PART-BASED DATA-DRIVEN SHAPE INTERPOLATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELİKE AYDINLILAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2018

Approval of the thesis:

PART-BASED DATA-DRIVEN SHAPE INTERPOLATION

submitted by **Melike Aydınlılar** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Yusuf Sahillioğlu _____
Supervisor, **Computer Engineering, METU**

Examining Committee Members:

Prof. Dr. Tolga Can
Computer Engineering, METU _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering, METU _____

Prof. Dr. Uğur Güdükbay
Computer Engineering, Bilkent University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Melike Aydınlılar

Signature :

ABSTRACT

PART-BASED DATA-DRIVEN SHAPE INTERPOLATION

Aydınlılar, Melike
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Yusuf Sahillioğlu

July 2018, 38 pages

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 rigid parts. In particular, we interpolate rigid 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 demonstrate visually appealing and natural shape interpolation results in comparison with two other techniques. As a side contribution, we provide a public articulated hand dataset with fixed connectivity, which can be used in the evaluation of other shape interpolation methods. Our code and executables are also publicly available.

Keywords: Shape Interpolation, Data-driven, Shape Deformation, Bezier Curve.

ÖZ

PARÇA BAZLI VERİ GÜDÜMLÜ ŞEKİL İNTERPOLASYONU

Aydınlılar, Melike
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Yusuf Sahillioğlu

Temmuz 2018 , 38 sayfa

Şekil interpolasyonu, dijital geometri işleme alanında aktif bir problemdir. Verilen başlangıç ve sonuç şekilleri arasında birbiri ardına gelen kesintisiz ara şekiller üretmeyi amaçlar. Başlangıç ve sonuç modellerini izole bir şekilde interpole eden geleneksel yöntemlerin aksine, yeni veri güdümlü yaklaşımlar bir veri seti içinde bulunan ara şekilleri kullanan interpolasyon yöntemlerini ortaya koymuştur. Bu yöntemler veri setinin sağlanmasına ihtiyaç duysa da daha iyi performans sağlarlar. Şekilleri parçalanamaz bütünler olarak değerlendiren hali hazırdaki veri güdümlü yaklaşımların aksine bizim sunduğumuz yeni yöntemde şekiller ayrılabilen rijid parçalar olarak değerlendirilmektedir. Bu şekilde, rijid parçalar veri seti içindeki farklı ara şekiller üzerinden interpole edilip, daha sonra birleştirilerek şekilleri bütün halinde değerlendirerek, belirli sayıda ara şekil kullanan mevcut yöntemlerden daha fazla esneklik ve çeşitlilik sağlamaktadır. Farklı ara şekil setleri üzerinde tutarlı bir şekilde çalışabilmek amacıyla, parametrik eğrilere dayalı birleşik bir yapı sunulmuştur. Yan katkı olarak artiküle el veritabanımızı sağlıyoruz. Kodumuz ve programımız da herkese açıktır.

Anahtar Kelimeler: Şekil İnterpolasyonu, Veri-güdümlü, Şekil Deformasyonu, Bezier Eğrisi.

To my family

ACKNOWLEDGMENTS

I would like to thank my advisor Assoc. Prof. Dr. Yusuf Sahillioğlu for the guidance and support. I also would like to thank the committee members for their participation. I would like to thank my sister and my parents for their support.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiii

LIST OF FIGURES xiv

CHAPTERS

1 INTRODUCTION 1

1.1 Scope 1

1.2 Overview 1

1.3 Contributions 2

1.4 Organization 3

2 RELATED WORK AND BACKGROUND 5

2.1 Shapes 5

2.1.1 Rigid and Non-rigid Shapes 5

2.2 Shape Interpolation 5

2.2.1 Geometry Based Methods 6

2.2.2 Data-Driven Methods 7

3 DATA-DRIVEN SHAPE INTERPOLATION 11
3.1 The Algorithm 11
3.1.1 Input and Output 11
3.1.2 Graph Structure 12
3.1.2.1 Edge Weights 12
3.1.3 Multi-Dimensional Scaling 13
3.2 Path Interpolation 13
3.2.1 Shortest Path Algorithm 14
3.2.2 Bezier Curves 15
3.3 Rigid Parts 16
3.3.1 Unified Merging 17
4 EXPERIMENTS 19
4.1 Datasets 19
4.1.1 SCAPE Dataset 19
4.1.2 Hand Dataset 20
4.1.3 Cat and Horse Datasets 22
4.1.4 Dancer Dataset 23
4.2 Results 25
4.3 Limitations 31
5 CONCLUSION AND FUTURE WORK 33
REFERENCES 35

LIST OF TABLES

TABLES

Table 4.1 Statistics of the datasets and average running times in milliseconds. . 31

LIST OF FIGURES

FIGURES

Figure 1.1 Comparison of simple linear interpolation results using Eq. 1.1 (bottom row) with a sophisticated method [1] (top row) (Image taken from [1]). 2

Figure 2.1 Comparison of data driven and geometry-based methods. First row shows the data driven results of [2], second and third rows show the results of [3] and [4]. In the second and the third rows, even using sophisticated geometric methods, unnatural poses have occured (Image taken from [2]). 9

Figure 3.1 SCAPE models are arranged in 2D with separations proportional to their dissimilarities captured by Eq. 3.1. 14

Figure 3.2 Multi Dimensional Scaling. The colored models show the shape sequence for the forth row of Figure 4.10. 15

Figure 3.3 Visualization of a cubic Bezier curve. 16

Figure 3.4 Representative models of databases. Parts highlighted. 17

Figure 3.5 Using Bezier curves for multiple paths (3 in this case) with different lengths allows unified timing and smooth transition. 18

Figure 4.1 MDS of SCAPE Dataset 20

Figure 4.2 MDS of Hand Dataset 21

Figure 4.3 Models in pink (top) guide the data-driven interpolation (middle) which performs better than simple linear interpolation (bottom). 21

Figure 4.4 MDS of Cat Dataset 22

Figure 4.5 MDS of Horse Dataset 23

Figure 4.6 MDS of Dancer Dataset as is. 23

Figure 4.7 MDS of Dancer Dataset after centering the models 24

Figure 4.8 MDS of Dancer Dataset after sampling centered models. 25

Figure 4.9 Three interpolations of three different parts (rows 1-3) for our algorithm whose final result (row 4) outperforms the results of M1 (row 5) and M2 (row 6). Note zoomed artifacts of M1 & M2. 26

Figure 4.10 Our results with changing number of parts (rows 1-3) in comparison with M1 (row 4) and M2 (row 5). 28

Figure 4.11 Our results with changing number of parts (rows 1-2) in comparison with M1 (row 3) and M2 (row 4). 29

Figure 4.12 Our results on the Hand and SCAPE datasets. 29

Figure 4.13 For each of the three datasets, our results (top) are shown in comparison with M1 (bottom). Note the volume loss in M1. 30

CHAPTER 1

INTRODUCTION

1.1 Scope

The goal of this work is to provide a method to interpolate two corresponding shapes which is both geometrically stable and as natural as possible. To achieve this, we explore the dataset these shapes belong to and get insights about the nature of these shapes.

1.2 Overview

Shape interpolation is an important problem in computer graphics. It has many application areas ranging from key frame animation to content creation. In 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:

$$v_{\text{result}}^k = \alpha v_2^k + (1-\alpha)v_1^k, \quad 0 \leq \alpha \leq 1, \quad 1 \leq k \leq N, \alpha \in \mathbb{R}, k \in \mathbb{N} \tag{1.1}$$

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

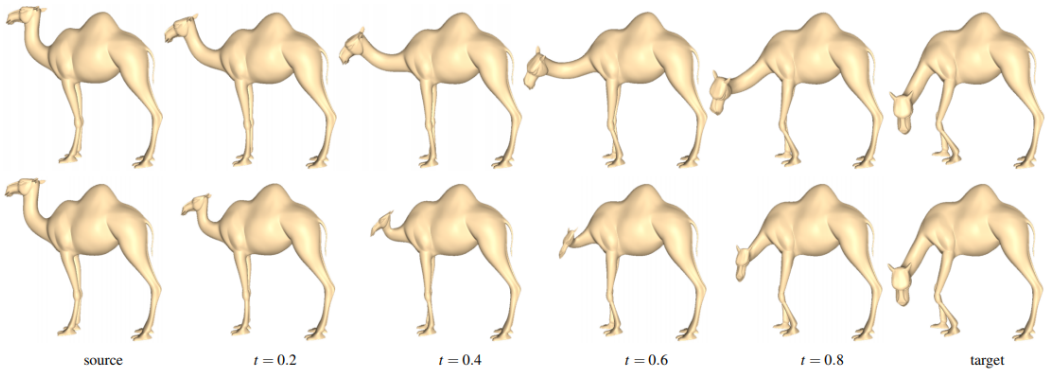

Figure 1.1: Comparison of simple linear interpolation results using Eq. 1.1 (bottom row) with a sophisticated method [1] (top row) (Image taken from [1]).

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 case where a sophisticated variant is satisfactory is demonstrated in Figure 1.1.

A data-driven approach, [2] for instance, has the potential to solve this problem by inserting intermediate database shapes between two dissimilar input shapes. In particular, it uses a set of smooth transitions between *similar* shapes that connect the source input to the target input which are *dissimilar* in the first place. The main drawback of this 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.

1.3 Contributions

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 rigid parts, not over themselves. The immediate advantage of this idea is best shown with an example: the intermediate shapes in the second row of the Figure 4.9 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 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 (row 4). 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 (row 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. 1.1.

With this work, we provide an improvement to data-driven shape interpolation approach by transforming shapes in parts rather than as a whole. To enable this novel approach, we established a unified framework of parametric Bezier curves defined over different sets of intermediate database shapes.

1.4 Organization

Over the proceeding sections, firstly, we provide an overview of previous approaches for the interpolation problem, divided into two methods, that are geometry based and data-driven. Then, we explain our part-based method in more detail, explaining the graph structure and path finding also the parametric curves that our method is based on. We continue by giving the experimental results over four different 3D datasets and run time statistics.

CHAPTER 2

RELATED WORK AND BACKGROUND

2.1 Shapes

In this work, we use 3D shapes discretized as triangle meshes. Each mesh is represented as a set of vertices that are points in 3D space and a set of edges connecting the vertices. These edges then, form triangles that represent the mesh surface.

2.1.1 Rigid and Non-rigid Shapes

Rigid shapes are exposed to rigid transformations, that are translation, rotation and reflection. These transformation do not change the intrinsic properties of a shape.

Non-rigid shapes, on the other hand, are exposed to non-rigid transformations such as bending, folding, stretching and scaling as well as rigid transformation.

2.2 Shape Interpolation

We investigate the literature that interpolate the models between a given source-target shape pair 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 [5]. The latter is a relatively new category.

2.2.1 Geometry Based Methods

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 [6]. To overcome this problem, [7] attempts to extend the intrinsic shape parameters for interpolation to 3D shapes using a local greedy approach. Alexa [8] and Lipman et al. [1] propose differential and rotation invariant coordinates, respectively, that can be used for relatively robust interpolations. Xu et al. [4] use a Poisson based approach. 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.

Instead of performing linear interpolation on absolute coordinates, [13] and [14] 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. [15] proposes an interpolation method that accelerates non-linear solutions to real-time rates. An alternating optimization technique by Huang et al. [3] use the as-rigid-as-possible deformation framework that keeps the local transformations close to a rigid motion. Defining blending operations on a spatio-structural graph composed of medial curves and sheets made 3D interpolation under topology variations possible [16].

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. [17], which interpolates the intrinsic definitions of the initial and final 2D shapes. Alexa's method [18] generates as-rigid-as possible interpolation for 2D images. Extending this idea to 3D models requires consistent tetrahedralization which is not trivial to obtain. Chien et al. [19] propose a fast and theoratically sound interpolation method that blends planar harmonic mappings represented in closed-form. Previous version of this algorithm [20] 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 [21] shows promising results under extreme deformation and topology change.

2.2.2 Data-Driven Methods

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. The first example to this paradigm dates back to 2001 [22] in an interactive framework and recently we see variations that are much more closer to our process [2, 23]. Specifically, [2] creates a model subspace from a given database and explore this using a shortest path approach. [23] differs in the way it interpolates the consecutive shapes in the shortest path. While [2] imports the interpolation procedure of [3] for this purpose, [23] uses rotation-invariant coordinates [1] 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. [24] 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 [25, 26] or sketch-based [27] shape retrieval techniques on large diverse databases such as SketchUp 3D Warehouse.

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 dense correspondence algorithms [28, 29]. In practice, however, artists deform a template mesh in its rest pose to different poses [30], 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 [31, 32].

An example of the difference between the behavior of the geometry based and data-driven methods is given in Figure 2.1. Even the results are geometrically sound (no volume loss etc.), they fail to account for the nature of the movement of the model, as they only have information on the source and the target models. If given access to the rest of the dataset, more accurate and realistic poses can be generated.

Figure 2.1: Comparison of data driven and geometry-based methods. First row shows the data driven results of [2], second and third rows show the results of [3] and [4]. In the second and the third rows, even using sophisticated geometric methods, unnatural poses have occured (Image taken from [2]).

CHAPTER 3

DATA-DRIVEN SHAPE INTERPOLATION

3.1 The Algorithm

An overview of our algorithm can be captured with the example given in the end of the Chapter 1. As explained, the main idea of the algorithm based on data-driven methods, is to create a full graph using every element of the dataset. Then, using a pairwise distance measure to calculate the edge weights. Having a graph with edge weights proportional to the pairwise similarity between the shapes, we can create an interpolation scheme basically traversing the graph. The shapes that are in the calculated path between an source and an end shape provide the intermediate shapes to be used in our Bezier curve based scheme.

We then, introduce rigid parts into the scheme. After dividing a representative model into given number of rigid parts, we consider them as separate entities and we run the algorithm on them to merge them later. This way, we increase the expressiveness of the given dataset and increase quality of the interpolation. The details of the algorithm is given in following sections.

3.1.1 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 rigid parts on one shape is transferred automatically to all other shapes.

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

3.1.2 Graph Structure

We build a complete graph where each shape is a vertex, and edges between shapes are weighted by the similarity between them, i.e., the more the connected shape pair is similar, the less their corresponding edge weight is. The main idea of using a graph of this form is to replace an interpolation between the source shape and the target shape by the composition of interpolations on the shortest path between them. This way we get a new representation of the given dataset. Using this representation, we can obtain the intermediate shapes to use them for a more natural interpolation scheme.

3.1.2.1 Edge Weights

We use the pairwise distance measure suggested by [2] as our edge weights. In this measure, the similarity between each shape is computed by accumulating the Euclidean distances between their corresponding vertices:

$$d(S_i, S_j) = \sum_{k=1}^{N} \|v_i^k - v_j^k\|^2 \tag{3.1}$$

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. The accumulated distances between the corresponding vertices gives the similarity between two shapes. The higher this value, it means that these two shapes are more different than each other, that is they are farther apart on the graph. Conversely, the shapes with smaller total distance between them are more similar and closer to each other on the graph.

We have also tried several other weighting schemes. The method described by Osada et al. [33] choses several vertex pairs and computes the Euclidean distances between them. Then, the histograms of these values are compared using 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. 3.1 based results.

We have 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. 1.1). 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 3.1.

3.1.3 Multi-Dimensional Scaling

We verify the accuracy of the similarity measure by the Multidimensional Scaling technique [34], which separates all shape pairs in 2D by the amount of dissimilarity (Eq. 3.1) between them.

Given our pairwise distance based measure, Multi-Dimensional Scaling(MDS) is suitable for providing beneficial visualization. It helps to spot clusters of shapes. Also makes it easy to assess the quality of the given measure. Figure 3.1 shows the MDS for the SCAPE[35] dataset using the distance measure described in 3.1.

In Figure 3.2, the path found between the blue shapes are visualized on the Multi Dimensional Scaling graph. The pink models show the control points of the interpolation curve between the start and the end shape.

3.2 Path Interpolation

After creating the graph and calculating the edge weights as in 3.1.2 and 3.1.2.1, we run Dijkstra's shortest path algorithm[36] on our graph, which returns the desired

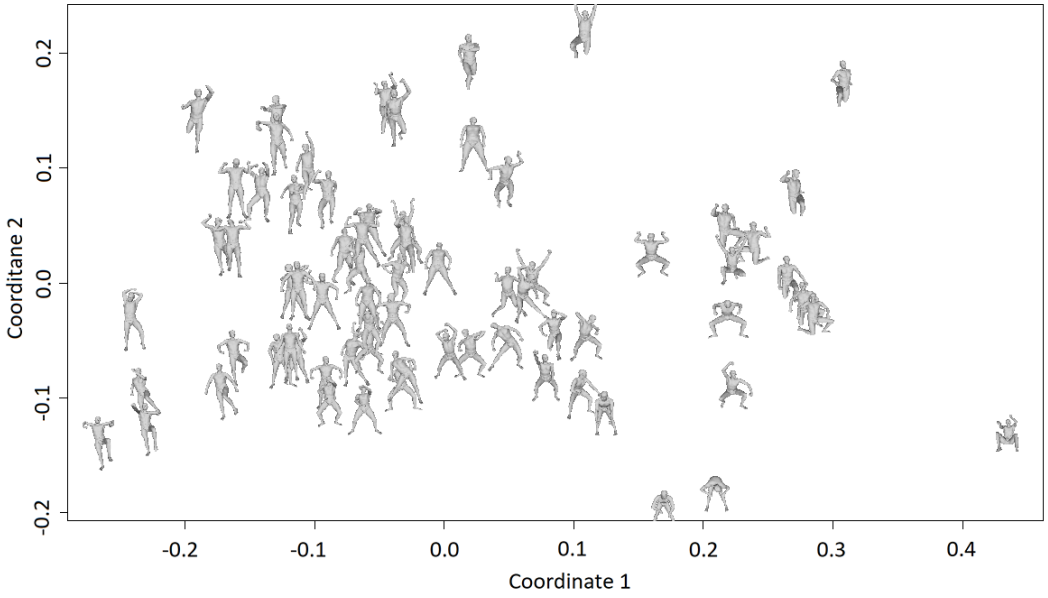

Figure 3.1: SCAPE models are arranged in 2D with separations proportional to their dissimilarities captured by Eq. 3.1.

set of intermediate shapes in order. Since the number of elements in the dataset is negligible compared to the number of vertices in each mesh, the computational cost of path finding does not cause an increase in the time cost of the algoritm.

3.2.1 Shortest Path Algorithm

We have used Dijkstra's shortest path algorithm[36] for finding a path between the source and end shapes. It uses a greedy bread-first search to find the shortest path.

The algorithm in a nutshell, can be described as follows; given non-negative edge weights, and finite number of nodes, starting from a single node, start by marking all nodes as unvisited. Give all nodes infinity distances, expect zero for the start node. Then, repeatedly update the distances visiting the unvisited nodes with the minimum distance value. The distance of a node is the total cost to reach it through its adjacent nodes. Update a node's distance only if the distance value is smaller than its previous distance value. The extracting the minimum distance valued node can be sped up using a priority queue for the already processed nodes. We have used this method in

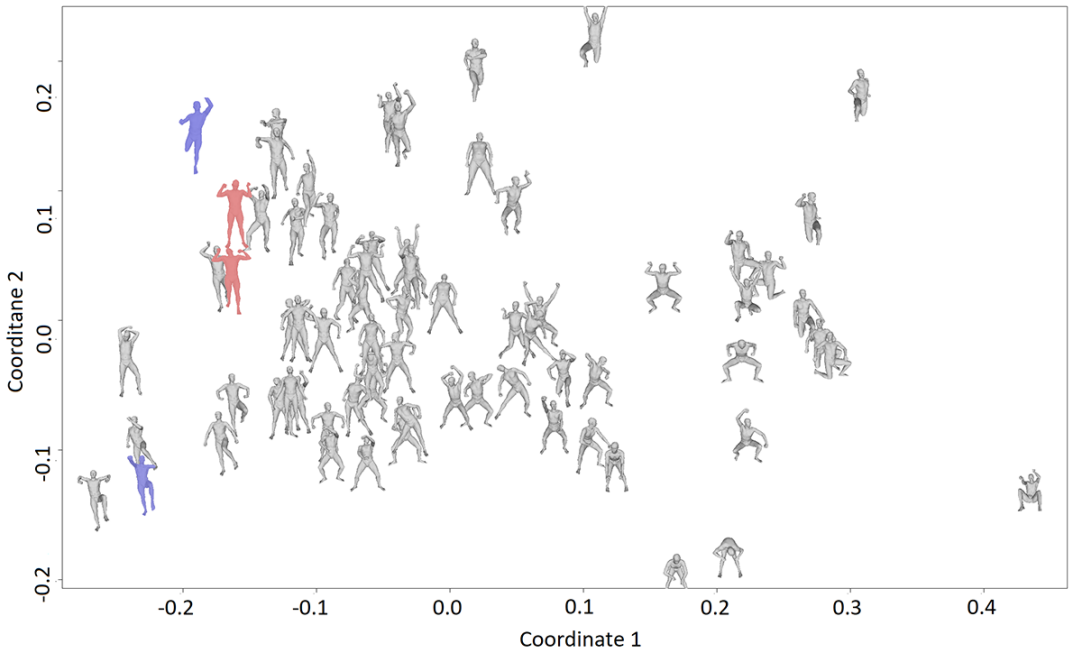

Figure 3.2: Multi Dimensional Scaling. The colored models show the shape sequence for the forth row of Figure 4.10.

our algorithm.

3.2.2 Bezier Curves

Given the 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. A Bezier curve is illustrated in figure 3.3. Note that, a Bezier curve of order r requires $r+1$ control points, e.g., 4 control points define the third order cubic Bezier curve. Note also that, a Bezier curve always passes through the first and the last control points.

$$v^k_{\text{result}}(\alpha) = \sum_{i=0}^{m} p^k_i b_{i,m}(\alpha), \quad 0 \leq \alpha \leq 1, 1 \leq k \leq N, \alpha \in \mathbb{R}, k \in \mathbb{N},$$
$$b_{i,m}(\alpha) = \binom{m}{i} \alpha^i (1-\alpha)^{m-i} \tag{3.2}$$

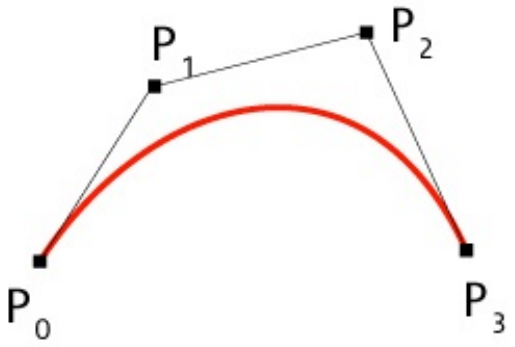

Figure 3.3: Visualization of a cubic Bezier curve.

We use the standard De Casteljau's algorithm to fit a Bezier curve to each vertex in a mesh using the vertices of the meshes in the calculated mesh sequnce as control points(Eq. 3.2). Similar to the linear interpolation formula (Eq. 1.1), we consider each of the N vertices separately and move them on their curves under unified time parameterization α. p_i^k denoting the k^{th} vertex of the control/intermediate meshes in the computed shape sequence of size $m + 1$, Eq. 3.2 gives us the position of the k^{th} vertex at a given time α.

3.3 Rigid Parts

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 divide database shapes into rigid parts and use these as inputs for our graph structure.

Specifically, we divide the SCAPE models into 2, 5, and 9 rigid parts and the Hand models into 6 and 11 rigid parts. We test with different number of parts and promote the result with the max number of parts since the variety is, expectedly, captured better with them. Other datasets are divided into 5, 7, and 7 parts for the Cat, Horse, and Dancer databases, respectively (Figure 3.4).

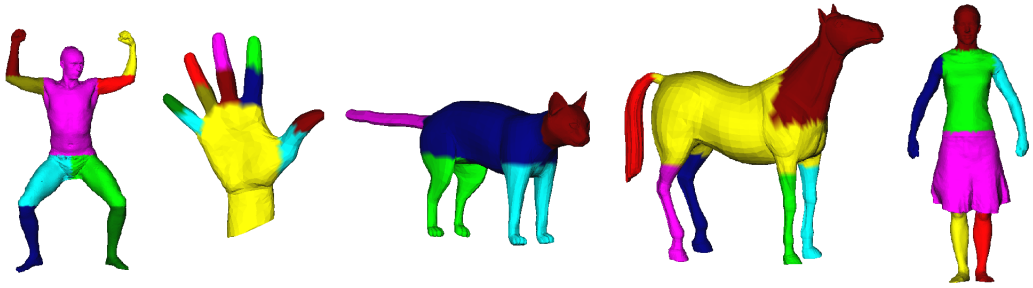

Figure 3.4: Representative models of databases. Parts highlighted.

Using this rigid-parts based approach, it is possible to find configurations of given a part throughout the dataset in isolation that would be otherwise lost when considering a whole shape. Since our distance measure is accumulative, using a whole shape for the graph based scheme, the distance between two shapes depends on the total similarity between them. However, for example on human models, the movement of limbs may have all the intermediate rotations that we desire, but on different overall configurations. Partitioning this movements and using the parts' configuration over the whole dataset, provides better local movements.

3.3.1 Unified Merging

Having obtained the rigid parts, we treat the parts separately to create their own graphs (Section 3.1.2). 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.

We apply the method described in 3.2 for each rigid 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 Figure 3.5.

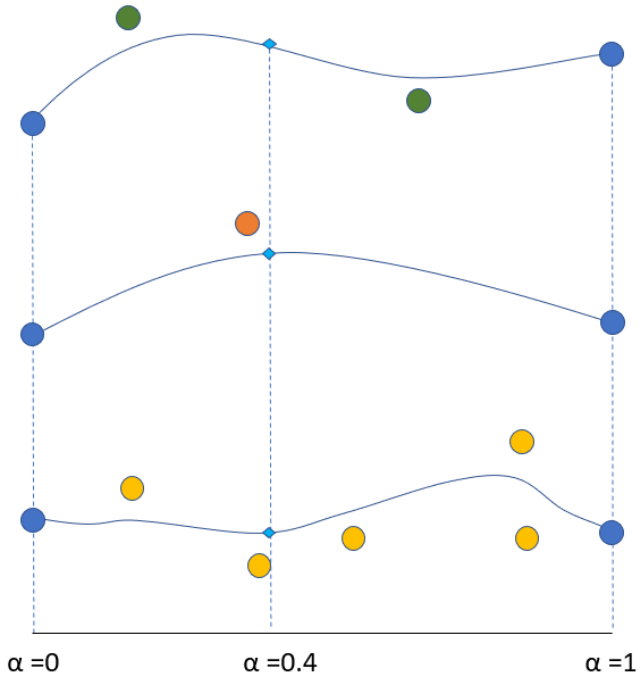

Figure 3.5: Using Bezier curves for multiple paths (3 in this case) with different lengths allows unified timing and smooth transition.

CHAPTER 4

EXPERIMENTS

4.1 Datasets

We have tested the performance of our data-driven shape interpolation algorithm on datasets of articulated objects with non-uniform sampling. The first dataset is a reconstructed pose sequence of a human actor from the SCAPE benchmark [37], which contains 71 models, whereas the second one is our new Hand dataset with 31 models. The other datasets consist of 27 Dancer models [38], 11 Horse models [39], and 10 Cat models [39]. Despite the small datasets, our part-based solution achieves visually pleasing and natural interpolations between shape pairs.

4.1.1 SCAPE Dataset

SCAPE dataset[35] consists of human pose sequences. It consists of 71 models with fixed connectivity. Each model has 12500 corresponding vertices and 24998 triangles. We have preprocessed the dataset to eliminate the global rotation differences.

In Figure4.1, Multi Dimensional Scaling (MDS) of the dataset using the distance measure in Eq. 3.1 is shown.

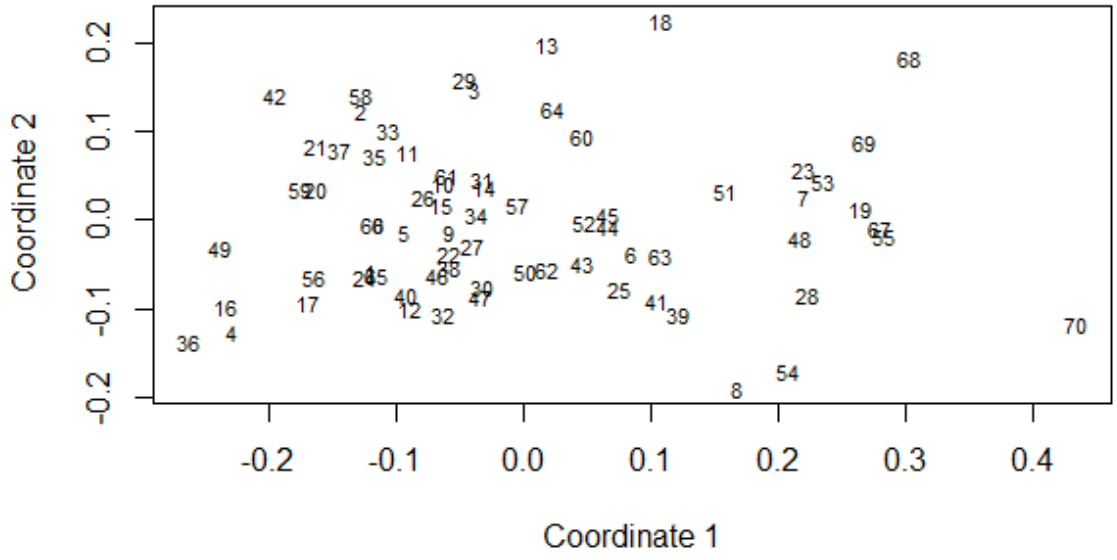

Figure 4.1: MDS of SCAPE Dataset

4.1.2 Hand Dataset

We have created a new dataset on hands, another popular articulated object like humans. This Hand dataset is created by deforming a single hand model using the Blender software [30], preserving the fixed connectivity. The hand gestures are based on the images taken from dataset for Hand Gesture Recognition (HGR) which contains Polish and American Sign Language gestures [40]. We have produced 31 hand models with 1515 vertices. Note that we made our Hand dataset public as a side contribution of this research.

In Figure4.2, MDS of the dataset using the distance measure in Eq. 3.1 is shown.

A set of Hand dataset results is depicted in Figure 4.3. This figure also illustrates the selected intermediate models from the dataset (top row in pink) which are the elements of the shortest path from the source model to the end model(given in blue).

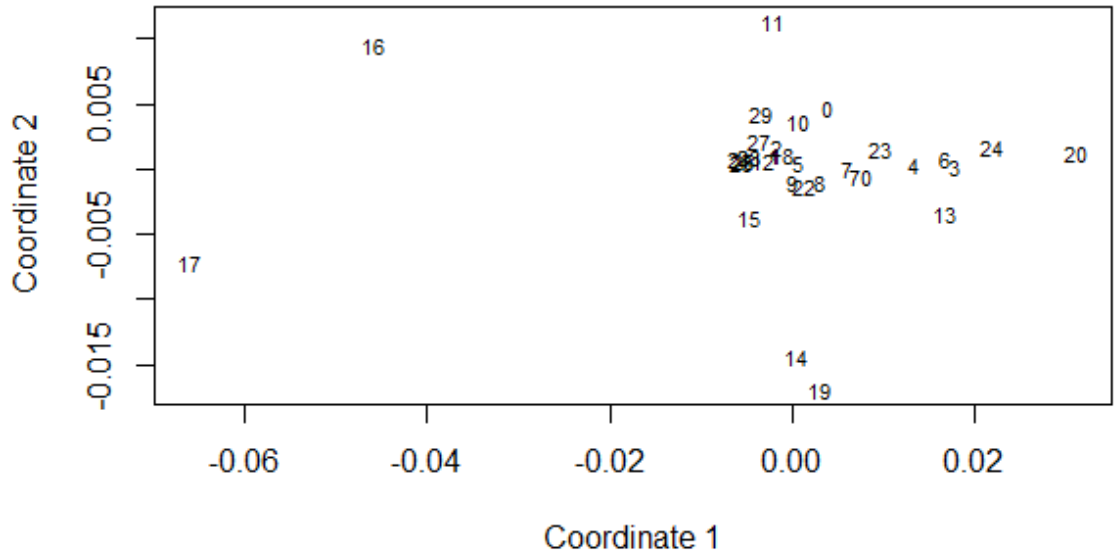

Figure 4.2: MDS of Hand Dataset

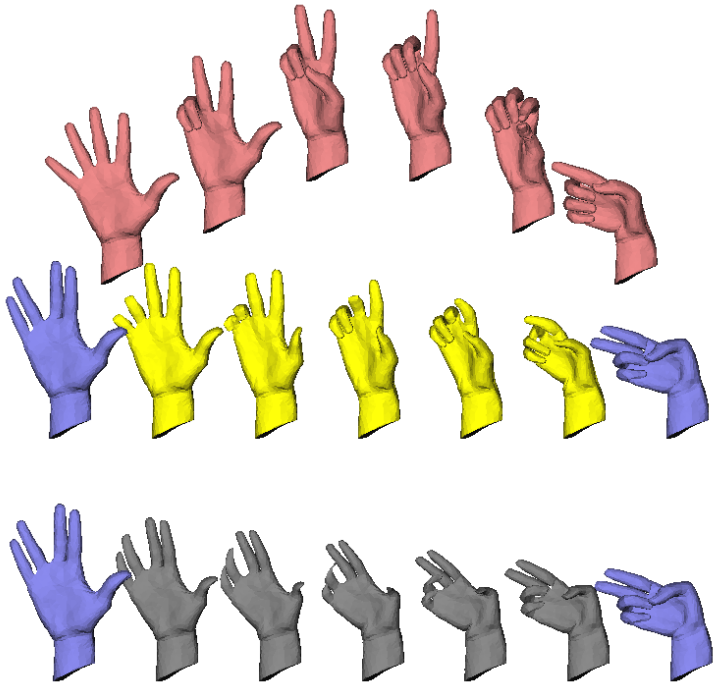

Figure 4.3: Models in pink (top) guide the data-driven interpolation (middle) which performs better than simple linear interpolation (bottom).

4.1.3 Cat and Horse Datasets

Cat and Horse datasets are taken from [39]. Cat dataset consists of 10 cat models and Horse dataset consists of 11 horse models. Cat models consist of corresponding 7207 vertices and 14410 triangles and horse models consist of corresponding 8431 vertices and 16843 triangles. All models are in different poses. Their MDS visualizations are shown in Figure 4.4 and Figure 4.5, respectively.

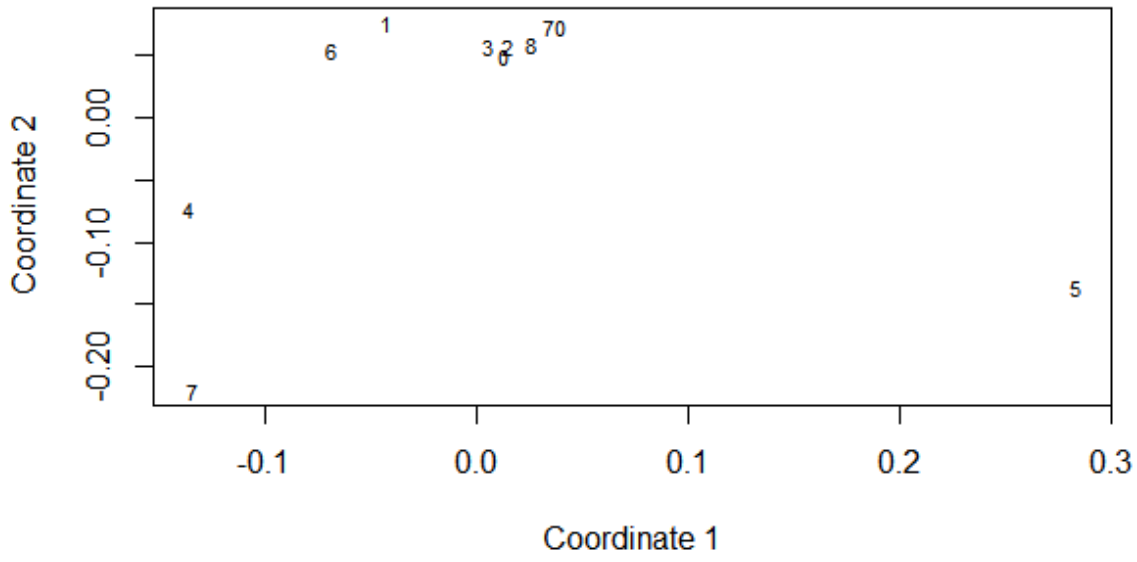

Figure 4.4: MDS of Cat Dataset

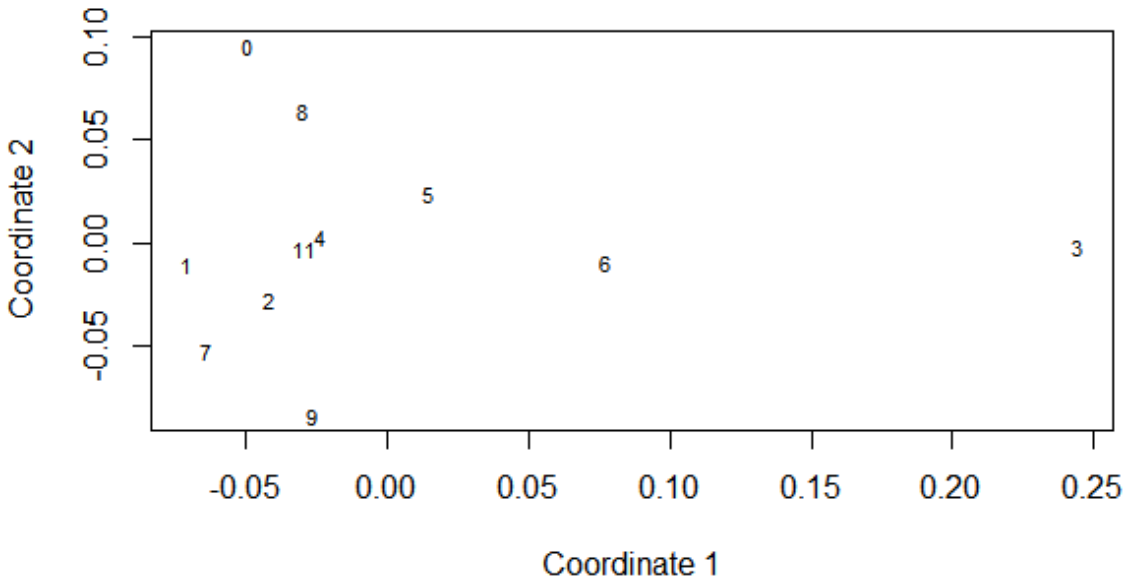

Figure 4.5: MDS of Horse Dataset

4.1.4 Dancer Dataset

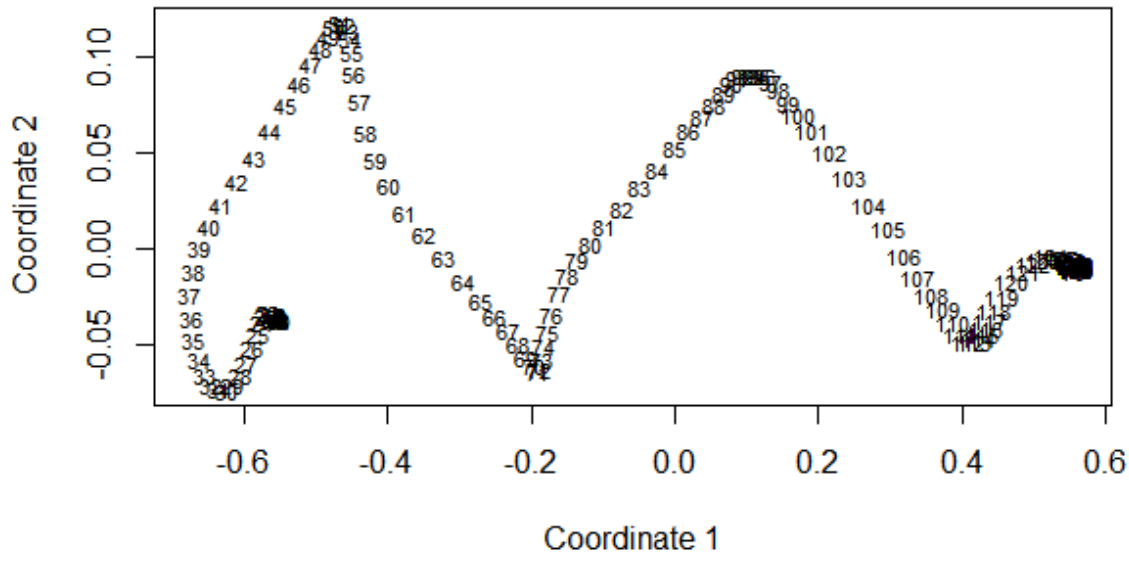

Figure 4.6: MDS of Dancer Dataset as is.

Dancer dataset taken from [38] originally consists of 175 models, taken from a video sequence. Firstly, since the models have large translations between them, we have centered the models. The MDS before centering can be seen in Figure 4.6 using Eq.

3.1.

After getting rid of the large translation differences, we obtain the MDS in Figure 4.7. However, the pairwise distances are very small as they are based on high frame-rate video. This caused unnecessarily long shape sequences in our interpolation scheme.

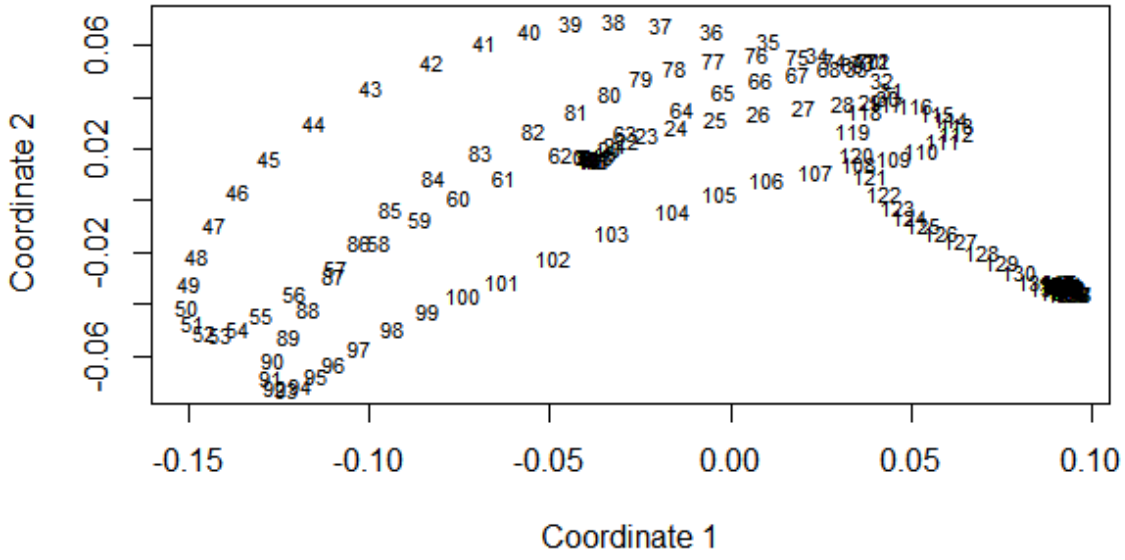

Figure 4.7: MDS of Dancer Dataset after centering the models

We then, uniformly sampled the dataset and get the MDS in Figure 4.8. We have used these 27 models in our experiments.

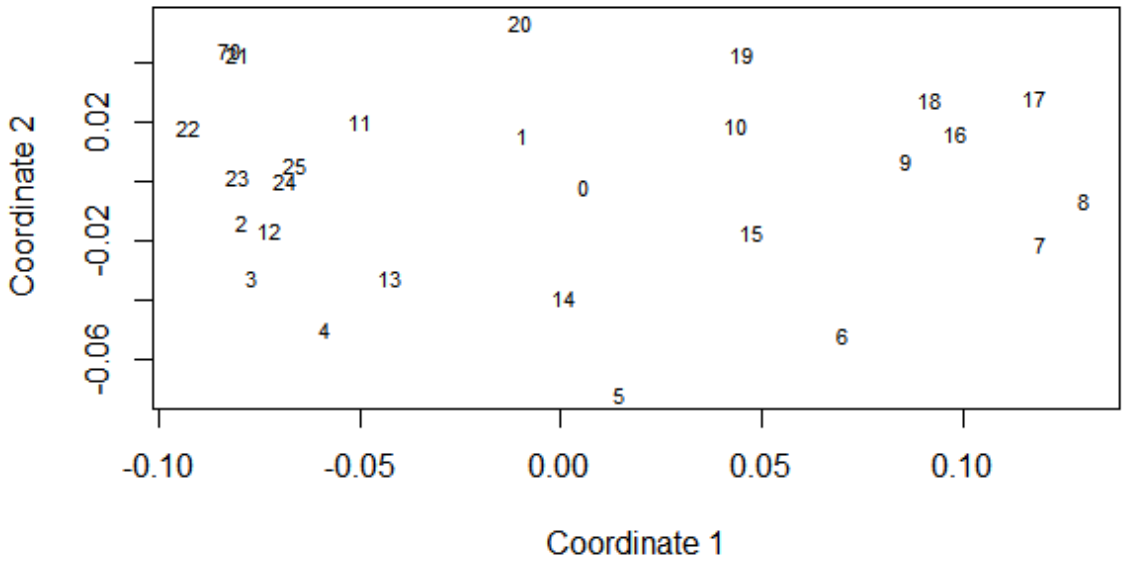

Figure 4.8: MDS of Dancer Dataset after sampling centered models.

4.2 Results

In order to emphasize our contribution, we compare with two related methods that we call M1 and M2. 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. In M2, on the other hand, we perform a direct linear interpolation (Eq. 1.1) between the source and the target, which is not data-driven at all.

The first example that shows our success is on SCAPE (Figure 4.9). Given the left-most 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). The fourth row shows the unified merging and hence our promoted result, which is better than the methods M1 (row 5) and M2 (row 6). 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 the top 3 rows whose uncolored parts behave completely differently, e.g., sudden jumps in the left arm of row 1. As a matter of fact,

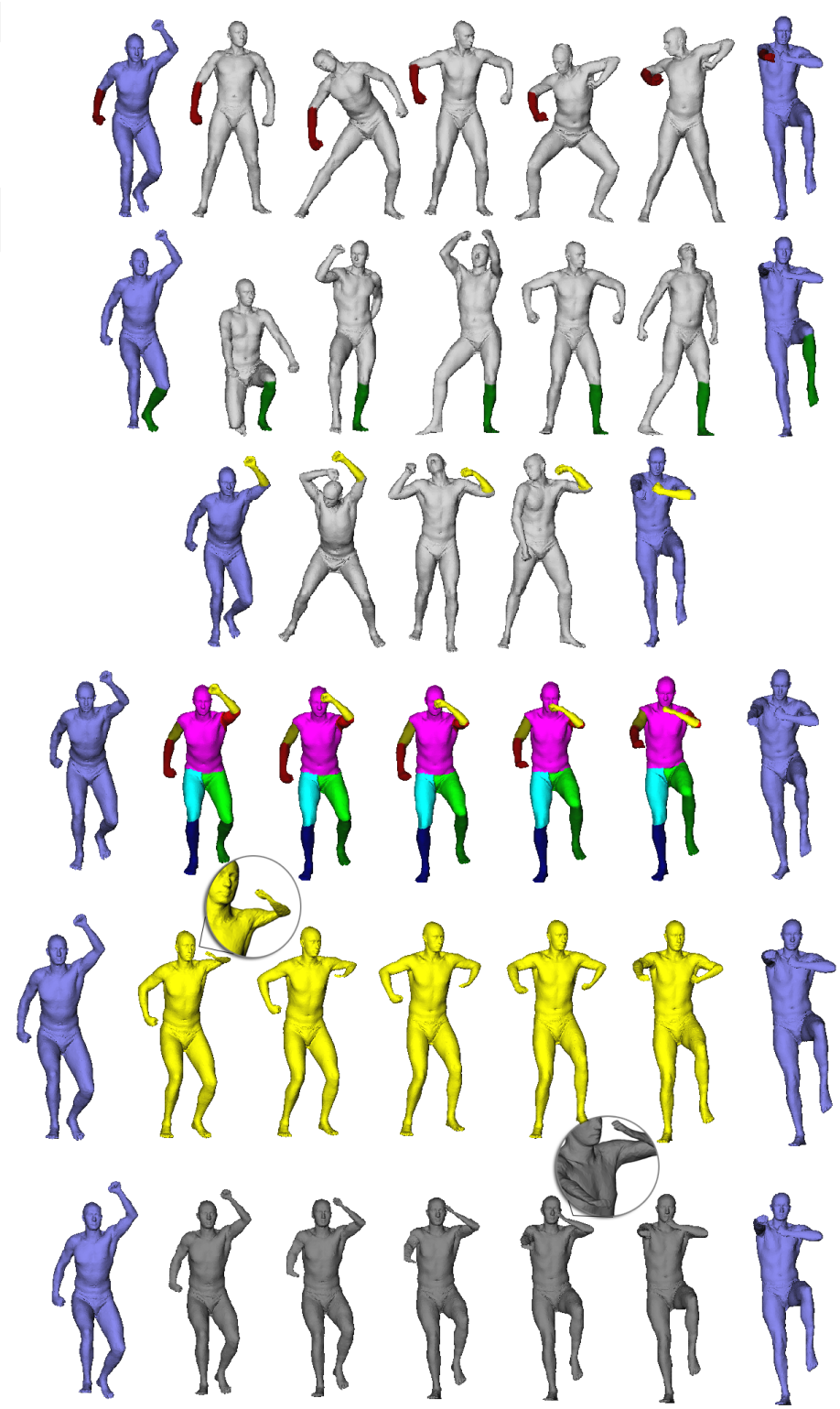

Figure 4.9: Three interpolations of three different parts (rows 1-3) for our algorithm whose final result (row 4) outperforms the results of M1 (row 5) and M2 (row 6). Note zoomed artifacts of M1 & M2.

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 datasets. A significantly larger dataset may upgrade the performance of M1 in the expense of increased computational time.

We provide further results on SCAPE in Figure 4.10. 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 of their feet in the air. Since the dataset only has models that have at least one foot on the ground, the 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.

Similar to Figure 4.10, we show in Figure 4.11 our results with varying number of parts. Final results for Hand and SCAPE pairs are given in Figure 4.12.

We finish our evaluations with three other articulated datasets: Dancer, Cat and Horse (Figure 4.13).

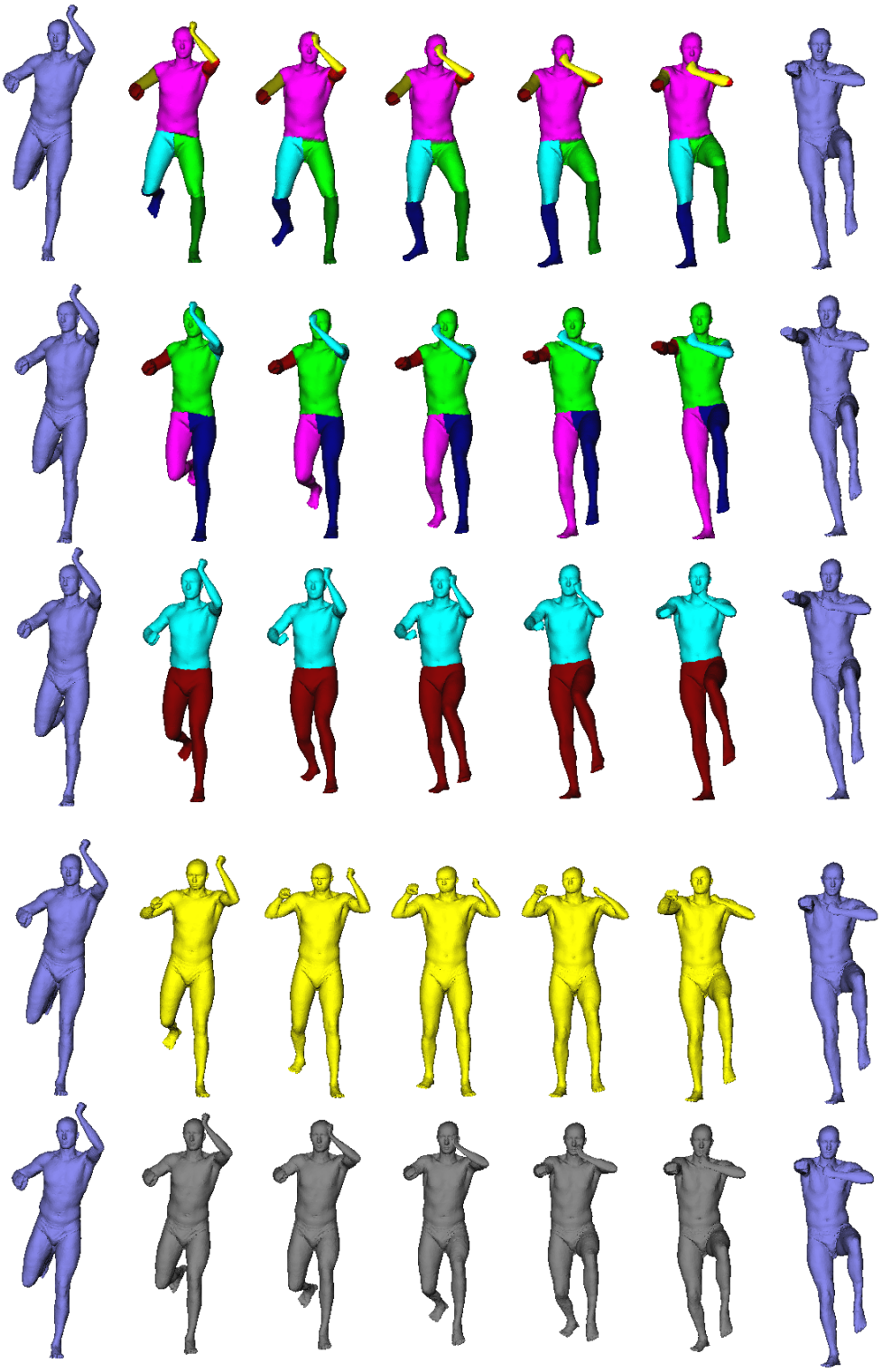

Figure 4.10: Our results with changing number of parts (rows 1-3) in comparison with M1 (row 4) and M2 (row 5).

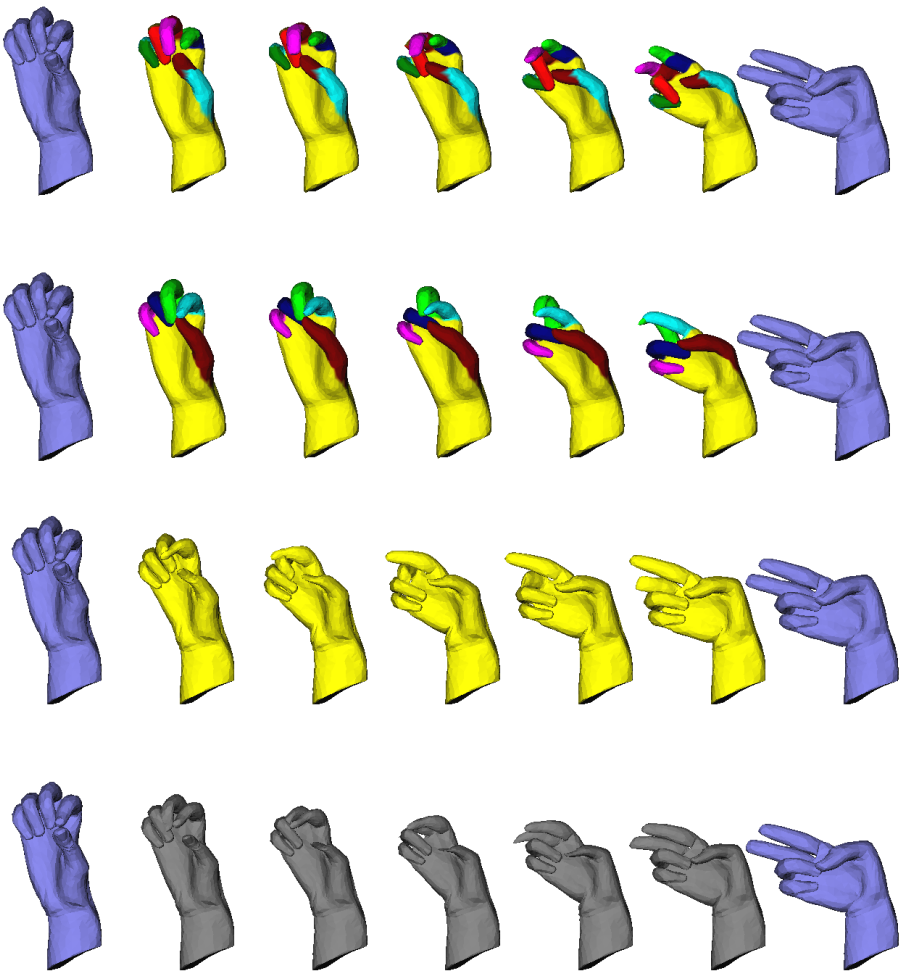

Figure 4.11: Our results with changing number of parts (rows 1-2) in comparison with M1 (row 3) and M2 (row 4).

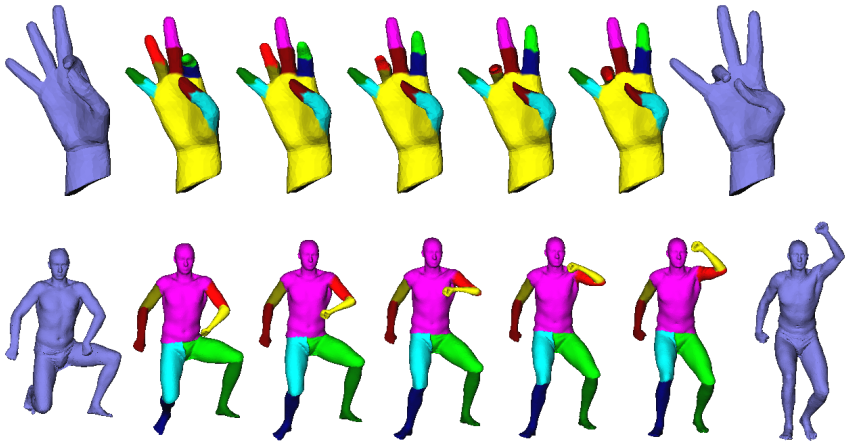

Figure 4.12: Our results on the Hand and SCAPE datasets.

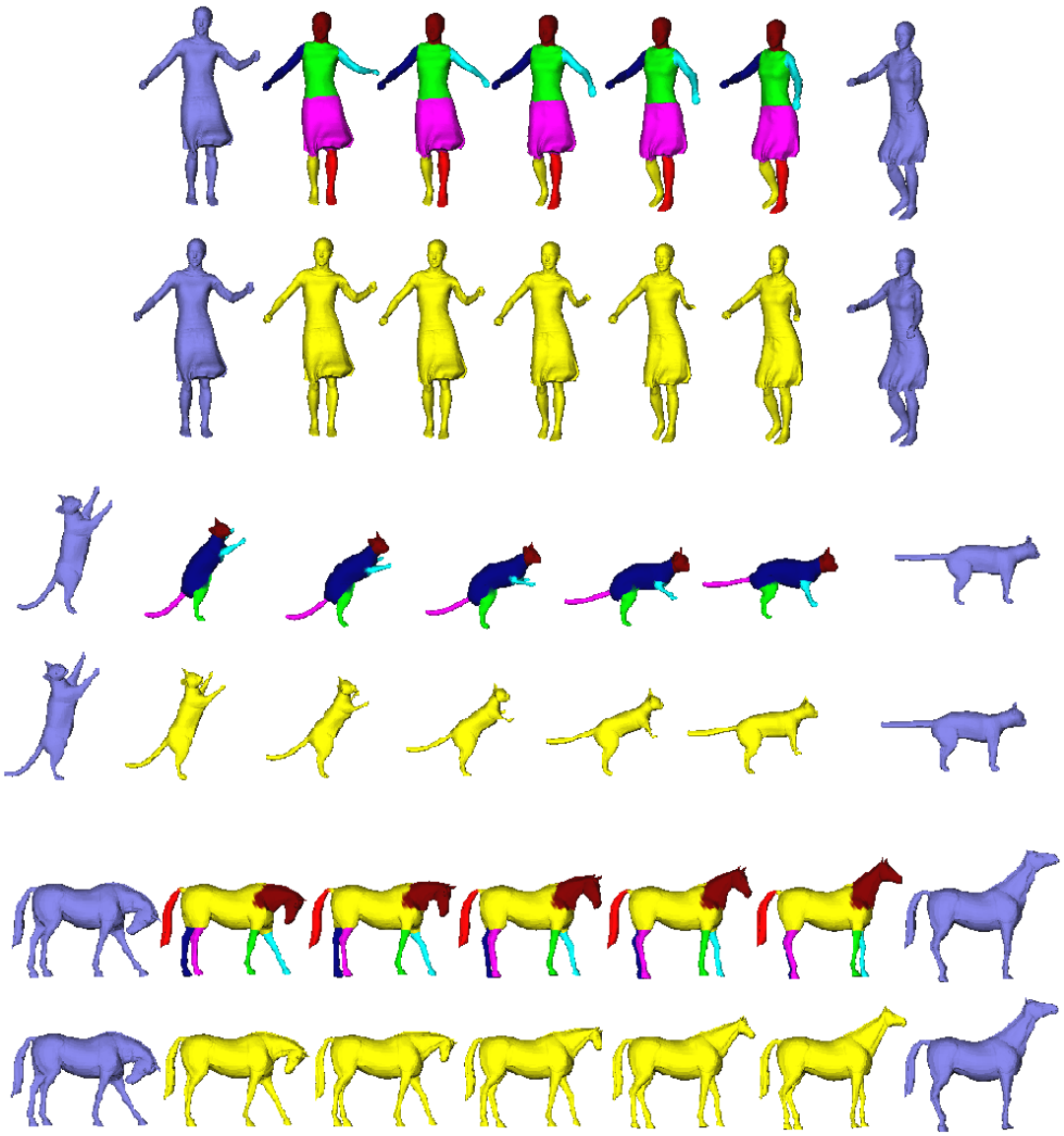

Figure 4.13: For each of the three datasets, our results (top) are shown in comparison with M1 (bottom). Note the volume loss in M1.

Table 4.1: Statistics of the datasets and average running times in milliseconds.

Dataset	# Vertices	# Models	# Parts	Edge Weights	Path Computations	Bezier Computations	M1	M2
SCAPE	12500	71	1	1049	26.1	3	29.1	0.9
SCAPE	12500	71	2	1862	70.1	3.4		
SCAPE	12500	71	5	2108	305.4	18.3		
SCAPE	12500	71	9	2126	942	34.9		
Hand	1515	31	1	31	5.3	0.2	5.5	0.1
Hand	1515	31	6	62	92.6	1.6		
Hand	1515	31	11	70	268.1	10		
Cat	7207	10	1	15	3.3	1.6	4.9	0.5
Cat	7207	10	5	31	15.9	2.5		
Horse	8431	11	1	23	1.5	0.6	2.1	0.3
Horse	8431	11	7	36	18.7	7.2		
Dancer	9971	27	1	122	6.4	1.5	7.9	0.3
Dancer	9971	27	7	229	90.4	15.9		

The execution times of our method and the compared methods on a computer with an Intel Corei7 2.40GHz CPU are shown in Table 4.1. 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 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 for once just in the preprocessing stage.

4.3 Limitations

Dividing a shape into a certain number of rigid parts in a fully automatic manner is an open research problem that may benefit from machine learning techniques. There already exist several semi-automated techniques. Fully automated methods tend to produce extra clusters around joints and need to improved.

The increased variety that is gained from dividing the models into rigid parts increases the results' expressiveness and stability. However, the price of this improvement can be the loss of the natural movements. Without connectivity constrains on the parts, it is possible to get parts merged in undesired and unnatural ways which were appropriate in isolation.

Our method is still sensitive to large rotations since it is our curve framework is still based on linear interpolation. However, the effect of providing more in-between shapes decreases the effect of large rotation since the intermediate shapes decrease the step-by-step rotation between an start and an end shape. Adding rigid-parts extends this idea one step further and makes our simple method much more effective.

CHAPTER 5

CONCLUSION AND FUTURE WORK

We presented a part-based data-driven shape interpolation method. Thanks to our part-based solution that highly utilizes the relatively small sized databases, we obtain physically plausible and geometrically sound smooth interpolations between source and target shape pairs in a data-driven fashion. Our unified merging scheme made the gathering of different part-level interpolations into one consistent interpolation possible.

An automated rigid part segmentation scheme is within the scope of active research. Even though providing a fully-automated scheme seems to be within reach after incorporating learning-based methods, a semi-automated non-learning-based technique can provide starting points for segmentation.

Providing connection information for neighboring rigid parts can solve the problems of isolated part interpolation and increase the quality of the interpolation.

It is still possible to integrate geometry based methods into the simple framework we have presented, e.g., while interpolating two consecutive shapes on the shortest path. This would increase the geometric stability of the results.

REFERENCES

[1] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear rotation-invariant coordinates for meshes," *ACM Transactions on Graphics*, vol. 24, pp. 479–487, 2005.

[2] L. Gao, Y.-K. Lai, Q.-X. Huang, and S.-M. Hu, "A data-driven approach to realistic shape morphing," *Computer Graphics Forum*, vol. 32, pp. 449–457, 2013.

[3] Q.-X. Huang, B. Adams, M. Wicke, and L. Guibas, "Non-rigid registration under isometric deformations," *Computer Graphics Forum*, vol. 27, pp. 1449–1457, 07 2008.

[4] D. Xu, H. Zhang, Q. Wang, and H. Bao, "Poisson shape interpolation," in *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, (New York, NY, USA), pp. 267–274, ACM, 2005.

[5] M. Alexa, "Recent advances in mesh morphing," *Computer Graphics Forum*, vol. 21, pp. 173–198, 2002.

[6] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Numerical geometry of non-rigid shapes," *Springer Science & Business Media*, 2008.

[7] S. Y. Man, W. Wenping, and C. Francis, "Interpolating polyhedral models using intrinsic shape parameters," *The Journal of Visualization and Computer Animation*, vol. 8, no. 2, pp. 81–96, 1997.

[8] M. Alexa, "Differential coordinates for local mesh morphing and deformation," *The Visual Computer*, vol. 19, no. 2, pp. 105–114, 2003.

[9] M. Kilian, N. J. Mitra, and H. Pottmann, "Geometric modeling in shape space," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. #64, 1–8, 2007.

[10] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, "A simple geometric model for elastic deformations," in *A Simple Geometric Model for Elastic Deformations*, pp. 38:1–38:6, 2010.

[11] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross, "Example-based elastic materials," *ACM Transactions on Graphics*, pp. 72:1–72:8.

[12] H. K. Chu and T. Y. Lee, "Multiresolution mean shift clustering algorithm for shape interpolation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 5, pp. 853–866, 2009.

[13] S. Fröhlich and M. Botsch, "Example-driven deformations based on discrete shells," *Computer Graphics Forum*, vol. 30, pp. 2246–2257, 2011.

[14] T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann, "Multi-scale geometry interpolation," *Computer Graphics Forum*, vol. 29, no. 2, pp. 309–318, 2010.

[15] C. Von-Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, "Real-time nonlinear shape interpolation," *ACM Transactions on Graphics*, vol. 34, no. 3, pp. 34:1–34:10, 2015.

[16] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, and H. Zhang, "Topology-varying 3d shape creation via structural blending," *ACM Transactions on Graphics*, vol. 33, no. 4, p. 158, 2014.

[17] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, "2-d shape blending: an intrinsic solution to the vertex path problem," in *SIGGRAPH*, 1993.

[18] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 157–164, 2000.

[19] E. Chien, R. Chen, and O. Weber, "Bounded distortion harmonic shape interpolation," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 105:1–105:15, 2016.

[20] R. Chen, O. Weber, D. Keren, and M. Ben-Chen, "Planar shape interpolation with bounded distortion," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 108, 2013.

[21] Y. Zhu, J. Popovic, R. Bridson, and D. M. Kaufman, "Planar interpolation with extreme deformation, topology change and dynamics," *ACM Transactions on Graphics*, vol. 36, no. 6, 2017.

[22] P. P. J. Sloan, C. F. Rose III, and M. F. Cohen, "Shape by example," *Proc. 2001 symposium on Interactive 3D graphics*, vol. 33, no. 4, pp. 135–143, 2001.

[23] L. Gao, S. Chen, Y.-K. Lai, and S. Xia, "Data-driven shape interpolation and morphing editing," *Computer Graphics Forum*, vol. 36, 2016.

[24] H. Averbuch-Elor, D. Cohen-Or, and J. Kopf, "Smooth image sequences for data-driven morphing," *Computer Graphics Forum*, vol. 35, no. 2, pp. 203–213.

[25] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "Shape google: Geometric words and expressions for invariant shape retrieval.," *ACM Transactions on Graphics*, vol. 30, no. 1, p. 1, 2011.

[26] Y. Sahillioğlu and L. Kavan, "Detail-preserving mesh unfolding for nonrigid shape retrieval," *ACM Transactions on Graphics*, vol. 35, no. 3, p. 27, 2016.

[27] Y. Sahillioğlu and M. Sezgin, "Sketch-based articulated 3d shape retrieval," *IEEE computer graphics and applications*, vol. 37, no. 6, pp. 88–101, 2017.

[28] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps.," *ACM Transactions on Graphics*, vol. 30, no. 4, p. 79, 2011.

[29] Y. Sahillioğlu and Y. Yemez, "Coarse-to-fine isometric shape correspondence by tracking symmetric flips," *Computer Graphics Forum*, vol. 32, no. 1, pp. 177–189, 2013.

[30] Blender Online Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.

[31] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas, "An optimization approach to improving collections of shape maps," *Computer Graphics Forum*, vol. 30, no. 5, pp. 1481–1491, 2011.

[32] Y. Sahillioğlu and Y. Yemez, "Multiple shape correspondence by dynamic programming," *Computer Graphics Forum*, vol. 33, no. 7, pp. 121–130, 2014.

[33] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.

[34] Y. Sahillioğlu and Y. Yemez, "Minimum-distortion isometric shape correspondence using em algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2203–2215, 2012.

[35] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM Transactions on Graphics*, vol. 24, pp. 408–416, 2005.

[36] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[37] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: Shape completion and animation of people," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 408–416, 2005.

[38] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 97, 2008.

[39] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 399–405, 2004.

[40] M. Kawulok, J. Kawulok, and J. Nalepa, "Spatial-based skin detection using discriminative skin-presence features," *Pattern Recognition Letters*, vol. 41, pp. 3–13, 2014.