Animation Control of Surface Motion Capture

Margara Tejera, *Student Member, IEEE,* Dan Casas, *Student Member, IEEE,* and Adrian Hilton, *Member, IEEE*

***Abstract*—Surface motion capture (SurfCap) of actor performance from multiple view video provides reconstruction of the natural nonrigid deformation of skin and clothing. This paper introduces techniques for interactive animation control of SurfCap sequences which allow the flexibility in editing and interactive manipulation associated with existing tools for animation from skeletal motion capture (MoCap). Laplacian mesh editing is extended using a basis model learned from SurfCap sequences to constrain the surface shape to reproduce natural deformation. Three novel approaches for animation control of SurfCap sequences, which exploit the constrained Laplacian mesh editing, are introduced: 1) space–time editing for interactive sequence manipulation; 2) skeleton-driven animation to achieve natural nonrigid surface deformation; and 3) hybrid combination of skeletal MoCap driven and SurfCap sequence to extend the range of movement. These approaches are combined with high-level parametric control of SurfCap sequences in a hybrid surface and skeleton-driven animation control framework to achieve natural surface deformation with an extended range of movement by exploiting existing MoCap archives. Evaluation of each approach and the integrated animation framework are presented on real SurfCap sequences for actors performing multiple motions with a variety of clothing styles. Results demonstrate that these techniques enable flexible control for interactive animation with the natural nonrigid surface dynamics of the captured performance and provide a powerful tool to extend current SurfCap databases by incorporating new motions from MoCap sequences.**

***Index Terms*—3-D reconstruction, 3-D video, surface motion capture (SurfCap), video-based animation.**

I. Introduction

ADVANCES in 3-D actor performance capture from multiple view video [1]–[4] have achieved detailed reconstruction and rendering of natural surface dynamics as mesh sequences, allowing free-viewpoint rendering of the captured performance with a visual realism approaching that of the captured video. These approaches are restricted to replay of the captured surface motion and do not allow animation control to adjust the movement or generate novel motions. In this paper, we introduce control techniques for surface motion capture (SurfCap) sequences, which enable reuse of the captured content by interactive artistic editing and combination with skeleton-driven animation.

Conventional marker-based skeletal motion capture (MoCap) of actor performance is widely used in animation for film and games as it reproduces natural character motion. A variety of control techniques for skeletal MoCap have been introduced which support artistic editing of the captured motion and interactive animation of characters for games. Due to the use of markers, MoCap can only reconstruct the motion of a sparse set of 3-D surface points that are used to estimate the underlying skeletal motion. In contrast, SurfCap captures the full nonrigid surface deformation allowing detailed reproduction of surface dynamics for skin, clothing, and hair.

This paper introduces control techniques for SurfCap sequences, analogous to those available for MoCap, to enable flexible reuse in animation, while preserving the detailed surface deformation. The challenge for control of surface motion is that surfaces generally have several thousand degrees of freedom (three for each vertex), whereas skeletal motion based on the human anatomy is typically represented with around 60 joint angles.

Animation from 3-D performance capture has previously been achieved by concatenation of segments of multiple captured mesh sequences based on manually defined transitions [5]. Automatic transition graph construction and path optimization have been introduced [6], allowing offline keyframe animation. The level of movement control in these approaches is limited to transition between the captured movement sequences. Recent work [7] has exploited skeletal tracking of mesh sequences to allow increased manipulation of the captured movement with a skeletal control rig. Novel skeletal motion is used to index a 3-D video database, allowing rendering of new motion sequences. Recent research [8]–[10] has introduced techniques for high-level parametric surface animation control by nonlinear blending of multiple SurfCap sequences for related motion. In this paper, we build on this work by introducing methods to extend the range of motion beyond the original captured sequences enabling greater artistic freedom in animation from SurfCap data.

We present an interactive framework for control of mesh animation from SurfCap performance sequences. Three approaches are introduced to achieve the flexibility of animation using skeletal MoCap with the detailed surface deformation of SurfCap: 1) space–time editing with learned surface deformation; 2) skeleton-driven animation with a learned surface deformation space; and 3) hybrid combination of skeletal-driven and captured surface motion to extend the range of

Manuscript received April 7, 2012; revised December 7, 2012; accepted April 10, 2013. This work was supported in part by the Engineering and Physical Sciences Research Council platform, under Grant EP/F02827X/1, and EU projects RE@CT (ICT-288369) and SCENE (ICT-287693). The skeletal data used in this paper was provided by the Carnegie Mellon University Motion Capture Database (mocap.cs.cmu.edu), which was created with funding from the National Science Foundation under Grant EIA-0196217. This paper was recommended by Associate Editor S. Zafeiriou.

The authors are with the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, GU2 7XH, U.K. (e-mail: m.tejerapadilla@surrey.ac.uk; d.casasguix@surrey.ac.uk; a.hilton@surrey.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2013.2260328

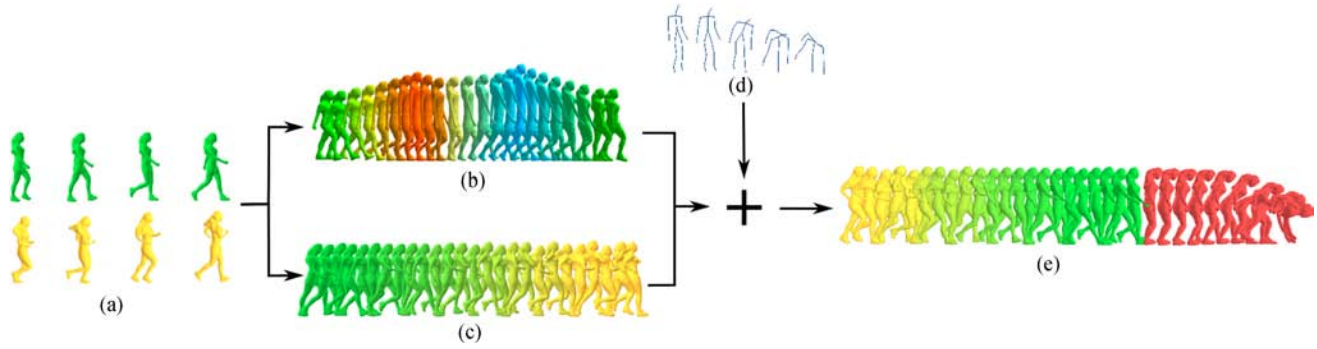

Fig. 1. Overview of surface motion animation control and integration with skeleton-driven animation. Space–time mesh editing and motion parameterization are applied to SurfCap input motions in order to synthesize novel mesh sequences. Subsequently, skeleton-driven animations are created and linked with the SurfCap sequences, resulting in hybrid mesh sequences with plausible surface deformation throughout all frames. (a) SurfCap input motions. (b) Space–time editing. (c) Parameterized motion. (d) MoCap input motion. (e) Hybrid skeletal surface animation control.

movement. This novel control techniques are combined with parametric control of SurfCap motion to provide a framework for flexible animation from SurCap data. An overview of the framework for surface motion animation is presented in Fig. 1.

Contributions are as follows.

Space–time mesh editing with a learned deformation space: Space–time editing of skeletal MoCap [11], [12] provides a powerful tool for artistic manipulation by propagating key-frame edits across the sequence. Application of similar approaches to animated mesh sequences [13], [14] by key-frame editing of meshes provides low-level control. In this paper, we introduce a space–time mesh sequence editing approach that extends previous work by the incorporation of a learned surface deformation space from the SurfCap sequences. This learned model constrains the edited surface deformation to be similar to previously seen examples which preserves the underlying anatomical structure and natural deformation of the captured performance, while allowing animation control, as shown in Fig. 2. Two novel nonlinear approaches are presented for propagation of key-frame edits which preserve the natural surface deformation. Space–time editing of SurfCap provides a flexible tool for interactive control allowing both low-level constraints and artistic stylization.

Skeleton-driven animation with learned surface deformation: Conventionally, MoCap sequences are used to animate characters with linear blend skinning producing unrealistic surface deformation which contrasts with the detailed deformation of SurfCap. To overcome this limitation, we exploit the learned surface deformation from SurfCap sequences to produce plausible detailed nonrigid surface deformation from MoCap, as depicted in Fig. 10.

Hybrid skeletal and surface motion control: To extend the range of character animation beyond the captured performance, we combine SurfCap animation with existing databases of MoCap sequences which include a wide variety of motions. First, the skeleton-driven animation technique described previously is applied to the desired MoCap sequence. The resulting animation is subsequently linked to a SurfCap sequence by automatically finding suitable transition points between the two mesh sequences [9], [10]. Finally, space–time mesh editing is employed to allow seamless transitions between SurfCap- and MoCap-driven sequences and to overcome the possible dissimilarity of the linking poses. This hybrid approach greatly extends the range of motion and allows integrated animation control from skeleton and surface performance capture.

In order to extend the flexibility of these animation techniques, parametric control of surface motion [8] is incorporated into our framework. This approach enables the creation of novel mesh sequences from a SurfCap database, multiplying the range of motions than can be synthesized within the presented hybrid SurfCap–MoCap animation framework. Results of each approach for animation control are presented on public databases of SurfCap actor performance capture [2] for a variety of motions and clothing styles. Hybrid control is used to produce animation combining SurfCap sequences with a variety of skeletal MoCap sequences from publicly available archives [15]. Evaluation demonstrates flexible animation control preserving the captured nonrigid surface deformation of SurfCap sequences.

II. Related Work

Reuse and editing of skeletal data: Since the introduction of marker-based technologies for skeletal performance capture to the entertainment industry in the early 1990s, a range of techniques to support editing and reuse have been developed. Brundelin and Williams [16] introduced parametric motion control by interpolating pairs of skeletal motions. Parametric motion synthesis was extended to blending multiple examples to create a parameterized skeletal motion space [17]–[20]. This allows continuous interactive motion control through high-level parameters such as velocity for walking or hand position for reaching.

Other research has followed the traditional animation process: first, edit a set of key frames of the sequence, creating a set of poses that satisfy the constraints set by the user; and second, create in-between poses that preserve the naturalness of the desired motion. The work of Gleicher [11] and Lee and Shin [12] are examples of space–time editing approaches. Gleicher [11] solves for both space and time constraints simultaneously. Lee and Shin [12] modify the poses of the skeleton in the key frames by means of an inverse kinematics (IK)

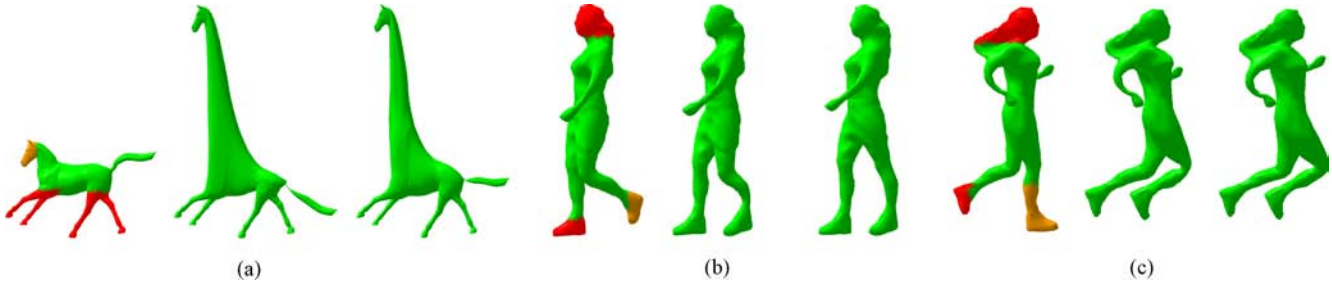

Fig. 2. Three examples of the effect of incorporating a learned space of deformations into the Laplacian framework. (Left) Original mesh with user-specified constraints colored in orange and red. (Center) Laplacian editing. (Right) Laplacian editing with learned deformation space. (a) Laplacian editing (center) causes the tail to fold over itself. Using the learned deformation space (right) preserves the tail shape. Dataset courtesy of [39]. (b) Using the learned deformation space (right) preserves the leg shape avoiding mesh collapse which occurs with Laplacian editing (center). (c) Using learned deformation space (right) preserves the leg shape avoiding mesh thinning which occurs with Laplacian editing (center).

solver, and then apply a multilevel B-spline approximation for the interpolation of poses.

As well as depicting an action, characters communicate feelings to the viewer and can perform the same action in different styles. Brand and Hertzmann [21] presented the first method to automatically separate the motion style from the structure by an unsupervised learning process based on hidden Markov models. Following the same learning approach, Hsu *et al.* [22] train style translation models that describe how to transition from one motion to another, and Shapiro *et al.* [23] apply independent component analysis to decompose the motion. Min *et al.* [24] construct a generative human motion model using multilinear data analysis techniques. This model is driven by two parameters and their adjustment produces personalized stylistic human motion.

Taking inspiration from the aforementioned editing techniques, in this paper, we present a framework that not only enables synthesis of novel mesh sequences, but also allows the extension of SurfCap databases by integration with skeletal-driven animation.

Multiple view performance capture and reconstruction: Kanade and Rander [25] pioneered the reconstruction of 3-D mesh sequences of human performance for free-viewpoint replay with the virtualized reality system using a 5 m dome with 51 cameras. Multiple view video reconstruction results in an unstructured mesh sequence with an independent mesh at each frame. Advances in performance capture from video have enabled reconstruction of mesh sequences for human performance capturing the detailed deformation of clothing and hair [1]–[4]. These approaches achieve a free-viewpoint rendering quality comparable to the captured video but are limited to performance replay.

A critical step for editing and reuse of captured mesh sequences is temporal alignment to obtain a consistent mesh structure with surface correspondence over time referred to as SurfCap [2]. A number of approaches have been proposed for alignment of mesh sequences based on sequential frame-to-frame surface tracking. These can be categorized into two methodologies: 1) model-based approaches, which align a prior model of the surface with successive frames [3], [4], [26]; and 2) surface tracking or scene flow approaches, which do not assume prior knowledge of the surface structure [27]–[29]. Sequential alignment approaches have three inherent limitations: 1) accumulation of errors in frame-to-frame alignment resulting in drift in correspondence over time; 2) gross errors for large nonrigid deformations, which occur with rapid movements requiring manual correction; and 3) they are limited to alignment across single sequences. Recently, nonsequential alignment approaches [30], [31] have been introduced to overcome these limitations, allowing the construction of temporally coherent 3-D mesh sequences from multiple view performance capture database. In this paper, we exploit these techniques to obtain SurfCap sequences with a consistent mesh topology and vertex correspondence for editing and animation control.

Reuse and editing of 3-D video data: The lack of temporal coherence in the mesh sequence has prohibited the development of simple methods for manipulation. Animation from databases of mesh sequences of actor performance has been demonstrated by concatenating segments of captured sequences [5], [6], which is analogous to previous example-based approaches to concatenative synthesis used for 2-D video [32]–[34]. Recently, example-based approaches through resampling video sequences have been extended to body motion [7], [34] allowing offline animation via key frame or skeletal motion. In [7], model-based skeletal tracking was used to resample segments from a database of video sequences based on pose allowing photorealistic rendering with skeletal control. These approaches preserve the realism of the captured sequences in rendering but are limited to replay segments of the captured motion examples and do not allow the flexibility of conventional animation. Recent advances in nonsequential alignment have allowed the introduction of techniques for parameterization and online interactive animation techniques to control the character's motion [9], [10].

Analogous to the skeletal IK methods, several mesh editing techniques have been developed. They generally consist of a global optimization that tries to preserve the local differential properties of the mesh, while satisfying user constraints. A comprehensive comparison between these methods is provided in [35]. Sumner *et al.* [36] formulate the problem as a least squares minimization that manipulates the deformation

gradients of the triangles, which describe their transformation with respect to a reference pose. A nonlinear feature space is constructed using the deformation gradients as feature vectors and applying polar decomposition and exponential maps to avoid naive linear blending of poses, which would lead to unnatural results. Laplacian-based approaches [37] define a linear operator according to the connectivity and the area of the triangles of the meshes. The application of this operator yields a set of differential coordinates whose direction approximates the direction of the local normals of the triangles, and whose magnitude is proportional to the local mean curvature. The main drawback of these methods is having to deal with rotations explicitly. Lipman *et al.* [38] introduced a mesh representation based on rotation-invariant linear coordinates that addresses this problem: linear shape interpolation using this representation handles rotations correctly.

Following the key-frame editing scheme, the mesh editing problem can be extended to sequences. Xu *et al.* [13] introduced an alternating least-squares method based on rotation-invariant linear coordinates [38] demonstrating natural deformation. Constraints at key frames are propagated by a handle trajectory editing algorithm, obtaining an overall natural-looking motion. Kircher and Garland [14] presented a differential surface representation which encodes first- and second-order differences of each vertex with respect to its neighbors giving rotation and translation invariance. These differences are stored in connection maps, one per triangle, which allow the development of motion processing operations with better results than vertex-based approaches. A limitation of these approaches is the lack of a mechanism to preserve the underlying structure of the character's surface. In this paper, we overcome this limitation by enhancing the mesh deformation with the incorporation of a learned deformation space that constrains the resulting surface based on previously observed examples and enables the synthesis of novel sequences with natural-looking deformations.

III. Space-Time Editing of Surface Motion

Skeletal motion sequences explicitly represent the anatomical structure which is preserved during editing. For mesh sequences, the underlying physical structure is implicit, requiring editing to be constrained to reproduce anatomically correct deformations. Based on a Laplacian surface deformation scheme [35], [37], we present a mesh sequence editing algorithm that incorporates a learned deformation space and preserves the anatomical structure of the character during editing. The computation of this learned model is performed by finding a set of basis vectors that best represents the space of differential coordinates defined by the 3-D performance capture data. This effectively combines previous free-form mesh sequence editing [13], [14] with learned spaces of mesh deformation [36] within a Laplacian mesh editing framework [35], [37].

A. Laplacian Mesh Editing Framework

Laplacian mesh editing is based on a differential representation of the mesh which allows local mesh properties to be encoded. The gradient of the triangles' basis functions ϕ_i yields a 3×3 matrix $\mathbf{G_j}$ for each of the triangles [35]

$$\mathbf{G}_j = (\nabla\phi_1, \nabla\phi_2, \nabla\phi_3) \cdot (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^\top \tag{1}$$

$$= \begin{pmatrix} (\mathbf{p}_1 - \mathbf{p}_3)^\top \\ (\mathbf{p}_2 - \mathbf{p}_3)^\top \\ \mathbf{n}^\top \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{pmatrix} \tag{2}$$

where $\mathbf{p_1}, \mathbf{p_2}$ and $\mathbf{p_3}$ are the vertex positions and $\mathbf{n}$ is the unit normal of the jth triangle. Applying this gradient to every triangle of the mesh, we can construct a matrix $\mathbf{G}$ of size $3m \times n$, where n is the number of vertices and m the number of triangles [40]

$$\begin{pmatrix} \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_m \end{pmatrix} = \mathbf{G} \begin{pmatrix} \mathbf{p}_1^\top \\ \vdots \\ \mathbf{p}_n^\top \end{pmatrix}. \tag{3}$$

Let $\mathbf{A}$ be a diagonal weighting matrix containing the areas of the triangles, the matrix $\mathbf{G}^\top\mathbf{A}$ represents the discrete divergence operator, and the discrete Laplace–Beltrami operator $\mathbf{L}$ can be constructed by performing the following multiplication: $\mathbf{L} = \mathbf{G}^\top\mathbf{A}\mathbf{G}$ [35]. Given a mesh, its differential coordinates can be obtained by multiplying the Laplacian operator by its absolute coordinates $\delta(\mathbf{x}) = \mathbf{L}\mathbf{x}$, $\delta(\mathbf{y}) = \mathbf{L}\mathbf{y}$, and $\delta(\mathbf{z}) = \mathbf{L}\mathbf{z}$. If we assume the addition of positional soft constraints $\mathbf{x_c}$, the $\tilde{\mathbf{x}}$ absolute coordinates of the reconstructed mesh (the same applies for $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{z}}$) can be computed in the least-squares sense [37]

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\|\mathbf{L}\mathbf{x} - \delta(\mathbf{x_o})\|^2 + \|\mathbf{W_c}(\mathbf{x} - \mathbf{x_c})\|^2 \right) \tag{4}$$

where $\mathbf{x_o}$ are the coordinates of the original mesh and $\mathbf{x_c}$ are the soft constraints on vertex locations given by the feature correspondence with a diagonal weight matrix $\mathbf{W_c}$. This equation allows the reconstruction of a mesh by means of the Laplacian operator $\mathbf{L}$ that, due to its linear nature, does not account for changes in rotation. To allow nonlinear interpolation of rotation, an iterative approach is taken [41]: in each step of the minimization, the changes in rotation of each triangle are computed and the Laplacian operator is updated accordingly. The nonrotational part of the transformations is discarded in order to help the preservation of the original shape of the triangles.

B. Laplacian Editing With a Learned Deformation Space

In this paper, we introduce a novel mesh editing framework based on the Laplacian deformation scheme presented in Section III-A. The novelty resides in incorporating into the algorithm the previously observed deformations of the character. This constrains the possible solutions of the deformation solver, ensuring the preservation of the captured motion characteristics and underlying anatomical structure of the actor performance.

For a sequence of meshes $\{M(t_i)\}_{i=1}^{F}$, where F is the number of frames, the mesh motion deformation space is built by taking each mesh represented in differential coordinates as a deformation example. Our data matrix $\mathbf{M}$ is built by placing

Fig. 3. Key-frame editing. Original horse (left). Original horse (center) showing the constrained vertices: in orange the handles and in red the fixed vertices. Edited horse (right). (Dataset courtesy of [39].)

the concatenated $\delta(\mathbf{x})$, $\delta(\mathbf{y})$, and $\delta(\mathbf{z})$ differential coordinates of each example in its rows

$$\mathbf{M} = \begin{pmatrix} \delta_1^\top(\mathbf{x}) & \delta_1^\top(\mathbf{y}) & \delta_1^\top(\mathbf{z}) \\ \delta_2^\top(\mathbf{x}) & \delta_2^\top(\mathbf{y}) & \delta_2^\top(\mathbf{z}) \\ \vdots & \vdots & \vdots \\ \delta_F^\top(\mathbf{x}) & \delta_F^\top(\mathbf{y}) & \delta_F^\top(\mathbf{z}) \end{pmatrix}. \tag{5}$$

The data matrix is centered obtaining $\mathbf{M_c} = \mathbf{M} - \bar{\mathbf{M}}$, where $\bar{\mathbf{M}}$ is a $F \times 3n$ 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, a singular value decomposition is performed over the matrix $\mathbf{M_c}$: $\mathbf{M_c} = \mathbf{UDV}^\top$, where matrix $\mathbf{V}$ is a $3n \times F$ with a vector of the basis in each of its columns. The first l eigenvectors $\mathbf{e}_k$ representing 95% of the variance are kept, which gives a linear basis of the form

$$\delta(\mathbf{r}) = \bar{\delta} + \sum_{k=1}^{l} r_k \mathbf{e}_k = \bar{\delta} + \mathbf{Er} \tag{6}$$

where r_k are scalar weights for each eigenvector, $\mathbf{r}$ is an l-D weight vector, and $\mathbf{E}$ is an $3n \times l$ matrix whose columns are the first l eigenvectors of length $3n$. Space–time editing of key frames in the mesh sequences is performed using a constrained Laplacian mesh editing within the space of deformations $\delta(\mathbf{r})$. From (4), we have

$$\tilde{\mathbf{r}}, \tilde{\mathbf{x}} = \arg\min_{\mathbf{r},\mathbf{x}} (\|\mathbf{Lx} - \delta(\mathbf{r})\|^2 + \|\mathbf{W_c}(\mathbf{x} - \mathbf{x_c})\|^2). \tag{7}$$

Equation (7) allows interactive editing of a key-frame mesh $M(t_k)$ to satisfy a set of user-defined constraints $\mathbf{x_c}$ resulting in a modified mesh $M'(t_k)$ with vertices $x'(t_k)$. To construct a basis with respect to the mesh $M(t_i)$ for each frame in the mesh sequence, the Laplacian $\mathbf{L}_i$, defined according to the discrete gradient operator matrix $\mathbf{G}_i$, is used as a reference in the construction of the data matrix $\mathbf{M}_i$ such that $\delta_i(\mathbf{x}) = \mathbf{L}_i\mathbf{x}$. Constructing a local basis defines changes in shape in the learned motion space taking the reference frame as the origin.

Examples of the effect of the basis are depicted in Fig. 2. Deformations applying the learned deformation space within the Laplacian framework preserve the surface details and the underlying structure of the character. This avoids artifacts such as thinning, unnatural bending of limbs, and collapsing of the mesh which occur if the Laplacian is not constrained to a learned deformation space.

C. Editing Mesh Sequences in a Learned Deformation Space

A space–time editing pipeline is introduced in order to edit a full sequence of meshes. A set of key frames are edited within the learned Laplacian framework described in the previous section, and subsequently these changes are propagated over a temporal window with the objective of seamlessly incorporating the edited frames into the sequence. User input is necessary to both choose the key frames and select the constrained vertices. Our space–time interface allows selection of any vertex on the mesh as a constraint.

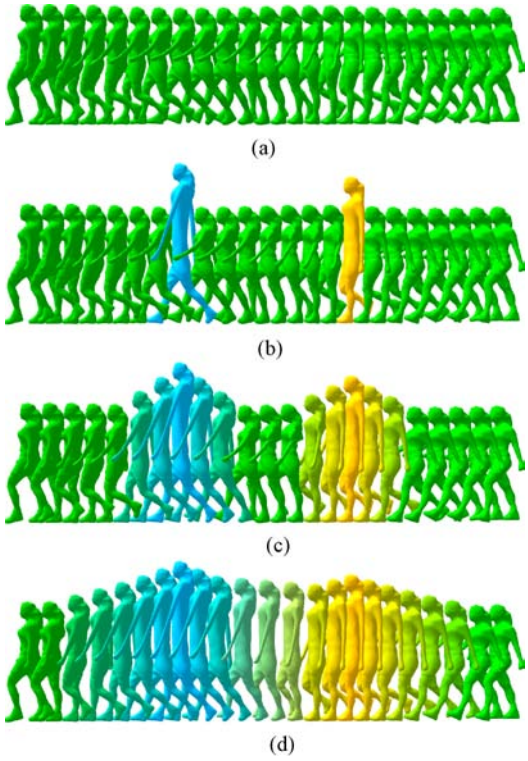

Fig. 4. Space–time editing pipeline for mesh sequence editing. The character height is deformed in an unnatural way during a walk in order to easily depict the influence of the length of the window of propagation. (a) Original sequence. (b) Two edited key frames. (c) Space–time editing with $T_k = 3\,frames$. (d) Space–time editing with $T_k = 6\,frames$

1) *Key-Frame Editing:* Key-frame editing is performed within the Laplacian framework described in Section III-B. During an offline process, each frame of the sequences of a given character is used as the reference frame for computing a space of deformations. In our implementation, all available frames for the character are considered as deformation examples for the construction of the deformation space.

The user interactively selects two sets of vertices: 1) the vertices whose position must stay unchanged during the deformation, and the 2) vertices of the handle, which will be dragged to a desired position. These positional constraints and the space of deformations associated with the given frame are incorporated in (7). Fig. 3 shows an example of a key-frame edit.

2) *Space–Time Propagation:* Changes to the key frames must be propagated over time in order to obtain a natural-looking motion. Three propagation methods are evaluated: 1) linear interpolation; 2) nonlinear interpolation; and 3) constraint interpolation. A discussion and comparison between these methods is included at the end of this section.

Fig. 4 illustrates the process of space–time editing for a walk sequence: a key frame is selected and modified; changes

are then propagated across a temporal window with weights shown by the mesh color. In this example, the characters height is modified in an unnatural way on two key frames to give an example of the space–time propagation which is easily visible. More subtle physically realistic editing examples are included in the results.

a) *Linear interpolation:* Given an edited key-frame mesh $M'(t_k)$ with vertices $x'(t_k)$, edits are temporally propagated to other frames of the mesh sequence $M(t_i)$ with vertices $x(t_i)$ using a spline to define the interpolation weights λ_i for the difference in mesh shape $\Delta_k = (x'(t_k) - x(t_k))$

$$x'(t_i) = x(t_i) + \lambda_i \Delta_k. \tag{8}$$

Multiple key-frame edits can be combined as a linear sum

$$x'(t_i) = x(t_i) + \sum_{k=1}^{K_f} \lambda_{ik} \Delta_k \tag{9}$$

where K_f is the number of key frames. This linear sum allows compositing of changes from multiple frames in the sequence with weighted influence on the shape at a particular frame providing intuitive control over mesh sequence deformation. In practice, weights are interpolated over a temporal window of influence around each key frame $t_k \pm T_k$ which can be selected by the user.

Linear interpolation is computationally efficient but may result in unrealistic deformation such as shortening of limbs. We therefore propose a nonlinear and a constraint interpolation approaches which aim to preserve the mesh structure. A comparative evaluation is presented in Section III-C3.

b) *Nonlinear interpolation:* We propose a nonlinear interpolation method based on the propagation of the triangles transformations between the edited and the original key frame. Given a key-frame mesh $M'(t_k)$ and its original version $M(t_k)$, the transformation that the jth triangle of $M(t_k)$ undergoes to transform into the corresponding triangle in $M'(t_k)$ is computed and polar decomposed into its rotational R and nonrotational S components: $T'^j_k = R'^j_k S'^j_k$. Let q'^j_k be the quaternion associated with R'^j_k and q^j_k the quaternion identity for all j, where the superscript refers to the jth triangle. The interpolated rotation q'^j_i for each frame i of the window of propagation is computed as

$$q'^j_i = slerp(q^j_k, q'^j_k, \lambda_i). \tag{10}$$

Let $S^j_k = I$ for all j, the nonrotational scale/shear part S'^j_i is linearly interpolated

$$S'^j_i = S^j_k + \lambda_i (S'^j_k - S^j_k). \tag{11}$$

Multiple key-frame edits can be combined analogous to (9)

$$q'^j_i = \prod_{k=1}^{K_f} slerp(q^j_k, q'^j_k, \lambda_{ik}) \tag{12}$$

$$S'^j_i = \sum_{k=1}^{K_f} S^j_k + \lambda_{ik}(S'^j_k - S^j_k) \tag{13}$$

where $\prod$ represents quaternion multiplication.

Converting q'^j_i to R'^j_i, a set of transformations $T'^j_i = R'^j_i S'^j_i$ can be computed. Applying these transformations directly to the triangles of $M(t_i)$ would result in an unconnected mesh. The Laplacian deformation framework of (4) is applied to link the triangles back together.

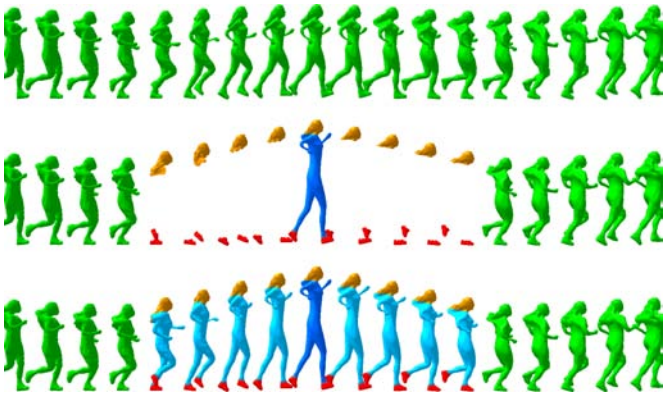

Fig. 5. Illustration of the constraint interpolation method. First row: original sequence. Second row: A key frame has been edited and the constraints (in red and orange) have been interpolated. Third row: for each frame within the window of propagation, the Laplacian deformer of (7) is run to deform the meshes subject to the interpolated constraints.

c) *Constraint interpolation:* The linear and nonlinear propagation methods discussed in the previous paragraphs find the edited meshes $M'(t_i)$ by processing information of the original meshes $M(t_i)$ and the key-frame edits. An alternative method consists in propagating the position of the constraints over the temporal window, and subsequently performing a Laplacian deformation according to (7) to obtain $M'(t_i)$ subject to the interpolated constraints. This offers the advantage of controlling the position of the constrained vertices along the window at the expense of a higher computational cost.

Directly interpolating the constraints coordinates does not guarantee the preservation of the shape of the *submesh* comprised by the selected vertices. Therefore, the nonlinear interpolation method presented in Section III-C2b is applied to compute the position of the constrained vertices over the propagation window. This approach differs from more simplistic approaches where these positions are found by averaging the rotations for each of the triangles [13]. An illustration of the method can be found in Fig. 5.

Although relatively computationally expensive, constraint interpolation provides full control on the positions of the constrained vertices along the window of propagation. This allows fixed constraints to be enforced over a temporal window, for example, on hand or foot location during contact.

3) *Discussion on Interpolation Methods:* A comparison between the three interpolation methods discussed in Section III-C2 is presented in Fig. 6. This shows the propagation of the edited key frame of Fig. 3, from the horse gallop sequence. Applying the linear interpolation causes the front legs to shorten, while the nonrotational and constraint interpolations achieve natural-looking results.

The nonlinear interpolation incorporates the transformation of the mesh triangle by triangle taking into account both the rotation and scale/shear components of the transformations. This avoids artifacts related to linear interpolations, such as shortening of the limbs or distortion of the original shape of the mesh.

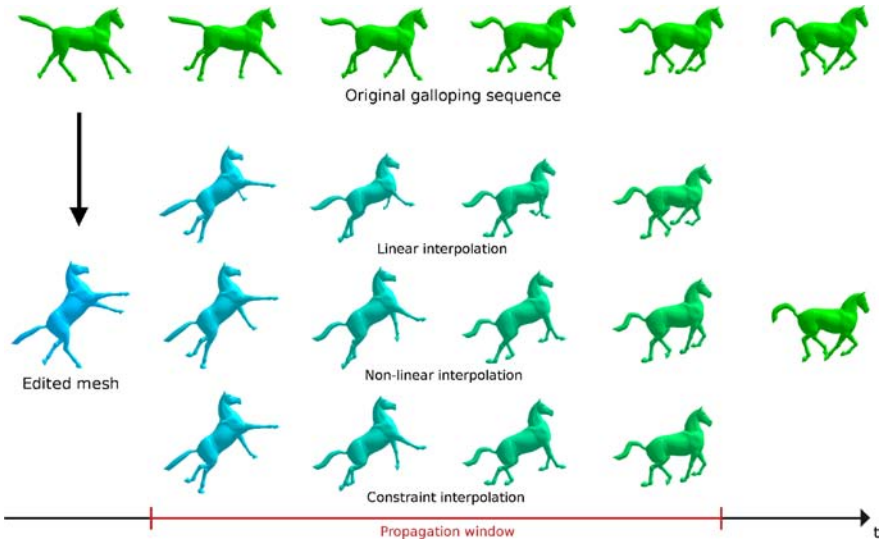

Fig. 6. Comparison of the propagation of an edit using three different interpolation methods. Above the original sequence, below the propagation window for each of the methods. Top row: linear interpolation. Middle row: nonlinear interpolation. Bottom row: constraint interpolation. The edited mesh is shown at the left of the figure, and the frame subsequent to the propagation window is shown at the right. (Dataset courtesy of [39].)

Since applying the constraint interpolation method means deforming each of the meshes within the propagation window subject to a set of constraints, it provides greater control over the position of the constrained vertices along the temporal window. However, it is computationally the most expensive method. Computation times for the propagation of the space–time editing example of Fig. 16 were 1.601, 5.567, and 13.926 s for the linear, nonlinear, and constraint interpolation methods, respectively.

IV. Parameterized Surface Motion Control

Interactive animation from temporally aligned mesh sequences requires the combination of multiple captured sequences to allow continuous real-time control of movement with intuitive high-level parameters such as speed and direction for walking or height and distance for jumping. Methods for parameterization of MoCap have previously been introduced [18]–[20] which allow continuous high-level movement control by linear interpolation of joint angles. Blending of meshes based on linear interpolation of vertex positions is computationally efficient but may result in unrealistic deformation or mesh collapse if there are significant differences in shape. Nonlinear blending of meshes produces superior deformation [13], [14], [37], [39] but commonly requires least-squares solution of a system of equations which is prohibitive for real-time interaction.

Recent work [8]–[10] has introduced methods for high-level parameterization of SurfCap sequences. In this paper, we combine parametric control of SurfCap sequences with space–time editing and skeletal MoCap-driven animation to provide a framework for flexible animation. Here, we summarize the parameterization approach for completeness, full details can be found in [8]–[10]. The nonlinear mesh sequence blending is given by

$$M_p(t) = f(\{M_1(t), ..., M_N(t)\}, \mathbf{p}) \quad (14)$$

where $M_1(t), \ldots, M_i(t)$ are the set of source meshes, $\mathbf{p}$ is the vector of high-level parameters, N is the number of sequences, and $f()$ is a nonlinear mesh blending function. This function produces a mesh sequence $M_p(t)$ with the required parameter values $\mathbf{p}$ as described in [8].

In this paper, we use a hybrid solution for real-time mesh blending presented in [9] and [10] which combines the realistic deformation of nonlinear blending with efficient online computation. This approach allows us to interpolate any pair of meshes in real time, enabling synthesis of novel parametric mesh sequences while maintaining the realism of the captured data. Furthermore, similarly to [8], [9], and [10], our framework uses a mapping function to provide high-level parametric control of the motion. This is required due to the lack of intuitive parameterization of the motion using blend weights. Hence, for example, a walk and a run motions can be combined in real time to create a parametric animation in which the user controls the speed of a character. Notice that this mesh blending approach is not limited to pairs of meshes; it can also be applied to any number of meshes, for example, a walk, a run, and a turn can be combined to create a character in which we do not only control the speed but also the direction. Fig. 7 shows control of speed in a run achieved by parameterizing a walk and a run motions. Fig. 8 shows results of the parametric control for two examples: 1) jumping with control of distance; and 2) jumping with control of height.

V. Hybrid Skeletal and Surface Motion Control

Space–time editing and parametric control techniques presented in Sections III and IV allow modification of SurfCap sequences but are constrained to producing similar motions. The objective of the hybrid skeletal-surface motion control, introduced in this section, is to combine SurfCap with skeleton-driven surface animation in order to extend the range of

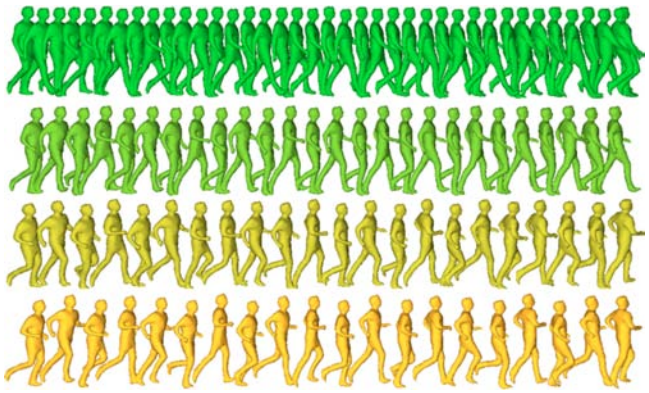

Fig. 7. Example of parameterized walk-run motion with user control of speed. First and fourth row: input walk and jog motions. Second and third rows: results of parametric control for two intermediate speeds.

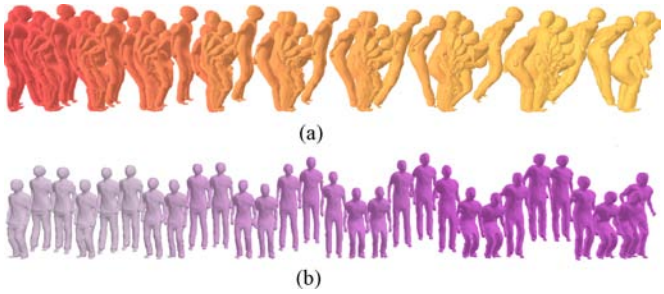

Fig. 8. Examples of parameterized motions between two motion sequences with continuous parameter variation. Results are shown every five frames. (a) Length of jump parameterized from short (red) to long (yellow). (b) Height of jump parameterized from low (gray) to high (purple).

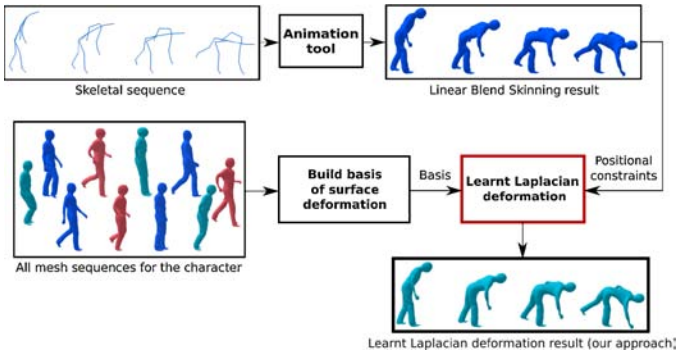

Fig. 9. Overview of the skeleton-driven animation with learned surface deformation approach.

motion beyond the SurfCap performance, while maintaining the detailed nonrigid surface dynamics. Integration with skeletal MoCap is motivated by two factors: first, the public availability of large motion capture archives covering a wide range of motion; and second, compatibility with existing skeleton animation pipelines and tools for skeletal control. Conventional skeletal animation of mesh surfaces is performed using linear blend skinning which can result in unnatural deformation such as mesh collapse around joints. Direct application of linear approaches to skeleton-driven animation of captured meshes does not produce natural surface deformation of skin or clothing resulting in a loss of visual fidelity compared to SurfCap sequences. Coherent integration of SurfCap and skeleton-driven animation to produce natural surface motion requires the transfer of captured surface deformation to skeleton-driven animation.

To achieve natural surface deformation that matches the SurfCap performance, we combine the learned surface deformation model introduced in Section III-B with skeleton-driven animation. This allows novel mesh sequences to be produced based on existing skeletal MoCap data and with surface deformation that is compatible with the SurfCap sequences. These sequences are then employed to extend the range of motion by transition from the SurfCap to MoCap sequences and vice versa. Hybrid surface-skeletal motion animation control comprises the following steps.

1) *Skeleton-driven animation with learned surface deformation:* The skeletal MoCap sequence is used to drive a learned model of surface deformation using the Laplacian framework introduced in Section III-B. This results in a skeleton-driven surface sequence with natural deformation compatible with the SurfCap data.
2) *Parameterization of SurfCap sequences:* Novel mesh sequences are created by interactively controlling high-level parameters of the character, such as speed in a run or height in a jump. These parameters can be adjusted by parameterizing a selection of the SurfCap sequences present in the database, as introduced in Section IV [8].
3) *Transitions between skeleton-driven animation and parameterized SurfCap:* Transition points between the parameterized SurfCap sequence and the skeleton-driven animation are automatically identified based on frames with similar surface shape and motion.
4) *Space–time editing for seamless transitions:* If the poses of the skeleton-driven animation and the SurfCap sequence in the transition points found in the previous step are not sufficiently similar, the space–time editing approach introduced in Section III can be employed to create seamless transitions. By making small modifications to the poses, the visual quality of the transition is improved, allowing the two sequences to be seamlessly linked.

A. Skeleton-Driven Animation With Learned Surface Deformation

Linear subspace deformation is widely used as it allows real-time skeleton-driven mesh animation but may result in mesh collapse at joints or the candy wrapper effect due to twisting [42]. In Fig. 10(a) (second row), mesh collapse can be observed between the leg and torso due to the bending of the character resulting in unrealistic surface deformation. An example of this linear approach is the Pinocchio library [43], which evaluates the blend weights of the mesh surface based on a reference skeleton. Rigging the captured 3-D surface produces a model which can be animated from the reference mesh according to a skeletal MoCap sequence, as illustrated in Fig. 10.

Plausible nonrigid surface deformation for the skeleton-driven animation is achieved using the learned Laplacian deformation as follows (see Fig. 9).

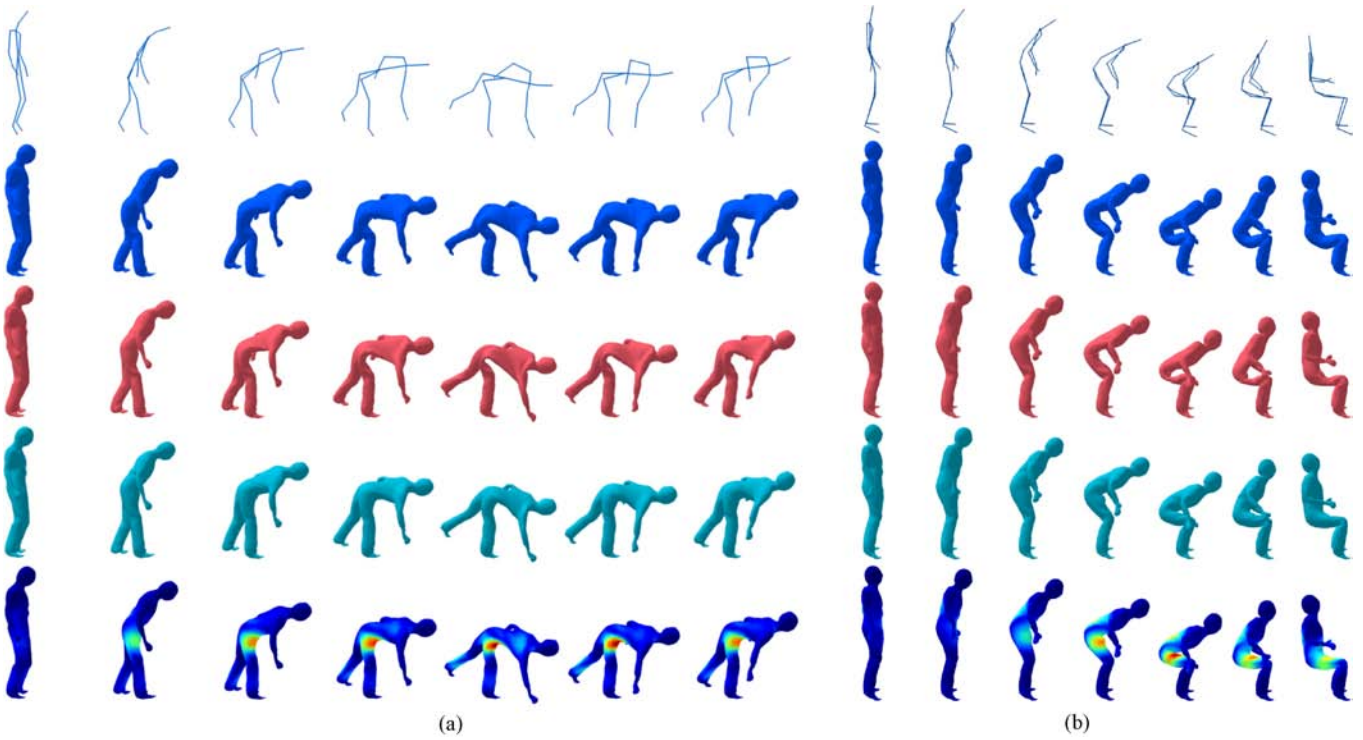

Fig. 10. Comparison of skeleton-driven animation techniques for pick up and sit down motions. First row: skeletal motion. Second row: linear subspace deformation. Third row: Laplacian mesh deformation. Fourth row: Laplacian mesh deformation with a learned deformation basis (proposed approach). Fifth row: difference between learned Laplacian and linear subspace deformation (heat map from low blue to high red). (a) Pick up motion. (b) Sit down motion.

1) *Reference mesh/skeleton and rigging weights:* A single reference mesh, preferably in a T-pose, is chosen from the SurfCap data. Subsequently, a reference skeleton is fitted into the reference mesh and the vertex weights are evaluated using the automatic skinning Pinocchio framework [43].
2) *Linear skeleton-driven animation:* An initial linear mesh animation sequence is generated by animating the reference mesh according to the desired skeletal MoCap sequence and the vertex weights found by Pinocchio.
3) *Constraint selection:* A sparse subset of the linearly animated mesh vertices are interactively selected for regions of the mesh which do not exhibit mesh collapse or other artifacts. These vertices should constrain the pose of the character.
4) *Learned deformation:* A surface deformation space is learned from the SurfCap performance sequences for a specific actor and clothing. All sequences for the actor are used to learn a basis model as described in Section III-B, retaining 95% of the variance in the data.
5) *Constrained learned Laplacian deformation:* The reference mesh is deformed using the learned Laplacian deformation according to (7). This nonlinear deformation is performed for each frame of the skeletal sequence, subject to the positional constraints given by the vertex selection on the linearly animated meshes and the learned surface deformation space.

Fig. 10 compares the performance of our method with other skeleton-driven animation techniques. In Fig. 10(a), the learned deformation approach (fourth row) produces a plausible nonrigid deformation in the pelvis region as the character bends over, correcting the mesh collapse that occurs with both linear subspace deformation (second row) and Laplacian surface deformation (third row). Due to the SurfCap sequences

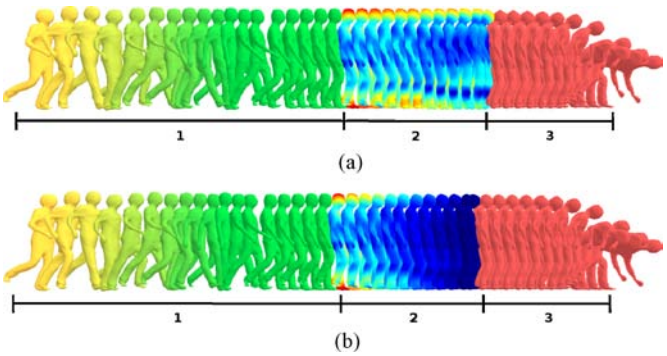

Fig. 11. Transition between parametric walk-run surface motion and skeleton driven animation of pick up motion without (a) and with (b) space-time editing. The sequence is in three sections. Section 1 (meshes sampled every fifth frame) is a parameterized motion where the user controlled the speed of the character (yellow, run; green, walk). Section 2 (meshes sampled every frame) depicts the transition between the parameterized section 1 and the skeleton driven motion of section 3, colored as a heatmap to highlight the difference in shape between each of the meshes and the target pose (first mesh of section 3). Finally, section 3 (meshes sampled every fifth frame) shows a skeleton driven motion using the learnt Laplacian deformation that was generated to provide a picking motion to our database. Notice how in (a), in which the linked poses were not edited, the heatmap shows a large error in section 2 due to the jump between source and target motions. A smooth transition is achieved in (b) with the application of space-time editing, which significantly reduces shape error between linked poses.

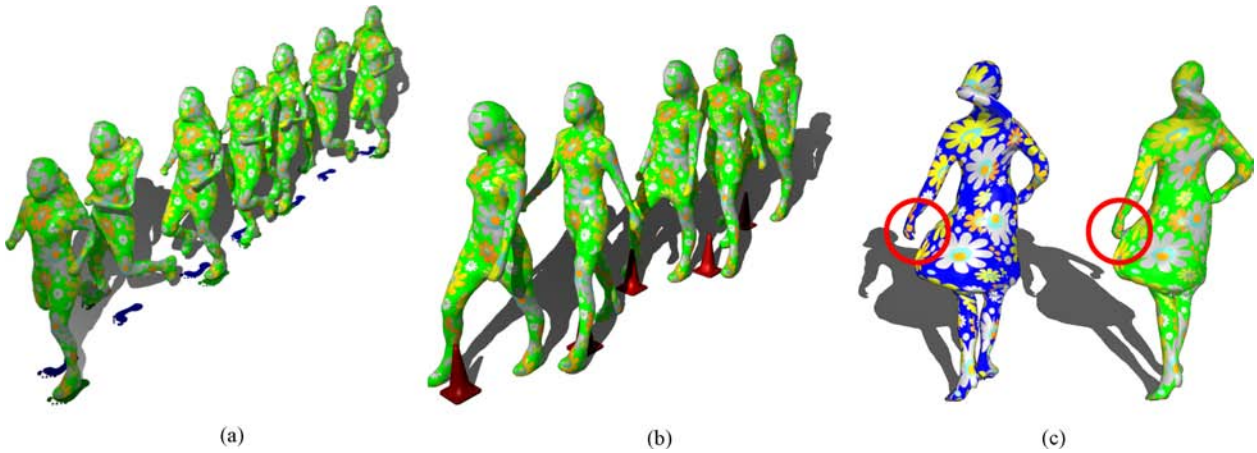

Fig. 12. Space–time editing of a walk sequence for changing feet positions (a), collision avoidance (b), and repairing reconstruction errors (c) (the hand has been moved to grasp the skirt correctly). Original (blue); edited (green).

containing natural shapes for the lower torso which are represented in the learned Laplacian deformation basis, our method achieves a natural-looking deformation even though the motion is different from all of the observed surface motions. Similar results can be observed in Fig. 10(b) for a skeleton-driven animation of a sitting down motion.

The difference between the linear subspace and learned Laplacian nonlinear deformation is presented in Fig. 10 (fifth row), clearly identifying the corrected region. The traditional Laplacian approach produces a smoother deformation but results in a greater collapse due to the preservation of triangle shape for the reference mesh. In contrast, learned surface deformation forms a nonlinear subspace (linear in the Laplacian differential coordinates) which constrains the skeleton-driven mesh animation to plausible surface deformations overcoming the limitations of linear deformation and produces a mesh animation with similar deformations to the SurfCap data for a specific actor.

B. Transitions Between Skeleton-Driven Animation and Parameterized SurfCap

Two stages are required for transition between parameterized SurfCap motion and skeleton-driven surface animation (or vice versa): 1) identification of transition points with similar shape and motion; and 2) space–time editing to produce a seamless transition if poses are not sufficiently similar. Possible transition points between this parametric space and the rigged skeletal sequences are found by measuring the shape and motion similarity between all points in the space. A regular grid sampling of the continuous parametric motion space is used with a step of $\Delta\alpha = 0.1$. The similarity is evaluated between each sample point in the parametric motion space and each frame of the skeleton-driven motion sequence.

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

$$s(M_i, M_j) = 1 - \frac{d(M_i, M_j)}{\max(d(M_i, M_j))}.$$

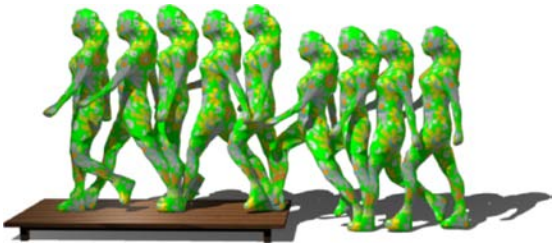

Fig. 13. Space–time editing of a walk sequence for stepping onto a platform.

The pair of meshes that have the highest similarity $\arg\max_{ij}\{s(M_i, M_j)\}$ are annotated as the optimal transition point between the source and target sequences. For real-time interactive animation, when the user chooses to transition from the parametric motion space to the skeleton-driven animation, the character will move to the source frame of the transition (in the parametric space) and subsequently transition to the target frame. To achieve smooth transitions, linear blending of meshes is performed over a window of n frames around the transition meshes. Experimental results show that $n = 5$ achieves natural results. Using higher values would cause the appearance of artifacts such as foot sliding.

C. Space–Time Editing for Seamless Transitions

If the shape and motion at the optimal transition point are not sufficiently similar, then space–time editing can be employed to modify the skeleton-driven animation to match the parametric motion. This is required to avoid jumps in character pose or motion where the skeletal motion is significantly different from the parametric surface motion.

The space–time mesh editing technique, presented in Section III, is applied to modify the skeleton-driven animation.

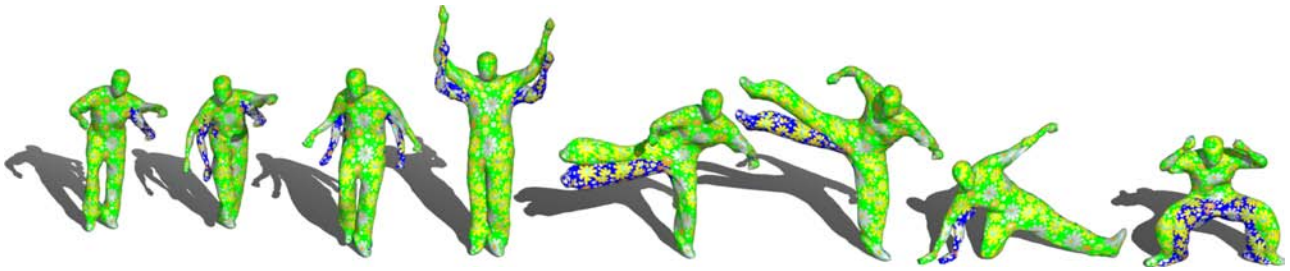

Fig. 14. Space–time editing of arm and leg movement for a street dancer sequence: original (blue); edited (green).

The transition target frame from the skeleton-driven animation sequence is interactively edited to match the source frame from the parametric motion space. Laplacian mesh editing with the learned deformation space is constrained to match vertex positions between the source and target sequences using (7). Space–time propagation is then used to propagate changes across the skeleton-driven animation sequence.

Space–time editing results in a new edited skeleton-driven animation sequence which allows a seamless real-time transition between the parametric motion space and the target skeletal motion. This enables extension of the range of character motion exploiting existing databases of skeletal MoCap sequences, while maintaining the natural surface deformation of SurfCap for a specific actor. Fig. 11 illustrates the transition between a parameterized surface and a skeleton-driven animation with and without space–time editing. The transition frames are depicted as a heat map showing the difference to the target frame (first frame of the target motion in red). The heat map shows that without space–time editing, Fig. 11(a), there is a large difference between the last frame of the transition and the first frame of the target motion, resulting in an unnatural jump in the motion. After space–time editing, the last frame of the transition has only a small difference to the first frame of the target motion. This allows a seamless transition between the parameterized surface motion and the skeleton-driven animation. Further results of transitions are presented in the accompanying video.

VI. Results and Evaluation

Evaluation is performed using a public database of SurfCap sequences [2] which includes 3-D mesh sequence reconstructions for multiple actors wearing a variety of clothing and performing a range of motions. Temporal surface alignment is performed using the nonsequential nonrigid surface tracking approach [31]. MoCap sequences used in the hybrid skeletal-surface animations are from the Carnegie Mellon University (CMU) archive [15].

A. Space–Time Editing of Surface Motion

Results of the space–time editing technique presented in Section III, which incorporate a learned deformation space, are performed on both synthetic and captured mesh sequences. Meshes are textured with a flower pattern to demonstrate how the smoothness of the temporal alignment is maintained throughout the editing process. A variety of editing operations illustrate the flexibility of the proposed approach. Space–time editing of a walk sequence to modify feet positions, avoid obstacles, and step up onto a platform are shown in Figs. 12 and 13. This illustrates a common application of space–time editing of captured sequences to modify contact positions according to scene constraints. In Fig. 12(c), the space–time editing approach has been used to repair reconstruction errors. The original sequence (see accompanying video) shows a twirl where there is a loss of contact between the hand and the skirt. In the edited sequence, the hand has been moved to grasp the skirt correctly.

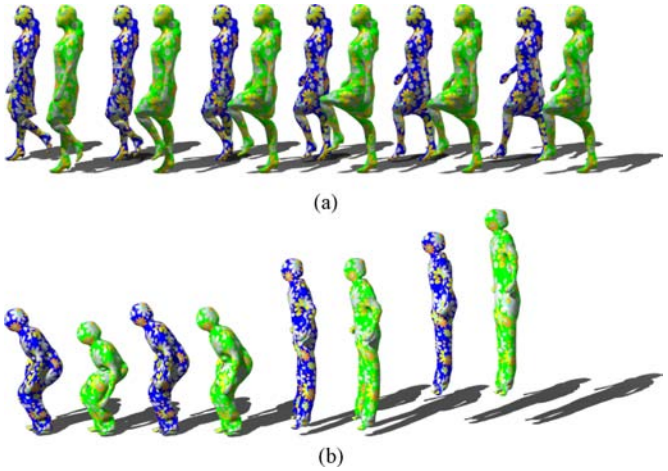

Fig. 15. Stylized sequences: original (blue); edited (green). (a) Walk with raised knees (b) Jump with squash and stretch effects.

Fig. 14 shows a more complex space–time edit to modify the arm and leg movements for the street dancer, while preserving both the anatomical structure and surface dynamics.

Space–time editing also allows artistic stylization of the motion to create common animation effects such as movement emphasis, exaggeration, and cartoon effects of squash stretch as well as retiming of the sequence for anticipation. Fig. 15 presents examples of motion stylization to exaggerate the walking of a character with a loose dress and to produce a cartoon style squash-stretch effect for a jump.

Finally, Fig. 16 shows the editing of a synthetic horse galloping sequence where the torso of the horse has been lifted. This example illustrates the effect of applying large changes to a mesh sequence. Constraining the deformation to a learned deformation space preserves the mesh structure ensuring a natural motion sequence.

Results of space–time editing demonstrate that the approach allows flexible interactive editing of captured sequences to satisfy user-specified constraints, while preserving the natural

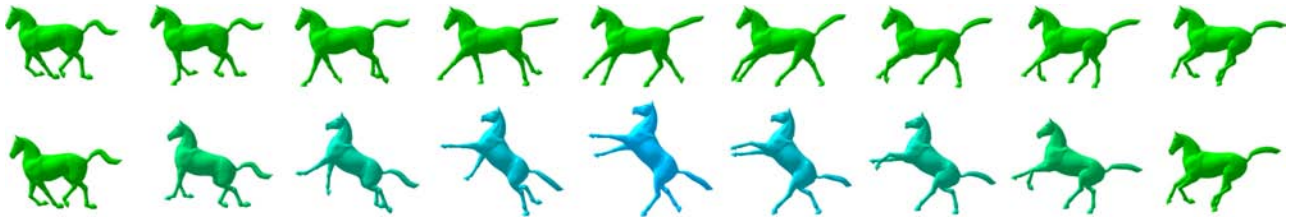

Fig. 16. Space–time editing of a synthetic horse galloping sequence. Top row: original. Bottom row: edited. (Dataset courtesy of [39].)

TABLE I
COMPUTATION TIMES OF A SELECTION OF KEY-FRAME EDITS FOR SEQUENCES OF FIGS. 12(B), 14, 12(C), AND 16

Data type	Sequence	No. of vertices	Edit	No. of constraints	Time (s)
Real data	*C*ones [Fig. 12(b)]	2886	1	236	0.636
			2	242	0.635
			3	268	0.644
			4	247	0.631
			5	255	0.643
	*D*ancer (Fig. 14)	5580	1	1585	1.187
			2	1345	1.446
			3	1270	1.430
			4	1494	1.171
			5	1508	0.880
			6	1497	1.042
			7	1560	1.082
			8	1147	1.057
			9	704	0.987
			10	1432	1.051
			11	1258	1.031
Real data for stylization	*S*kirt [Fig. 15(a)]	2854	1	691	0.829
			2	722	0.396
			3	613	0.673
			4	595	0.820
			5	550	0.658
			6	588	0.536
Synthetic	*H*orse (Fig. 16)	8431	1	6753	1.601

Each edit number corresponds to a single key-frame edit and has a deformation time associated with it.

spatiotemporal dynamics of the captured motion. The linear space–time interpolation approach has been used to generate the resulting sequences of Figs. 12–15. Since the edits performed over these examples are small deformations, this has not introduced visual artifacts. For the *Horse* sequence of Fig. 16, where the key frame undergoes a large deformation, the nonlinear interpolation was preferred to generate the final sequence. As shown in Fig. 6, in this case the linear method introduces significant errors if applied.

Computation times for a selection of key-frame edits can be found in Table I. The learned Laplacian deformation solver takes 0.5–1 s for meshes of 3000–6000 vertices, allowing interactive editing with rapid feedback. These timings are for a CPU implementation of the approach; real-time performance could potentially be achieved with transfer of the Laplacian solver to a GPU. Typical values of T_k are in the range 4–8 frames.

B. Hybrid Skeleton and Surface Animation Control

Fig. 17 presents two examples of hybrid motion control transitioning from a parameterized surface motion space to a skeleton-driven animation. The corresponding animation sequences can be found in the accompanying video. A drinking and a boxing skeletal sequences from the CMU database were animated employing the technique introduced in Section V-A, presenting surface deformation which is coherent with the SurfCap data. As demonstrated in the results, the proposed hybrid approach based on learned Laplacian deformation enables the creation of plausible animations from the combination of SurfCap and MoCap sequences with seamless transitions. This extension of SurfCap using existing archives of skeletal MoCap greatly increases the range of character motion without compromising the natural nonrigid deformation of captured surface motion.

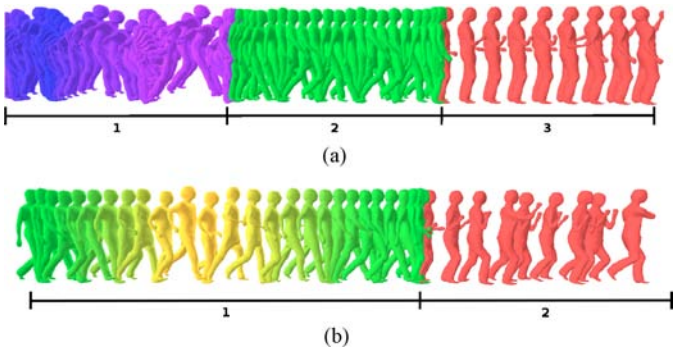

Fig. 17. Hybrid sequences transitioning from parameterized surface motion to skeleton-driven animation (every fifth frame shown). (a) Transition from jumping parameterized motion (purple, section 1) to walk motion (green, section 2) to skeleton-driven drinking motion (red, section 3) (b) Transition from walk-run parameterized motion (yellow-green with change in parameter, section 1) to skeleton-driven boxing motion (section 2).

VII. DISCUSSION AND LIMITATIONS

The flexibility of the presented hybrid animation technique is increased when the SurfCap database contains a broad collection of motions. In order to achieve parameterized surface motion control, the database should contain parameterizable motions (walk/run at different speeds, jumps at different heights, etc.) and Laplacian deformation benefits from the incorporation of a learned deformation space under the assumption that the database contains a sufficient range of motion for a given character. If the user constrains the mesh such that the target deformation lies completely outside the learned space, the underlying structure of the character might not be preserved. An example of this effect is demonstrated in Fig. 18. Applying the skeletal animation technique presented in Section V-A, although the torso deformation is natural

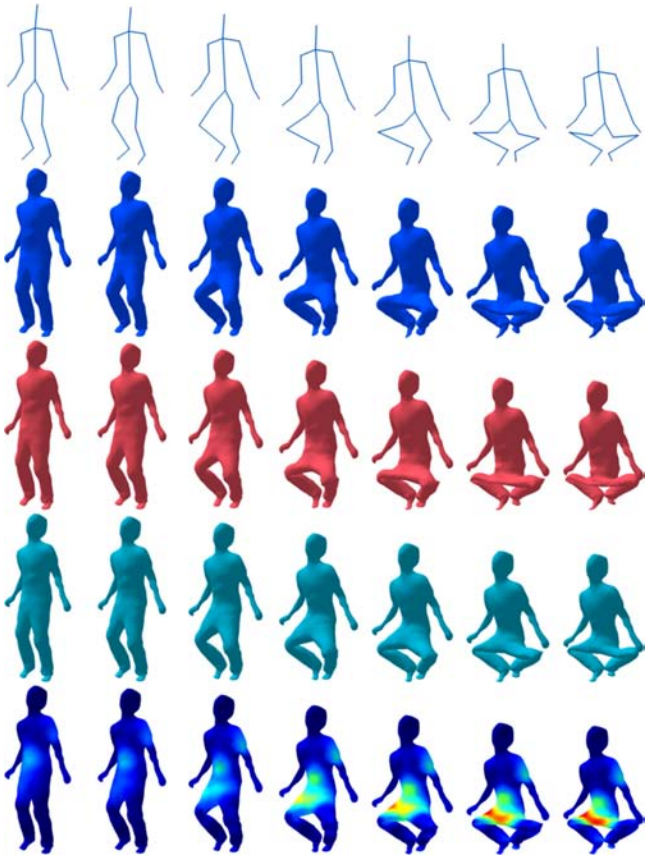

Fig. 18. Skeleton-driven animation for a squat motion showing the limitations of our approach using a learned deformation model. First row: skeletal motion. Second row: linear subspace deformation. Third row: Laplacian mesh deformation. Fourth row: Laplacian mesh deformation with a learned deformation basis (proposed approach). Fifth row: difference between learned Laplacian and linear subspace deformation (heat map from low blue to high red).

throughout the squat sequence, the leg shape is unnatural under extreme bending as no similar pose is present in the SufCap sequences used to learn the deformation basis. Note in this example linear and traditional Laplacian deformation suffer even greater visual artifacts such as unnatural shape of the torso and the pelvis area. To overcome the limitation of the current approach and increase the range of movement for a limited database of SurfCap examples, an independent deformation basis could be learned for each body part. In principle, using a local shape basis rather than the global basis used in this paper would allow greater range of manipulation of the whole body shape.

VIII. Conclusion

This paper introduced a framework for interactive animation control of SurfCap sequences. Three novel approaches were introduced to facilitate flexible modification and extension of the captured motion.

Space–time editing using a learned surface deformation basis allowed interactive low-level control and editing of captured surface sequences of actor performance. This low-level control was essential for artistic manipulation and imposing constraints such as foot-floor or hand-object contact. To extend the range of motion beyond that of captured surface sequences, a hybrid control approach was introduced which allowed the combination of MoCap-driven animation with SurfCap. A critical contribution to achieving this hybrid solution was the introduction of skeleton-driven mesh animation using a learned Laplacian deformation basis. This enabled meshes to be animated from skeletal motion with plausible natural nonrigid surface deformation of skin and clothing. Space–time editing was employed to achieve seamless transitions between parameterized surface motion and skeleton-driven sequences. Hybrid skeleton-surface animation allowed integration of existing MoCap archives with SurfCap, greatly extending the range of motion while maintaining the natural nonrigid surface deformation characteristics of skin and clothing. The framework introduced in this paper allowed flexible interactive control of SurfCap, providing the foundation for a new approach to character animation based on SurfCap of actor performance. SurfCap has the potential to reproduce the complex dynamics of real nonrigid surface deformation for skin, clothing, and hair which requires computationally expensive simulation in current skeleton-driven character animation pipelines. Future challenges include the integration of surface appearance and control techniques for modeling real-time character interaction and simulation of motion dynamics.

References

[1] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *Proc. ACM SIGGRAPH*, 2004, pp. 600–608.

[2] J. Starck and A. Hilton, "Surface capture for performance based animation," *IEEE Comput. Graph. Appl.*, vol. 27, no. 3, pp. 21–31, May–Jun. 2007.

[3] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 98:1–98:10, 2008.

[4] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," in *Proc. ACM SIGGRAPH*, 2008, pp. 97:1–97:9.

[5] J. Starck, G. Miller, and A. Hilton, "Video-based character animation," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animat.*, 2005, pp. 49–58.

[6] P. Huang, A. Hilton, and J. Starck, "Human motion synthesis from 3D video," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1478–1485.

[7] F. Xu, Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.-P. Seidel, J. Kautz, and C. Theobalt, "Video-based characters: Creating new human performances from a multi-view video database," in *Proc. ACM SIGGRAPH*, 2011, pp. 32:1–32:10.

[8] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton, "Parametric control of captured mesh sequences for real-time animation," in *Proc. 4th Int. Conf. Motion Games*, 2011, pp. 242–253.

[9] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton, "4D parametric motion graphs for interactive animation," in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graph. Games*, 2012, pp. 103–110.

[10] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton, "Interactive animation of 4D performance capture," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 5, pp. 762–773, May 2013.

[11] M. Gleicher, "Motion editing with spacetime constraints," in *Proc. Symp. Interactive 3D Graph.*, 1997, pp. 139–148.

[12] J. Lee and S. Y. Shin, "A hierarchical approach to interactive motion editing for human-like figures," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Tech.*, 1999, pp. 39–48.

[13] W. Xu, K. Zhou, Y. Yu, Q. Peng, and B. Guo, "Gradient domain editing of deforming mesh sequences," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 84:1–84:10, 2007.

[14] S. Kircher and M. Garland, "Free-form motion processing," *ACM Trans. Graph.*, vol. 27, no. 2, pp. 1–13, 2008.

[15] CMU graphics lab motion capture database. (2003) [Online]. Available: http://mocap.cs.cmu.edu/
[16] A. Brundelin and L. Williams, "Motion signal processing," in *Proc. ACM SIGGRAPH*, 1995, pp. 97–104.
[17] D. Wiley and J. Hahn, "Interpolation synthesis for articulated figure motion," in *Proc. IEEE Virtual Reality Int. Symp.*, Mar. 1997, pp. 156–160.
[18] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: Multidimensional motion interpolation," *IEEE Comput. Graph. Appl.*, vol. 18, no. 5, pp. 32–40, Sep.–Oct. 1998.
[19] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large date sets," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 559–568, 2004.
[20] T. Mukai and S. Kuriyama, "Geostatistical motion interpolation," in *Proc. ACM SIGGRAPH*, 2005, pp. 1062–1070.
[21] M. Brand and A. Hertzmann, "Style machines," in *Proc. 27th Annu. Conf. Comput. Graph. Interactive Tech.*, 2000, pp. 183–192.
[22] E. Hsu, K. Pulli, and J. Popović, "Style translation for human motion," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1082–1089, 2005.
[23] A. Shapiro, Y. Cao, and P. Faloutsos, "Style components," in *Proc. Graph. Interface*, 2006, pp. 33–39.
[24] J. Min, H. Liu, and J. Chai, "Synthesis and editing of personalized stylistic human motion," in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graph. Games*, 2010, pp. 39–46.
[25] T. Kanade and P. Rander, "Virtualized reality: Constructing virtual worlds from real scenes," *IEEE MultiMedia*, vol. 4, no. 1, pp. 34–47, Jan.–Mar. 1997.
[26] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," in *Proc. ACM SIGGRAPH*, 2003, pp. 565–577.
[27] C. Cagniart, E. Boyer, and S. Ilic, "Free-form mesh tracking: A patch-based approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1339–1346.
[28] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 475–480, Mar. 2005.
[29] M. Wand, B. Adams, M. Ovsianikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling, "Efficient reconstruction of non-rigid shape and motion from real-time 3D scanner data," *ACM Trans. Graph.*, vol. 28, no. 2, pp. 15:1–15:15, Apr. 2009.
[30] P. Huang, C. Budd, and A. Hilton, "Global temporal registration of multiple non-rigid surface sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2011, pp. 3473–3480.
[31] C. Budd, P. Huang, M. Klaudiny, and A. Hilton, "Global non-rigid alignment of surface sequences," *Int. J. Comput. Vis.*, vol. 102, nos. 1–3, pp. 256–270, 2012.
[32] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: Driving visual speech with audio," in *Proc. ACM SIGGRAPH*, 1997, pp. 353–360.
[33] A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *Proc. 27th Annu. Conf. Comput. Graph. Interactive Tech.*, 2000, pp. 489–498.
[34] M. Flagg, A. Nakazawa, Q. Zhang, S.-B. Kang, Y. Ryu, I. Essa, and J. Rehg, "Human video textures," in *Proc. ACM Symp. Interactive 3D Graph.*, 2009, pp. 199–206.
[35] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 1, pp. 213–230, Jan. 2008.
[36] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, "Mesh-based inverse kinematics," in *Proc. ACM SIGGRAPH*, 2005, pp. 488–495.
[37] O. Sorkine, "Differential representations for mesh processing," *Comput. Graph. Forum*, vol. 25, no. 4, pp. 789–807, 2006.
[38] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear rotation-invariant coordinates for meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 479–487, Jul. 2005.
[39] R. Sumner and J. Popovic, "Deformation transfer for triangle meshes," in *Proc. ACM SIGGRAPH*, 2004, pp. 399–405.
[40] M. Botsch, R. W. Sumner, M. Pauly, and M. Gross, "Deformation transfer for detail-preserving surface editing," in *Proc. Vis., Model., Vis.*, 2006, pp. 357–364.
[41] C. Stoll, E. de Aguiar, C. Theobalt, and H.-P. Seidel, "A volumetric approach to interactive shape editing," Max-Planck-Institut fur Informatik, Saarbrucken, Germany, Tech. Rep. MPI-I-2007-4-004, Jun. 2007.
[42] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation," in *Proc. ACM SIGGRAPH*, 2000, pp. 165–172.
[43] C. Stoll, E. de Aguiar, C. Theobalt, and H.-P. Seidel, "A volumetric approach to interactive shape editing," Res. Rep. MPI-I-2007-4-004, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Jun. 2007.

Margara Tejera (SM'08) received the B.Sc. and M.Sc. degrees in telecommunication engineering from the University of Seville, Seville, Spain. She is currently pursuing the Ph.D. degree from the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, U.K.

During 2008–2009, she was a Research Scholar with the Human Sensing Lab, Carnegie Mellon University, Pittsburgh, PA, USA. Her current research includes generating algorithms and tools that allow the synthesis of novel 3-D video sequences.

Dan Casas (SM'13) received the M.Eng. degree from the Universitat Autònoma de Barcelona, Barcelona, Spain, in 2009. He is currently pursuing the Ph.D. degree from the Centre for Vision, Speech and Signal Processing, University of Surrey, Guilford, U.K.

During 2008–2009, he was a Research Scholar with Human Sensing Lab, Carnegie Mellon University, Pittsburgh, PA, USA. His current research interests include character animation, motion synthesis, and free-viewpoint video.

Adrian Hilton (M'98) received the B.Sc. (Hons.) and D.Phil. degrees from the University of Surrey, Guildford, U.K., in 1988 and 1992, respectively.

He is currently a Professor of computer vision and the Director of the Centre for Vision, Speech and Signal Processing, University of Surrey, Guilford, U.K.. His contributions include technologies for the first hand-held 3-D scanner, modeling of people from images, and 3-D video for games, broadcast, and film. He currently leads research investigating the use of computer vision for applications in entertainment content production, visual interaction, and clinical analysis. His current research interests include robust computer vision to model and understand real-world scenes.

Dr. Hilton is a Chartered Engineer.