Computers & Graphics 82 (2019) 129–139

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SMI 2019

Efficient 4D shape completion from sparse samples via cubic spline fitting in linear rotation-invariant space

Qing Xia[a,1,*], Chengju Chen[a,1], Jiarui Liu[a], Shuai Li[a,b], Aimin Hao[a], Hong Qin[c]

[a] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
[b] Beihang University Qingdao Research Institute, China
[c] Department of Computer Science, Stony Brook University, USA

ARTICLE INFO

Article history:
Received 10 March 2019
Revised 21 May 2019
Accepted 22 May 2019
Available online 28 May 2019

Keywords:
Shape sequence completion
Cubic spline fitting
Linear rotation-invariant space

ABSTRACT

Computer animation is frequently produced via interpolating a few sparse samples created by artists or reverse-engineered from physical prototypes, however, existing interpolation techniques fall short in efficiently generating a smooth 4D shape sequence from sparse samples. In this paper, we extend traditional curve fitting technique to 4D shape completion in shape space with novel technical components. In particular, we seek a smooth 4D shape sequence by minimizing the total shape distortion along the sequence trajectory. After embedding the shapes into a linear rotation-invariant feature space, the complex global minimization of shape distortion in shape space can be converted into simple cubic spline fitting problems in feature domains, which can be solved analytically. With cubic splines, we can not only handle in-between shapes interpolation, but also perform extrapolation towards more exciting results. To further improve the computational efficiency, we devise a hierarchical framework, in which the shape space is decomposed into high-frequency and low-frequency domains, the interpolation is only operated on the low-frequency domain, while the high-frequency details are enabled via deformation transfer techniques. We have conducted extensive experiments and comprehensive evaluations that showcase many attractive advantages of our novel method, including smooth interpolation between shapes, plausible extrapolation outside conventional shape domain, robustness under large deformations, and interactive performance for complicated shapes with high-quality details.

1. Introduction and motivation

Interpolation of shapes is important for various tasks in computer graphics, especially for the generation of computer animation. In practice, an animated shape sequence is usually generated via iteratively interpolating between successive key-frames created by artists with 3D modeling software, as shown in Fig. 1. In technical essence, shape interpolation is to find averaged shapes of two or more reference shapes according to certain non-linear geometric or physical properties [3–5]. These refinement schemes actually do linear interpolation between shapes in corresponding feature space and then reconstruct the averaged shape according to the interpolated features, and the transition between any two shapes of these methods usually seems desirable. However, when considering a smooth 4D shape sequence completed from a few sparse samples, simply subdividing between two successive key-frames may lead to artifacts around key-frames. As shown in the top row of Fig. 1, the linearity in shape space results in an obvious sharp corner in the middle of the cactus sequence. Thus, it is still challenging to complete a smooth 4D shape sequence from a few sparse shapes sampled along temporal axis, namely to produce a smooth curve in shape space that goes through those sparse sampled points, and guarantee the quality and efficiency simultaneously.

Smoothly interpolating key-frames of animated characters could date back to the pioneering works of Kochanek and Bartels [6] and Lasseter [7]. However, directly interpolating in Euclidean space will give rise to many artifacts, such as distortion, shearing, and other undesirable transitions, and thus researchers move their foci on interpolation in shape feature space afterwards. The idea of using curves for shape interpolation is introduced by Kilian et al. [1], wherein they treat shapes as points in a shape space and compute geodesics for interpolating between two given shapes. However, the geodesic between two endpoints with natural boundary is actually a straight line, and the curve fitted here is piece-wise linear rather than smooth everywhere. Inspired by the curve processing technique in Euclidean space, Brandt et al. [8] propose geometric flows to smooth the motions or animations of shape

* Corresponding author.
E-mail addresses: neijiangxiaqing@gmail.com (Q. Xia), qin@cs.stonybrook.edu (H. Qin).
[1] These two authors contribute equally to this research.

https://doi.org/10.1016/j.cag.2019.05.014
0097-8493/

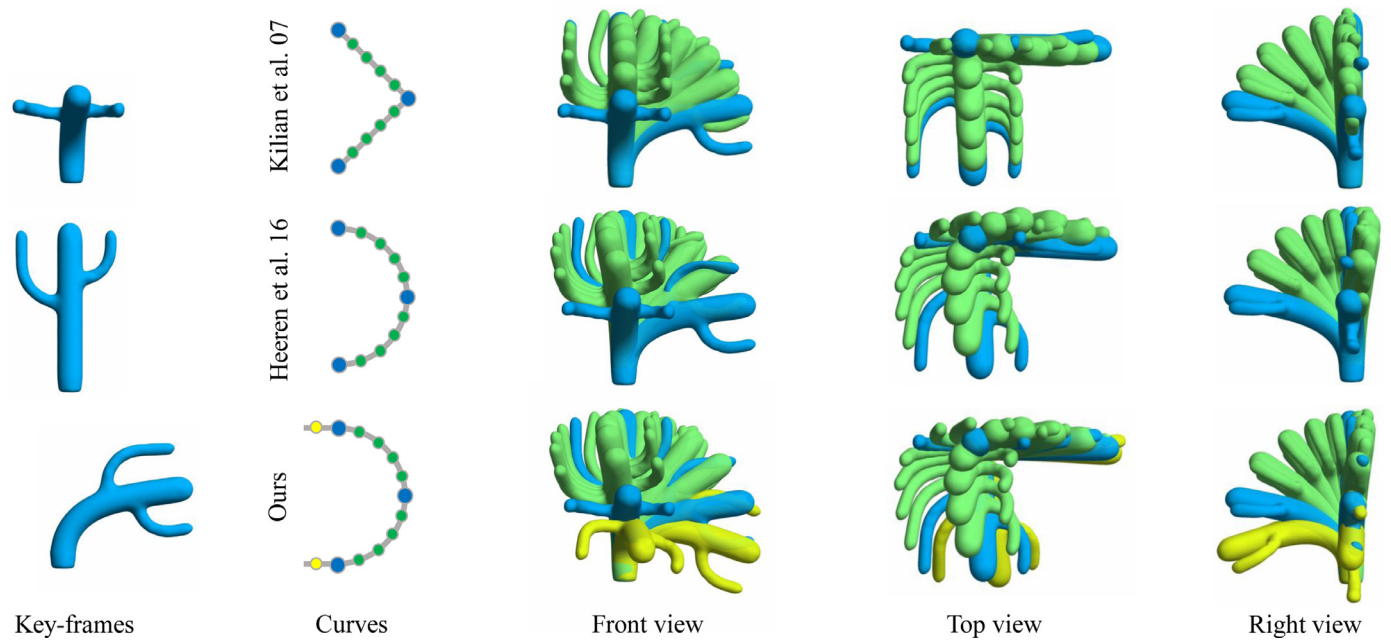

Fig. 1. Comparison of the shape sequence completion results via computing geodesics [1] in shape space, spline in shell space [2] and our cubic splines in linear rotation invariant space, respectively listed in the top, middle, and bottom row. From left to right, it shows 3 different poses of a cactus shape, the fitted curves in shape space, the completion sequences viewed from front, top, and right. The input key-frames, interpolated and extrapolated shapes are separately colored in blue, green and yellow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sequence, which iteratively replace each shape with a weighted average shape of a local neighborhood by globally minimizing an energy to obtain smooth discrete geodesics in shape space. However, such post-processing is only suitable for refining traditional subdivision interpolation between successive key-frames, which still does not solve this problem fundamentally. Recently, Heeren et al. [2] extend Euclidean splines to the Riemannian manifold of discrete shells [9], in which the associated metric measures both bending and membrane distortion, and the problem is formulated as a global optimization that finds a spline in shell space. However, the global optimization for splines in shape space and the reconstruction of discrete shells are not efficient, especially when the shapes have many details and too many vertices, or when the desired interpolating sequence is very dense.

Strongly inspired by the idea of fitting splines in shape space, we also formulate the problem as a global optimization that minimizes the accumulated shape distortion. Different from that in [2], we don't solve the optimization directly. Instead, we first embed the shapes into a feature space via patch-based linear rotation-invariant (LRI) coordinates [10], which is robust to noise, efficient both in representation and reconstruction, and also equipped with capacity of describing large deformations. Then, the complex minimization in shape space can be converted into a few standard cubic spline fitting problems in feature domains, and each spline is a smooth curve consisting of piece-wise cubics. Next, we solve for the coefficients of each cubic polynomial by a very small linear system in each feature domain, whose dimension only depends on the number of key-frames and is independent of the number of desired interpolating shapes. Finally, we reconstruct the interpolated shapes in this sequence according to the interpolated coordinates by also solving least squares systems efficiently. To further improve the computational efficiency, we also devise a multi-resolution scheme for model completion. We first apply mesh simplification [11] on all key-frames, which decomposes the shape space into a low-frequency part (primary poses) and a high-frequency part (geometric details), then the interpolation is conducted on the simplified shapes, and the final interpolating shape sequence is obtained via deformation transfer [12], which transfers the geometric details onto the simplified shapes and results in a smooth and detailed high-quality shape sequence. In particular, the primary contributions of this paper can be summarized as follows:

- We propose an efficient interpolation method for 4D shape sequence completion from a few sparse samples along temporal axis via extending the idea of smooth curve fitting to 4D shape sequence completion, which can produce smoother and more plausible computer animations.
- We re-formulate the problem of shape sequence completion as a global optimization that minimizes the accumulated distortion along the temporal axis and simplify it as cubic spline fitting problems, which can be efficiently and analytically solved by small linear systems.
- We devise a hierarchical framework for interactive shape sequence completion, in which the shape space is decomposed into high-frequency and low-frequency domains, the interpolation is only operated on low-frequency domain and the high-frequency details are synthesized by deformation transfer techniques.
- The new scheme is used to conduct interpolation between very large deformations because of the good properties of patch-based LRI coordinates. As a result, we can not only interpolate in-between shapes but also extrapolate more plausible shapes, because of the advantages of cubic splines.

2. Related works

Closely relevant to the central theme, we now briefly review previous approaches and their related applications in two categories: shape interpolation and smooth shape sequence creation.

Shape interpolation. Given the source and target shapes represented with triangular meshes, the first step of shape interpolation is to establish one-to-one correspondence between them. So far, shape correspondence has been well studied in computer

graphics community. Various techniques have been proposed, such as performing common embedding [13], parameterizing over coherent base domains [14] and incrementally aligning the surface patches [12]. However, even with absolutely reliable correspondences, natural shape interpolation still remains to be challenging in general when the shape deformation is large. To combat the unwanted artifacts caused by linear interpolation of vertex positions, several approaches have been proposed to interpolate triangle-wise affine transformations instead. For example, Alexa et al. [15] propose the famous as-rigid-as-possible (ARAP) shape interpolation method that reconstructs the interpolated shape by separately blending the rotational and scaling components of the transformation matrix between a pair of triangles. But this method could exhibit serious artifacts for shapes with large deformations. Xu et al. [16] interpolate the shapes from gradient fields by blending those of the given shapes, which is named as Poisson shape interpolation. Similarly, Li et al. [17] interpolate the shape using relative velocity fields in an as-isometric-as possible(AIAP) manner, and later Zhang et al. [18] propose a fast AIAP shape interpolation method to improve the computational efficiency. Weber et al. [19] achieve a controllable conformal interpolation by blending the representations of two given shapes. Chen et al. [20] propose a bounded distortion interpolation which reduces the triangle distortion by blending the squared edge lengths of the given shapes. These methods are all using a conformal parameterization algorithm [21], and thus can not precisely satisfy the interpolation requirement, which are also only suitable for interpolating shapes with small deformations. Besides, Kilian et al. [1] propose to interpolate models along geodesics in specific shape space. Winkler et al. [3] propose to represent and interpolate meshes in terms of edge lengths and dihedral angles, which adopts the same local mesh properties for mesh interpolation as discrete shells [9]. Under this representation, Fröhlich and Botsch [4] propose an example-driven method by incorporating structural or anatomical knowledge learned from given example poses. To improve the computation efficiency, Fröhlich and Botsch perform the nonlinear optimization on reduced meshes and transfer the coarse meshes to the high-resolution meshes with a multi-resolution modeling method. Then von Tycowicz et al. [22] introduce a real-time non-linear interpolation by exploiting the structure of the shape interpolation problem and restricting the solution into a low-dimensional shape space and Xia et al. [23] use ghost mesh to accelerate the computation performance for interactive shape interpolation applications. Lipman et al. [24] represent the geometric details as discrete forms based on the local frames of vertices, which is called linear rotation-invariant coordinates. This kind of coordinates is very suitable for interpolation. And the famous MESHIK [25] linearly blends the deformation gradient between triangles to recover intermediate shapes. Then, Baran et al. [10] partition the shape into a few non-overlapped patches and achieve patch-based linear rotation-invariant coordinates for semantic deformation transfer. In this paper, we follow the representation of patch-based linear rotation-invariant coordinates because of its many favorable advantages.

Smooth shape sequence creation. Since our foci in this paper are to reconstruct the shape deformation sequence from a few key-frames, here we concentrate on the space-time constrained deformation-creating methods. In fact, such methods usually involve complex geodesics curve based geometric interpolation among shapes [1]. And this problem is first introduced into computer graphics by Witkin and Kass [26]. Since then, various techniques have been proposed to control different types of physical deformations, including ropes and strings [27], fluids [28], and elastic solids [29]. The underlying optimization problems are typically solved via gradient-decent-based approaches to deal with the complex calculation of objective functions. Kircher et al. [30] used rotation and translation invariant differential surface representation for surface editing in both space and time. Recently, Hildebrandt et al. [31] achieve this by conducting model reduction on the underlying variational problem and employing wiggly splines to solve a set of the decoupled one-dimensional space-time problems. Cashman et al. [32] propose a fully geometric method to separate editable signals by extracting redundancy embedded in the time, pose and shape phases, and derive a continuous representation from a discrete sequence of key frames. Brandt et al. [8] extend the idea of smoothing curves in Euclidean space and propose geometric flows, which iteratively replaces each shape with the average shape within a local neighborhood and produces smoother motions of shape sequence. However, these methods are used to smooth a sequence that is already dense and not suitable for creating smooth shape sequence from sparse samples. Recently, Gao et al. [33,34] develop a data-driven approach for shape morphing to obtain more realistic results by finding a minimal distance path within the local spaces based on an existing database. However, the learned prior knowledge is usually model-specific and can not be directly applied to other cases. Meanwhile, Heeren et al. [35] find a time-discrete geodesic path in shape space, similar to those in [1], by a computational model for geodesics in the space of thin shells, and extend it to Riemannian manifolds of discrete shells [9] via spline fitting in shell space [2]. But the large-scale non-linear global optimization leads to a very bad computational performance even though a multi-resolution acceleration method proposed in [4] has been used. And Huber et al. [36] introduce discrete Bézier curves in shell space and solve solely local problems in time instead of a fully coupled global optimization problem with a large set of nonlinear constraints. However, a Bézier curve usually do not pass through the sample points except the ends, and is actually not suitable for our target.

Our method is closely related to [2,10]. We follow the idea of fitting splines in shape space in [2], and construct the shape space by the patch-based LRI coordinates used in [10]. Unlike theirs, we solve the problem of generating a smooth shape sequence by a set of small tridiagonal linear systems and avoid the large-scale non-linear global optimization, together with the efficient LRI representation, our method can handle shapes with extremely large number of faces and achieves interactive performance.

3. Review on smooth curve fitting in euclidean space

Before discussing the problem of completing 4D shape sequence from a few sparse samples, we firstly review how to interpolate smooth curve from a few given points in Euclidean space. Given a finite set of known points, fitting a curve that goes through these points is a practical problem in various fields, including computer-aided geometric design, graphical rendering or even numerical calculation. The main problem here is to obtain and examine an integration rule based on integrating a curve passing through the given set of points, so that the curve will have a small amount of twisting or bending, which will be spread out whatever twisting and bending is necessary.

3.1. Mathematical formulation of smooth curve fitting

Given a set of $J \geq 2$ different time points $t_j \in [0, 1]$ and associated data points $p_j \in R, j = 1, \ldots, J$, we want to find a smooth curve f: $[0, 1] \rightarrow R$ that goes through these points. Here, we only consider the 1-D case, and it is easy to generalize this to arbitrary dimension. In general, the twisting or bending of a curve in certain location can be evaluated by the second order derivative, which describes the rate of changes of the tangent direction of the curve, and thus the problem of fitting a smooth curve can be mathematically formulated as minimizing the following integrated

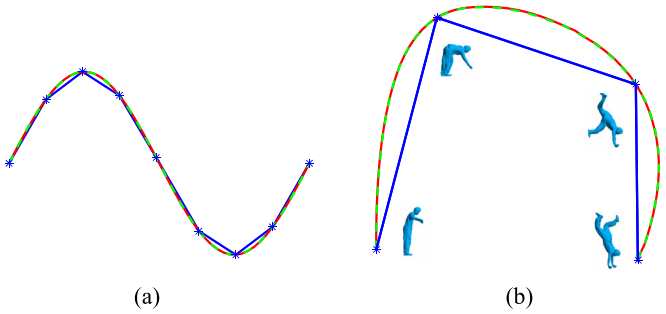

Fig. 2. The illustration of smooth curve approximation using global distortion optimization, cubic spline fitting and linear interpolation, colored in red, green and blue, respectively. (a) The results of 3 curves approximated from 9 points marked by blue stars in 1-D Euclidean space. (b) The plot of the leading two principal components of 3 shape sequences completed from 4 given shapes in LRI feature space. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

bending energy

$$E(f) = \int_0^1 \left[\frac{d^2}{dt^2} f(t) \right]^2 dt, \tag{1}$$

where f ranges over an appropriate set of fitting curves, which are continuous and also have continuous first and second order derivatives. And the square norm used here is to make sure the bending energy is non-negative. This minimization is to find a curve that has minimal bending or twisting energy in a global sense and the minimal second order derivatives everywhere guarantee the smoothness of this curve in a local sense. Besides, the optimal fitting curve f should also satisfy the interpolation constraints

$$f(t_j) = p_j,\ j = 1, \ldots,\ J. \tag{2}$$

That means the optimal curve should pass through all these given data points. When we have $f(0) = f_0$ and $f(1) = f_1$, the optimal f here is actually a straight line with f_0 and f_1 as endpoints, because the second order derivative of a straight line equals zero everywhere, and thus the bending energy E is obviously zero, which is also the minimum of course. The curve colored in red in Fig. 2 (a) shows an example of smooth curve fitting by minimizing the integrated bending energy which passes through 9 points in Euclidean space, and this curve is smooth everywhere comparing to linear interpolation which has obvious sharp corners at key-points.

3.2. Analytical solution to smooth curve fitting

For general cases when $J > 2$, the optimization in Eq. (1) can be solved via its Euler-Lagrange equation as done in [2]. However, if we want to produce a curve represented by K points, its discretized Euler-Lagrange equation is a $K \times K$ linear system, when we want a very dense points set, namely K is extremely large which is a common requirement for computer animation generation, solving such a linear system is not efficient even for the 1-D case, and of course not suitable for interactive applications. In this paper, we adopt a different but much more efficient way to solve the problem in Eq. (1) based on an observation that, the minimization of integrated bending energy along the curve formulated as in Eq. (1) will result in a piece-wise cubic spline, and the proof can be found in any Mathematics textbook related to spline interpolation or one can directly refer to the accompany Appendices of this paper. Thus, the optimal curve f in Eq. (1) is actually a piece-wise cubic spline consisting of $n = J - 1$ cubics, which can be represented with piece-wise cubic polynomials, and each cubic contains four coefficients. Therefore we need $4n$ linear equations to determine the $4n$ coefficients of f, and leads to a simple $J \times J$ tridiagonal system which can be solved very easily to give the coefficients of the polynomials [37], usually $J << K$. The dashed curve colored in green in Fig. 2 (a) shows an example of cubic spline fitting, and the fitted smooth curve is actually the same with that fitted by solving global blending minimization colored in red. As we know, boundary conditions at endpoints for cubic splines are not unique, usual conditions such as natural, periodic or Hermite can be applied for different purposes, as shown in Fig. 3. Note that if t_0 and t_n are interior to the end points of the interval on which f is defined, then to minimize Eq. (1), f would be the same as that on $[t_0, t_n]$. Also, as cubic splines are expressed with piece-wise cubic polynomials for point that lies in $[t_0, t_n]$, we can compute the derivatives at t_0 and t_n using the analytic expression of f, which makes it possible to do extrapolation outside the range of $[t_0, t_n]$. One intuitive and simple way is linear extrapolation, wherein the value of f at $t < t_0$ or $t > t_n$ can be computed based on the first order derivative of f at t_0 or t_n as $f(t) = f'(t_0)(t - t_0) + f(t_0)$ for $t < t_0$ or $f(t) = f'(t_n)(t - t_n) + f(t_n)$ for $t > t_n$, shown as the bottom case in Fig. 1, and this linear extrapolation is used by default in all our experiments. It should be noted that, the linearity approximation is only reasonable when the extrapolated point is near the given defined interval.

4. Cubic splines in linear rotation invariant space

As we know minimizing the blending or twisting of a curve (which can be described as the second order derivative and is also related to curvature) will result in a smooth cubic spline with natural boundary. In this section, we will transfer this simple but effective idea to spline fitting in shape space, following the formulation in [2], namely producing a smooth 4D shape sequence from sparse samples. The difference here is that we use a more complicated representation of "point" in shape space than that in Euclidean space.

4.1. Splines in shape space

In Euclidean space, a smooth curve is fitted by minimizing an accumulated bending or twisting energy which is quantized as integration of the squared second order derivative along this curve.

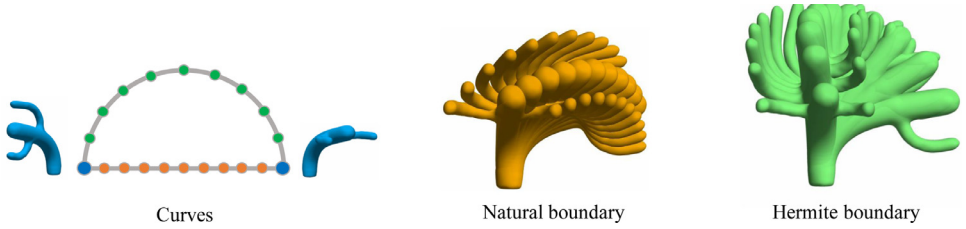

Fig. 3. The results of our method for interpolating two cactus shapes colored in blue under different boundary conditions. Two fitted curves in shape space are shown on the left side, and the round circles indicate the interpolated shapes under natural (in orange) and Hermite (in green) boundary conditions shown on the right side. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

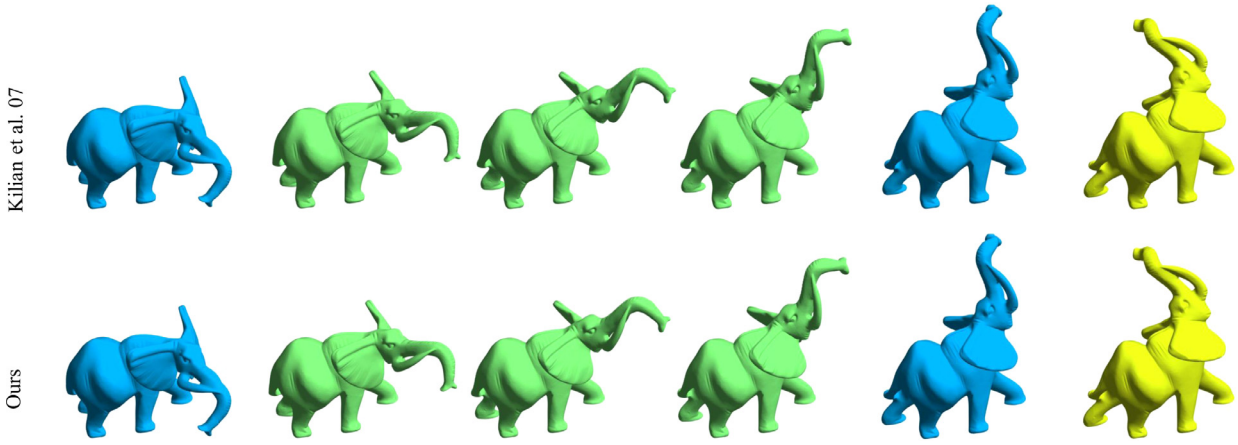

Fig. 4. The interpolation and extrapolation results of our method comparing to the geodesics in shape space [1]. When only given two shapes, our method degenerates to the geodesics. Shapes colored in blue, green, and yellow indicate given, interpolated and extrapolated shapes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

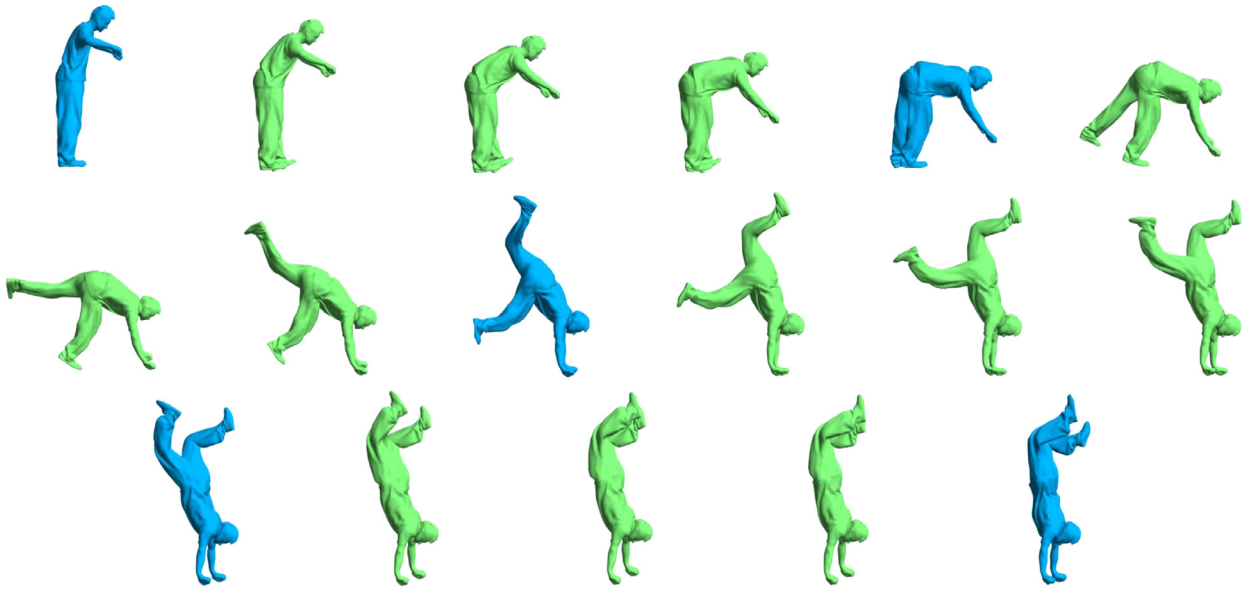

Fig. 5. The shape sequence completed from 5 human shapes. Shapes colored in blue, and green indicate given and interpolated shapes. It should be noted that, the rotation of the first and last poses is nearly 180 degrees, that means triangles on the mesh may flip over from the first pose to the last, and the LRI coordinates still work well under such large deformations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Similarly, in the case of fitting 4D shape sequence in shape space, the smooth sequence can also be intuitively obtained by minimizing distortions along the shape sequence. Let $\mathcal{S}$ be a shape space, and s is a smooth curve $s : [0, 1] \to \mathcal{S}$, the total distortion of curve s can be defined as

$$\mathcal{E}[s] = \int_0^1 \ddot{s}^2(t)dt, \tag{3}$$

where $\ddot{s}(t)$ is the second order derivative of curve s in shape space similar to that in Eq. (1). The result of minimizing $\mathcal{E}$ among all curves s with $s(0) = s_0$ and $s(1) = s_1$ with natural boundary condition is known as a geodesic in shape space [1], as shown in Fig. 4, similar to the straight line in Euclidean case and this is actually a generalization of Eq. (1) in a more general case. We consider J high-dimensional points $s_j \in \mathcal{S}, j = 1, \ldots, J$, and we want to obtain a smooth curve in $\mathcal{S}$, the same as that in Euclidean space, the optimal curve should also satisfies the interpolation constraints

$$s(t_j) = s_j, j = 1, \ldots, J. \tag{4}$$

The minimization of this distortion under the interpolation constraints defined in Eq. (4) is a spline in shape space. In addition we can also define different boundary conditions as those in Euclidean space, examples of interpolation under Natural and Hermite boundary conditions are shown in Fig. 3.

4.2. Linear rotation invariant representation

A computer animation is usually characterized as a mesh sequence with varying vertex coordinates and fixed connectivity, thus in this paper, we assume the connectivities of a set of given meshes are the same and we only need to calculate vertex coordinates for each shape along the sequence. In order to apply the spline fitting in shape space to discrete settings in the case of shape interpolation, we should firstly represent the 3D shapes appropriately in shape space. In this paper, we adapt the patch-based Linear Rotation Invariant (LRI) [10] representation to embed the shape into a feature space, because it has many favorable advantages, such as being capable of describing large deformations shown in Fig. 5, very suitable for not only interpolation but also extrapolation shown in Fig. 4, suitable for PCA analysis

shown in Fig. 2, and efficient both for representation and reconstruction shown in Table 2.

In patch-based LRI representation, one shape is referred as the reference shape, and given another deformed shape, the deformation can be represented as follows: given a face $\mathbf{f}$ with vertices $\mathbf{v}_{\mathbf{f}_i}(i = 1, 2, 3)$ and the unit normal direction $\mathbf{n}_\mathbf{f}$, the deformation gradient $\mathbf{D}_\mathbf{f}$ for face $\mathbf{f}$ can be obtained as $\mathbf{D}_\mathbf{f} = [\mathbf{v}_{\mathbf{f}_2} - \mathbf{v}_{\mathbf{f}_1}, \mathbf{v}_{\mathbf{f}_3} - \mathbf{v}_{\mathbf{f}_1}, \mathbf{n}_\mathbf{f}][\tilde{\mathbf{v}}_{\mathbf{f}_2} - \tilde{\mathbf{v}}_{\mathbf{f}_1}, \tilde{\mathbf{v}}_{\mathbf{f}_3} - \tilde{\mathbf{v}}_{\mathbf{f}_1}, \tilde{\mathbf{n}}_\mathbf{f}]^{-1}$, where $\mathbf{v}$ are the vertex coordinates and $\mathbf{n}$ are the normals of the deformed shape, and $\tilde{\mathbf{v}}/\tilde{\mathbf{n}}$ are those of the reference shape. And then $\mathbf{D}_\mathbf{f}$ is further decomposed into a rotation component $\mathbf{R}_f$ and scaling/shear component $\mathbf{S}_\mathbf{f}$ by polar decomposition: $\mathbf{D}_\mathbf{f} = \mathbf{R}_\mathbf{f}\mathbf{S}_\mathbf{f}$. As those in [10], we also partition shape into a collection of non-overlapping patches using the method in [38] to improve the computation efficiency and the robustness under noise. Then the patch containing face $\mathbf{f}$ defined as $\mathbf{p}(\mathbf{f})$ and the average rotations of the faces in patch i are defined as $\mathbf{G}_i$. The patch-based LRI coordinates are vectors, including five components: the scaling/shear matrix $\mathbf{S}_\mathbf{f}$ of each face, the connection map between each pair of adjacent patches, the relative rotation of each face w.r.t. its belonging patch, the mean vertex position of all the vertices $\bar{\mathbf{v}}$ and the mean rotation of all the faces $\bar{\mathbf{R}}$. The coordinates $\mathbf{x}$ are then represented as a vector

$$\mathbf{x} = [\mathbf{S}_\mathbf{f}, \log(\mathbf{G}_i^{-1}\mathbf{G}_j), \log(\mathbf{G}_{\mathbf{p}(\mathbf{f})}^{-1}\mathbf{R}_\mathbf{f}), \bar{\mathbf{v}}, \bar{\mathbf{R}}], \tag{5}$$

for any face $\mathbf{f}$ and any pair of adjacent patches i and j, where log is the matrix logarithm operation to allow rotation to be better combined, $\mathbf{G}_i^{-1}\mathbf{G}_j$ is the connection map between adjacent patches i and j. These components are put together as a feature vector. Given such a coordinate, we can reconstruct the shape by solving two linear systems, as those in [10], the first is to reconstruct the rigid rotation of each face and the second to recover the coordinate of shape vertices, which can be factored once for a given mesh using a sparse Cholesky solver and each new pose requires only a back-substitution. Thus it is very suitable for shape sequence completion, where we need to reconstruct many poses using the same reference.

After embedding the shape into a feature space X using patch-based LRI coordinates, wherein a shape $\mathcal{M}$ is represented as a feature vector like that in Eq. (5), the smooth curve s: $[0, 1] \rightarrow S$ we are seeking in shape space now becomes the curve x: $[0, 1] \rightarrow X$ in LRI feature space and the total distortion in Eq. (3) can be rewrite as

$$\mathcal{E}[x] = \int_0^1 ||\ddot{x}||^2 dt = \int_0^1 \sum_{i=0}^{M} ||\ddot{x}_i||^2 dt = \sum_{i=0}^{M} \int_0^1 ||\ddot{x}_i||^2 dt, \tag{6}$$

where x_i is the i-th entry of x and M is the dimension of x. Obviously, the minimizer of Eq. (6) equals the concatenation of M minimizers of $\int_0^1 ||\ddot{x}_i||^2 dt$, and $\int_0^1 ||\ddot{x}_i||^2 dt$ is exactly the same as that of the 1-D case in Eq. (1), whose minimizer has been proved to be a cubic spline. Therefore, the problem of completing a smooth shape sequence is now converted into fitting M independent 1-D cubic splines in feature space, namely solving M small tridiagonal linear systems as those in Section 3, and then reconstructing the vertex coordinates from the interpolated features. Fig. 2 (b) shows 3 shape sequences completed from 4 shapes by solving the global distortion minimization, cubic spline fitting, and linear interpolation in LRI feature space. For better visualization, we apply PCA on the patch-based LRI coordinates and plot the leading two principal components. It is obvious that the splines in Eq. (3) solved by global minimization and our cubic spline fitting in LRI feature space are the same to each other and smooth everywhere.

It should be noted that, the patch-based LRI representation is not the only choice in our 4D shape sequence completion framework, other shape representation methods with comparable properties and advantages, which represent the shape with a long feature vector that describes local or global properties and can be used to reconstruct a shape according to itself, such as the rotation-strain coordinates in [39] and the rotation-invariant mesh difference (RIMD) representation in [40], can also be easily integrated into our framework. We use this representation here only because of its favorable advantages discussed before.

4.3. Multi-resolution framework

We have converted the problem of completing a smooth shape sequence into computing many 1-D cubic splines, which can be efficiently solved via small tridiagonal linear systems instead of solving the global optimization. However, when the given shapes are with a lot of details and have a very large number of vertices or faces, the patch-based LRI coordinates for one shape are represented as a very long feature vector. Thus, we should compute so many cubic splines to reconstruct the shape sequence, which may not be efficient enough for interactive or real-time applications. Fortunately, we observed that, the details, like the textures, on a 3D shape are usually fixed in the deformed sequence, so we can deal with the completion in a multi-resolution sense.

Similar to the way used in [23], we firstly use an edge-collapse [11] scheme to coarsen the given meshes with around 2K faces and 1K vertices (actual numbers depends on mesh complexity), and we call it as ghost mesh, and each mesh is simplified with exactly the same rule to keep the topology consistent across the coarse meshes. It should be noted that, the method in [23] adopts vibration modes to reduce the solution space for interactive shape interpolation, which is only suitable for interpolating between shapes with small deformations and can not be directly applied to our framework because we want to deal with shapes with very large deformations. In this paper, we only use their mesh coarsening method. The ghost meshes are actually the low-frequency part of the shapes, which describes the primary variation of poses. Then, we embed these ghost meshes into shape space via patch-based LRI coordinates, and here the dimension of these coordinates is much smaller than that of the original meshes. With these coordinates, we can fit 1-D cubic splines in each dimension of the coordinates with analytical expression as cubic polynomials, and then we can reconstruct a smooth shape sequence using the interpolated LRI coordinates at arbitrary time point on interval [0, 1]. Of course, the shape in this sequence is coarse and without details. Finally, we adopt the deformation transfer technique in [12] to transfer the details of the original mesh onto the interpolated coarse mesh, and obtain a smooth shape sequence with high-quality details. The deformation transfer technique needs to manually assign correspondences between the source and target shape, fortunately edge-collapse [11] schemes simplify a mesh by collapsing a few vertices into one, so each vertex on the ghost mesh has a few corresponding vertices on the original mesh. For a vertex on the ghost mesh, we choose the nearest one of these vertices as its corresponding vertex on the original mesh, and the results are shown in Fig. 6, together with the ghost meshes we used.

5. Experimental results and evaluations

To verify the effectiveness and versatility, we have designed different kinds of experiments, including the smooth interpolation and extrapolation from sparse samples, robust interpolation under large deformations and highly-detailed shapes with extremely large number of vertices or faces, and the time statistics of all experiments are detailed in Table 1, the time comparison to [2] is given in Table 2.

Implementation details. We have implemented our interactive 4D shape sequence framework using C/C++ on a PC with Intel Core

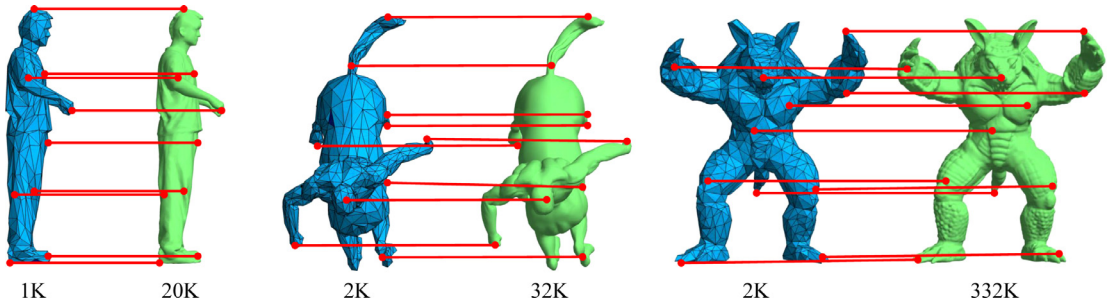

Fig. 6. The automatically-built correspondences between key-frames and their ghost meshes. The key-frames are colored in green and smoothly shaded, the ghost meshes are colored in blue and shaded in faceted way, and the correspondences are shown with red lines. The numbers of faces for each shape are listed in the bottom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Performance statistics (in seconds).

Models	Cactus (10K)	Handstand (20K)	Human (25K)	Centaur (32K)	Elephant (80K)	Armadillo (233K)	Armadillo (1.3M)
t_{ir}	0.013	0.016	0.021	0.025	0.024	0.027	0.039
t_{ts}	0.014	0.022	0.029	0.038	0.053	0.46	2.862
t_o	0.25	0.60	0.65	0.75	1.29	6.5	37.2

The runtime statistics of our framework to interpolate one shape. From top to bottom, t_{ir}: time cost for interpolation and reconstruction on ghost mesh, t_{ts}: time cost for deformation transfer, t_o: time cost for interpolate one shape without using ghost.

Table 2
Performance comparison to [2] (in seconds).

Model (K, γ)	Method in [2]			Ours		
	t_{ir}	t_{ts}	t_o	t_{ir}	t_{ts}	t_o
Cactus (20,.05)	1.7	1.7	83	0.16	1.0	4.35
Cactus (170,.05)	22	14	762	0.31	3.2	23.9
Horse (40,.10)	22	5	238	0.35	2.2	11.4
Armadillo (50,.006)	80	200	/	1.34	47.2	258.6

Total runtime of shape sequence completion for generating a smooth curve of K shapes with γ indicating the fraction of remaining vertices on the ghost mesh. From left to right, t_{ir}: time cost for interpolation and reconstruction on ghost mesh, t_{ts}: time cost for deformation transfer, t_o: time cost for interpolate one shape without using ghost, each one is stated both using the method in [2] and ours.

i7-3770 CPU @ 3.40 GHz. In all of our experiments, the parameters involved in our method are kept fixed unless otherwise stated. All shapes are approximately normalized into a bounding box with unit size and the simplified meshes used in all experiments are with about 1K vertices and 2K faces. The patch-based LRI representation and reconstruction, as well as deformation transfer method are implemented with the C/C++ codes shared by the authors with default settings, then we implemented the edge-collapse algorithm according to [11], and used Intel Math Kernel Library to solve the tridiagonal linear systems for cubic spline fitting as described in Section 3. The number of sample points used to build sparse correspondence is set to be 50 similar to those in [12].

Choice of ghost resolutions. Fig. 7 shows an example of shape sequence completion from 3 poses of a centaur shape, this centaur is triangulated with about 32K faces, and we conduct a group of 10 experiments with ghost mesh of 0.5K, 1K, 2K, 3K, 5K, 10K, 12K, 15K, 24K and 32K faces, and generate a shape sequence of 13 shapes each. We show the 11th shape in each sequence, indicated as a blue point on the fitted curve, and the shapes with ghost meshes of different resolutions are color coded by the difference between the 11th shape in the sequence without using ghost mesh and themselves. The difference is directly quantized as the L2-norm of vertex positions. We can see that, using a ghost mesh with only 500 faces, the reconstruction error is very large, because the ghost mesh with too few faces can not cover the variation of poses. For example, the deformation of left foreleg and two arms of the centaur are not correctly described, which results in large error. However, when the ghost mesh has more than 2K faces, the interpolations between the cases with and without using ghost mesh are almost the same. Besides, we also plot the Karni-Gotsman (KG) error [41,42] of each sequence reconstructed using ghost meshes of different resolutions for shapes of centaur, human, armadillo and elephant in Fig. 8 (a). And the average computation time to reconstruct one shape for the centaur using ghost meshes of different resolutions is shown in Fig. 8 (b). KG-error is a distortion metric that measures the difference between two shape sequences, which is defined as

$$e = 100\frac{||\mathbf{A} - \tilde{\mathbf{A}}||}{||\mathbf{A} - \mathbf{E}(\mathbf{A})||}, \tag{7}$$

where $\mathbf{A}$ is a matrix containing the sequence without using ghost, $\tilde{\mathbf{A}}$ is the one using ghost, and $\mathbf{E}(\mathbf{A})$ is an average matrix of $\mathbf{A}$. We can see that, for the centaur shape, when the ghost mesh has more than 2K faces, the KG-error is under 5% and has no obvious decrease when the number of faces on ghost mesh increases. However, the increase of ghost faces will of course affect the performance for generating shapes, as shown in Fig. 8(b), thus we can find a balance between the interpolation quality and the computation efficiency when using a ghost with 2K faces. For other shapes used in this paper, their geometry is simpler than the centaur, 2K faces of course are enough to cover the variation of poses and using ghost meshes for simpler shape will introduce smaller errors as shown in Fig. 8 (b).

Smooth shape sequence completion. Fig. 1 shows an example of smooth shape sequence completion from 3 poses of a cactus shape shown in the left most column, using the geodesics in shape space [1], the splines in shell space [2], and our methods. The first and third poses are created by bending the second one in two orthogonal directions. The method in [1] shown in top row computes geodesic path in shape space, which is actually a piece-wise linear curve as shown in the second column, and there is an obvious sharp corner around the middle of the sequence shown in the right three columns. As for the result of [2] shown in the middle row, the sequence is much smoother, because they use the idea of fitting spline in shell space. However, the large-scale global optimization make it have a very bad computation performance. For example, they reconstruct a cactus sequence

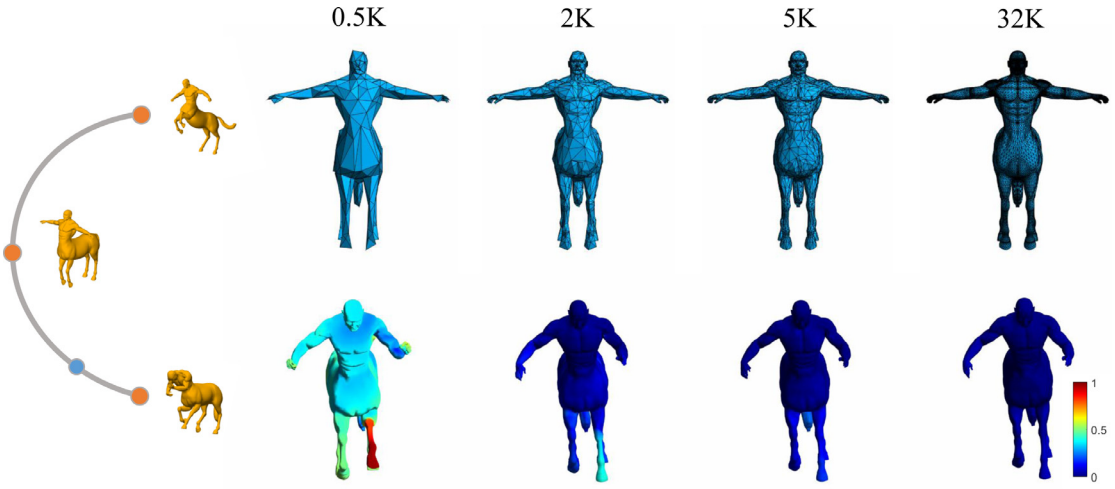

Fig. 7. An example of shape sequence completion from 3 poses of a centaur shape, colored in orange on the left side, using different resolutions of ghost mesh. The top row shows the ghost mesh with different resolutions and the bottom row shows the interpolation error with respect to the one without using ghost. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

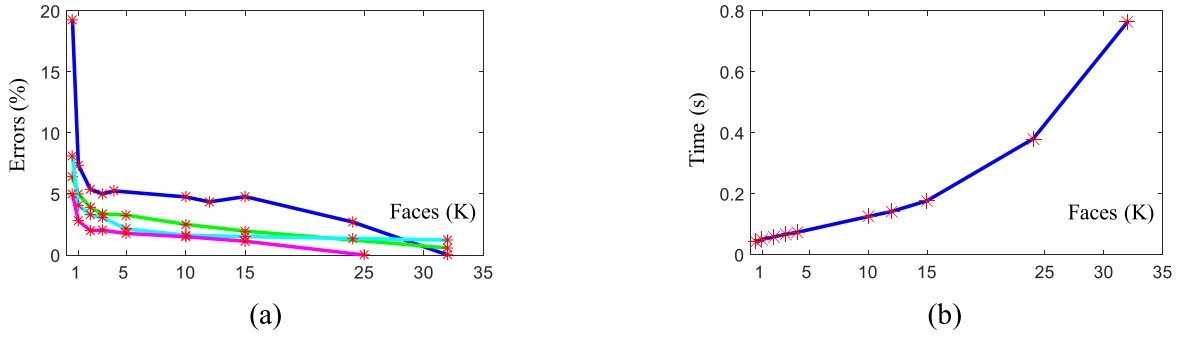

Fig. 8. The quantitative evaluations of shape sequence completion using ghost mesh with different resolutions. (a) The KG-errors of sequences completed using different ghost resolutions for the shape of centaur, human, armadillo and elephant, colored in blue, pink, green and cyan respectively. (b) The average computation time (in second) to interpolate one shape of centaur using different ghost resolutions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with 170 shapes using more than 700 seconds, in sharp contrast, we only need 24 seconds even without using ghost mesh. Because we solve the spline fitting problem in a much more efficient way as described in Sections 3 and 4, and obtain nearly same results as shown in the bottom row. Besides, due to the linearity of the cubic spline beyond the defined time interval, we can intuitively do extrapolations, shown in yellow colored shapes in the bottom row of Figs. 1–4, which [2] can not achieve in contrast. Fig. 9 shows an example of smooth shape sequence from 10 poses of a human shape. This figure is to demonstrate the smoothness of sequence created by our method comparing to that in [1] and [2]. The smoothness is evaluated by the second order derivative of the energy in Eq. (3). To remove the difference of using different shape representation, we normalize these three kinds of energy by the average deformation energy among the 10 given shapes. We can see that, the second order derivative of the sequence created by [1] is very large around the key-frames, which means the transition here is not smooth similar to those in Fig. 1. In contrast, the second order derivatives of the sequence created by [2] and our method are both small, which means our sequences are much smoother, because our objective functions both require to minimize the second order derivatives globally.

Robustness and versatility. Fig. 3 shows an example of shape sequence completion from 2 poses of the cactus shape used in Fig. 1 under different boundary conditions. As described in Section 3, the choice of boundary conditions to determine a cubic spline is not unique, and other boundary conditions can also be used instead. Here, we show the interpolation results of using natural and Hermite conditions. The natural condition sets the second order derivatives of the curve to zero at end points, and the Hermite condition is to manually assign first order derivatives to end points. The first order derivatives are computed by the difference between the first and last two shapes in the smooth sequence of the cactus in Fig. 1. Thus, the sequence completed here is almost the same as those in Fig. 1. Fig. 4 shows an example of shape sequence completion from two poses of an elephant shape of our method comparing to [1]. As described in Section 4, when given 2 reference shapes, the minimizer of Eq. (3) is a geodesic in shape space, namely the same as those in [1], in this case, the cubic splines of our method degenerate to the geodesics under natural boundary condition. Fig. 5 shows an example of shape sequence completed from 5 human shapes, and the rotation of the first and last poses is nearly 180 degrees, that means triangles on the mesh may flip over during the deformation. However, benefited from the robustness of patch-based LRI representation, our method still works well under such large deformations and also produces a smooth and plausible shape sequence. Fig. 10 shows an example of shape sequence completion from 3 poses of an armadillo with different resolutions. The given shapes are triangulated with 332K and 1.3 million faces. The interpolation results of different resolu-

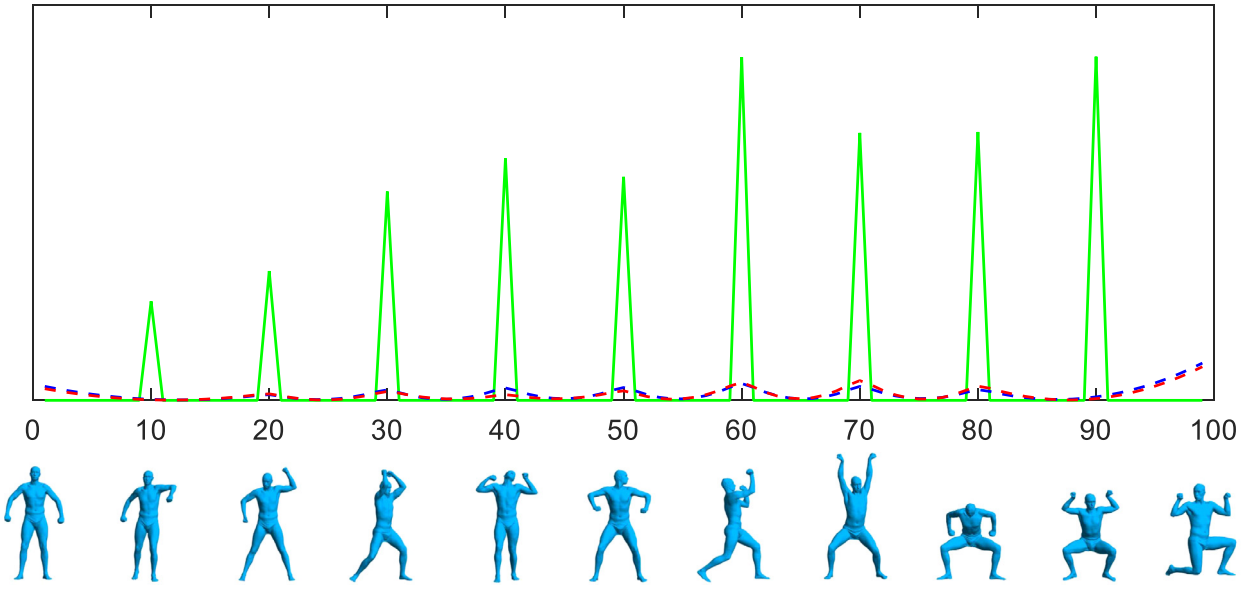

Fig. 9. The quantitative comparison of the smoothness of shape sequence completed using the geodesics in shape space [1] (green line), the splines in shell space [2] (blue dashed line), and our methods (red line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

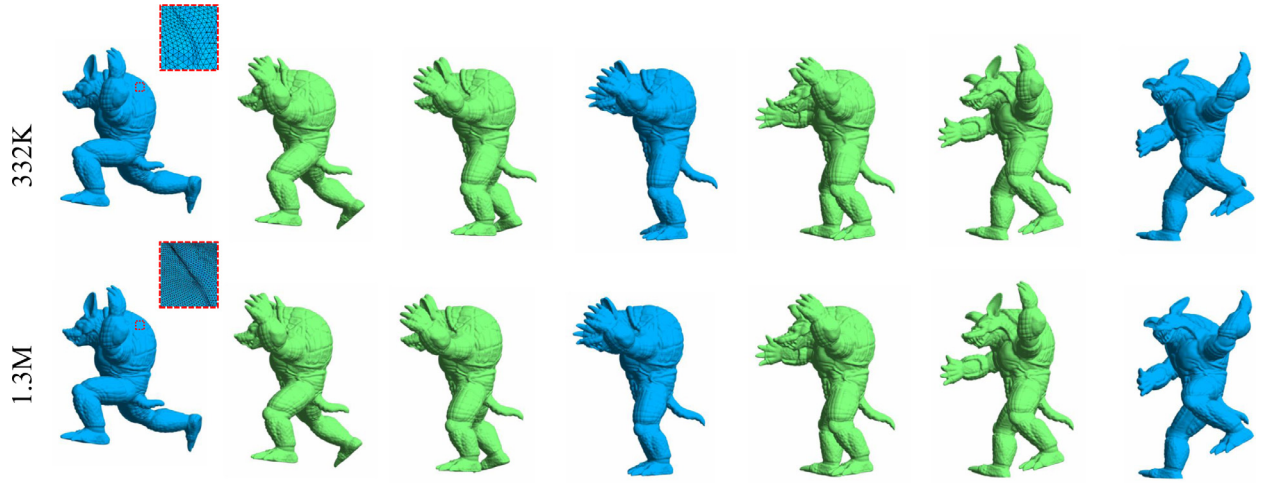

Fig. 10. An example of shape sequence completion from 3 poses of a armadillo shape with 332K and 1.3M faces.

tions are nearly the same. Besides, because of the simplicity of our framework, we can directly conduct interpolation on shapes with even more than 1 million faces, which will of course fail in [2] because of memory restriction, actually they already fail when the number of faces is 332K. And it only costs about 40 seconds to generate one shape with so many faces. With the ghost mesh used, we can cut down the computation time within about 3 seconds.

Comparison with [2]. The method in [2] and ours both aim to find a smooth curve which has minimal distortion as defined in Eq. (3), but we solve the same problem with totally different solutions. In general, there are three main differences between our method and that in [2]: (1) We use different types of shape representation. In [2], they embed shapes with discrete shells which represents shape using edge lengths, dihedral angles and areas of triangles. However, this representation is actually not suitable for extrapolation, for example, extrapolating on lengths of two edges may result in an edge of negative length which of course not exists. And the LRI representation we used is very suitable for both interpolation and extrapolation as demonstrated in Figs. 1 and 4. Besides, the reconstruction of discrete shells is usually not efficient enough for interactive or real-time applications, because it requires to solve a non-linear minimization using Gauss-Newton iterations, and when we solve for shapes in a sequence, the minimization will repeatedly restart as a new one. However, the reconstruction of LRI coordinates is very fast, wherein we only need to solve two linear least square systems, and Cholesky factorization can be applied once, then for other shapes in the sequence we only need to do simple back-substitutions which is pretty fast; (2) We solve the minimization of Eq. (1) in different ways. In [2], they solve the minimization via its discretized Euler-Lagrange equations which is a $K \times K$ linear systems, where K is the number of shapes we want to complete in the sequence. However, we solve the problem via cubic spline fitting, which is to solve a $J \times J$ tridiagonal linear system. In the case of 4D shape sequence completion, usually $J << K$. This difference on performance can be easily found in Table 2, where the time for interpolation and reconstruction of theirs increase a lot when we complete a sequence of 170 shapes comparing to that of 20 shapes, and ours is only slightly influenced because our solver is only dependent on the number of given shapes and independent on the number of shapes we want. (3) We use a

similar but different strategy to deal with the details. In [2], they adopt the method in [4], which consists of 5 steps, smoothing the shapes, simplifying them, reconstructing new simplified meshes using Gauss-Newton iterations, transfering the deformation of simplified meshes on to the smoothed shapes, and reconstructing details on the deformed smoothed shapes using Gauss-Newton iterations again. However, we simplify this strategy into 3 steps, simplifying the shapes, reconstructing new simplified meshes, and transfering the deformation of simplified meshes onto the original shapes, which decreases the runtime of deformation transfer, as shown in Table 2. The improvements on these three aspects make our method much more efficient than that in [2] and we achieve interactive performance as detailed in Table 1.

6. Conclusion and discussion

In this paper, we have detailed an efficient 4D shape sequence completion method. The central theme of this paper is to extend the idea of fitting smooth curve in Euclidean space to shape space. In particular, we proposed a different yet much more efficient way to solve this problem via piece-wise cubic spline fitting in linear rotation-invariant space, wherein we only need to solve a set of small tridiagonal linear systems instead of the large-scale global optimization. To further improve the computational efficiency, we adopted a multi-resolution strategy, which decomposes the shape into low-frequency and high-frequency domains, and the interpolation is only conducted on the low-frequency domain and the high-frequency details are recovered via deformation transfer techniques. Benefited from the simplicity of our fitting strategy and the efficient representation and reconstruction of patch-based LRI coordinates, our method can achieve interactive performance even when the shapes are having extremely large number of vertices or faces. Moreover, we have also designed diverse types of experiments over different kinds of shapes, which all confirmed the advantages and great potentials of our novel method.

Limitations and future works. Despite the attractive methodology properties of our method, it still has some limitations. The first one is co-shared by all spline interpolation methods as that in [2], it is the so-called overshooting problem, an unavoidable consequence of the smoothness requirement (see Fig. 5 in [2]). The second one is the control of splines in shape space. In this paper we only permitted the control at the end points, the same as cubic spline interpolation in Euclidean space. However, solely operating around the end points is usually not enough for artist to create exciting animations, more diverse types of controls and handles like those in traditional curve or surface editing in computer aided design could be of great value to the editing of shape sequence in the near future. Moreover, other types of shape representation or space reduction technique can be exploited to further improve the computational efficiency or interpolating quality. For example, in [22], the reduced shape space is constructed by basis derived from all given shapes, computing a smooth curve in this low-dimensional space for generating a smooth shape sequence maybe an interesting research in the future.

Declaration of interest

We declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is supported in part by the National Natural Science Foundation of China under Grant Nos. 61672077, 61532002, and 61872347, the Applied Basic Research Program of Qingdao under Grant No. 161013xx, the National Science Foundation of USA under Grant Nos. IIS-1715985, and IIS-1812606.

References

[1] Kilian M, Mitra NJ, Pottmann H. Geometric modeling in shape space. ACM Trans Gr 2007;26(3):64.
[2] Heeren B, Rumpf M, Schröder P, Wardetzky M, Wirth B. Splines in the space of shells. Comput Gr Forum 2016;35(5):111–20.
[3] Winkler T, Drieseberg J, Alexa M, Hormann K. Multi-scale geometry interpolation. Comput Gr Forum 2010;29(2):309–18.
[4] Fröhlich S, Botsch M. Example-driven deformations based on discrete shells. Comput Gr Forum 2011;30(8):2246–57.
[5] Baek S-Y, Lim J, Lee K. Isometric shape interpolation. Comput Gr 2015;46:257–63.
[6] Kochanek DH, Bartels RH. Interpolating splines with local tension, continuity, and bias control. ACM SIGGRAPH Comput Gr 1984;18(3):33–41.
[7] Lasseter J. Principles of traditional animation applied to 3d computer animation. ACM Siggraph Comput Gr 1987;21(4):35–44.
[8] Brandt C, von Tycowicz C, Hildebrandt K. Geometric flows of curves in shape space for processing motion of deformable objects. Comput Gr Forum 2016;35(2):295–305.
[9] Grinspun E, Hirani AN, Desbrun M, Schröder P. Discrete shells. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association; 2003. p. 62–7.
[10] Baran I, Vlasic D, Grinspun E, Popović J. Semantic deformation transfer. ACM Trans Gr 2009;28(3):36.
[11] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co.; 1997. p. 209–16.
[12] Sumner RW, Popović J. Deformation transfer for triangle meshes. ACM Trans Gr 2004;23(3):399–405.
[13] Lin C-H, Lee T-Y. Metamorphosis of 3d polyhedral models using progressive connectivity transformations. IEEE Trans Visual Comput Gr 2005;11(1):2–12.
[14] Schreiner J, Asirvatham A, Praun E, Hoppe H. Inter-surface mapping. ACM Trans Gr 2004;23(3):870–7.
[15] Alexa M, Cohen-Or D, Levin D. As-rigid-as-possible shape interpolation. In: Proceedings of the 27th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co.; 2000. p. 157–64.
[16] Xu D, Zhang H, Wang Q, Bao H. Poisson shape interpolation. Gr Models 2006;68(3):268–81.
[17] Li G, Yang L, Wu S, Tan W, Chen X, Xian C. Planar shape interpolation using relative velocity fields. Comput Gr 2013;37(5):364–75.
[18] Zhang Z, Li G, Lu H, Ouyang Y, Yin M, Xian C. Fast as-isometric-as-possible shape interpolation. Comput Gr 2015;46:244–56.
[19] Weber O, Gotsman C. Controllable conformal maps for shape deformation and interpolation. ACM Trans Gr 2010;29(4):78.
[20] Chen R, Weber O, Keren D, Ben-Chen M. Planar shape interpolation with bounded distortion. ACM Trans Gr 2013;32(4):108.
[21] Springborn B, Schröder P, Pinkall U. Conformal equivalence of triangle meshes. ACM Trans Gr 2008;27(3):77.
[22] Von-Tycowicz C, Schulz C, Seidel H-P, Hildebrandt K. Real-time nonlinear shape interpolation. ACM Trans Gr 2015;34(3):34.
[23] Xia Q, Li S, Qin H, Hao A. Modal space subdivision for physically-plausible 4d shape sequence completion from sparse samples. Pac Gr Short Pap Eurogr Assoc 2015:19–24.
[24] Lipman Y, Sorkine O, Levin D, Cohen-Or D. Linear rotation-invariant coordinates for meshes. ACM Trans Gr 2005;24(3):479–87.
[25] Sumner RW, Zwicker M, Gotsman C, Popović J. Mesh-based inverse kinematics. ACM Trans Gr 2005;24(3):488–95.
[26] Witkin A, Kass M. Spacetime constraints. ACM SIGGRAPH Comput Gr 1988;22(4):159–68.
[27] Barzel R. Faking dynamics of ropes and springs. IEEE Comput Gr Appl 1997;17(3):31–9.
[28] Treuille A, McNamara A, Popović Z, Stam J. Keyframe control of smoke simulations. ACM Trans Gr 2003;22(3):716–23.
[29] Barbič J, da Silva M, Popović J. Deformable object animation using reduced optimal control. ACM Trans Gr 2009;28(3):53.
[30] Kircher S, Garland M. Free-form motion processing. ACM Trans Gr 2008;27(2):12.
[31] Hildebrandt K, Schulz C, von Tycowicz C, Polthier K. Interactive spacetime control of deformable objects. ACM Trans Gr 2012;31(4):71.
[32] Cashman TJ, Hormann K. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. Comput Gr Forum 2012;31:735–44.
[33] Gao L, Lai Y-K, Huang Q-X, Hu S-M. A data-driven approach to realistic shape morphing. Comput Gr Forum 2013;32:449–57.
[34] Gao L, Chen S-Y, Lai Y-K, Xia S. Data-driven shape interpolation and morphing editing. Comput Gr Forum 2017;36(8):19–31.
[35] Heeren B, Rumpf M, Wardetzky M, Wirth B. Time-discrete geodesics in the space of shells. Comput Gr Forum 2012;31(5):1755–64.
[36] Huber P, Perl R, Rumpf M. Smooth interpolation of key frames in a Riemannian shell space. Comput Aided Geom Des 2017;52:313–28.

[37] Bartels RH, Beatty JC, Barsky BA. An introduction to splines for use in computer graphics and geometric modeling. Morgan Kaufmann; 1987.
[38] Wang RY, Pulli K, Popović J. Real-time enveloping with rotational regression. ACM Trans Gr 2007;26(3):73.
[39] Huang J, Tong Y, Zhou K, Bao H, Desbrun M. Interactive shape interpolation through controllable dynamic deformation. IEEE Trans Visual Comput Gr 2011;17(7):983–92.
[40] Gao L, Lai Y-K, Liang D, Chen S-Y, Xia S. Efficient and flexible deformation representation for data-driven surface modeling. ACM Trans Gr 2016;35(5):158.
[41] Karni Z, Gotsman C. Compression of soft-body animation sequences. Comput Gr 2004;28(1):25–34.
[42] Vasa L, Skala V. A perception correlated comparison method for dynamic meshes. IEEE Trans Visual Comput Gr 2011;17(2):220–30.