Computer-Aided Design 136 (2021) 103027

Contents lists available at ScienceDirect

Computer-Aided Design

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

Part-based data-driven 3D shape interpolation[☆]

Melike Aydınlılar [a], Yusuf Sahillioğlu [b,*]

[a] INRIA, LORIA, Nancy, France
[b] Middle East Technical University, Computer Engineering Department, Ankara, 06800, Turkey

ARTICLE INFO

Article history:
Received 10 June 2020
Received in revised form 3 March 2021
Accepted 6 March 2021

Keywords:
3D shape interpolation
3D shape morphing
Part-based
Data-driven

ABSTRACT

An active problem in digital geometry processing is shape interpolation which aims to generate a continuous sequence of in-betweens for a given source and target shape. Unlike traditional approaches that interpolate source and target shapes in isolation, recent data-driven approaches utilize multiple interpolations through intermediate database shapes, and consequently perform better at the expense of a database requirement. In contrast to the existing data-driven approaches that consider intermediate shapes as full inseparable entities, our novel data-driven method treats the shapes as separable parts. In particular, we interpolate parts over different intermediate shapes and merge them all in the end, which brings more flexibility and variety than the existing ways of interpolating the full shape as a whole over one fixed set of intermediates. To be able to proceed consistently over different sets of intermediate shapes, we construct a unified framework based on parametric curves. We justify the two key points in the proposed method, interpolating parts separately and data-driven by curve parameterization, in the qualitative and quantitative evaluations. We demonstrate promising results in comparison with five other techniques. Our method morphs not only poses but also forms, e.g., turning one person to another. The results are improved further with a mild data augmentation procedure that is based on the original algorithm. As a side contribution, we provide a public articulated hand dataset with fixed connectivity, which can be used in the evaluation of other interpolation methods.

1. Introduction

Shape interpolation is an important problem in computer graphics. It has many application areas ranging from key frame animation to content creation. In the interpolation problem, given a source model and a target model, the goal is to create a smooth transition of in-between models that are both physically plausible and geometrically sound.

The simplest interpolation technique is the linear interpolation. Given two shapes represented as mesh structures with n corresponding vertices denoted by v_1^k and v_2^k, the resulting interpolated mesh can be expressed as:

$$v_{\text{result}}^k = \alpha v_2^k + (1-\alpha)v_1^k, 0 \leq \alpha \leq 1, 1 \leq k \leq n, \alpha \in \mathbb{R}, k \in \mathbb{N} \quad (1)$$

It is possible to create as many interpolated meshes as desired by updating the α value. Note that the fixed connectivity between the first two meshes is maintained in the resulting mesh.

This scheme and its more sophisticated variants tend to produce artifacts such as self-intersections and unnatural poses in the result mainly because of the fact that two arbitrary input shapes to be interpolated are too dissimilar from each other.

A data-driven approach, [1] for instance, has the potential to solve this problem by inserting intermediate database shapes between two dissimilar input shapes. In particular, a data-driven approach uses a set of smooth transitions between *similar* shapes that connect the source input to the target input which were *dissimilar* in the first place. The main drawback of data-driven approach is the demand of extra database shapes, whereas the main advantage is the increased accuracy.

Data-driven methods differ mainly by the way they perform the interpolation between two similar intermediate shapes. Since the expressiveness of this line of methods is limited with the size of the used database and also the variation within the database, there is no perfect way to define the similarity, and hence the interpolation scheme.

We improve these data-driven methods by bringing more variety over the same database. The main idea is to get more out of each shape in the database by considering smooth transitions over their parts, not over themselves. The immediate advantage of this idea is best shown with an example: the intermediate shapes in the second row of Fig. 1 provide a smooth transition for the left leg (green), but not for the right arm. The first row, however, has a smooth transition for the right arm (brown) over

[☆] This paper has been recommended for acceptance by Shi-Min Hu.
[*] Corresponding author.
E-mail address: ys@ceng.metu.edu.tr (Y. Sahillioğlu).

https://doi.org/10.1016/j.cad.2021.103027
0010-4485/© 2021 Elsevier Ltd. All rights reserved.

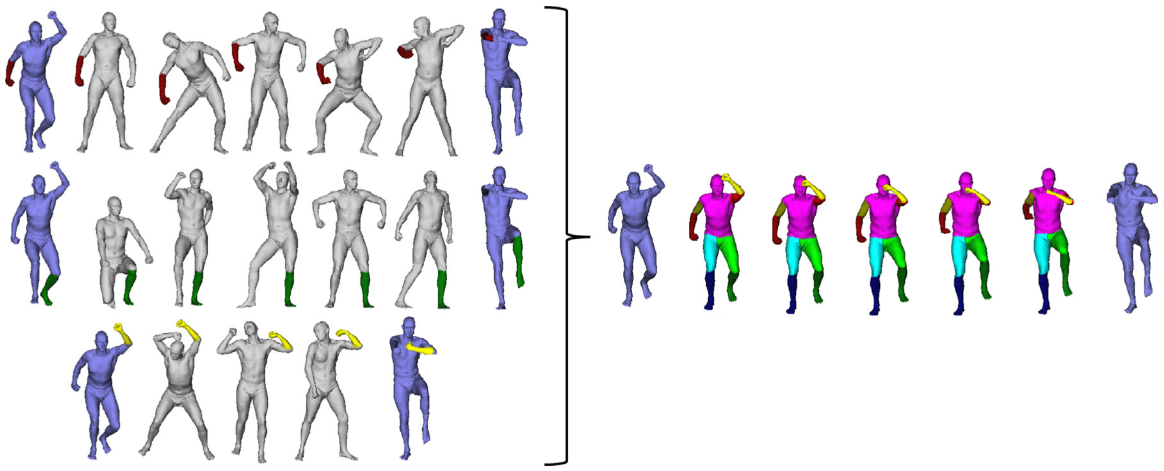

Fig. 1. Part-based interpolations over different paths (three rows at left) are merged into the final global data-driven interpolation (right). Treating the dataset part-wise, the key idea of this paper, improves its richness, leading to better interpolation results, as shown in Section 5 and the supplementary video. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a different set of intermediate shapes. Similarly, we get a smooth transition for the left arm from yet another set of intermediates, namely row 3 (yellow). When the smooth transitions of the remaining parts are merged, a smooth overall transition is obtained (right). Note that it is not possible to find an as smooth transition between the same source–target shape pair if one treats them as inseparable whole objects (Fig. 6 - rows 2 and 5) due to the lack of model variety in our small database. Note also that the smooth transitions of different parts are obtained through different sets of intermediate shapes under this regime. To be able to consistently merge these transitions, or equivalently interpolations, we introduce a unified Bezier curve framework which parametrizes the sets of intermediate shapes the same regardless of the differences in their cardinalities. This unified merging operation is also based on closed-form evaluations, which makes its speed close to that of the simplest scheme in Eq. (1).

We note that the source code and the executables for the method that we present in this paper, as well as our new articulated hand database with fixed connectivity, are publicly available at: http://ceng.metu.edu.tr/~ys/pubs/interp.zip.

A preliminary short version of this work was published in [2]. Compared with that version, we in this paper improve comparisons, evaluation, timing, and literature review as well as algorithmic details such as dissimilarity measure and merging step. We also add separate sections on further findings and limitations as well as provide a video which can be watched at: http://ceng.metu.edu.tr/~ys/pubs/interp.mp4.

2. Related work

We overview the literature on interpolation of the source and target shape models in two categories: isolated and data-driven methods, where the former deals with two input shapes without using any intermediate shape support from a database and has been studied quite extensively [3]. The latter is a relatively new category.

The simplest mechanism to compute in-between shapes is the linear interpolation which is likely to cause artifacts like volume loss and self-intersections in the presence of deformations. Especially in large deformations these artifacts become intolerable [4]. To overcome this problem, [5] attempts to extend the intrinsic shape parameters for interpolation to 3D shapes using a local greedy approach. Differential coordinates [6], rotation-invariant coordinates [7], and isometry-invariant intrinsic coordinates [8] can be used for relatively robust interpolations, where only the first one is able to work with non-manifold surfaces. Kilian et al. [9], on the other hand, treat shapes as points in Riemannian surfaces and interpolate along geodesics. [10] also use geodesics in a shape space but works only with solid models. Physically-based approach of [11] also works with solid tetrahedral models. Chu and Lee [12] use nearly rigid parts and perform interpolation in their gradient space. Solutions based on block-coordinate descent [13] and BFGS method [14] also proved useful for interpolating shapes between source–target pairs.

Instead of performing linear interpolation on absolute coordinates, [15] and [16] linearly interpolate the edge lengths and dihedral angles which are used in the solution of coordinates. Reconstruction from edge lengths and dihedral angles to coordinates, however, requires computationally expensive non-linear optimization. [17] proposes an interpolation method that accelerates non-linear solutions to real-time rates. Alternating optimization techniques [18,19] use the as-rigid-as-possible deformation framework that keeps the local transformations close to a rigid motion. [20] in this category decomposes the transformation matrices into rotational and non-rotational components and interpolates them separately. Defining blending operations on a spatio-structural graph composed of medial curves and sheets made 3D interpolation under topology variations possible [21]. Concepts from continuum mechanics are also adapted to the interpolation problem [22–25]. The global optimization for splines in shape space, which is the main theme in this line of works, is relatively inefficient. To address this issue, [26] converts the complex minimization problem in shape space into a few standard cubic spline fitting problems in the embedded feature space. Smooth interpolation between concatenated animations, instead of static models, is also achieved by geometric flows of curves [27].

In addition to the 3D shape interpolation techniques discussed thus far, there also exist planar shape interpolation methods that are designed for two dimensional domains and can be extended to the third dimension after some effort. One of the pioneering examples in this area is Sederberg et al. [28], which interpolates the intrinsic definitions of the initial and final 2D shapes. Alexa's method [29] generates as-rigid-as possible interpolation for 2D images [30]. Extending this idea to 3D models requires consistent tetrahedralization which is not trivial to obtain. Chien et al. [31] propose a fast and theoretically sound interpolation method that blends planar harmonic mappings represented in closed-form. Previous version of this algorithm [32] provides guarantees on local injectivity and a pointwise bound on conformal distortion at the expense of a slow nonlinear optimization. Recently, the interactive planar interpolation method of [33] shows promising results under extreme deformation and topology change.

All of the methods given so far are inherently prone to artifacts like self-intersections, physically improbable movements, and unnatural poses as they consider the input shape pair in isolation without any prior knowledge on the shape family they come from. A data-driven, or equivalently example-based, approach can alleviate this problem by enabling usage of existing shapes from the same family, hence providing a prior knowledge. Observed recently on real-time elastic [34–36] and plastic [37] deformation applications, the first example to this paradigm on shape interpolation problem dates back to 2001 [38] in an interactive framework and we see variations that are much more closer to our process [1,39,40]. Specifically, [39] employs a hybrid approach that combines vertex and edge shape spaces. [1] creates a model subspace from a given database and explore this using a shortest path approach. [40] differs in the way it interpolates the consecutive shapes in the shortest path. While [1] imports the interpolation procedure of [18] for this purpose, [40] uses rotation-invariant coordinates [7] which handles large-scale deformations better. We, on the other hand, employ a much simpler and faster interpolation procedure based on closed-form polynomial evaluations in the Bezier framework. Averbuch-Elor et al. [41] propose a data-driven method for 2D image interpolation based on shortest path computation. Although database construction is the main disadvantage of these data-driven methods, one can automate this tedious process by employing articulated example-based [42,43] or articulated sketch-based [44] shape retrieval techniques on large diverse databases such as SketchUp 3D Warehouse.

Once sufficient amount of data is made available, deep neural networks such as [45,46] appear as valid alternatives to data-driven methods. More recent learning-based approaches utilize hierarchical graph network [47], two-level variational autoencoder network [48], or decomposer–composer network [49] for learning structure-aware shape generation. They require consistent part segmentation information throughout the diverse shape collection, which is not trivial to achieve. Besides, outputs of these methods are merely part-wise watertight whereas our method produces one global watertight mesh for the whole object. Two convolutional variational autoencoders are learned by [50] in order to encode source and target shapes to their latent spaces more compactly. Shapes are then interpolated in the learned latent space.

Regardless of the two categories we discussed, i.e., isolated and data-driven methods, one first needs one-to-one correspondences between the shapes to be interpolated. This tedious process can be automated with fully automatic correspondence algorithms [51]. More recent deep learning methods, in particular, make the correspondence computation process more robust especially in the presence of diversity, noise, and partiality [52, 53]. In practice, however, artists deform a template mesh in its rest pose to different poses, which in turn preserves the fixed connectivity and hence the one-to-one correspondence between shapes. In this work, we work on such databases that are already in one-to-one correspondence. Note finally that, the utilization of shortest paths on data-driven methods arises in other geometry processing problems as well, e.g., shape correspondence over collections [54,55].

3. The algorithm

Overview of our algorithm can be captured with the example given in the end of Section 1. We continue to provide details.

3.1. Input and output

Our input is a source shape and a target shape as well as a database of potentially intermediate shapes from the same family. All the shapes are discretized as mesh structures with vertices, edges, and polygonal faces. We assume one-to-one correspondence between all shape pairs is available. A manual segmentation of parts on one shape is transferred automatically to all other shapes using the one-to-one correspondences.

Our output is a physically plausible and geometrically sound smooth interpolation between the source shape and the target shape. This is made possible by a data-driven approach which utilizes our databases of potentially intermediate shapes.

3.2. Graph structure

We build a complete graph where each shape is a vertex, and edges between shapes are weighted by the dissimilarity between them, i.e., the more the connected shape pair is dissimilar, the more their corresponding edge weight is. The main idea of using a graph of this form is to replace a “bad” interpolation between the source shape and the target shape by the composition of “good” interpolations on the shortest path between them. To this end, we run Dijkstra’s shortest paths algorithm on our graph, which returns the desired set of intermediate shapes in order. For accuracy and efficiency purposes, we prune our graph prior to Dijkstra by removing edges whose weights (Section 3.2.1) are too large, i.e., greater than 1.5 times the average edge weight. The rationale behind this action is to prevent a potential shortest path that involves edges with outlier weights, or equivalently shape pairs with potentially “bad” interpolations, e.g., the path $7+3+5+6+4+4=29$ should be preferred over the shortest path $4+19+3+2=28$ that involves the outlier 19 (see also Fig. 2). Removing outlier edges also accelerates Dijkstra whose execution time depends on the number of edges.

3.2.1. Edge weights

We use the pairwise distance measure suggested by [1] as our edge weights. In this measure, the dissimilarity between two shape is computed by accumulating the Euclidean distances between their corresponding vertices:

$$d(S_i, S_j) = \sum_{k=1}^{n} \|v_i^k - v_j^k\|^2 \tag{2}$$

where v_i^k is the k^{th} vertex of the shape S_i that is in correspondence with v_j^k from S_j. We visualize the behavior of this dissimilarity measure by the multidimensional scaling technique [56], which separates all shape pairs in 2D by the amount of dissimilarity (Eq. (2)) between them (Fig. 2).

We make our dissimilarity measure rigid motion invariant by a transformation-based preprocessing step performed before the computation of Eq. (2). In this step, we resolve the translation and rotation ambiguities optimally [57] based on the most dominant parts, e.g., the torso segment for humans, in the guidance of perfect correspondences. The resulting rigid transformation is applied not only to the most dominant part but also to all of the other parts, hence aligning the shapes globally. Note that at this point individual parts do not necessarily, and are not likely to, align perfectly, which is meaningful for the computation of part-based dissimilarity scores while computing a separate graph for each part (Section 4.1).

This fast and theoretically optimal preprocessing puts the shapes into relatively correct orientations, e.g., coinciding center of masses, except a few cases where the faces look at opposite directions (back/front flip). In such cases, we manually rotate

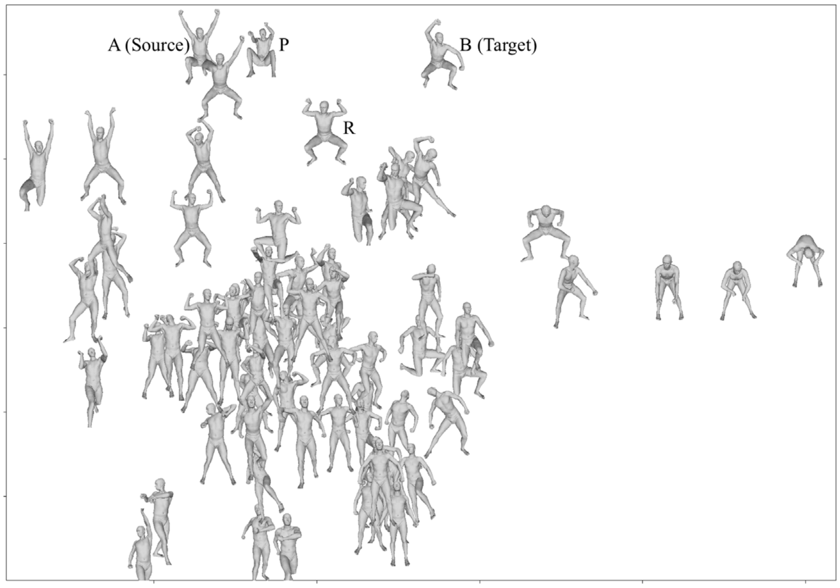

Fig. 2. SCAPE models are arranged in 2D with separations proportional to their dissimilarities captured by Eq. (2). We label some models to demonstrate the benefit of our outlier edge removal idea: A→P→R→B is preferred over the shortest path A→P→B that involves an outlier edge connecting 2 dissimilar models P and B.

one shape 180 degrees. Note that [1] uses exactly the same measure after a similar, but fully manual, preprocessing since their method requires the intermediate meshes to not only have suitable shapes, but also be in suitable global orientations to be useful.

We also tried several other weighting schemes. The method described by Osada et al. [58] chose several vertex pairs and computes the Euclidean distances between them. Then, the histograms of these values are compared using the chi-square method. Another metric we used is to compute the distance of several vertices to the center of mass of the model. Then, the histograms of these values are compared. These two methods did not significantly improve Eq. (2) based results.

We also used another family of experimental measures based on linear interpolation. Specifically, for each shape pair, we computed the linear interpolation for α = 0.5 (Eq. (1)). Then, we used several methods to assess the quality of the interpolated mesh. The weight would then be assigned inversely proportional to that quality. We tried using total mesh volume, total triangle area, and shape thickness quality metrics. However, most of the time the interpolated model, especially if the input shape pair differs significantly, were not reliable enough to run the quality metrics. Collapsed edges, inverted triangles and tetrahedra, as well as vertex positioning that damage the mesh integrity prevented reliable comparisons. These measures are also much more computationally expensive than Eq. (2).

4. Parts to be interpolated

As motivated in Section 1, the quality of data-driven interpolation results are determined by the expressiveness of the database in use. Using arbitrary databases, it is not possible to have all the possible shape configurations necessary to create satisfactory interpolation results. To overcome this problem, we apply the following partitioning and use the resulting parts as inputs for our graph structure.

We divide the SCAPE and FAUST models into 2, 5, and 9 parts and the Hand models into 6 and 11 parts. Other datasets are divided into 5, 7, 7, and 9 parts for the Cat, Horse, Dancer, and Centaur, respectively (Fig. 3). Our algorithm is not sensitive to the number of parts selected as one can see plausible interpolations

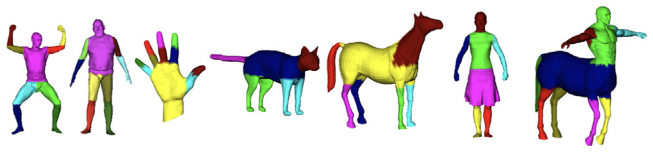

Fig. 3. Representative models of databases. Parts highlighted.

with different numbers, e.g., Fig. 7-rows 1–3, Fig. 10-rows 1–2, Fig. 15-rows 3–5, and Fig. 16. The main motivation of going with rigid part segmentation is the availability of automatic algorithms that can produce them over mesh sequences [59,60]. Besides, it is shown to be a good idea to treat rigid parts with fast interpolation and resort to slower non-linear interpolation only for the non-rigid connector joints [39]. Thanks to our data-driven approach, which is lacking in [39] that handles shapes in isolation, we are, however, not that strict about non-rigid joint separation, i.e., we move each vertex over its own closed-form Bezier curve (under unified time parameterization) in order to achieve a similar effect. As a matter of fact, our parts sometimes include rigid parts and non-rigid joints at the same time (Fig. 7-rows 2–3, Fig. 10-row 2, Fig. 14-rows 1, 3, 5, Fig. 15-rows 4–5, and Fig. 16-rows 2, 4–5) without any problems, which in turn enables the usage of many automatic semantic segmentation algorithms that may miss rigid parts [61,62].

Consequently, setting the upper limit of the number of parts parameter to the maximum rigid parts is the appropriate choice. Dividing already-rigid parts into further rigid parts is redundant and inefficient as the subparts from the same rigid region are expected to perform the same motion.

4.1. Unified merging

Having obtained the parts to be interpolated, we treat them separately to create their own graphs (Section 3.2). Note that the shapes are already globally aligned via [57] and parts are not aligned within themselves again before creating their graphs based on the dissimilarity scores (Eq. (2)). After creating the graphs, we find the sequence for each part, i.e., the source part and the target part plus the intermediate parts on the shortest path from the source to the target.

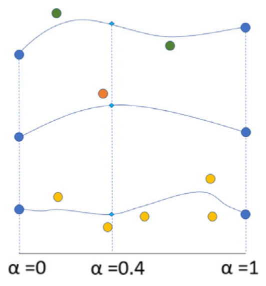

Fig. 4. Using Bezier curves for multiple paths (3 in this case) with different lengths allows unified timing and smooth transition.

Given a shape sequence, we make use of Bezier curves to move the involved vertices. Using the Bezier curve parameterization below, we are able to interpolate between any number of shapes in the sequence smoothly. This approximation is smoother than interpolating through mesh after mesh on line segments with sharp turns.

$$v^k_{\text{result}}(\alpha) = \sum_{i=0}^{m} v^k_i b_{i,m}(\alpha), 0 \le \alpha \le 1, 1 \le k \le n, \alpha \in \mathbb{R}, k \in \mathbb{N},$$
$$b_{i,m}(\alpha) = \binom{m}{i}\alpha^i(1-\alpha)^{m-i} \quad (3)$$

We use the standard De Casteljau's algorithm to fit a Bezier curve to each vertex in a mesh (Eq. (3)). Similar to the linear interpolation formula (Eq. (1)), we consider each of the n vertices separately and move them on their curves under unified time parameterization α. v^k_i denoting the kth vertex of the control/intermediate meshes in the computed shape sequence of size $m+1$, Eq. (3) gives us the position of the kth vertex at a given time α.

We apply the above method for each segmented part of a model separately. Each part has a different mesh sequence that can have different lengths. Another benefit of using the above mentioned curve approach is that given a time α, it is possible to compute vertex positions at the same time frame even with different mesh sequence lengths. This is illustrated in Fig. 4.

Since the source and target shapes are initially on top of each other, all their parts need to be translated after the interpolation. To this end, we first fix the torso, i.e., the reference frame, and then translate the center of mass of the limb interface to the center of the corresponding torso interface, hence allowing the meeting of the part interfaces. We provide Fig. 5 that demonstrates the shapes before and after these translations. Translation itself is not enough to obtain smoothness at the boundaries between parts of the merged models. To remedy this problem, we apply multiple passes of Taubin's fairing [63] only to the boundary region between the parts, which we call local smoothing ($\lambda = 0.6307$, $\mu = -0.6732$). We empirically decide on 20 passes, which is used in all experiments. Notice that a global smoothing without special consideration on the boundaries would over-smooth the mesh with the same number of passes. We, consequently, promote local smoothing over global smoothing (Fig. 5). Local smoothing is stopped automatically when the total displacement over the affected region is sufficiently small.

We close this section by discussing two potential issues about using Bezier curves. First, although interpolating positions using Bezier has a potential to break rigidity, i.e., forearm not straight anymore, we have not observed this shortcoming thanks to our part-based analysis that highly exploits the dataset. We, in other words, have always ended up with part poses that are sufficiently similar that this issue has never arisen. Second, since

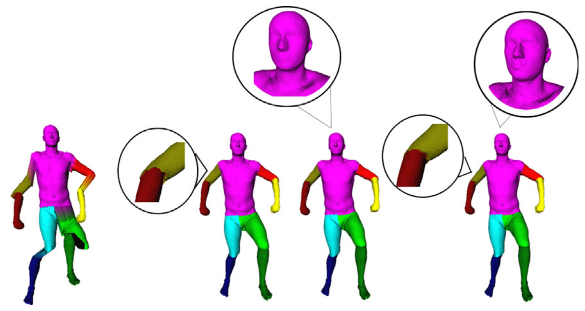

Fig. 5. Leftmost model shows the result of the part-based approach before aligning all the parts, i.e., the fixed torso and the limbs that are waiting to be translated. Middle left model shows the alignment result after the translations. Although it provides correct placements, boundaries between parts are not smooth (zoomed elbow). In the middle right model, the result of global smoothing is shown. Although it provides smooth boundaries between interpolated parts, it also loses the surface details (zoomed face). In the rightmost model, the result of local smoothing applied only to the boundary regions are shown. It provides smooth boundaries (zoomed elbow) yet preserves surface details (zoomed face).

Bezier curves do not necessarily pass through the interior control points, some of our intermediate shapes has the potential to be less influential on the result. Our part-based analysis utilizing enough variety of part poses has again solved this issue naturally. To validate this, we have replaced the Bezier curve framework with the Catmull–Rom spline framework which is guaranteed to pass through the intermediate shapes. We have not seen a significant difference between the results of these two parametric curve frameworks and consequently stuck with the more common Bezier framework.

5. Experimental results

We tested the performance of our data-driven shape interpolation algorithm on databases of articulated objects with non-uniform sampling. The first database is a reconstructed pose sequence of a human actor from the SCAPE benchmark [64], which contains 71 models, whereas the second one is our new Hand database with 31 models. The other databases consist of 27 Dancer models [65], 11 Horse models [66], 10 Cat models [66], and 6 Centaur models [4]. Despite the small sizes of these databases, our part-based solution achieves visually pleasing and natural interpolations between shape pairs, as demonstrated by the following figures and the supplementary video at http://ceng.metu.edu.tr/~ys/pubs/interp.mp4. We provide quantitative evaluations as well.

In order to emphasize our contribution, we make comparisons with five related methods that we call M1, M2, M3, M4, and M5. In M1, we maintain the data-driven paradigm by computing the Bezier approximation over the computed shortest paths. The important difference here is that the shapes are considered as a whole unlike our part-based consideration. M1 can also be considered as a simplified variant of [1] as the framework is almost identical except that [18]-based interpolation along the path is replaced here with a Bezier-based interpolation. In M2, on the other hand, we perform direct linear interpolation (Eq. (1)) between the source and the target, which is not data-driven at all. M3 is the state-of-the-art data-driven shape interpolation method [40]. We add a recent deep learning method [50] to our test suite under the name of M4. Finally, we use as-rigid-as-possible surface morphing [20] as M5.

We demonstrate our performance not only for pose interpolations but also for form interpolations using the inter-subject pairs from the FAUST dataset [67]. Also, a mild data augmentation is shown to improve our results even further (Section 5.3).

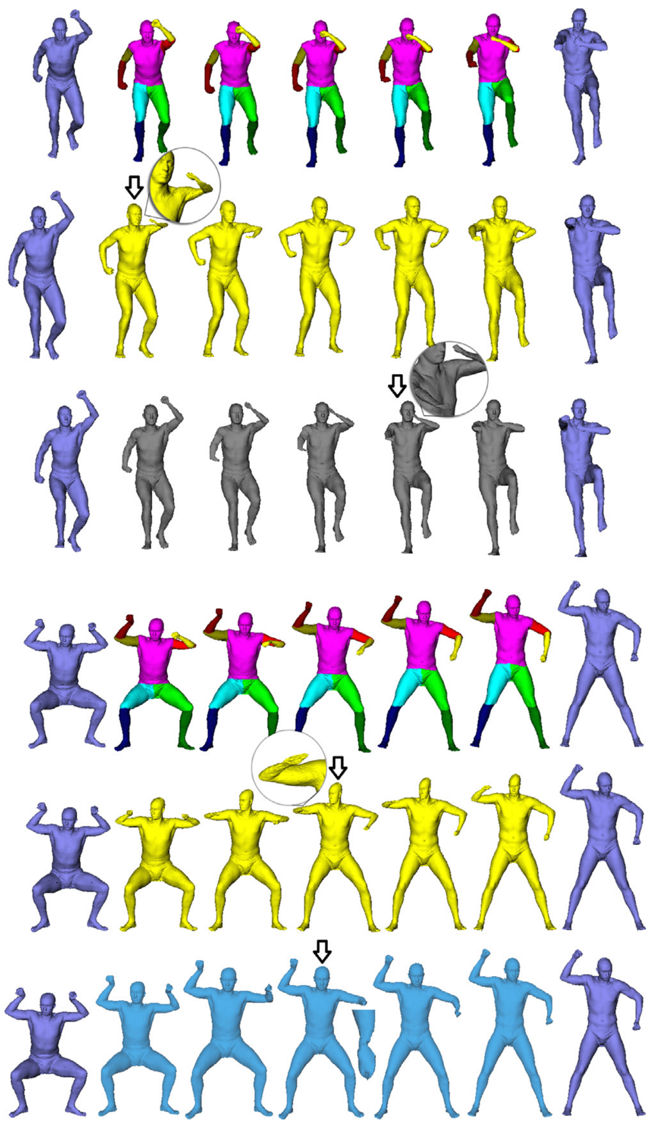
Fig. 6. Our results on two different pairs (rows 1 and 4) outperform the results of M1 (rows 2 and 5), M2 (row 3), and M5 (row 6). Notice the zoomed artifacts of M1, M2, and M5.

5.1. Qualitative evaluations

Our first example is from the SCAPE database (Fig. 6). Given the leftmost source shape and the rightmost target shape, our data-driven algorithm finds 3 different shortest paths for three different parts, namely the right and left arms (below the elbow), and the left leg (below the knee), as demonstrated in Fig. 1. The first row in Fig. 6 shows the unified merging of all 9 parts and hence our promoted result, which is better than the methods M1 and M2. Not surprisingly, we outperform M2 which is not allowed to use additional data at all. To understand the reason of our success over data-driven M1, one may observe the intermediate shapes in Fig. 1 whose uncolored parts behave completely differently, e.g., sudden jumps in the left arm of row 1. As a matter of fact, it is difficult to find global inseparable shapes that behave in tune at once. This motivates our part-based treatment whose tuning can be arranged more easily, especially for small size databases. A significantly larger database may upgrade the performance of M1 in the expense of increased computational time. We append a different SCAPE result to Fig. 6, this time in comparison with M1 and M5. We actually do not display any further M2 results as it is well known that linear interpolation between vertex positions fails to recover rotational motions.

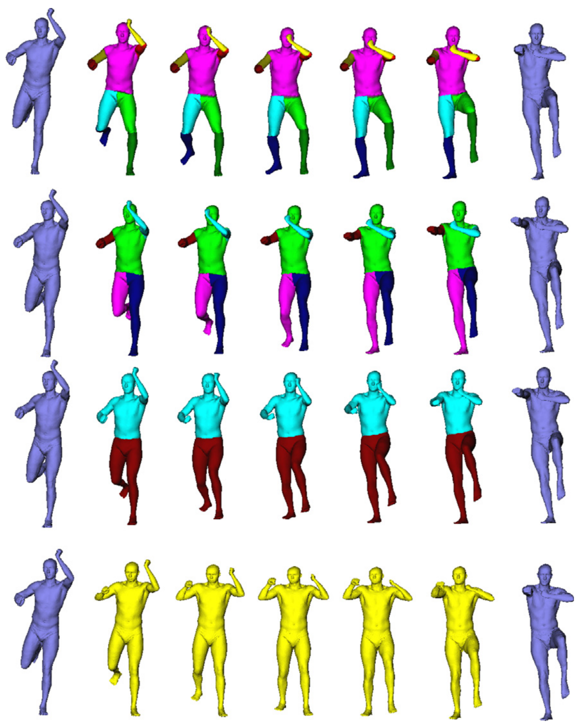
Fig. 7. Our results with changing number of parts (rows 1–3) in comparison with M1 (row 4).

We provide further results on SCAPE in Fig. 7. First three rows show our results with 9, 5, and 2 parts, respectively, followed by the results of M1 and M2. Our natural resulting poses can be compared with M1 in the last row. In M1, the middle figures show a model that has both feet in the air. Since the dataset only has models that have at least one foot on the ground, our part-based data-driven interpolation results on the other rows are more consistent with the models' physical properties. It can be observed that to switch feet, first it is necessary to put one foot on floor, then lift the other one.

Results of M4 and M5 on these pairs and other ones are compared with our method in Fig. 8. M4 is quite fast, e.g., taking 10 ms to generate a shape from the latent code, which is about 50 times faster than our 507.4 ms generation. We should, however, note the 30–45 min training time for each of their networks. M4 also deals with a significant number of parameters, 43 200 324 floats, whereas our method is training- and parameter-free.

M5, on the other hand, generally runs as accurate as our method but takes 2.5 times longer, 1.3 s per frame. It also fails when the largest rotation angle is a little larger than π since the quaternion interpolation for rotations cannot distinguish angles larger or smaller than π. Co-author of [50] kindly executed his program (M4), and we executed the public code of [20] (M5).

For comparison with M3, the first author of [40] kindly executed his program on some of our test data, namely SCAPE and Horse. M3 achieves significant improvement over [1], and consequently M2, by finding the optimized interpolation path in the global continuous space. While producing visually appealing results, this continuous space analysis causes redundant moves as demonstrated in Fig. 9 and in our accompanying video. Our discrete space analysis based method produces more preferable results in terms of natural movement. This redundancy issue is

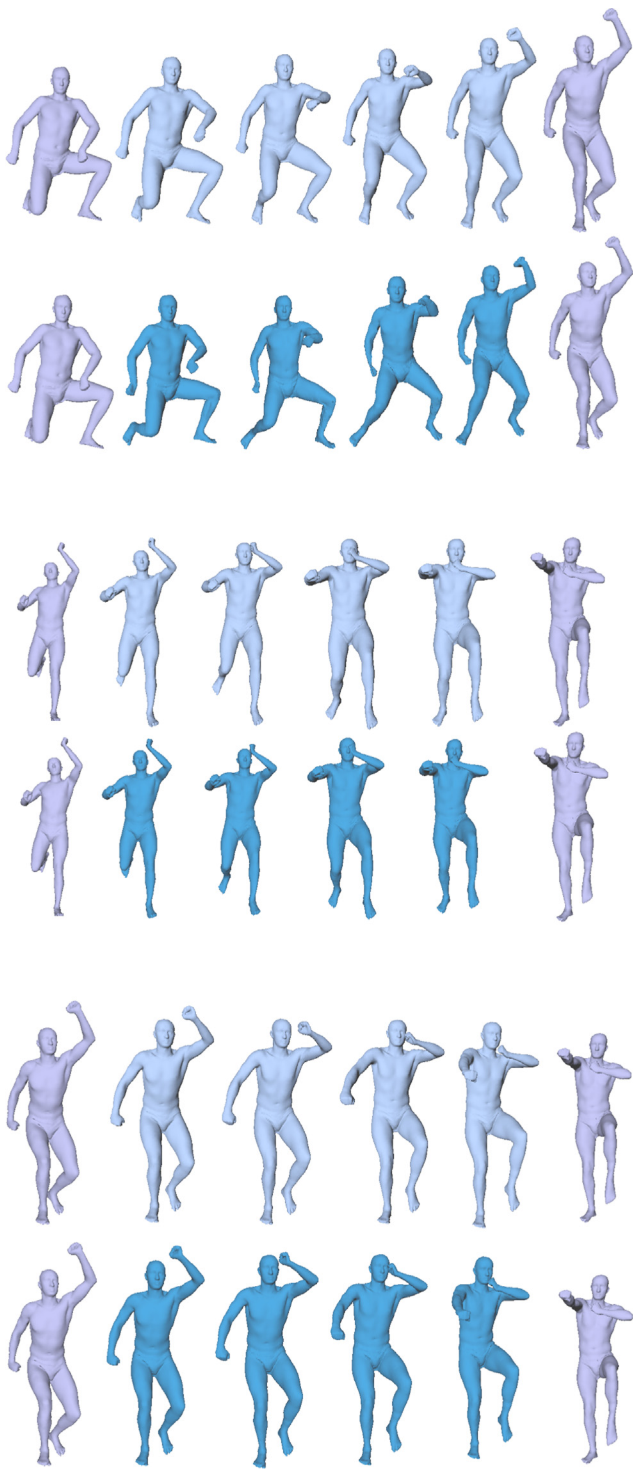

Fig. 8. Three different executions of M4 (top) and M5 (bottom). For our corresponding results, please refer to Fig. 12-row 1 at right, Figs. 7, 6-row 1, and our accompanying video.

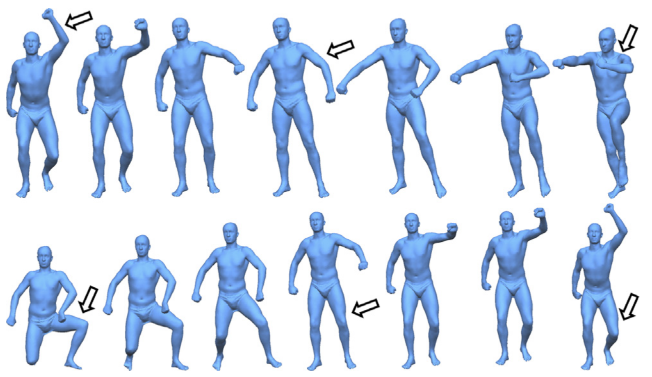

Fig. 9. Results of M3 are visually appealing and has no geometric artifacts. They are, however, likely to exhibit redundant moves, e.g., left hand coming down and up again, or left leg unbending and bending again. Our corresponding results with no such redundancies can be found at Fig. 6 - row 1 and Fig. 12 - row 1 at right as well as the accompanying video.

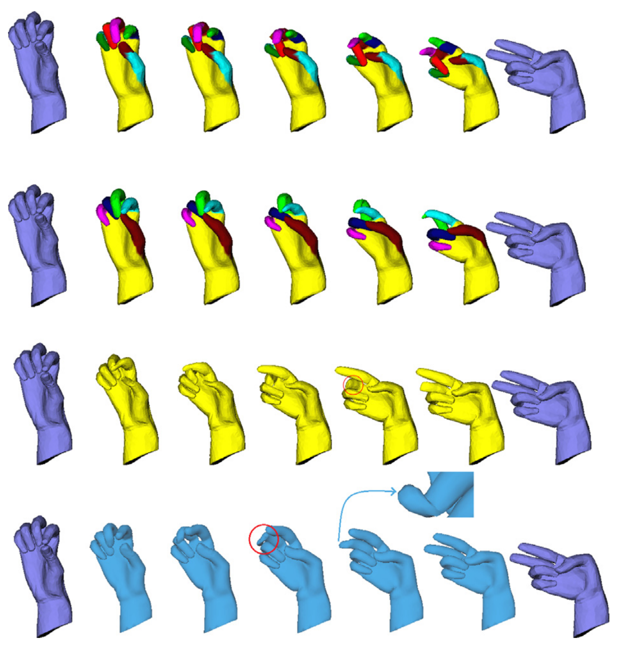

Fig. 10. Our results with changing number of parts (rows 1–2) in comparison with M1 (row 3) and M5 (row 4). Notice the highlighted artifacts of M1 and M5.

also reported in [45] which alleviates the problem using a deep neural net.

We created a new database of hands, another popular articulated object like humans. This Hand database is created by deforming a single hand model using the Blender software, preserving the fixed connectivity. The hand gestures are based on the images taken from the database for Hand Gesture Recognition (HGR) which contains Polish and American sign language gestures. Similar to Fig. 7, we show our results in Fig. 10 with varying number of parts. Another result set is depicted in Fig. 11 where we also demonstrate the selected intermediate shapes from the dataset in pink.

We finish our qualitative evaluations with four other articulated datasets: Dancer, Cat, Horse, and Centaur (Fig. 13 and Fig. 14), where visually appealing results are obtained.

5.2. Quantitative evaluations

We involve quantitative evaluations in terms of timing and accuracy.

The execution times of our method and the compared methods on a computer with an Intel Core i7 2.40 GHz CPU are shown in Table 1. Increasing the number of parts increases the amount of time to produce our interpolation results linearly. Note that for our method, Edge Weights computation is done only once

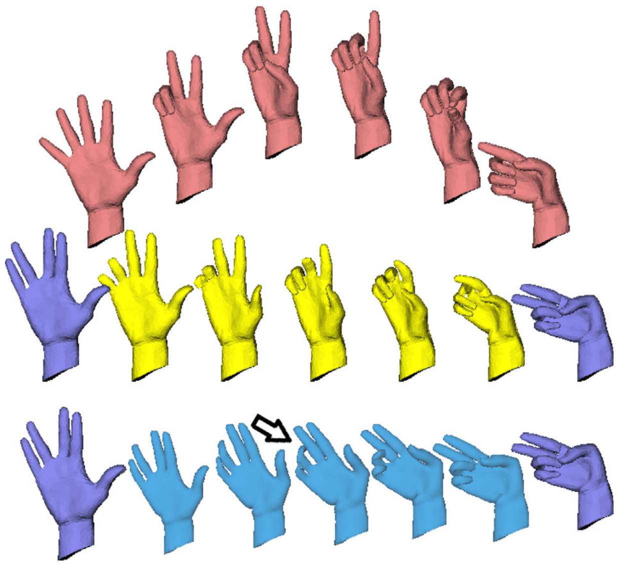
Fig. 11. Intermediate models in pink (top) guide the data-driven interpolation M1 (middle). As-rigid-as-possible morphing M5 does not require the pink models to operate (bottom).

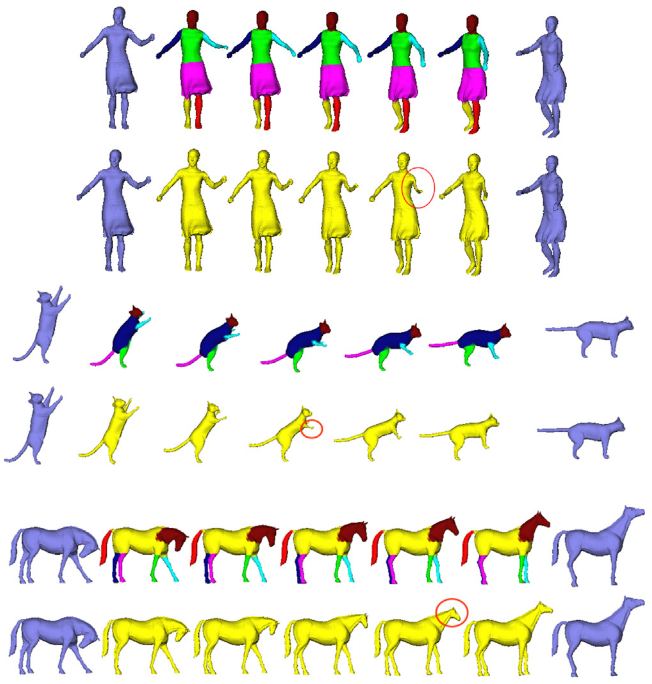
Fig. 13. For each of the three databases, our results (top) are shown in comparison with M1 (bottom). Notice volume loss as well as the highlighted artifacts in M1.

in an offline preprocessing stage and then all queries can be answered very quickly without taking that time into account. For M1, we also need the Edge Weights part only once just in the preprocessing stage.

Timings are consistent with the asymptotic time complexities as there are $O(m^2)$ model pairs to compute Edge Weights, where m is the number of models. Path Computations require Dijkstra processing over our graph structure in $O(m \log m)$ time. Finally, we compute positions of all n vertices over $O(m)$ models using the closed Bezier Computation in Eq. (3), hence $O(mn)$ more time. Total asymptotic time complexity is $O(m^2 n)$, which is the overall cost of the most dominant Edge Weights preprocessing action.

In order to numerically assess the geometric stability of our method, we have computed the volume change in the interpolated shapes. We also count the number of self-intersected elements to measure the deviation from physical plausibility.

For the volume change metric, we first tetrahedralize the source and target shapes as well as the interpolated shapes $\{I_1, I_2, .., I_m\}$ via [68] that is robust to geometric defects. We then compute the desired metric V via

$$\mathrm{vol}_i = \sum_{t \in I_i} |\mathrm{vol}(t)|, \qquad V = \frac{1}{a}\frac{1}{m}\sum_{i=1}^{m} |\mathrm{vol}^* - \mathrm{vol}_i| \tag{4}$$

where vol(t) measures the volume of tetrahedron t, and vol* is the optimal volume obtained by averaging the total volumes of the input source and target shapes. Division by average source tetrahedron volume a helps us interpret the results geometrically, i.e., V measures the change in volume in terms of a multiplicative factor of an average healthy tetrahedron volume, which is about a human nose volume. The comparative results are given in Table 2, which is complemented here with M5's information due to lack of space: $V = [10\ 18\ 10\ 6\ 14\ 19\ 8\ 10]^\top$ and $P = [0.5\ 1.4\ 0.4\ 0.2\ 0.6\ \ 1.0\ 0.4\ 0.5]^\top$.

For the counting metric, we count the number of triangles that intersect the mesh and then use the ratio between this number and the number of all triangles in percentage format P, e.g., 50 means half of the triangles are marked as self-intersecting. In a plausible motion where the shape does not touch itself, this value becomes 0. Most of the non-zero values in Table 2 can be explained by the presence of input models that are already self-intersecting, e.g., the crouching model around (0.23,0.02) in Fig. 2. Our interpolation scheme would want to reach these models in case they are the target shapes or the intermediate path shapes.

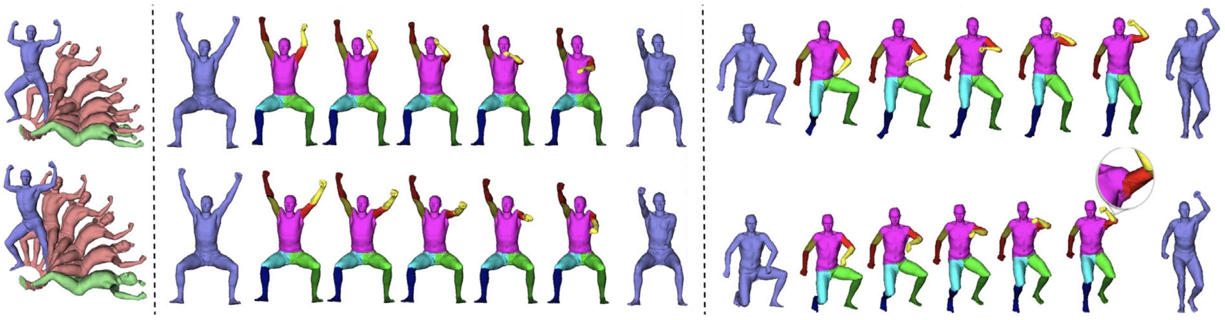
Fig. 12. Spherical linear interpolation (SLERP) of quaternions (left bottom) produces a rotation with more uniform velocity than LERP does (left top) and hence is preferred for this test. Note that, SLERP to the quaternions are applied over a Bezier curve, not separately. (Middle and right) Quaternion interpolation, however, is prone to errors at joints (bottom rows) when acting on multiple parts, a problem we avoid by our original algorithm (top rows). Average V and P values over 20 SCAPE runs are 53 and 0.8, respectively, which are worse than the corresponding entries of our original algorithm (Table 2).

Table 1
Statistics of the databases and average running times in milliseconds. Our solution on Dancer, for instance, is computed in 229 (computation of all edge weights via Eq. (2) — one-time preprocessing) + 44.9 (computation of the shortest path via Dijkstra's algorithm) + 17.3 (computation of vertex positions via closed-form Bezier) = 291.2 ms ≈ 0.29 s. SCAPE-A denotes SCAPE Augmented (Section 5.3).

Dataset	# Vertices	# Models	# Parts	Edge weights	Path computations	Bezier computations	M1	M2	M5
SCAPE	12 500	71	1	1049	16.1	3.7	19.8	1.1	1300
SCAPE	12 500	71	2	1862	33.9	3.9			
SCAPE	12 500	71	5	2108	155.2	19.1			
SCAPE	12 500	71	9	2126	470.7	36.7			
Hand	1515	31	1	31	3.7	0.3	4	0.2	149
Hand	1515	31	6	62	45.2	1.8			
Hand	1515	31	11	70	133.5	11.4			
Cat	7207	10	1	15	2.5	1.9	4.4	0.6	757
Cat	7207	10	5	31	8.8	3.3			
Horse	8431	11	1	23	0.8	0.6	1.4	0.3	890
Horse	8431	11	7	36	9.1	7.2			
Dancer	9971	27	1	122	3.1	1.8	4.9	0.4	1068
Dancer	9971	27	7	229	44.9	17.3			
Centaur	15 768	6	1	28	0.2	1.9	2.1	0.03	1700
Centaur	15 768	6	9	71	3.8	19.9			
FAUST	6890	100	1	1210	24.65	2.9	27.55	0.5	713
FAUST	6890	100	2	1982	50.88	3.3			
FAUST	6890	100	5	2094	210.31	22.7			
FAUST	6890	100	9	2116	755.3	48.7			
SCAPE-A	12 500	200	1	8687	144.3	5.5	149.8	1	1300
SCAPE-A	12 500	200	9	16 108	4867	119.5			

Table 2
Quantitative evaluation of our method in comparison with others. An entry of $V = 9$ means the shape loses about 9 human noses in volume on average over the computed sequence. Similarly, P gives the average percentage of triangles that intersect the mesh through the sequence. SCAPE-A denotes SCAPE Augmented (Section 5.3).

Dataset	V				P			
	Our	M1	M2	M3	Our	M1	M2	M3
SCAPE	9	114	248	5	0.4	0.9	0.9	0.4
Hand	17	125	260	n/a	1.2	1.3	2.5	n/a
Cat	10	122	249	n/a	0.5	0.9	2.1	n/a
Horse	7	82	167	3	0.2	0.6	0.8	0.2
Dancer	12	68	193	n/a	0.6	1.1	1.9	n/a
Centaur	21	187	297	n/a	1.3	2.2	2.9	n/a
FAUST	8	92	205	n/a	0.3	0.7	0.9	n/a
SCAPE-A	4	76	248	n/a	0.2	0.5	0.9	n/a

Please note that, our quantitative comparisons based on V in Table 2 are meaningful especially when the difference between the entries of the same row is high, e.g., 9 vs. 248, but not 9 vs. 5. This is due to the unit in use: the size of a nose which is a very small portion compared to the whole shape size. The main part that favors our method over geometrically-sound M3 is our more natural movement that avoids redundancies. This fact cannot be captured by Table 2, but it is rather evident in the accompanying video, Fig. 6 - row 1, and Fig. 12 - row 1 at right. We finally note that the quantitative results for a given dataset with varying number of parts are nearly identical. The ones reported in Table 2 correspond to 9-part SCAPE and FAUST, and 11-part Hand pairs. See Section 4 for the number of parts in other datasets.

5.3. Further findings

We finally provide three additional findings obtained in our method.

5.3.1. Form interpolation

All examples thus far demonstrate our capability to deform articulated poses from one to another, which is already a useful feature, e.g., for motion capture animations. Although there is theoretically no issue that prevents our method from working on more heterogeneous inputs, it is unlikely to find such sets in correspondence. Without correspondence, it would also be challenging or tedious to segment each model. This is indeed a limitation inherent to all data-driven methods, e.g., setting up the data to morph from a cat to a giraffe is not trivial for these methods.

We find a heterogeneous dataset that is in vertex-to-vertex correspondence, namely FAUST [67] which has 100 high-resolution scans of 10 human actors and a low-resolution (6890 vertices) template model registered to each of these scans. We are then able to verify empirically that our algorithm morphs not only poses but also forms (turning from one person to another) as shown in Table 2, Fig. 15, and video. We also verify that our method is not very sensitive to the number of parts in use (Fig. 16).

5.3.2. Data augmentation

We manage to obtain high utilization from small databases thanks to our part-based treatment. This high utilization already gives us stable interpolation results. We, however, can improve our results even further by populating the small databases with the results of the original algorithm, hence in a fully-automatic and fast manner. With the augmented database, part-based solution works in an even more flexible way because higher number of parts made available provides more alternatives and flexibility for path creation, which in turn brings more accuracy (Table 2, Fig. 17, Fig. 18, and video). Note that, this augmentation is achieved due to the capability of our original algorithm to work on very small databases.

To perform the population for augmentation, we run our part-based data-driven interpolation algorithm over 100 random pairs and sample 10 new meshes from each computed interpolation sequence. We only admit a subset of these mesh samples to the augmented database based on our volume loss criterion V in Section 5.2. Namely, meshes with the smallest values are chosen.

5.3.3. Quaternions vs. Bezier paths

Instead of creating a Bezier path for each vertex, we interpolate the optimal rotation of each part. To this end, we convert the optimal rotation matrices into quaternions and perform spherical

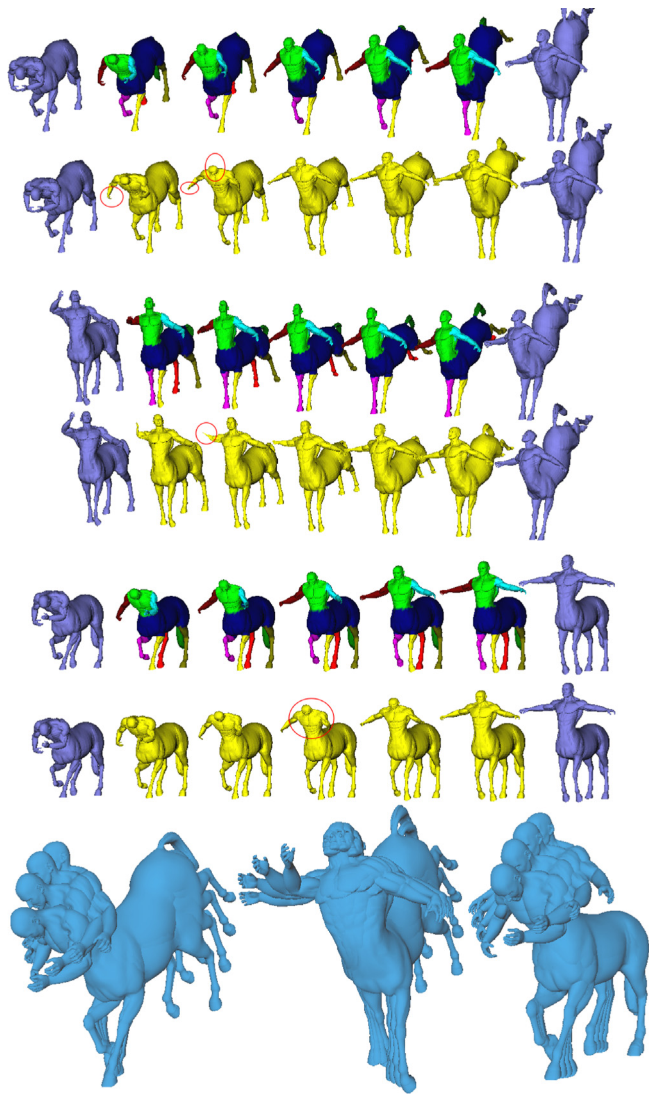

Fig. 14. Results on the Centaur dataset for three different shape pairs. Our result is followed by M1 (notice the highlighted artifacts) in the first 6 rows. The last row is by M5, which is as accurate as our interpolation but took 1.7 s for frame generation, much slower than our 23.7 ms.

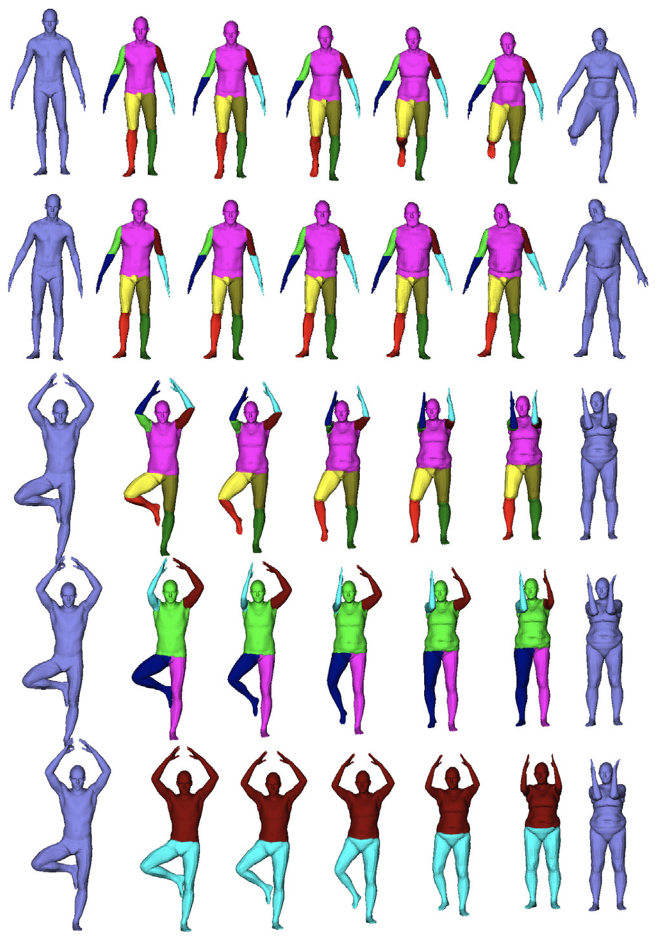

Fig. 15. Three results depicting morphs of poses and forms. We do the last pair (3rd row) with changing number of parts to demonstrate the insensitivity of our method to this parameter.

linear interpolation (Fig. 12 — left bottom). Although we could get valid results by first interpolating quaternion q_{i-1} and q_i and then separately q_i and q_{i+1}, we would then end up with a non-differentiable curve with potential smoothness problems. To address this issue, we build a Bezier curve of quaternions as described in [69,70]. This alternative approach is, however, still outperformed by our original vertex position interpolation proposal (Fig. 12 — middle and right) because the rigid parts obtained from different models may possess slight differences, noticeable especially around the joints — note that quaternion solution is only valid when the parts are rigidly segmented. This means that the parts will not be matched perfectly after we apply a single rotation followed by the same amount of smoothing. Consequently, our current vertex-based movement performs better from this groupwise movement.

6. Limitations

Although we rarely experience during our tests, our method may fail in terms of geometric stability and physical plausibility. The former is possible especially when the database is so sparse that we cannot find sufficiently similar part poses, a problem alleviated by our automatic data augmentation (Section 5.3.2). The other alternative to reduce such geometric artifacts is interpolating the rotation of each rigid segment instead of interpolating vertices one-by-one via the Bezier basis (Section 5.3.3). This alternative, however, did not turn out to be sufficiently effective because treating all vertices within the rigid part the same is problematic, considering that the same part on different models may have slight differences. We show results where geometric stability problem arises in the form of collapses (Fig. 17) and interpenetration (Fig. 18), which are both fixed by applying our automatic augmentation (Section 5.3.2) as demonstrated within the same figures. The physical plausibility problem, on the other hand, exists as the parts are currently unaware of each other, e.g., because the forearm does not know about the head, it may pass through it, similar to the case in (Fig. 18). Note that, barrier- or projection-based solutions in non-data-driven methods [17, 24] may alleviate this self-intersection problem. These computationally expensive solutions can, however, be replaced with less costly data-driven solutions such as establishing awareness between parts, which we intend to do in future. Finally, our method is heavily relied on segmentation results, which is currently not an issue at all as we transfer a single manual segmentation to all other database model consistently through existing correspondence information. For a fully automatic pipeline, correspondences must be computed accurately in order to obtain

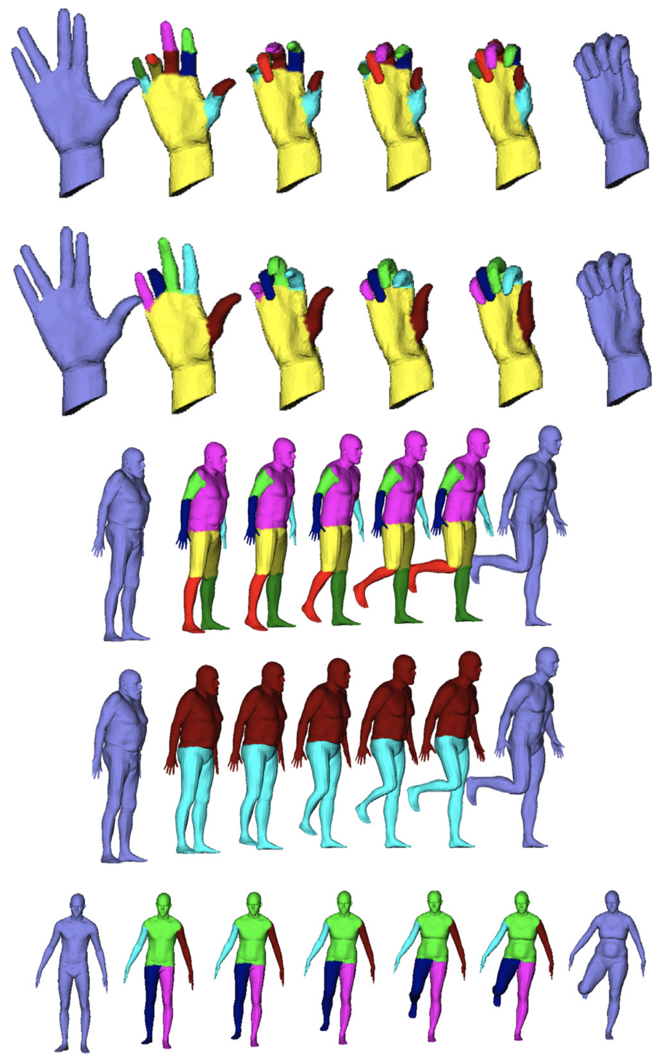
Fig. 16. Our interpolation results do not change significantly as the number of parts varies. 9-part version of the last row can be found in Fig. 15-row 1.

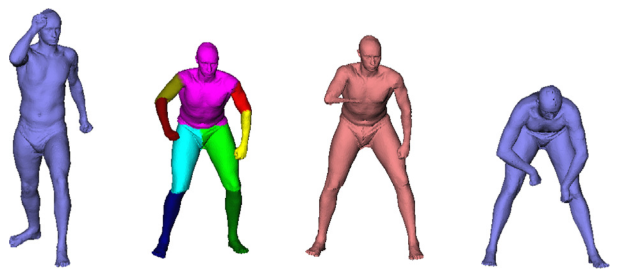
Fig. 17. Part-based data-driven interpolations obtained at $\alpha = 0.5$ using the original (pink) and augmented (multicolor) SCAPE. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the required consistent segmentation throughout the database. Alternatively, individual segmentations that are consistent over the database must be computed.

7. Conclusion and future work

Interpolating parts is shown to be a good idea to cover the shape space more densely, which in turn produces promising results with even small databases. Thanks to our part-based solution that highly exploits the information in the relatively small databases, we obtain plausible smooth interpolations between the source and target shape pairs in a data-driven fashion. Our unified merging scheme enables the gathering of different part-level interpolations into one consistent global interpolation. Our algorithm is also quite fast and reproducible since it consists mainly of easy-to-implement blocks such as Dijkstra, Taubin, and De Casteljau methods. Extensive evaluation involving figures, comparisons, accompanying video, and quantification metrics validated the algorithm for pose interpolations as well as form interpolations. Original algorithm is also proved useful in populating the small database it is run on, effectively improving the performance of the new runs on such augmented databases.

An interesting future work is to establish awareness between the parts so as to improve plausibility. We also plan to evaluate the performance of our form morphing capability further. Our current experimentation is not as comprehensive as our pose morphing experiments due to lack of correspondence-equipped heterogeneous databases for this task. We may consider creating our own using an automatic correspondence computation method for such databases.

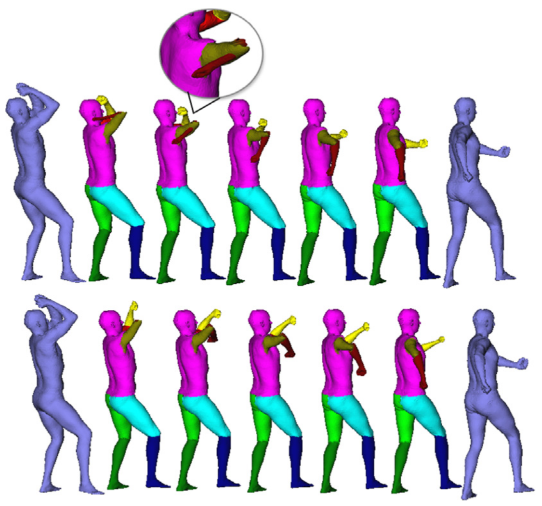
Fig. 18. Interpenetration issue observed in our result (top) is fixed when the same method runs over the augmented database (bottom).

Declaration of competing interest

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

Acknowledgments

We thank Lin Gao and Jie Yang for the help with their papers' [40,50] experiments, and Javier Dehesa (javidcf) for [20]. We also thank the anonymous reviewers for their valuable comments. This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the project EEEAG-115E471.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2021.103027.

References

[1] Gao L, Lai Y-K, Huang Q, Hu S-M. A data-driven approach to realistic shape morphing. Comput Graph Forum 2013;32:449–57.
[2] Sahillioğlu Y, Aydınlılar M. Shape interpolation via multiple curves. In: Proceedings of the 26th Pacific conference on computer graphics and applications: posters. Eurographics Association; 2018, p. 9–10.
[3] Alexa M. Recent advances in mesh morphing. Comput Graph Forum 2002;21:173–98.
[4] Bronstein AM, Bronstein MM, Kimmel R. Numerical geometry of non-rigid shapes. Springer Science; 2008.
[5] Man SY, Wenping W, Francis C. Interpolating polyhedral models using intrinsic shape parameters. J Vis Comput Anim 1997;8(2):81–96.
[6] Kircher S, Garland M. Free-form motion processing. ACM Trans Graph 2008;27(2):12.
[7] Lipman Y, Sorkine O, Levin D, Cohen-Or D. Linear rotation-invariant coordinates for meshes. ACM Trans Graph 2005;24:479–87.
[8] Baek S-Y, Lim J, Lee K. Isometric shape interpolation. Comput Graph 2015;46:257–63.
[9] Kilian M, Mitra NJ, Pottmann H. Geometric modeling in shape space. ACM Trans Graph 2007;26(3).
[10] Chao I, Pinkall U, Sanan P, Schröder P. A simple geometric model for elastic deformations. In: A simple geometric model for elastic deformations. 2010, p. 38:1–6.
[11] Martin S, Thomaszewski B, Grinspun E, Gross M. Example-based elastic materials. ACM Trans Graph 2011;30(4):72:1–8.
[12] Chu H, Lee TY. Multiresolution mean shift clustering algorithm for shape interpolation. IEEE Trans Vis Comput Graphics 2009;15(5):853–66.
[13] Zhang S, Li G, Lu H, Ouyang Y, Yin M, Xian C. Fast as-isometric-as-possible shape interpolation. Comput Graph 2015;244–56.
[14] von Radziewsky P, Eisemann E, Seidel H, Hildebrandt k. Optimized subspaces for deformation-based modeling and shape interpolation. Comput Graph 2016;58:128–38.
[15] Fröhlich S, Botsch M. Example-driven deformations based on discrete shells. Comput Graph Forum 2011;30:2246–57.
[16] Winkler T, Drieseberg J, Alexa M, Hormann K. Multi-scale geometry interpolation. Comput Graph Forum 2010;29(2):309–18.
[17] Von-Tycowicz C, Schulz C, Seidel H-P, Hildebrandt K. Real-time nonlinear shape interpolation. ACM Trans Graph 2015;34(3):34:1–34:10.
[18] Huang Q, Adams B, Wicke M, Guibas L. Non-rigid registration under isometric deformations. Comput Graph Forum 2008;1449–57.
[19] Levi Z, Gotsman C. Smooth rotation enhanced as-rigid-as-possible mesh animation. IEEE Trans Vis Comput Graph 2015;21(2):264–77.
[20] Liu Y-S, Yan H-B, Martin RR. As-rigid-as-possible surface morphing. J Comput Sci Tech 2011;26(3):548–57.
[21] Alhashim I, Li H, Xu K, Cao J, Ma R, Zhang H. Topology-varying 3D shape creation via structural blending. ACM Trans Graph 2014;33(4):158.
[22] Wirth B, Bar L, Rumpf M, Sapiro G. Geodesics in shape space via variational time discretization. In: International workshop on energy minimization methods in computer vision and pattern recognition. Springer; 2009, p. 288–302.
[23] Wirth B, Bar L, Rumpf M, Sapiro G. A continuum mechanical approach to geodesics in shape space. Int J Comput Vis 2011;93(3):293–318.
[24] Heeren B, Rumpf M, Wardetzky M, Wirth B. Time-discrete geodesics in the space of shells. Comput Graph Forum 2012;31(5):1755–64.
[25] Heeren B, Rumpf M, Schroder P, Wardetzky M, Wirth B. Splines in the space of shells. Comput Graph Forum 2016;35(5):111–20.
[26] Xia Q, Chen C, Liu J, Li S, Hao A, Qin H. Efficient 4D shape completion from sparse samples via cubic spline fitting in linear rotation-invariant space. Comput Graph 2019;82:129–39.
[27] Brandt C, von Tycowicz C, Hildebrandt K. Geometric flows of curves in shape space for processing motion of deformable objects. Comput Graph Forum 2016;35(2):295–305.
[28] Sederberg T, Gao P, Wang G, Mu H. 2-D shape blending: an intrinsic solution to the vertex path problem. In: SIGGRAPH. 1993.
[29] 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. 2000, p. 157–64.
[30] Baxter W, Barla P, Anjyo K. Compatible embedding for 2d shape animation. IEEE Trans Vis Comput Graphics 2009;15(5):867–79.
[31] Chien E, Chen R, Weber O. Bounded distortion harmonic shape interpolation. ACM Trans Graph 2016;35(4):105:1–105:15.
[32] Chen R, Weber O, Keren D, Ben-Chen M. Planar shape interpolation with bounded distortion. ACM Trans Graph 2013;32(4):108.
[33] Zhu Y, Popović J, Bridson R, Kaufman DM. Planar interpolation with extreme deformation, topology change and dynamics. ACM Trans Graph 2017;36(6):213.
[34] Gao L, Lai Y-K, Liang D, Chen S-Y, Xia S. Efficient and flexible deformation representation for data-driven surface modeling. ACM Trans Graph 2016;35(5):158.
[35] Wampler K. Fast and reliable example-based mesh ik for stylized deformations. Trans Graph 2016;35(6):235.
[36] Dvoroznak M, Benard P, Barla P, Wang O, Sykora D. Example-based expressive animation of 2D rigid bodies. ACM Trans Graph 2017;36(4):127.
[37] Jones B, Thuerey N, Shinar T, Bargteil AW. Example-based plastic deformation of rigid bodies. ACM Trans Graph 2016;35(4):34.
[38] Sloan P, Rose C, Cohen M. Shape by example. In: Symp. on Interactive 3D graphics, vol. 33, no. 4. 2001, p. 135–143.
[39] Marras S, Cashman T, Hormann K. Efficient interpolation of articulated shapes using mixed shape spaces. Comput Graph Forum 2013;8:258–70.
[40] Gao L, Chen S-Y, Lai Y-K, Xia S. Data-driven shape interpolation and morphing editing. Comput Graph Forum 2016;36(8):19–31.
[41] Averbuch-Elor H, Cohen-Or D, Kopf J. Smooth image sequences for data-driven morphing. Comput Graph Forum 2016;35(2):203–13.
[42] Sahillioğlu Y. A shape deformation algorithm for constrained multidimensional scaling. Comput Graph 2015;53:156–65.
[43] Sahillioğlu Y, Kavan L. Detail-preserving mesh unfolding for nonrigid shape retrieval. ACM Trans Graph 2016;35(3):27.
[44] Sahillioğlu Y, Sezgin M. Sketch-based articulated 3D shape retrieval. IEEE Comput Graph Appl 2017;37(6):88–101.
[45] Tan Q, Gao L, Lai Y, Xia S. Variational autoencoders for deforming 3d mesh models. IEEE Comput Vis Pattern Recogn 2018;5841–50.
[46] Huang R, Rakotosaona M-J, Achlioptas P, Guibas LJ, Ovsjanikov M. OperatorNet: Recovering 3D shapes from difference operators. in: Proceedings of the IEEE international conference on computer vision. 2019, p. 8588–97.
[47] Mo K, Guerrero P, Yi L, Su H, Wonka P, Mitra N, Guibas L. Structurenet: Hierarchical graph networks for 3D shape generation. ACM Trans Graph 2019;38(6):242.
[48] Gao L, Yang J, Wu T, Yuan Y-J, Fu H, Lai Y-K, Zhang H. SDM-NET: Deep generative network for structured deformable mesh. ACM Trans Graph 2019;38(6):1–15.
[49] Dubrovina A, Xia F, Achlioptas P, Shalah M, Groscot R, Guibas LJ. Composite shape modeling via latent space factorization. In: Proceedings of the IEEE international conference on computer vision. 2019, p. 8140–49.
[50] Gao L, Yang J, Qiao Y-L, Lai Y-K, Rosin PL, Xu W, Xia S. Automatic unpaired shape deformation transfer. ACM Trans Graph 2018;37(6):1–15.
[51] Sahillioğlu Y. Recent advances in shape correspondence. Vis Comput 2020;36:1705–21.
[52] Guo J, Wang H, Cheng Z, Zhang X, Yan D-M. Learning local shape descriptors for computing non-rigid dense correspondence. Comput Vis Media 2020;6(1):95–112.
[53] Donati N, Sharma A, Ovsjanikov M. Deep geometric functional maps: robust feature learning for shape correspondence. in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 8592–601.
[54] Nguyen A, Ben-Chen M, Welnicka K, Ye Y, Guibas L. An optimization approach to improving collections of shape maps. Comput Graph Forum 2011;30(5):1481–91.
[55] Sahillioğlu Y, Yemez Y. Multiple shape correspondence by dynamic programming. Comp Gra Forum 2014;33(7):121–30.
[56] Sahillioğlu Y, Yemez Y. Minimum-distortion isometric shape correspondence using EM algorithm. IEEE Trans Pattern Anal Mach Intell 2012;34(11):2203–15.
[57] Horn B. Closed-form solution of absolute orientation using unit quaternions. J Opt Soc Amer A 1987;4(4):629–42.
[58] Osada R, Funkhouser T, Chazelle B, Dobkin D. Shape distributions. ACM Trans Graph 2002;21(4):807–32.
[59] Wuhrer S, Brunton A. Segmenting animated objects into near-rigid components. Vis Comput 2010;26(2):147–55.
[60] Marras S, Bronstein MM, Hormann K, Scateni R, Scopigno R. Motion-based mesh segmentation using augmented silhouettes. Graph Models 2012;74(4):164–72.
[61] Rodrigues RS, Morgado JF, Gomes AJ. Part-based mesh segmentation: a survey. In: Computer graphics forum, vol. 37. (6):Wiley Online Library; 2018, p. 235–74.
[62] Shu Z, Shen X, Xin S, Chang Q, Feng J, Kavan L, Liu L. Scribble based 3D shape segmentation via weakly-supervised learning. IEEE Trans Vis Comput Graph 2019.
[63] Taubin G. A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. ACM; 1995, p. 351–8.
[64] Anguelov D, Srinivasan P, Koller D, Thrun S, Rodgers J, Davis J. SCAPE: shape completion and animation of people. ACM Trans Graph 2005;24:408–16.
[65] Vlasic D, Baran I, Matusik W, Popović J. Articulated mesh animation from multi-view silhouettes. ACM Trans Graph 2008;27(3):97.

[66] Sumner RW, Popović J. Deformation transfer for triangle meshes. ACM Trans Graph 2004;23(3):399–405.

[67] Bogo F, Romero J, Loper M, Black M. FAUST: Dataset and evaluation for 3D mesh registration. In: IEEE conf. on computer vision and pattern recognition. 2014, p. 3794–3801.

[68] Barill G, Dickson N, Schmidt R, Levin DI, Jacobson A. Fast winding numbers for soups and clouds. Trans Graph 2018;37(4):43.

[69] Shoemake K. Animating rotation with quaternion curves. In: Proceedings of the 12th annual conference on computer graphics and interactive techniques. 1985, p. 245–54.

[70] Kremer VE. Quaternions and SLERP. 2008, Embots. dfki. de/doc/seminar_ca/Kremer_Quaternions. pdf.

Melike Aydınlılar received her BS and MS degrees from the Computer Engineering department of Middle East Technical University, Turkey. She is now pursuing a computer graphics PhD in INRIA, France.

Yusuf Sahillioğlu is an associate professor in the Department of Computer Engineering at Middle East Technical University. He is also an associate editor of The Visual Computer journal. His research interests include digital geometry processing and computer graphics. He has a PhD in computer science from Koç University, Turkey. Contact him at ys@ceng.metu.edu.tr or visit http://www.ceng.metu.edu.tr/~ys.