DOI: 10.1111/j.1467-8659.2012.03032.x
EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape

Thomas J. Cashman   Kai Hormann

University of Lugano

Abstract

It is increasingly popular to represent non-rigid motion using a deforming mesh sequence: a discrete sequence of frames, each of which is given as a mesh with a common graph structure. Such sequences have the flexibility to represent a wide range of mesh deformations used in practice, but they are also highly redundant, expensive to store, and difficult to edit in a time-coherent manner. We address these limitations with a continuous representation that extracts redundancy in three separate phases, leading to separate editable signals in time, pose and shape. The representation can be applied to any deforming mesh sequence, in contrast to previous domain-specific approaches. By modifying the three signal components, we demonstrate time-coherent editing operations such as local repetition of part of a sequence, frame rate conversion and deformation transfer. We also show that our representation makes it possible to design new deforming sequences simply by sketching a curve in a 2D pose space.

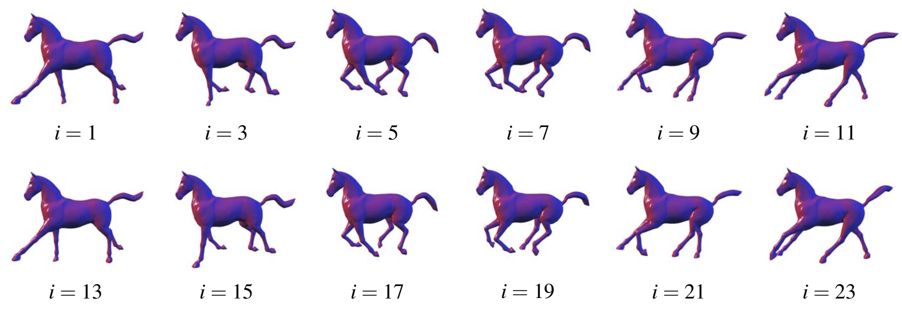
Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

There are a large and growing number of applications for dynamic geometry data, that is, 3D objects with time-varying shape. Where non-rigid motion is important, it is popular to represent dynamic geometry using a 'deforming mesh sequence' [XZY*07]: a surface mesh with changing vertex positions but fixed connectivity. Deforming mesh sequences provide a very expressive description for an object under complex motion; they appear, for example, in cloth simulation [GHF*07], in human reconstruction from captured data [dAST*08], and in facial animation from realtime depth scans [WBLP11]. In each of these cases, the deforming sequence is *discretely sampled* in time, and it is either expensive or impossible to gain a denser time sampling. The resulting discrete sequence is also highly *redundant*, as it represents the same object under deformation and so different frames capture broadly similar shapes. As a result of these two properties, the sequence is very *inflexible* and difficult to edit [KG06].

Animation systems also work with deforming mesh sequences, but very rarely expose the discretized data directly. Instead they offer a *continuous*, *compact* and *editable* representation, which is sampled to gain discrete frames only for rendering. Skeletal subspace deformation [LCF00], for example, generates articulated deformation of a dense mesh by associating vertices with a low-dimensional skeletal model. However, this type of representation is applicable only when the deforming sequence portrays articulated skeletal motion with little variation in high-frequency detail. In this paper, we extend the benefits of a continuous and editable representation to a much wider class of deforming mesh sequences. The proposed representation decomposes dynamic geometry data into signals at three different levels, allowing separate analyses and editing operations in time, pose and shape.

Our main target is a representation for short sequences: typically no longer than about 30 seconds. This limits complexity so that we can provide a useful overview of an entire sequence at once (see Figure 1 for an example). We discuss this design decision further in Section 6. We emphasize that *compressing* a deforming mesh sequence is not our primary goal; although compression is one effect of using our representation, it would also be possible to gain further compression rates as a post-process. The most important property is not the size of storage, but whether the representation is sufficiently compact to make typical editing operations possible, such as those shown in Section 4. We compress the sequence to the extent that these two goals are compatible, but avoid a representation that is so compact that it is no longer editable.



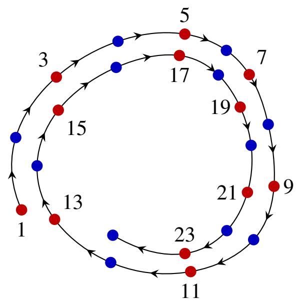


Figure 1: *An example of our representation for a galloping horse, generated by reconstructing a sequence with $n = 24$ frames. The shapes shown left are the result $\psi(i)$ of evaluating our representation at the given values for i. The time signal (right) shows positions $\{f(i)\}_1^n$ marked with red and blue points; red points indicate frames where the corresponding mesh $\psi(i)$ is shown left. As this sequence consists of two similar 12-frame cycles, the time signal loops twice. Positions $f(i)$ and $f(i+12)$ are close in pose space, and therefore generate similar shapes. This example uses the constants $k = 16$, $l = 12$ and $m = 7$ (see Section 3.4).*

1.1. Related work and contributions

To take advantage of a continuous, compact and editable representation, the first step is often to *convert* an existing deforming mesh sequence into the target representation. Most existing work addresses the conversion into a skeleton-based representation. Following the seminal work of James and Twigg [JT05], there are methods for finding and fitting a suitable skeleton to single meshes [BP07], mesh sequences [dATTS08], motion capture data [ASK*05], or even non-skeletal objects like cloth [KSO10]. Conversion into our representation does not require skeletal, physical or semantic annotations, and we show that it can be achieved by using just the information in discretely sampled meshes (Section 3). We do not limit ourselves to articulated motion, as in skeletal deformation [LCF00], and neither do we specialize our representation for a particular type of animation, like linear blendshapes for facial expression [ARL*09]. Kircher and Garland [KG06, KG08] and Xu et al. [XZY*07] also work without these constraints, providing an editable representation for arbitrary deforming mesh sequences, but they maintain the inherently discrete nature of the source data. While this allows them to modify the spatial characteristics of the data in a time-coherent way, it rules out, for example, the addition of new frames.

In our representation it is possible to add new frames and extend the animation by simply editing a curve in a two-dimensional pose space (Section 4.1). More generally, we can explore meaningful poses by interactively navigating this pose space and even create new animations from scratch (Section 4.4), in a similar way to the systems Kilian et al. [KMP07] and Yang et al. [YYPM11] propose for shape-space exploration. Lewis et al. [LCF00] pioneer this idea of exploring and editing animations by means of a simple user interface and, like us, use scattered data interpolation to connect the simple navigation space to the complicated nonlinear manifold of meaningful poses. Working from video rather than mesh sequences, Favreau et al. [FRDC04] also create animations by using radial basis functions to map from a reduced-dimension pose space: the same techniques that we use to construct shape and pose signals (Sections 3.1 and 3.2).

Another advantage of the continuous nature of our representation is the ability to modify the temporal characteristics of a deforming mesh sequence both globally and locally. This allows us to convert the frame rate as well as to slow down and speed up certain parts of the animation (Section 4.2).

Finally, our representation can also be used for deformation transfer [SP04], where a given animation of a certain character is adapted to another (Section 4.3). While the method and results turn out to be similar to semantic deformation transfer [BVGP09] and are not novel in this sense, this emphasizes the flexibility of our approach.

2. Proposed representation

This section explains the time, pose and shape signals that we use to encode a deforming mesh sequence. In Section 3 we address the question of how these signals should be set to reconstruct a sequence given as discrete mesh frames, and Section 4 covers editing and creating deforming mesh sequences by manipulating these signals.

We start by choosing a static representation for meshes with a given graph structure. Since a deforming mesh sequence does not modify the mesh connectivity, each frame is specified completely by the positions of all mesh vertices. We require a static representation that encodes each possible configuration of the mesh vertices as a single point in a *shape space* that we denote $\mathcal{S}$. Mesh representations which fit this description include deformation gradients [SP04], linear rotation-invariant coordinates [LSLCO05], and mesh storage based on edge lengths and dihedral angles [WDAH10, FB11]. Section 5 discusses the choice of shape space further. With $\mathcal{S}$ in place, we can write the n frames of the deforming mesh sequence as $\{\mathbf{z}_i\}_1^n = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\} \subset \mathcal{S}$.

Since every possible configuration of the 3D target mesh is contained in $\mathcal{S}$, the dimension d of $\mathcal{S}$ must be at least $3V$, where V is the number of vertices in the mesh. Indeed, the simplest possible shape space is simply $\mathbb{R}^{3V}$, where the mesh vertices are directly stacked into a single vector of length $3V$ (see Section 5). As discussed in Section 1, a deforming mesh sequence $\{\mathbf{z}_i\}_1^n \subset \mathcal{S}$ is discrete, redundant and inflexible. Our representation aims to find a *continuous trajectory* in $\mathcal{S}$ from which these frames might have been sampled. That is, we view a deforming mesh sequence as a sampled curve in $\mathcal{S}$. To represent this high-dimensional trajectory compactly, we introduce three spaces of lower dimension, namely a shape subspace $\mathbb{R}^m$, a pose space $\mathbb{R}^2$, and a time interval $\mathcal{I} \subset \mathbb{R}$. We connect these spaces together using functions f, g and h, which serve as our time, pose and shape signals, respectively:

$$\mathbb{R} \supset \mathcal{I} \xrightarrow{\ f\ } \mathbb{R}^2 \xrightarrow{\ g\ } \mathbb{R}^m \xrightarrow{\ h\ } \mathcal{S}. \tag{1}$$

Together, these functions define a continuous map $\psi : \mathcal{I} \to \mathcal{S}$, where $\psi = h \circ g \circ f$. Each signal is described in detail below.

2.1. Shape signal

For a sequence of n frames $\{\mathbf{z}_i\}_1^n$, where typically $n \ll 3V$, the dimension of $\mathcal{S}$ is far greater than we need to represent the data with a linear space. We exploit this *shape redundancy* by replacing $\mathcal{S}$ with a linear shape subspace $\mathbb{R}^m$ instead, where $m < n$, so that the frame with index i is represented by a vector $\mathbf{x}_i \in \mathbb{R}^m$. This vector gives the coordinates of the frame with respect to m basis vectors $\{\mathbf{h}_j\}_1^m \subset \mathcal{S}$ which form the columns of a matrix $\mathbf{H} \in \mathbb{R}^{d \times m}$.

To give meaningful basis vectors which allow a useful range of editing operations, we find it useful to offset the shape signal by $\mathbf{h}_0 \in \mathcal{S}$, leading to the definition

$$h : \mathbb{R}^m \to \mathcal{S}, \qquad h(\mathbf{x}) = \mathbf{h}_0 + \mathbf{H}\mathbf{x}$$

for the map from the reduced-dimension subspace to the original shape space. Our goal is that $h(\mathbb{R}^m)$ contains the sequence of frames $\{\mathbf{z}_i\}_1^n$ that form the deforming mesh sequence. This is always possible by setting m sufficiently high (see Section 3.4), but the success of the linear shape signal h in spanning useful mesh deformations is affected by the choice of shape space $\mathcal{S}$ (see Figure 7 and Section 5).

2.2. Pose signal

Where the deforming sequence includes the same mesh pose more than once, the shape coordinates $\{\mathbf{x}_i\}_1^n$ contain a duplicated point in $\mathbb{R}^m$. We extract this *pose redundancy*, specifying the mesh pose more concisely, by constructing a map to the shape subspace from a 2D pose space. We expect that the dimension m of the shape subspace is minimal with respect to linear projection, so this map from the pose space must be non-linear instead. That is, we construct a 2-manifold in the shape subspace $\mathbb{R}^m$ and want this manifold to pass through all the target mesh poses exactly once.

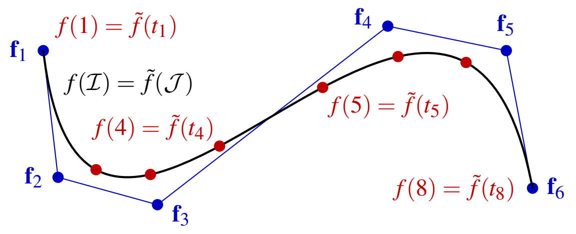


Figure 2: *An example time signal curve, for a simple case where $n = 8$ and $k = 6$. Spline control points $\{\mathbf{f}_j\}_1^k$ are shown in blue; points evaluated from the spline are shown in red.*

If we already know the pose-space positions $\{\mathbf{p}_i\}_1^n \subset \mathbb{R}^2$ for each frame (see Section 3.2.1), then finding this non-linear map is a scattered data approximation problem. This is the inspiration for us to use radial basis functions to construct the pose manifold. Given a radial basis function $\phi : \mathbb{R}_+ \to \mathbb{R}$ and a set of l *sites* $\{\mathbf{s}_j\}_1^l$ in the plane, the map from pose space is

$$g : \mathbb{R}^2 \to \mathbb{R}^m, \qquad g(\mathbf{p}) = \mathbf{g}_0 + \sum_{j=1}^{l} \phi(\|\mathbf{s}_j - \mathbf{p}\|)\,\mathbf{g}_j + \mathbf{G}\mathbf{p},$$

where the offset $\mathbf{g}_0$ and the matrix $\mathbf{G} = (\mathbf{g}_{l+1}, \mathbf{g}_{l+2}) \in \mathbb{R}^{m \times 2}$ ensure that the radial basis approximation has linear precision. This also guarantees the uniqueness of the representation when $\phi(r) = r^2 \log r$, to give a thin-plate spline [Wah90] (see Section 3.2). Each radial basis function is multiplied by a coefficient $\mathbf{g}_j \in \mathbb{R}^m$, which controls how the shape of the pose signal is related to distance from the site $\mathbf{s}_j$ in pose space.

2.3. Time signal

Each frame is now represented by a position $\mathbf{p}_i$ in pose space $\mathbb{R}^2$. The final step in our representation extracts *temporal redundancy* by providing a continuous approximation to these discrete positions. We might be able to exploit parts of the sequence that hold the same pose, for example, such that the points $\mathbf{p}_i$ cluster in the same position of pose space. Alternatively, for sequences that show smooth deformation of the target mesh, we expect the points $\mathbf{p}_i$ to move smoothly through the pose space, allowing us to represent the same data with fewer than n points in the plane (see Figures 1 and 2 for examples).

To approximate discrete positions with a continuous curve, a classical approach is spline fitting [dB01]. We use a cubic spline curve to map from an interval $\mathcal{J} \subset \mathbb{R}$ to pose space:

$$\tilde{f} : \mathcal{J} \to \mathbb{R}^2, \qquad \tilde{f}(u) = \sum_{j=1}^{k} B_j^{\mathbf{u}}(u)\,\mathbf{f}_j$$

where $B_j^{\mathbf{u}}$ is the jth B-spline basis function, and $\mathbf{f}_j$ is the corresponding control point (see Figure 2). These k cubic B-splines are defined on a knot vector $\mathbf{u}$ containing $k+2$ non-decreasing real-valued knots $\{u_j\}_0^{k+1}$ and $\mathcal{J} := [u_3, u_{k-1}]$. The positions $\mathbf{p}_i$ of the frames in pose space are approximated by evaluating $\tilde{f}$ at n values $t_1 \le t_2 \le \cdots \le t_n \in \mathcal{J}$.

For the sake of obtaining a good spline fit, it may be necessary to use unequally-spaced values t_i [PT97]. However, the n frames are understood to be equally spaced in time, and to make this link with the original time interval $\mathcal{I} := [1,n]$, we reparametrize $\tilde{f}$ with a monotone function $\rho : \mathcal{I} \to \mathcal{J}$ such that $\rho(i) = t_i$. We can set ρ to be the monotone interpolant described by Fritsch and Carlson [FC80], for example. We then define $f = \tilde{f} \circ \rho$, i.e.

$$f : \mathcal{I} \to \mathbb{R}^2, \qquad f(t) = \sum_{j=1}^{k} B_j^{\mathbf{u}}(\rho(t))\,\mathbf{f}_j.$$

2.4. Summary

We now have the complete chain of signals given by (1), forming the continuous function ψ from the time interval $\mathcal{I}$ to the shape space $\mathcal{S}$. When used to represent an existing deforming mesh sequence, the function h extracts shape redundancy, g extracts pose redundancy and f extracts time redundancy. This leads to a representation of each frame $\mathbf{z}_i$ as $\mathbf{x}_i$ in the shape subspace, $\mathbf{p}_i$ in the pose space, $t_i \in \mathcal{J}$, and finally $i \in \mathcal{I}$:

$$\begin{tikzcd} \mathcal{I} \arrow[r, "f"] & \mathbb{R}^2 \arrow[r, "g"] & \mathbb{R}^m \arrow[r, "h"] & \mathcal{S} \\ \{i\}_1^n \arrow[u, phantom, "\cap"] & \{\mathbf{p}_i\}_1^n \arrow[u, phantom, "\cap"] & \{\mathbf{x}_i\}_1^n \arrow[u, phantom, "\cap"] & \{\mathbf{z}_i\}_1^n \arrow[u, phantom, "\cap"] \end{tikzcd}$$

Each instance of our representation ψ has three component parts: f forms a *time signal* in the pose space, g gives a *pose signal* in the shape subspace, and the linear map h is our *shape signal* in the shape space $\mathcal{S}$.

3. Reconstruction from frames

The representation described in Section 2 can be used to create deforming mesh sequences from scratch (see Section 4.4), but it is likely to be most useful when applied to the reconstruction of an existing deforming mesh sequence specified as discrete frames in $\mathcal{S}$. Converting a sequence to our representation makes it possible to apply a variety of editing, smoothing and resampling operations (see Section 4). First, however, we must be able to find the parameters of the signals f, g and h from discrete frame samples $\{\mathbf{z}_i\}_1^n \subset \mathcal{S}$. Our ultimate goal is to minimize

$$\sum_{i=1}^{n} \|\mathbf{z}_i - \psi(i)\|^2 \tag{2}$$

but the non-linear function g makes it very expensive to search for this ideal solution directly. Instead we minimize error in each of the three reconstruction steps separately, as detailed in this section.

3.1. Constructing h

Our first goal is to find the shape signal parameters $\{\mathbf{h}_j\}_1^m$, and the representation $\{\mathbf{x}_i\}_1^n \subset \mathbb{R}^m$ of the frames in the shape subspace, to minimize

$$\sum_{i=1}^{n} \|\mathbf{z}_i - h(\mathbf{x}_i)\|^2. \tag{3}$$

For any $\mathbf{h}_0$, the $\mathbf{H}$ minimizing (3) is given by the closest rank-m approximation to the offset frames $\{\mathbf{z}_i - \mathbf{h}_0\}_1^n$. If these vectors are entered as the rows of a matrix $\mathbf{Y} \in \mathbb{R}^{n \times d}$, then a singular value decomposition (SVD) gives $\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ and the closest rank-m approximation is given by the singular vectors corresponding to the m largest singular values. That is, with $\mathbf{D}$ ordered by singular value, $\mathbf{H}$ is defined as the leftmost m columns of $\mathbf{V}$. Since the singular vectors are orthogonal, each vector $\mathbf{x}_i$ is then simply $\mathbf{H}^T(\mathbf{z}_i - \mathbf{h}_0)$. We choose $\mathbf{h}_0$ as the mean $\frac{1}{n}\sum_{i=1}^n \mathbf{z}_i$, as the SVD then gives a principal component analysis [Jol02] of the initial frames $\{\mathbf{z}_i\}_1^n$. Alexa and Müller [AM00] pioneer this application of principal component analysis to deforming mesh sequences.

3.2. Constructing g

Given the vectors $\{\mathbf{x}_i\}_1^n$, our next task is to find radial basis function sites $\{\mathbf{s}_j\}_1^l$ and coefficients $\{\mathbf{g}_j\}_0^{l+2}$, as well as pose-space positions $\{\mathbf{p}_i\}_1^n$ for the n frames. In this reconstruction step the error is

$$\sum_{i=1}^{n} \|\mathbf{x}_i - g(\mathbf{p}_i)\|^2. \tag{4}$$

However, if h uses the construction described in Section 3.1, then minimizing (4) is equivalent to minimizing

$$\sum_{i=1}^{n} \|\mathbf{z}_i - h(g(\mathbf{p}_i))\|^2,$$

which completely accounts for the contributions of g and h to the overall error (2).

We choose the radial basis function as $\phi(r) = r^2 \log r$, to give g the properties of a *thin-plate spline*. By using a global rather than compactly-supported function, we obtain good properties when g is used for extrapolation (i.e. evaluated far from the sites $\{\mathbf{s}_j\}_1^l$). Thin-plate splines also minimize the thin-plate bending energy

$$\int_{\mathbb{R}^2} \left(\frac{\partial^2 g}{\partial p_1 \partial p_1}\right)^2 + 2\left(\frac{\partial^2 g}{\partial p_1 \partial p_2}\right)^2 + \left(\frac{\partial^2 g}{\partial p_2 \partial p_2}\right)^2 \mathrm{d}\mathbf{p} \tag{5}$$

where $\mathbf{p} = (p_1, p_2)$. Using this definition for ϕ therefore creates a pose manifold with guaranteed smoothness.

Assuming we have chosen $\{\mathbf{p}_i\}_1^n$ and $\{\mathbf{s}_j\}_1^l$, we can calculate $\{\mathbf{g}_j\}_0^{l+2}$ by finding one QR decomposition and solving a linear system of size $n \times l$ [Wah90]. The linear system is dense, given our choice of ϕ as a non-local function, but not too large as we expect to work with short sequences (see Section 1), so the number of frames n is limited. As a result these parameters of g can be found quickly (see Table 1) despite the density of the linear system.

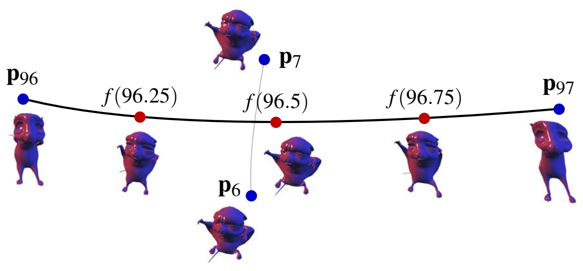


Figure 3: *A saddle in the pose signal created by a poor choice of positions $\{\mathbf{p}_i\}_1^n$. We show $\mathbf{p}_i$ marked in blue for four frames, and the two parts of the time signal that interpolate those positions. This layout in pose space forces a surface g which gives an unnatural transition along the darkened part of the time signal (sample evaluations $\psi(t)$ are shown with $f(t)$ marked by red points). We avoid this by incorporating user constraints into an MDS solution; Figure 5 shows a representation of this sequence with a better choice for $\{\mathbf{p}_i\}_1^n$.*

3.2.1. Choosing pose-space positions

Finding a thin-plate approximation that minimizes (4) depends on choosing pose-space positions $\{\mathbf{p}_i\}_1^n$ at which to evaluate g. This is not a straightforward task: essentially we have a sampled curve in a high-dimensional space $\mathbb{R}^m$, which we want to incorporate into a surface. Furthermore, we want the positions $\mathbf{p}_i$ and $\mathbf{p}_j$ to be close wherever $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar; otherwise g will not fulfil the goal of extracting pose redundancy. This rules out the trivial solution, for example, where the frames are positioned evenly along a single line.

The condition on proximity suggests that to find $\{\mathbf{p}_i\}_1^n$, we should consider minimizing an energy of the form

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\|\mathbf{p}_i-\mathbf{p}_j\|-\|\mathbf{x}_i-\mathbf{x}_j\|\right)^2 \qquad (6)$$

which compares all pairwise distances in the pose space with the corresponding distances in the shape subspace. This is exactly the stress energy minimized by multidimensional scaling (MDS) [BG05]. However, MDS is inadequate if used alone, since the pairwise comparison (6) takes no account of the ordering between the frames; MDS finds positions for a collection of point samples, not for a (discrete) curve. This can lead to unwanted saddles in the pose signal: if four or more poses are similar and are therefore correctly placed close together by MDS minimizing (6), a surface interpolating the poses may give unnatural transitions between adjacent frames (see Figure 3). Reconstruction error is not affected by this problem, since the *discrete frames* are well reproduced by minimizing the least-squares error (4). However, poor *continuous transitions* between frames would limit the usefulness of the representation to applications like resampling (Section 4.2).

To work around this limitation, we propose the addition of user constraints to an MDS solution. This technique places a human 'in the loop', allowing the user to make the semantic decisions on which parts of the sequence should be judged as similar for a particular application. Our system allows the user to specify pose-space positions $\mathbf{p}_i$ for some values of i, but finds the remaining positions automatically by minimizing (6) under the given constraints. The process is fast enough to give real-time feedback to the user, who can therefore work in collaboration with MDS to produce the desired configuration (see accompanying video).

3.2.2. Selecting radial basis function sites

Once the pose-space positions have been set, the last task before constructing g is to select sites for the radial basis function approximation. If $l = n$, a thin-plate spline *interpolant* allows us to find a manifold where the error (4) is zero, by setting each site $\mathbf{s}_i$ to the pose-space position $\mathbf{p}_i$ for $i = 1,\ldots,n$. Where there is pose redundancy to extract, we expect to have $l < n$ and therefore non-zero error, but this property motivates us to choose $\{\mathbf{s}_j\}_1^l$ as a subset of $\{\mathbf{p}_i\}_1^n$. To do so, we calculate an order in which frame positions are chosen as sites, given by $\{\sigma_i\}_1^n$ as a permutation of $\{1,\ldots,n\}$. For a given l, the selected sites are then taken from the first l indices of the calculated ordering, giving $\mathbf{s}_j = \mathbf{p}_{\sigma_j}$ for $j = 1,\ldots,l$. This results in a nested sequence of reconstruction spaces as l changes, where increasing l is guaranteed to reduce (4).

We choose to add sites where the error $\mathbf{x}_i - g(\mathbf{p}_i)$ is greatest. This heuristic leads to the following greedy algorithm for calculating the required order [Fas07, Ch. 21]:

1. Compute a best-fit plane to the points $\{\mathbf{x}_i\}_1^n$. If an SVD is used to compute $\mathbf{H}$ as described in Section 3.1, then the spanning vectors of the plane are given by $\mathbf{h}_1$ and $\mathbf{h}_2$.
2. Set $\{\sigma_j\}_1^4$ to be the indices of the four points with the greatest distance to this best-fit plane.
3. For $l = 4$ to $n-1$;
   - Calculate g using l sites, as given by $\{\mathbf{p}_{\sigma_j}\}_1^l$,
   - Let σ_{l+1} be the index of the point with the greatest error $\mathbf{x}_i - g(\mathbf{p}_i)$ from the current approximation, ignoring those frames which are already chosen for radial basis function sites:

$$\sigma_{l+1} = \underset{i\notin\{\sigma_j\}_1^l}{\operatorname{argmax}}(\mathbf{x}_i - g(\mathbf{p}_i)).$$

Note that in each step of this algorithm, we construct a least-squares fit minimizing (4) for *all* frames, using a space of radial basis functions centred on a *subset* of pose-space positions. This distributes error more evenly across the reconstructed frames than finding a thin-plate spline *interpolant* to the selected subset [DB02].

3.3. Constructing f

To construct the representation (1), our one remaining task is to find the parameters of f. Each frame is now represented by its position $\mathbf{p}_i$ in pose space, and this final stage replaces these discrete points with a continuous spline approximation. As explained at the start of this section, the error we minimize here is

$$\sum_{i=1}^{n} \|\mathbf{p}_i - f(i)\|^2 \equiv \sum_{i=1}^{n} \|\mathbf{p}_i - \tilde{f}(t_i)\|^2 \qquad (7)$$

as an approximation that leads to small, but not necessarily minimal, values of the overall error (2). To compute the spline fit, we must choose parameter values $\{t_i\}_1^n$, knots $\{u_j\}_0^{k+1}$, and control points $\{\mathbf{f}_j\}_1^k$. As with the radial basis function approximation (Section 3.2), f is linear in the coefficients $\{\mathbf{f}_j\}_1^k$, and so these can be found by solving an $n \times k$ linear system [PT97]. However, f is a non-linear function of the parameter values and knots, and these must be chosen first.

For parametrization, Lee [Lee89] suggests setting the parameter-space interval $t_{i+1} - t_i$ as $\|\mathbf{p}_{i+1} - \mathbf{p}_i\|^{\alpha}$. We choose $\alpha = \frac{1}{2}$ to give the 'centripetal' parametrization which usually yields the best results. Then for a given k, there are a large variety of algorithms for selecting $\{u_j\}_0^{k+1}$ from $\{t_i\}_1^n$; Cox et al. [CHK02] provide a good summary of available methods. We use the algorithm described by Piegl and Tiller [PT97, Sec. 9.4.1], as this ensures that the Schoenberg-Whitney [SW53] conditions are always satisfied for any choice of k. Their method also degenerates to the interpolation configuration $u_{i+1} = t_i$ $(i = 1, \ldots, n)$ if $k = n + 2$: the setting for k which guarantees that the error (7) is zero.

3.4. Signal fidelity

Apart from the optional user constraints described in Section 3.2.1, each of the reconstruction steps described above is completely automatic, and depends only on a constant which specifies the signal fidelity. The quality of the shape signal h is determined by m, and the parameters of the signal are $\{\mathbf{h}_j\}_0^m$. The pose signal g is given by the parameters $\{\mathbf{g}_j\}_0^{l+2}$ and $\{\mathbf{s}_j\}_1^l$, where quality is set by the constant l, and the accuracy of the time signal f is controlled by k, with parameters $\{\mathbf{f}_j\}_1^k$, $\{t_i\}_1^n$ and $\{u_j\}_0^{k+1}$.

Each of the three signals includes interpolation as a special case: if $k - 2 = l = m + 1 = n$, then we are guaranteed a perfect reconstruction where (2) is zero. On the other hand, lower bounds on these constants are $k \geq 4$, $l \geq 3$ and $m \geq 0$, so there is scope to store far less data if the resulting reconstruction error is still acceptable.

By specifying a tolerance for error, it is also possible to determine k, l and m automatically. These constants reflect the nature of the dynamic geometry under consideration; for example, the sequence shown in Figure 5 is more repetitive than that in Figure 6, with l at 46% of n in the first case and 71% in the second. The complicated time signal in Figure 4 requires a higher value for k than that shown in Figure 1. Finally the simple range of shapes used in Figure 8 leads to the smallest value for m/n out of all the given examples.

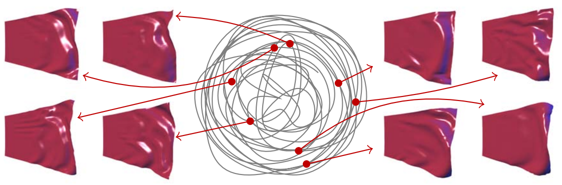


Figure 4: *Our representation reconstructing a billowing flag in the deformation gradient shape space. The simulated motion of the flag creates a complicated time signal; the repetitive effect of the wind is evident as consistent circular motion in the pose space. Sample reconstructions $\psi(i)$ are shown with links to the corresponding positions $f(i)$. This example reconstructs 100 frames using $k = 98$, $l = 100$ and $m = 79$.*

							Time to find (ms)			
Fig.	$\mathcal{S}$	d	k	l	m	n	f	RBF	g	h
1	$\mathbb{R}^{3V}$	25293	16	12	7	24	1	1	10	166
4	$\mathbb{R}^{9F+3}$	51987	98	100	79	100	31	13	1295	7625
5	$\mathbb{R}^{3V}$	11505	82	46	19	99	26	5	234	1326
7c	$\mathbb{R}^{9F+3}$	151590	16	12	7	24	1	6	57	135
6	$\mathbb{R}^{3V}$	8712	155	144	42	204	239	39	5626	5127

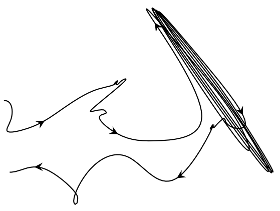
Table 1: *Computation times and constants for examples in this paper, using an Intel Core i7 processor clocked at 2.67 GHz. Under 'RBF' we list the average time to find a single radial basis function approximation; we also give the time required to find g by iteratively computing approximants as described in Section 3.2.2. The data given for Figure 7c refers to our representation for the galloping horse sequence with $\mathcal{S} = \mathbb{R}^{9F+3}$, before we replace the shape signal. For details of the shape spaces $\mathcal{S}$, see Section 5.*

4. Applications

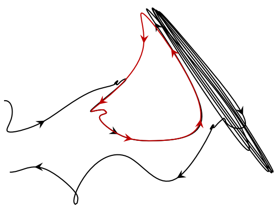
The continuous nature of our representation allows a rich set of editing operations by modifying or replacing one signal component, while leaving other parts of the representation intact. This section introduces several possible editing applications, using the examples that we summarize in Table 1. Our implementation uses OpenMesh [BSBK02] as a mesh data structure, and Eigen [GJ*11] and CHOLMOD [CDHR08] for dense and sparse linear algebra respectively. CHOLMOD is used only for shape spaces which involve solving sparse linear systems (see Section 5); it has no role when using the vertex-coordinate shape space $\mathcal{S} = \mathbb{R}^{3V}$.

4.1. Time-local editing

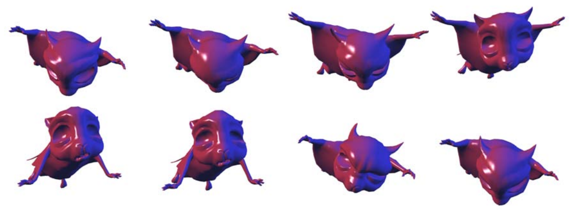
As the time signal f uses B-splines, which are locally-supported, we can modify the time signal curve by editing the control points $\{\mathbf{f}_j\}_1^k$ with the guarantee that only the local region surrounding an edit will be affected. Furthermore the smoothness of the pose signal, given by minimizing the thin-plate energy (5), means that small modifications to the



(a) Original time signal.



(b) New time signal with added loop shown in red.



(c) Frames from the newly-added loop.

Figure 5: *An example of an editing operation which is local in time. To the time signal shown in (a) we add a loop, to gain the new time signal shown in (b). This adds frames to the deforming mesh sequence which blend smoothly with the existing frames, while leaving unrelated parts of the sequence unmodified. Here* $(k,n) = (82,99)$ *before adding the loop, and* $(97,118)$ *afterwards, while* $l = 46$ *and* $m = 19$.

spline control points result in correspondingly small changes to the deforming mesh sequence.

Figure 5 shows an example time signal edit to add a new section to the deforming mesh sequence which joins smoothly to the existing frames. At all frames i where t_i falls outside of the support of the modified control points' basis functions, the edit has no affect and hence the sequence is unmodified.

4.2. Resampling

Besides modifying the time signal *curve*, it is also possible to edit the *rate* of the time signal by leaving the curve unmodified, and changing only the positions from the curve at which frames are sampled. This generates a new deforming mesh sequence $\{\psi(\tau_i)\}_1^{n'}$ by sampling ψ at the positions $\{\tau_i\}_1^{n'}$, where the new sequence has n' frames, and n' may be different to n.

If the resulting sequence is played at the same rate as the original, this has the effect of speeding up or slowing down the sequence. In Figure 6, we show an example which sets the positions $\{\tau_i\}_1^{n'}$ to produce both fast- and slow-motion effects. Alternatively, the resampling can be applied uniformly across an entire deforming mesh sequence by setting $\tau_i = hi$ for some $h \in \mathbb{R}_+$, and the resulting sequence could be played at a rate $1/h$, to perform *frame rate conversion*; an important process for video [WZHT10] that has not previously been available for deforming mesh sequences.

4.3. Shape signal editing

To reuse existing mesh sequences, Sumner and Popović introduce deformation transfer [SP04], where the deformation of a mesh in an existing sequence is transferred to a new model. Baran et al. [BVGP09] generalize their technique to *semantic deformation transfer*, where the user provides a set of example matching poses, each one pairing a deformation of the source mesh with a deformation of the target. We can apply our representation to the same task by replacing the shape signal but leaving the pose and time signals unmodified.

The input for this operation is identical to that for Baran et al. [BVGP09]: we need an existing deforming mesh sequence $\{\mathbf{z}_i\}_1^n \subset \mathcal{S}$, and a set of examples $\{(\mathbf{q}_j, \mathbf{r}_j)\}_0^m \subset \mathcal{S} \times \mathcal{S}'$ that show the correspondence between the existing shape space $\mathcal{S}$ and the replacement $\mathcal{S}'$. In general $\mathcal{S} \neq \mathcal{S}'$, and the new shape space may describe a mesh with completely different connectivity. Like Baran et al., we expect both $\mathbf{q}_0$ and $\mathbf{r}_0$ to show the meshes in a reference pose.

To represent the existing sequence in terms of the provided examples, we simply use $\mathbf{h}_0 = \mathbf{q}_0$ and $\mathbf{h}_j = \mathbf{q}_j - \mathbf{q}_0$ $(j = 1, \ldots, m)$ in place of the construction described in Section 3.1. Since the vectors $\{\mathbf{h}_j\}_1^m$ obtained in this way are not orthogonal, in general, we then need to solve the normal equation

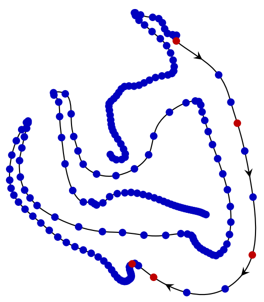
$$\mathbf{H}^T\mathbf{H}\mathbf{X} = \mathbf{H}^T\mathbf{Y}^T$$

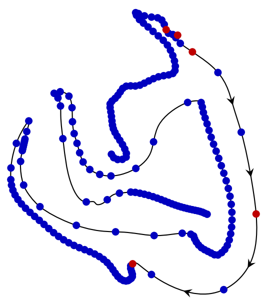
to find the least-squares solution for the shape coordinates $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. The pose and time signals can be found from the $\{\mathbf{x}_i\}_1^n$ in the usual way described in Sections 3.2 and 3.3.

The final step that completes the deformation transfer is to replace $\mathbf{h}_0$ with $\mathbf{r}_0$ and $\mathbf{h}_j$ with $\mathbf{r}_j - \mathbf{r}_0$ for $j = 1, \ldots, m$. This gives a new representation $\psi' : \mathcal{I} \to \mathcal{S}'$ that maps to the replacement shape space $\mathcal{S}'$ such that the example shapes $\{\mathbf{q}_j\}_0^m$ are converted into their corresponding target shapes $\{\mathbf{r}_j\}_0^m \subset \mathcal{S}'$. Figure 7 shows an example transfer of a galloping sequence from a horse to an elephant using eight examples $(m = 7)$.

4.4. Dynamic geometry from scratch

Our representation can also use arbitrary time and pose signals, making it possible to design deforming mesh sequences simply by editing the control points $\{\mathbf{f}_j\}_1^k$ and pose-space positions $\{\mathbf{p}_j\}_1^l$. We show an example sequence created in this way in Figure 8. For this application, there is no need to use MDS as described in Section 3.2.1, but the pose signal can otherwise be constructed in the same way, including the selection of radial basis function sites described in Section 3.2.2. The time signal can be entirely user-specified by combining the control points $\{\mathbf{f}_j\}_1^k$ with uniform knots $u_j = j$ (for $j = 0, \ldots, k+1$) and uniform parameter values $t_i = hi$ $(i = 1, \ldots, n$ and $h \in \mathbb{R}_+)$.


(a) Time signal, with positions $\{f(i)\}_1^n$ marked.


(b) After resampling, showing modified positions $\{f(\tau_i)\}_1^{n'}$.

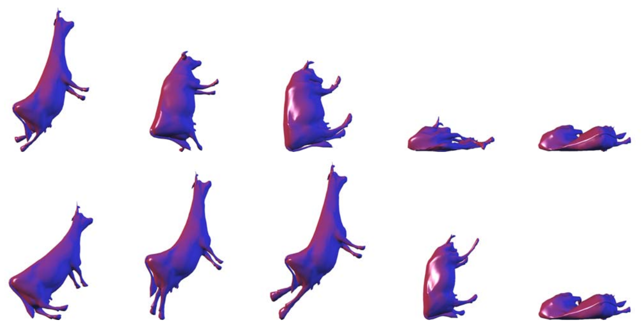
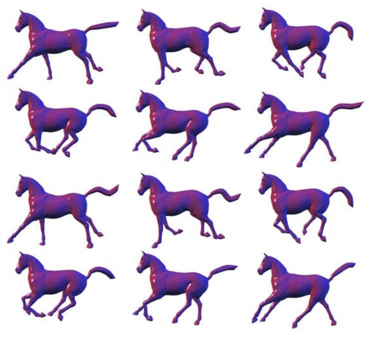
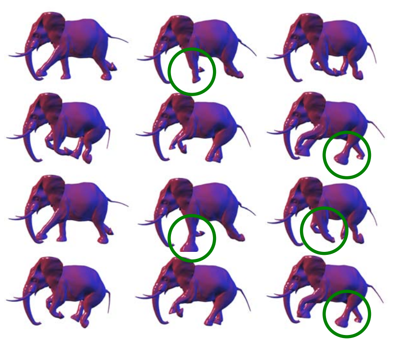

(c) Example frames $\psi(i)$ (top) before resampling and (bottom) after resampling; the corresponding positions in pose space $f(i)$ are marked in red in (a) and (b).

Figure 6: *An example of non-uniform resampling (Section 4.2) on a reconstructed sequence that shows a cow under elastic motion. The time signal curve, shown in (a) and (b), has been resampled in (b) to slow down parts of the sequence where the cow is lifted, and speed up sections where the cow drops to the ground. This leads to the new deforming mesh sequence shown in the bottom half of (c). This sequence uses $k = 155$, $l = 144$, $m = 42$ and $n' = n = 204$.*


(a) Original deforming mesh sequence.


(b) Deformation transfer using the vertex-coordinate shape space $\mathcal{S} = \mathbb{R}^{3V}$.

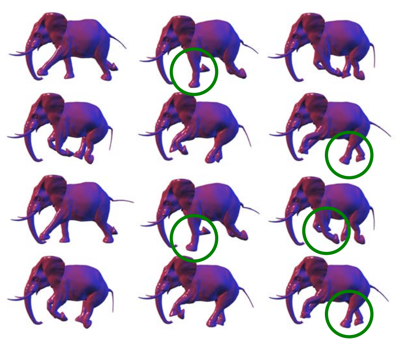
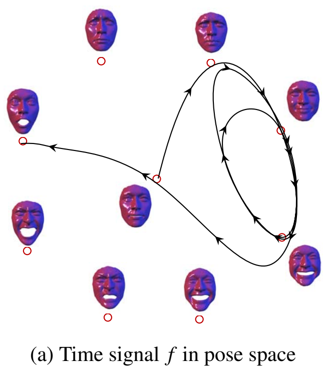

(c) Deformation transfer using the deformation gradient shape space $\mathcal{S} = \mathbb{R}^{9F+3}$.

Figure 7: *An example of deformation transfer [BVGP09]; we transfer the animation shown in (a) to an elephant model, by replacing the 8 components $\{\mathbf{h}_j\}_0^{m=7}$ of the shape signal but keeping the time and pose signals intact. This figure shows the process for two different shape spaces: the result in (c) using the deformation gradient shape space avoids the linear-interpolation artifacts shown in (b). Notable differences between the two results are highlighted. This example uses $k = 16$, $l = 12$ and $m = 7$.*


(a) Time signal f in pose space

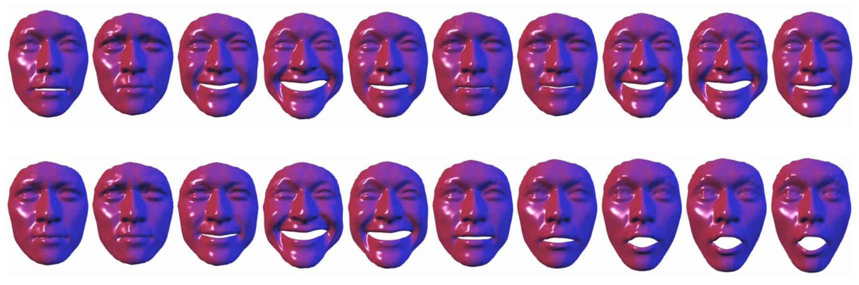

(b) Frames $\psi(i)$ from the resulting deforming mesh sequence

Figure 8: *A deforming mesh sequence created from scratch by positioning facial expressions in pose space. Positions $\{\mathbf{p}_i\}_1^9$ are shown as open circles in (a), accompanied by the corresponding facial expression. The time signal f starts at a neutral expression, loops three times through a smile, and finishes by moving towards a surprised expression. This leads to the deforming mesh sequence shown in (b). Note that the four poses which are far from the curve f do not appear in the resulting sequence.*

5. Shape spaces

Our representation is designed to be applicable to a wide range of shape spaces. However, we do need $\mathcal{S}$ to be a Euclidean space, so that mesh shapes can be combined with the normal Euclidean operators. This allows us to use all the tools of linear algebra, but it rules out the space Kircher and Garland [KG08] propose, as their mesh representation requires specialized operators to combine quaternion rotations. The shape space described by Kilian et al. [KMP07] is also not a Euclidean space, and therefore not applicable. However, a wide range of shape spaces do allow the use of Euclidean operators, including:

- the vertex-coordinate shape space $\mathcal{S} = \mathbb{R}^{3V}$,
- the deformation gradient [SP04] shape space $\mathcal{S} = \mathbb{R}^{9F+3}$, which stores 3 rotation and 6 scale/shear parameters for each of the F faces in the mesh, and a further 3 values to resolve translational invariance,
- the shape space based on linear rotation-invariant coordinates [LSLCO05], where $\mathcal{S} = \mathbb{R}^{3E+6}$, storing 3 parameters for each of the E edges in the mesh, and a further 6 values to resolve translational and rotational invariance,
- the shape space based on storing edge lengths and dihedral angles used by Winkler et al. [WDAH10] and Fröhlich and Botsch [FB11], where $\mathcal{S} = \mathbb{R}^{2E+6}$,
- the shape space developed by Baran et al. [BVGP09] using patch-based linear rotation-invariant coordinates.

The choice of shape space can have a large impact on the appearance of the time signal, the result of editing operations, and the amount of data our representation requires. Most of the examples in this paper have remained with the simplicity of the vertex-coordinate shape space $\mathcal{S} = \mathbb{R}^{3V}$, showing that this choice is sufficient for many practical applications. We are often able to avoid the problems associated with large interpolations in this space [KG08], since a deforming mesh sequence is densely sampled in time, and we are typically interested in evaluating the pose signal g only near a reconstructed time signal f. Furthermore, since the pose signal aims to build a meaningful surface of shapes, a linear path in the pose space may follow typical poses more faithfully than a linear path in the shape subspace.

However, for some applications, particularly when evaluating g far from the sites $\{\mathbf{s}_j\}_1^l$, or when specifying a pose signal with a very large separation between the positions $\{\mathbf{p}_i\}_1^n$, artifacts can appear using the shape space $\mathcal{S} = \mathbb{R}^{3V}$, motivating the use of a higher-order shape space instead (see Figure 7 for an example). The main cost of doing so is that the reconstruction problems discussed in Section 3 require the solution of larger linear systems, since the dimension d of $\mathcal{S}$ is then greater than $3V$. We also find that the fidelity constants l and m usually need to be larger for a higher-order shape space, possibly because of the additional redundancy that is introduced by these representations.

6. Conclusion

The representation (1) allows deforming mesh sequences to be modified and reused with a variety of time-coherent techniques. We can evaluate ψ not only at the integers, but at any $t \in \mathcal{I}$, since ψ is a continuous *curve* in $\mathcal{S}$ which approximates the given discrete mesh frames. Decomposing the dynamic data into three separate signals allows the flexibility of modifying one part while keeping other signals intact.

One key to the editable nature of our representation is the choice of a 2D pose space, and therefore a pose signal *surface*. Future work could consider pose signal volumes, or even manifolds of higher dimension. However, any gain in expressiveness would have to be balanced against the increased difficulty, in higher dimensions, of editing and visualizing a deforming mesh sequence in pose space.

A significant limitation to our system at present is the need to incorporate user constraints to find satisfactory pose-space positions $\{\mathbf{p}_i\}_1^n$. It is always possible to resolve the problem illustrated in Figure 3, but it is sometimes necessary to add many constraints, and it may not be clear where they should be introduced. Unfortunately, addressing this problem is not as simple as eliminating intersections; occasionally a time signal which intersects itself is necessary or even desirable, as long as the continuous transitions between frames remain reasonable (see Figure 5 for an example). Future work may be able to detect and react to the regions of negative Gaussian curvature in the pose signal which are associated with this defect, leading to a fully automatic reconstruction procedure.

Another limitation we introduced is in the number of frames that our representation can handle. Although there is no theoretical limit on duration, long sequences can be slower to reconstruct and may lead to illegible time signals. However, our representation can always be used on subsections of longer sequences; for time-local editing (Section 4.1), this is sufficient. Future work could extend this idea by reconstructing a long sequence, possibly with the aid of multi-scale techniques, but displaying only a limited section of the time signal. A user could then 'scrub through' the sequence until the region of interest is displayed in the pose space.

Figure 8 shows that we can add additional shapes to the pose space to use a pose signal in new contexts. It would therefore be powerful to combine our representation with a direct manipulation system, allowing the user to create new shapes with existing techniques, and then add them directly to the pose space for further time-coherent editing.

One final area for future work is an improved visualization for a pose signal, without having to 'explore' by evaluating the pose space at different positions. We envisage a user interface which allows zooming and panning in the pose space, while showing the pose signal evaluated at an adaptive grid of points. It would also be advantageous to be able to modify the view on each evaluated shape, by making a standard 3D trackball interface apply to all the sampled shapes at once.

Acknowledgements

This work was supported by the SNF under project number 200021-134639. We thank Robert Sumner for the horse, elephant and face meshes (Figures 1, 7 and 8), Matthias Müller for the cow sequence (Figure 6), and the Blender Foundation (www.blender.org) for the sequence that appears in Figures 3 and 5 under a Creative Commons Attribution 3.0 License.

References

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Comput. Graph. Forum 19*, 3 (2000), 411–418. 4

[ARL*09] ALEXANDER O., ROGERS M., LAMBETH W., CHIANG M., DEBEVEC P.: The Digital Emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH 2009 Courses* (2009), pp. #12:1–15. 2

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS J.: SCAPE: Shape completion and animation of people. *ACM Trans. Graph. 24* (2005), 408–416. 2

[BG05] BORG I., GROENEN P. J. F.: *Modern multidimensional scaling: Theory and applications*, second ed. Springer, 2005. 5

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3D characters. *ACM Trans. Graph. 26* (2007), #72:1–8. 2

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: OpenMesh – a generic and efficient polygon mesh data structure. In *OpenSG Symposium* (2002). 6

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM Trans. Graph. 28*, 3 (2009), #36:1–6. 2, 7, 8, 9

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35* (2008), #22:1–14. 6

[CHK02] COX M., HARRIS P., KENWARD P.: Fixed- and free-knot univariate least-squares data approximation by polynomial splines. In *Algorithms for Approximation IV* (2002), Levesley J., Anderson I., Mason J. C., (Eds.), pp. 330–345. 6

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. *ACM Trans. Graph. 27*, 3 (2008), #98:1–10. 1

[dATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Comput. Graph. Forum 27*, 2 (2008), 389–397. 2

[dB01] DE BOOR C.: *A Practical Guide to Splines*, revised ed., vol. 27 of *Applied Mathematical Sciences*. Springer-Verlag, 2001. 3

[DB02] DONATO G., BELONGIE S.: Approximate thin plate spline mappings. In *Computer Vision – ECCV 2002*, Heyden A., Sparr G., Nielsen M., Johansen P., (Eds.), vol. 2352 of *LNCS*. Springer, 2002, pp. 21–31. 5

[Fas07] FASSHAUER G. E.: *Meshfree approximation methods with* MATLAB. World Scientific, 2007. 5

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8 (2011), 2246–2257. 2, 9

[FC80] FRITSCH F. N., CARLSON R. E.: Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal. 17*, 2 (1980), 238–246. 4

[FRDC04] FAVREAU L., REVERET L., DEPRAZ C., CANI M.-P.: Animal gaits from video. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 277–286. 2

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26* (2007), #49:1–7. 1

[GJ*11] GUENNEBAUD G., JACOB B., ET AL.: Eigen version 3. http://eigen.tuxfamily.org, 2011. 6

[Jol02] JOLLIFFE I. T.: *Principal Component Analysis*, second ed. Springer Series in Statistics. Springer, 2002. 4

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Trans. Graph. 24*, 3 (2005), 399–407. 2

[KG06] KIRCHER S., GARLAND M.: Editing arbitrarily deforming surface animations. *ACM Trans. Graph. 25* (2006), 1098–1107. 1, 2

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Trans. Graph. 27*, 2 (2008), #12:1–13. 2, 9

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Trans. Graph. 26* (2007), #64:1–8. 2, 9

[KSO10] KAVAN L., SLOAN P.-P., O'SULLIVAN C.: Fast and efficient skinning of animated meshes. *Comput. Graph. Forum 29*, 2 (2010), 327–336. 2

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH 2000* (2000), Akeley K., (Ed.), pp. 165–172. 1, 2

[Lee89] LEE E. T. Y.: Choosing nodes in parametric curve interpolation. *Comput.-Aided Des. 21*, 6 (1989), 363–370. 6

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3 (2005), 479–487. 2, 9

[PT97] PIEGL L. A., TILLER W.: *The NURBS Book*. Springer, 1997. 4, 6

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405. 2, 7, 9

[SW53] SCHOENBERG I. J., WHITNEY A.: On Pólya frequency functions. III. The positivity of translation determinants with an application to the interpolation problem by spline curves. *Trans. Am. Math. Soc. 74*, 2 (1953), 246–259. 6

[Wah90] WAHBA G.: *Spline models for observational data*, vol. 59 of *CBMS-NSF Regional Conference series in applied mathematics*. Society for Industrial Mathematics, 1990. 3, 4

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: Realtime performance-based facial animation. *ACM Trans. Graph. 30* (2011), #77:1–10. 1

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Comput. Graph. Forum 29*, 2 (2010), 309–318. 2, 9

[WZHT10] WANG C., ZHANG L., HE Y., TAN Y.: Frame rate up-conversion using trilateral filtering. *IEEE Trans. Circuits Syst. Video Technol. 20*, 6 (2010), 886–893. 7

[XZY*07] XU W., ZHOU K., YU Y., TAN Q., PENG Q., GUO B.: Gradient domain editing of deforming mesh sequences. *ACM Trans. Graph. 26* (2007), #84:1–10. 1, 2

[YYPM11] YANG Y.-L., YANG Y.-J., POTTMANN H., MITRA N. J.: Shape space exploration of constrained meshes. *ACM Trans. Graph. 30* (2011), #124:1–12. 2