

CURVE-SKELETON AND MESH TRANSFER BETWEEN SHAPES

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

BY

ÇAĞLAR SEYLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JANUARY 2021

Approval of the thesis:

CURVE-SKELETON AND MESH TRANSFER BETWEEN SHAPES

submitted by **ÇAĞLAR SEYLAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy 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. Ahmet Oğuz Akyüz
Computer Engineering, METU

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

Prof. Dr. Tolga Can
Computer Engineering, METU

Prof. Dr. Uğur Gündükbay
Computer Engineering, Bilkent University

Assist. Prof. Dr. Ufuk Çelikcan
Computer Engineering, Hacettepe 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, Surname: Çağlar Seylan

Signature :

ABSTRACT

CURVE-SKELETON AND MESH TRANSFER BETWEEN SHAPES

Seylan, Çağlar

Ph.D., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Yusuf Sahillioğlu

January 2021, 112 pages

The set of points having more than one closest point to the boundary of a shape is called the medial-axis or skeleton. Since its first introduction, many of its variants have been developed and used in shape analysis, retrieval, deformation, and many other applications. Medial-axis of a 3D shape consists of both 2D and 1D structures and may become quite complex. Thus, 1D curve representations of medial-axis, called curve-skeletons, have been becoming more popular. The thesis considers two related problems regarding curve-skeletons. In the first part, a method for transferring the curve-skeleton of a source shape to target shapes without extracting them from scratch is given. The second part deals with the inverse problem, that is, constructing mesh structure from a curve-skeleton given a source shape with its curve-skeleton. For the both parts, we demonstrate the advantages of our methods over the existing ones through comprehensive evaluations, both visually and quantitatively.

Keywords: Skeleton, 3D shape abstraction, ARAP, Rigging, Skinning

ÖZ

ŞEKİLLER ARASINDA EĞRİ-İSKELET VE KAFES AKTARIMI

Seylan, Çağlar

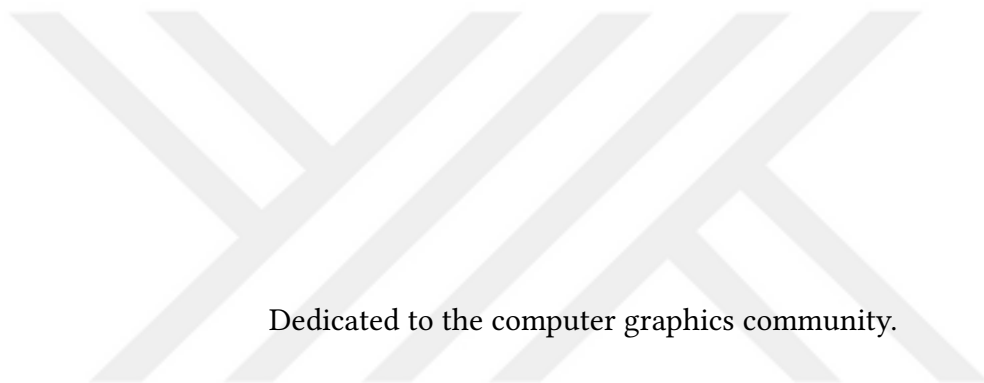
Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Yusuf Sahillioğlu

Ocak 2021 , 112 sayfa

Bir şeklin yüzeyinde birden fazla en yakın nokta içeren ve şeklin içinde yer alan noktalar kümesine orta eksen ya da iskelet denir. Bu kavram ilk ortaya atıldığından beri bir çok varyasyonu geliştirilmiştir ve şekil analizi, şekil sorgulama, şekil deformasyonu ve başka bir çok alanda başarılı şekilde uygulanmıştır. 3 boyutlu bir şeklin orta eksenini 2 ve 1 boyutlu yapılardan oluşmaktadır ve oldukça karmaşık olabilmektedir. Bu nedenle, orta eksenin tek boyutlu bir varyasyonu olan eğri iskelet kavramı ortaya çıkmıştır ve oldukça popüler hale gelmektedir. Bu tez iki farklı fakat birbiriyle ilgili problemi ele almaktadır. İlk kısımda bir kök şeklin eğri iskeletini hedef şekillere, onların eğri iskeletlerini en baştan hesaplamadan, transfer eden bir yöntem anlatılmaktadır. İkinci kısımda ise bu problemin tersi, yani, elimizde bir kök şekil ve onun eğri iskeleti varken, o şeklin diğer eğri iskeletleri için kafes yapısı hesaplanması ele alınmaktadır. her iki kısım için önerdiğimiz metodun var olan metodlara göre avantajları kapsamlı şekilde hem görsel hem de niceliksel olarak değerlendirilmiştir.

Anahtar Kelimeler: İskelet, 3B şekil soyutlama, ARAP, 3B model üretimi



Dedicated to the computer graphics community.

ACKNOWLEDGMENTS

First and foremost, I want to thank my supervisor Prof. Yusuf Sahillioğlu, for his consistent support throughout my Ph.D. studies. Without any doubt, his vision and guidance helped me much to do this work.

I would also like to thank Scientific and Technological Council of Turkey (TÜBİTAK) for supporting and funding this work under the project EEEAG-115E471.

And I would like to mention my thanks for my colleagues for their constructive technical and non-technical discussions: Ömer, Adnan, Barış, Merve, Mustafa, and Amin.

Moreover, I would like to mention the names of invaluable ones, who were spread over the world, yet always stayed by my side, supported me mentally and inspired me: Ünver, Marie, Uğur, and Anıl.

Last but not least, I want to thank my family for all kinds of support during my Ph.D. studies.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vi
ACKNOWLEDGMENTS	viii
TABLE OF CONTENTS	ix
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF ABBREVIATIONS	xxi
CHAPTERS	
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Contributions	2
1.3 Organization of the Thesis	3
2 BACKGROUND	5
2.1 3D Point-Set Matching	5
2.2 ARAP Deformation	7
2.3 Skinning	11
2.3.1 Linear Blend Skinning (LBS)	12
2.3.2 Heat-Bone Weight Computation	13

2.3.3	Dual Quaternion Skinning (DQS)	14
2.3.4	Fast Automatic Skinning Transformations (FAST)	17
2.3.5	Direct Delta Mush (DDM)	18
3	3D SKELETON TRANSFER BETWEEN MESHES AND CLOUDS	19
3.1	Introduction	19
3.2	Related Work	21
3.3	Method	25
3.3.1	Surface Correspondence	25
3.3.2	Skeleton Transfer	29
3.3.3	Postprocessing	31
3.4	Results and Discussion	32
3.4.1	Datasets	33
3.4.2	Intra-Subject Skeleton Transfer Experiments (FAUST)	34
3.4.3	Inter-Subject Skeleton Transfer Experiments (FAUST)	34
3.4.4	Experiments for Skeleton Transfer to Downsampled Point Clouds (FAUST)	36
3.4.5	Experiments for Skeleton Transfer to Punctured Meshes (FAUST)	37
3.4.6	Experiments for Skeleton Transfer Between Non-Human Subjects (TOSCA)	38
3.4.7	Experiments for Skeleton Transfer Between Non-Registered Shapes	40
3.4.8	Quantitative Analysis	41
3.4.9	Timings	44
3.4.10	Limitations	44

3.5	Conclusion and Future Work	47
4	TEMPLATE SKELETON EMBEDDING INTO 3D SHAPES	61
4.1	Introduction	61
4.2	Related Work	64
4.3	Method	65
4.3.1	Dissimilarity of Two Skeletons	66
4.3.2	Skeleton Matching	67
4.3.3	Skeleton Segment Reordering and Vertex Positioning	71
5	SKELETON-DRIVEN ARAP DEFORMATION FOR 3D SHAPE TRANSFER	73
5.1	Introduction	73
5.2	Related Work	73
5.2.1	ARAP Deformation	73
5.2.2	Skinning	75
5.3	Method	76
5.3.1	Selection and Movement of Handle Vertices	77
5.3.2	Mesh Augmentation	79
5.3.3	Skeleton-Driven ARAP Deformation	80
5.4	Results and Discussion	84
5.4.1	Visual Results	84
5.4.2	Quantitative Analysis	90
5.4.3	Timings	95
5.4.4	Limitations	95
5.4.5	Example Application	97

5.5 Conclusion and Future Work	99
6 CONCLUSION AND FUTURE WORK	101
REFERENCES	103
APPENDICES	
CURRICULUM VITAE	111



LIST OF TABLES

TABLES

Table 3.1	Resulting error values for various experiments. $\mu_{\bar{\epsilon}}$ and $\sigma_{\bar{\epsilon}}$ for an experiment represent average of the error values $\bar{\epsilon}$ and standard deviation of $\bar{\epsilon}$, respectively, among the models used in the experiment. $\mu_{\bar{\kappa}}$ and $\sigma_{\bar{\kappa}}$ for an experiment represent average of the error values $\bar{\kappa}$ and standard deviation of $\bar{\kappa}$, respectively, among the models used in the experiment. l_{avg} for an experiment represents average edge lengths of the models used in the experiment.	45
Table 3.2	Time elapsed in the different stages of the method along with number of input mesh vertices and skeleton vertices. Original meshes are used. All timings are reported in seconds.	46
Table 3.3	Time elapsed in the different stages of the method along with number of input mesh vertices and skeleton vertices. Downsampled ($\sim 60\%$ removed) point clouds are used. All timings are reported in seconds. . .	46
Table 5.1	Change in the volume of the cylinder in Figure 5.3 throughout the deformations given in Figure 5.5. Initial volume V is 30.78. V' represents the volume of the resulting mesh. Percentage changes are also reported in the neighboring columns.	94
Table 5.2	Average computation times for ARAP and our SD-ARAP deformations for human, horse, and wolf meshes in milliseconds, with the relevant vertex and edge set sizes. The column with caption "Pre." shows the total time consumed in mesh augmentation stage and LU factorization of the system matrix.	97

LIST OF FIGURES

FIGURES

Figure 2.1	Point sets $\mathbf{P}$ and $\mathbf{Q}$. The points are registered to each other. That is, for every point in set $\mathbf{P}$, there is a point in set $\mathbf{Q}$. Best match is fitting $\mathbf{P}$ onto $\mathbf{Q}$ with a rigid transformation so that the sum of squared distanced between registered points are minimum.	6
Figure 2.2	Left shows a cell in the undeformed initial triangular mesh and right shows the same cell after deformation.	10
Figure 2.3	Top row depicts a cylinder mesh with a two bone skeleton embedded inside it. The bottom two figures show blend weights computed by the heat-bone weighting procedure.	15
Figure 2.4	Left figure shows averaging points linearly which results in a point inside of the unit circle. Right figure shows averaging points by using angles which results in a point on the unit circle (Reproduced from [1]).	17
Figure 2.5	Visual summary of Direct Delta Mush (Figures are reproduced from presentation of [2]).	18
Figure 3.1	Source skeletons to be transferred (human in two views). . . .	26
Figure 3.2	Two cylinders represented as triangular meshes. The cylinder at bottom is thicker than the one at top, that is, $r'_i > r_i$, and also $h' \geq h$. Subfigures at right show the top views of the cross sections of the cylinders to better depict the setup. As r_i increases, the area of the cylinder increases non-linearly.	28

Figure 3.3	An inter-subject skeleton transfer case (from the source in Figure 3.1 to this different human) showing the resulting skeleton when just the closest mesh vertices to skeleton vertices were chosen as corresponding vertices (top-right) and when ρ_i parameter was used (bottom right).	29
Figure 3.4	An extreme case in 2D setup in which the shape boundary is zigzaggy.	33
Figure 3.5	Input meshes for the first subject of FAUST dataset with the transferred skeletons inside of them, resulting from the intra-subject experiments.	35
Figure 3.6	Some of the inter-subject skeleton transfer results. Each column shows different poses of the same subject and each row shows the same poses of different subjects.	49
Figure 3.7	Some of the skeleton transfer results when 4000 out of 6890 vertices were removed from the meshes; the subjects are represented as disconnected point clouds with 2890 points. Each column shows same poses of different subjects and each row represents different poses of the same subject.	50
Figure 3.8	Subfigures of the first two rows show four of the skeleton transfer results when 5000 of 6890 vertices were removed from the meshes. Subfigures of the last row show two of the skeleton transfer results when 6000 of 6890 vertices were removed from the meshes.	51
Figure 3.9	Four of the skeleton transfer results when 91 vertices connected together on the upper-left arm of the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are in the first column. The other two columns show the resulting skeletons for two different poses.	52

Figure 3.10	Four of the skeleton transfer results when 88 vertices connected together on the upper-right leg of the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are shown in the first column. The other two columns show the resulting skeletons for two different poses.	53
Figure 3.11	Four of the skeleton transfer results when 176 vertices connected together on the torso the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are shown in the first column. The other two columns show the resulting skeletons for two different poses.	54
Figure 3.12	Results obtained from the TOSCA dataset. Each row corresponds to a different subject. The first column shows shaded versions of the source meshes, the other columns show the transferred skeleton to a different pose than the source pose for each type.	55
Figure 3.13	Skeleton transfer results for two downsampled point cloud models from the TOSCA dataset. Each row represents a different subject. The first column shows the source skeletons in the point clouds along with the number of removed vertices. The second column shows the transferred skeletons to other poses again inside the point clouds. . .	56
Figure 3.14	Skeleton transfer results for two punctured models from TOSCA dataset. Each row represents a different subject. Shaded versions of the models along with the number of removed vertices are shown in the first column. The middle and last columns show the source and transferred skeletons, respectively.	57
Figure 3.15	Some of the skeleton transfer results between non-registered shapes. The meshes in each row belongs to a different subject. . . .	58
Figure 3.16	One skeleton transfer result for each of centaur, gorilla and wolf models from TOSCA dataset.	59

Figure 3.17 The subfigure at left shows a didactic example introducing the variables pictorially used in the error metrics. The subfigure at right shows a real case in our models in which using the error metrics as they are would be inappropriate. 59

Figure 3.18 The top row shows the punctured leg sequence and the middle row shows the punctured wolf sequence. For the mesh pairs at both rows, the shaded figure at left shows the source mesh and target meshes with transferred skeletons are shown at right. Bottom row shows mesh transfer in horse collapse sequence, the leftmost figure in this row shows the source mesh and skeleton. 60

Figure 4.1 The overall method consists of two stages. The first phase takes a source mesh and a target skeleton, and outputs the source mesh with the embedded version of the target skeleton inside of it. The second phase takes the source mesh with embedded skeleton and the target skeleton as inputs. It augments the source mesh by adding additional edges and deforms the shape towards the target skeleton by using ARAP deformation. 66

Figure 4.2 A skeleton segment in a target skeleton and its embedded version. $\mathbf{v}_j$ and $\mathbf{v}_t$ are the end points of the segment and $\mathbf{v}'_j$ and $\mathbf{v}'_t$ are their corresponding points in the embedded version. The position of $\mathbf{v}'_r$ is determined according to the parametric position of $\mathbf{v}_r$ 68

Figure 4.3 First column shows the extracted curve-skeleton of a human mesh to be used as the source. Second column shows a target skeleton. Third column shows a correct matching between the extracted skeleton and the target skeleton. Fourth column is an example to an invalid matching. 69

Figure 4.4 Chromosomes and genes in the skeleton matching problem. . . 70

Figure 5.1	Three cases for a bone while selecting free vertices according to ρ in a 2D case. In the top, neither u nor v is terminal, in the middle u is terminal but v is not terminal, in the bottom, u is not terminal but v is terminal. Green points represent free vertices and black points represent handles.	78
Figure 5.2	Top figure shows the normal plane at point s_i of the skeleton, which is shown as a red stick, inside of a hexagonal prism. Bottom figure shows two parallel planes to the normal plane s_i with ϵ gap in-between.	79
Figure 5.3	Green edges show support edges of a cylinder mesh connecting its skeleton to its surface. The middle subfigures show the same mesh from the inside to reveal the support edges more clearly. Right subfigure shows the source mesh, green region represents free vertices, yellow region represents handles.	81
Figure 5.4	Left subfigure shows the skeleton embedded into the human mesh. Middle subfigure shows support edges added to the source mesh which is the augmented mesh. Right subfigure shows the source mesh, green regions represent free vertices, yellow regions represent handles.	82
Figure 5.5	Deforming the cylinder shown in Figure 5.3 with various bending angles by using ARAP, SR-ARAP, and our SD-ARAP. Top row shows the target skeletons for various bending angles column by column. Other rows show the deformation results for ARAP, SR-ARAP, and our SD-ARAP, respectively.	85
Figure 5.6	Producing a new pose of the horse with the original ARAP, SR-ARAP and our SD-ARAP. The leftmost column shows the embedded skeleton, source mesh and the target skeleton row by row. While the first row of other columns shows the deformed meshes, other two rows zooms to the highlighted regions.	86

Figure 5.7	Producing a new pose of the wolf figure with the original ARAP, SR-ARAP and our SD-ARAP. The leftmost column shows the embedded skeleton and source mesh row by row. While the first row of other columns shows the deformed meshes, the second row zooms to the highlighted regions.	87
Figure 5.8	Various poses generated by deforming the source mesh in Figure 5.4 towards the target skeletons shown in the left column with the resulting meshes. Right column zooms to the highlighted regions on the results.	88
Figure 5.9	Twisting a box by rotating the skeleton edge as shown. The embedded skeleton and source mesh are given in the top row. Middle row shows the results of rotating the edge for 90° and 180° , The last row shows the results of defining 60° bending in addition to the rotations.	89
Figure 5.10	Rotating the right arm of a human around its corresponding bone without suffering from the candy-wrapper artifact. Result shown in different views.	89
Figure 5.11	Skinning weights computed with heat bone method [3] as a color map for the cylinder. Its source skeleton consists of two bones.	90
Figure 5.12	Deforming the cylinder shown in Figure 5.3 with various bending angles using LBS, DQS, FAST, DDM, and our SD-ARAP. Please note that we did not add any abstract handle for FAST as we obtained the best results in that way. The top row shows the target skeletons for various bending angles column by column. Other rows show the deformation results for LBS, DQS, FAST, DDM, and our SD-ARAP, respectively.	91
Figure 5.13	Skinning weights computed with heat bone method [3] as a color map for the human shape whose skeleton consists of 16 bones.	92

Figure 5.14	Some poses shown at the top and bottom rows of 5.8 are obtained by deforming the source mesh given in 5.4 by using different skinning methods. Volume loss artifacts still occur with the skinning methods around the right armpit of the human while the proposed method results in more natural deformations around those areas. . . .	92
Figure 5.15	The box shown in Figure 5.9 are twisted 180° by using different skinning methods. The proposed method is significantly more successful than skinning methods to handle twisting near joints without requiring any blend weight painting and parameter tuning.	93
Figure 5.16	Reconstructing some poses in the FAUST dataset. The source mesh is shown in Figure 5.4. Target skeletons are given in the first and transferred meshes in the second column. Ground-truth poses from the dataset are shown in the third column. Displacement errors $\Delta_i/l_{N(i)}$ between the approximated and ground-truth mesh vertices are shown in the last column as a color code. Average displacement errors on the overall surfaces of meshes in terms of the given metric are 0.189, 0.192, and 0.182 from top to bottom.	96
Figure 5.17	Results of transferring human mesh from FAUST dataset to the skeleton obtained from CMU MoCap dataset for some of the frames. The motion was captured at 120 Hz framerate.	98

LIST OF ABBREVIATIONS

ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
ARAP	As-Rigid-As-Possible
BiCGSTAB	Bi-Conjugate Gradient Stabilized Method
DDM	Direct Delta Mush
DQS	Dual-Quaternion Skinning
FAST	Fast Automatic Skinning Transformationsh
GUI	Graphical User Interface
IK	Inverse Kinematic
LBS	Linear Blend Skinning
MCF	Mean Curvature Flow
SD-ARAP	Skeleton-Driven As-Rigid-As-Possible
SR-ARAP	Smooth Rotation enhanced As-Rigid-As-Possible
SVD	Singular Value Decomposition



CHAPTER 1

INTRODUCTION

1.1 Motivation

Curve-skeleton and surface are two primitives related to a shape. Different poses of a shape are usually represented with manifold meshes registered to each other vertex-by-vertex. Curve-skeleton is an abstraction capturing geometric and topological properties of a shape. We plausibly expect for the curve-skeleton of a shape to be same throughout every pose of a character. However, existing curve-skeleton extraction algorithms produce different skeletons for different poses of the same character: The number of bones, the number of vertices, and even how those primitives are connected are usually different. Given a curve-skeleton (called source skeleton) along with a mesh representing a pose of a character (called source shape), we should be able to "transfer" the same skeleton to the other poses (called target shapes). In the first part of this study, we address this skeleton transfer problem: Instead of computing curve-skeletons of other poses of a character, we directly transfer curve-skeleton of a source shape to the target shapes, such that the transferred skeleton will be registered to the source skeleton vertex-by-vertex. We demonstrated versatility of our algorithm on many different test cases, also verified our transfer method is much more fast than skeleton extraction algorithms.

Reversely, we can have a closed manifold mesh representing a shape and only skeletons representing other poses of this shape. The second part of this study addresses this inverse problem called mesh transfer: Given simple curve-skeletons consisting of only a few bones (called target skeletons) along with an initial template shape (called source shape), transfer the surface of the source shape to the target skeletons

to obtain new poses of the source shape represented by these target skeletons. We also propose a solution to embedding each target skeleton into the source shape. To do this, we first compute curve-skeleton of the source shape by using an existing skeleton extraction algorithm and match the target skeleton with the curve-skeleton before transferring the surface. While the embedding part of the algorithm can be used for character rigging purposes, the second part of the algorithm can be used as a skeleton-based skinning method. We also show with various test cases that our shape transfer method produces more accurate results than classical and recent skinning methods when used as a character skinning method.

1.2 Contributions

The study consists of two independent parts: Skeleton transfer, and mesh transfer, which were mentioned in 1.1.

The contributions for the skeleton transfer part are as follows:

1. A fast curve-skeleton transfer method which does not require skeleton computation for each target mesh from scratch, preserves connectivity and complexity of the source skeleton and works even if the target meshes have degeneracies up to some extent. Meshes are not necessarily watertight and in fixed topology, and also can be disconnected triangle soups or point clouds.
2. Two novel error metrics allowing us to measure the accuracy of centeredness property of a curve skeleton inside of tubular structures quantitatively and to compare the results with other methods in a more systematic way. To our knowledge, no quantitative error metric was proposed in previous studies to assess centeredness property of curve-skeletons.

The contributions for the mesh transfer part are as follows:

1. A user-friendly framework to create new poses of an articulated human or non-human character with two basic inputs: a target skeleton (from any source such as motion capture, sketching) and the source template shape.

2. A skeleton embedding approach which can also be used as an automatic rigging method.
3. A method to augment the source mesh for a novel volume and detail preserving as-rigid-as-possible deformation.

1.3 Organization of the Thesis

In Section 2, background information necessary to understand our methods proposed to solve skeleton transfer and mesh transfer problems are provided. In Chapter 3, we briefly explain our solution to skeleton transfer problem. In Chapter 4, we explain our solution to the first part of the mesh transfer problem which deals with embedding an IK skeleton into a template shape. In Chapter 5, we explain our solution to the second part of the mesh transfer problem, which is about ARAP deformation based on skeleton. Please note that each of these three chapters are self-contained like a paper: We briefly introduce problem, provide related work, explain method, and provide results with a conclusion. In Chapter 6, we provide conclusions, last words, and possible future research directions about the thesis.



CHAPTER 2

BACKGROUND

2.1 3D Point-Set Matching

Suppose we have two sets each consists of n points in 3D: $\mathbf{P}$ and $\mathbf{Q}$. Further suppose the points in the sets are registered, i.e., for every point $\mathbf{p}_i$ in $\mathbf{P}$, there is one and only one point $\mathbf{q}_i$ in $\mathbf{Q}$. Our aim in this problem is to find a rigid transformation, i.e., a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ such that when applied to the points in $\mathbf{P}$, sums of squared distances between registered pairs will be minimum. This setup in 2D can be seen visually in Figure 2.1. A solution to this problem is given in [4] step-by-step but the solution will be summarized here too for the integrity of the thesis. The solution of this problem is used in Chapter 3.

The problem is formulated mathematically as follows:

$$(\hat{\mathbf{R}}, \hat{\mathbf{t}}) = \underset{\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3}{\operatorname{argmin}} \sum_{i=1}^n w_i \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2, \quad (2.1)$$

where $w_i > 0$ is the weight indicating the importance of matching i^{th} points.

Let us suppose we know the optimum rotation matrix $\mathbf{R}$ already and trying to find the optimum translation vector $\mathbf{t}$.

$$\begin{aligned} \frac{\partial}{\partial \mathbf{t}} \left[\sum_{i=1}^n w_i \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 \right] &= \sum_{i=1}^n 2w_i ((\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i) \\ &= 2\mathbf{t} \sum_{i=1}^n w_i + 2\mathbf{R} \sum_{i=1}^n w_i \mathbf{p}_i - 2 \sum_{i=1}^n w_i \mathbf{q}_i \\ &= 0. \end{aligned} \quad (2.2)$$

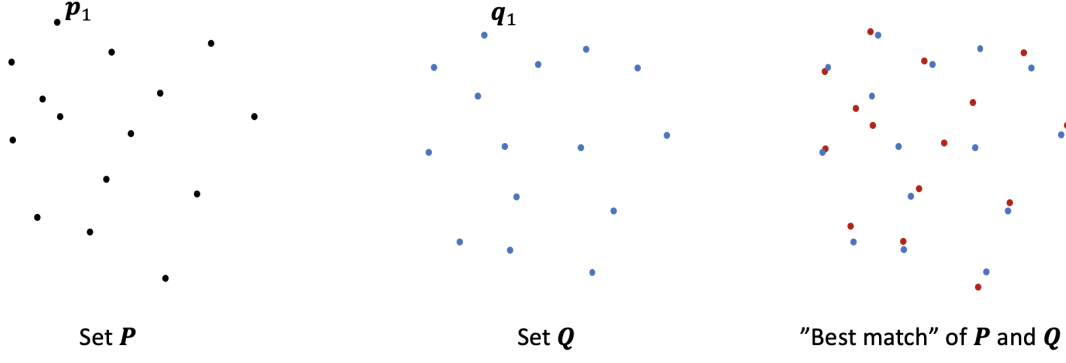


Figure 2.1: Point sets P and Q . The points are registered to each other. That is, for every point in set P , there is a point in set Q . Best match is fitting P onto Q with a rigid transformation so that the sum of squared distanced between registered points are minimum.

Let

$$\bar{\mathbf{p}} := \frac{\sum_{i=1}^n w_i \mathbf{p}_i}{\sum_{i=1}^n w_i} \quad (2.3)$$

and

$$\bar{\mathbf{q}} := \frac{\sum_{i=1}^n w_i \mathbf{q}_i}{\sum_{i=1}^n w_i}. \quad (2.4)$$

Inserting 2.3 and 2.4 into 2.2 gives us optimum translation as

$$\mathbf{t} = \bar{\mathbf{q}} - \mathbf{R}\bar{\mathbf{p}}. \quad (2.5)$$

Now, plug 2.5 into 2.1 and arrange,

$$\begin{aligned} \mathbf{R} &= \operatorname{argmin}_{\mathbf{R} \in SO(3)} \sum_{i=1}^n w_i \|\mathbf{R}(\mathbf{p}_i - \bar{\mathbf{p}}) - (\mathbf{q}_i - \bar{\mathbf{q}})\|^2 \\ &= \operatorname{argmin}_{\mathbf{R} \in SO(3)} \sum_{i=1}^n w_i \|\mathbf{R}\mathbf{x}_i - \mathbf{y}_i\|^2 \end{aligned} \quad (2.6)$$

where $\mathbf{x}_i := \mathbf{p}_i - \bar{\mathbf{p}}$ and $\mathbf{y}_i := \mathbf{q}_i - \bar{\mathbf{q}}$.

Writing 2.6 as and then after arrangements, we obtain,

$$\begin{aligned}
\mathbf{R} &= \operatorname{argmin}_{\mathbf{R} \in SO(3)} \sum_{i=1}^n w_i (\mathbf{R}\mathbf{x}_i - \mathbf{y}_i)^\top (\mathbf{R}\mathbf{x}_i - \mathbf{y}_i) \\
&= \operatorname{argmin}_{\mathbf{R} \in SO(3)} \left(-2 \sum_{i=1}^n w_i \mathbf{y}_i^\top \mathbf{R}\mathbf{x}_i \right) \\
&= \operatorname{argmax}_{\mathbf{R} \in SO(3)} \sum_{i=1}^n w_i \mathbf{y}_i^\top \mathbf{R}\mathbf{x}_i \\
&= \operatorname{argmax}_{\mathbf{R} \in SO(3)} \operatorname{tr}(\mathbf{R}\mathbf{X}\mathbf{W}\mathbf{Y}^\top)
\end{aligned} \tag{2.7}$$

where

$$\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_n], \mathbf{W} := \operatorname{diag}(w_1, w_2, \dots, w_n), \text{ and } \mathbf{Y} = [\mathbf{y}_1 \ \mathbf{y}_2 \ \dots \ \mathbf{y}_n].$$

Let $\mathbf{S} := \mathbf{X}\mathbf{W}\mathbf{Y}^\top$ and SVD of it be $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Inserting this to 2.7 yields

$$\mathbf{R} = \operatorname{argmax}_{\mathbf{R} \in SO(3)} \operatorname{tr}(\mathbf{\Sigma}\mathbf{V}^\top \mathbf{R}\mathbf{U}). \tag{2.8}$$

This equation is maximized only when $\mathbf{R} = \mathbf{V}\mathbf{U}^\top$. To handle also the case when $\mathbf{R}$ contains a reflection, we can write optimum $\mathbf{R}$ as,

$$\mathbf{R} = \mathbf{V} \operatorname{diag}(1, 1, \dots, \det(\mathbf{V}\mathbf{U}^\top)) \mathbf{U}^\top. \tag{2.9}$$

2.2 ARAP Deformation

Let $\mathbf{r}(u, v)$ be a parametrized surface with (u, v) as the parametric coordinates. The first fundamental form $\mathbf{I}(u, v)$ is defined with three coefficients in a 2×2 matrix as

$$\mathbf{I}_c = \begin{bmatrix} (\mathbf{r}_u \cdot \mathbf{r}_u) & (\mathbf{r}_u \cdot \mathbf{r}_v) \\ (\mathbf{r}_u \cdot \mathbf{r}_v) & (\mathbf{r}_v \cdot \mathbf{r}_v) \end{bmatrix}.$$

Metric properties of a surface such as length of arcs or areas of regions on the surface are completely determined by $\mathbf{I}(u, v)$. In a similar fashion, differential properties such as curvatures of the same surface is completely determined by the second fundamental form $\mathbf{II}(u, v)$ which can also be defined with three coefficients in a 2×2

matrix as $\mathbf{II}_c = \begin{bmatrix} (\mathbf{r}_{uu} \cdot \mathbf{n}) & (\mathbf{r}_{uv} \cdot \mathbf{n}) \\ (\mathbf{r}_{uv} \cdot \mathbf{n}) & (\mathbf{r}_{vv} \cdot \mathbf{n}) \end{bmatrix}$ where $\mathbf{n} = \frac{\mathbf{r}_u \times \mathbf{r}_v}{|\mathbf{r}_u \times \mathbf{r}_v|}$.

The change of the fundamental forms throughout a deformation gives a measure of stretching and bending [5],

$$E(\mathcal{S}, \mathcal{S}') = \iint_{\Omega} k_s \|\mathbf{I}'_c - \mathbf{I}_c\|_F^2 + k_b \|\mathbf{II}'_c - \mathbf{II}_c\|_F^2 dudv \quad (2.10)$$

where k_s is the weight of tangential distortion, k_b is the weight of distortion along surface normal $\mathcal{S}$ is the initial shape, $\mathcal{S}'$ is the deformed shape, $\mathbf{I}_c, \mathbf{II}_c$ are the first and second fundamental forms of the initial shape, and $\mathbf{I}'_c, \mathbf{II}'_c$ are the first and second fundamental forms of the deformed shape. Equation 2.10 is impractical to compute in a reasonable amount of time. Thus, it is approximated [5, 6] in the discretized domain as,

$$E(\mathcal{S}, \mathcal{S}') = \sum_{i=1}^N w_i \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2, \quad (2.11)$$

where

- $w_i = 1$,
- $\mathcal{N}(i)$ is the 1-ring neighbor of i_{th} vertex,
- $\mathbf{p}_i$ is the coordinates of i_{th} initial mesh vertex,
- $\mathbf{p}'_i$ is the coordinates of i_{th} deformed mesh vertex,
- $\mathbf{R}_i$ is the rigid transformation mapping 1-ring neighborhood of i_{th} initial mesh vertex (also called as a cell) to that of i_{th} deformed mesh vertex.

Given an initial solution to Equation 2.11, we can solve it to find "best" $\mathbf{p}'$ making the deformations as-rigid-as-possible [6]. An example cell for initial and deformed mesh is shown in Figure 2.2. If this deformation is rigid, then the following should hold:

$$\mathbf{p}'_i - \mathbf{p}'_j = \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j), \forall j \in \mathcal{N}(i). \quad (2.12)$$

Equation 2.12 should hold for every vertex in an as-rigid-as-possible deformation. This intuition leads us to express the global rigidity difference as an energy functional, as in Equation 2.11. The higher $E(\mathcal{S}, \mathcal{S}')$ is, the higher the nonrigidity of the deformation. Thus, one should find $\mathbf{p}'$'s making $E(\mathcal{S}, \mathcal{S}')$ minimum. Reproducing from [6], 2.11 is solved in a two stage optimization, where in the first stage the optimum local rotations $\mathbf{R}_i$'s are found, and in the second stage of each iteration, $\mathbf{p}'$'s are computed.

Let

$$E(\mathcal{C}_i, \mathcal{C}'_i) := w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2. \quad (2.13)$$

$$E(\mathcal{S}, \mathcal{S}') = \sum_{i=1}^N w_i \sum_{j \in \mathcal{N}(i)} w_{ij} E(\mathcal{C}_i, \mathcal{C}'_i), \quad (2.14)$$

To minimize $E(\mathcal{S}, \mathcal{S}')$, we should minimize $E(\mathcal{C}_i, \mathcal{C}'_i)$. Let $\mathbf{e}_{ij} := \mathbf{p}_i - \mathbf{p}_j$ and $\mathbf{e}'_{ij} := \mathbf{p}'_i - \mathbf{p}'_j$. First, optimum local rotations are found,

$$\begin{aligned} \operatorname{argmin}_{\mathbf{R}_i} E(\mathcal{C}_i, \mathcal{C}'_i) &= \operatorname{argmin}_{\mathbf{R}_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{e}'_{ij} - \mathbf{R}_i \mathbf{e}_{ij})^\top (\mathbf{e}'_{ij} - \mathbf{R}_i \mathbf{e}_{ij}) \\ &= \operatorname{argmin}_{\mathbf{R}_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{e}'_{ij}{}^\top \mathbf{e}'_{ij} - 2\mathbf{e}'_{ij}{}^\top \mathbf{R}_i \mathbf{e}_{ij} + \mathbf{e}_{ij}{}^\top \mathbf{e}_{ij}) \\ &= \operatorname{argmin}_{\mathbf{R}_i} \sum_{j \in \mathcal{N}(i)} -2w_{ij} \mathbf{e}'_{ij}{}^\top \mathbf{R}_i \mathbf{e}_{ij} \\ &= \operatorname{argmax}_{\mathbf{R}_i} \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{e}'_{ij}{}^\top \mathbf{R}_i \mathbf{e}_{ij} \\ &= \operatorname{argmax}_{\mathbf{R}_i} \operatorname{Tr} \left(\sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{R}_i \mathbf{e}_{ij} \mathbf{e}'_{ij}{}^\top \right). \end{aligned} \quad (2.15)$$

Let $\mathbf{S}_i := \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{e}_{ij} \mathbf{e}'_{ij}{}^\top$. Further, let $\mathbf{S}_i = \mathbf{U}_i \mathbf{\Sigma}_i \mathbf{V}_i{}^\top$ be its SVD. Then, the optimum rotation per cell is obtained as [4]

$$\mathbf{R}_i = \mathbf{V}_i \mathbf{U}_i{}^\top. \quad (2.16)$$

After obtaining optimum local rotations, Equation 2.11 is solved for $\mathbf{p}_i$'s which is

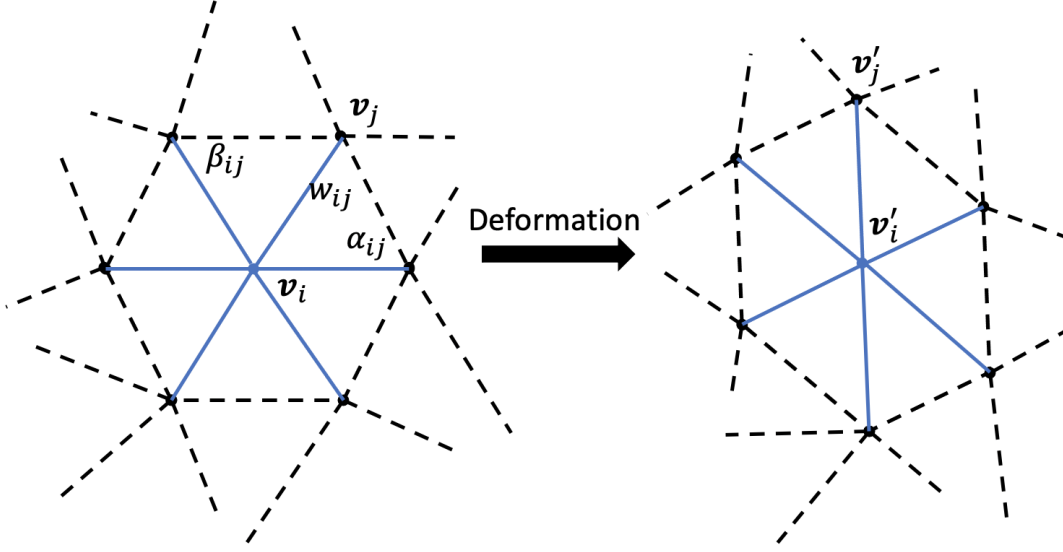


Figure 2.2: Left shows a cell in the undeformed initial triangular mesh and right shows the same cell after deformation.

called as the global step, as the coordinates of the mesh vertices are computed considering the whole shape instead of cells.

$$\begin{aligned}
 \frac{\partial E(\mathcal{S}, \mathcal{S}')}{\partial \mathbf{p}'_i} &= w_i \frac{\partial}{\partial \mathbf{p}'_i} \left[\sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \right. \\
 &\quad \left. + \sum_{j \in \mathcal{N}(i)} w_{ji} \|(\mathbf{p}'_j - \mathbf{p}'_i) - \mathbf{R}_j(\mathbf{p}_j - \mathbf{p}_i)\|^2 \right] \\
 &= 0.
 \end{aligned} \tag{2.17}$$

After arranging the equality we arrive at

$$\sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}'_i - \mathbf{p}'_j) = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} (\mathbf{R}_i + \mathbf{R}_j) (\mathbf{p}_i - \mathbf{p}_j) \tag{2.18}$$

At this point, w_i should be chosen as 1 as it should be proportional to cell area and we are already integrating over cells. To best approximate Equation 2.10, w_{ij} should be $\frac{1}{2}(\cot(\alpha_{ij}) + \cot(\beta_{ij}))$ [7]. In Figure 2.2 definitions of angles can be seen.

Equation 2.18 results in the following system equation with the weight selections

mentioned above:

$$\mathbf{L}\mathbf{p}' = \mathbf{b}. \quad (2.19)$$

Let $\mathcal{H}$ be the set of handle vertices whose transformations are completely determined by the user. Then the elements of Laplace operator for discrete meshes L is,

$$\mathbf{L}_{ij} = \begin{cases} w_{ij} & \text{if } i \neq j \text{ and } i \notin \mathcal{H} \\ -\sum_{j \in \mathcal{N}(i)} w_{ij} & \text{if } i = j \text{ and } i \notin \mathcal{H} \\ 1 & \text{if } i = j \text{ and } i \in \mathcal{H}. \end{cases} \quad (2.20)$$

$\mathbf{b}$ in Equation 2.19 is the right-hand-side of Equations 2.18. $\mathbf{p}'_i$ is $[p'_{i,x} \ p'_{i,y} \ p'_{i,z}]^\top$. Equation 2.19 is a sparse linear system, which can be solved by various sparse numerical solvers like *Eigen* or *TAUCS*.

2.3 Skinning

Skinning is binding the surface of a shape to some control structures allowing it to be deformed by using those in the field of digital geometry processing. Although researchers state some classifications about skinning in their articles, there is not an agreement on the taxonomy of skinning methods among researchers up to date. For example, Le et. al. [8] categorized skinning methods into three: Geometric, example-based, and physically-based. According to his classification, geometric methods are those not having any physical meaning but binding the surface to the control structures through some geometric and algebraic derivations. On the other hand, example-based methods learn how to deform the surface according to the control structures through a set of examples given to the system as the training set. Physically-based methods deform the surface according to some procedures derived by using the principles of classical mechanics.

As we did not provide any training set to our method, example-based skinning methods are irrelevant to our discussion. Physically-based methods require too many parameters like Lamé coefficients and/or Young's modulus [9] and we do not provide those parameters to our framework. Thus, physically-based skinning methods

are also out of scope of the thesis like example-based ones. Geometric methods, on the other hand, require only skinning blend weights, which can be computed via automatic methods making them candidate methods to be integrated into our framework. For this reason, we compared our method with geometric skinning methods. We will provide background information about geometric skinning methods in this section and provide the comparison results in 5.4.

2.3.1 Linear Blend Skinning (LBS)

The most basic skinning method is Linear Blend Skinning (LBS) [10]. It uses the following formula for deforming an initial mesh through control structures.

$$\mathbf{v}'_i = \sum_{j=1}^M w_{ij} \mathbf{T}_j \mathbf{v}_i \quad (2.21)$$

where

- $\mathbf{v}_i \in \mathbb{R}^{4 \times 1}$ is i^{th} initial mesh vertex in homogeneous coordinates, i.e., the last member is 1,
- M is the number of bones,
- $w_{ij} \in [0, 1]$ ($\sum_{j=1}^M w_{ij} = 1$) is influence of j^{th} bone on i^{th} vertex,
- $\mathbf{T}_j \in \mathbb{R}^{3 \times 4}$ is the rigid transformation j^{th} bone is undergoing,
- $\mathbf{v}'_i \in \mathbb{R}^{4 \times 1}$ is the deformed position of i^{th} initial mesh vertex.

Please note that control structures can be either a skeleton consisting of several bones embedded inside the character or a cage built around the character in geometric skinning. Cage-based skinning methods are not suitable for our framework, so in this work we mean skeleton-based geometric skinning methods by geometric skinning methods. Although Equation 2.21 does not limit number of bones that a

vertex can be influenced, it is limited by 4 in practice for enabling hardware implementation. Equation 2.21 can be written as,

$$\mathbf{v}'_i = \left(\sum_{j=1}^M w_{ij} \mathbf{T}_j \right) \mathbf{v}_i \quad (2.22)$$

from which we can interpret the deformation as first blending $\mathbf{T}_j$'s according to the weights to obtain a new transformation and applying it to vertex $\mathbf{v}_i$. This interpretation reveals the source of the main drawback of LBS: While $\mathbf{T}_j$'s are rigid transformations (A rotation and a translation only), $\sum_{j=1}^M w_{ij} \mathbf{T}_j$ is not a rigid transformation anymore which creates artifacts on the deformed shape [11]. This can best be seen

by an example. Consider, $\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ and $\mathbf{T}_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$ with weights $w_{i,1} = w_{i,2} = 0.5$.

While $\mathbf{T}_1$ and $\mathbf{T}_2$ are rigid transformations, $\mathbf{T} = 0.5\mathbf{T}_1 + 0.5\mathbf{T}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ is not a rigid transformation and it constraints every argument it takes to y-axis which causes artifacts.

As mentioned in the discussions above, the quality of the resulting deformation with LBS depends heavily on the chosen weights. In practice, the blend weights are still chosen manually: Rig artists adjust blend weights by painting the shape in programs like Maya or Blender. If we had incorporated LBS into our framework, we would not have the chance to adjust the weights manually. Thus, we do not prolong the manual weight adjustment discussion here. In the following section, we will briefly explain an automatic skinning weight computation method using heat-bone equation [3] with which we computed skinning weights in the comparisons.

2.3.2 Heat-Bone Weight Computation

The blend weights should smoothly vary across the surface of the shape so that the deformations will be smooth too, also they should also be summed up to 1 for

each mesh vertex and $w_{ij} \in [0, 1]$. Heat-bone method satisfies these requirements in a neat and natural way. We briefly explain the algorithm in this section which is reproduced from [3].

Let us assume the "temperature" at the skeleton is 1° and it "heats up" the surface of the shape. The temperature at any point on the surface of the shape gives us the blend weight at that point of the shape. Solving heat diffusion equation for volumetric shape gives us the exact weight, however, it takes too much time to solve it for volume making it impractical. Approximating the discretized version of the equation for triangular meshes still provides practical blend weights:

$$(\mathbf{L} + \mathbf{H})\mathbf{W} = \mathbf{HP} \quad (2.23)$$

where

- $\mathbf{L} \in \mathbb{R}^{N \times N}$ is discrete Laplacian,
- $\mathbf{H}$ is a diagonal matrix such that $\mathbf{H}_{ii} = \frac{A(i)}{d_S^2(\mathbf{v}_i)}$, $A(i)$ is the total area of the triangles incident to i^{th} mesh vertex and $d_S(\mathbf{v}_i)$ is the distance between i^{th} mesh vertex and closest point on the skeleton to it,
- $\mathbf{P}$ is a matrix such that $\mathbf{P}_{ij} = 1$ if the closest point on the skeleton to i^{th} mesh vertex is on j^{th} bone, and it "sees" this closest point, $\mathbf{P}_{ij} = 0$ otherwise,
- $\mathbf{W}$ is the weight matrix we are seeking such that $w_{ij} := \mathbf{W}_{ij}$.

Equation 2.23 is solved for each column of $\mathbf{W}$ separately by using BiCGSTAB [12]. Result of applying the method on a cylinder mesh is depicted in Figure 2.3.

2.3.3 Dual Quaternion Skinning (DQS)

Linearly blending bone transformations in LBS setting causes artifacts near joints, specifically, elbow-collapse effect for bending and candy-wrapper effect for twisting. Figure 2.4 explains the case better. Let us assume points $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ lies on the unit circle, each having blend weight $1/3$. When we compute $\mathbf{P}_{avg}$, average those points,

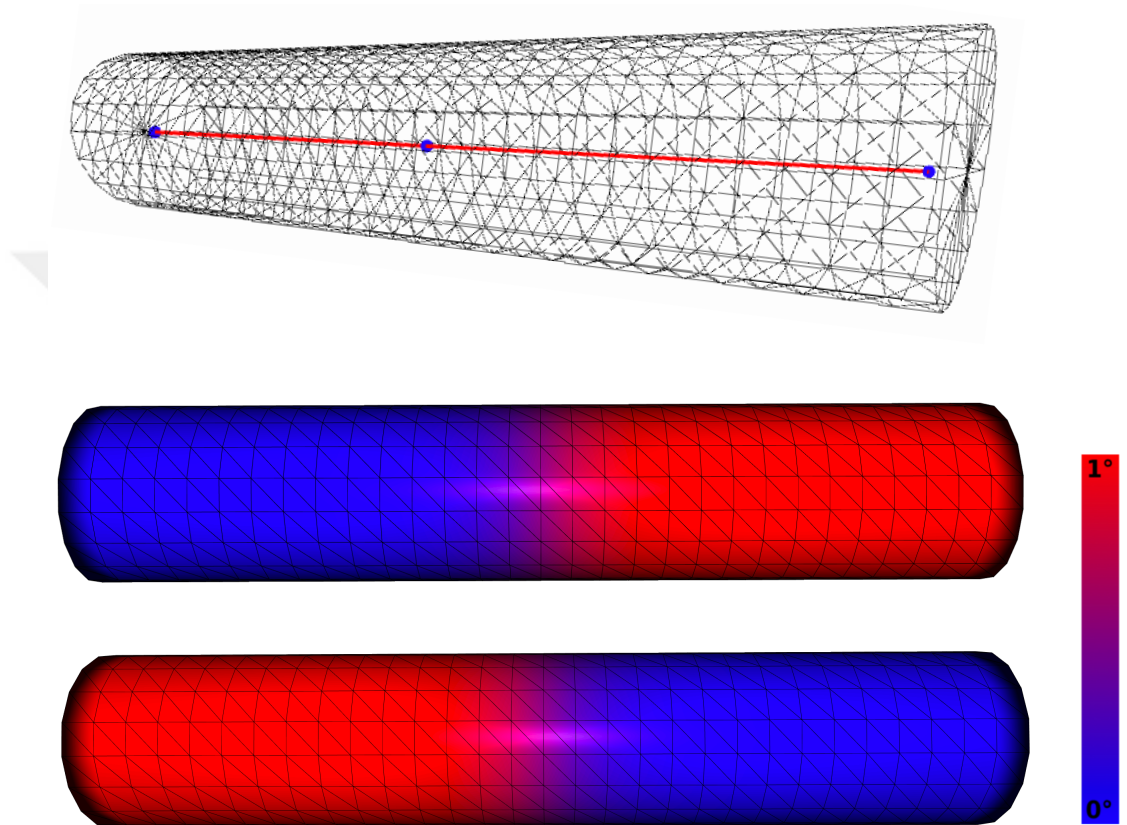


Figure 2.3: Top row depicts a cylinder mesh with a two bone skeleton embedded inside it. The bottom two figures show blend weights computed by the heat-bone weighting procedure.

we can see that it does not lie on the circle anymore but inside of it as in the left figure, which is the case not wanted in LBS. Instead, $\mathbf{P}_{avg}$ should lie on the unit circle so that artifacts will not occur as in the right figure. To achieve this, "angles" should be averaged instead of positions in Cartesian coordinates.

Averaging angles on the unit circle can be most easily and intuitively handled by using complex numbers. Let $e^{i\theta_1}$, $e^{i\theta_2}$, and $e^{i\theta_3}$ be complex numbers representing $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ in the complex plane, respectively. We can obtain $\mathbf{P}_{avg}$ by first converting $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ to $e^{i\theta_1}$, $e^{i\theta_2}$, and $e^{i\theta_3}$, computing $e^{i(\theta_1+\theta_2+\theta_3)/3}$ and then converting it back to the Cartesian coordinates.

Now assume we want to apply the same procedure in 3D this time. Assume $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ lies on the unit sphere, and we want $\mathbf{P}_{avg}$ to lie on the unit sphere too. We can use quaternions, an extension of complex numbers, for this purpose. A quaternion q is represented as $\mathbf{q} = w + ix + jy + kz$ where $w, x, y, z \in \mathbb{R}$ and i, j, k are quaternion units. In a similar fashion how we convert $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ to complex numbers, they can also be converted to quaternions along with bone transformations. After averaging the quaternions, when we convert the result back to the Cartesian coordinates, we obtain the average position of $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ on the unit sphere.

Nevertheless, directly using quaternions with Equation 2.21 is not enough. This is because while $\mathbf{T}_j$ consists of both rotations and translations, quaternions represent only the former. Dual numbers equip us with translations instead of rotations. A dual number $\hat{a}$ is represented as $\hat{a} = a_0 + \varepsilon a_\varepsilon$ where a_0 is the non-dual part, a_ε is the dual part and ε is the dual unit satisfying $\varepsilon^2 = 0$.

By defining dual-quaternion $\hat{\mathbf{q}}$ as $\hat{\mathbf{q}} = \hat{w} + i\hat{x} + j\hat{y} + k\hat{z}$, we achieve combining the representation of rotations property of quaternions and representation of translations property of dual numbers. DQS takes advantage of this representation: Bone transformations of mesh vertices are first converted to dual-quaternions, the weighted averaging is performed on dual-quaternions, and the result is converted back to the Cartesian coordinates. As the operations are computed on a sphere rather than a line, elbow-collapse and candy-wrapper artifacts are expected not to occur. Nevertheless, it causes bulges near bending joints instead of elbow-collapses, also for

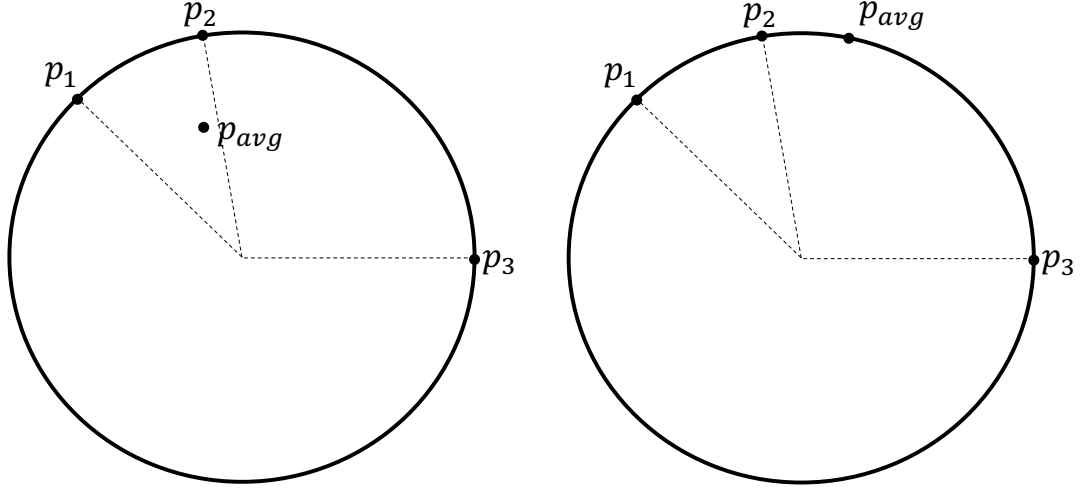


Figure 2.4: Left figure shows averaging points linearly which results in a point inside of the unit circle. Right figure shows averaging points by using angles which results in a point on the unit circle (Reproduced from [1]).

candy-wrapper effects not to occur in twist cases, the weights should be chosen very carefully (See Section 5.4. This section is mostly reproduced from [1], and those who are interested more information about quaternions and dual-numbers are suggested to read the related article.

2.3.4 Fast Automatic Skinning Transformations (FAST)

Fast Automatic Skinning Transformations proposed by Jacobson and his colleagues [13] incorporates ARAP framework into skinning. Equation 2.21 can be expressed in matrix form as $\mathbf{V}' = \mathbf{MT}$. Also, the user can define some constraints on transformations which can be expressed as $\mathbf{M}_{eq}\mathbf{T} = \mathbf{P}_{eq}$ enabling users to add abstract handles by adding extra columns to $\mathbf{T}$. Equation 2.14 can also be expressed in matrix form. Those who are interested in the full derivation can see it in [13]. Then, $\mathbf{V}'$ incorporated into $E(\mathcal{S}, \mathcal{S}')$ and the optimization problem is stated as,

$$\begin{aligned} & \underset{\mathbf{R}, \mathbf{T}}{\operatorname{argmin}} E(\mathcal{S}, \mathcal{S}') \\ & \text{subject to } \mathbf{M}_{eq}\mathbf{T} = \mathbf{P}_{eq}. \end{aligned} \tag{2.24}$$

Solution of this optimization problem is similar to that of ARAP, and is briefly ex-

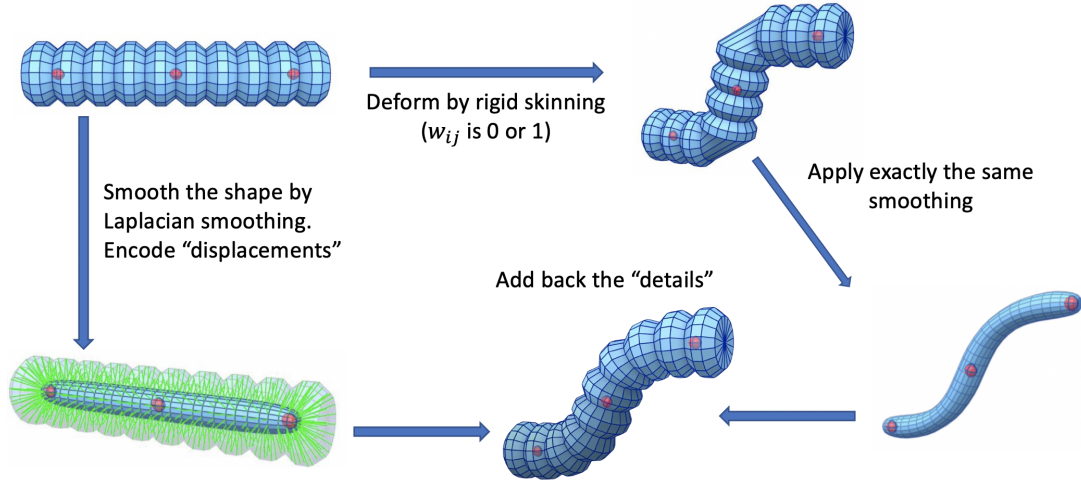


Figure 2.5: Visual summary of Direct Delta Mush (Figures are reproduced from presentation of [2]).

plained in [13].

Without abstract handles, FAST reduces to standard LBS so it is suggested to define extra handles. Comparisons fo our method with this method with related discussions are provided in Section 5.4.

2.3.5 Direct Delta Mush (DDM)

Direct Delta Mush proposed by Le et. al. [2] is the most suitable skinning method for our framework because it does not require any blend weight (manual or automatic) unlike its predecessors. It first smooths out the initial mesh with well-known Laplacian smoothing yielding "mush", and the displacements of mesh vertices before and after smoothing are kept. Also, initial mesh is deformed via rigid skinning, in which each mesh vertex is simply attached to the closest bone to it, and transferred with the same transformation of the bone it is attached to. Exactly the same smoothing is applied to the deformed mesh. Then, the displacements kept initially while smoothing the initial mesh is added back to the deformed shape. The process is summarized in Figure 2.5.

CHAPTER 3

3D SKELETON TRANSFER BETWEEN MESHES AND CLOUDS

We present a curve-skeleton extraction method in the form of curve-skeleton transfer from the source shape to the target. The shapes we deal with need to be in correspondence. They are discretized as meshes embedded in 3D and are not necessarily watertight. They can even be in arbitrary topology, and furthermore be disconnected point clouds. Our method associates the source skeleton with the source shape and computes the optimal rigid transformations towards the corresponding query shape points. This transformation for each source skeleton point is followed by a post-processing operation. Fixed skeleton connectivity maintained throughout the transfer guarantees that the resulting skeleton is as complex as the source skeleton, e.g., no redundant branches and complicated joint hierarchies. The performance and versatility of our method as well as its advantages over a direct skeleton extraction method are demonstrated.

3.1 Introduction

The set of points having more than one closest point to the boundary of a shape is called the medial-axis or skeleton. Since its first introduction in [14], many of its variants have been developed and used in shape analysis, retrieval, deformation, and many other applications.

One problematic issue with the initial version of the medial-axis is being too sensitive to small variations on the boundary. Even with a boundary with small noise, the produced medial-axis is too complex to be processed or to be practical especially for 3D shapes because medial-axis of a 3D shape not only consists of curves, but also

contains 2D surfaces. Such structures are unnecessarily complex for some applications. To promote usage of medial-axis, 1D curve representations of medial-axis, called curve-skeletons, have been introduced. In spite of their differences, the terms curve-skeleton and skeleton are used interchangeably in the literature. Throughout the paper, we refer to curve-skeleton by the phrase “skeleton”.

Numerous methods have been proposed for computing medial-axes of shapes. In cases where there are several shapes in correspondence such that they represent different poses of the same entity at different times, e.g., animation sequences, some recent techniques like [15] allow transfer of medial-axis computed for one shape to other shapes. Analogously, many methods have been proposed for extracting curve skeletons of shapes, however, no study has been done specifically for transferring an already extracted curve-skeleton between shapes in correspondence. There exist common scenarios where this correspondence information is available, e.g., animation sequences, simulated models, or registered models. Computing the skeleton for each shape directly in such scenarios with the existing methods has three disadvantages. First, the extraction process for all the shapes may take a long time as each one is treated independently in isolation. Second, it is highly unlikely that the resulting skeletons will be registered to each other vertex-by-vertex, however, this missing one-to-one correspondence property may also be desirable. Third, many of the skeleton extraction methods require that the mesh, arguably the most common shape representation, is watertight, i.e., possess no holes, voids, or non-manifold structures, and complete, and represented by fine tessellations. Majority of these direct methods are also likely to suffer from producing artifacts such as redundant branches.

In this study, we propose a simple yet accurate method that enables curve-skeleton transfer between shapes in correspondence. To do this, we need an already extracted skeleton, which we refer to as the source skeleton throughout the paper. Firstly, for each source skeleton vertex, we associate a set of vertices of the mesh from which it is extracted. This mesh is called the source mesh throughout the paper. The correspondence process to achieve this association is explained in detail in Section 3.3.1. Subsequently, for each skeleton vertex, the optimum rigid transformation that maps its associated vertices to the corresponding point set of the target shape is computed

separately. Note that, the target shape in our versatile method admit many representations but, without loss of generality, we assume mesh representation hereafter. Note also that this target mesh is also referred to as the input mesh since we want to extract the skeleton for this query mesh. Optimum transformation computation results in a rotation matrix and a translation vector for each skeleton vertex. The resulting transformations are applied to their skeleton vertices one-by-one, transferring the skeleton vertices to the input mesh. This transfer process is explained in Section 3.3.2. Finally, 1D curve skeleton is smoothed during the post-processing stage whose details are given in Section 3.3.3.

The study has two main contributions.

- A curve-skeleton transfer method which works quite fast, does not require skeleton computation for each target mesh from scratch, preserves connectivity and complexity of the source skeleton and works even if the target meshes have degeneracies up to some extent. Meshes are not necessarily watertight and in fixed topology, and furthermore can be disconnected triangle soups or point clouds.
- Two novel error metrics that allow us to measure the accuracy of centeredness property of a curve-skeleton inside of tubular structures quantitatively and to compare the results with other methods in a more systematic way. To our knowledge, no quantitative error metric was proposed in previous studies to assess centeredness property of curve-skeletons.

3.2 Related Work

Unlike medial-axis, curve-skeleton is not formally defined. Although some attempts were made to mathematically define what a curve-skeleton is, e.g., [16], these definitions have not been accepted widely by the community. Due to the lack of formal definition, there has not been an exact algorithm to compute the curve-skeleton of a shape. However, several properties that should be satisfied by a curve-skeleton is listed at least: Topology preserving, invariance under isometric transformations, reconstructability, thinness, centeredness, reliability, component-wise differentiation,

connectedness, robustness, smoothness, and hierarchicalness. We will not explain what those properties mean for the brevity of the discussion, those who are interested in brief definitions and analysis of those properties are encouraged to read the survey [17]. Identification of the properties does definitely not imply that a curve-skeleton should satisfy all of them. In fact, it is impossible for a curve-skeleton to satisfy all of the properties because some of them are conflicting. For example, centeredness and robustness are conflicting properties which further implies a curve-skeleton does not have to be a subset of the medial-axis invalidating the definition in [16]. Instead of fulfilling all of the properties, non-conflicting and different subsets of them are satisfied for different applications. This has led to development of numerous techniques and algorithms to compute the curve-skeleton of a shape.

First computing medial-axis and then pruning it to obtain the curve-skeleton is one of the most widely used strategies for computing curve-skeleton of a shape, [18] and [19] are recent examples to algorithms adopting this strategy. Low robustness to surface noise and computational expensiveness are two major drawbacks of such approaches. Some techniques to produce robust skeletons have also been developed. For example, a generalized potential field generated by charges placed on the surface of the object is used by [20] in order to define a repulsive force that pushes the surface towards centering skeleton. Even though curve-skeletons produced by such techniques are almost excellent in terms of robustness and smoothness, it is challenging to trace the field discontinuities due to numerical issues and their computational cost is quite high. Exploiting curve-skeleton extraction in 2D which is a relatively well-studied area is another strategy for 3D curve-skeleton extraction. For example in [21], the shape is first orthographically projected onto 2D scenes, its curve-skeleton is computed in 2D domain, and it is projected back to 3D. Although robust curve-skeletons can be computed with a low computational cost with this method, it may be out of the shape and a large number of projections may be required. Contraction-based methods have been becoming more popular in curve-skeleton extraction area. [22] is a good representative of this family, which was also used in this study to extract the source skeleton. Major drawbacks of such methods are they usually require many parameters to tune, quite different skeletons can be generated for different discretization of the same surface, and they usually assume

that the surface is watertight, possessing no holes. Other than these, many different methods were developed for curve-skeleton computation from probabilistic methods [23] to the ones using shape diameter function [24]. Curve-skeleton extraction methods are certainly not limited the mentioned ones, those who are interested are encouraged to read the surveys [17] and [25]. Both medial-axis and curve-skeleton computation has drawn noticeable attention up to now, and this trend will most probably continue in the future due to the diverse range of applications taking advantages of them.

One of the applications in which skeletons are quite useful is the compact encoding of animation sequences. A naïve approach that stores each animation frame model explicitly with the full mesh details would be too redundant and expensive in terms of streaming, storage, and processing. Extracting and rigging the skeleton of a single model is a much more efficient alternative for the faithful reproduction of the sequence poses [26, 27, 28]. These methods have been improved recently, at the expense of increased computational load, by automatic pruning of redundant bones [29] and introduction of deformable medial spheres [15]. The input to this family of methods is a set of example poses with temporal consistency, i.e., different poses with the same fixed connectivity. We should note that while most of the sequences come with this information naturally, e.g., meshes produced by animations, deformations, simulations, and registrations, one can also employ fully automatic dense correspondence algorithms to obtain this knowledge [30, 31, 32, 33, 34].

Another usage of skeletons is to represent static models efficiently for various purposes such as animation or deformation. Given a static character mesh and a generic template skeleton, for instance, [3] adjusts the skeleton so it can fit into the mesh properly and then attaches it to the surface with the appropriate blend skinning weights that derive the animation afterwards. Electors voting approach [35] performs similarly for this so-called animation re-targeting task except it extracts the skeleton automatically instead of fitting a template. Regardless of how skeleton is placed in this family of animation re-targeting methods, one needs to bind the skeleton to the surface vertices with the appropriate blend skinning weights to enable the animation of the skeletonized object. This is either a manual burdensome process or an automated process up to some accuracy issues [1, 13]. Note that, if the

source mesh is weighted properly then the skeletonized object through our transfer is already equipped with this important weight information since the new skeleton comes with its binding to the new surface through perfect correspondences.

There are several other useful static shape representations through skeletons. Shape retrieval application, for instance, greatly benefits from skeletons while comparing the similarity between a query model and many database models without getting confused and slowed down by the geometric factors [36, 17, 37]. Sometimes, however, these geometric factors may be useful, e.g., skeletons of a horse and a camel are similar but they are geometrically different and should not be matched [38]. Consequently, one needs to keep this trade-off in mind while employing skeletons for this task.

Several applications like the ones mentioned above usually use a series of poses of the same model. A computationally lightweight method for computing curve-skeletons for all of the poses of a model such that the resulting curve-skeletons will be topologically equivalent and have the same number of skeletal elements is lacking. At this point, one may attempt to use medial-axis deformation based methods like [39] and [15]. Nevertheless, such methods are proposed for generating surfaces for animating mesh sequences and using them for curve-skeleton transfer comes with a few important disadvantages. For example, in the approach in [15], which was based on [40], the two stage optimization process is computationally costly. Even for a small number of medial primitives (~ 50), the computation takes dozens of seconds yet it takes less than a second with the method proposed here for a large number of skeleton vertices (~ 500) (See Section 3.4.9). Moreover, the proposed method allows curve-skeleton transfer when the input subject changes, e.g., transfer from a male to a female, or from a slim subject to a fat one (See Section 3.4.3). However, it is unclear how deformation based methods behave when used for such a purpose. As mentioned, medial-axis deformation methods and this study bring solutions to different problems. If the objective is to embed surface information into a medial structure and reconstruct the animated surface, then deformation methods like [15] should be favoured. Nevertheless, if the objective is to compute curve-skeletons for a series of poses of the same model, or even a different model up to some extent (See Section 3.4.3), in a very fast manner, it is advantageous to use our

transfer approach.

3.3 Method

The method we propose consists of three stages. A correspondence between the source mesh and its skeleton is established in the first stage which we call as surface correspondence. Once the surface correspondence is established, the optimum rigid transformations transferring the skeleton of the source mesh to the possible skeletons of the other input meshes are estimated. Skeleton transfer stage is followed by a post-processing stage in which the estimated skeletons -which are 3D medial curves- are smoothed by using Laplacian smoothing.

3.3.1 Surface Correspondence

In order to transfer to the input mesh, we need a source mesh and its skeleton, which can be created manually or computed by any skeleton extraction method. We should make sure the skeleton extraction method we are using can extract the skeleton as correct as possible. We used the Mean Curvature Flow (MCF) method proposed in [22] to extract the source skeleton from the source mesh. The source skeletons used in this study are shown in Figure 3.1. Please note that the part of the extracted source skeleton in the right foot of the human mesh does not precisely satisfy the medial-axis property. Consequently, the transferred skeletons also do not satisfy the medial-axis property precisely in that region of the meshes (Section 3.4).

A correspondence between the source mesh and the source skeleton should be established so that the source skeleton can be transferred to the input meshes using this correspondence. We continue to provide details for this correspondence computation.

It is assumed that the source skeleton consists of piece-wise linear 1D curves wherein joints are represented as vertices (which we call as skeleton vertices) and if any two vertices are connected, they form a line segment. For the i^{th} vertex of the source skeleton $\mathbf{s}_i = [s_{i_x} \ s_{i_y} \ s_{i_z}]^T$, we first determine the number of mesh vertices N_i with

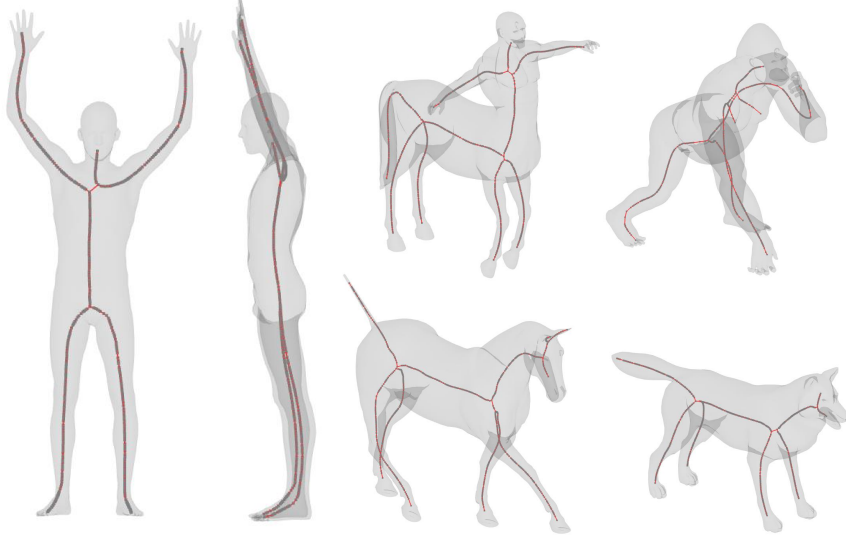


Figure 3.1: Source skeletons to be transferred (human in two views).

which we are going to establish correspondence. Let $(s_i, v_{i,1})$ be a pair of vertices where $v_{i,1} = [v_{i,1x} \ v_{i,1y} \ v_{i,1z}]^T$ is the closest vertex of the source mesh to s_i in terms of Euclidean distance. Let us denote the distance between s_i and $v_{i,1}$ as r_i . We define r_{min} as the smallest of such distances for all the skeleton vertices, i.e., $r_{min} = \min\{r_1, r_2, \dots, r_N\}$ where N is the total number of vertices in the source skeleton.

Let us consider a hypothetical case in which s_i is inside of a cylindrical shape with height h . $v_{i,1}$, r_i , and the setup is depicted at the top row of Figure 3.2. The top view of the cross section of the cylinder is shown at right to better see the setup. s_i is corresponded with a number of vertices chosen from the surface of the cylinder. At the bottom row of Figure 3.2, a thicker cylinder is given such that $r'_i > r_i$. It is clear that the surface area of the cylinder increases with increasing r_i if $h' \geq h$. As the cylinder gets thicker, h' tends to be larger than h implying that the surface area non-linearly depends on r_i . Consequently, N_i should be increased non-linearly with increasing r_i , otherwise, lacking number of corresponding vertices leads to sub-optimal skeleton transfer results. Besides, it is not a proper way to directly use Euclidean distance between s_i and $v_{i,1}$ while deciding N_i because the scales of meshes may differ. Using the ratio $\rho_i := r_i/r_{min}$ instead of r_i by-passes this problem.

This leads us to choose N_i as

$$N_i = \begin{cases} 8 + \lfloor 8\rho_i \rfloor, & \text{if } \rho_i \leq 3 \\ 8 + \lfloor 12\rho_i \rfloor, & \text{if } 3 < \rho_i \leq 6 \\ 8 + \lfloor 16\rho_i \rfloor, & \text{if } 6 < \rho_i. \end{cases} \quad (3.1)$$

When we do not use any constant term and ρ_i is close to 1, N_i becomes small for a robust transformation estimation. Thus, we added a constant term to the equation. To reflect the non-linearity between ρ_i and N_i mentioned at the previous paragraph, we divided the equation into three cases and we increase the coefficient of ρ_i with increasing ρ_i . The value of the initial term which is 8 and the coefficients of ρ_i which are 8, 12, and 16 are determined through various experiments covering many different cases. Please note that there can exist different procedures for determining N_i , the one presented in this study produces fairly accurate results which will be discussed thoroughly in Section 3.4.

Determination of N_i is followed by choosing N_i vertices on the mesh. It is better to choose the corresponding vertices on the rigid body part of the source mesh to which $\mathbf{s}_i$ is related so that the estimated rotation matrix and translation vector will be consistent. Choosing corresponding vertices on an unrelated body part would lead to suboptimal rotation matrix and translation vector estimation. Nevertheless, choosing simply closest N_i mesh vertices to $\mathbf{s}_i$ may result in corresponding vertices clustered in a small area which would provide redundant information or again lead to suboptimal rotation matrix and translation vector estimation. The corresponding vertices of $\mathbf{s}_i$ should lie on the body part of the mesh to which $\mathbf{s}_i$ is related as much as possible while the distance between them should be as large as possible. To achieve this, we choose N_i vertices on the mesh which are closest to $\mathbf{s}_i$ and they do not belong to each others' 1-ring neighbourhoods. More formally, let $\mathcal{N}(\mathbf{v})$ be the set of vertices in 1-ring neighbourhood of vertex $\mathbf{v}$ and $C_i = \{\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \dots, \mathbf{v}_{i,N_i}\}$ be the set of correspondent points of $\mathbf{s}_i$, we choose the N_i closest vertices on the source mesh to $\mathbf{s}_i$ such that $\mathcal{N}(\mathbf{v}_{i,l}) \cap \mathcal{N}(\mathbf{v}_{i,m}) \not\subset C_i$ for $l \neq m$. The selection routine can easily be implemented first by sorting the mesh vertices according to their distances to $\mathbf{s}_i$ and then constructing the set C_i .

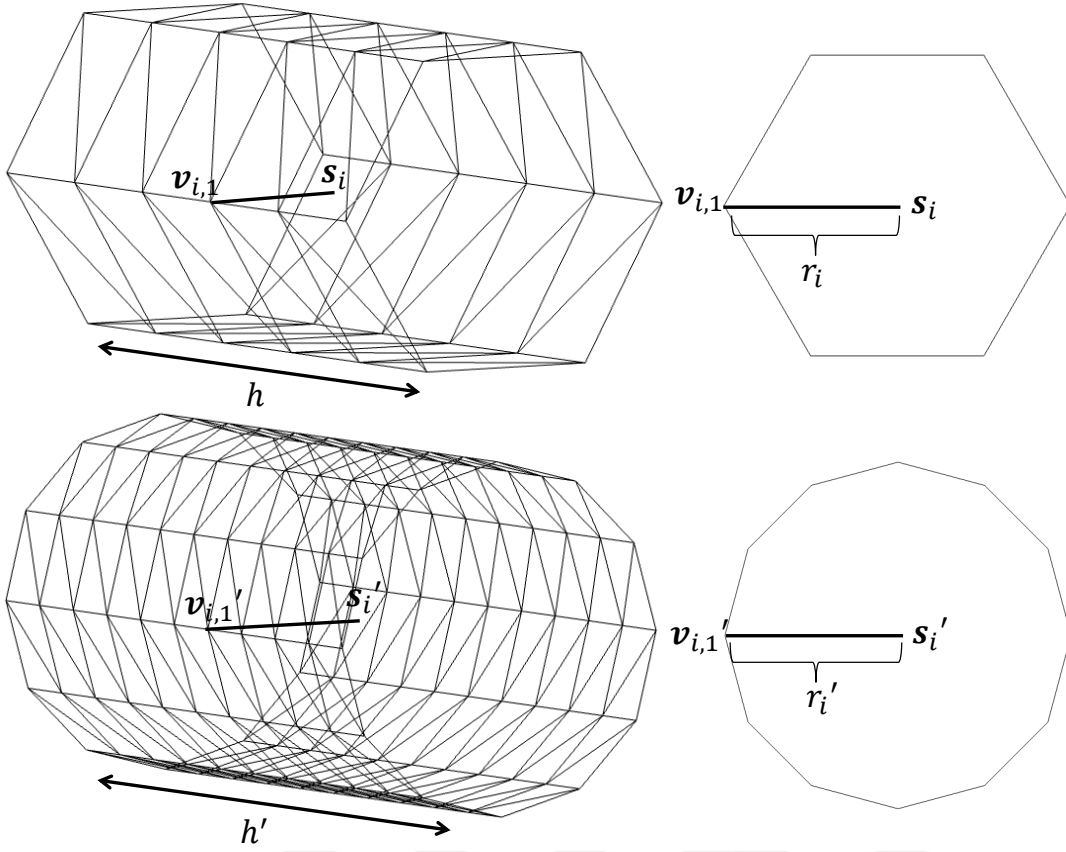


Figure 3.2: Two cylinders represented as triangular meshes. The cylinder at bottom is thicker than the one at top, that is, $r'_i > r_i$, and also $h' \geq h$. Subfigures at right show the top views of the cross sections of the cylinders to better depict the setup. As r_i increases, the area of the cylinder increases non-linearly.

Before finishing this subsection, we would like to justify the need of making the number of corresponding vertices for a skeleton vertex s_i dependent on ρ_i and making them not to belong to each others' 1-ring neighbourhoods. As mentioned, if we correspond s_i just with a number of mesh vertices closest to it, then the corresponding vertices have the risk of clustering in a small area on a part of the mesh. This may further lead to distorted transferred skeletons especially between inter-subjects. Figure 3.3 depicts an example of this phenomenon. In this example, the clustered vertices for the skeleton vertices going through the torso of the human were all selected from the back of the subject. When the skeleton was transferred to a mesh of another subject, the skeleton segment in the torso became distorted, given in the top-right subfigure. This was due to the fact that, as all corresponding

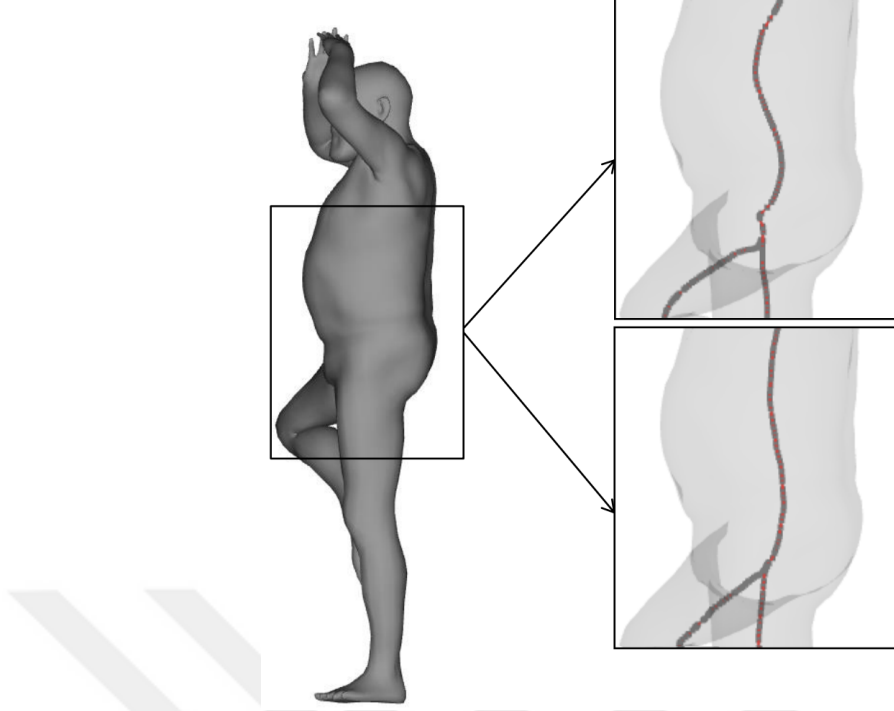


Figure 3.3: An inter-subject skeleton transfer case (from the source in Figure 3.1 to this different human) showing the resulting skeleton when just the closest mesh vertices to skeleton vertices were chosen as corresponding vertices (top-right) and when ρ_i parameter was used (bottom right).

points for the distorted part of the skeleton were from the back, the estimated transformations for those skeleton vertices cannot capture the surface on the stomach of the mesh. Resulting skeleton when we used ρ_i alleviates this problem as shown at the bottom-right subfigure.

3.3.2 Skeleton Transfer

While transferring the source skeleton to a target with correspondence, its number of vertices and connectivity of vertices are preserved so that we obtain a skeleton for the target mesh registered to the source skeleton vertex-by-vertex. To achieve this, we first compute the optimum rotation matrix $\hat{\mathbf{R}}_i$ and translation vector $\hat{\mathbf{t}}_i$ that map the corresponding mesh vertices C_i of the source skeleton vertex s_i to the corresponding vertices D_i of the input mesh, and then apply the computed transformation

to $\mathbf{s}_i$ to find the corresponding skeleton vertex of the input mesh $\hat{\mathbf{s}}_i$. Detailed derivation of the optimum rotation matrix and translation vector computation method we used can be found in [4]. Not to extend the discussion by repeating the steps in the indicated study, we just state the optimization problem and give its solution.

Let $D_i := \{\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, \dots, \mathbf{w}_{i,N_i}\}$ be the set of vertices of the input mesh corresponding to its skeleton vertex $\hat{\mathbf{s}}_i$. As the input mesh is registered to the source mesh, the vertices in D_i are also registered to the vertices in the set C_i with the same id numbers. To obtain $\hat{\mathbf{R}}_i$ and $\hat{\mathbf{t}}_i$, we define the following error metric:

$$E_i(\mathbf{R}_i, \mathbf{t}_i) := \sum_{j=1}^{N_i} \|(\mathbf{R}_i \mathbf{v}_{i,j} + \mathbf{t}_i) - \mathbf{w}_{i,j}\|^2. \quad (3.2)$$

Our aim is to solve the optimization problem

$$(\hat{\mathbf{R}}_i, \hat{\mathbf{t}}_i) = \underset{\mathbf{R}_i, \mathbf{t}_i}{\operatorname{argmin}} E_i(\mathbf{R}_i, \mathbf{t}_i) \quad (3.3)$$

subject to $\mathbf{R}_i \mathbf{R}_i^T = \mathbf{R}_i^T \mathbf{R}_i = \mathbf{I}$.

Please note that optimum rotation matrix and translation vector for each skeleton vertex of the input mesh, i.e., $\hat{\mathbf{R}}_i$ and $\hat{\mathbf{t}}_i$ are computed for each $\mathbf{s}_i$, $1 \leq i \leq N$, by using the method described in [4]. Subsequently, we find the location of the source skeleton vertex $\mathbf{s}_i$ when it is transferred to the input mesh by using the linear transformation

$$\hat{\mathbf{s}}_i = \hat{\mathbf{R}}_i \mathbf{s}_i + \hat{\mathbf{t}}_i. \quad (3.4)$$

Computation of skeleton vertices $\hat{\mathbf{s}}_i$ for the input mesh is followed by copying the edges in the source skeleton directly to the skeleton for the input mesh. In other words, if there is an edge between $\mathbf{s}_i$ and $\mathbf{s}_j$ in the source skeleton, we connect $\hat{\mathbf{s}}_i$ and $\hat{\mathbf{s}}_j$ by adding an edge between them.

Note that, we build our approach on the observation that even if the shape exhibits large non-rigid deformations, majority of the skeleton moves locally rigidly, e.g., from ankle to elbow. The optimal rotations and translations handle the rigid movement of the local parts as well as the non-rigid movement of the minor joint regions

which are sufficiently small, e.g., elbow. This is validated through our comprehensive evaluations which involve skeleton transfer under very large non-rigid transformations (Section 3.4).

3.3.3 Postprocessing

When we follow the procedure mentioned up to now, we obtain a skeleton for the input mesh registered to the source skeleton, and we call it as the transferred skeleton. 1D curve representing the transferred skeleton may have ripples like in the figure at right due to the nonzero error values resulting in Equation 3.2. In order to get rid of those ripples as much as possible, we applied Laplacian smoothing method to the 1D curve representing the transferred skeleton.



Let $\{s_1, s_2, \dots, s_{L-1}, s_L\}$ be a skeleton segment and let $\{s'_1, s'_2, \dots, s'_{L-1}, s'_L\}$ be the smoothed version of it. Positions of the terminal vertices are preserved, i.e., $s'_1 = s_1$ and $s'_L = s_L$. Let s_j be a non-terminal vertex in the segment. Let s_{j-1} be the preceding vertex and s_{j+1} be the succeeding vertex of it. Weight w_{j-1} of the edge between s_{j-1} and s_j and weight w_j of the edge between s_j and s_{j+1} are defined as

$$w_{j-1} := \frac{1}{\|s_j - s_{j-1}\|} \quad (3.5)$$

and

$$w_j := \frac{1}{\|s_{j+1} - s_j\|}, \quad (3.6)$$

respectively. After computation of the weights, the displacement vector d_j for s_j is computed as

$$d_j := \frac{w_{j-1}s_{j-1} + w_js_{j+1}}{w_{j-1} + w_j} - s_j. \quad (3.7)$$

Finally, s'_j is computed by the relation

$$s'_j = s_j + 0.5d_j. \quad (3.8)$$

Applying Laplacian smoothing procedure just one time may not be enough to reach a satisfying smooth skeleton segment. Thus, the smoothing procedure is applied repeatedly to a skeleton segment until the displacement is sufficiently small. To this end, iterations continue until $0.5\|\mathbf{d}_j\| < 0.05\lambda_{avg}$ condition is satisfied for any $\mathbf{s}_j$ where λ_{avg} is the average edge length of the source skeleton segment.

The smoothing operation we used is computationally lightweight, as supported by the timing values given in Section 3.4.9. It smoothes the curve-skeleton mostly in one pass, acting just like a low-pass filter, if not, in a few passes. Note that, it does not check any condition about surface boundary which may arise the question whether it can push the transferred skeleton out of the shape boundary. For this to happen for a skeleton vertex $\mathbf{s}_j$, $0.5\|\mathbf{d}_j\| > r_j$ should be satisfied where r_j is the distance to the closest mesh vertex to $\mathbf{s}_j$. Although it is mathematically possible, this can be considered as an extreme case. An example setup to this case is given in Figure 3.4 in 2D to easily explain the case. In the figure, black lines represent shape boundary, red lines represent its curve-skeleton, $\mathbf{s}_{j-1}$, $\mathbf{s}_j$, $\mathbf{s}_{j+1}$ represent three consecutive skeleton vertices, $0.5\mathbf{d}_j$ represents half of the displacement vector, and $\mathbf{s}'_j$ represents new position of $\mathbf{s}_j$. It is hard to encounter such very thin and zigzaggy mesh structures in animation characters. Legs of the horse and centaur models are closest examples to such structures in the datasets we used and the smoothing operation did not push the transferred skeleton out of those structures. Despite all of these, one can still modify the smoothing operation adding a boundary check condition or use a completely different one to ensure the skeleton is inside of the shape boundary if the dataset includes such very thin and zigzaggy structures as in Figure 3.4. However, this may make the smoothing operation computationally expensive relative to surface correspondence and transfer stages.

3.4 Results and Discussion

In this section, we will first present the datasets used in the experiments. Subsequently, we will briefly explain the experiments we conducted to test the performance of the method in various cases and discuss the corresponding results. Finally, we will present the time consumed by the method for each model.

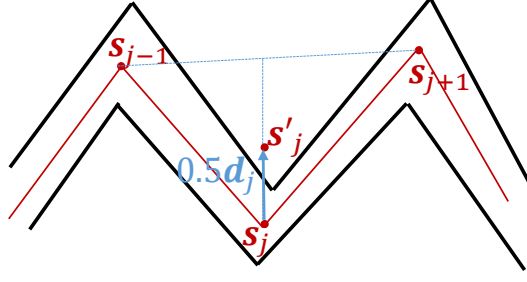


Figure 3.4: An extreme case in 2D setup in which the shape boundary is zigzaggy.

3.4.1 Datasets

We used the FAUST dataset [41] and the TOSCA dataset [42] in the experiments, for human and non-human skeleton extraction demonstrations, respectively. FAUST comes with triangulated meshes of 10 human subjects, each in 10 different poses. The meshes were obtained by registering a template mesh to all these poses, hence one-to-one correspondence is available over the dataset. We used a subset of this dataset that consists of 4 subjects. One pose of the first subject is used as the source mesh from which the source skeleton was extracted. 9 other meshes of this first subject are used as inputs to our intra-subject skeleton transfer experiments (Section 3.4.2), and the remaining 30 meshes are the inputs to the inter-subject part (Section 3.4.3). We also create and use the downsampled point clouds (Section 3.4.4) and punctured versions (Section 3.4.5) of these meshes for other experiments that demonstrate versatility.

TOSCA, on the other hand, is a synthetic dataset obtained by deforming reference mesh models towards new poses, which also brings one-to-one correspondence information. We used Centaur, Horse, Gorilla, and Wolf subjects from TOSCA. One mesh of each subject class provides the respective source skeleton. In addition to the original meshes, we again create and use the downsampled point clouds and punctured versions of these meshes to perform a thorough evaluation (Section 3.4.6).

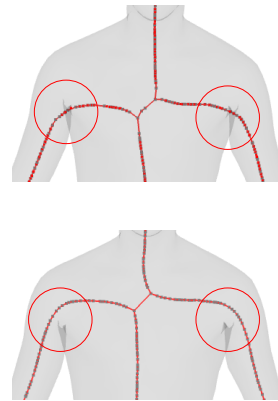
3.4.2 Intra-Subject Skeleton Transfer Experiments (FAUST)

In this set of experiments, intra-subject skeleton transfer ability of the method was tested, that is, transferring the source skeleton to the other poses of the same subject. Our method transferred the source skeleton, which is inside of the human body shown in Figure 3.1, to the input meshes successfully. The transferred skeletons for the 9 poses of the first subject can be seen in Figure 3.5 which shows that results maintain the medial axis property. Note that the deviation of medial axis in the right foot of the source skeleton (Figure 3.1) is reflected in our transfer results. The accurate and smooth results verify the robustness of our surface correspondence, rigid transformation, and postprocessing schemes in Section 3.3.

3.4.3 Inter-Subject Skeleton Transfer Experiments (FAUST)

In this set of experiments, inter-subject skeleton transfer ability of the method was tested. In these experiments the source skeleton was transferred to the 10 meshes representing all 10 poses of the second, third and fourth subjects of the FAUST dataset. Some of the resulting skeletons transferred from the source mesh to the meshes of the second, third and fourth subjects of this dataset are shown in Figure 3.6. Our method successfully transferred the source skeleton to 30 different poses extremely quickly in-one shot. The results show that the established rotation matrices and translation vectors can represent pose changes not only between intra-subjects but also between inter-subjects providing the method with the abilities such as skeleton transfer between genders and subjects with different weights and heights.

Fast transfer of the source skeleton to several input meshes in one-shot is not the only advantage of the proposed method. In some cases, the transferred skeleton is also more accurate than the skeleton for the input mesh computed from scratch via a direct method. An example to this case can be observed in the figure at right. The skeleton at the top was computed by using MCF [22] and the one at the bottom was transferred from the source skeleton by using our method. As it can be



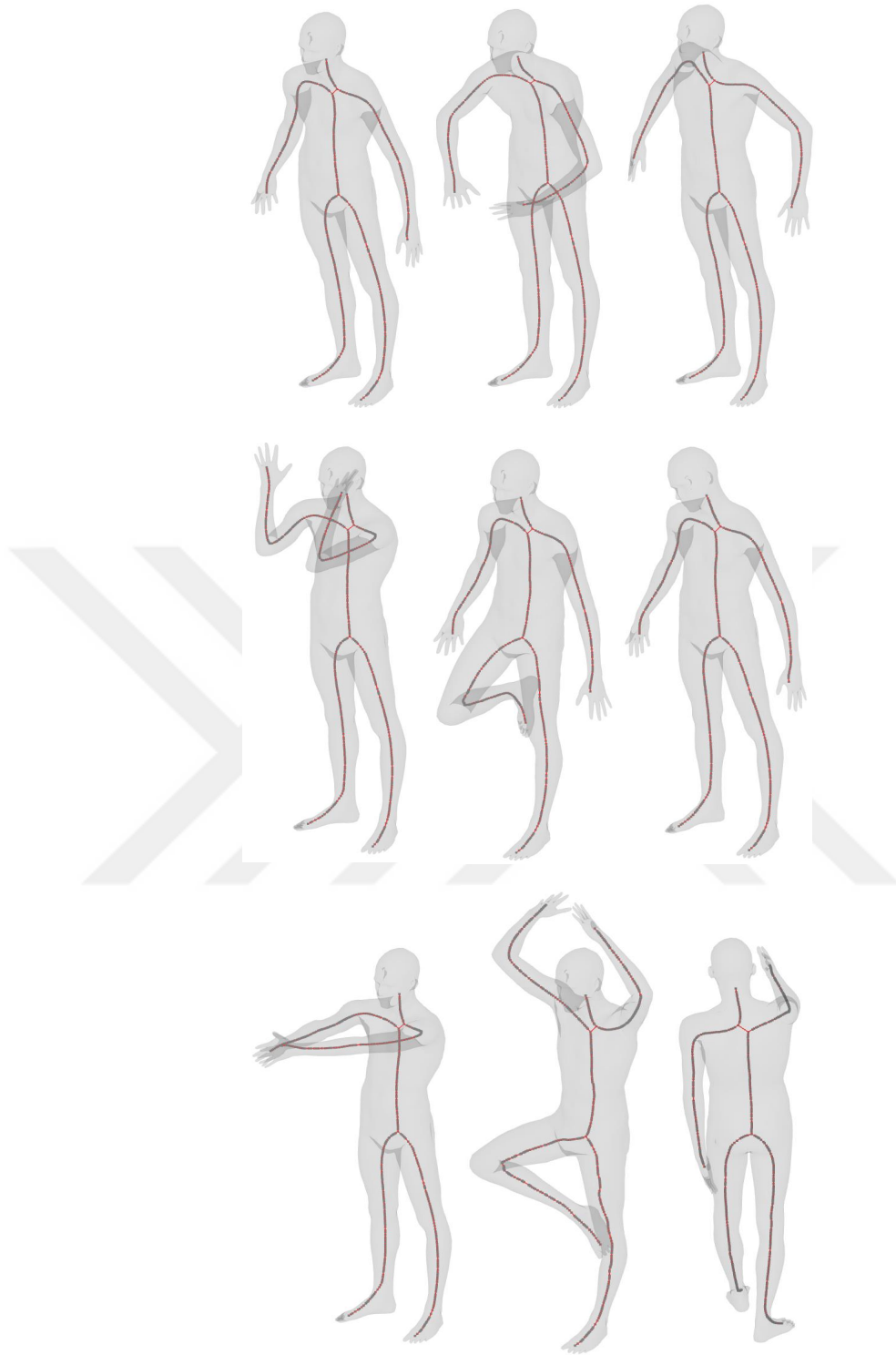


Figure 3.5: Input meshes for the first subject of FAUST dataset with the transferred skeletons inside of them, resulting from the intra-subject experiments.

seen, our result is better than that of MCF in terms of medial-axis property around the armpit (region inside of the red circles) of the subject. We also note that MCF, and many other existing algorithms, require watertight input meshes, meaning that they will fail on various other input types that we support, e.g., point clouds (Section 3.4.4) and punctured models with boundaries (Section 3.4.5).

3.4.4 Experiments for Skeleton Transfer to Downsampled Point Clouds (FAUST)

In this set of experiments, the method was tested on input meshes having missing vertices and no connectivities. For this purpose, firstly, a number of vertices of the source mesh were selected randomly uniformly and the id numbers of these vertices were recorded. After that, vertices with the recorded id numbers were removed from all of the meshes, including the source mesh and all the input meshes of all of the subjects. We did not do any remeshing operations subsequently, instead, for each removed vertex, we further removed all the edges emanating from that vertex, and all the triangles associated with that vertex. To observe the performance degradation clearly, the number of removed vertices were increased 1000-by-1000 from 1000 to 6000, creating a downsampled point cloud data in each step. Subsequently, surface correspondences were established between the source skeleton and every downsampled versions of the source mesh, separately. Please note that, it is better to assume the inputs as downsampled point clouds at this point for this experiments because while transferring the source skeleton to them, we do not use and need any face or edge information. After the surface correspondence stage, the source skeleton was transferred to the point clouds of inputs by using the approach presented in Section 3.3.2 and the resulting skeletons were smoothed by using the approach presented in Section 3.3.3.

The performance of the method did not decrease noticeably even when approximately 60% of the vertices of the meshes were uniform randomly removed. Some of the results of this set of the experiments are shown in Figure 3.7. This high level of robustness is due to the fact that, as long as the correspondence method men-

tioned in Section 3.3.1 can find alternative vertices for the removed vertices in close neighbourhoods of skeleton vertices, the established local transformations are able to represent both intra-subject and inter-subject pose changes.

The transferred skeletons started to deteriorate as the number of removed vertices were increased. The first two rows of Figure 3.8 show four of the results when 5000 of 6890 (72.5% approx.) vertices of the meshes were uniform randomly removed. As one can verify from the figure, although the transferred skeletons are still valid for some input meshes (e.g., first two subfigures of the first column), they start to become inaccurate for some other meshes (e.g., first two subfigures of the second column). When 6000 of 6890 (87% approx.) vertices were removed uniform randomly, the method could not compute valid skeletons for the input meshes. Two of the results of this case are shown at the last row of Figure 3.8, where one can see that although the transferred skeleton is not bad in some cases (e.g., left subfigure of the last row), most of them are deteriorated (e.g., right subfigure of the last row). The severe decrease in the accuracy is due to the fact that, when the surface correspondence method cannot find enough number of points in close neighbourhood of a skeleton vertex, it starts to correspond it to far-away vertices, resulting in erroneous rotation matrix and translation vector.

3.4.5 Experiments for Skeleton Transfer to Punctured Meshes (FAUST)

In this set of experiments, a set of connected vertices in a specific area of the meshes was removed instead of random removal, resulting in punctured meshes with boundaries. The id numbers of the removed vertices were the same for all the meshes including the source mesh and all the input meshes of all of the subjects. We generated three types of punctured meshes each one having a hole in a specific area: on the upper left arm, on the upper right leg and on the torso. As in the case for downsampling, after creation of new input meshes, surface correspondences were established between the source skeleton and every punctured version of the source mesh, separately. Surface correspondence step was followed by the skeleton transfer and smoothing steps in sequence.

The method transferred the source skeleton accurately in various cases of the punc-

tured meshes. The results related with the inputs from which 91 connected vertices were removed resulting a hole in the upper left arm are shown in Figure 3.9. In this case, when the vertices were removed, the skeleton vertices close to the hole were corresponded to the mesh vertices in the same rigid part with them, that is, in the upper left arm. As a consequence, the computed rotation matrices and translation vectors could capture the pose changes accurately. Of course, as the number of vertices gets larger, the results would become deteriorated.

The accuracy decrease in cases for punctured meshes depends both on the size and place of the hole. For example, when 88 vertices (almost same with the previous case) were removed from right-legs of the meshes, the accuracy dropped relatively severely for the most cases. Four examples to this case are given in Figure 3.10. While the results at the middle column are accurate due to little pose change at the legs, the ones at the last column have inaccurate skeletons.

Four of the results can be seen in Figure 3.11 when we removed 176 vertices from the torsos of the meshes. As it can be verified from the figure, when the position of torso of the subject changes significantly, the quality of the transferred skeleton has a high potential to decrease. The subfigure at the bottom-right is an example to this case where the skeleton segment in the torso has an abrupt change. The results associated with the punctured test cases implies that as the method corresponds a skeleton vertex s_i to the vertices in different rigid parts of the mesh than the part in which s_i lies, the estimated transformation for s_i becomes erroneous.

3.4.6 Experiments for Skeleton Transfer Between Non-Human Subjects (TOSCA)

To assess how well the method can be generalized, we used some meshes from the TOSCA dataset which contains meshes of non-human subjects. More specifically, for each set of Centaur, Horse, Gorilla and Wolf meshes from TOSCA dataset, we chose a source mesh separately. Following that, we extracted the source skeletons for each kind of non-human source mesh by using the MCF method as for the FAUST dataset. Subsequently, for each type, we transferred the source skeleton to other meshes for the same type. Figure 3.12 shows the results of this set of experiments. Each row corresponds to a different non-human type, the first column shows the

shaded versions of the source meshes whose skeletons can be seen in Figure 3.1. The other columns show the transferred skeleton to the poses different than the source pose for each type. The mesh models in the dataset do not only correspond to non-human models but also have different number of vertices and sampling densities compared to the models in the FAUST dataset. We did not modify any parameter of the proposed method specifically for this dataset implying that the method is highly and easily generalizable.

We also tested the performance of the method on downsampled and punctured models from the TOSCA dataset to check whether the successful results from the FAUST dataset still hold. To prepare the downsampled models, we removed $\sim 60\%$ of the vertices uniform randomly. The rest of the procedure was the same as the case for the downsampled models from the FAUST dataset described in Section 3.4.4. Results for two of the downsampled models of two different subjects from TOSCA dataset are shown in Figure 3.13 as point clouds, each row showing a different subject. We removed 9500 out of 15768 vertices randomly and uniformly from the centaur model and 11500 out of 19248 from the horse model. The source skeletons are shown in the first column while the transferred ones to different poses are shown in the second column. The results imply that conclusions for the models from the FAUST dataset can be generalized to this dataset seamlessly.

To prepare the punctured models, a set of connected vertices from a specific part of the mesh were removed. The rest of the procedure was the same as the case for the punctured models from the FAUST dataset described in Section 3.4.5. Results for two of the punctured models from the TOSCA dataset are shown in Figure 3.14, each row showing a different subject. We removed 579 connected vertices out of 25438 from the back of the Gorilla model and 464 connected vertices out of 4344 from the neck of the Wolf model. Shaded versions of the models and the source skeletons are shown in the first and second columns, respectively. The transferred skeletons to different poses are shown in the last columns. As it can be seen by the red circles in the figure, the transferred skeletons started to become deteriorated, similar to the case for FAUST models, when we removed a set of vertices from the models non-uniformly, like creating large holes on the surface.

3.4.7 Experiments for Skeleton Transfer Between Non-Registered Shapes

In the experiments of Sections 3.4.2 and 3.4.3, the source mesh and target meshes were registered vertex-by-vertex. The ones used in the experiments of Sections 3.4.4 and 3.4.5, a set of vertices were removed from the meshes but the remaining vertices were still registered vertex-by-vertex. In this new set of experiments, on the other hand, we test how well the method performs when true correspondence information between source and target meshes is unavailable.

To this end, we first need to choose source FAUST meshes for each of the second, third, and fourth subjects in order to perform intra-subject skeleton transfer experiments over these subjects. We chose the meshes in the same pose with the source mesh for the first subject, and then extracted their skeletons by using MCF [22]. After skeleton extraction, we registered source meshes and target meshes to each other for each subject by using the automatic shape correspondence method given in [30]. The rest of the skeleton transfer method is the same as in the other experiments.

The shape correspondence method [30] was run with 9 maximum levels and 8 initial samples. The details of these parameters can be found in [30]. In this setting, the algorithm computes approximately 4600 corresponding vertices on average between the source meshes and target meshes that have 6890 vertices each. Some of the results from FAUST dataset are given in Figure 3.15. Each row represents a different subject.

We conducted the same experiment on TOSCA dataset for Centaur, Gorilla and Wolf models. For all of the models, [30] was run with 10 maximum levels and 8 initial samples for Centaur model. The respective parameters for Gorilla models were 9 and 8, and for Wolf model they were 8 and 8. Besides, it computed approximately 5000 corresponding vertices out of 15768 for Centaur model, 3400 corresponding vertices out of 25438 for Gorilla model and 2800 corresponding vertices out of 4344 for Wolf model on average. We provide one result for each model in Figure 3.16. The number of computed corresponding vertices for Centaur and Gorilla models are far below $\sim 40\%$ of number of original vertices, which is insufficient for estimating optimum rigid transformations. Their results are consequently not as robust as the Wolf case,

which aligns with the conclusions drawn in Section 3.4.4.

Accuracy of the skeleton transfer method certainly depends on the accuracy of the shape correspondence method if vertex-by-vertex correspondence is unknown beforehand. Therefore, it would be appropriate to share the accuracy of the shape correspondence method here. Average ground-truth correspondence cost for a computed map ϕ defined in [30] is given as

$$D_{\text{grd}}(\phi) = \frac{1}{|\phi|} \sum_{(s_i, t_j) \in \phi} g(s_i, t_j) \quad (3.9)$$

where (s_i, t_i) is the ground-truth match and (s_i, t_j) is the computed one. g gives the normalized geodesic distance between any two vertices such that the maximum distance is 1. Average D_{grd} values for FAUST and TOSCA are ~ 0.042 and ~ 0.033 , respectively, which verify that our skeleton transfer method works fine in the presence of deviations from the ground-truth zero-cost maps.

Accuracy and robustness of skeleton transfer between non-registered shapes depend on two key points: The number of computed corresponding vertices and accuracy of the computed corresponding vertices. The accuracy and robustness of the transferred skeleton increase as these two quantities increase. To conclude, the proposed method allows skeleton transfer between non-registered meshes if the shape correspondence method run beforehand is robust and produces sufficient number of corresponding points.

3.4.8 Quantitative Analysis

In addition to visually inspecting the transferred skeletons through all the preceding figures, we propose two novel error metrics to quantify the quality of centeredness property of the results. Let $\mathbf{p}_{i,1}$ be the closest point to the skeleton point $\mathbf{s}_i$ lying on the mesh in terms of Euclidean distance. Note that $\mathbf{p}_{i,1}$ does not have to be a vertex of the mesh, it can lie on a triangle or edge of it. Let $\mathbf{d}_{i,1} := \mathbf{p}_{i,1} - \mathbf{s}_i$. We compute intersection points of the mesh and the ray $\mathbf{s}_i - t\mathbf{d}_{i,1}$, and denote the closest one to $\mathbf{s}_i$ as $\mathbf{p}_{i,2}$. Let $\mathbf{d}_{i,2} := \mathbf{p}_{i,2} - \mathbf{s}_i$. The setup can be seen at left in Figure 3.17.

With this setup, we define ε_i as

$$\varepsilon_i := ||\mathbf{d}_{i,1} + \mathbf{d}_{i,2}|| \quad (3.10)$$

which gives us a value measuring how much $\mathbf{s}_i$ deviates from the medial-axis of the mesh. Close to zero values imply that $\mathbf{s}_i$ is close to the medial-axis of the mesh. We denote $\bar{\varepsilon}$ as the average value of ε_i among all skeleton vertices not outliers, which is going to be explained later in this section. It indicates how much a skeleton vertex of the mesh deviates from the medial-axis on average.

For the second error metric, we define κ_i as

$$\kappa_i := \frac{||\mathbf{d}_{i,2}||}{||\mathbf{d}_{i,1}||} - 1. \quad (3.11)$$

Unlike ε_i , κ_i provides us a unitless quantity measuring how much $\mathbf{s}_i$ deviates from the medial-axis of the mesh. Values close to zero imply that $\mathbf{s}_i$ is close to the medial-axis of the mesh. We denote $\bar{\kappa}$ as the average value of κ_i among all skeleton vertices not outliers. Like $\bar{\varepsilon}$, it also indicates how much a skeleton vertex of the mesh deviates from the medial-axis on average yet unlike $\bar{\varepsilon}$ it is unitless.

Error metrics in Equations 3.10 and 3.11 implicitly assume the curve-skeleton goes through the center of a tubular structure. The models in this study mostly consists of such structures: Legs, arms, torsos, necks, and tails. However, this assumption is not true for some parts of the models like shoulders. An example to this case is shown at right in Figure 3.17. In this case, it is clear that $||\mathbf{d}_{i,1}||$ is not equal to $||\mathbf{d}_{i,2}||$ yet we can say that position of $\mathbf{s}_i$ is appropriate, at least visually. Such skeleton points should somehow be discarded from the accuracy computation. We call such vertices as outliers and spot them using the ratio $||\mathbf{d}_{i,2}||/||\mathbf{d}_{i,1}||$. If the ratio for $\mathbf{s}_i$ is above a threshold value determined beforehand, which we denote as η , we exclude $\mathbf{s}_i$ from the computations of the error metrics. Please note that similar outlier removal actions are common in the literature. For example, long rays are eliminated while computing the well-known and commonly-used shape diameter function [24]. According to our evaluations and observations, setting $\eta = 4$ is enough to spot most of the outliers for FAUST models. η values for the centaur, horse, gorilla, and wolf models are set to be 6, 6, 7, and 7, respectively. With these thresholds, $\sim 5\%$ of ver-

tices of FAUST models were spotted as outliers. Outlier percentages for the centaur, horse, gorilla, and wolf models are $\sim 9\%$, $\sim 11\%$, $\sim 11\%$, and $\sim 12\%$, respectively.

Quantitative results for various experiments using the error metrics introduced are provided in Table 3.1. In order to have a baseline to compare the results with, we also computed skeletons of the models using MCF [22], evaluated them, and report the results in the table. $\mu_{\bar{\varepsilon}}$ and $\sigma_{\bar{\varepsilon}}$ for an experiment represent average of the error values $\bar{\varepsilon}$ and standard deviation of $\bar{\varepsilon}$, respectively, among the models used in the experiment. Likewise, $\mu_{\bar{\kappa}}$ and $\sigma_{\bar{\kappa}}$ for an experiment represent average of the error values $\bar{\kappa}$ and standard deviation of $\bar{\kappa}$, respectively, among the models used in the experiment. As $\mu_{\bar{\varepsilon}}$ represents length, it would be better to have a quantity to compare it. For that purpose, we report l_{avg} which represents average edge length of the models used in an experiment. For the experiments with the downsampled models, results with the meshes whose $\sim 60\%$ of the vertices were removed are reported.

It is important that the source skeleton ensures medial-axis property as much as possible so we selected the skeleton with the lowest error values as the source skeleton among a set of models. Error values related with them are provided separately in the last 5 lines of Table 3.1. For FAUST dataset, errors values for both original and downsampled models for intra-subject skeleton transfer are just a little below the error values for MCF. This is most probably due to the phenomenon explained in Section 3.4.3. Error values for inter-subject skeleton transfer are slightly above error values for MCF. For TOSCA dataset, error values for both the original and downsampled wolf and horse models are a little below the error values for MCF. For both the original and downsampled centaur and gorilla models, they are slightly above the ones for MCF. Furthermore, errors for downsampled models tend to be slightly greater than, if not they are almost equal with, the values for the original models, as expected. To sum up, the method provides us a tool allowing fast skeleton transfer between shapes in correspondence, possibly with boundaries or missing vertices at the expense of a slight, or for some cases no, decrease in accuracy.

We believe that our error metrics are valuable in the sense they enable further quantitative assessments for visually similar skeletons. For example, thanks to our metrics we notice that, although visually similar, inter-subject transfer affects centered-

ness more than downsampling does. Also, these error metrics serve the purpose of comparing methods in terms of centeredness of the resulting curve-skeletons.

3.4.9 Timings

The experiments were conducted on a computer with Intel® Core™ i7-2630QM CPU running at 2 GHz. Time consumed by the surface correspondence stage and by the skeleton transfer plus smoothing stages along with mesh and skeleton vertex counts for each subject are reported in Table 5.2. Time consumed by smoothing stage along is almost negligible compared to the other stages so we reported time elapsed in the skeleton transfer and smoothing stages together. Human model in Table 5.2 refer to the models in the FAUST dataset in which all the models have same number of vertices. Centaur, Horse, Gorilla, and Wolf models are from TOSCA dataset. We also report the timings related with the downsampled versions of the same models obtained by removing $\sim 60\%$ of their vertices in Table 3.3. One can notice the $\sim 60\%$ decrease in time consumed by the surface correspondence stage. This phenomenon is expected because of the linear dependence of surface correspondence to the mesh vertex count. However, time consumed in skeleton transfer and smoothing stage did not decrease at the same ratio as these stages depend on skeleton vertex count rather than mesh vertex counts. In punctured models, the number of removed vertices is small so timings of processing the punctured inputs are almost the same as the original inputs (Table 5.2). Please note that the time consumed in the surface correspondence and skeleton transfer stages can further be decreased by making the implementation parallel.

3.4.10 Limitations

Equation 3.1 implicitly assumes that models has enough vertices on the part of the mesh to which s_i is related for a reliable surface correspondence. Drawback of this implicit assumption can best be explained by an example. Let us discuss about punctured leg sequence given in the top row of Figure 3.18. As the hole gets larger, the number of mesh vertices at the vicinity of skeleton vertices in the upper right leg decreases. Consequently, the surface correspondence stage of the method constructs

Table 3.1: Resulting error values for various experiments. $\mu_{\bar{\varepsilon}}$ and $\sigma_{\bar{\varepsilon}}$ for an experiment represent average of the error values $\bar{\varepsilon}$ and standard deviation of $\bar{\varepsilon}$, respectively, among the models used in the experiment. $\mu_{\bar{\kappa}}$ and $\sigma_{\bar{\kappa}}$ for an experiment represent average of the error values $\bar{\kappa}$ and standard deviation of $\bar{\kappa}$, respectively, among the models used in the experiment. l_{avg} for an experiment represents average edge lengths of the models used in the experiment.

Experiment	$\mu_{\bar{\varepsilon}}$	$\sigma_{\bar{\varepsilon}}$	l_{avg}	$\mu_{\bar{\kappa}}$	$\sigma_{\bar{\kappa}}$
Intra-Subject (FAUST)	0.0238	0.0016	0.0163	0.5386	0.0583
Inter-Subject (FAUST)	0.0286	0.0028	0.0168	0.5487	0.0515
Intra-Subject on D. S. Models (FAUST)	0.0239	0.0027	0.0163	0.5487	0.0715
Inter-Subject on D. S. Models (FAUST)	0.0289	0.0032	0.0168	0.5830	0.0726
MCF Human Models (FAUST)	0.0248	0.0042	0.0167	0.5482	0.0728
Centaur Model (TOSCA)	6.6116	0.5257	1.6696	0.9887	0.0679
Downsampled Centaur Model (TOSCA)	6.7015	0.4830	1.6696	1.0030	0.0514
MCF on Centaur Model (TOSCA)	6.1178	0.7676	1.6564	0.8953	0.0871
Horse Model (TOSCA)	8.6509	0.8597	2.0634	0.7819	0.1015
Downsampled Horse Model (TOSCA)	8.6794	0.9241	2.0634	0.7910	0.1140
MCF on Horse Model (TOSCA)	8.7558	0.6345	2.0590	0.7664	0.0497
Gorilla Model (TOSCA)	6.7356	0.6250	1.0729	1.3140	0.2074
Downsampled Gorilla Model (TOSCA)	6.7600	0.5930	1.0729	1.3118	0.1190
MCF on Gorilla Model (TOSCA)	5.8962	0.2415	1.0680	1.2381	0.2664
Wolf Model (TOSCA)	4.5667	0.1731	2.2529	0.8899	0.0118
Downsampled Wolf Model (TOSCA)	4.5556	0.1402	2.2529	0.8875	0.0075
MCF on Wolf Model (TOSCA)	4.7830	0.1582	2.2538	0.9596	0.0462
Source Skel. of Human Models (FAUST)	0.0226	N/A	0.0165	0.4972	N/A
Source Skel. of Centaur Model (TOSCA)	5.5089	N/A	1.6298	0.7675	N/A
Source Skel. of Horse Model (TOSCA)	8.2753	N/A	2.0218	0.7375	N/A
Source Skel. of Gorilla Model (TOSCA)	5.3832	N/A	1.0531	0.9107	N/A
Source Skel. of Wolf Model (TOSCA)	4.3227	N/A	2.2557	0.8108	N/A

Table 3.2: Time elapsed in the different stages of the method along with number of input mesh vertices and skeleton vertices. Original meshes are used. All timings are reported in seconds.

Model	Mesh vertex count	Skeleton vertex count	Surface corr.	Transfer + Smoothing (Avg.)
Human	6890	522	4.121	0.588
Centaur	15768	664	12.038	1.216
Horse	19248	654	14.840	1.344
Gorilla	25438	493	14.752	1.360
Wolf	4344	508	2.684	0.867

Table 3.3: Time elapsed in the different stages of the method along with number of input mesh vertices and skeleton vertices. Downsampled ($\sim 60\%$ removed) point clouds are used. All timings are reported in seconds.

Model	Mesh vertex count	Skeleton vertex count	Surface corr.	Transfer + Smoothing (Avg.)
Human	2890	522	1.820	0.512
Centaur	6268	664	4.732	0.976
Horse	7748	654	5.787	1.065
Gorilla	10238	493	5.888	0.852
Wolf	1744	508	1.160	0.768

correspondence sets for the skeleton vertices in the upper right part of the man with points from upper left leg, lower right leg and lower torso, which is a faulty correspondence. As those parts undergo different transformations than the upper right leg, transferred skeleton becomes inaccurate. However, when the faulty correspondence sets undergo similar transformations with the removed part and/or the pose

change is little between shapes, the skeleton does not deteriorates drastically. Punctured wolf sequence given at the middle row of Figure 3.18 shows an example to this case. Full results of all of the experiments with punctured mesh sequences are provided in the supplementary file.

In addition to that, when the surface correspondence set undergoes an extreme deformation, Equation 3.2 is incapable to capture the required transformation mapping C_i to D_i . This failure can be observed in the experiments with horse and camel collapse sequences [43]. For example, in the horse collapse sequence shown at the bottom row of Figure 3.18, the front legs of the horse undergo such a drastic deformation that Equation 3.2 cannot capture the true mapping for correspondence sets on the front legs of the shape. Results for the camel collapse sequence are provided in the supplementary file.

Another limitation of the method emerges when the deformation between the source shape and a target shape is not distance preserving, i.e., non-isometric. The method can tolerate such deformations and perform transfer successfully in those cases up to some extent; inter-subject skeleton transfer (Section 3.4.3) can be regarded as mild cases of such deformations. Nonetheless, if the degree of non-isometry is high, the resulting skeletons will not be accurate because of the faulty correspondences. For example, if we transfer skeleton of a cat to a giraffe, the result would be inaccurate due to high non-isometry (Distance between cat’s chin and leg is not preserved over the giraffe’s chin and leg due to stretched neck.). In this case, one can employ an automatic non-isometric correspondence method, or can even manually provide these correspondences to overcome this shortcoming.

3.5 Conclusion and Future Work

We have introduced skeleton transfer between two shapes in correspondence. The query shape can be discretized as a mesh (not necessarily watertight, not necessarily with fixed topology) or point cloud. To perform the transfer, firstly, the method is supplied a source shape and its skeleton that can be obtained manually or with any of the existing methods. For each point of the source skeleton, we associate a

set of source shape points. After that, for each set of associated points, optimum rotation matrix and translation vector mapping the point set to the corresponding query point set is computed. By using these transformations, the source skeleton is transferred to the query shape point by point. Skeleton transfer process is completed by a smoothing postprocessing stage.

We conducted experiments to test both intra-subject and inter-subject human skeleton transfer performance of the method as well as non-human cases. Various types of shape discretizations are used in order to show versatility. The results of the experiments demonstrated visually appealing skeletons, even for the cases where the number of vertices in the meshes are decreased uniformly by $\sim 60\%$. Nevertheless, the transferred skeletons started to lose their accuracy when we removed more of the vertices than this percentage. When the removal is done non-uniformly, i.e., in the form of a hole around a specific region, our method is still successful but not as tolerant as the $\sim 60\%$ case. These two removal cases, consequently, can be cast as the limitations of our method if they exceed a certain amount. The method was also tested on cases when no correspondence information is available. In such cases, we first established a correspondence between the source shape and the query shapes by an existing shape correspondence method. Results of these experiments show that the method is still capable of transferring the skeleton accurately with accuracy depending on the performance of the shape correspondence method used.

We currently handle skeleton transfer between non-rigid shapes such as different poses of an articulated model. Another interesting research direction is to extend the scope of the work so that it can handle shapes with arbitrarily scaled parts, e.g., transferring horse skeleton to a giraffe which has a longer neck. Moreover, as the method keeps topology of the source skeleton during transfer operation, it cannot handle the cases when genus of the source mesh changes across input meshes. We leave enhancement of the method to manage such cases as a future work. In addition to skeleton transfer problem, investigating the inverse problem, that is, transferring the outer shape to other target skeletons, can be addressed as a future work.

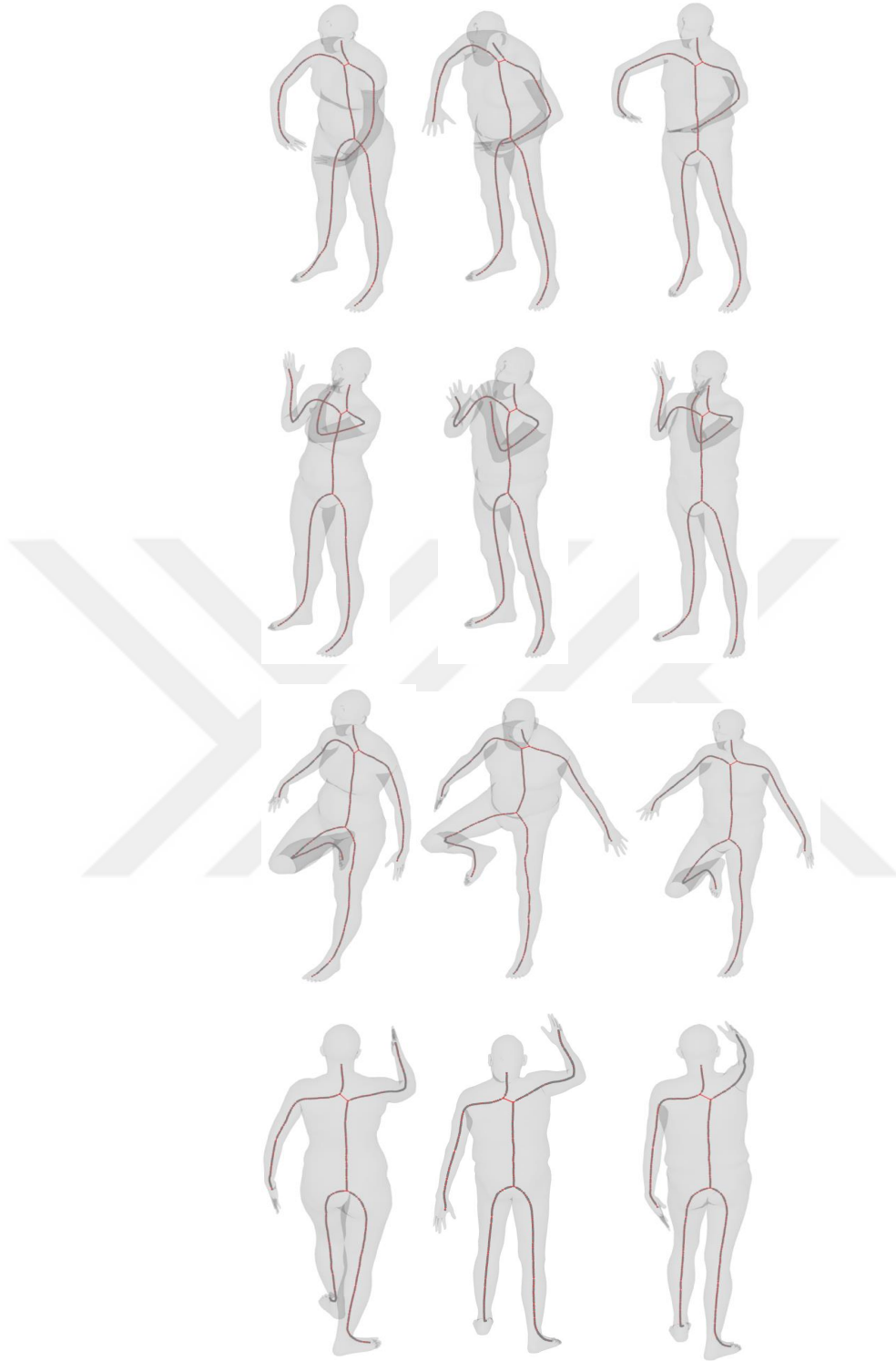


Figure 3.6: Some of the inter-subject skeleton transfer results. Each column shows different poses of the same subject and each row shows the same poses of different subjects.

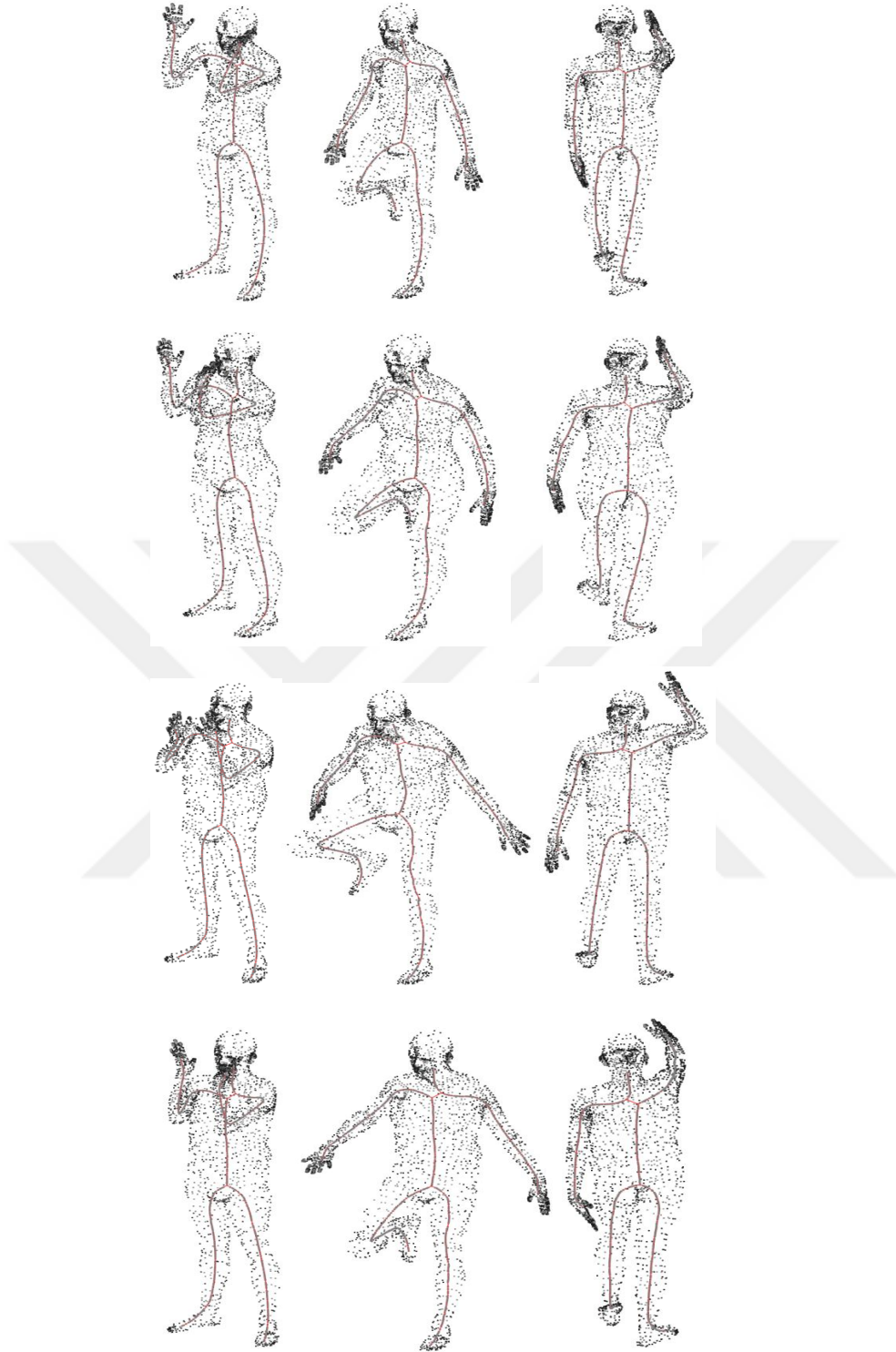


Figure 3.7: Some of the skeleton transfer results when 4000 out of 6890 vertices were removed from the meshes; the subjects are represented as disconnected point clouds with 2890 points. Each column shows same poses of different subjects and each row represents different poses of the same subject.

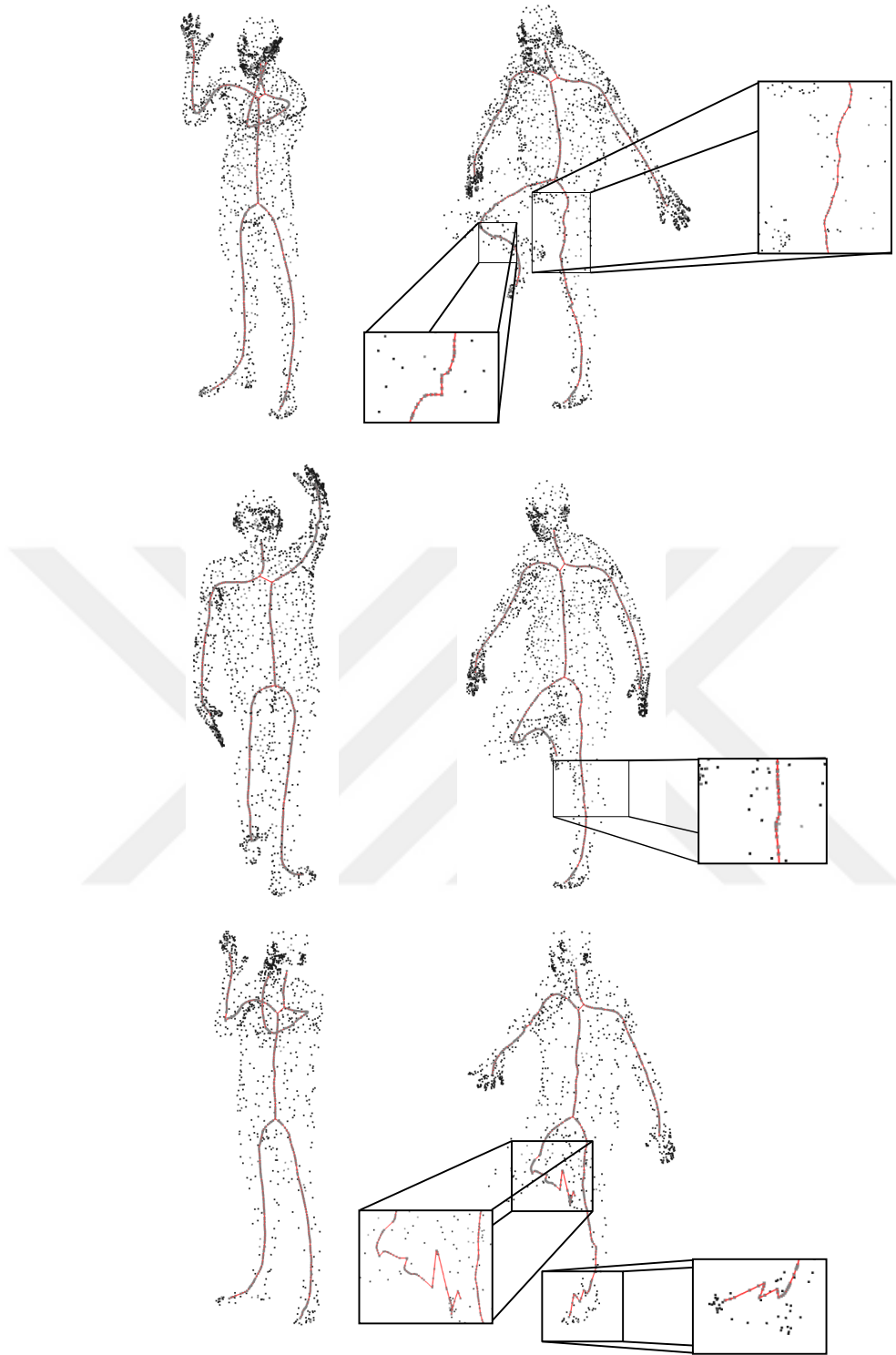


Figure 3.8: Subfigures of the first two rows show four of the skeleton transfer results when 5000 of 6890 vertices were removed from the meshes. Subfigures of the last row show two of the skeleton transfer results when 6000 of 6890 vertices were removed from the meshes.

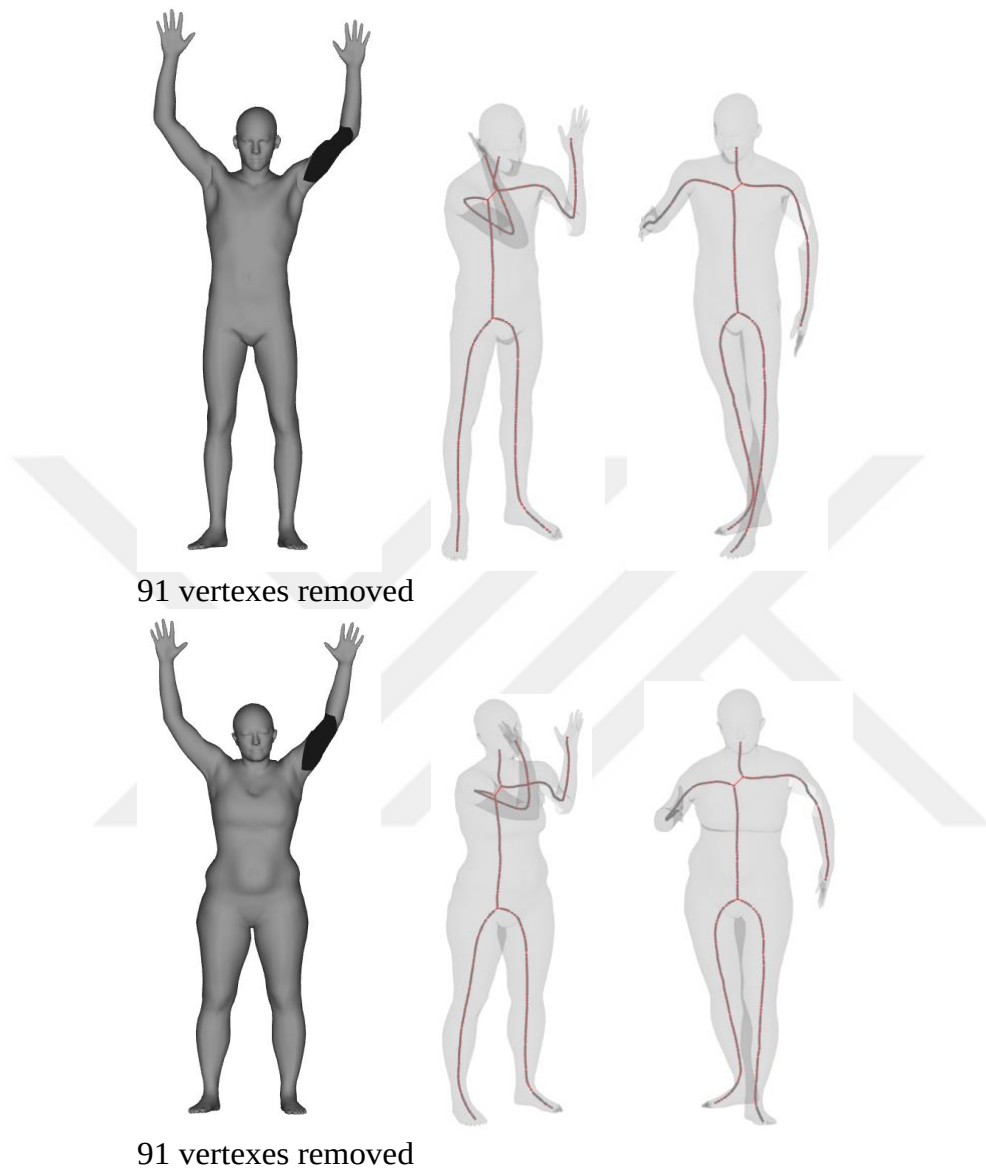


Figure 3.9: Four of the skeleton transfer results when 91 vertices connected together on the upper-left arm of the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are in the first column. The other two columns show the resulting skeletons for two different poses.

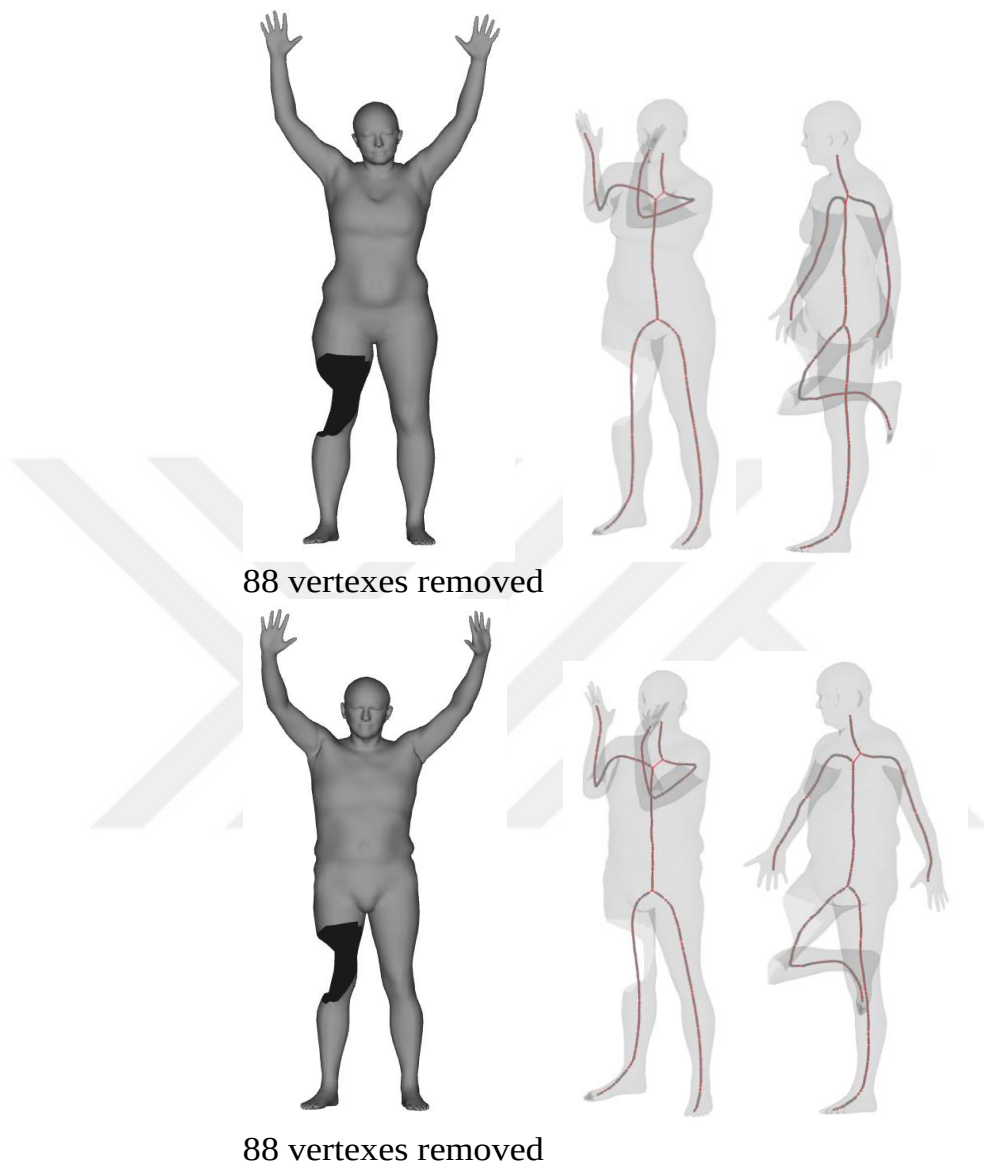


Figure 3.10: Four of the skeleton transfer results when 88 vertices connected together on the upper-right leg of the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are shown in the first column. The other two columns show the resulting skeletons for two different poses.

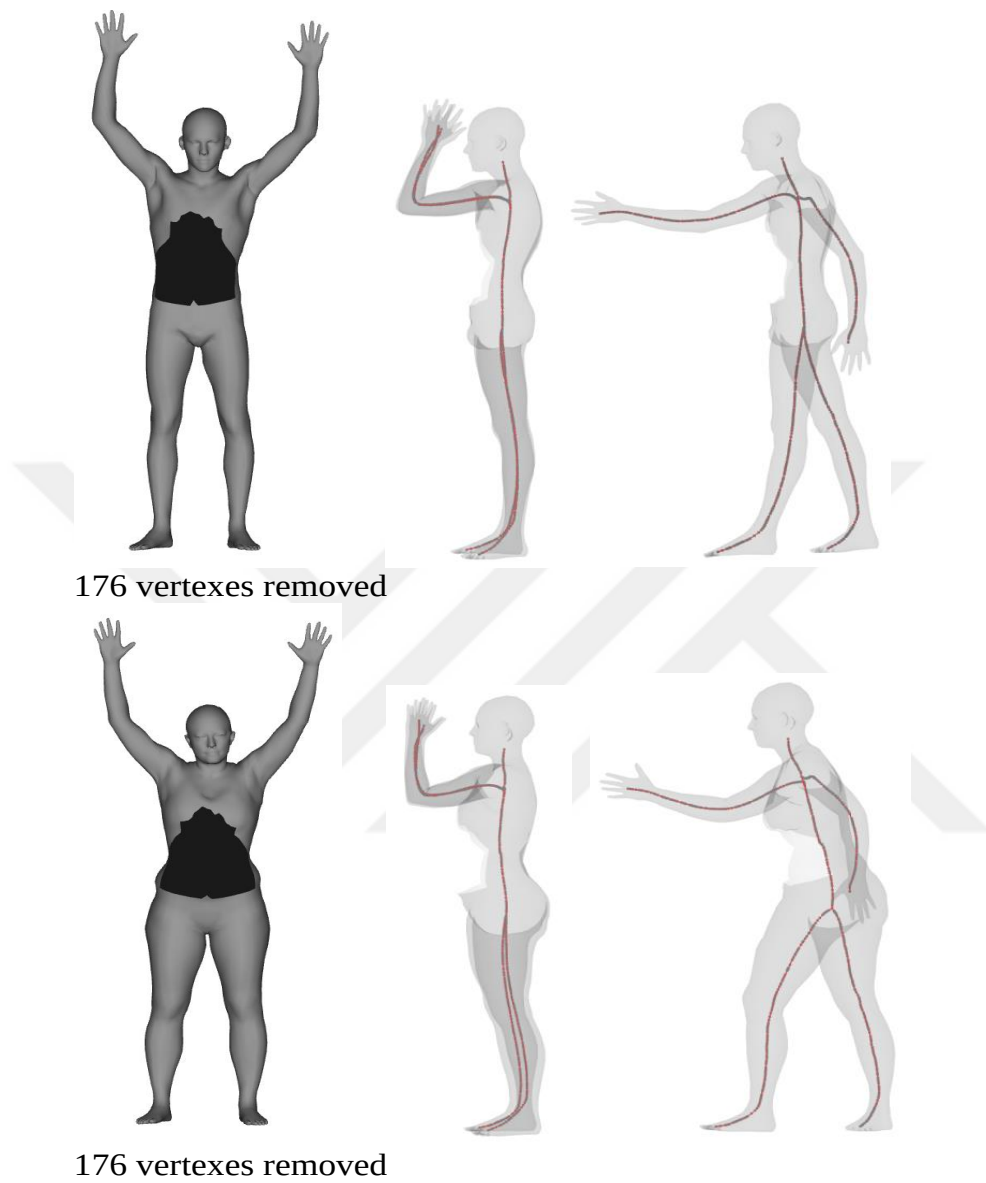


Figure 3.11: Four of the skeleton transfer results when 176 vertices connected together on the torso the subjects were removed, forming a hole. Each row represents a different subject. Shaded versions of the subjects are shown in the first column. The other two columns show the resulting skeletons for two different poses.

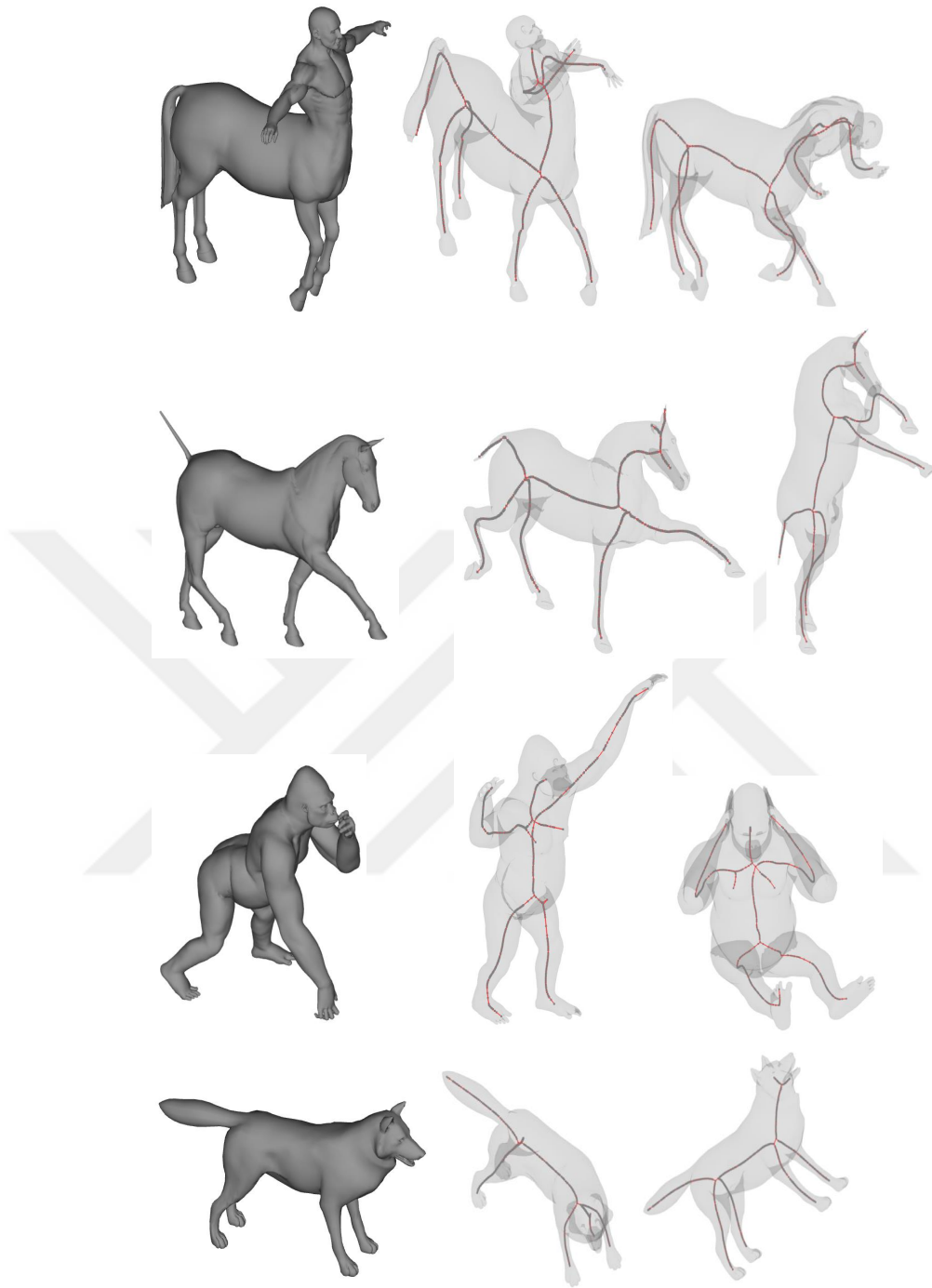


Figure 3.12: Results obtained from the TOSCA dataset. Each row corresponds to a different subject. The first column shows shaded versions of the source meshes, the other columns show the transferred skeleton to a different pose than the source pose for each type.

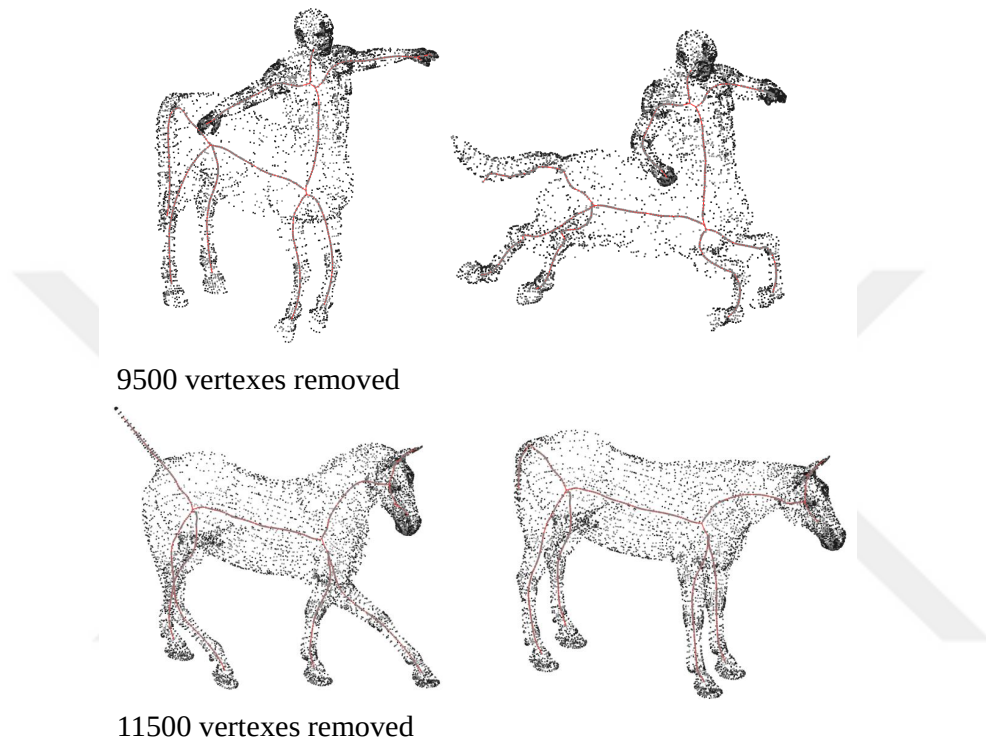


Figure 3.13: Skeleton transfer results for two downsampled point cloud models from the TOSCA dataset. Each row represents a different subject. The first column shows the source skeletons in the point clouds along with the number of removed vertices. The second column shows the transferred skeletons to other poses again inside the point clouds.

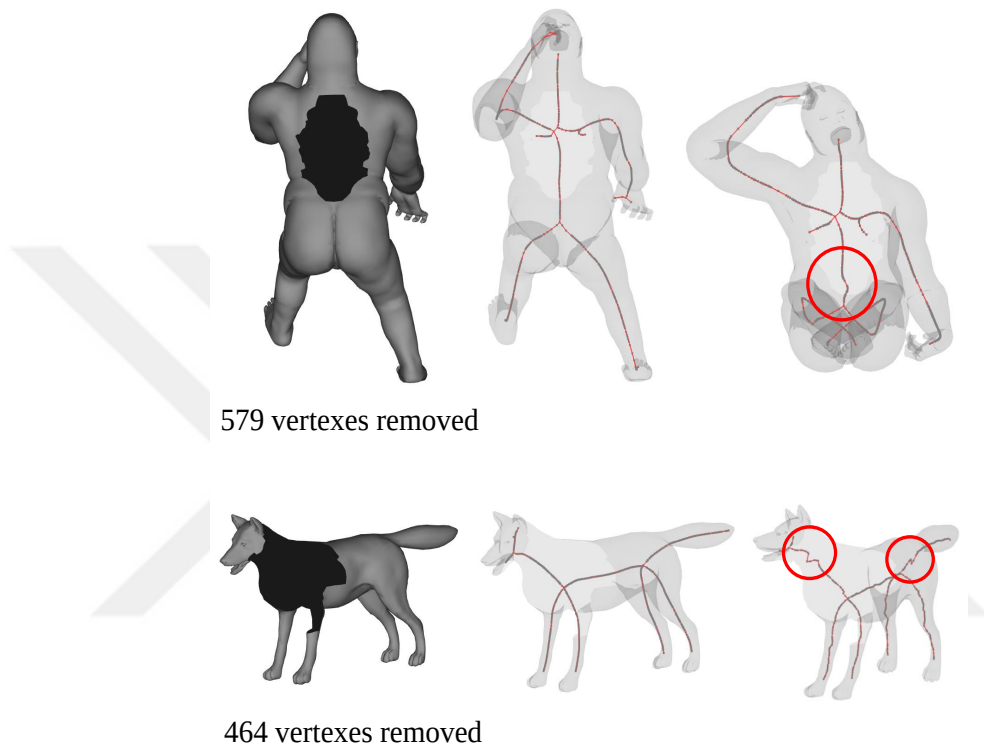


Figure 3.14: Skeleton transfer results for two punctured models from TOSCA dataset. Each row represents a different subject. Shaded versions of the models along with the number of removed vertices are shown in the first column. The middle and last columns show the source and transferred skeletons, respectively.

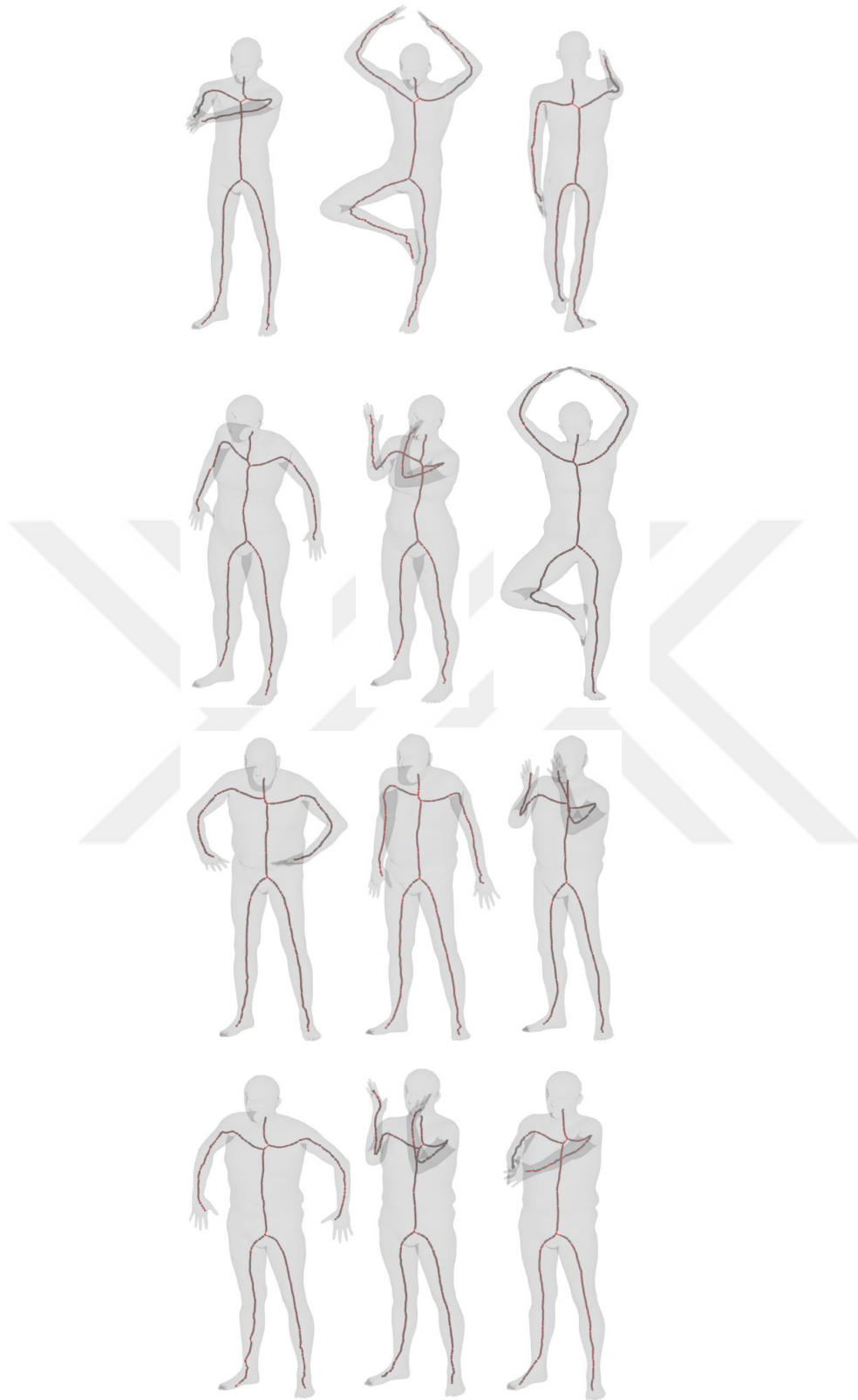


Figure 3.15: Some of the skeleton transfer results between non-registered shapes. The meshes in each row belongs to a different subject.

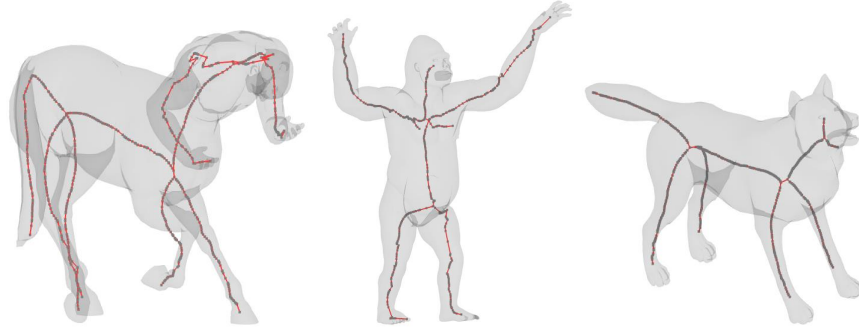


Figure 3.16: One skeleton transfer result for each of centaur, gorilla and wolf models from TOSCA dataset.

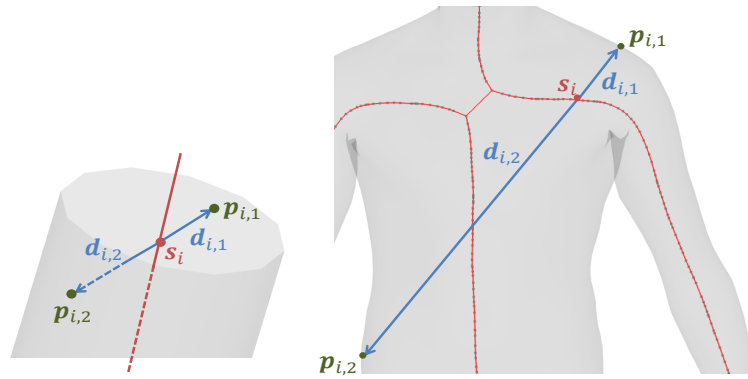


Figure 3.17: The subfigure at left shows a didactic example introducing the variables pictorially used in the error metrics. The subfigure at right shows a real case in our models in which using the error metrics as they are would be inappropriate.

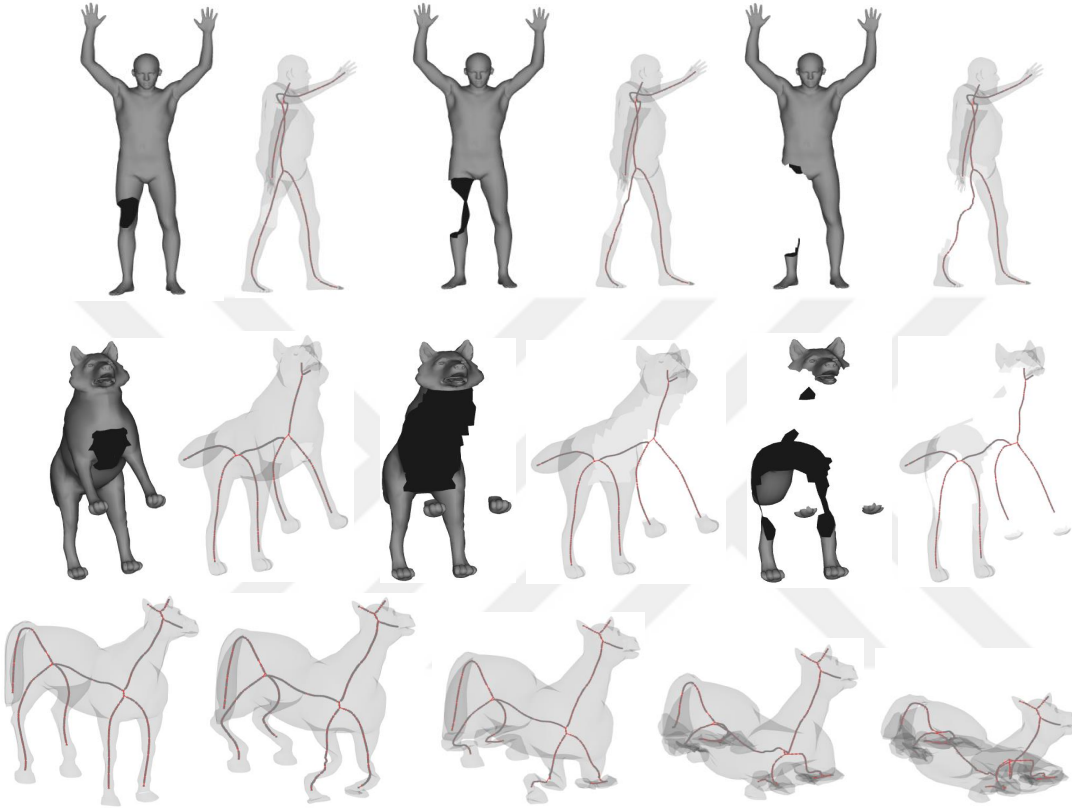


Figure 3.18: The top row shows the punctured leg sequence and the middle row shows the punctured wolf sequence. For the mesh pairs at both rows, the shaded figure at left shows the source mesh and target meshes with transferred skeletons are shown at right. Bottom row shows mesh transfer in horse collapse sequence, the leftmost figure in this row shows the source mesh and skeleton. .

CHAPTER 4

TEMPLATE SKELETON EMBEDDING INTO 3D SHAPES

4.1 Introduction

3D content creation is an important computer graphics task as many applications such as video games and simulations depend on it. To this end, one can repose an existing character or generate it from scratch based on a basic input. Our method can be seen as a mean to serve both these two purposes: We have a fixed template shape which is reposed based on an arbitrary 3D skeleton input. Since the skeleton input is independent from the template shape, one can also see our method as a from-scratch shape generator as we manage to surface this independent skeleton, a task we refer to as shape transfer. Note that this task corresponds to the inverse problem of skeleton extraction [44].

Different reposing techniques can be used for modifying existing content. Two prominent approaches for obtaining new poses are Skeletal Subspace Deformation (SSD), also known as skinning and As-Rigid-As-Possible (ARAP) surface deformation.

Skinning approaches are widely used in real-time interactive animations like games and movies. Basically, the character is rigged by a simple skeleton, called Inverse Kinematic (IK) skeleton composed of bones, and the deformation is driven by this IK skeleton. Rigging process involves both embedding the IK skeleton into the character mesh and establishing a correspondence between each mesh vertex and the skeleton. Bone transformations are blended according to the surface correspondence and then applied to each vertex of the mesh. Such approaches are intuitive in the sense that articulated character motion is driven in the same manner in reality: Mus-

cles actuate bones, and bones move skin attached to them. The first version of these approaches is linear blend skinning (LBS) [10]. As this approach is very simple and can be implemented on GPU, the character poses in the upcoming frames of the animation are computed in a very fast manner. However, this swiftness comes with a drawbacks: The new poses are usually not accurate. Especially the skin near skeleton joints collapses towards inside of the mesh near bending regions (referred to as elbow collapse [45]), and also an undesired artifact called candy-wrapper effect occurs near joints in bone-twist cases. Several different skinning techniques have been proposed after the introduction of LBS in order to eliminate those artifacts yet the accuracy of most of those techniques depends on careful adjustment of blend weights which are best done by rig artists or the selection of some shape-specific parameters contradicting with our design requirements.

ARAP shape deformation is another popular method for mesh editing. It is formulated for meshes embedded in 3D by Sorkine and Alexa [6]. The main point of this method is keeping surface details as much as possible while editing mesh surface. Basically, the mesh is considered to be composed of overlapping cells. A subset of the mesh vertices are defined as control points (aka handles) and the user is allowed to move only these points. The positions of other vertices (aka free vertices) are iteratively and automatically updated such that the transformation of each cell will be composed of only rotation and translation as much as possible. ARAP deformation technique is promising for mesh reposing because the surface details are sufficiently preserved owing to the observation that although the deformation is quite nonrigid globally, it is locally almost rigid. Another important requirement in mesh reposing is volume preservation yet original ARAP deformation formulation does not take this into account. Several variants of ARAP deformation has been developed, which are discussed thoroughly in Section 5.2. Nevertheless, they are still not successful at volume preservation as those formulations also do not directly consider this issue.

We propose a new shape deformation approach for mesh reposing and transfer, such that, it will be intuitive like skinning, that is, the deformation will be driven by skeleton, and produce fairly accurate results alleviating elbow collapse and candy-wrapper effects near joints, and preserve surface details as in ARAP deformations. It can be both used to i) obtain new poses of an articulated character or ii) transfer

the surface of a character to 3D stick figures that can be acquired arbitrarily by any means, e.g., motion capture, 3D sketching, databases [46]. The user will provide the desired pose as a simple skeleton composed of a few edges and joints (referred to as target skeleton) along with the template shape to be deformed (referred to as source mesh). Our approach first embeds the target skeleton into the source mesh (referred to as embedded skeleton) and then establishes a correspondence between the source mesh and the embedded skeleton. Finally, the source mesh is iteratively deformed towards target pose in the guidance of the embedded skeleton by using a new ARAP deformation formulation.

Our contributions in this study are as follows:

- A user-friendly framework to create new poses of an articulated human or non-human character with two basic inputs: a target skeleton (from any source such as motion capture, sketching, [46]) and the source template shape (to be referred as the source mesh from now on).
- A skeleton embedding approach which can also be used as an automatic rigging method.
- A method to augment the source mesh for a novel volume-preserving ARAP deformation.

This chapter is devoted to the first part of the method: Embedding a target skeleton into the source shape. As this part resembles character rigging process, we will first review rigging methods in Section 4.2. Then, we will give an overview of the overall method in 4.3 and briefly explain it in the same section. The second phase is explained in Chapter 5. The related work for the second stage is also provided in that chapter in Section 5.2. In Section 5.3, the skeleton-driven ARAP deformation will briefly be explained. The results and discussions will be provided in Section 5.4 and the conclusions will be provided in Section 5.5.

4.2 Related Work

Most of the previous work about skeleton embedding is related to automatic rigging methods which include both embedding IK skeleton into source mesh and determining blend weight of each mesh vertex.

We have to emphasize that computing an IK skeleton from scratch and embedding an IK skeleton inside of the mesh are different and only the latter one serves our purpose. A well-known example to the former one was proposed by Aujay et al. [47] which constructs the Reeb graph of a mesh by using a Reeb graph computation method [48], and then embeds this graph into the 3D mesh. Machine learning can also be used other than analytical methods for this purpose, a recent approach uses deep neural networks to predict animation skeletons for 3D articulated models [49]. Even though these methods produce appealing IK skeletons, they are not suitable for our purpose. We need the embedded version of the target skeleton defined by the user, i.e, the embedded skeleton should have the same number of joints and edges, and the connection of these primitives should exactly be same with the target skeleton.

A closely related approach was proposed by Baran and Popović [3]. The approach constructs a connected graph using medial surface of the input mesh such that each node represents center of a maximal ball. A discrete penalty function composed of 9 basis functions is defined. The input skeleton is embedded onto the graph by minimizing this penalty function. Learning procedure is required to identify the optimum coefficients of the basis functions. The identification process was intervened manually. As a requirement, the input skeleton and mesh should be in a similar orientation, otherwise the embedding result is wrong. On the contrary, our embedding method is completely orientation agnostic, and no coefficient learning procedure is required.

Instead of trying to fit the input skeleton onto a complex graph, matching it with the curve-skeleton of the input mesh is more promising, which is also what we do in this study. Pourier et al. [50] adopted a similar strategy. They first extract the curve-skeleton using Reeb graph which combines the methods in [47] and [51]. Then the

input skeleton is matched with the curve skeleton using a sub-tree based shape descriptor. The approach can potentially produce good embeddings yet some parameters called length variation threshold, and weight balancing parameter should be tuned by the user, and apparently they seem to be different for each different mesh.

Another IK skeleton embedding approach based on matching with curve-skeleton is proposed by Pantuwong et al. [52]. As their matching method is based on distance field, their extraction algorithm is based on voxel mesh [53]. This implies, if the voxelization resolution is low, then computed skeletons will be of low quality. If the resolution is high, then the computation will take too much time. Also, the method embeds a predefined IK skeleton attached with semantic information from an IK skeleton database rather than directly embedding a skeleton given by the user.

A recent IK skeleton embedding approach [54] also matches the input skeleton with the curve skeleton of the input mesh extracted by mesh contraction [27]. Although the method does not require user involvement, it is only specific for human body models which seriously limits its usage. Stick figures in the form of skeletons are used to repose non-human shapes like chairs [55]. Deep learning techniques also enable reposing of 3D human bodies via 2D stick figures [56]. All of the methods discussed thus far are skinning-based [57] and hence prone to elbow collapses and candy-wrapper artifacts, a problem we alleviate via skeleton-driven ARAP deformation.

4.3 Method

Overall mesh transfer method takes two inputs: a closed 2-manifold triangular source mesh, and a target skeleton which is a 1D connected graph embedded in 3D representing the pose to which the source mesh will be deformed. Its output is a closed 2-manifold triangular mesh representing the new pose of the source mesh dictated by the target skeleton such that the target skeleton is embedded inside of the output mesh. Overall approach consists of two consecutive phases and summarized in Figure 4.1.

The target skeleton is embedded into the source mesh in the first phase. To do this,

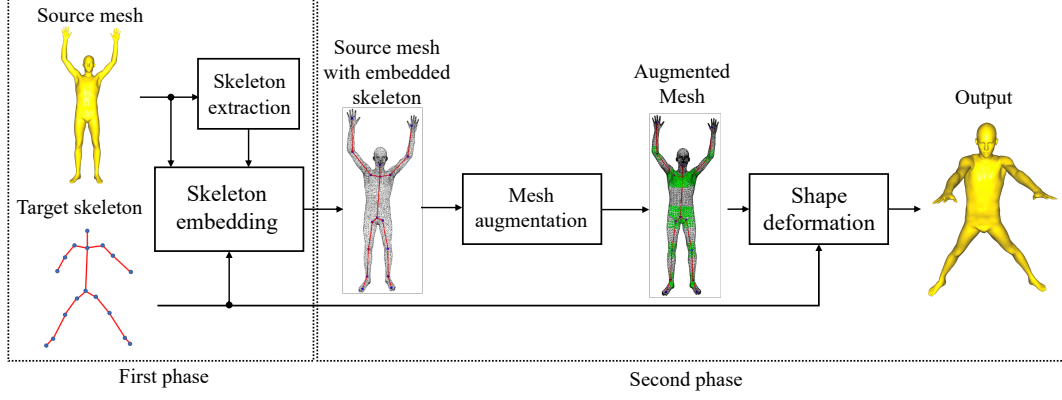


Figure 4.1: The overall method consists of two stages. The first phase takes a source mesh and a target skeleton, and outputs the source mesh with the embedded version of the target skeleton inside of it. The second phase takes the source mesh with embedded skeleton and the target skeleton as inputs. It augments the source mesh by adding additional edges and deforms the shape towards the target skeleton by using ARAP deformation.

firstly, smooth, i.e., high resolution, curve-skeleton of the source mesh is extracted by using Mean Curvature Flow (MCF) approach [27]. This method takes a closed 2-manifold triangular mesh and computes its curve-skeleton consisting of many edges and vertices. Our genetic algorithm based graph matching method matches the target skeleton with the curve-skeleton, embedding the target skeleton onto the curve-skeleton in essence. The output of the first phase is the source mesh with this embedded skeleton in it. Details of the skeleton embedding are given in this chapter.

4.3.1 Dissimilarity of Two Skeletons

To match the target skeleton with the extracted skeleton we define a measure indicating dissimilarity between two curve-skeletons. First we will give the idea in continuous case and then state an approximation in discrete setting.

Let $d_S(\mathbf{x})$ be the distance between a point $\mathbf{x}$ and the closest point on curve-skeleton $\mathcal{S}$ to $\mathbf{x}$. $d_S(\mathbf{x})$ defines a scalar field such that $d_S : \mathbb{R}^3 \rightarrow \mathbb{R}$. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two curve-skeletons. As $d_S(\mathbf{x})$ is a scalar field, curve integral of $d_{\mathcal{S}_2}(\mathbf{x})$ over $\mathcal{S}_1$ gives us a measure of how dissimilar is $\mathcal{S}_1$ from $\mathcal{S}_2$. If we denote dissimilarity of $\mathcal{S}_1$ from $\mathcal{S}_2$

as $D_{\mathcal{S}_2}(\mathcal{S}_1)$, it can be computed with the formula,

$$D_{\mathcal{S}_2}(\mathcal{S}_1) := \int_{\mathcal{S}_1} d_{\mathcal{S}_2}(\mathbf{r}) ds \quad (4.1)$$

where $\mathbf{r}$ is a bijective parametrization of $\mathcal{S}_1$ and ds is infinitesimally small curve element. If we uniformly choose N sample points on curve-skeleton $\mathcal{S}_1$, then Equation 4.1 can also be written as

$$D_{\mathcal{S}_2}(\mathcal{S}_1) = \lim_{N \rightarrow \infty} \left[\frac{1}{N} \sum_{i=0}^N d_{\mathcal{S}_2}(\mathbf{s}_i) \right] \quad (4.2)$$

where $\mathbf{s}_i$ is the i^{th} sample point. Our approximation to Equation 4.1 is based on Equation 4.2.

Notice that $D_{\mathcal{S}_2}(\mathcal{S}_1)$ is non-commutative. This can lead to non-intuitive results. For example in case $\mathcal{S}_2 \subset \mathcal{S}_1$, $D_{\mathcal{S}_2}(\mathcal{S}_1) = 0$ and $D_{\mathcal{S}_1}(\mathcal{S}_2) > 0$ yet it is clear that $\mathcal{S}_1$ and $\mathcal{S}_2$ are not similar and the latter one is correct. We define commutative dissimilarity function D as

$$D(\mathcal{S}_1, \mathcal{S}_2) := D_{\mathcal{S}_2}(\mathcal{S}_1) + D_{\mathcal{S}_1}(\mathcal{S}_2), \quad (4.3)$$

on which our skeleton matching algorithm is based.

4.3.2 Skeleton Matching

A skeleton is represented in our discrete setting as $\mathcal{S} = (V, E)$ where V is the set of vertices and E is the set of edges connecting these vertices. $\mathcal{S}_{\text{trg}} = (V_{\text{trg}}, E_{\text{trg}})$ denotes the target skeleton, $\mathcal{S}_{\text{cs}} = (V_{\text{cs}}, E_{\text{cs}})$ denotes the extracted curve-skeleton of the source mesh, and $\mathcal{S}_{\text{emb}} = (V_{\text{emb}}, E_{\text{emb}})$ denotes the embedded version of the target skeleton. There are three types of vertices: terminal, joint, and regular. A vertex with only one neighbour is called a terminal vertex, with exactly two neighbours is called a regular vertex, and with more than two neighbours is called a joint vertex. A sequence of line segments connected with only regular vertices between joint or terminal vertices is called a skeleton segment.

The definitions above are valid for both $\mathcal{S}_{\text{cs}}$ and $\mathcal{S}_{\text{trg}}$. To embed $\mathcal{S}_{\text{trg}}$ into the source mesh we match terminal vertices of $\mathcal{S}_{\text{trg}}$ with terminal vertices of $\mathcal{S}_{\text{cs}}$ and joint vertices of $\mathcal{S}_{\text{trg}}$ with joint vertices of $\mathcal{S}_{\text{cs}}$. We place a regular vertex of $\mathcal{S}_{\text{trg}}$ on $\mathcal{S}_{\text{cs}}$ by

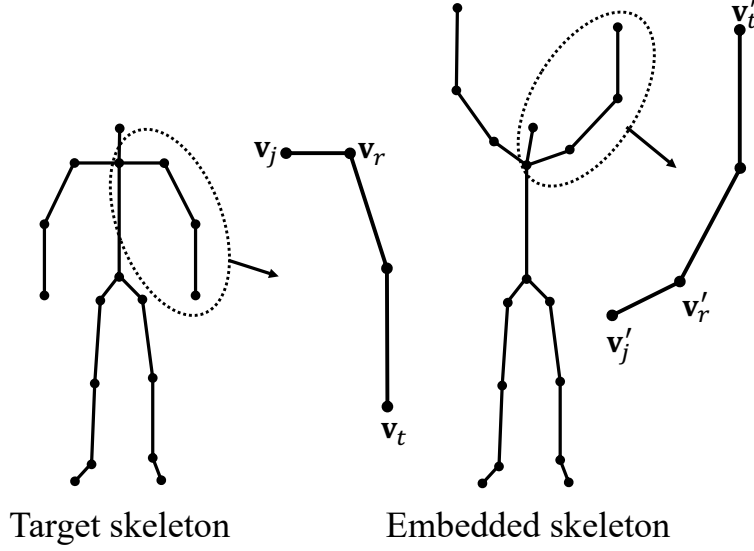


Figure 4.2: A skeleton segment in a target skeleton and its embedded version. $\mathbf{v}_j$ and $\mathbf{v}_t$ are the end points of the segment and $\mathbf{v}'_j$ and $\mathbf{v}'_t$ are their corresponding points in the embedded version. The position of $\mathbf{v}'_r$ is determined according to the parametric position of $\mathbf{v}_r$.

keeping the ratio of geodesic distances from it to the endpoints of the segment it lies on. More specifically, let us consider the setup in Figure 4.2, suppose $\mathbf{v}_j$ matches with $\mathbf{v}'_j$ and $\mathbf{v}_t$ matches with $\mathbf{v}'_t$. We know $\mathbf{v}_r$ and want to compute $\mathbf{v}'_r$. Let $g(\mathbf{v}_1, \mathbf{v}_2)$ be the geodesic distance between $\mathbf{v}_1$ and $\mathbf{v}_2$, let $\mathbf{r}'_{\text{seg}}(t)$ be the parametric representation of the skeleton segment belonging to the embedded skeleton such that $\mathbf{r}'_{\text{seg}}(0) = \mathbf{v}'_j$, $\mathbf{r}'_{\text{seg}}(1) = \mathbf{v}'_t$. Then, $\mathbf{v}'_r$ is computed as

$$\mathbf{v}'_r = \mathbf{r}'_{\text{seg}}\left(\frac{g(\mathbf{v}_j, \mathbf{v}_r)}{g(\mathbf{v}_j, \mathbf{v}_t)}\right). \quad (4.4)$$

The mentioned matching rules are not enough to obtain a valid embedding. Usually many combinations satisfy joint-joint and terminal-terminal matching yet are invalid. An example is shown in Figure 4.3. The segment of the target skeleton representing the right arm is matched with the segment of the extracted skeleton representing the head in this example. This embedding is clearly not what the user wants. A valid embedding to this case is also shown in the figure.

The embedded skeleton should be similar to the extracted skeleton as much as possible, that is, no other matching should give smaller value than $D(\mathcal{S}_{\text{emb}}, \mathcal{S}_{\text{cs}})$ according

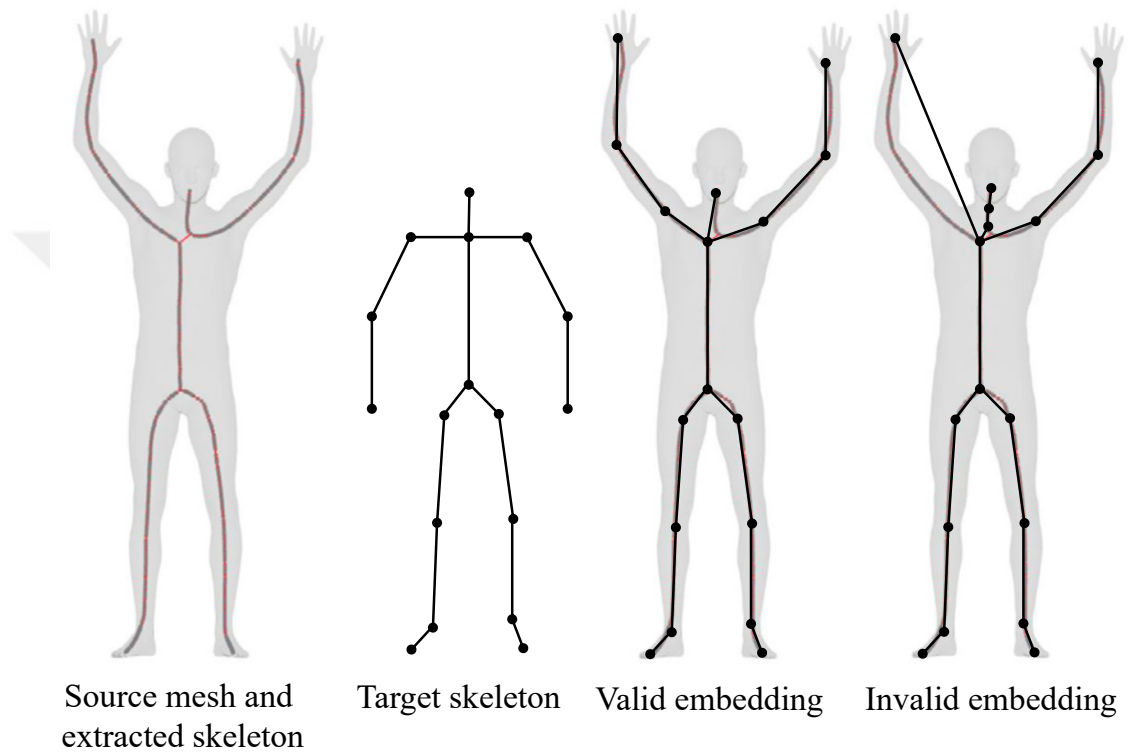


Figure 4.3: First column shows the extracted curve-skeleton of a human mesh to be used as the source. Second column shows a target skeleton. Third column shows a correct matching between the extracted skeleton and the target skeleton. Fourth column is an example to an invalid matching.

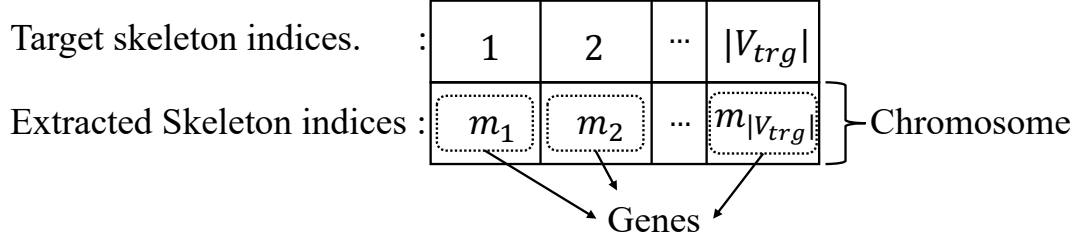


Figure 4.4: Chromosomes and genes in the skeleton matching problem.

to Equation 4.3. We state the optimization problem as follows:

$$\mathcal{S}_{\text{emb}} = \arg \min_{\mathcal{S}} D(\mathcal{S}, \mathcal{S}_{\text{cs}}) \quad (4.5)$$

such that,

- $E = E_{\text{trg}}$,
- Joint vertices of $\mathcal{S}$ match with only joint vertices of $\mathcal{S}_{\text{cs}}$,
- Terminal vertices of $\mathcal{S}$ match with only terminal vertices of $\mathcal{S}_{\text{cs}}$,
- Regular vertices of $\mathcal{S}$ are placed according to Equation 4.4.

This is a constrained non-linear discrete optimization problem with no concept of gradient. Moreover, some parts of an embedding may belong to the optimal solution. These observations as well as the recent success of the evolutionary methods on shape matching [34] suggests us to use genetic algorithm optimization framework for skeleton matching.

Let indices of target skeleton vertices be $(1, 2, \dots, |V_{\text{trg}}|)$ and indices of the extracted curve-skeleton vertices to which they are matched be $(m_1, m_2, \dots, m_{|V_{\text{trg}}|})$. In this representation, i^{th} vertex of the target skeleton matches with m_i^{th} vertex of the extracted skeleton. Please note that regular vertices are not matched and represented by -1 , they instead placed according to Equation 4.4 after joint and terminal vertices are matched. Chromosome and gene definitions are shown in Figure 4.4.

We choose the fitness function f of chromosome χ to be

$$f(\chi) = \frac{1}{D(\text{chrom2Skel}(\chi), \mathcal{S}_{\text{cs}})} \quad (4.6)$$

where $chrom2Skel(\chi)$ is a procedure converting chromosome χ to the skeleton represented by χ .

We chose initial population size to be 40 where chromosomes are initialized randomly obeying terminal-terminal and joint-joint matching rule. The Partially-Mapped Crossover (PMX) [58] strategy is used for crossover operation. During crossover, terminal vertices are matched only with terminal vertices and joint vertices are matched with only joint vertices. Two parents are selected with fitness proportionate selection strategy and two offsprings are produced in each crossover. The population is same during the optimization process, i.e., if a crossover happens, two victims are selected by using inverse fitness proportionate selection. Only one randomly selected gene is swapped in the mutation operation which may follow crossover operation. A terminal vertex is swapped with a terminal vertex, and a joint vertex is swapped with a joint vertex in the mutation. Probabilities of crossover and mutation operations to be applied in a single iteration is 0.8 and 0.4, respectively.

4.3.3 Skeleton Segment Reordering and Vertex Positioning

It is not guaranteed that the genetic optimization will result in the embedding exactly the user desires. For example, imagine a source mesh with two legs and a tail whose lengths are all equal and the configurations of the regular vertices are same. The optimization may end up an embedding in which one of the legs and the tail may be swapped. We allow the user to correct such embeddings by manually swapping some of the skeleton segments in a drag-and-drop manner after the optimization process ends.

Vertex positions of a rigged skeleton often depends on the user's or artist's decision and it is rather subjective [49]. Thus, we also allow the user to change the positions of the vertices of the embedded skeleton after the optimization ends. Yet, please note that, we did not alter the vertex positions of the embedded skeletons manually in all of the experiments in this study.



CHAPTER 5

SKELETON-DRIVEN ARAP DEFORMATION FOR 3D SHAPE TRANSFER

5.1 Introduction

The second phase of our approach is mesh transfer. Like the first one, this phase can also be used on its own. That is, if the user already has an embedded skeleton inside of the source mesh, new poses can be obtained by suitably transforming the bones of the embedded skeleton by using this phase.

At the beginning of this stage, before deformation, the source mesh is augmented. More specifically, the embedded skeleton is sampled and additional edges are added between these sample points and a subset of mesh vertices, which we call support edges. Then the shape is deformed both by using the point-by-point registration between the embedded skeleton and the target skeleton and by minimizing the ARAP energy defined for the augmented mesh. While ARAP deformation scheme preserves the surface details during deformation, the support edges prevents volume loss and undesired artifacts near bending joints.

5.2 Related Work

5.2.1 ARAP Deformation

Preservation of surface details of the source mesh is a necessity for our purpose. ARAP deformation offers this feature in a neat way. It is shown in [6] that it is successful at preserving edge lengths and curvatures at vertices. As the method is originally formulated to keep only surface details, volume loss occurs near articula-

tion joints in pose deformation.

ARAP deformation can be enhanced by adding a regularization term to the ARAP energy equation. Levi et al. [59] enhanced ARAP deformation by adding a regularization term to ARAP energy penalizing the difference between rotation matrices of overlapping cells. Minimizing the ARAP energy results in smoother bendings near articulation joints. It has better volume preservation capability than the original ARAP formulation yet it may smooth out the skin near joints and mesh transfer may end up in unnatural results. The approach we propose is also good at volume preservation and does not smooth out skin near skeleton joints. This issue is briefly discussed in Section 5.4 where we report the comparison of our approach with [59] along with the original ARAP deformation in [6].

Another variant of ARAP deformation is proposed by [55] et al. which allows to define different rigidity for different parts of a mesh. The rigidity is controlled by changing the neighbourhood sizes of the local cells, which is 1-ring in [6]. In this context, while larger neighbourhood size makes the surface more rigid, small neighbourhood size decreases rigidity. A small rigidity is needed near articulation joints so that the skin can adapt to the bending easily. The smallest neighbourhood size is 1-ring in this approach, which is same with the original ARAP deformation. Thus, it does not prevent volume loss near bending areas, making it not suitable for our purpose.

The method proposed by Qin et al. [60] allows even more control on surface deformation. It permits to adjust both overlapping cell sizes, called the global stiffness parameter, and weight coefficient of non-rigidity energy per cell, called local stiffness parameter. It can produce more natural results than the previous approaches provided that local stiffness parameters are adjusted properly. Nonetheless, adjustment of those parameters are non-trivial, they should be either defined manually by an expert, or should be learnt from a set of poses of the same shape. In our approach, neither manual tuning nor example based learning is required to obtain natural deformations.

5.2.2 Skinning

The second stage of our approach resembles skinning methods in the sense that skin deformation of a character is computed from the movement of bones to which the skin is attached. The first and most basic skinning approach LBS [10] deforms vertex positions by linearly blending bone transformations according to the bone influence weights. However, linearly combining rigid transformations violates orthogonality principle [11] which causes volume-loss artifacts near joints known as elbow-collapse in bending and candy-wrapper effect in twisting. To remedy this issue, spherical blending of transformations [61] and usage of dual-quaternions [1] were proposed yet these approaches introduced bulging artifacts near joints.

Several interesting skinning methods have been developed to alleviate those artifacts over time. For example, Rohmer et al. [62] proposed a method that exactly preserves volume of the shape. However, this method requires additional inputs like amount of corrections and profile curves to control the volume preservation which does not fit our requirements. Realistic results can be obtained with physically-based skinning methods like [9] but those methods also need some parameters like Young's modulus and Lamé parameters which differ from shape to shape. It is also possible to learn optimum blend weights by using results of accurate physically-based methods [63]. By introducing new deformer at joints in addition to bones, accurate results can be obtained but still the parameters of physically-based methods should be tuned carefully for the blend weight learning stage.

Jacobson and his colleagues [13] proposed a fast skinning method that integrates hierarchical bone structure and ARAP framework. The method reduces to standard LBS when adapted to our framework without abstract handles. Using abstract handles according to the provided algorithm improves the result but still some previously mentioned artifacts remain (comparisons with this method are provided in Sec. 5.4).

Le et al. [8] proposed to pre-compute optimum rotation centers of the mesh vertices before deformation, which significantly improves the results. However, this method requires careful setting of blend weights which also contradicts with our design re-

quirements. Moreover, the method also causes some volume loss artifacts around places where multiple bones meet like chest.

Other than direct skinning methods, implicit surface based methods may produce high quality results but the quality of the deformation depends on the accuracy of the initial skinning solution [64]. Also, the final pose depends both on the skeleton pose and on the whole animation history, which causes the final pose to be affected by chosen time step [65]. Moreover, those methods require setting of many parameters, adjustment of which may require a few experiments.

In fact, all the methods using blend weights and/or requiring setting of several parameters are not suitable for our framework because performance of those methods highly depends on the blend weights and those parameters. For example, manual adjustment of blend weights by rig artists gives better results than computing them with automatic methods like [3]. Nonetheless, we do not provide any other input to our framework other than the source mesh and target skeleton.

There are some recent skinning methods like [2] not requiring any blend weight to compute the deformed shape. Although appealing results can be obtained with this method, some artifacts still remain especially in highly deformed cases (comparisons with this method are provided in Sec. 5.4).

5.3 Method

Before deforming the shape to obtain new poses, the mesh should be augmented by adding extra edges between its free vertices and the skeleton. Herein, one may be curious about why we add extra edges instead of defining an ARAP energy like,

$$E(\mathcal{M}, \mathcal{M}') = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 + w_{is} \|d_{\mathcal{S}_{\text{emb}}}(\mathbf{p}_i) - d_{\mathcal{S}_{\text{trg}}}(\mathbf{p}'_i)\|^2 \quad (5.1)$$

where $d_{\mathcal{S}_{\text{emb}}}(\mathbf{p}_i)$ is the closest distance of $\mathbf{p}_i$ to the embedded skeleton, $d_{\mathcal{S}_{\text{trg}}}(\mathbf{p}'_i)$ is the closest distance of $\mathbf{p}'_i$ to the target skeleton and w_{is} is an appropriate weight. Nevertheless, such an energy equation cannot be solved for $\mathbf{p}'_i$'s as in the original

ARAP. However, inserting such pointwise correspondences between the mesh and its skeleton as in-between edges into the energy equation results in a much simpler optimization problem.

5.3.1 Selection and Movement of Handle Vertices

In ARAP deformation, a subset of the mesh vertices are selected as handle vertices. The user is only allowed to move those handle vertices freely. Positions of non-handle, also called free vertices, are updated automatically by minimizing the ARAP energy. In a similar fashion, we choose a subset of mesh vertices as handles. Some parts of the source mesh can be considered as almost rigid during deformation, especially the parts faraway to joints. It is suitable to choose those parts as handles. Contrary to tradition, we do not let the user to move those handles directly. Instead, movement of handle vertices are dictated by the skeleton. Thus, we first establish a correspondence between the skeleton and mesh to decide which bones will move which handles.

Each mesh vertex is corresponded with a point on the skeleton. The corresponding point of a mesh vertex is the closest point to it on the skeleton. We assign a parametrization $\mathbf{r}_{\text{bone}}(t), t \in [0, 1]$ to each bone, such that, $\mathbf{r}_{\text{bone}}(0) = \mathbf{u}$ and $\mathbf{r}_{\text{bone}}(1) = \mathbf{v}$ where $\mathbf{u}$ and $\mathbf{v}$ are coordinates of two end points of the bone. With this way, every corresponding point of a mesh vertex has a parametric coordinate. If the length of a bone with joint vertices in both ends is less than the average bone length in the skeleton, we choose all mesh vertices corresponding to that bone as free vertices. For other bones, handle selection procedure is controlled with a parameter $\rho \in [0, 1]$. Let t_i be the parametric coordinate of the corresponding point of mesh vertex $\mathbf{p}_i$ on a bone. We have three possibilities for the bone:

1. If neither $\mathbf{u}$ nor $\mathbf{v}$ is terminal, and if $t_i \in [0.5 - \rho/2, 0.5 + \rho/2]$, then we define $\mathbf{p}_i$ as handle.
2. If $\mathbf{u}$ is terminal but $\mathbf{v}$ is not terminal, and if $t_i \in [0, 0.5 + \rho/2]$, then we define $\mathbf{p}_i$ as handle.
3. If $\mathbf{u}$ is not terminal but $\mathbf{v}$ is terminal, and if $t_i \in [0.5 - \rho/2, 1]$, then we define

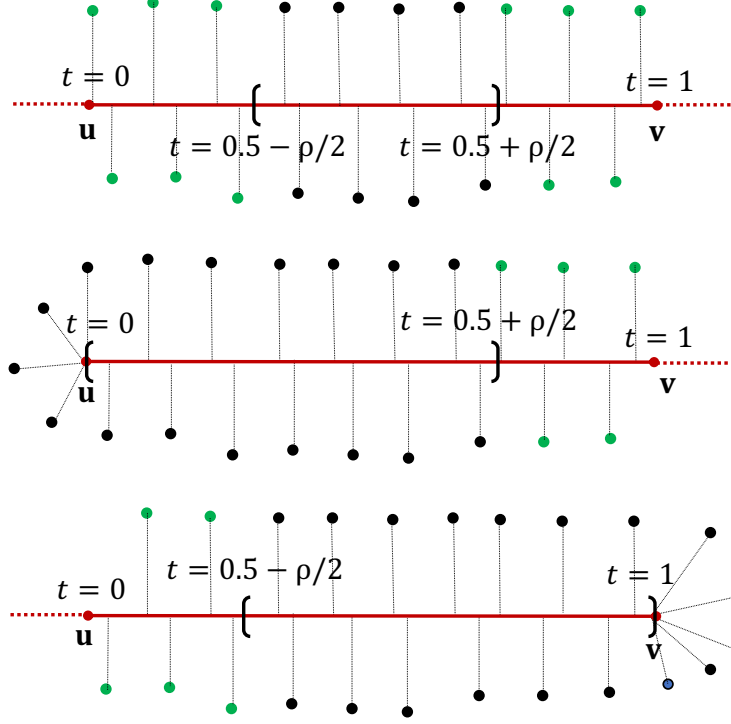


Figure 5.1: Three cases for a bone while selecting free vertices according to ρ in a 2D case. In the top, neither u nor v is terminal, in the middle u is terminal but v is not terminal, in the bottom, u is not terminal but v is terminal. Green points represent free vertices and black points represent handles.

p_i as handle.

Those three cases are depicted in 2D in Figure 5.1 where black points represent handles and green points represent free vertices. With this way the movements of only the mesh vertices near joints will be updated by the minimization of ARAP energy.

Bones in the embedded skeleton are moved to their positions in the target skeleton one by one. The handles corresponding to the points on a bone are moved to the target in the same way as that bone. The angle indicating how much the handles will rotate around the bone can be defined by the user (See Figures 5.9 and 5.10.). If it is not defined, the handles are not rotated around the bone.

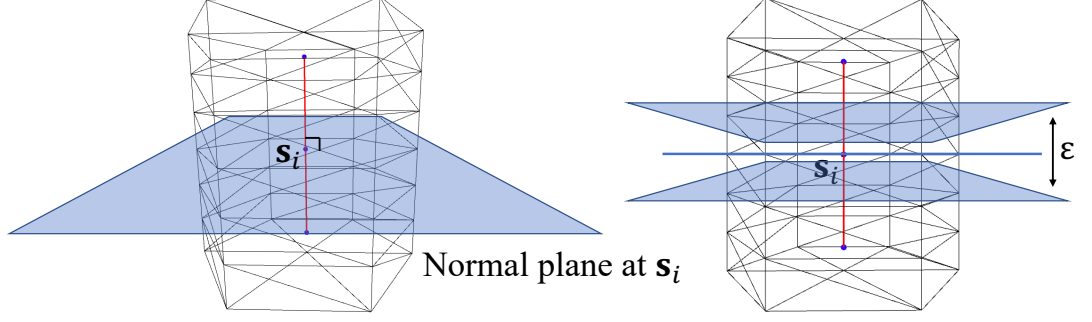


Figure 5.2: Top figure shows the normal plane at point s_i of the skeleton, which is shown as a red stick, inside of a hexagonal prism. Bottom figure shows two parallel planes to the normal plane s_i with ϵ gap in-between.

5.3.2 Mesh Augmentation

The essential point preventing mesh from volume loss artifacts is the additional edges between some sample points on the embedded skeleton and free vertices of the source mesh. We call these edges as support edges because basically they support mesh vertices incident to them and we call the mesh updated with the support edges as the augmented mesh.

At the beginning of mesh augmentation we uniformly choose sample points on the embedded skeleton where the gap between each is ϵ . If we denote i^{th} sample as s_i , we compute normal plane $\mathcal{P}_i$ of the embedded skeleton at s_i as shown at left of Figure 5.2. Then we mark the mesh vertices whose distances to $\mathcal{P}_i$ are smaller than $\epsilon/2$ and which are free as candidates. With this way we actually have chosen as candidate all the free vertices of the mesh between two parallel planes which are also parallel to $\mathcal{P}_i$, and distance between $\mathcal{P}_i$ and both are $\epsilon/2$. Those two planes can be seen at right of Figure 5.2. We chose ϵ to be the average edge length of the source mesh throughout the study.

We do not directly add an edge for each candidate. Firstly, for each candidate, we check whether the edge between it and its corresponding vertex on the skeleton intersects the mesh. If it does, we discard the candidate because support edges should completely be inside of the source mesh. If the sample points on the skeleton are incident to the edges supporting unrelated parts of the mesh, the transfer can result

in inconsistent deformations. Thus, all the support edges emanating from a sample should support the vertices only in the most related part to it. To find the most related mesh part of a sample point, we paint each disconnected free vertex set (They are disconnected by handles.) to a different color. The most related part of a sample point is the free vertex set having the color of the closest one of remaining candidates to it. We then discard the candidates in other than the most related parts.

Some mesh vertices may still be corresponded to more than one skeleton sample after selecting candidates for each sample point in the mentioned way. Multiple support edges may cause inconsistent deformations around neighbourhood of these vertices. Thus, we choose the closest corresponded skeleton sample in multiple correspondence cases and discard the others. We add edges between all remaining corresponding pairs of free mesh vertices and skeleton samples. Support edges between a cylinder mesh and its embedded skeleton can be seen in Figure 5.3. Free vertices of a mesh are indicated with green color and handles are indicated with yellow color both in Figure 5.3 and all other figures in the paper. The support edges for a human mesh are shown in Figure 5.4.

5.3.3 Skeleton-Driven ARAP Deformation

Let us suppose the source mesh is represented as $\mathcal{M}_{\text{src}} = (V_{\text{src}}, E_{\text{src}})$ where V_{src} is the vertex set and E_{src} is the edge set. Let the set of support edges be E_{sup} and additional vertices incident to support edges be V_{sup} . Then the augmented mesh can be represented as $\mathcal{M}_{\text{aug}} = (V_{\text{aug}}, E_{\text{aug}})$ where $V_{\text{aug}} = V_{\text{src}} \cup V_{\text{sup}}$ and $E_{\text{aug}} = E_{\text{src}} \cup E_{\text{sup}}$.

The local cell $\mathcal{C}_i$ corresponding to vertex i consists of the vertex i itself and its 1-ring neighbourhood $\mathcal{N}(i)$. The local cell is defined over the augmented mesh, which means a skeleton edge may be incident to a mesh vertex i . Also, there are no edges between vertices in V_{sup} .

The structure of the local rigidity equation is same with the one in [6]. Namely, if

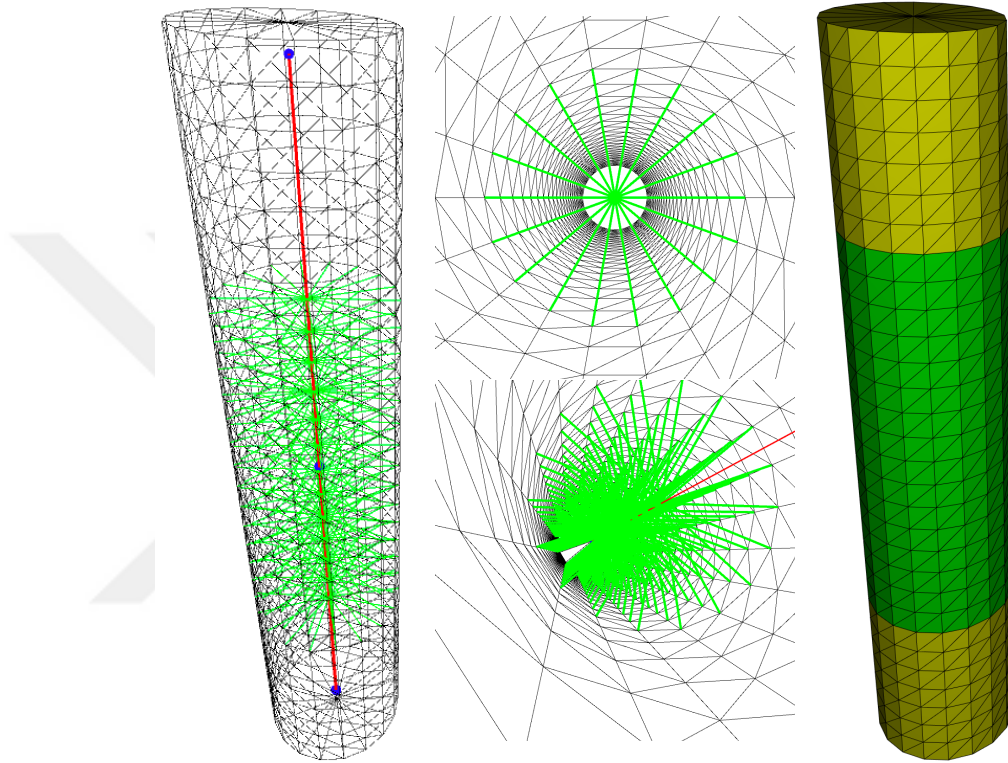


Figure 5.3: Green edges show support edges of a cylinder mesh connecting its skeleton to its surface. The middle subfigures show the same mesh from the inside to reveal the support edges more clearly. Right subfigure shows the source mesh, green region represents free vertices, yellow region represents handles.

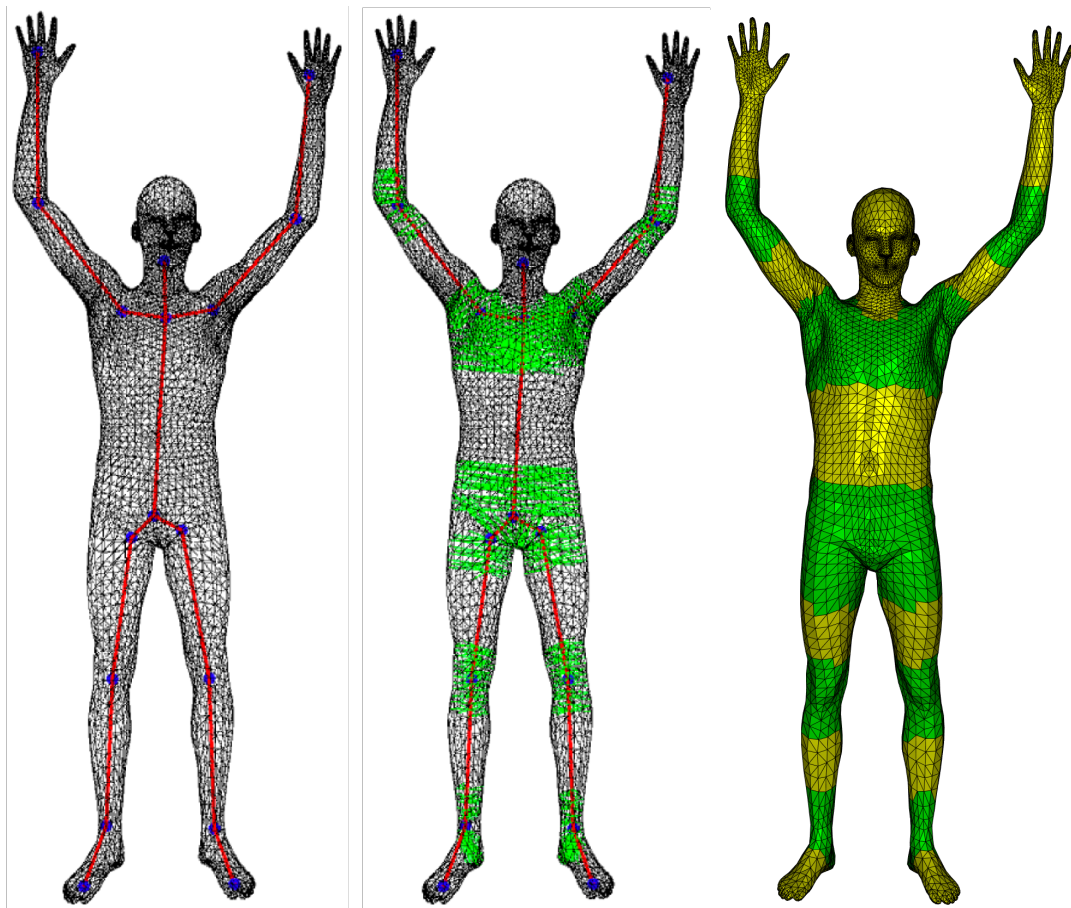


Figure 5.4: Left subfigure shows the skeleton embedded into the human mesh. Middle subfigure shows support edges added to the source mesh which is the augmented mesh. Right subfigure shows the source mesh, green regions represent free vertices, yellow regions represent handles.

we represent local rigidity energy per cell as

$$E(\mathcal{C}_i, \mathcal{C}'_i) = \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \quad (5.2)$$

where $\mathcal{C}_i$ is the cell of i^{th} vertex and $\mathcal{C}'_i$ is its deformed version. However, choosing weights of support edges are tricky here. As $\mathcal{M}_{\text{aug}}$ is not manifold anymore, we cannot compute them with the well known cotangent formula. If we just assign them unit weights, the deformation does not give satisfying results. For example, if w_{ij} 's for non-support edges incident to i^{th} vertex are much smaller than 1, deformation process does not preserve surface details and its goal becomes only preserving the length of the support edge. On the contrary, if they are much greater than 1, the weight of the support edge becomes negligible and the process is reduced to original ARAP deformation. The weight of the support edge incident to i^{th} vertex should be related to the weights of non-support edges incident to it. Choosing the weight of the support edge as the average of weights of non-support edges produce satisfying and consistent results, i.e., deformation process both preserves surface details and prevents the surface from collapsing inwards. Thus, edge weights are computed with the following formula:

$$w_{ij} = \begin{cases} \frac{1}{2} [\cot \alpha_{ij} + \cot \beta_{ij}] & \text{if } (i, j) \in E_{\text{src}} \\ \frac{1}{2|\mathcal{N}'(i)|} \sum_{k \in \mathcal{N}'(i)} [\cot \alpha_{ik} + \cot \beta_{ik}] & \text{if } (i, j) \in E_{\text{sup}} \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

where α_{ij} and β_{ij} are the opposite angles of edge (i, j) and $\mathcal{N}'(i) = \mathcal{N} - \{j\}$.

The ARAP energy is defined as in [6] and computed by using the augmented mesh and its deformed version $\mathcal{M}'$. Namely,

$$E(\mathcal{M}_{\text{aug}}, \mathcal{M}'_{\text{aug}}) = \sum_{i=1}^{N_{\text{aug}}} \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \quad (5.4)$$

where N_{aug} is the number of vertices in the augmented mesh. Notice that we implicitly set the weight of per cell rigidity energy to 1 as suggested.

Solution of Equation 5.4 is approximated iteratively where each iteration consists of two steps. In the first step, $\mathbf{p}'_i$'s are used from the previous step and optimum $\mathbf{R}_i$'s minimizing $E(\mathcal{C}_i, \mathcal{C}'_i)$ in Equation 5.2 are computed. Computation of optimum

$\mathbf{R}_i$'s is briefly explained in [4] and we will not repeat the same explanation here. In the implementation, we used McAdams' fast SVD algorithm for 3 by 3 matrices to compute optimum $\mathbf{R}_i$'s [9].

After computing optimum local rotations in the first step, optimum $\mathbf{p}'_i$'s minimizing $E(\mathcal{M}_{\text{aug}}, \mathcal{M}'_{\text{aug}})$ in Equation 5.4 are computed. Taking derivative of Equation 5.4 w.r.t. $\mathbf{p}'_i$ and doing the required arrangements results in the linear system

$$\mathbf{L}_{\text{aug}} \mathbf{p}' = \mathbf{b}. \quad (5.5)$$

$\mathbf{L}_{\text{aug}}$ in Equation 5.5 differs from the system matrix in original ARAP deformation in the sense that $\mathbf{L}_{\text{aug}}$ contains weights for support edges. Let $\mathcal{H}$ be the set of indices of handle vertices, i.e., the vertices computed with the procedure explained in Section 5.3.1 and skeleton samples incident to a support edge. The elements of $\mathbf{L}_{\text{aug}}$ are computed as,

$$\mathbf{L}_{\text{aug},ij} = \begin{cases} w_{ij} & \text{if } i \neq j \text{ and } i \notin \mathcal{H} \\ -\sum_{j \in \mathcal{N}(i)} w_{ij} & \text{if } i = j \text{ and } i \notin \mathcal{H} \\ 1 & \text{if } i = j \text{ and } i \in \mathcal{H}. \end{cases} \quad (5.6)$$

If $i \in \mathcal{H}$ then $\mathbf{b}_i$ is set to the position of i^{th} vertex which is known.

We first pre-factored the system matrix of the sparse linear system in Equation 5.5 by LU decomposition then solved the system for $\mathbf{p}'_x$, $\mathbf{p}'_y$, and $\mathbf{p}'_z$ independently by using Eigen linear algebra library in C++. The iteration is stopped when $(E' - E)/E < 10^{-5}$ where E' and E are the global rigidity errors in the current iteration and previous iteration, respectively. Please note that, we moved each vertex of the source mesh by applying the transformation of the closest bone (rigid skinning) prior to the iterative solution, and used this shape as the initial solution.

5.4 Results and Discussion

5.4.1 Visual Results

Results of deforming the cylinder shown in Figure 5.3 by bending the two-segment embedded skeleton inside with various angles can be seen in Figure 5.5. As it can be

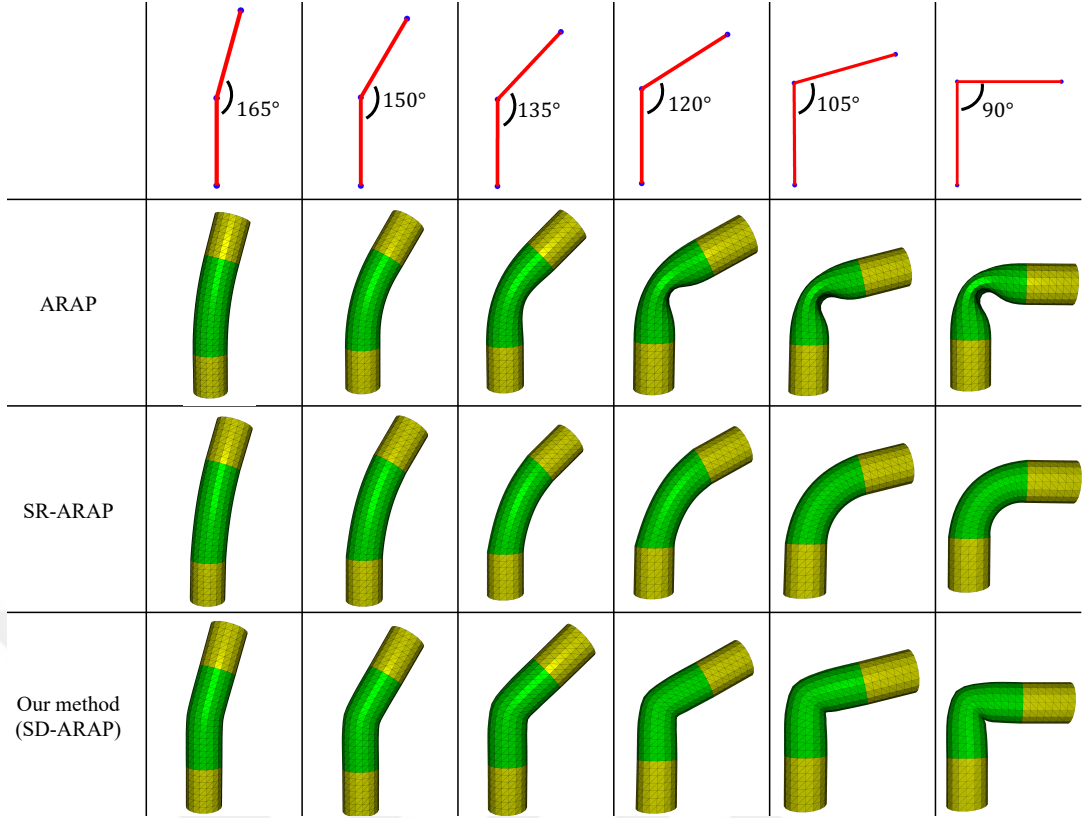


Figure 5.5: Deforming the cylinder shown in Figure 5.3 with various bending angles by using ARAP, SR-ARAP, and our SD-ARAP. Top row shows the target skeletons for various bending angles column by column. Other rows show the deformation results for ARAP, SR-ARAP, and our SD-ARAP, respectively.

seen in the figure, the surface near bending joint collapses towards inside in ARAP deformation especially for large bending angles. SR-ARAP is a remedy to this phenomenon, by generating smooth surface near bending joints it prevents surface collapse. However, although SR-ARAP offers visually appealing smooth results, this may not be what the user wants. For example, the user may want to reflect the sharpness of a 90° corner to the bending shape without surface collapsing. Our method SD-ARAP handles those cases successfully as seen in the last row of Figure 5.5.

Aforementioned more natural-looking results near bending joints for articulated characters can be seen in Figures 5.6 and 5.7 which are from TOSCA dataset [66]. In the first column of Figure 5.6, embedded skeletons, source meshes, and target skele-

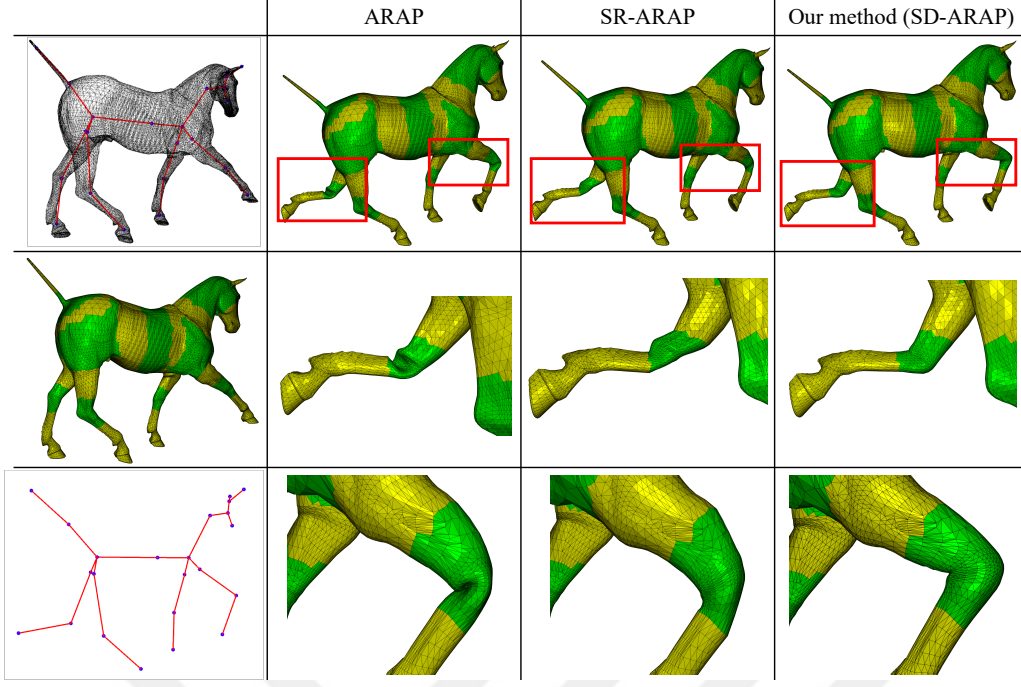


Figure 5.6: Producing a new pose of the horse with the original ARAP, SR-ARAP and our SD-ARAP. The leftmost column shows the embedded skeleton, source mesh and the target skeleton row by row. While the first row of other columns shows the deformed meshes, other two rows zooms to the highlighted regions.

tons are given in row by row, respectively. We did not provide the target skeleton in Figure 5.7 as it is obvious. Results for deformations with different approaches can be seen in the other columns of the first row. The surface collapse happened in the cylinder case also happened for bending joints in the front-right and rear-left legs of the horse with ARAP deformation. The smooth surface generated by SR-ARAP looks unnatural while results generated by SD-ARAP are more natural-looking, as it can be seen in the highlights in Figure 5.6. The same is valid for the tail of the wolf in Figure 5.7 which can be observed in the highlights in the figure.

Different poses of the human figure from MPI FAUST dataset [41] in Figure 5.4 generated by our method can be seen in Figure 5.8. Target skeletons and new poses can be seen in the left column and zoomed regions highlighted in the right column. It can be verified from the figure that surface details are preserved around armpits, shoulders, ankles, waist and knee during the deformation.

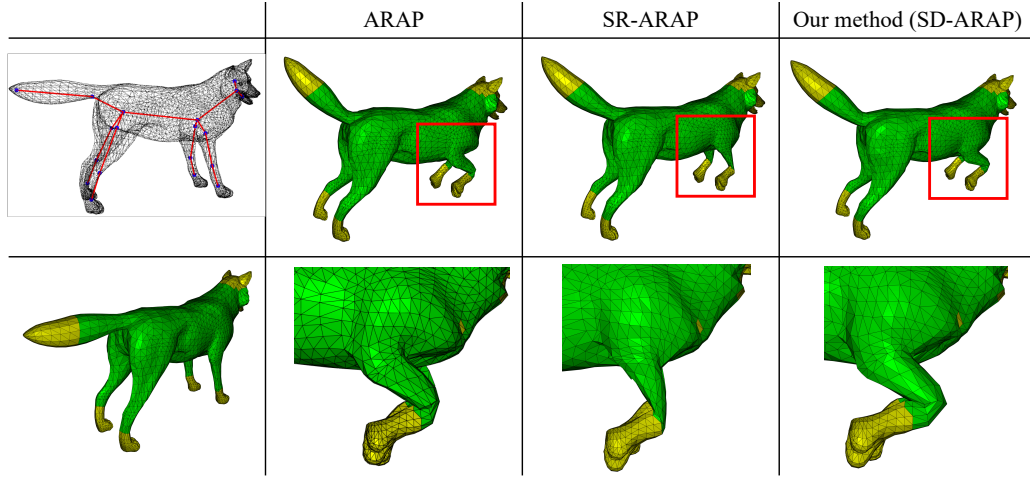


Figure 5.7: Producing a new pose of the wolf figure with the original ARAP, SR-ARAP and our SD-ARAP. The leftmost column shows the embedded skeleton and source mesh row by row. While the first row of other columns shows the deformed meshes, the second row zooms to the highlighted regions.

The user can define rotation around the axis of a target skeleton edge. The effect is formed as a twist near joint regions on the surface. An example is given in Figure 5.9. The embedded skeleton and source mesh of a box is given in the top row. Resulting deformations for 90° and 180° rotations are given in the middle row. Deformation results of adding 60° to the target skeletons are given in the last row. A result of applying rotation around skeleton edge axis is given in Figure 5.10. Here, right arm of the source mesh was rotated 90° , and our method produced a plausible deformation.

We also deformed the source mesh by using classical and recent skinning methods to compare them with our method. Deforming the cylinder in Figure 5.3 as in the same ways in Figure 5.5 with LBS [10], DQS [1], Fast Automatic Skinning Transformations (FAST) [13], and Direct Delta Mush (DDM) [2] are shown in 5.12. The skinning weights for cylinder are shown in Figure 5.11 as a color map. We also generated the same poses given in the top and bottom rows of Figure 5.8 by using LBS, DQS, FAST, and DDM. The skinning weights for the human mesh can be seen in Figure 5.13 as color maps for various bones. Aforementioned and well known volume-loss artifacts occurring with LBS and DQS can be noticed around armpits of the human shape in

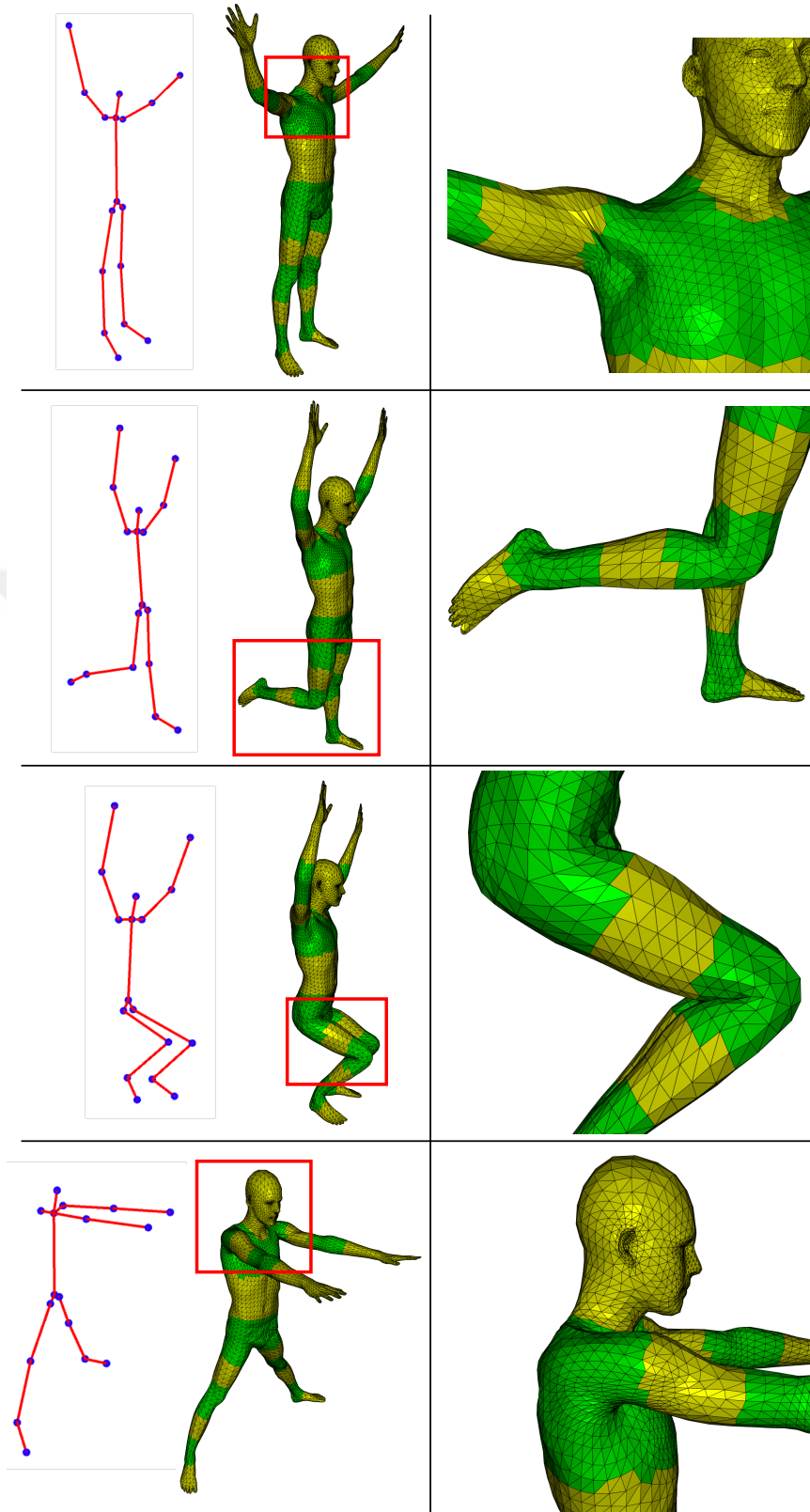


Figure 5.8: Various poses generated by deforming the source mesh in Figure 5.4 towards the target skeletons shown in the left column with the resulting meshes. Right column zooms to the highlighted regions on the results.

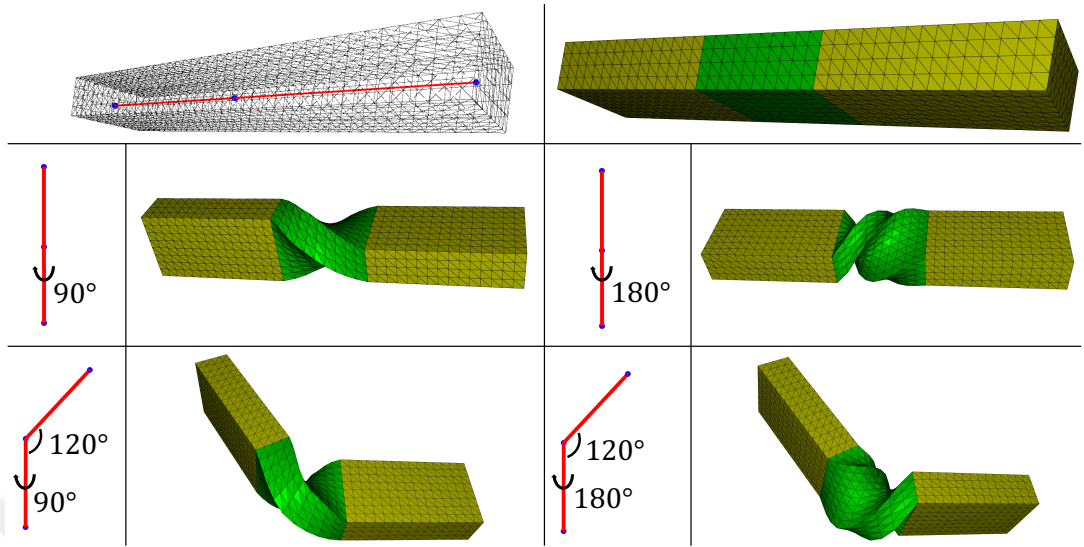


Figure 5.9: Twisting a box by rotating the skeleton edge as shown. The embedded skeleton and source mesh are given in the top row. Middle row shows the results of rotating the edge for 90° and 180° , The last row shows the results of defining 60° bending in addition to the rotations.

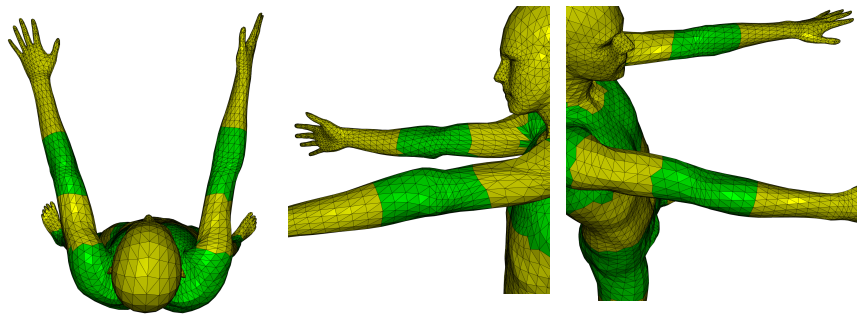


Figure 5.10: Rotating the right arm of a human around its corresponding bone without suffering from the candy-wrapper artifact. Result shown in different views.

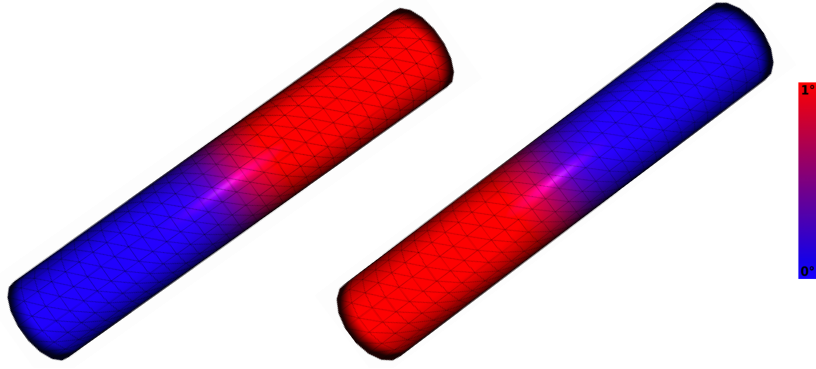


Figure 5.11: Skinning weights computed with heat bone method [3] as a color map for the cylinder. Its source skeleton consists of two bones.

Figure 5.14 obtained by deforming the source in Figure 5.4. FAST reduces to LBS in our framework without using abstract handles. Thus, we enriched the deformation space by adding abstract handles as suggested. Although this improves the results, the deformation around armpits of the human shape is still unnatural. The quality of the deformations can be increased by careful setting of blend weights. However, as our framework aims to minimize human intervention we have to rely on automatic blend weight computation methods which usually do not produce better results than manual setting [3]. DDM is a recent skinning method not requiring any blend weight setting yet the results obtained with that are still not as natural as results obtained by our method.

The difference between our method and skinning methods are much more prominent in twisting cases. While twisting with skinning methods causes distortions and singularities around the twisting joint, our method results in much more intuitive and visually appealing results as can be seen in Figure 5.15.

5.4.2 Quantitative Analysis

Apart from visually inspecting the results, we also analyzed them quantitatively. Volume of a shape in pose deformations should not change as most of the articulated characters are assumed to be consisting of incompressible fluids. We report volumes of the deformed cylinders in Figure 5.5 along with the volume of the source mesh in

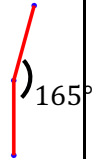
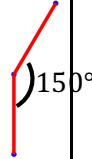
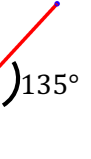
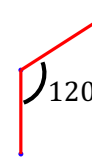
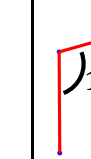
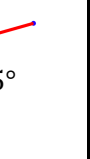
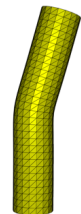
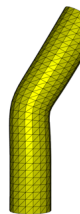
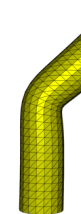
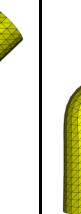
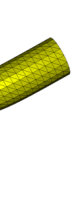
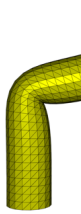
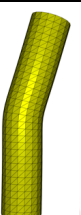
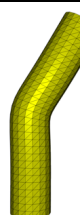
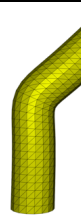
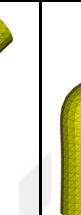
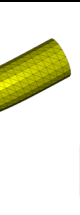
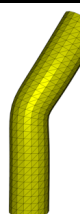
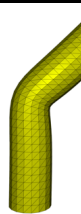
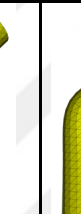
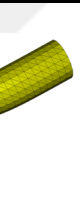
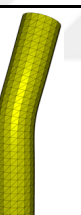
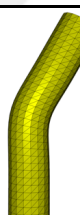
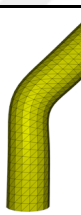
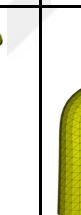
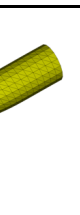
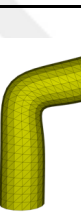
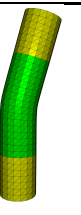
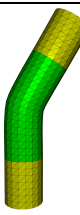
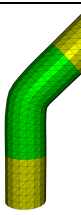
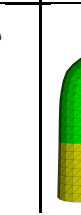
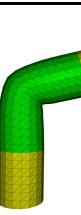
						
LBS						
DQS						
FAST						
DQS						
Our Method (SD-ARAP)						

Figure 5.12: Deforming the cylinder shown in Figure 5.3 with various bending angles using LBS, DQS, FAST, DDM, and our SD-ARAP. Please note that we did not add any abstract handle for FAST as we obtained the best results in that way. The top row shows the target skeletons for various bending angles column by column. Other rows show the deformation results for LBS, DQS, FAST,, DDM, and our SD-ARAP, respectively.

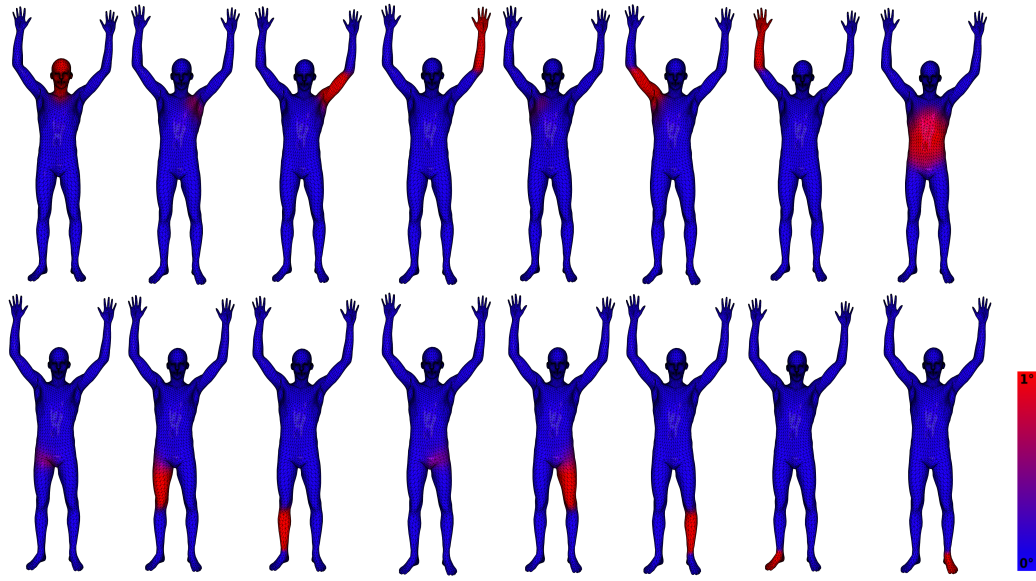


Figure 5.13: Skinning weights computed with heat bone method [3] as a color map for the human shape whose skeleton consists of 16 bones.

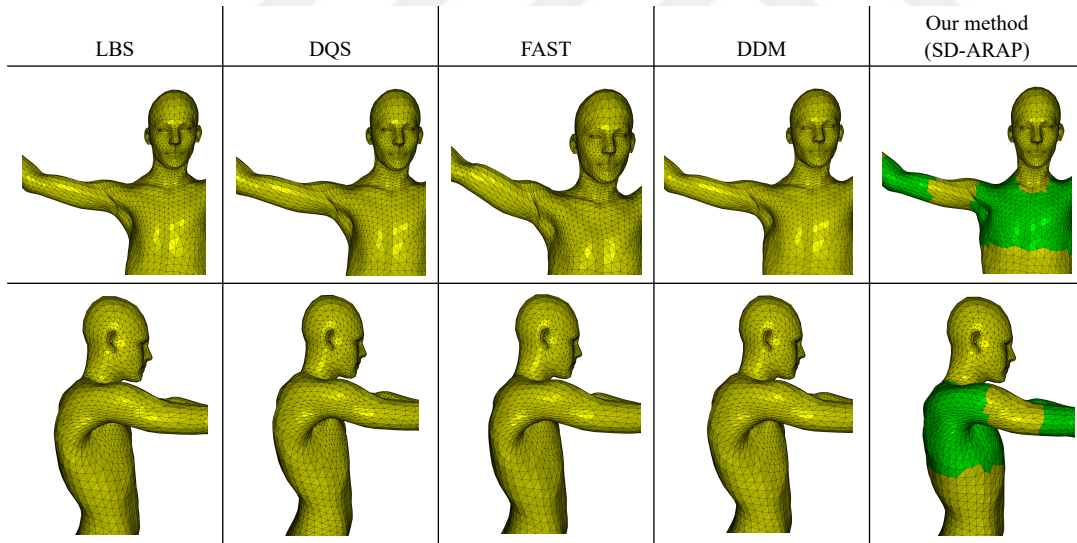


Figure 5.14: Some poses shown at the top and bottom rows of 5.8 are obtained by deforming the source mesh given in 5.4 by using different skinning methods. Volume loss artifacts still occur with the skinning methods around the right armpit of the human while the proposed method results in more natural deformations around those areas.

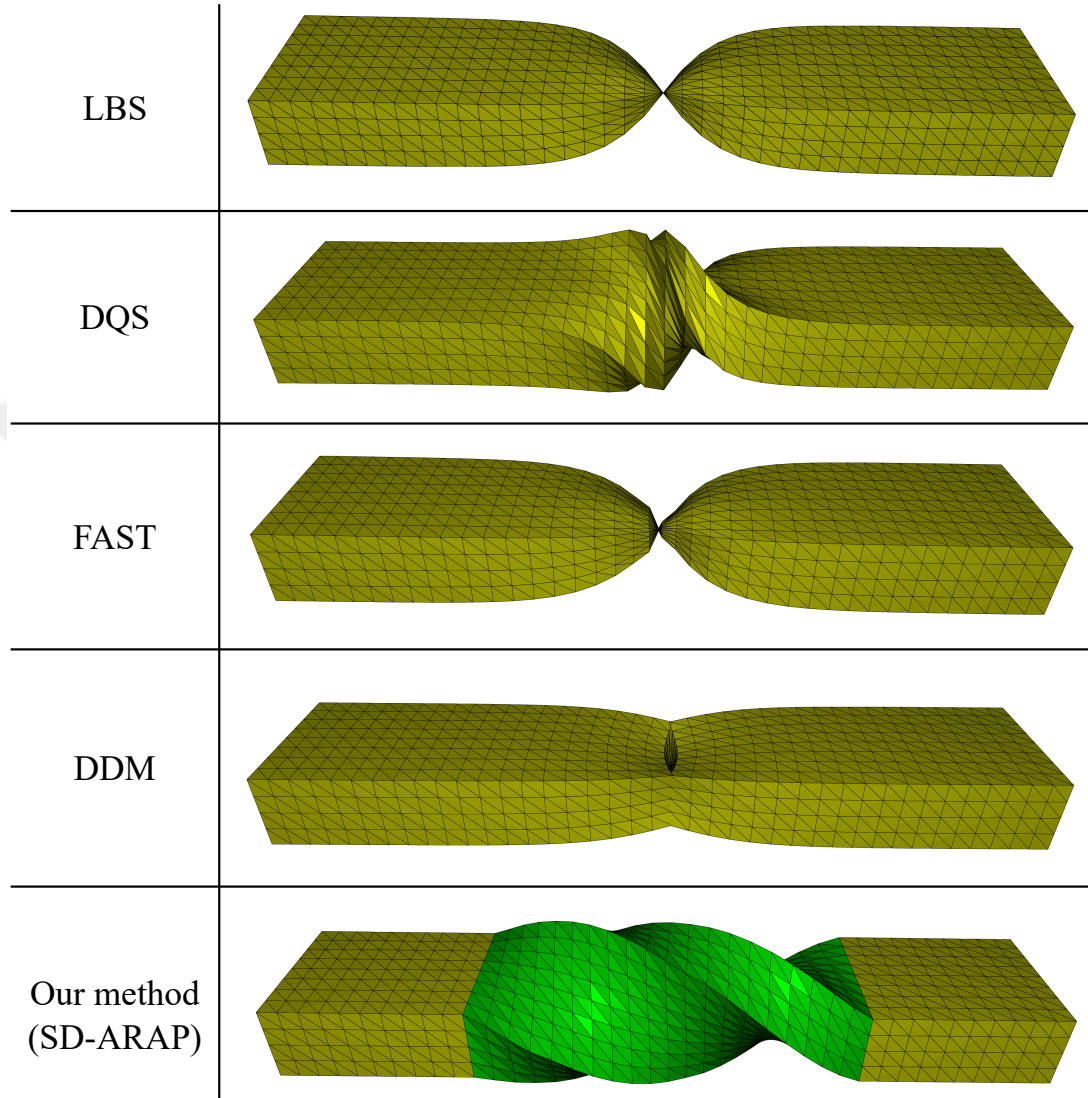


Figure 5.15: The box shown in Figure 5.9 are twisted 180° by using different skinning methods. The proposed method is significantly more successful than skinning methods to handle twisting near joints without requiring any blend weight painting and parameter tuning.

Table 5.1: Change in the volume of the cylinder in Figure 5.3 throughout the deformations given in Figure 5.5. Initial volume V is 30.78. V' represents the volume of the resulting mesh. Percentage changes are also reported in the neighboring columns.

Bending ang.	15°		30°		45°		60°		75°		90°	
$V = 30.78$	V'	Ch. (%)	V'	Ch. (%)	V'	Ch. (%)	V'	Ch. (%)	V'	Ch. (%)	V'	Ch. (%)
ARAP	30.68	-0.33	30.36	-1.36	29.88	-2.95	29.32	-4.73	28.74	-6.64	28.01	-8.72
SR-ARAP	30.69	-0.31	30.41	-1.21	29.97	-2.65	29.39	-4.53	28.72	-6.71	27.99	-9.07
LBS	30.73	-0.17	30.57	-0.68	30.32	-1.49	30	-2.55	29.62	-3.77	29.21	-5.09
DQS	30.78	0	30.77	-0.03	30.76	-0.06	30.75	-0.11	30.73	-0.18	30.70	-0.26
FAST	30.73	-0.17	30.57	-0.68	30.32	-1.49	30	-2.55	29.62	-3.77	29.21	-5.09
DDM	30.73	-0.18	30.6	-0.69	30.32	-1.51	29.99	-2.58	29.61	-3.81	29.21	-5.12
SD-ARAP	30.74	-0.12	30.64	-0.47	30.46	-1.04	30.23	-1.8	29.95	-2.71	29.64	-3.71

Table 5.1. DQS is the best method for volume preservation, however, it is not good at handling twisting cases. SD-ARAP is the second best at volume preservation during deformation and this is more prominent for large bending angles, e.g., the mesh losses $\sim 9\%$ of its volume with ARAP and SR-ARAP in 90° bending, while the loss is only $\sim 3.7\%$ with SD-ARAP.

We quantitatively assessed how well SD-ARAP deformation can approximate to the ground truth deformations in addition to volume loss. To do this, we used the human figure in Figure 5.4 as the source mesh, and tried to obtain other poses in MPI FAUST [41] dataset, which were obtained by using real subject. To compute the target skeletons best describing the poses, we first transferred the skeleton samples of the embedded skeleton into the target pose by using registration between the two poses, and constrained them to lie on linear segments so that we can obtain the target skeleton by combining these samples. We transferred the free vertices of the source mesh to new pose with SD-ARAP by moving handles of the source mesh to their registered vertices on the ground-truth pose and by moving the embedded skeleton to the computed target skeleton. We measured the distance Δ_i between each vertex of the deformed and ground-truth poses. Then we scaled each distance with $1/l_{\mathcal{N}(i)}$ where $l_{\mathcal{N}(i)}$ is the average length of the edges between i^{th} vertex and the vertices in its 1-ring neighbourhood. The results of this experiment are stated

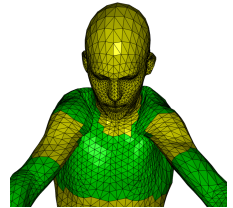
in Figure 5.16 where the target skeleton, transferred mesh, ground-truth mesh, and $\Delta_i/l_{\mathcal{N}(i)}$ values as color codes are given in column by column. Each row shows a different pose.

5.4.3 Timings

We ran all the experiments on a personal desktop system with 2.2 GHz 6-Core Intel® Core™ i7 processor. Average computation times for obtaining new poses are summarized in Table 5.2 in milliseconds for both the original and SD-ARAP. As support vertices and edges are added to the vertex and edge sets of the source mesh, one iteration of SD-ARAP deformation takes more time than one iteration of ARAP. However, number of iterations needed to meet the convergence criteria has a more dominant effect on total time and changes from case to case, e.g., while ARAP needs less number of iterations for human shape, it requires much more for horse and wolf. Thus, we cannot state additional vertices and edges increases deformation time. It should be noted that initial solution affects number of iterations needed for convergence, we achieved the least number of iterations by deforming the source shape with rigid skinning initially. Nevertheless, we did not observe any effect of initial solution on the quality of the final shape. Although the running times does not allow the method to be used in real-time applications like computer games, the overall deformations end in a reasonable amount of time and interactive frame rates are not impossible with a more powerful hardware and advanced sparse matrix manipulation techniques for moderately-sized meshes. Also, please note that mesh augmentation procedure takes very little time if a bounded volume hierarchy is used, as can be seen from "Pre." column in Table 5.2.

5.4.4 Limitations

The extracted skeleton and target skeleton matching procedure described in Section 4.3.2 places the regular vertices from the target skeleton to the embedded skeleton according to Equation 4.4. This implies the user should carefully place the regular vertices



in the target skeleton. If the user places them far away to articulation regions, the

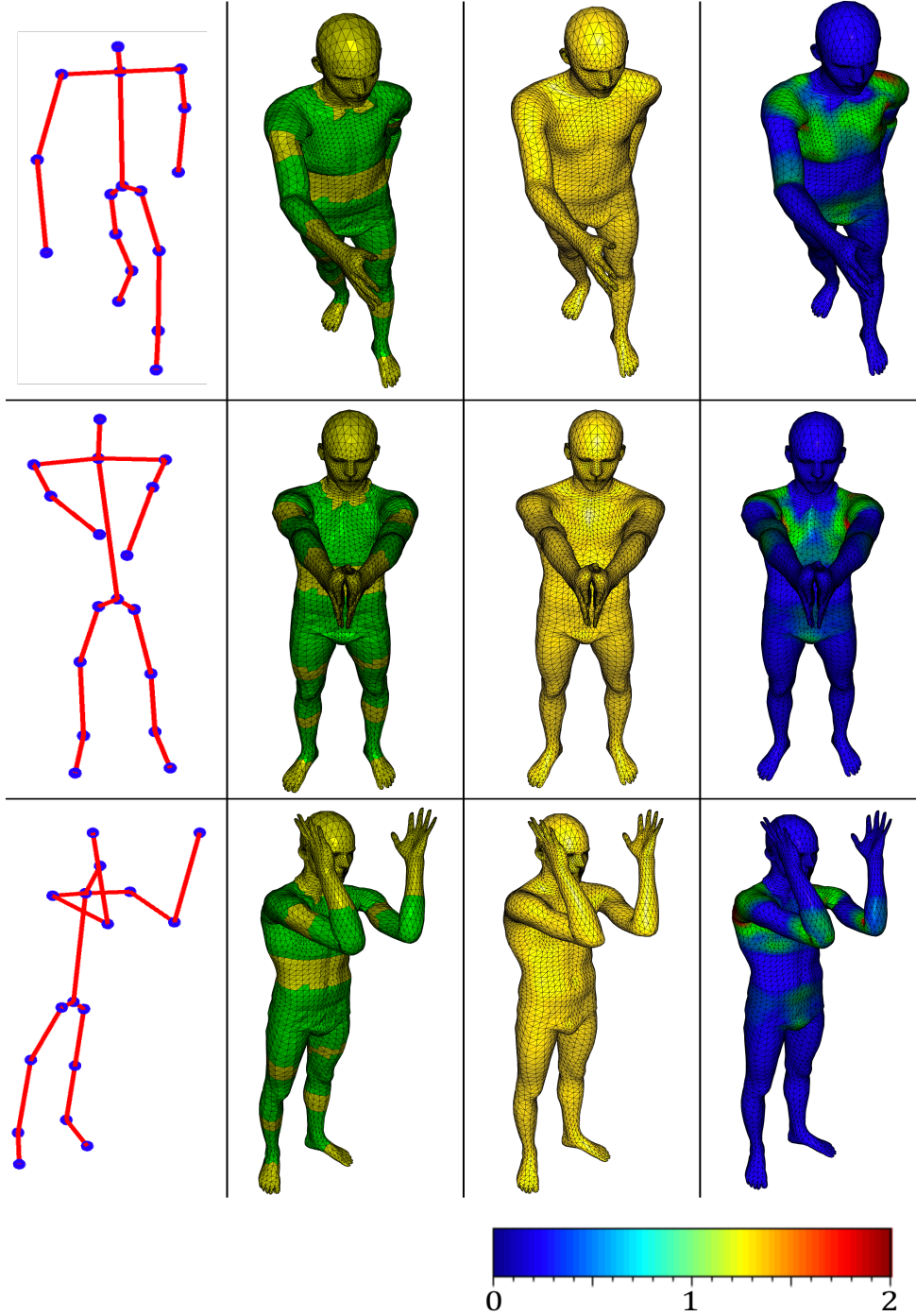


Figure 5.16: Reconstructing some poses in the FAUST dataset. The source mesh is shown in Figure 5.4. Target skeletons are given in the first and transferred meshes in the second column. Ground-truth poses from the dataset are shown in the third column. Displacement errors $\Delta_i/l_{\mathcal{N}(i)}$ between the approximated and ground-truth mesh vertices are shown in the last column as a color code. Average displacement errors on the overall surfaces of meshes in terms of the given metric are 0.189, 0.192, and 0.182 from top to bottom.

Table 5.2: Average computation times for ARAP and our SD-ARAP deformations for human, horse, and wolf meshes in milliseconds, with the relevant vertex and edge set sizes. The column with caption "Pre." shows the total time consumed in mesh augmentation stage and LU factorization of the system matrix.

Source Mesh	$ V_{src} $	$ E_{src} $	ARAP	Num. iter.	Per iter.	$ V_{sup} $	$ E_{sup} $	Pre.	SD-ARAP	Num. iter.	Per iter.
Human	6890	20664	15	4	3.75	131	991	234	42	8	5.25
Horse	19248	57738	2423	168	14.42	226	3291	1106	653	34	19.21
Wolf	4344	13026	335	127	2.64	135	810	25	100	30	3.33

matching procedure has no means to correctly embed them close to articulation regions, which in turn results in unnatural deformations. The figure inset shows a resulting mesh from a target skeleton in which the regular vertices around shoulders are poorly placed. Moreover, our method cannot be used for stretching out limbs of the source mesh as the handles on the mesh surface are transferred rigidly, not non-isometrically, to the target skeleton.

5.4.5 Example Application

We also assessed the performance and robustness of our method on wild data. We used jumping jack motion from Carnegie Mellon University Motion Capture Dataset [67]. More specifically, we used recording of subject 14 and trial 6 for this purpose. The recording was captured by a motion capture system at 120 Hz framerate. The skeleton consists of 31 vertices and 30 bones. We transferred human mesh of FAUST dataset into the skeleton of the subject used in CMU MoCap database for each frame. The transfer results can be seen in Figure 5.17 for some of the frames. The artifact around the shoulders is due the difference between shoulder lengths of the subject in FAUST dataset, and CMU MoCap dataset, which can be considered as a mild case explained in 5.4.4 section (Placement of regular vertices). Those artifacts can be alleviated by a smoothing operation of some other postprocessing. In the figure, raw output is given.

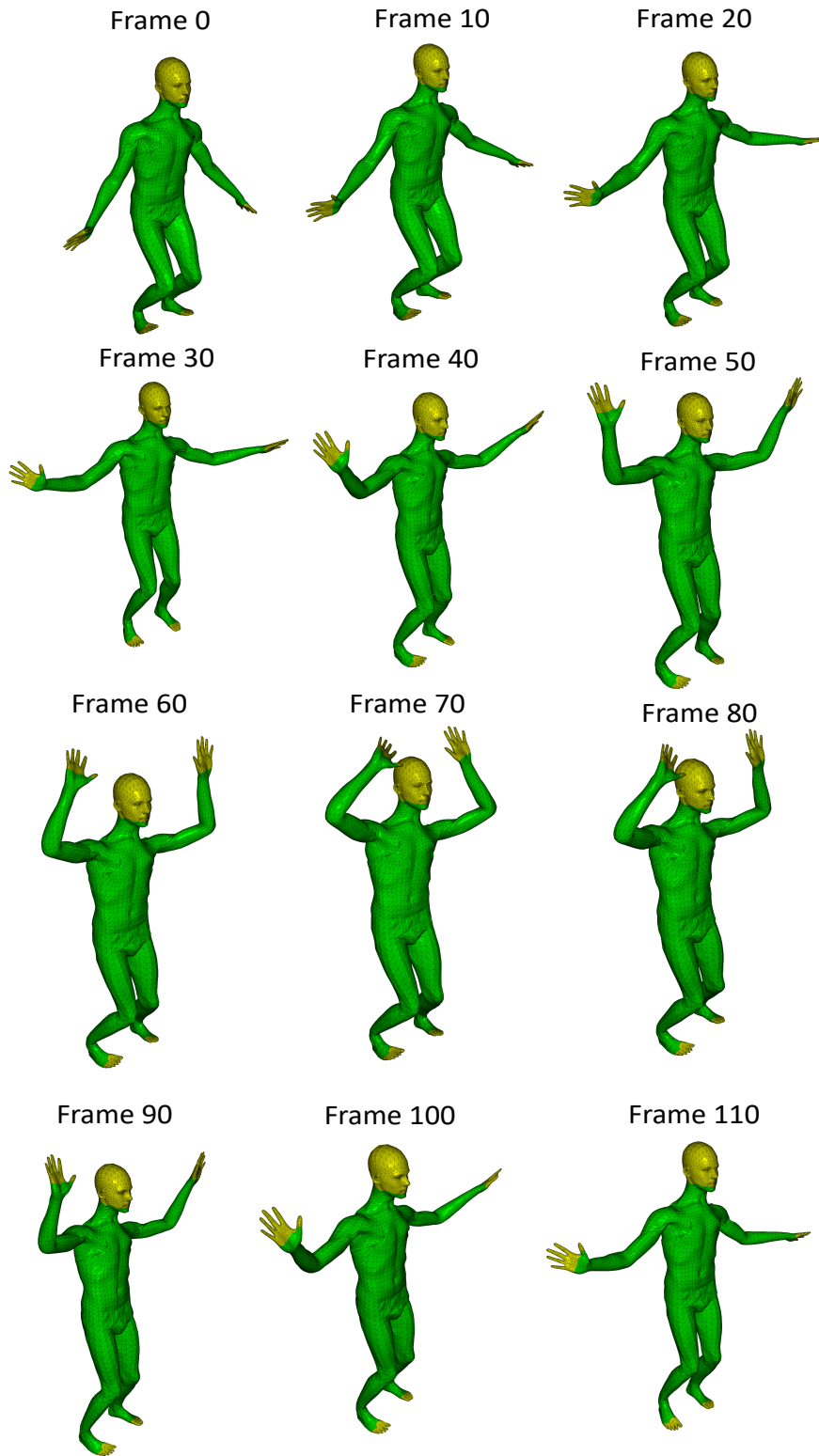


Figure 5.17: Results of transferring human mesh from FAUST dataset to the skeleton obtained from CMU MoCap dataset for some of the frames. The motion was captured at 120 Hz framerate.

5.5 Conclusion and Future Work

We proposed a skeleton based shape deformation method in which the source mesh is first augmented by introducing new edges between the embedded skeleton and the source mesh. Then, the mesh is deformed towards the new pose by utilizing the one-to-one correspondence between the embedded skeleton and target skeleton, and by minimizing an ARAP energy defined for the augmented mesh.

The current version of the method does not consider possible self-intersections of the deformed mesh. One of the future work directions is introducing a contact model, so that we can get rid of self-intersections with bulging effect in a realistic manner. Non-isometry support that would allow additional stretching and squeezing deformations can also be incorporated into our framework. Lastly, the method will be tested more on wild data such as skeletons from motion capture systems and still images to assess the performance of the method on real data.



CHAPTER 6

CONCLUSION AND FUTURE WORK

In the first part of this thesis, we introduced skeleton transfer between shapes in correspondence. The target mesh can have holes on it, or can even be downsampled point clouds. The method can effectively transfer the skeleton between inter-subjects, too. We tested the proposed approach both with human and non-human datasets. As a limitation, when the mesh is downsampled too much (Specifically below $\approx 60\%$), or the hole on the mesh is too large, the accuracy of the method drops significantly. In the study, we handled skeleton transfer between non-rigid shapes such as different poses of the same articulated model. An interesting future research direction would be extending the scope of the work so that it can handle shapes with arbitrarily scaled parts. Also, handling topology difference between source and target shapes can be another future research direction.

In the second part of the thesis, we proposed a two-stage shape deformation method which can be used to repose an articulated character with a 3D stick figure, or transfer a source mesh to a target skeleton. The method consists of two stages: First, the target skeleton is embedded inside of the source mesh so that the pose of the source mesh is captured by the target skeleton. Then, the source mesh is deformed towards the new pose imposed by the target skeleton by exploiting the matching between the source and target skeletons. Our method currently does not consider self-intersections after transfer operation. Although there are methods to handle self intersections, a future research direction can be removing self-intersections by introducing bulging effects in a realistic manner. A second research direction is adding non-isometry support to our framework such that it will support stretching and squeezing deformations. Also, although not a future research direction, design-

ing a more user-friendly GUI to allow the users sketch target skeletons easily is another future work.



REFERENCES

- [1] L. Kavan, S. Collins, J. Žára, and C. O’Sullivan, “Geometric skinning with approximate dual quaternion blending,” *ACM Transactions on Graphics*, vol. 27, no. 4, p. 105, 2008.
- [2] B. H. Le and J. P. Lewis, “Direct delta mush skinning and variants,” *ACM Trans. Graph*, vol. 38, no. 4, 2019.
- [3] I. Baran and J. Popović, “Automatic rigging and animation of 3d characters,” *ACM Transactions on Graphics*, vol. 26, no. 3, p. 72, 2007.
- [4] O. Sorkine and M. Rabinovich, “Least-squares rigid motion using SVD.” Department of Computer Science, ETH Zurich, January 2014.
- [5] M. Botsch and O. Sorkine, “On linear variational surface deformation methods,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, p. 213–230, January 2008.
- [6] S. O. and M. Alexa, “As-rigid-as-possible surface modeling,” in *Proc. of the Fifth Eurographics Sym. on Geometry Processing*, p. 109–116, 2007.
- [7] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, “Discrete differential-geometry operators for triangulated 2-manifolds,” in *Visualization and Mathematics III*, pp. 35–57, 2003.
- [8] B. H. Le and J. K. Hodgins, “Real-time skeletal skinning with optimized centers of rotation,” *ACM Transactions on Graphics*, vol. 35, july 2016.
- [9] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, “Efficient elasticity for character skinning with contact and collisions,” *ACM Transactions on Graphics*, vol. 30, july 2011.
- [10] M. N. Thalmann, R. Laperrière, and D. Thalmann, “Joint-dependent local deformations for hand animation and object grasping,” in *Proc. on Graph. Interface ’88*, p. 26–33, 1989.

- [11] M. Alexa, “Linear combination of transformations,” *ACM Transactions on Graphics*, vol. 21, p. 380–387, July 2002.
- [12] H. A. van der Vorst, “Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, vol. 13, p. 631–644, March 1992.
- [13] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine, “Fast automatic skinning transformations,” *ACM Transactions on Graphics*, vol. 31, no. 4, p. 77, 2012.
- [14] H. Blum, “A transformation for extracting new descriptors of shape,” in *Models for the Perception of Speech and Visual Form*, pp. 362–380, Cambridge: MIT Press, 1967.
- [15] B. Yang, J. Yao, and X. Guo, “Dmat: Deformable medial axis transform for animated mesh approximation,” *Computer Graphics Forum*, 2018.
- [16] T. K. Dey and J. Sun, “Defining and computing curve-skeletons with medial geodesic function,” *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pp. 143–152, 2006.
- [17] N. D. Cornea, D. Silver, and P. Min, “Curve-skeleton properties, applications, and algorithms,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 530–548, 2007.
- [18] P. Li, B. Wang, F. Sun, X. Guo, C. Zhang, and W. Wang, “Q-mat: Computing medial axis transform by quadratic error minimization,” *ACM Transactions on Graphics*, vol. 35, no. 1, p. 8, 2015.
- [19] Y. Yan, K. Sykes, E. Chambers, D. Letscher, and T. Ju, “Erosion thickness on medial axes of 3d shapes,” *ACM Transactions on Graphics*, vol. 35, no. 4, p. 38, 2016.
- [20] J. H. Chuang, C. H. Tsai, and M. C. Ko, “Skeletonisation of three-dimensional object using generalized potential field,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1241–1251, 2000.

- [21] M. K. Livesu and R. Scateni, “Extracting curve-skeletons from digital shapes using occluding contours,” *The Visual Computer*, vol. 29, pp. 907–916, september 2013.
- [22] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, “Mean curvature skeletons,” *Computer Graphics Forum*, vol. 31, no. 5, pp. 1735–1744, 2012.
- [23] J. K. Feldman and S. Manish, “Bayesian estimation of the shape skeleton,” *Proceedings of the National Academy of Sciences*, vol. 103, no. 47, pp. 18014–18019, 2006.
- [24] L. Shapira, A. Shamir, and D. Cohen-Or, “Consistent mesh partitioning and skeletonisation using the shape diameter function,” *The Visual Computer*, vol. 24, pp. 249–259, january 2008.
- [25] A. Tagliasacchi, D. Thomas, M. Spagnuolo, N. Amenta, and A. Telea, “3d skeletons: A state-of-the-art report,” *Computer Graphics Forum*, vol. 35, no. 2, pp. 573–597, 2016.
- [26] S. Schaefer and C. Yuksel, “Example-based skeleton extraction,” in *Symposium on Geometry Processing*, pp. 153–162, 2007.
- [27] O. K. C. Au, C. L. Tai, H. K. Chu, D. Cohen-Or, and T. Y. Lee, “Skeleton extraction by mesh contraction,” *ACM Transactions on Graphics*, vol. 27, no. 3, p. 44, 2008.
- [28] N. Hasler, T. Thormählen, B. Rosenhahn, and H. P. Seidel, “Learning skeletons for shape and pose,” in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 23–30, 2010.
- [29] B. H. Le and Z. Deng, “Robust and accurate skeletal rigging from mesh sequences,” *ACM Transactions on Graphics*, vol. 33, no. 4, p. 84, 2014.
- [30] Y. Sahillioğlu and Y. Yemez, “Coarse-to-fine combinatorial matching for dense isometric shape correspondence,” *Computer Graphics Forum*, vol. 30, no. 5, pp. 1461–1470, 2011.
- [31] V. G. Kim, Y. Lipman, and T. Funkhouser, “Blended intrinsic maps,” *ACM Transactions on Graphics*, vol. 30, no. 4, p. 79, 2011.

- [32] 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.
- [33] D. Nogneng, S. Melzi, E. Rodolà, U. Castellani, M. Bronstein, and M. Ovsjanikov, “Improved functional mappings via product preservation,” *Computer Graphics Forum*, vol. 37, no. 2, pp. 179–190, 2018.
- [34] Y. Sahillioğlu, “A genetic isometric shape correspondence algorithm with adaptive sampling,” *ACM Transactions on Graphics*, vol. 37, no. 5, p. 175, 2018.
- [35] O. K. C. Au, C. L. Tai, D. Cohen-Or, Y. Zheng, and H. Fu, “Electors voting for fast automatic shape correspondence,” *Computer Graphics Forum*, vol. 29, no. 2, pp. 645–654, 2010.
- [36] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, “Skeleton based shape matching and retrieval,” in *Shape Modeling International*, pp. 130–139, 2003.
- [37] Y. Sahillioğlu and M. Sezgin, “Sketch-based articulated 3d shape retrieval,” *IEEE Computer Graphics and Applications*, vol. 37, no. 6, pp. 88–101, 2017.
- [38] Y. Sahillioğlu and L. Kavan, “Detail-preserving mesh unfolding for nonrigid shape retrieval,” *ACM Transactions on Graphics*, vol. 35, no. 3, p. 27, 2016.
- [39] L. Lan, J. Mao, P. Huang, and X. Guo, “Medial-axis-driven shape deformation with volume preservation,” *The Visual Computer*, vol. 33, pp. 789–800, 2017.
- [40] J. M. Thiery, E. Guy, T. Boubekur, and E. Eisemann, “Animated mesh approximation with sphere-meshes,” *ACM Transactions on Graphics*, vol. 35, pp. 1–13, may 2016.
- [41] F. Bogo, J. Romero, M. Loper, and M. J. Black, “FAUST: Dataset and evaluation for 3D mesh registration,” *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pp. 3794 –3801, 2014.
- [42] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, “Numerical geometry of non-rigid shapes,” *Springer Science & Business Media*, 2008.
- [43] R. W. Sumner and J. Popovic, “Deformation transfer for triangle meshes,” *ACM Transactions on Graphics*, vol. 23, pp. 399–405, august 2004.

- [44] Ç. Seylan and Y. Sahillioğlu, “3d skeleton transfer for meshes and clouds,” *Graphical Models*, vol. 105, p. 101041, 2019.
- [45] A. Jacobson and O. Sorkine, “Stretchable and twistable bones for skeletal shape deformation,” *ACM Transactions on Graphics*, vol. 30, no. 6, p. 1–8, 2011.
- [46] V. G. Kim, S. Chaudhuri, L. Guibas, and T. Funkhouser, “Shape2pose: Human-centric shape analysis,” *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.
- [47] G. Aujay, F. Hétroy, F. Lazarus, and C. Depraz, “Harmonic skeleton for realistic character animation,” in *Proc. of the 2007 ACM SIGGRAPH/Eurographics Sym. on Comp. Anim.*, p. 151–160, 2007.
- [48] C. K. McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, “Loops in reeb graphs of 2-manifolds,” in *Proc. of the Nineteenth Annual Sym. on Comp. Geometry*, p. 344–350, Association for Computing Machinery, 2003.
- [49] Z. Xu, Y. Zhou, E. Kalogerakis, L. Chris, C. Landreth, and K. Singh, “Rignet: Neural rigging for articulated characters,” *ACM Transactions on Graphics*, vol. 39, no. 4, 2020.
- [50] M. Poirier and E. Paquette, “Rig retargeting for 3d animation,” in *Proc. of Graph. Interface 2009*, p. 103–110, 2009.
- [51] V. Pascucci, G. Scorzelli, P. Bremer, and A. Mascarenhas, “Robust on-line computation of reeb graphs: Simplicity and speed,” *ACM Transactions on Graphics*, vol. 26, no. 3, p. 58–es, 2007.
- [52] N. Pantuwong and M. Sugimoto, “A novel template-based automatic rigging algorithm,” *Computer Animation and Virtual Worlds*, vol. 23, no. 2, p. 125–141, 2012.
- [53] N. Pantuwong and M. Sugimoto, “Skeleton-growing: A vector-field-based 3d curve-skeleton extraction algorithm,” in *ACM SIGGRAPH ASIA 2010 Sketches*, 2010.
- [54] S. Luo and J. Feng, “Symmetry-aware kinematic skeleton generation of a 3d human body model,” *Multimedia Tools and Applications*, 4 2020.

- [55] S. Chen, L. Gao, Y. Lai, and S. Xia, “Rigidity controllable as-rigid-as-possible shape deformation,” *Graphical Models*, vol. 91, pp. 13–21, 2017.
- [56] A. Akman, Y. Sahillioğlu, and M. Sezgin, “Generation of 3d human models and animations using simple sketches,” in *Proc. of Graph. Interface*, 2020.
- [57] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine, “Bounded biharmonic weights for real-time deformation,” *ACM Transactions on Graphics*, vol. 30, no. 4, p. 78, 2011.
- [58] D. Goldberg and R. Lingle, “Alleles, loci, and the traveling salesman problem,” in *Proc. of the 1st Int. Conf. on Genetic Algorithms*, p. 154–159, 1985.
- [59] Z. Levi and C. Gotsman, “Smooth rotation enhanced as-rigid-as-possible mesh animation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 2, pp. 264–277, 2015.
- [60] X. Qin, T. Wu, and Y. Liu, “A surface deformation method based on stiffness control,” *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 14, no. 1, 2020.
- [61] L. Kavan and J. Žára, “Spherical blend skinning: A real-time deformation of articulated models,” in *Proc. of the 2005 Sym. on Interactive 3D Graphics and Games, I3D ’05*, p. 9–16, 2005.
- [62] D. Rohmer, S. Hahmann, and M.-P. Cani, “Exact volume preserving skinning with shape control,” in *Proc. of the 2009 ACM SIGGRAPH/Eurographics Sym. on Comp. Anim., SCA ’09*, p. 83–92, 2009.
- [63] L. Kavan and O. Sorkine, “Elasticity-inspired deformers for character articulation,” *ACM Transactions on Graphics*, vol. 31, november 2012.
- [64] R. Vaillant, L. Barthe, G. Guennebaud, M. Cani, D. Rohmer, B. Wyvill, O. Gourmel, and M. Paulin, “Implicit skinning: Real-time skin deformation with contact modeling,” *ACM Transactions on Graphics*, vol. 32, no. 4, 2013.
- [65] R. Vaillant, G. Guennebaud, L. Barthe, B. Wyvill, and M. Cani, “Robust iso-surface tracking for interactive character skinning,” *ACM Transactiopns on Graphics*, vol. 33, no. 6, 2014.

- [66] A. Bronstein, M. Bronstein, and R. Kimmel, *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, 1 ed., 2008.
- [67] “Cmu graphics lab motion capture database.” <http://mocap.cs.cmu.edu/>. Accessed: 24.01.2021.





CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Seylan, Çağlar

Nationality: Republic of Turkey (TC)

E-Mail: caglarseylan@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	Middle East Technical University	2021 February
M.S.	Middle East Technical University	2016 February
B.S.	Middle East Technical University	2013 February

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
11.2013 - 11.2020	Middle East Technical Univ.	Research Assistant
11.2020 -	ASELSAN A.Ş.	Software Engineer

PUBLICATIONS

Journal Articles

- [1] Ç. Seylan, Y. Sahillioğlu, "3D Skeleton Transfer for Meshes and Clouds", Graphical Models 105 (2019).
- [2] Ç. Seylan, U. Saranlı, "Estimation of Ground Reaction Forces Using Low-Cost

Instrumented Forearm Crutches", IEEE Trans. Instrum. Meas. 67(6): 1308-1316 (2018).

Master's Thesis

- [3] Ç. Seylan, "Estimation of Ground Reaction Forces Using Forearm Crutches Instrumented with Pressure Sensors and Accelerometers", MSc Thesis, Middle East Technical University (2016).

