Volume 0 (1981), Number 0 pp. 1–11 COMPUTER GRAPHICS *forum*

Data-Driven Shape Interpolation and Morphing Editing

Lin Gao[†1], Shu-Yu Chen[1], Yu-Kun Lai[2 ‡] and Shihong Xia[1]

[1] Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, CAS
[2] School of Computer Science and Informatics, Cardiff University, UK

Abstract

Shape interpolation has many applications in computer graphics such as morphing for computer animation. In this paper we propose a novel data-driven mesh interpolation method. We adapt patch-based linear rotational invariant coordinates to effectively represent deformations of models in a shape collection, and utilize this information to guide the synthesis of interpolated shapes. Unlike previous data-driven approaches, we use a rotation/translation invariant representation which defines the plausible deformations in a global continuous space. By effectively exploiting the knowledge in the shape space, our method produces realistic interpolation results at interactive rates, outperforming state-of-the-art methods for challenging cases. We further propose a novel approach to interactive editing of shape morphing according to the shape distribution. The user can explore the morphing path and select example models intuitively and adjust the path with simple interactions to edit the morphing sequences. This provides a useful tool to allow users to generate desired morphing with little effort. We demonstrate the effectiveness of our approach using various examples.

Keywords: data-driven, shape interpolation, shape space, morphing editing

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—object representations

1. Introduction

Shape interpolation produces in-between shapes given two (or more) shapes. It has many applications in computer graphics to produce new models based on existing ones, and generate smooth animation sequences between a given pair of models. The latter is often referred to as morphing. Existing research for shape interpolation and morphing focuses on establishing correspondences between shapes. Once this is done, interpolation is performed by either simple linear interpolation of (geometric or other) coordinates or using more sophisticated (e.g. physically based) models. Even with well established correspondence and a suitable deformation model, realistic shape morphing is still challenging, especially when the models used for interpolation differ substantially. Using shape information alone, the essential characteristics of the objects cannot be captured. Even with physically-based interpolation models, it is still challenging to acquire a detailed and accurate physical model for the deforming objects and it could also be very time-consuming. As a result, significantly simplified interpolation models are typically used, leading to loss of realism. For challenging cases, existing methods produce interpolated shapes which may not be practically meaningful: e.g. they may contain self-intersections, or they do not follow physical laws or the object behaviors (e.g. an impossible pose for human motion interpolation).

With the proliferation of shape repositories, data-driven approaches have recently received a lot of attention. By exploiting the latent knowledge in shape repositories, such methods have demonstrated effectiveness in various geometry processing applications. Gao et al. [GLHH13] propose a data-driven mesh morphing approach which uses locally blended examples from the model repository as soft constraints to guide the morphing process. Compared with existing geometry based methods, this method produces more

† Emaill: gaolin@ict.ac.cn
‡ Emaill: Yukun.Lai@cs.cardiff.ac.uk

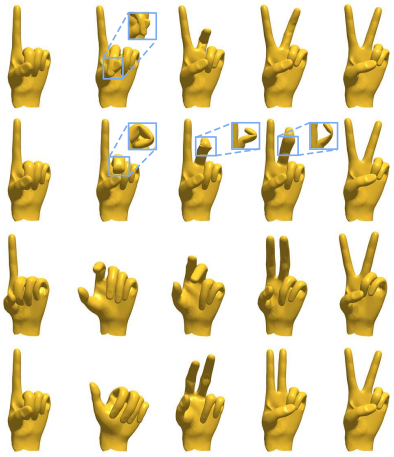


Figure 1: *Morphing results of the hand model. First row: non data-driven patch based linear rotation invariant (LRI) coordinates [BVGP09], second row: existing data-driven approach [GLHH13], third row: our data-driven morphing, fourth row: our data-driven morphing after editing.*

realistic results which tend to follow the characteristics and behavior of the deforming objects, when suitable example models exist in the repository. While being successful at addressing certain challenging situations, the method has its limitations: the use of explicit geometric coordinates is sensitive to translations and rotations. As a result, the method requires not only the guiding meshes to have suitable shapes, but also in suitable global *orientations* to be useful. Also, as example models are sparse, the method uses linear blending of geometric coordinates to produce a continuous representation. However, the representation can only be applied *locally* as linear blending does not work well for large-scale deformations. To address this, the example models are split into clusters and the plausible shapes are formulated as a *discrete* set of linear subspaces, which is quite complicated to optimize, more likely to converge to suboptimal local minima, and not able to fully utilize the information beyond the local clusters.

In this paper, we propose a novel data-driven approach to mesh interpolation and morphing, which addresses the fundamental limitations of [GLHH13]. We adapt the patch-based linear rotation invariant (LRI) coordinates [BVGP09] so that deformations of example models are represented in a rotation and translation invariant way. Example models in the repository are used to form a *globally* continuous space that represents plausible deformations. We formulate interpolation or morphing as finding a smooth, energy minimizing path in the plausible deformation space. As we will demonstrate in the paper, our method produces substantially improved results over state-of-the-art methods. We further develop techniques to allow morphing results to be edited intuitively by users. A challenging example is shown in Fig. 1 where the source (first column) and the target (last column) shapes differ substantially. Existing non data-driven method (first row) produces self-intersections. By exploiting the knowledge in a collection of hand models, existing data-driven morphing method [GLHH13] (second row) avoids self-intersections but still produces distorted fingers which are not realistic. Artifacts are highlighted in blue squares with additional views making them more visible. Our proposed approach better utilizes hidden knowledge of plausible deformations and produces realistic results without such artifacts (third row). The user can also easily edit the morphing sequence; in this case, the four fingers are now curled into a fist before extending the two fingers to form a 'V' shape (fourth row).

The main contributions of this paper are as follows:

- We propose a novel data-driven mesh interpolation/morphing method which produces more realistic results than existing methods, by effectively exploiting the knowledge in example shapes.
- Building on this, we further propose a novel approach to interactive morphing editing which produces realistic morphing following user constraints. This tool enables users to create desired morphing results with little effort.

The remaining sections are organized as follows. We review the relevant previous work in Sec. 2. Our data-driven interpolation technique is described in detail in Sec. 3, followed by our morphing editing technique in Sec. 4. We present various experimental results in Sec. 5 and finally draw conclusions in Sec. 6.

2. Related work

Due to its wide applicability, shape interpolation or morphing has been intensively researched. For models such as human bodies, one way of representing various poses is to use skeletons, and poses can be interpolated using skeleton interpolation, followed by skinning to recover the shape. The problem is easier due to the low dimensionality; however, such techniques are restricted to surfaces which can be well represented using skeletons. Our work considers general surface based interpolation/morphing. Most relevant work is reviewed in this section.

Given two (or more) models, the first step for interpolation is to establish one-to-one correspondence between shapes, often driven by a sparse set of correspondence points specified by artists [LDSS99, SP04, HAWG08]. Even with well established correspondences, realistic shape morphing is not trivial. Direct interpolation of the vertex positions may cause substantial artifacts, in particular when the shapes being interpolated have much difference (see Fig. 2(a)).

Geometry-based shape interpolation. Alexa et al. [ACOL00] generate morphing sequences that are locally as rigid as possible, by factoring a local affine transform matrix

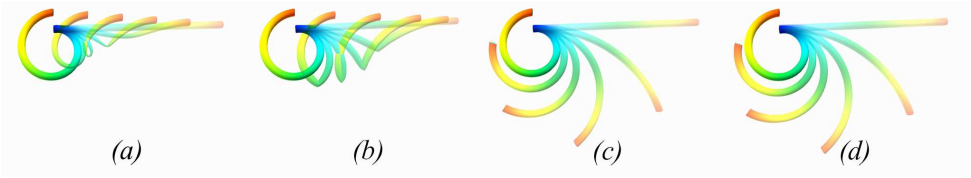


Figure 2: *Shape interpolation results. (a) direct interpolation of coordinates, (b) [XZWB06], (c) [FB11], (d) [BVGP09].*

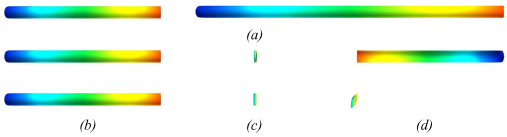


Figure 3: *Shape extrapolation results. (a) two shapes to be blended ($t = 0$ and $t = 1$), (b) results with $t = 0$, (c) results with $t = -1$, (d) results with $t = -2$. Middle row: results of [BVGP09], bottom row: results of [FB11].*

into a rotation matrix and a symmetric matrix and applying linear interpolation based on the decomposed matrices. However, the method requires *consistent* tetrahedron meshes as input which are generally difficult to obtain. Huang et al. [HAWG08] use a formulation that minimizes the overall displacement of vertices and is locally as rigid as possible. The algorithm takes surface meshes as input; however, the method is expensive: the number of unknowns scales linearly with the number of in-between frames and thus typically a very large linear system needs to be solved.

Geometry based morphing benefits from coordinates (feature spaces) that better preserve geometry. Instead of interpolating mesh vertex positions directly, Laplacian coordinates are used for shape interpolation [Ale03]. Other work also uses interpolation of mean Laplacian flow in the dual Laplacian domain [HLW07] and interpolation of the gradient fields followed by Poisson-based fusion [XZWB06]. Chu and Lee [CL09] interpolate the shape in the gradient space of near-rigid components, extracted using multiresolution mean shift clustering. The method works particularly well for pose interpolation.

Alternatively, shape interpolation can be achieved by interpolating edge lengths and dihedral angles of the trianlge meshes [WDAH10,FB11]. Edge lengths and dihedral angles are invariant to translation and rotation; however, reconstruction of the interpolated shape from them involves non-linear optimization. To address this, Winkler et al. [WDAH10] use multi-scale registration to reconstruct the vertex coordinates hierarchically. Although largely for shape deformation, Fröhlich and Botsch [FB11] reconstruct the interpolated shape by an iterative Gauss-Newton method. These approaches are too slow for interactive applications, and do not work well for extrapolation which may require edge lengths to be negative (see Fig. 3 for an example).

Rotation invariant coordinates are particularly suitable for interpolation/morphing as shapes with different orientations can be effectively blended. These methods use the idea of connection maps, representing frames in the local coordinates of their adjacent frames. Lipman et al. [LSLCO05] propose linear rotation invariant coordinates which can be used for interpolation by blending discrete form coefficients. This approach does not store connection maps explicitly so they need to be reconstructed first. After this, reconstruction of meshes requires solving two linear systems, one for reconstructing the absolute frames from connection maps, and the other for solving vertex positions. The method is known to be sensitive to noise [BVGP09]. Kircher and Garland [KG08] propose an alternative approach which stores connection maps explicitly but the connection maps used are not orthonormal, which may introduce global shear [BVGP09]. For the purpose of semantic deformation transfer, Baran et al. [BVGP09] propose rotation invariant coordinates which are patch-based and the coordinates record the connection maps between adjacent patches, as well as mesh faces with their belonging patches. The coordinates can be effectively blended and produce more robust results. The patch-based approach also makes the algorithm more efficient compared with [LSLCO05, KG08] as the number of patches can be significantly smaller than that of the mesh elements. Our method is based on this representation; however, we take a data-driven approach which substantially improves the results, thanks to the use of shape repositories.

Killian et al. [KMP07] consider shapes as points on Riemannian surfaces derived from typical transformations (e.g. isometric), and formulate plausible morphing as finding a shortest path in the shape space. Recently, von Tycowicz et al. [vTSSH15] propose an efficient non-linear shape interpolation technique which achieves real-time performance even for dense meshes. This method however does not represent shapes as rotation invariant coordinates so cannot be directly used in our data-driven approach.

Physically-based shape interpolation. Some research work improves interpolation realism by utilizing physical models. For the purpose of realistic simulation, Martin et al. [MTGG11] combine example-based interpolation with a strain field approach for improved physical correctness. An

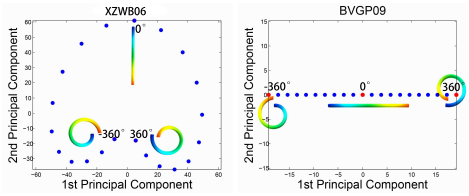


Figure 4: *Visualization of blended cylinder from* -360° *to* 360° *using dimensionality reduction to 2D. Left: result of poisson shape interpolation [XZWB06], right: result of [BVGP09].*

elastic deformation energy is defined in [HRWW12] for interpolating shapes and derived from the Hession of this energy, a Riemannian metric is introduced in [HRS*14]. Although these techniques tend to produce more realistic interpolation, such methods typically need several seconds for interpolating one shape so are not suitable for interactive applications. Moreover, the physical models are significantly simplified and cannot accurately reproduce the behavior of complex objects, e.g. human bodies.

Data-driven shape interpolation. Instead of using expensive physically based modeling, data-driven approaches resort to existing examples to improve the realism of interpolation. Data-driven shape deformation has been widely researched [SZGP05,FB11]. However, such techniques cannot be directly used for data-driven interpolation, where a large number of examples are typically needed to cover the plausible interpolation space. For mesh morphing and interpolation, Sloan et al. [SRC01] produce new shapes by example shape interpolation, controlled by an abstract space which needs to be manually constructed and specified offline. Recently, Hadar et al. [ACK16] propose a data-driven method for image morphing, which finds as-smooth-as-possible image sequences by calculating the shortest paths in the discrete neighborhood graph. By contrast, our work deals with 3D mesh morphing and optimizes paths continuously in the feature space. Gao et al. [GLHH13] propose a data-driven method that formulates morphing as a shortest path problem in the local linear subspaces derived from a given example model database. However, the vertex coordinates are directly used for linear blending which may cause distortions with large-scale deformation. Compared with [GLHH13], this work uses rotation-invariant coordinates which handles large-scale deformations well. Instead of optimizing the morphing sequence discretely defined over a set of local subspaces, our approach finds the optimized morphing path in the global continuous space, which not only improves the efficiency substantially, but also uses the knowledge in the repository better as it not only interpolates but also extrapolates between example models. Building on this, we further provide an intuitive interface for users to visualize the morphing path and edit the path interactively.

3. Data-Driven Shape Morphing

In this section, we propose a data-driven interpolation/morphing approach. Given the source shape M_s and the target shape M_t, our aim is to produce a realistic sequence of in-between shapes that smoothly transits from the source to the target. We assume that an example model database is provided. Models in the database are assumed to have the same topology. This is often satisfied for existing deformable object repositories and can be achieved by either fitting a dynamic template to a collection of objects, or consistent remeshing. If the models in the database are different from the models to be morphed, a deformation transfer approach is used [SP04,GLHH13], although these models should have similar behaviors so that the data-driven approach is meaningful. We first describe the shape representation used to encode the deformation of shapes, followed by our data-driven approach to morphing.

3.1. Shape representation

Given the dataset of the models with deformations we adapt the patch-based linear rotation invariant (LRI) representation [BVGP09] to represent the deformation between different models, as it has the following advantages: (i) It can handle large deformations well. As shown in Fig. 2, direct interpolation of coordinates (a) and poisson shape interpolation [XZWB06] (b) cannot blend shapes with large relative rotations well. (ii) It copes well with not only *interpolation* but also *extrapolation* (see a comparison with [FB11] in Fig. 3); the latter is particularly important to fully exploit the hidden knowledge in example shapes. (iii) It is suitable for linear dimensionality reduction (PCA analysis). More details are given later. A simple example is shown in Fig. 4 where the cylinder is rotated from -360° to 360°. The interpolated shapes are nicely distributed over the line after dimensionality reduction to 2D. The distribution obtained with the alternative poisson shape interpolation approach [XZWB06] does not exhibit such meaningful distribution. As we will show later, this visualization is essential to allow intuitive morphing editing. (iv) The representation is also efficient, which is critical for interactive applications.

For this purpose, a model is chosen as the base model, and given a deformed model, the deformation can be encoded explicitly as follows: Given a face f with vertices $\mathbf{v}_{f_i}$ $(i = 1,2,3)$ and the unit normal direction $\mathbf{n}_f$, the deformation gradient $\mathbf{D}_f$ for face f can be obtained as [BVGP09]

$$\mathbf{D}_f = [\mathbf{v}_{f_2} - \mathbf{v}_{f_1}, \mathbf{v}_{f_3} - \mathbf{v}_{f_1}, \mathbf{n}_f][\tilde{\mathbf{v}}_{f_2} - \tilde{\mathbf{v}}_{f_1}, \tilde{\mathbf{v}}_{f_3} - \tilde{\mathbf{v}}_{f_1}, \tilde{\mathbf{n}}_f]^{-1},$$

where $\mathbf{v}/\mathbf{n}$ are vertex positions and normal directions of the deformed surface, and $\tilde{\mathbf{v}}/\tilde{\mathbf{n}}$ are those of the base surface. $\mathbf{D}_f$ can be further decomposed into the rotation component $\mathbf{R}_f$ and scaling/shear component $\mathbf{S}_f$ by the polar decomposition: $\mathbf{D}_f = \mathbf{R}_f\mathbf{S}_f$.

In order to reduce the size of the linear systems during

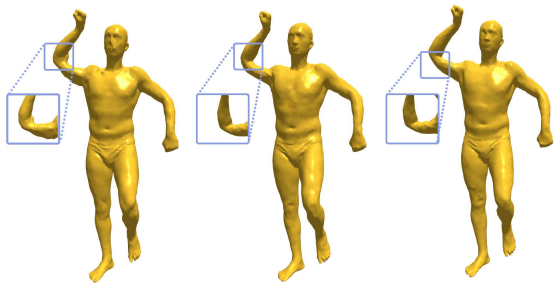


Figure 5: *Model reconstruction with different numbers of patches. From left to right: 1000, 2000 and 3000 patches.*

reconstruction and make the representation robust to noise, we follow [BVGP09] and partition the surface models into a collection of non-overlapping patches using the method in [WPP07]. Unlike [BVGP09] which uses a relatively small number of patches (e.g. dozens), we find using a larger number of patches more suitable in our settings because to cover the plausible deformation space of objects, our experiments use much more examples with significant variations in the training datasets, as we will demonstrate later. Using a small number of patches may fail to separate major deforming parts, leading to suboptimal results. Fig. 5 compares the reconstruction results using the SCAPE dataset with a varied number of patches. Using 1000 patches, the reconstructed model contains slight but visible distortions (left). Using 2000 patches (middle) produces very similar results as 3000 patches (right) with no visible difference, but is much more efficient, so 2000 patches provide a good balance between efficiency and quality, and are used in all our experiments. As individual patches are small enough, deformations will be more local and we do not observe potential gap artifacts with this setting.

We denote the patch that face f belongs to as $p(f)$. Let us further denote the average rotations of the faces in patch i as $\mathbf{G}_i$. The patch-based LRI coordinates in [BVGP09] are vectors including five components: the scaling/shear matrix $\mathbf{S}_f$ of each face, the connection map between every pair of adjacent patches, the relative rotation of each face w.r.t. its belonging patch, the mean vertex position of all the vertices and the mean rotation of all the faces. The last two components make this feature vector rotation and translation dependent. These components however are used for global rigid alignment, rather than reconstruction from the coordinates. To ensure rotation/translation invariance, we remove these two components from the coordinates and introduce an additional step to interpolate the rigid transforms from source and target pairs. The coordinates $\mathbf{x}$ include the following:

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

for any face 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. Empirically the values of these components are in a similar scale, so there is no need to introduce additional weights between these components.

Given this coordinate, we can reconstruct the shape by solving two linear systems, the first to reconstruct the rigid rotation of each face and the second to recover the coordinate of shape vertices.

3.2. Data-driven morphing

Given the example database, we first extract our shape representation as features for each deformed model. We now consider examples as samples representing reasonable morphing in the high dimensional coordinate space and formulate data-driven morphing as finding an energy minimizing curve in the coordinate space connecting the source shape M_s and the target shape M_t. Compared with previous data-driven work [GLHH13], our work has significant advantages: Our method can describe the shape distribution more effectively. As shown in Figs. 12, 13 and 14, our method effectively extract much more useful shape distribution information for morphing. As a result, there is no need for database upsampling (i.e. adding additional interpolated models to the database). As we will show later, our method runs much faster than [GLHH13] and can thus allow the user to edit the morphing path interactively. The use of rotation invariant coordinates [BVGP09] also helps parameterize the shape space to a 2D plane for morphing path editing (see Sec. 4).

Our problem is similar to finding geodesic paths on 3D point clouds; the major difference is that a much higher dimension is being considered, and hence the given plausible shapes provide a much sparser sampling. We first initialize the path using Dijkstra's algorithm on the K nearest neighbor graph, followed by path refinement using iterative quadratic energy minimization.

3.2.1. Initial Morphing Path

Discrete shape distribution approximation. We first provide a discrete approximation to the shape distribution representing plausible shape deformations. We use a simple K nearest neighbor (KNN) approach. For every model, we get the K nearest models in the coordinate space using the L_2 norm. In this paper, except for the *lion* dataset where $K = 3$ because the dataset only contains 10 example models, $K = 6$ is used for all the other data-driven morphing examples. We connect these nearest models to form the KNN graph $\hat{G}$. The obtained KNN graph $\hat{G}$ may have disjoint components. We form a singly connected graph $\tilde{G}$ by adding additional edges with shortest distances between connected components in the coordinate space to make the graph connected, similar to Prim's algorithm for minimum spanning trees.

Initial morphing path. Given the source model M_s and target model M_t, we can use Dijkstra's algorithm to get the

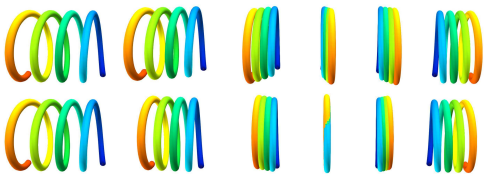


Figure 6: *Non data-driven morphing results. The leftmost and rightmost models are the source and target models, respectively. Top row: results of [HAWG08]; bottom row: linear interpolation using patch-based LRI [BVGP09].*

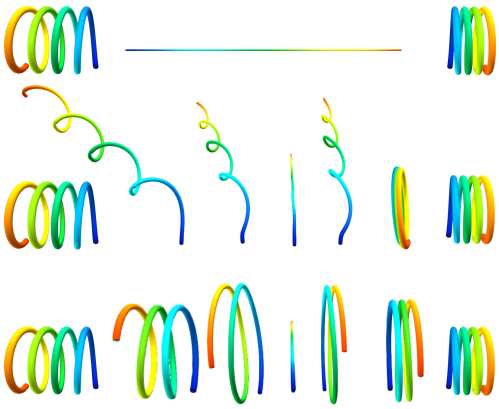


Figure 7: *Data-driven morphing. First row: source model, transition model and target model; second row: result of [GLHH13]; third row: our data-driven morphing result.*

path from the source model to target model. This provides an initial solution, which is often quite reasonable. However, the initial path is only C^0 smooth. The corresponding morphing sequence is thus not visually smooth and the morphing velocity is non-uniform. We can re-parameterize the morphing path based on the total length to get uniform velocity. In order to get a smooth morphing path along the shape distribution, we use the following path optimization.

3.2.2. Path Optimization

We follow two principles when optimizing the path. The first principle is that the interpolated path should be located on the shape distribution (or near the known samples from the shape distribution). Another principle is that the interpolated path should be short and smooth. We model these two principles using the following optimization, which is solved iteratively, starting from the initial morphing path.

Let us denote $\mathbf{x}_k$ to be the interpolated path after k^{th} iteration. $\mathbf{x}_{k,i}$ is the i^{th} sample position in $\mathbf{x}_k$ (200 sample points are used in our experiments). As we only have a sparse sampling of the shape distribution, we find the $\tilde{k}$ nearest models $\hat{\mathbf{x}}_{k,ij}$ from the examples in the coordinate space ($\tilde{k}=6$ in our experiments), and constrain the coordinates of $\mathbf{x}_{k,i}$ to be close to these models, with a weight decreasing with an increasing distance. To keep the path short and smooth, the total length and Laplacian operator are used. Overall the following energy is minimized:

$$E(\mathbf{x}_k) = \sum_i \sum_{\hat{\mathbf{x}}_{k,ij} \in N_{\mathbf{x}_{k,i}}} w_{k,ij} \|\mathbf{x}_{k,i} - \hat{\mathbf{x}}_{k,ij}\|^2 + \gamma \sum_i d_{k,i}^2 + \lambda \|\mathbf{L}_k \cdot \mathbf{x}_k\|^2 + \delta \|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2, \quad (2)$$

where $w_{k,ij} = \exp(-\|\mathbf{x}_{k-1,i} - \hat{\mathbf{x}}_{k,ij}\|/\sigma)$ is the weight measuring the impact of a nearby example $\hat{\mathbf{x}}_{k,ij}$ on $\mathbf{x}_{k,i}$. $d_{k,i-1} = \|\mathbf{x}_{k,i} - \mathbf{x}_{k,i-1}\|$ is the distance between $\mathbf{x}_{k,i}$ and its previous sample $\mathbf{x}_{k,i-1}$. $\mathbf{L}$ is the tridiagonal Laplacian matrix, satisfying $\mathbf{L}_k(i,i) = 1, \mathbf{L}_k(i,i-1) = -\frac{d_{k-1,i-1}}{d_{k-1,i-1}+d_{k-1,i}}, \mathbf{L}_k(i,i+1) = -\frac{d_{k-1,i}}{d_{k-1,i-1}+d_{k-1,i}}$. The first term favors path to be closer to the samples on the shape distribution. The second and third terms prefer shorter and smoother paths. The last term specifies that the path should be somewhat close to the previous iteration, which ensures numerical stability. A small $\delta = 0.0001$ is used in our experiments.

In each iteration, given the optimal solution of the previous step (or from the initial morphing path) and after finding the nearest sample shapes for each sample point, this energy function is quadratic, and thus can be solved efficiently by solving a linear system. The iterative optimization terminates when converged (i.e. when the average position change $\overline{\|\Delta \mathbf{x}_k\|} < \varepsilon, \varepsilon = 10^{-3}$ in our experiments).

Our method is insensitive to the choice of parameters. $\sigma = 0.01, \gamma = 0.01$ and $\lambda = 100$ are used in *all* of our experiments. Fig. 8 shows the results of changing one parameter while keeping others unchanged. The experimental results are based on the SCAPE dataset and the morphing paths are visualized using PCA (see Sec. 4 for more details). σ controls the impact of nearby samples. The resulting path is fairly stable even with a significant change of σ. Increasing γ and λ tends to produce shorter and smoother paths (at a cost of slightly more deviation from the examples). Even with substantial change of parameters (by a factor of 10), the resulting paths still look plausible. We set these parameters such that they provide a good balance between smoothness and closeness to examples.

3.2.3. Morphing Sequence Reconstruction

After obtaining the optimized path in the feature coordinate space, we can generate a sequence of morphing shapes. We first re-sample the path with a specified number of samples and by default with even spacing between samples. This produces time-uniform morphing results. Alternatively, samples could be distributed along the time in a non-uniform manner if this is preferred. Given a sample $\tilde{\mathbf{x}}$, the interpolated shape $\tilde{S}$ in the morphing sequence can be reconstructed from the corresponding feature coordinates [BVGP09]. Since the coordinates are rotation and translation invariant, the reconstructed shape does not come with consistent orientation information. We further interpolate a rigid transform for each interpolated

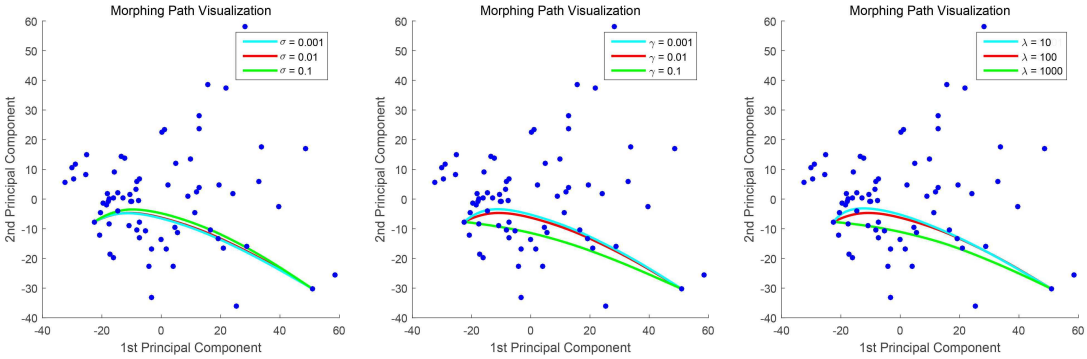


Figure 8: *Visualization of the morphing paths with varying parameters. Left: changing σ to 0.001, 0.01 (default) and 0.1; middle: changing γ to 0.001, 0.01 (default) and 0.1; right: changing λ to 10, 100 (default) and 1000.*

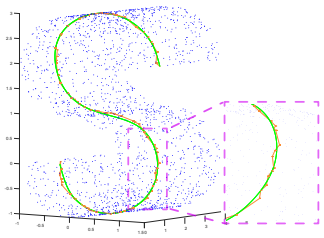


Figure 9: *Path optimization using our approach (green) and [RDSK06] (red).*

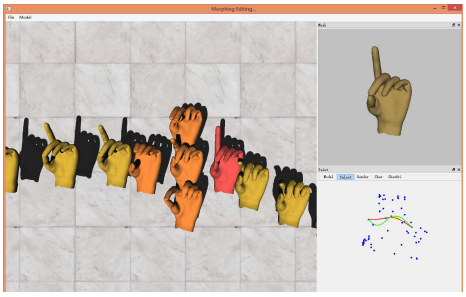


Figure 10: *Our interface for morphing editing.*

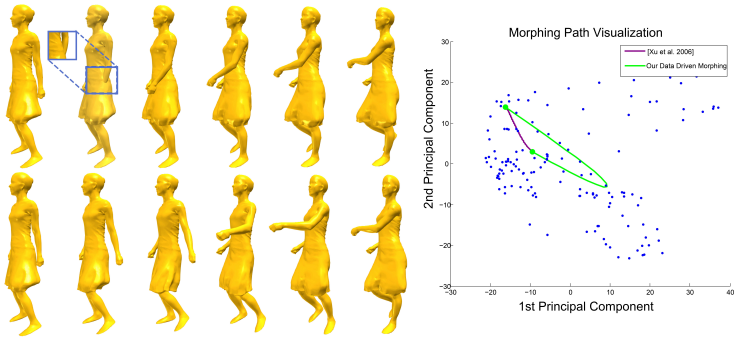
shape. We obtain rigid transforms T_s and T_t from the interpolated shape $\tilde{S}$ to the source model M_s and target model M_t, respectively. As correspondences are known, the rigid transforms can be obtained explicitly using a small number of correspondences. The rigid transforms are then decomposed into the rotation and translation components, with rotations interpolated using quaternion interpolation and translations interpolated using linear interpolation based on the time. The interpolated transform $\tilde{T}$ is applied to $\tilde{S}$ to obtain a globally aligned interpolated shape.

We demonstrate our data-driven morphing using a challenging synthetic example. As shown in Figs. 6 and 7, the source and the target models are springs but rotated along opposite directions (as indicated by the texture). Using non data-driven methods, including [HAWG08] and linear interpolation of patch-based LRI [BVGP09], self-intersections occur due to the substantial change from the source to the target (Fig. 6). For data-driven morphing with a cylinder (pointing inwards) as a reference model for guidance, the deformation is still significant, and the previous data-driven method [GLHH13] does not work well. Our data-driven method produces reasonable output (Fig. 7).

3.2.4. Comparison with approximating geodesic paths

Ruggeri et al. [RDSK06] propose an approximating geodesic path algorithm for point sets. Their approach uses an energy formulation with similar terms to ours, including closeness to samples, and the path length. However, our approach is different in that instead of using one nearest neighbor for each sample on the path, we use multiple samples in the neighboring space. We further include a Laplacian term for smoothness and use an iterative approach to incrementally refine the curves. While their method is effective for point sets with good distribution, as shown in Fig. 9, when the points are unevenly distributed, the path produced by their method is less smooth than ours, and our path tends to pass nearby areas with sufficient number of samples rather than passing through samples directly.

As the underlying surface is analytical, we further perform quantitative comparison. The geodesic path can be obtained exactly and used as ground truth. Following [RDSK06], we use $e_{geod}(g^*) = |g^* - g_a|/g_a$ to measure the normalized difference between the length of a given path g^* and the length of the ground truth geodesic path g_a. Our approach has $e_{geod} = 0.27\%$ whereas for [RDSK06], $e_{geod} = 4.5\%$. This shows that our path is significantly shorter than that of [RDSK06]. $e_{path}(P^*) =$



(a) (b)

Figure 11: *Comparison of dancer morphing results using the dataset from [VBMP08]. (a) the result of [XZWB06] (top) and our method (bottom), (b) visualization of morphing using feature coordinates and dimensionality reduction to 2D. Purple: [XZWB06]; Green: our data-driven morphing.*

$max(d_{path}(P^*,P_a),d_{path}(P_a,P^*))/g_a$ is another measure, where P^* is a given path, P_a is the ground truth path, and $d_{path}(P_1,P_2)$ measures the average distance from points on the path P_1 to path P_2. $e_{path}(P^*)$ measures how close points on P^* are to the ground truth path P_a on average. $e_{path} = 1.75\%$ for our method, which is smaller than 1.85% for [RDSK06].

As the coordinate space is of high dimensionality, and hence inevitably has a sparse sampling of examples, our method is thus more suitable.

4. Interactive Morphing Editing

By using a data-driven approach, our method automatically generates a smooth morphing sequence given a pair of source and target models at interactive rates. However, in practice, although the obtained path looks generally realistic, it may not be what the user expects. Instead of generating an entirely new morphing sequence, it would be much more efficient to enable the user to edit or adjust the morphing sequence to satisfy their needs. Building on our efficient data-driven morphing, we further develop a novel tool for interactive data-driven morphing editing.

Fig. 10 shows our interface for morphing editing. The user is able to see the current morphing result (in the top left pane). Moreover, we visualize the examples in the coordinate space by dimensionality reduction using principal component analysis (PCA) to two dimensions. Note that this transformation is used to map example shapes, as well as newly *interpolated* shapes to the 2D space for visualization. Standard non-linear mappings such as Multi-Dimensional Scaling can be used for effective dimensionality reduction and visualization of *example* shapes, and may better depict the relationships between them. However, when new shapes are added, pairwise distances need to be recalculated, which is too slow for interactive editing, and can also cause confusion as the positions of example shapes change after each interaction. A possible alternative is to use out-of-sample extension [BPV03] to embed newly interpolated shapes without recalculating the mapping. However, it is only approximate and can be sensitive to the distribution of the original data.

In the bottom right pane, each blue point represents an example, and the green and red curves represent morphing paths before and after editing. The morphing sequence is also visualized in the left pane as a sequence of models for easier selection. The user is free to choose a model on the current path. Once a model is selected, the four nearest examples are shown (in orange) around the selected model. These four models are also selectable and the selected model will be shown in the top right pane for exploration. If the user believes one of them is in the direction of their intended editing, the user can choose one of them, and it is incorporated into the energy formulation (Eqn. 2) as a hard constraint. The data-driven morphing algorithm is applied to generate a new morphing sequence. If the user is not yet satisfied with the new result, further editing can be incrementally applied.

5. Results

Our experiments were carried out on a computer with an Intel Xeon E5-2620 CPU with 8GB memory and sped up in parallel by OpenMP. We used various datasets from the existing research, including SCAPE [ASK*05], face [ZSCS04], dancer, lion [SP04], handstand, jumping [VBMP08] and a shape collection containing 23 face scans and 65 hand models (see the supplementary material for a list of shapes). Running times for typical examples are shown in Table 1. Our method is interactive, even with a reasonably large number of example models. After a one-off path optimization which takes under a second, it takes less than a second to reconstruct a frame, whereas [GLHH13] takes 25.45s for path optimization for the hand example (Fig. 1) and several seconds to reconstruct a frame. Note that the timing for frame reconstruction is based on a single thread, and in practice as each frame can be reconstructed *independently*, we use

Dataset	# Vertices	# Models.	Path Optimization (s)	Frame Reconstruction (s)
Scape	12500	71	0.860	0.690
Face	13637	41	0.902	0.716
Hand	7207	65	0.629	0.683

Table 1: *Statistics of the datasets and average running times. The timing of frame reconstruction is based on a single thread.*

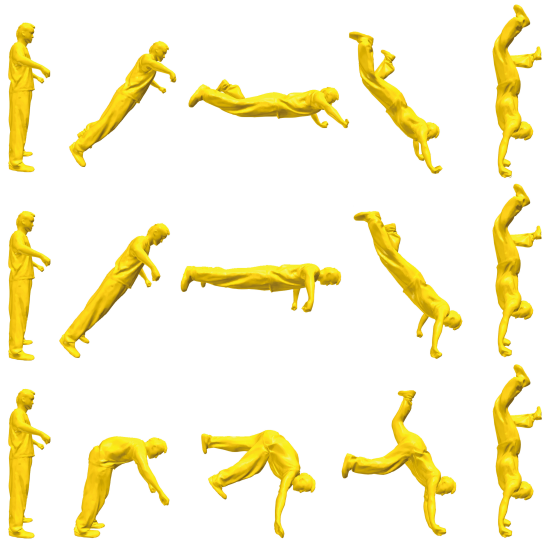


Figure 12: *Morphing results using the handstand dataset from [VBMP08]. First row: [HAWG08], second row: existing data-driven morphing [GLHH13], third row: our data-driven morphing.*

Figure 13: *Morphing results using the jumping dataset from [VBMP08]. First row: patched based LRI [BVGP09], second row: existing data-driven approach [GLHH13], third row: our data-driven morphing.*

multi-threading so that the morphing sequence can be generated within a couple of seconds, which makes interactive editing feasible.

We now show various morphing results and compare our method with state of the art (either non data-driven or data-driven). For these results, the leftmost and rightmost shapes are the source and target models and several in-between models from the animation sequence are shown in the paper; please refer to the supplementary video for the morphing animations. For our method, we use the first model in each dataset as reference. While different reference models may lead to somewhat different morphing results, our approach is robust to the choice of reference models: similar and plausible morphing results are obtained with different reference models.

Fig. 11(a) shows an example of a *dancer* morphing using the dataset from [VBMP08]. Geometry based methods such as [XZWB06] produce a smooth transition but have self-intersections in space. Our data-driven method produces a smooth and artifact-free result. To better understand how these methods compare, we use PCA to visualize the example models and morphing paths by projecting them to the first two principal dimensions (Fig. 11(b)). The geometry based method (purple) is shorter but out of the shape distribution. Our method (green) produces a smooth path that nicely follows the shape distribution.

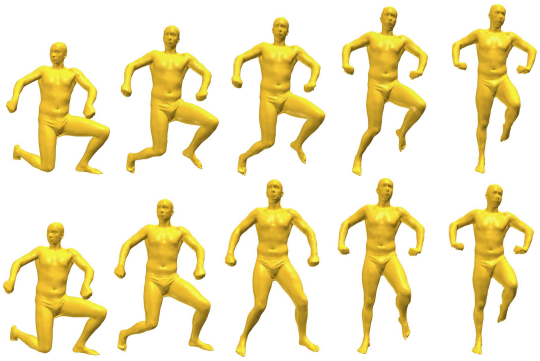


Figure 14: *Morphing results using the SCAPE dataset from [ASK*05]. Top row: existing data-driven approach [GLHH13], bottom row: our data-driven morphing.*

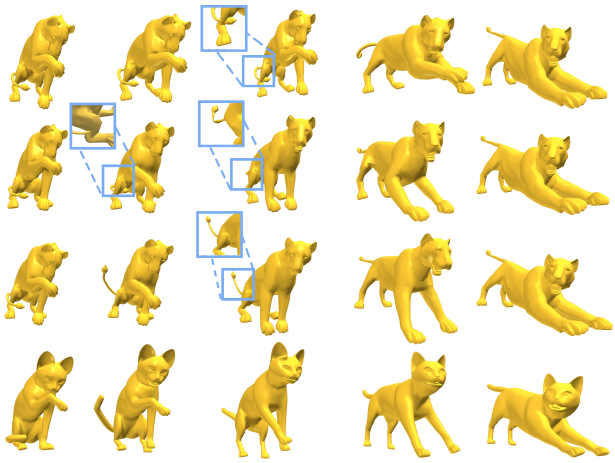


Figure 15: *Morphing results of a lion using the dataset from [VBMP08] . First row: [HAWG08], second row: existing data-driven approach [GLHH13], third row: our data-driven morphing, fourth row: our data-driven morphing result transferred to a cat.*

Figs. 12 and 13 show two examples of human morphing. For the *handstand* example, significant deformation exists and the existing geometry based method produces a very rigid result which cannot be performed by a human and thus does not look realistic. The existing data-driven method [GLHH13] produces a slightly better result. However, because discrete subspaces are used, their method fails to find good reference models. Our method produces a realistic morphing result. For the *jumping* example, existing methods produce self-intersections (although less artifacts exist on the result of [GLHH13]). Our method manages to find a smooth path following the shape distribution, thus avoids the artifacts. Our method also produces more time-uniform animation than [GLHH13]; see the supplementary video. Fig. 14 gives the morphing results of a human using the *SCAPE* dataset. The existing data-driven method [GLHH13] fails to find useful references, producing smooth morphing but involves unbalanced poses which are not humanly possible. Our data-driven method produces natural morphing result.

Similar to [GLHH13], our method assumes an example model database is provided. To apply to models which are different from those in the database but has similar behavior, deformation transfer [SP04, GLHH13] is used. An example is shown in Fig. 15. Geometry based method [HAWG08] produces self-intersection and the existing data-driven approach [GLHH13] generates a rather twisted tail (see the region highlighted in the blue squares). Our data-driven method produces more realistic results for *lion* morphing. By using deformation transfer, morphing results are transferred to a cat model (bottom row).

Figs. 16 and 17 show two examples of our data-driven

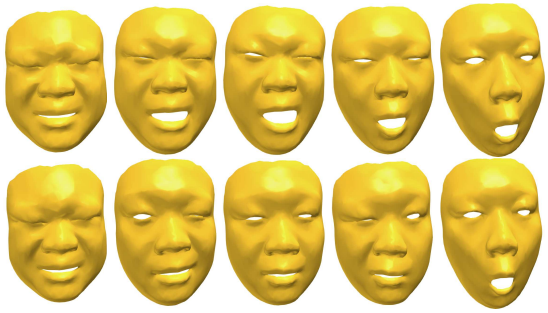


Figure 16: *Data-driven morphing editing of the face dataset. Top row: before editing, bottom row: after editing.*

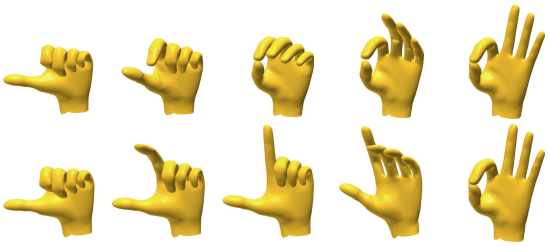


Figure 17: *Data-driven morphing editing of the hand dataset. Top row: before editing, bottom row: after editing.*

morphing editing. For each example, only *two* reference models are interactively selected, and the resulting morphing sequences are both realistic and much richer: the face example introduces additional eye blinking and the hand example produces multiple finger actions. The morphing is solved as a whole, so the smoothness is not compromised even with introduced user constraints.

6. Conclusion

In this paper, we propose a novel data-driven approach to realistic shape morphing by exploiting knowledge in the example models. Significantly better results than state-of-the-art methods are obtained. We further propose a novel interactive data-driven morphing editing technique which allows users to produce desired morphing with little effort.

As a data-driven approach, our method may not work well if the example database does not sufficiently cover the plausible deformation space. Compared with the existing data-driven approach [GLHH13], we have demonstrated that given the same example shape repository, our approach better utilizes the knowledge hidden in the shape repository and can often produce better results. It is still an open question to further exploit the information in the shape repository to produce realistic morphing with fewer example models.

Our current implementation is purely CPU-based which achieves interactive performance. The method could be further sped up by GPU acceleration, potentially providing

a more responsive, possibly real-time user interaction. We would like to investigate this in the future.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61502453 and No. 61611130215), Royal Society-Newton Mobility Grant (No. IE150731), CCF-Tencent Open Research Fund (No. CCF-TencentIAGR20150116), the Key Project of Institute of Computing Technology (No. 20166040), the Key Project of Chinese Academy of Sciences (No. KJZD-EW-TZ-L03), and the National Key Technology Research and Development Program of China (No. 2013BAK03B07).

References

[ACK16] AVERBUCH-ELOR H., COHEN-OR D., KOPF J.: Smooth image sequences for data-driven morphing. *Comput. Graph. Forum 35*, 2 (2016), 203–213. 4

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proc. ACM SIGGRAPH* (New Orleans, USA, 2000), pp. 157–164. 2

[Ale03] ALEXA M.: Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2-3 (2003), 105–114. 3

[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*, 3 (2005), 408–416. 8, 9

[BPV03] BENGIO Y., PAIEMENT J.-F., VINCENT P.: Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *In Advances in Neural Information Processing Systems* (2003), MIT Press, pp. 177–184. 8

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM Trans. Graph. 28*, 3 (2009), 36:1–6. 2, 3, 4, 5, 6, 7, 9

[CL09] CHU H.-K., LEE T.-Y.: Multiresolution mean shift clustering algorithm for shape interpolation. *IEEE Trans. Vis. Comp. Graph. 15* (2009), 853–866. 3

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. *Comput. Graph. Forum 30*, 8 (2011), 2246–2257. 3, 4

[GLHH13] GAO L., LAI Y.-K., HUANG Q.-X., HU S.-M.: A data-driven approach to realistic shape morphing. *Comput. Graph. Forum 32*, 2 (2013), 449–457. 1, 2, 4, 5, 6, 7, 8, 9, 10

[HAWG08] HUANG Q.-X., ADAMS B., WICKE M., GUIBAS L. J.: Non-rigid registration under isometric deformations. In *Proc. Symposium on Geometry Processing* (Copenhagen, Denmark, 2008), Eurographics Association, pp. 1449–1457. 2, 3, 6, 7, 9, 10

[HLW07] HU J., LIU L., WANG G.: Dual Laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 271–277. 3

[HRS*14] HEEREN B., RUMPF M., SCHRÖDER P., WARDETZKY M., WIRTH B.: Exploring the geometry of the space of shells. *Comput. Graph. Forum 33*, 5 (2014), 247–256. 4

[HRWW12] HEEREN B., RUMPF M., WARDETZKY M., WIRTH B.: Time-discrete geodesics in the space of shells. *Comput. Graph. Forum 31*, 5 (2012), 1755–1764. 4

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Trans. Graph. 27*, 2 (2008), 12:1–13. 3

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Trans. Graph. 26*, 3 (2007), 64:1–8. 3

[LDSS99] LEE A. W. F., DOBKIN D., SWELDENS W., SCHRÖDER P.: Multiresolution mesh morphing. In *Proc. ACM SIGGRAPH* (Los Angeles, USA, 1999), pp. 343–350. 2

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

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M. H.: Example-based elastic materials. *ACM Trans. Graph. 30*, 4 (2011), 72:1–8. 3

[RDSK06] RUGGERI M. R., DAROM T., SAUPE D., KIRYATI N.: Approximating geodesics on point set surfaces. In *Symposium on Point-Based Graphics* (Boston, USA, 2006), Eurographics Association, pp. 85–93. 7, 8

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405. 2, 4, 8, 10

[SRC01] SLOAN P.-P. J., ROSE III C. F., COHEN M. F.: Shape by example. In *Proc. Symposium on Interactive 3D Graphics* (Research Triangle Pk, North Carolina, USA, 2001), ACM, pp. 135–143. 4

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3 (2005), 488–495. 4

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (2008), 97:1–9. 8, 9, 10

[vTSSH15] VON TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: Real-time nonlinear shape interpolation. *ACM Trans. Graph* (2015), to appear. 3

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Comput. Graph. Forum 29*, 2 (2010), 309–318. 3

[WPP07] WANG R. Y., PULLI K., POPOVIC J.: Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3 (2007), 73. 5

[XZWB06] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. *Graph. Models 68*, 3 (2006), 268–281. 3, 4, 8, 9

[ZSCS04] ZHANG L., SNAVELY N., CURLESS B., SEITZ S. M.: Spacetime faces: High resolution capture for modeling and animation. *ACM Trans. Graph. 23*, 3 (2004), 548–558. 8