Surface-based Character Animation

Dan Casas, Peng Huang and Adrian Hilton

16.1 Introduction

Current interactive character authoring pipelines commonly consist of two steps: modelling and rigging of the character model which may be based on photographic reference or high-resolution 3D laser scans, as discussed in chapters 13 and 14; and move-trees based on a database of skeletal motion capture together with inverse dynamic and kinematic solvers for secondary motion. Motion graphs [Kovar et al. 02] and parametrized skeletal motion spaces [Heck and Gleicher 07] 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.

Chapters 11, **??** and 12 introduced recent advances in performance capture [Starck and Hilton 07b, Gall et al. 09b] that have demonstrated highly realistic reconstruction of motion using a temporally coherent mesh representation across sequences, referred to as 4D video. This allows replay of the captured motions with free-viewpoint rendering and compositing of performance in post-production whilst maintaining photo-realism. Captured sequences have been exploited for retargeting surface motion to other characters [Baran et al. 09] and analysis of cloth motion to simulate novel animations through manipulation of skeletal motion and simulation of secondary cloth movement [Stoll et al. 10]. However, these approaches do not enable authoring of interactive characters which allow continuous movement control and reproduction of secondary motion for clothing and hair.

This chapter presents a framework for authoring interactive characters based on actor performance capture. A Surface Motion Graph representation [Huang et al. 09] is presented to seamlessly link captured sequences, allowing authoring of novel animations from user specified space-time constraints. A 4D Parametric Motion Graph representation [Casas et al. 13] is described for real-time interactive animation from a database of captured 4D video sequences. Finally, a rendering approach referred to as 4D Video Textures [Casas et al. 14] is introduced to synthesize realistic appearance for parametric characters.

16.2 Surface Motion Graphs

As discussed in chapters 11, **??** and 12, multiple view reconstruction of human performance as a 3D video has advanced to the stage of capturing detailed non-rigid dynamic surface shape and appearance of the body, clothing and hair during motion [Starck and Hilton 07b, de Aguiar et al. 08a, Vlasic et al. 08a, Gall et al. 09b]. Full 3D video scene capture holds the potential to create truly realistic synthetic animated content by reproducing the dynamics of shape and appearance currently missing from marker-based skeletal motion capture. There is considerable interest in the reuse of captured 3D video sequences for animation production. For conventional skeletal motion capture (MoCap), *Motion Graph* techniques [Molina-Tanco and Hilton 00, Kovar et al. 02, Arikan and Forsyth 02, Lee et al. 02] are widely used in 3D character animation production for games and film. In this section, we present a framework that automatically constructs motion graphs for 3D video sequences, called *Surface Motion Graphs* [Huang et al. 09], and synthesizes novel animations to best satisfy user specified constraints on movement, location and timing. Figure 16.1 shows an example novel animation sequence produced using a Surface Motion Graph with path optimization to satisfy the user constraints.

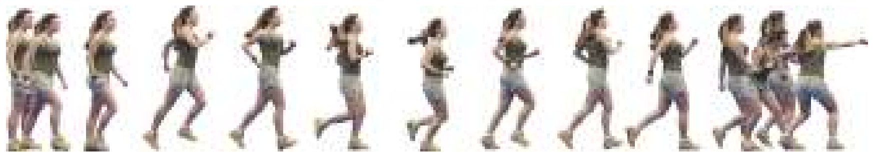
Figure 16.1: An example of synthesized 3D character animation (10 transitions). Target: Stand#1→Hit#45, 10 metres, 250 frames.

Character Animation Pipeline

A frame-to-frame similarity matrix is first computed between all frames across all 3D video motion sequences in the 3D video database. Potential transitions between motions are automatically identified by minimising the total dissimilarity of transition frames. The idea behind this is to minimize the discontinuity that may be introduced by transitions when transferring within or across different motions. A Surface Motion Graph is then constructed using these transitions. Once the graph structure is computed, a path on the graph is optimized according to user input such as key-frames, global timing and distance constraints. Finally, concatenative motion synthesis and rendering is performed to produce video-realistic character animation.

Graph Representation

A Surface Motion Graph represents possible inter- and intra-sequence transitions for 3D video sequences, analogous to motion graphs [Kovar et al. 02] for skeletal motion capture sequences. It is defined as a directed graph: each node denotes a 3D video sequence; each edge denotes one or multiple possible transitions. A path on the graph then provides a possible motion synthesis. Figure 16.2 shows an example of Surface Motion Graph constructed from multiple 3D video motion sequences for the actor shown. The method to automatically identify transitions is described in Section 16.2.

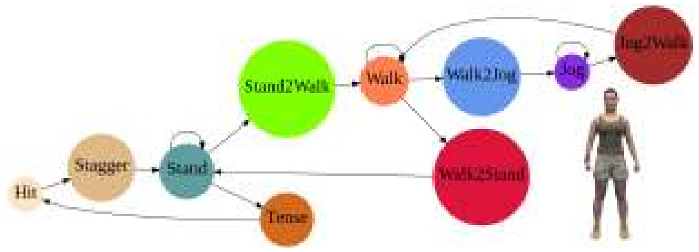

Figure 16.2: An example of Surface Motion Graph for a game character.

Graph Optimization

Graph Optimization is performed to find the path through the Surface Motion Graph which best satisfies the required animation constraints. Intermediate key-frames selected by the user provide hard constraints defining the desired movement. 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, which define global constraints on the animation. The cost function for graph path optimization 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},

$$C(P) = C_{tran}(P) + w_{dist} \cdot C_{dist}(P) + w_{time} \cdot C_{time}(P). \tag{16.1}$$

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 $w_{time} = 1/10$ which equates the penalty for an error of $30cm$ in distance with an error of 10 frames in time [Arikan and Forsyth 02].

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 $\{M(t_f)\}$,$f = [0, N_f - 1]$,

$$C_{dist}(P) = |dist(P) - d_V|. \tag{16.2}$$

$$dist(P) = \sum_{f=0}^{N_f-2} |centre(M(t_{f+1})) - centre(M(t_f))|. \tag{16.3}$$

where function $centre()$ computes the projection of the centroid of the mesh onto the ground.

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 $time(P)$,

$$C_{time}(P) = |time(P) - t_V|. \tag{16.4}$$

$$time(P) = N_f \cdot \Delta t. \tag{16.5}$$

where Δt denotes the frame rate (e.g. 25 frames per second).

Transition Cost. $C_{tran}(P)$ for a path P is defined as the sum of distortion for all transitions between concatenated 3D video segments. If we denote the index for concatenated 3D video segments as $\{f_i\}$, $i = 0, ..., N_{f-1}$, the total transition cost $C_{tran}(P)$ is computed as

$$C_{tran}(P) = \sum_{i=0}^{N_P-2} D(S_{f_i \to f_{i+1}}). \tag{16.6}$$

where N_P denotes the total number of transitions on path P and $D(S_{f_i \to f_{i+1}})$ (Section 16.2 Equation 16.9) the distortion for transition from motion sequence S_{f_i} to motion sequence $S_{f_{i+1}}$.

Path Optimization. Finally, the optimal path P^{opt} for a given set of constraints is found to minimize the combined cost $C(P)$, as defined in Equation 16.1,

$$P^{opt} = \arg\min_P C(P). \tag{16.7}$$

An efficient approach using Interger Programming to search for the optimal path that best satisfies the user-defined soft constraints can be found in [Huang et al. 09].

Transitions

A transition of the Surface Motion Graphs $S_{i\rightarrow j}$ is defined as a seamless concatenation of frames from two 3D video sequences 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 $M_{i\rightarrow j}(t_k) = G(M_i(t_{m+k}), M_j(t_{n+k}), \alpha(k))$ will be generated by a non-linear 3D mesh blend [Tejera et al. 13]. The distortion measure of a transition frame $M_{i\rightarrow j}(t_k)$ is then defined as the weighted 3D shape dissimilarity (Equation 16.11) between frame $M_i(t)$ and $M_j(t)$,

$$d(M_{i\rightarrow j}(t_k)) = \alpha'(k) \cdot c(M_i(t_{m+k}), M_j(t_{n+k})). \tag{16.8}$$

where $\alpha'(k) = \min(1 - \alpha(k), \alpha(k))$. The total distortion for a transition sequence $S_{i\rightarrow j}$ is then computed as the sum of the distortion of all transition frames,

$$D(S_{i\rightarrow j}) = \sum_{k=-L}^{L} d(M_{i\rightarrow j}(t_k)). \tag{16.9}$$

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

$$S^{opt}_{i\rightarrow j} = \arg \min_{S_{i\rightarrow j}} D(S_{i\rightarrow j}). \tag{16.10}$$

3D Shape Similarity. To measure the similarity of the 3D mesh geometry within each frame of the 3D video database, Shape histograms are used. This has previously demonstrated to give good performance for measuring non-rigid deformable shape similarity for 3D video sequences of human performance [Huang et al. 10]. The volumetric shape histogram $H(M)$ represents the spatial occupancy of the bins for a given mesh M. We define a measure of shape 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.

$$c_{shape}(M_r, M_s) = \min_{\phi} \|H(M_r) - H(M_s, \phi)\|. \tag{16.11}$$

The volumetric shape histogram $H(M)$ partitions the space which contains a 3D object into disjoint cells and counts the number of occupied voxels falling in each bins to construct a histogram as a signature for this 3D object. The space is represented in a spherical coordinate system (R, θ, ϕ) around the centre of mass. An example of a 3D Shape Histogram is shown in Figure 16.3.

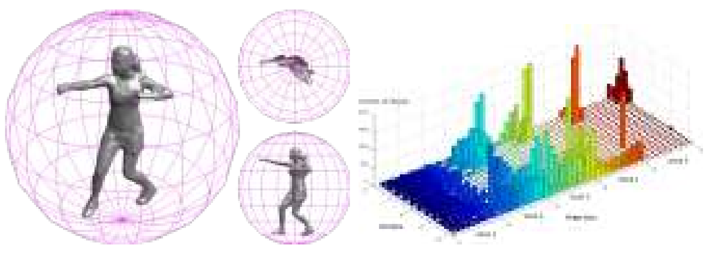

Figure 16.3: A 3D Shape Histogram of 5 shells, 10 bins for θ and 20 bins for ϕ together with the space partitions on the 5th shell is illustrated.

A frame-to-frame static similarity matrix $\mathbf{C} = [c_{r,s}]_{N\times N}$, where N denotes the total number of frames, between all frames across all 3D video sequences in the 3D video database is pre-computed according to Eq.16.11,

$$c_{r,s} = c_{shape}(M_r, M_s). \tag{16.12}$$

Adaptive Temporal Filtering Each transition is determined by a tuple (m, n, L). The global optimization is then performed by testing all possible tuples (m, n, L) and so finding optimal arguments for the minimum,

$$(m^{opt}, n^{opt}, L^{opt}) = \arg\min_{m,n,L} \sum_{k=-L}^{L} \alpha'(k) \cdot c_{m+k,n+k}. \tag{16.13}$$

This equates to performing an adaptive temporal filtering with window size $2L + 1$ and weighting $\alpha'(k)$ on the pre-computed static similarity matrix $\mathbf{C}$. The process is computationally efficient. To obtain multiple transitions between a pair of sequences, top T best transitions are preserved. T is pre-determined by the user. An example of the frame-to-frame 3D shape similarity matrix and transition frames (marked in yellow) evaluated using a temporal window $L = 4$ and $T = 4$ are shown in Figure 16.4.

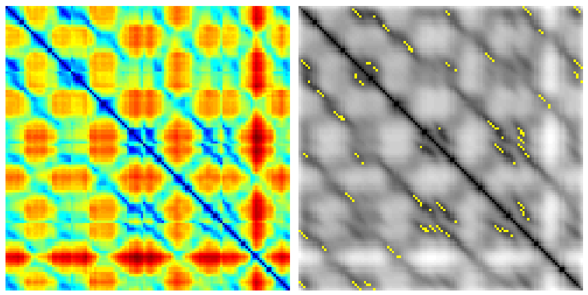

Figure 16.4: Shape similarity matrix and identified transitions for motions *Jog*, *Jog2Walk*, *Walk* and *Walk2Jog* (left to right and top to bottom) .

16.3 4D Parametric Motion Graphs

Parametric Control of Mesh Sequences

Parametric 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 parametrization of skeletal motion capture have previously been introduced [Kovar et al. 02, Rose et al. 98] based on linear interpolation of joint angles. Analogously, as described by Casas *et al.* [Casas et al. 12b], parametric animation of mesh sequences can be achieved by interpolation between mesh sequences.

This requires temporal alignment of frames across multiple 3D video mesh sequences, such that all frames have a consistent mesh connectivity and vertices correspond to the same surface point over time. The set of temporally aligned 3D video sequences are referred to as 4D video [Budd et al. 13b], see Chapter 11 for further details.

Given a set of N temporally aligned 4D mesh sequences $\mathbf{M} = \{M_i(t)\}_{i=1}^N$ of the same or similar motions (e.g. high jump and low jump) we are interested in parametric control by blending between multiple mesh sequences

$$M_B(t, \mathbf{w}) = b(\mathbf{M}, \mathbf{w}) \tag{16.14}$$

where $\mathbf{w} = \{w_i\}_{i=1}^N, w_i \in [0..1]$ is a vector of weights for each input motion and $b()$ is a mesh sequence blending function. We require this function to perform at online rates, $\geq 25\ Hz$, and the resulting mesh $M_B(t, \mathbf{w})$ to maintain the captured non-rigid dynamics of the source $\{M_i(t)\}_{i=1}^N$ meshes.

Three steps are required to achieve high-level parametric control from mesh sequences: time-warping to align the mesh sequences; non-linear mesh blending of the time-warped sequences; and mapping from low level blending weights to high-level parameters (speed, direction, etc.).

Sequence time-warping: Each 4D video sequence of related motions (e.g. walk and run) is likely to differ in length and location of corresponding events, for example foot-floor contact. Thus, the first step for mesh sequence blending is to establish the frame-to-frame correspondence between different sequences. Mesh sequences $M_i(t)$ are temporally aligned by a continuous time-warp function $t = f(t_u)$ [Witkin and Popovic 95] which aligns corresponding poses of related motions prior to blending such that $t \in [0, 1]$ for all sequences. Temporal sequence alignment helps in preventing undesirable artefacts such as foot skating in the final blended sequence.

Real-time Mesh Blending: Previous research in 3D mesh deformation concluded that linear-methods for mesh blending, despite being computation-

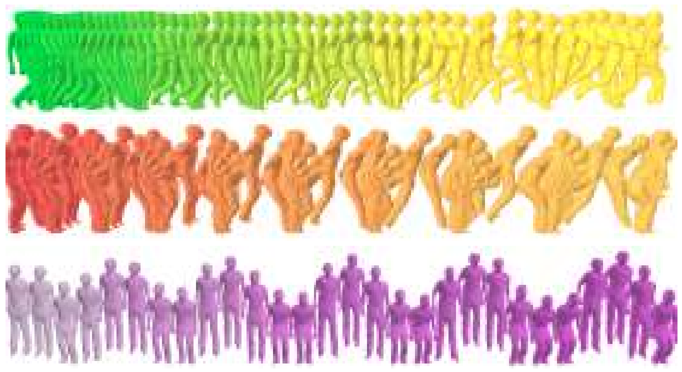

Figure 16.5: Mesh sequence parametrization results [Casas et al. 12b]. In each row, two input motions are interpolated, generating in-between parametric motion control. Top row, speed control; middle row, jump length control; bottom row, jump height control.

ally efficient, may result in unrealistic results [Lewis et al. 00]. Non-linear methods based on differential surface representation [Botsch and Sorkine 08] overcome this limitation, achieving plausible surface deformation. However, the price paid is a significant increase in processing requirements, hindering their use for online applications. Piece-wise linear blending methods [Casas et al. 13] have demonstrated successful results for online applications by precomputing a set of non-linear interpolated meshes. At run time, any requested parametric mesh is computed by linearly blending the relevant offline interpolated meshes. This results in an approximation to the robust non-linear blending approach with a computational cost similar to the linear approach.

High-level parametric control: High-level parametric control is achieved by learning a mapping function $f(\mathbf{w})$ between the blend weights $\mathbf{w}$ and the user specified motion parameters $\mathbf{p}$. A mapping function $\mathbf{w} = f^{-1}(\mathbf{p})$ is learnt from the user-specified parameter to the corresponding blend weights required to generate the desired motion because the blend weights $\mathbf{w}$ do not provide an intuitive parametrization of the 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, and height and distance for a jump. As proposed by Ahmed *et al.* [Ahmed et al. 01], the inverse mapping function $f^{-1}()$ from parameters to weights can be constructed by a discrete sampling of the weight space $\mathbf{w}$ and evaluation of the corresponding motion parameters $\mathbf{p}$.

Figure 16.5 presents three examples of mesh sequence parametriza-

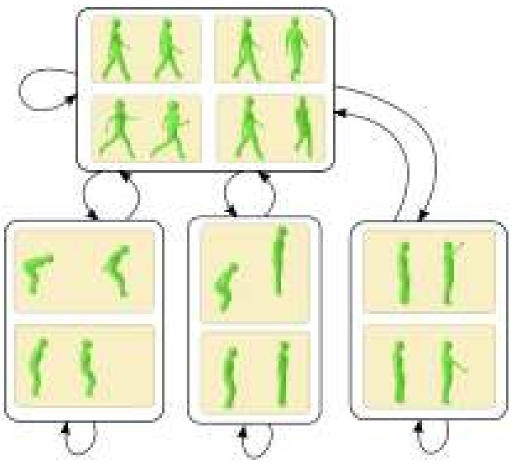

Figure 16.6: Illustration of a 4D Parametric Motion Graph. 10 different motions are combined to create a 4-node 4DPMG, enabling speed, direction, jump and reach parametric control. Each node represents an independent parametric space, edges represent links between them.

tion, enabling control of speed, jump length and jump height, respectively. Change in colour represents change in parametrization. In each example, two motions are interpolated to generate in-between poses, enabling interactive parametric control of the motion.

4D Parametric Motion Graphs

Given a dataset of mesh sequences for different movements with a consistent mesh structure at every frame, referred to as 4D video, parametric control of the motion can be achieved by combining multiple sequences of related motions, as discussed in Section 16.3. This gives a parametrized motion space controlled by high-level parameters (for example walk speed/direction or jump height/length). However, parametric motions can only be synthesized by combining semantically similar sequences (i.e: walk and run), whereas the interpolation of non-similar sequences, such as jump and walk, would fail in generating a human-realistic motion. Therefore, in order to fully exploit a dataset of 4D video sequences, methods for linking motions performing different actions are required.

An approach referred to as 4D Parametric Motion Graph, 4DPMG, [Casas et al. 12a,Casas et al. 13] tackles this problem by employing a graph representation that encapsulates a number of independent mesh parametric spaces as well as links between them, to enable real-time parametric control

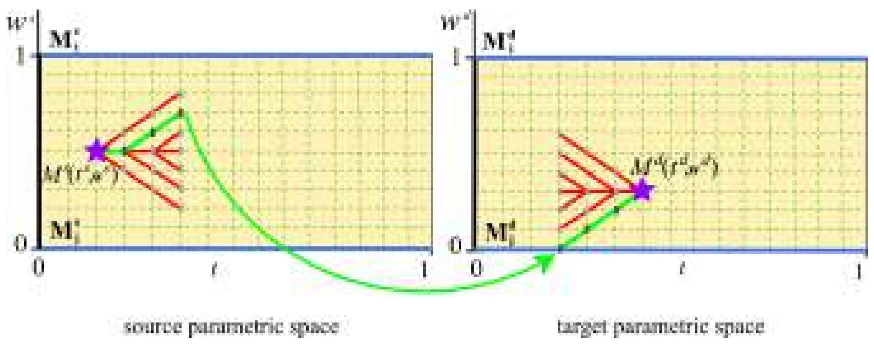

Figure 16.7: Diagram depicting the process of finding a transition between two parametric meshes, $M^s(t^s, \mathbf{w}^s)$ and $M^d(t^d, \mathbf{w}^d)$, here arbitrarily selected for illustration purposes, and marked with a purple star inside their corresponding parametric spaces. Two trellis, depicted in red in each parametric space, are built to find the sets of transitions mesh candidates. Optimal transition path P_{opt}, also arbitrarily selected in this diagram, is highlighted in green over the trellis.

of the motion. Figure 16.6 presents an illustration of a 4DPMG created using 10 different motions used to create 4 parametric spaces.

The nodes of a 4DPMG are created by the combination of similar motions. Each node can be considered as an independent parametric motion space created as described in Section 16.3. The problem is then how to find transitions between parametric motions at run time. Natural transitions require a similar shape and non-rigid motion between the linked meshes, otherwise the resulting animation will not look realistic due to sudden change of the character's speed and pose. In the literature, parametric transitions for skeletal data have been approached by precomputing a discrete set of *good* transitions, evaluating the similarity in pose and motion between pairs of the linked motion spaces [Heck and Gleicher 07]. However, precomputation of a fixed set of transition points may result in a relatively high latency due to the delay between the current pose and the next pre-computed good-transition pose.

In a 4DPMG, to evaluate the best transition path P_{opt} between two parametric points, a cost function representing the trade-off between similarity in mesh shape and motion at transition, $E_S(P)$, and the latency, $E_L(P)$, or delay in transition for a path P, is optimized

$$P_{opt} = \arg\min_{P \in \Omega} \left(E_S(P) + \lambda E_L(P)\right) \tag{16.15}$$

where λ defines the trade-off between transition similarity and latency. The transition path P is optimized over a trellis of frames starting at the

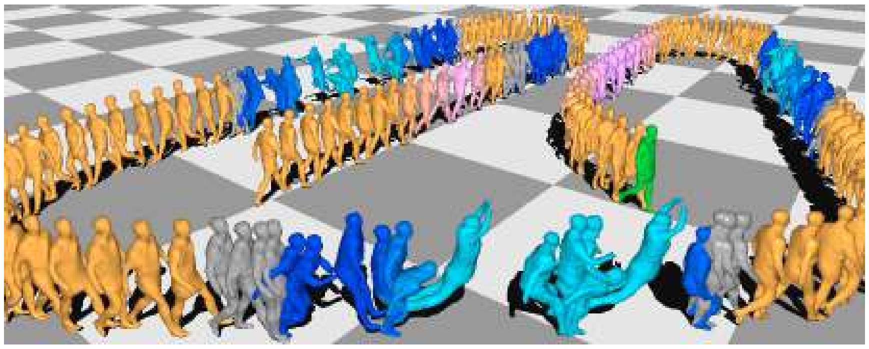
Figure 16.8: An interactively controlled character generated using 4DPMG, combining walk, jog, left turn, right turn, short jump and long jump motions. Grey meshes indicate transitions between parametric spaces.

current frame $M^s(t^s, \mathbf{w}^s)$ in the source motion space and a trellis ending at the target frame $M^d(t^d, \mathbf{w}^d)$ in the target motion space, where M^s and M^t are interpolated meshes as defined in Equation 16.14. The trellis is sampled forward in time at discrete intervals in time Δt and parameters $\Delta \mathbf{w}$ 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 possible 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. Online mesh similarity computation is prohibitely expensive for large sets of transition candidates. To overcome this, Casas *et al.* [Casas et al. 12a] proposed a method based on precomputing a set of similarities between the input data, and interpolate them at runtime to approximate any requested parametric pose similarity.

Figure 16.8 presents a parametric character interactively synthesized combining 10 different motions. Qualitative and quantitative results using 4DPMG have demonstrated [Casas et al. 12a] realistic transitions between parametric mesh sequences, enabling interactive parametric control of a 3D-mesh character.

4D Video Textures

The 4D Parametric Motion Graph allows real-time interactive control of a character's motion to produce novel animation sequences. However, this is missing the realistic appearance of the source video for free-viewpoint rendering. To achieve video-realistic rendering of the dynamic appearance for 4D parametrically controlled animation a representation referred to as 4D Video Textures, 4DVT, is used [Casas et al. 14].

With 4DVT, appearance for an intermediate motion is produced by aligning and combining multiple-view video from the input examples to produce plausible video-realistic dynamic appearance corresponding to the modified movement. As the character motion changes, so does the dynamic appearance of the rendered view reflecting the change in motion.

A 4D video $F(t) = \{V(t), M(t)\}$ combines multiple view video sequences $V(t) = \{I_c(t)\}_{c=1}^{C}$ with C camera views with a 4D proxy of the dynamic surface shape represented as a mesh sequence $M(t)$, where vertices correspond to the same surface point over time. This form of representation has previously been employed for free-viewpoint video rendering of dynamic scenes [Carranza et al. 03,Starck and Hilton 07b,de Aguiar et al. 08a,Vlasic et al. 08a]. Free-viewpoint video renders a novel video sequence $I(t, v)$ for a viewpoint v from the set of input videos $V(t)$ using the mesh sequence $M(t)$ as a geometric proxy [Buehler et al. 01b]. The objective of free-viewpoint video is to maintain the visual realism of the source video whilst providing the flexibility to interactively control the viewpoint. However, free-viewpoint video is limited to the replay of the captured performance and does not allow for any change in scene motion.

4DVT overcomes this limitation [Casas et al. 14]. Given a set of motion control parameters $\mathbf{w}$ and viewpoint v, we aim to render a novel video $I(t, \mathbf{w}, v)$:

$$I(t, \mathbf{w}, v) = h(F_1(t), ..., F_N(t), \mathbf{w}, v), \tag{16.16}$$

where $h(.)$ is a function which combines the source 4D videos according to the specified motion parameters $\mathbf{w}$ and viewpoint v. The rendered video $I(t, \mathbf{w}, v)$ should preserve the visual quality of both the scene appearance and motion.

A two-stage approach is used to synthesize the final $I(t, \mathbf{w}, v)$ video. First, a 4D shape proxy $M(t, \mathbf{w})$ is computed by the combination of the input mesh sequences using the approach presented in Section 16.3. Finally, exploiting the known vertex-to-vertex correspondence across sequences, view-dependent rendering of the source videos $V_i(t)$ is perform using the same 4D shape proxy $I(t, \mathbf{w}, v)$. The output video $I(t, \mathbf{w}, v)$ is generated based on real-time alignment of the rendered images.

Qualitative and quantitative results have demonstrated that 4DVT succesfully enables the creation of video characters with interactive video and motion control and free-viewpoint rendering which maintain the visual quality and dynamic appearance of the source videos [Casas et al. 14].

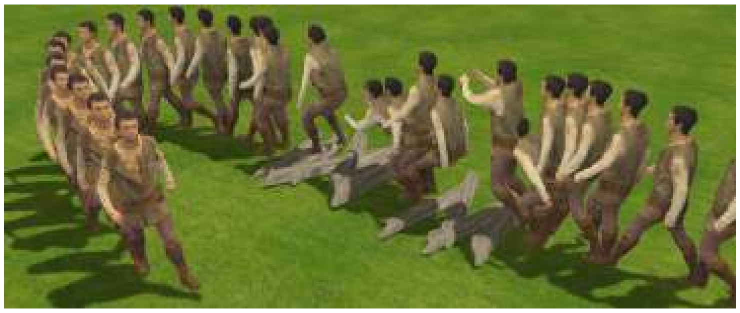

Figure 16.9: A character animated using 4D Video Textures jumping over obstacles of varying length. Notice the texture detail in face and clothing.

Intermediate motions are rendered with plausible dynamic appearance for the whole-body, cloth wrinkles and hair motion. Figure 16.9 presents an animation interactively created combining 4D Video Textures and 4D Parametric Motion Graph, a character combines different styles of jumps and walks to avoid the obstacles, and finally performs a sharp left turn.

16.4 Summary

Recent research on multi-camera performance capture has enabled detailed reconstruction of motions as 3D mesh sequences with a temporally coherent geometry. This chapter has presented a set of methods to allow the reutilization of reconstructed motions, with the goal of authoring novel character animation while maintaining the realism of the captured data.

A representation referred to as Surface Motion Graph has been introduced to synthesize novel animations that satisfy a set of user defined constraints on movement, location and timing. A graph optimization technique is used to find transitions between originally captured sequences that best satisfy the animation constraints. A 3D shape similarity descriptor is used to evaluate the transitions between different motions.

A parametric approach referred to as 4D Parametric Motion Graph has been presented to synthesize novel interactive character animation by concatenating and blending a set of mesh sequences. This enables real-time control of a parametric character that preserves the realism of the captured geometry. Video-realistic appearance for novel parametric motions is generated at run-time using 4D Video Textures, a rendering technique that combines multi-camera captured footage to synthesize textures that changes with the motion while maintaining the realism of the captured video.