ denotes a learned deformation space. This deformation space is constructed by principal component analysis (PCA) of a set of F deformation examples represented in the space of Laplacian coordinates. First, a data matrix $\mathbf{M}$ is constructed by placing the concatenated $\boldsymbol{\delta}(\mathbf{x})$, $\boldsymbol{\delta}(\mathbf{y})$, and $\boldsymbol{\delta}(\mathbf{z})$ Laplacian coordinates of each example in its rows:

(a) Character1 (b) Fashion1 (c) Fashion2

Fig. 9. An example of body-part decomposition. Note that the leg parts for both Fashion1 and Fashion2 characters have a bigger overlap to account for the deformation in the areas of the skirt far from the given leg.

$$\mathbf{M} = \begin{pmatrix} \boldsymbol{\delta}_1^\top(\mathbf{x}) & \boldsymbol{\delta}_1^\top(\mathbf{y}) & \boldsymbol{\delta}_1^\top(\mathbf{z}) \\ \boldsymbol{\delta}_2^\top(\mathbf{x}) & \boldsymbol{\delta}_2^\top(\mathbf{y}) & \boldsymbol{\delta}_2^\top(\mathbf{z}) \\ \vdots & \vdots & \vdots \\ \boldsymbol{\delta}_F^\top(\mathbf{x}) & \boldsymbol{\delta}_F^\top(\mathbf{y}) & \boldsymbol{\delta}_F^\top(\mathbf{z}) \end{pmatrix}. \tag{33}$$

This data matrix is centred obtaining $\mathbf{M_c} = \mathbf{M} - \bar{\mathbf{M}}$, where $\bar{\mathbf{M}}$ is an $F \times 3n$ matrix whose rows are the mean of the rows of the data matrix $\mathbf{M}$. In order to obtain a basis representing the space of deformations, an SVD is performed over the matrix $\mathbf{M_c}$: $\mathbf{M_c} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{V}$ is a $3n \times F$ matrix with each column containing a basis eigenvector for the shape deformation space. The first l eigenvectors $\mathbf{e}_k$ representing 95% of the variance are kept, which gives a linear basis of the form:

$$\boldsymbol{\delta}(\mathbf{r}) = \bar{\boldsymbol{\delta}} + \sum_{k=1}^{l} r_k \mathbf{e}_k = \bar{\boldsymbol{\delta}} + \mathbf{E}\mathbf{r}, \tag{34}$$

where r_k are scalar weights for each eigenvector, $\mathbf{r}$ is an l-dimensional weight vector, and $\mathbf{E}$ is a $3n \times l$ matrix whose columns are the first l eigenvectors of length $3n$. Since $\boldsymbol{\delta}(\mathbf{r})$ represents the main modes of shape deformation as described by the Laplacian coordinates, constraining the edited mesh to lie within this space strives to achieve plausible deformations as exhibited in the example meshes.

The described deformation framework can be applied by either considering the entire body as a deformation instance [Tejera et al. 2013] or by decomposing the mesh into a set of regions that will each produce an independent deformation space [Tejera and Hilton 2013]. The latter technique is preferred within this context, since it allows extrapolation outside the observed set of poses by modelling the deformation of each body part independently. This greatly extends the range of plausible output deformations beyond the example whole body poses used to learn the deformation space. The part decomposition chosen for each of the characters is shown in Figure 9. Torso, head, and limbs are edited following a hierarchical editing process to fulfil the positional constraints for each body part. If a constrained vertex is shared by a parent-child part pair, the following assignment is performed: for the parent, it is assigned the position of the vertex in the LBS sequence; for the child, it is set to the position of the vertex in the deformed parent. This minimises possible discontinuities between vertex positions shared by more than one body region, ensuring smooth transitions between parts in the complete whole body mesh.

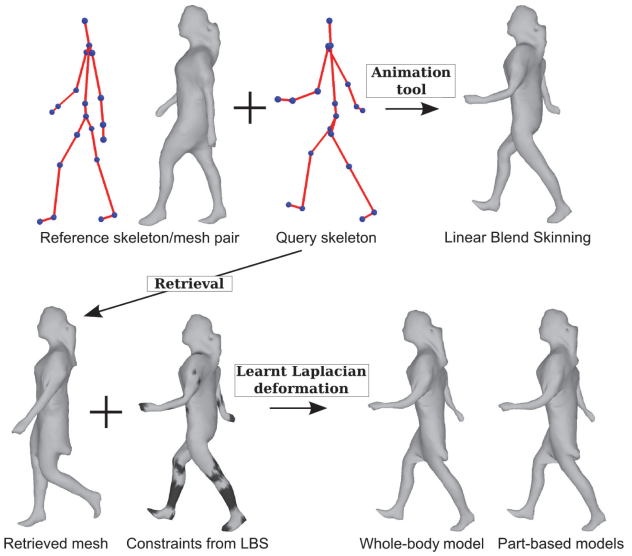


Fig. 10. Comparison of the part-based Laplacian deformation technique against the use of whole body models and traditional LBS for Fashion2.

Table I. 4DPC Database

Subjects	Motions	N_s	N_f
Character1	walk, jog, stand, stagger, hit, tense + transitions	10	442
Fashion1	walk, pose, twirl + transitions	6	491
Fashion2	walk, pose, twirl + transitions	6	435

In order to maximise the range of plausible deformations that can be synthesised using the proposed framework, the set of deformation examples chosen to build the deformation space for each part should represent as much pose variation as possible. In practice, this set is constructed by manual selection of frames from the database of each character, which also helps to avoid frames containing reconstruction artefacts. Automatic extraction of the most suitable deformation examples for each body part could be achieved by selecting extreme poses as examples. Figure 10 shows a comparison between the performance of the proposed skeleton-driven technique using both whole body and part-based deformation models and the traditional LBS approach. The collapse of the skirt area with LBS and the thinning of body and right wrist for the global approach are corrected with the proposed part-based learned Laplacian deformation. Further comparisons are presented in the supplementary video available at the ACM Digital Library.

6. EXPERIMENTAL RESULTS AND EVALUATION

A publicly available 3D video database [Starck and Hilton 2007] is aligned using Budd et al. [2012] to create a 4DPC video database containing sequences of an actress Roxanne in three different costumes, performing multiple motions with complex nonrigid movement of clothing and hair. The 4DPC database sequences are each used to construct surface motion graphs for all of our experiments. The publicly available CMU MoCap database [CMU 2014] is used to provide query motion sequences for skeletal MoCap-driven animation synthesis. Tables I and II show the summary of motions used in the experiments, where N_s and N_f denote the number of sequences and frames.

Table II. CMU MoCap Database

Motions	Subjects	N_s	N_f
Walk	2, 5, 6, 7, 8, 9, 10, 12, 16, 26, 27, 29, 32, 35, 37, 38, 39, 43, 45, 46, 47, 49,	100	8056
Run	2, 9, 16, 35	31	1094

6.1 Key-Frame Motion Synthesis

Surface motion graphs are automatically constructed from 4DPC sequences for the performer with three different costumes. Optimisation is performed in seconds for user-defined constraints on distance and time (synthesis results are presented in accompanying videos). An example of selected frames from a synthesised motion, respectively, for Fashion1, Fashion2, and Character1 captured in a virtual camera view are shown in Figure 11. These results demonstrate that the motion synthesis preserves the detailed clothing and hair dynamics in the captured 4DPC sequences and does not produce unnatural movements at transitions.

Motion synthesis is evaluated for the three surface motion graphs that represent potential transitions with four pairs of key-frames for each costume, as shown in Table III. Evaluation is performed by synthesising motions for target constraints on distances of 1–20m in 1m intervals and times of 1–40s in 1s intervals giving 800 sequences for each key-frame pair and 9600 synthesised sequences in total. The maximum, minimum, and root mean square errors over all synthesised sequences for distance moved and timing are presented in Table III for the sequences generated from each key-frame pair. This analysis shows that the maximum distance and timing errors are less than 1% of the target indicating that the path optimisation generates sequences which accurately satisfy the user-defined constraints. Smoothness cost is evaluated as a weighted average of Hausdorff distance for overlapping frames at transitions. The weights are set to decrease about the central frame in the same way as $a'(k)$ in Eq. (2). Computation times are given for an ILP solver together with a Matlab implementation of the synthesis framework running on a single processor machine. The computation time is approximately constant with respect to the distance and timing constraints, as indicated by the low standard deviation.

6.2 Skeleton Retrieval

Test query sequences are taken from the CMU MoCap database across different subjects performing "Walk", "Run", "Walk2Stand", "Stand2Walk", "Walk2Run", and "Run2Walk". Three surface motion graphs for Fashion1, Fashion2, and Character1 are used to retrieve MoCap sequences. Synthesis results are presented in accompanying videos. Examples of selected frames of synthesised results are shown in Figure 12. Evaluation results for Character1 are shown in Figures 13 and 14. Similar results are obtained for Fashion1 and Fashion2.

For each MoCap sequence, the average per-joint error is computed as a per-joint velocity vector difference to evaluate how similar retrieved surface motions are compared to the query MoCap sequences. The standard deviation of this error is also shown as an error bar for each query sequence. Evaluation is performed for animation with temporal interpolation only and with interpolation plus Laplacian deformation. The latter further reduces the motion dissimilarity between the query and animation. For all "Walk" including "Walk2Stand" and "Stand2Walk" sequences, synthesis-with-interpolation-only error is around $2cm/frame$ with standard deviation less than $0.6cm/frame$, except for 08_07 which is a walk with exaggerated stride, the error shows around $3.5cm/frame$ with standard deviation $1.1cm/frame$. Synthesis-with-interpolation-plus-

(a) Fashion1: Pose#1 → Twirl#85 (b) Fashion2: Pose#1 → Twirl#100 (c) Character1: Stand#1 → Hit#45

Fig. 11. Example rendering results for synthesised character animation from 4DPC database of Roxanne spanning three actions. The target traversing distance is set as 10 metres for (a), (b), and (c); the target traversing time is set as 20 seconds for (a) and (b), 10 seconds for (c).

Table III. Evaluation for Roxanne

SMG: Key-frames	Smoothness (m)		Distance error (m)			Time error (s)			Cputime (s)
	min	max	min	rms	max	min	rms	max	$mean \pm dev.$
Character1: Stand#1 → Hit#45	0.04	0.02	0.0001	0.17	0.96	0	0.16	0.88	14.43 ± 7.05
Stand#1 → Walk#16	0.04	0.03	0.0002	0.21	0.92	0	0.17	0.92	12.21 ± 4.74
Walk#16 → Jog#13	0.04	0.03	0.0001	0.21	0.96	0	0.19	0.96	13.54 ± 4.98
Jog#13 → Hit#45	0.04	0.03	0.0002	0.20	0.98	0	0.15	0.96	12.20 ± 3.42
Fashion1: Pose#1 → Twirl#85	0.06	0.02	0.0001	0.23	0.95	0	0.15	0.96	12.79 ± 2.92
Pose#1 → Walk#15	0.06	0.02	0.0001	0.47	0.99	0	0.17	0.96	5.65 ± 1.46
Walk#15 → WalkPose#37	0.04	0.01	0.0000	0.33	1.00	0	0.23	1.00	12.01 ± 4.85
WalkPose#37 → Twirl#85	0.04	0.02	0.0001	0.29	1.00	0	0.19	1.00	10.10 ± 3.07
Fashion2: Pose#1 → Twirl#100	0.04	0.02	0.0000	0.23	0.95	0	0.31	1.00	12.87 ± 3.61
Pose#1 → Walk#15	0.05	0.02	0.0008	0.39	0.98	0	0.35	1.00	7.09 ± 1.91
Walk#15 → WalkPose#37	0.04	0.02	0.0002	0.36	1.00	0	0.33	1.00	14.95 ± 6.11
WalkPose#37 → Twirl#100	0.04	0.02	0.0002	0.28	1.00	0	0.33	0.96	10.51 ± 2.77

A grid of target 20 × 40 (metres×seconds) is tested for each pair of key-frames shown in the first column.

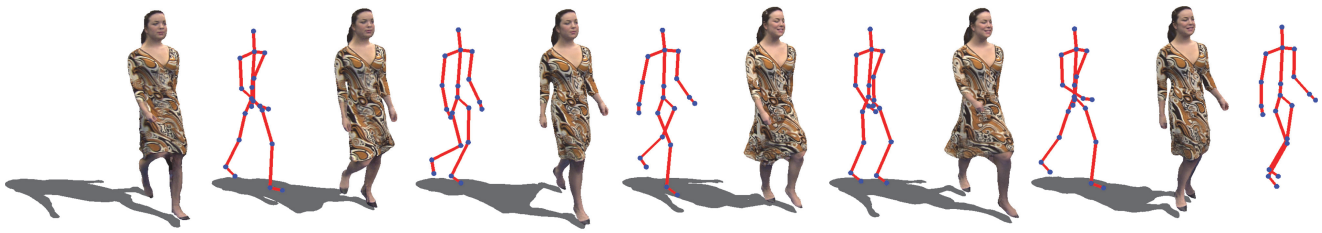
deformation error falls to $0.3cm/frame$ with standard deviation less than $0.1cm/frame$, and for 08_07 to $0.6cm/frame$ with standard deviation $0.2cm/frame$. For all "Run" including "Walk2Run" and "Run2Walk" sequences shown in Figure 14, synthesis-with-interpolation-only error is also around $3.5cm/frame$ with standard deviation less than $1.2cm/frame$ and synthesis-with-interpolation-plus-deformation error is $0.8cm/frame$ with standard deviation $0.2cm/frame$. This evaluation of skeleton-driven animation of the hybrid skeletal surface motion graph with 131 sequences demonstrates that the approach accurately matches the target skeletal motion. The hybrid representation extends the range of animation beyond the original 4DPC.

7. DISCUSSION AND LIMITATIONS

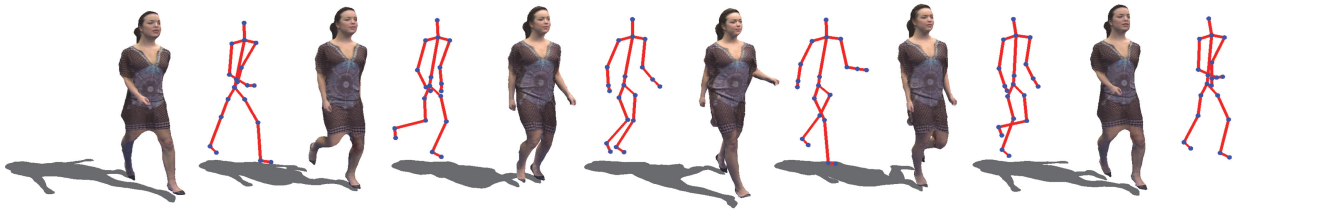
The introduction of 4DPC techniques to capture time-varying geometry and appearance of an actor's performance allows us to exploit surface motion graphs to produce video-realistic character animation with reduced time and effort compared to traditional MoCap-based animation techniques. Detailed surface dynamics are reconstructed from the capture and reproduced in the synthesised animation together with realistic appearance based on the captured video.

In this work the use of temporally aligned 4DPC sequences enables a learned space of plausible surface deformations. This allows nonlinear interpolation between captured sequences at transitions, giving improved motion continuity over previous surface motion graph approaches based on single-frame transitions [Starck et al. 2005; Huang et al. 2009]. The use of a separate learned deformation space for each body part also extends the range of motion, allowing synthesis of novel poses outside the observed range of motion. This allows accurate MoCap-driven animation by editing the retrieved surface motion data to match the skeletal motion. The hybrid combination of skeletal animation with surface motion graphs greatly increases the flexibility and range of animated motions which can be achieved. This is illustrated by the synthesis of variations in walk and run motions from a library of MoCap data. Extrapolation of the surface motion outside the range of 4DPC data is limited by the range of poses observed for each body part. If a novel motion includes body-part poses significantly outside this range, then unnatural mesh deformation artefacts may occur. The part-based representation alleviates this problem compared to previous global learnings of deformation spaces [Tejera et al. 2013] which were limited to similar whole body poses. The range of motion for part-based deformation could be further extended by capturing the full range of motion for each body part in the 4DPC dataset.

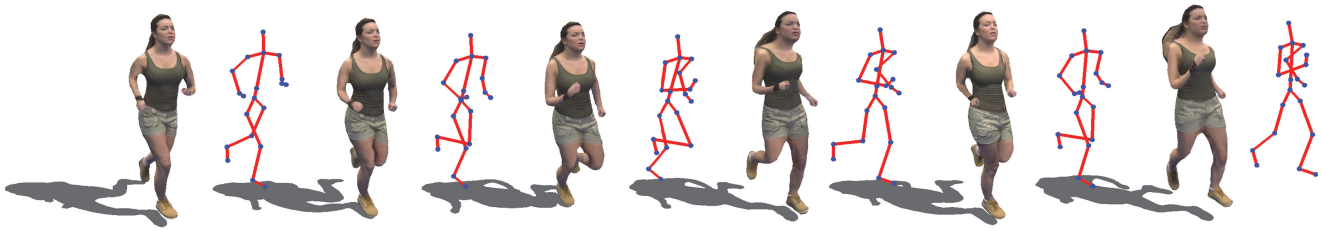
Resulting animation sequences using the hybrid skeletal-surface motion graph still exhibit some artefacts in the mesh shape and appearance. This is primarily due to errors and noise in the reconstruction and temporal alignment process [Budd et al. 2012]. Geometric reconstruction errors in the shape are exhibited as distortion in the leg/arm/clothing shape and noise on the silhouette boundary that is visible for slow motions. The learned surface deformation space will also model any temporally coherent distortions in shape but acts as a low-pass filter for noise in the per-frame reconstruction, due to the eigen basis only representing 95%



(a) Fashion 1 for query "Walk" Sequence 02_01



(b) Fashion 2 for query "Walk" Sequence 02_02



(c) Character 1 for query "Run" Sequence 02_03

Fig. 12. Example rendering results for synthesised character animation from Roxanne dataset for query MoCap motion "Walk" and "Run".

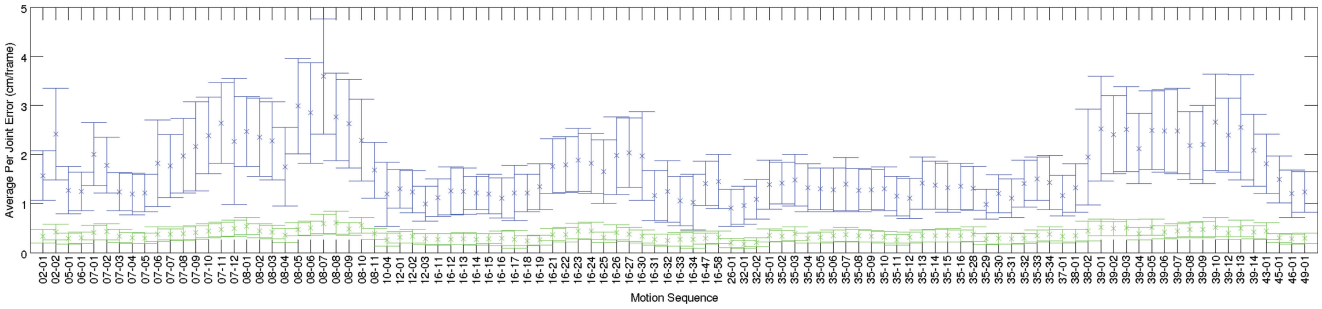


Fig. 13. Evaluation for query MoCap motion "Walk" retrieved surface motion from surface motion graph of Character1. Average per-joint position errors of motion sequences are shown for only interpolation performed (blue) and both interpolation and deformation performed (green). Note: exact CMU references of sequences appear in the figure.

of the variance. Future improvement in the multiple-view reconstruction and alignment process used for 4DPC will reduce these artefacts.

The hybrid skeletal-surface motion graph relies on the automatic annotation of the 4DPC with a skeletal pose estimate at each frame as estimated from the temporally aligned surface. Inaccuracies in skeletal pose estimation may occur due to loose clothing, which causes large surface distortions and inaccuracies in the temporal surface alignment. In practice, for the 4DPC sequences used in this work with moderately loose clothing, the skeletal pose estimation has been found to achieve sufficient accuracy for the retrieval of similar motions to a target MoCap sequence. The space-time surface deformation then allows accurate matching to the target skeletal motion to produce the final animation, as illustrated in the quantitative evaluation. For more complex cases where loose clothing obscures the underlying body pose, it may be necessary to introduce additional constraints for skeletal pose estimation.

The final rendering quality of the character animation depends on the quality of the multiple-view image sequences from the actor performance capture and the accurate alignment of textures at transitions using the 4D video texture approach [Casas et al. 2014]. Captured image quality is limited by the camera resolution, and artefacts occur due to motion blur, depth of field, and colour balancing across views. Blending of textures at transitions between captured sequences depends on the optic flow alignment. While this approach has been demonstrated to achieve a visual quality

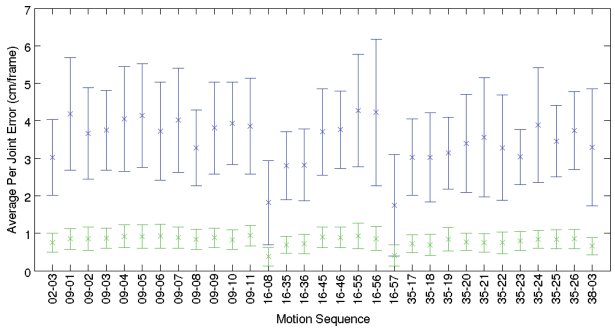


Fig. 14. Evaluation for query MoCap motion "Run" retrieved surface motion from surface motion graph of Character1. Average per-joint position errors of motion sequences are shown for only interpolation performed (blue) and both interpolation and deformation performed (green). Note: exact CMU reference of sequences appear in the figure.

comparable to the captured video with dynamic appearance, artefacts may occur if the alignment fails. In the results presented, the main rendering artefacts occur on the face due to changes in texture appearance between views and at transitions. Future research will consider refinement of the texture alignment and relighting to improve colour matching at transitions.

8. CONCLUSION

A novel hybrid skeletal-surface motion graph representation of 4D performance capture data has been introduced. This allows skeletal-driven animation control for production of novel surface sequences. Constrained graph path optimisation generates novel sequences by concatenation of fragments of the captured 4D data while ensuring smooth transitions between sequences. Algorithms are introduced to support animation driven by user-specified key-frame constraints or matching to a skeletal motion capture sequence. Skeleton-driven animation is combined with space-time mesh sequence editing using a learned part-based model of surface deformation to ensure seamless transitions and extend the range of motion. This allows variation in the motion beyond that of the captured 4D data while maintaining plausible surface dynamics of shape and appearance.

Quantitative evaluation on over 9000 key-frame animation sequences shows that the maximum distance and timing errors are less than 1% of the target motion, demonstrating that the path optimisation generates sequences which accurately satisfy the user-defined constraints. Skeleton-driven animation is evaluated using 131 sequences from the CMU motion capture database. Synthesised motions with space-time editing match the target motion joint velocity to within $1cm/frame$ average error. Space-time editing to match the target skeletal motion reduces the error by a factor of two compared to samples from the original captured 4D data, with the remaining error resulting primarily from differences in skeletal dimensions of the character from the target motion. Results demonstrate that the proposed approach accurately matches the target motion capture sequence. This extends the range of synthesised surface motion beyond the captured 4D performance, allowing stylistic variation in the character animation while maintaing plausible dynamics of shape and appearance.

To support future research on animation from 4DPC, the datasets used in this article are available for download from the CVSSP 3D data repository at `http://cvssp.org/cvssp3d`.

ACKNOWLEDGMENT

We would like to thank Marco Volino for setting up a new Web site for CVSSP 3D data repository.

REFERENCES

O. Arikan and D. Forsyth. 2002. Interactive motion generation from examples. *ACM Trans. Graph.* 21, 3, 483–490.

O. Arikan, D. Forsyth, and J. O'Brien. 2003. Motion synthesis from annotations. *ACM Trans. Graph.* 22, 3, 402–408.

M. Botsch and O. Sorkine. 2008. On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph.* 14, 1, 213–230.

C. Bregler, M. Covell, and M. Slaney. 1997. Video rewrite: Driving visual speech with audio. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'97)*. ACM, New York, NY, USA, 353–360.

C. Budd, P. Huang, M. Klaudiny, and A. Hilton. 2012. Global non-rigid alignment of surface sequences. *Int. J. Comput. Vis.* 102, 1–3, 256–270.

D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton. 2013. Interactive animation of 4D performance capture. *IEEE Trans. Vis. Comput. Graph.* 19, 5, 762–73.

D. Casas, M. Volino, J. Collomosse, and A. Hilton. 2014. 4D video textures for interactive character appearance. *Comput. Graph. Forum* 33, 2, 371–380.

G. L. Cmu. Carnegie mellon university motion capture database. http://mocap.cs.cmu.edu/.

T. Ezzat and G. Geiger. 2002. Trainable videorealistic speech animation. *ACM Trans. Graph.* 21, 3, 388–398.

M. Flagg, A. Nakazawa, Q. Zhang, S. B. Kang, Y. K. Ryu, I. Essa, and J. M. Rehg. 2009. Human video textures. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'09)*. ACM, New York, NY, USA, 199–206.

J. Gall, C. Stoll, and E. D. Aguiar. 2009. Motion capture using joint skeleton tracking and surface estimation. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. 1746–1753.

M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen. 2003. Snap-together motion: Assembling run-time animations. *ACM Trans. Graph.* 22, 3, 181–188.

R. Heck and M. Gleicher. 2007. Parametric motion graphs. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'07)*. ACM Press, New York, 129–136.

A. Hornung and E. Dekkers. 2007. Character animation from 2D pictures and 3D motion data. *ACM Trans. Graph.* 26, 1, 1–9.

P. Huang, A. Hilton, and J. Starck. 2009. Human motion synthesis from 3D video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. 1478–1485.

P. Huang, A. Hilton, and J. Starck. 2010. Shape +. *Int. J. Comput. Vis.* 89, 2–3, 362–381.

A. J. Hunt and A. W. Black. 1996. Unit selection in a concatenative speech synthesis system. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'96)*. 373–376.

D. L. James, C. D. Twigg, A. Cove, and R. Y. Wang. 2007. Mesh ensemble motion graphs. *ACM Trans. Graph.* 26, 4.

S. Kircher and M. Garland. 2008. Free-form motion processing. *ACM Trans. Graph.* 27, 2, 1–13.

L. Kovar, M. Gleicher, and F. Pighin. 2002. Motion graphs. *ACM Trans. Graph.* 21, 3, 473–482.

A. Laurentini. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.* 16, 2, 150–162.

J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.* 21, 3, 491–500.

Y. Rubner, C. Tomasi, and L. Guibas. 1998. A metric for distributions with applications to image databases. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'98)*. 59–66.

A. Schodl and I. Essa. 2002. Controlled animation of video sprites. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'02)*. ACM Press, New York, 121–127.

A. Schodl, R. Szeliski, D. Salesin, and I. Essa. 2000. Video textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*. ACM Press, 489–498.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rossl, and H.-P. Seidel. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'04)*. ACM Press, New York, 175–184.

J. Starck and A. Hilton. 2007. Surface capture for performance-based animation. *IEEE Comput. Graph. Appl.* 27, 3, 21–31.

J. Starck, G. Miller, and A. Hilton. 2005. Video-based character animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'05)*. ACM Press, New York, 49–58.

R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popovic. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph.* 24, 3, 488–495.

M. Tejera, D. Casas, and A. Hilton. 2013. Animation control of surface motion capture. *IEEE Trans. Cybernet.* 43, 1532–1545.

M. Tejera and A. Hilton. 2013. Learning part-based models for animation from surface motion capture. In *Proceedings of the International Conference on 3D Vision (3DV'13)*. 159–166.

F. Xu, Y. Liu, J. Tompkin, G. Bharaj, J. Kautz, C. Theobalt, C. Stoll, Q. Dai, and H.-P. Seidel. 2011. Video-based characters creating new human performances from a multi-view video database. *ACM Trans. Graph.* 30, 4, 32:1–32:10.

W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo. 2007. Gradient domain editing of deforming mesh sequences. *ACM Trans. Graph.* 26, 3.

Received March 2013; revised February 2014; accepted September 2014