Skeletal character animation from examples

Xiao, Xian

2009

Xiao, X. (2009). Skeletal character animation from examples. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47478>

<https://doi.org/10.32657/10356/47478>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:33:25 SGT

288
4x
D/Blue

SKELETAL CHARACTER ANIMATION FROM EXAMPLES

XIAO XIAN

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

Abstract

Accesses to 3D character models are becoming more and more handy and affordable through 3D scanning, reconstruction, artistic sculpting, etc. Animating these characters to tell a story opens up many applications in educations, communications, scientific research and entertainments. Current procedure of character animations requires considerable amount of labor and time dedicated by many skilled and talented artists. Exploring potentials in a character shape together with a set of example meshes in varied poses to enhance the animation methodology will definitely help in producing animations efficiently. However, most of those examples are pure geometry meshes and lack sufficient topology and pose information, which are critical in producing faithful animations. Meanwhile many available example meshes are too complex to use directly for animation production and associated computing demands heavy memory and CPU load. To tackle the above problems, in this dissertation, a set of methods is proposed and investigated to facilitate animation by exploring the potentials of example meshes. These methods, as the results demonstrate, provide an effective pipeline for generating faithful, visually pleasing and pose consistent animations.

In this dissertation, first, a novel method on skeletonization from examples using harmonic one-forms is proposed. Extracting skeletons from 3D objects is challenging, especially for finding exact centered skeletons. The proposed method is motivated by the observation that many real-world deformations are isometric or near isometric from the global point of view. Compared to existing skeletonization methods, using harmonic one-forms bears no restriction on the connectivity of the input meshes, which is a common situation when a character is digitized in varied poses.

Second, a method of pose parameterization of given example meshes is proposed. Pose parameters are suggested in this dissertation as Euler angles and scaling values for skeletal

joints. Arguments for this choice are discussed in details. To recover these parameters from example meshes, a rough skinning model, known as skeletal subspace deformation (SSD), is assumed. Using SSD, a character in rest pose is skinned to the skeleton obtained by the above skeletonization method. Positions of skeletal joints in example poses are found by solving a nonlinear minimization problem. The skeleton structure together with joint positions in example poses serve as input to the parametrization scheme which outputs semantic pose parameters for flexible manipulations. These pose parameters, in turn, are used to enhance the underlying SSD by fitting joint weights using Non-Negative Least Squares (NNLS).

Third, taking the above recovered skeleton and pose parameters as input, an improved shape interpolation method is proposed by inserting an inverse operator to existing Pose Space Deformation (PSD), which interpolates example geometries at runtime. Pose consistent animation is therefore generated. The analysis of reasons why an inverse operator is superior is investigated in details. A practical framework on implementation of the inverse operator in production environment is presented by formulating an optimization problem, which is solved using Powell minimization.

Fourth, SSD is augmented to Spherical Blending Skinning (SBS), an existing method to blend transformations using quaternions, which is integrated with the improved PSD to tackle some difficult animation problems such as elbow twist.

Finally, augmented deformation sensitive decimation (ADSD) is proposed to present example meshes in multiresolution forms, which provides flexible level of details for animations. During the decimation, the connectivities throughout all example meshes are maintained given any specific resolution while large deformations in some examples are respected at the same time. The dimensions of some linear systems dependent on example geometries can be reduced using the decimated examples with guaranteed connectivities.

The results demonstrate that using examples is distinguished from the traditional animation methods since it automates the procedure in skeletonization, skinning (rigging) and posing. Meanwhile, the generated animation is scalable as applications or users desire.

Acknowledgments

I would like to thank Dr. Seah Hock Soon, my supervisor, for his time, effort, and constant support during this research. His personal enthusiasm and serious attitude to research work benefit me well. I am also thankful to Dr. Tian Feng, who gave me a lot of help in my research. I would like to give my special thanks to Dr. Wu Zhongke, his invaluable suggestions, honest criticism and kind encouragement were a constant source of inspiration throughout the research.

I'm indebted to my internship mentor John Lewis on the idea of inverse PSD. His guidance in my coding skill and research methods are invaluable to my progress. I also thank Dr. He Ying for his support and guidance in this research, especially his idea and suggestions on skeletonization using harmonic 1-forms and solving the optimal joints in example poses. I also want to thank Prof. Sylvain Brandel from who I learned great programming style. Deep thanks also go to Dr. Qian Kemao, Miss Li Li, Mr. Qiu Jie, Mr. Chen Quan, Mr. Xie Xuexiang, Mr. Wang Yiming, Mr. Quah Chee Kwang, Mr. Widya Andyardja Weliamto, Mr. Melikhov Konstantin, Miss Wang Jing and Miss Cao Youfang for their warm assistance and encouragement. I'm also grateful for help from EggStoryCP, especially the talented artists Chris, Remi, Sanka and Jack.

I should appreciate the financial support for this research provided by Nanyang Technological University, and the space and equipment provided by CAMTech, CGIT, IERC, PDCC and GameLab. Finally, I would like to take this opportunity to express my gratitude to my family members, my parents and my brother, for their constant support, encouragement and blessings.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	xiii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Statement	3
1.3 Overview of Methods	5
1.4 Organization	6
2 Literature Review	9
2.1 Shape Representation	9
2.1.1 Local Boundary Representations	10
2.1.2 Global Boundary Representations	10
2.1.3 Subdivision Surfaces	11
2.1.4 Surface Reconstruction Methods	12
2.1.5 Multiresolution Representation	13
2.2 Mesh Deformation and Editing	14
2.2.1 Point / Parameter Handles	15
2.2.2 Curve Handles	16
2.2.3 Lattice Handles	17
2.3 Skinning	18
2.3.1 Skeletal Skinning	19
2.3.2 Curve Skinning	21

2.3.3	Non-skeletal Skinning	22
2.3.4	Anatomy based Skinning	24
2.3.5	Skeleton Extraction	24
2.3.6	Harmonic One-Forms and Reeb Graph	25
2.3.7	Recovering Articulated Skinning Model	26
2.4	Animation	28
2.4.1	Key Framing	28
2.4.2	Motion Capture	29
3	Skeletonization from Example Meshes	33
3.1	Differential Geometry	33
3.1.1	Metrics and Isometric Mapping	34
3.1.2	Curvatures	34
3.1.3	Harmonic Scalar function	35
3.1.4	Harmonic One-Forms	37
3.2	Algorithm Overview	38
3.3	Reeb Graph Extraction	39
3.4	Evaluations of Joint Positions	41
3.4.1	Identifying the Joints	41
3.4.2	Optimizing the Joint Locations	42
3.5	Comparison with Alternative Methods	42
3.6	Results and Discussions	44
4	Parameter-based Skeletal Skinning from Examples	47
4.1	Skeletal Subspace Deformation	47
4.1.1	Formulation	48
4.1.2	Limitations	51
4.1.3	Existing Solutions	52
4.2	Optimal Joint Positions in Example Poses	52
4.2.1	Binary Segmentation	52
4.2.2	Conjugate Gradient Method	54
4.3	Semantic Parameters for Example Pose Space	56

4.3.1	Euler Angles	56
4.3.2	From Joint Positions to Euler Angles	57
4.3.3	Scaling Joints	59
4.4	Fitting Joint Weights	60
4.4.1	Least Squares Problems	60
4.4.2	Fitting joint weights through Non-Negative Least Squares	61
4.5	Results and Discussions	62
5	An Optimization Scheme for Example Based Approaches	67
5.1	Scattered Data Interpolation	68
5.1.1	Radial Basis Functions	68
5.1.2	Shape by Example	70
5.1.3	EigenSkin	71
5.2	Pose Space Deformation	72
5.2.1	Formulating Pose Space	72
5.2.2	Algorithm Overview	73
5.2.3	PSD in Practical Applications	75
5.3	Improvement of PSD by Inverse Operation	76
5.3.1	Deformation Direction Analysis	76
5.3.2	Direction Set (Powell's) Methods for Minimization	80
5.3.3	A Unified Framework for Inverse Skinning Model	82
5.3.4	Singular Poses	85
5.4	Results and Discussions	86
6	Skinning Examples upon Spherical Blending	91
6.1	Interpolation in Spherical Space	92
6.1.1	Quaternion	92
6.1.2	Interpolating using Quaternions	94
6.2	Modified Spherical Blend Skinning	95
6.2.1	Singular Value Decomposition	96
6.2.2	Determine Rotation Center	96
6.2.3	Quaternion Rotations Blending	97

6.3	Pose Space Deformation on Spherical Blended Models	98
6.4	Results and Discussions	100
7	Skinning on Progressive Decimated Examples	105
7.1	Introduction	105
7.2	Decimating Example Meshes	107
7.2.1	QSlim Algorithm	107
7.2.2	Augmented Deformation Sensitive Decimation (ADSD)	108
7.3	Skinning on Progressive Models	109
7.3.1	Mapping between Progressive Meshes	110
7.3.2	Automatic Joint Weight Updating	111
7.3.3	Fitting through Decimated Models	111
7.4	Pose Space Deformation on Progressive Decimated Examples	114
7.5	Results and Discussions	115
8	Conclusions	121
8.1	Contributions	121
8.2	Future Works	122
A	Implementation	127
A.1	Software and Libraries	127
A.1.1	Maya Architecture and Programmability	127
A.1.2	Dependent Libraries	129
A.2	Implementation Details	131
A.2.1	Skeleton Subspace Deformation	131
A.2.2	Example Skeletons by Optimization	132
A.2.3	Paramerizaiton of Poses	132
A.2.4	Spheric Blend Skinning	133
A.2.5	Inverse PSD	134
A.2.6	Skinning on Progressive Decimating Examples	136
A.3	System Integration	137
	References	141
	Publication	157

List of Figures

1.1	Problem Space	1
2.1	From the Left to the Right: Polygonal Mesh, NURBS Surface, Subdivision Surface	10
2.2	Multiresolution Representation of Bunny. From the front to the back: 34834 vertices, 5357 vertices, 1216 vertices, 284 vertices.	13
2.3	Barr-Style Deformations (Image Courtesy of [134])	15
2.4	Free-Form Deformation based on Lattice Handles (Image Courtesy of [134])	16
2.5	A Typical Working Scene for Character Skinning (Image Courtesy of [41])	18
2.6	Deformation Style is Reused (Image Courtesy of [60])	21
2.7	Isolines Handles (Image Courtesy of [17])	22
2.8	Mesh-Based Inverse Kinematics (Image Courtesy of [137])	23
2.9	Bouncing Ball	29
2.10	Motion Capture (Image Courtesy of [87])	30
3.1	Harmonic 1-form on genus two surface. The red line in (a) is the boundary.	38
3.2	Algorithm Pipeline for extracting the skeleton using harmonic one forms. (a) Input examples with user specified source point. (b) Harmonic one form. (c) Reeb graph. (d) Joints are identified by evaluating the mean curvature of isolines. (e) Joints are optimized	39
3.3	Skeletons at the reference poses	45
4.1	Illustration of parameterization scheme.	48
4.2	Skeletal Subspace Deformation	49
4.3	Limitations of SSD: left: collapse elbow; right: candy-wrapper.	51

4.4 Joint skin and bone skin segmentation. 52

4.5 Undesirable weight distribution will not influence the evaluation of optimal joint positions by tuning threshold parameter. Left: non-connected weight distribution; Right: joint skin set filtered by weight threshold. 53

4.6 Binary segmentation for horse and camel. Red: joint skin; Green: bone skin. 54

4.7 A non-centered joint is circled, which is resulted by a local minimum in solving Eq. 4.3. 55

4.8 Estimating pose space parameters: JP: parent joint; JCr: child joint in rest pose; JCe: child joint in example pose; q: quaternion rotation; s: scaling value. 57

4.9 Illustration on bone scaling and translating 59

4.10 Joint Weight Fitting: (a)A tube is skinned through SSD; (b)One of five examples from manually re-sculpting; (c)fitted tube corresponding to (b). 61

4.11 Estimating world positions of joints in example poses. 65

4.12 New poses. 66

5.1 1D Radial Basis Function Interpolation 68

5.2 Scattered Interpolation. Left: Shepard’s method; Right: RBF interpolation. (Image Curtesy of [96]) 69

5.3 Pose Space Formulation (Image Curtesy of [96]) 73

5.4 (a)Rest pose; (b)Example pose with rotation of 90 degrees 76

5.5 Deformation direction 77

5.6 A simple test case: two example poses with one vertex sculpted 80

5.7 Interpolated Poses: the rotated joints are at 30, 45 and 60 degree. Left column: PSD; Right column: inverse PSD. 81

5.8 System Pipeline for Unified Inverse Skinning. 84

5.9 **toad**: closeup of circled part from figure 9. left: PSD; right: inverse PSD. 86

5.10 **toad**: a sequence of animated frames. left: PSD; right: inverse PSD. . . 87

5.11 PSD vs. Inverse PSD: with two example poses, one in rest pose and the other is sculpted when the thumb is rotated down 66.8 degrees. SSD shows an obvious collapse. The intermediate poses from SSD, PSD and inverse PSD are shown in the second (45 degrees) and third (60 degrees) rows respectively. 89

5.12 human arm 90

6.1 Illustration for SSD, SBS and PSD 92

6.2 Gimbal Lock 93

6.3 SLERP and QLERP (Image Courtesy of [84]) 94

6.4 Skeleton Subspace Deformation vs. Spherical Blend Skinning 95

6.5 Example based skinning strategies: (a) an example at the rest pose; (b) an example at the pose of “elbow-twist” (rotated -130 degrees along x axis); (c) Blended shape by PSD being layered on SSD (rotated -110 degrees along x axis); (d) Blended shape by PSD being layered on SBS (rotated -110 degrees along x axis). 99

6.6 System Flowchart 100

6.7 Spherical Skinning from examples: Arm; Upper row shows 4 examples; bottom row shows interpolated models from examples 101

6.8 Spherical Skinning from examples: Leg; Upper row shows 2 examples; bottom row shows interpolated models from examples 103

6.9 Spherical Skinning from examples: Hand; Upper row shows 5 examples; bottom row shows interpolated models from examples 104

7.1 Deformation Sensitive Decimation on two cylinder examples with 70% decimations. Left: DSD; Right: Augmented DSD. 108

7.2 An example of ADSD 109

7.3 Mapping from progressive models; From left to right, edge contractions are performed progressively. Large greyed points are corresponded in mapping through multiresolution representations. 110

7.4 vertex split. Image courtesy of Hugues Hoppe [74] 110

7.5	Joint Weight Mapping: (a) joint weight is painted for a mesh with resolution of 30%; (b) mesh of 5% is skinned by automatic mapping from the model on the left.	111
7.6	Two example results from Augmented DSD on 11 horse poses	112
7.7	Fitting examples to approximate joint transformations using $R_i = 0.3$, $R_j = 0.1$ and $M_i = 2535$, $M_j = 850$; (a) decimated examples; (b) fitted examples.	113
7.8	A flowchart of PSD on decimated examples.	114
7.9	Fitting horse model with 10 examples: error of fitting for increasing decimating resolutions. Even for resolution lowered to 0.1, the fitting quality is almost as accurate as resolution of 0.9.	117
7.10	Fitting example using decimated reference meshes. (a) Two decimated reference meshes: Lion(1000 verts) and Cat(723 verts); (b) Fitted examples with original resolutions: Lion (5000 verts) and Cat (7207 verts).	118
7.11	Horse: original mesh has 16843 faces. 3 example meshes are used; Interpolated Poses: Left: 1684 Faces (10%); Middle: 6738 faces (40%); Right: 13474 faces (80%).	118
7.12	Camel: original mesh has 43814 faces. 4 example meshes are used; Interpolated Poses: Left: 4382 faces (10%); Middle: 17526 faces (40%) ; Right: 35052 faces (80%).	119
7.13	Armadillo: original model has 331904 faces. 2 examples are used; Upper row: models with 1% of resolution; Bottom row: models with 30% resolution; Left and middle columns: example poses; Right column: interpolated poses.	120
8.1	Multiresolution Skinning. Left: a rough decimated mesh is deformed by a skeleton through skinning method; Right: a detailed mesh is recovered by propagating deformations through the rough mesh.	123
8.2	Pose Adapted Skinning. Left: a heavy bending arm demanding more vertices to display details; Right: a rest arm with fewer vertices.	124
A.1	Maya Interfaces	128
A.2	Dependent Graph Node for Inverse PSD	136

A.3 A set of plug-in files 137

A.4 User Interface of Character Skinning from Examples 138

A.5 A Working Scene under Maya 139

List of Tables

6.1	Statistics for SBS from examples	102
7.1	Statistics for fitting a decimated horse model: fitting time and required memory is proportional to decimating resolutions	115
7.2	Runtime performance for PSD on decimated models.	116

Chapter 1

Introduction

1.1 Background and Motivation

Figure 1.1: Problem Space

3D animation has been found not only in movies and games, but also in many other fields like scientific simulation, behavior modeling, education, artificial intelligence, etc. In the broad range of 3D animation, which includes rigid motion, cloth and fur simulation or particle effects etc., animating articulated characters is very challenging, since a small artifact can be easily detected by any one who needs not to be a professional animator.

CHAPTER 1. INTRODUCTION

Producing character animation is therefore challenging. It not only requires large amount of time and effort, but also requires animators be talented, well-skilled, expert in wide range of knowledge and capable of discerning slight details with sharp eyes. Hence a large scale production definitely means a long term procedure with heavy labors involved. On the other hand, the demand of art orientated form of 3D animation cannot be easily satisfied by using a uniform mathematic model, although there are many software and tools to assist animation productions.

Producing animation from examples in 2D is a common practice. Key frames are usually drawn first and then, guided by these key frames, in-between frames are generated either manually or automatically. While in 3D, key-framing is a concept borrowed from 2D and the work flow is very different. In 3D, a model of a character is first setup in the rest pose, and then bound to a skeleton or other control objects. The relation between the model and control objects are built through a sequence of tedious experiments, which are called “rigging” or “skinning”. After that, the controllers will be configured with specific values to obtain desired poses at different time frames, which is called “posing”. For other frames, an interpolation scheme will cope with how to calculate the values of control objects with which the shape of the 3D character will be deformed accordingly. Although there is also a key-frame concept, the entire work flow only involves one 3D shape that is setup in the rest pose. This dissertation considers how to animate a character providing a set of shapes in varying poses. Nowadays, more and more well posed character examples are available through artistic drawing, 3D scanning and physical or anatomical modeling. Especially 3D scanning, which measures real 3D objects and produces accurate digital format, is not a costly procedure any more and the relevant technologies such as mesh parameterization and registration are sufficiently mature that digital results from 3D scanning can be used for producing animations. Exploring the capability of these examples for representing visually pleasing or anatomically faithful deformations to assist

animation production is the key goal of this dissertation. To achieve this target, a set of problems need to be explicitly presented in technical point of view. They are illustrated in Figure 1.1 and discussed in details in the next section.

1.2 Problem Statement

How to Extract Skeletons from 3D Objects

In terms of character animation, a skeleton bound to a character is perhaps the most common and convenient tool. Manually building a skeleton with most joints centered in the character is a tedious procedure. And further, placing those joints in physically correct positions is challenging. Many methods are dedicated to this field to seek an exactly centered skeleton for a single 3D object. But most of these results cannot be used directly in animations since the skeleton of a character is usually referred as a tree structure of bones linked by joints. The hierarchical relation and positions of joints should follow basic anatomic rules. Designing a scheme to extract the skeleton of a character by exploring its example geometries in varied poses is thus promising and intuitive given that example meshes are anatomically faithful. Therefore, by considering sufficient number of mesh examples, of which deformations are generated by skeletal joints, correct positions of these joints can be obtained through optimization procedures.

How to Find Pose Parameters

Given example geometries, control objects and a skinning model, appropriate pose parameters corresponding to each example geometry have to be found. Abstractly, this procedure is called parametrization of example geometries as shown in Figure 1.1. A good parametrization can provide a flexible interface to animators to fine tune the parameters. On the other hand, if a runtime interpolation from examples is needed, pose parameters are necessary to serve as interpolation variables.

How to Represent Example Shapes

CHAPTER 1. INTRODUCTION

Almost all shapes or models used in animation productions are represented using polygonal meshes. They are simple, flexible and supported by GPU for rendering. But most of these meshes are too detailed to be used directly, finding a method to represent example shapes with reasonable details to perform animation efficiently is a key issue for this research. Progressive decimation is usually adopted to handle this problem for a static mesh. However for multiple meshes (examples), maintaining the connectivity throughout all examples is critical to animations since example-based approach usually requires runtime interpolation which is performed based on each vertex. Thus, the representation should provide continuous level of details as a user or an application requires and still maintain the connectivity for decimated example meshes.

How to Animate (Deform) a Shape

3D shapes are commonly animated by attaching a reduced control objects such as skeletons, free-form grids, curves or other handles. Manipulating these control objects will drive the movement of original shapes. No matter which types of objects are used, animating 3D shapes can be generalized as a deformation problem: given some predefined constraints (yielding by manipulating control objects or some parts of original shapes), find the physically best positions of all components of the shape. In character animation, this procedure is normally referred as skinning (rigging or developing). In this dissertation, the problem is extended to how to animate a shape with a set of examples in different poses using a skeleton. In other words, considering a shape and its associated skeleton, the task is to find an appropriate relation that indicates which part of shape is influenced by which part of skeleton and by which method.

How to Explore Potential of Examples

Generating a shape or a deformation effect in animation productions is more of an artistic taste than a technical problem. Editing deformations manually in each frame demands tremendous labors and most likely lead to many inconsistent motions. Introducing examples and blending them to generate smooth motion are a natural interaction

CHAPTER 1. INTRODUCTION

and combination of computer automation and human effort. The conventional pipeline of animation should be revised if extra example shapes are available. How to add these examples seamlessly still remains unresolved. Technologically, deformations induced from examples should be respected by the underlying skinning model. In this dissertation, a rough model is first assumed and how to enhance this model based on given examples is investigated.

1.3 Overview of Methods

According to the description of related problems in the previous section, the methods proposed and investigated in this dissertation include 1) Extracting skeletons from examples using harmonic one-forms. The extracted skeleton for a character is well centered, anatomically structured and positioned. 2) Pose parameterization based on extracted skeletons by assuming a rough skinning model Subspace Skeleton Deformation (SSD). The obtained pose parameters together with the given example meshes can improve the SSD model by fitting joint weights. These pose parameters are semantic meaningful for users and can serve as interpolation variables for shape blending. 3) Pose Space Deformation (PSD) is improved by inserting an inverse operation, which is implemented by formulating an optimization problem in production environment. 4) Spherical Blend Skinning (SBS) is modified to remove artifacts induced by the conventional method SSD. The improved PSD is performed on SBS to improve visual qualities. 5) Example meshes will be reconstructed into muliresolution forms using augmented deformation sensitive decimation (ADSD), which can maintain the connectivities of the given example meshes during the decimation and respect the large deformations at the same time. All the above methods will be combined together for applying PSD on progressively decimated examples with recovered pose parameters to generate visually pleasing and pose consistent animation.

1.4 Organization

The proposed methods and relevant techniques involved in this dissertation are a set of encompassing approaches for articulate character animation from examples, which imposes nontraditional work flow, and therefore there are no restrictions on how to utilize the proposed methods to perform a complete animation task. However, the way that these methods are organized in this dissertation is more like a traditional pipeline, which presents tight connections to real productions.

- Related literature reviews given in Chapter 2 include shape representation, mesh editing and deformation, harmonic one-forms and Reeb graph, skinning and animation.
- In Chapter 3, a skeletonization method is presented by applying harmonic one-forms, of which iso-curves and Reeb graph are extracted. Initial joint positions are obtained by measuring the mean curvature from iso-curves. And the final skeleton positions are obtained by solving an optimization scheme by minimizing the distance between Reeb graph and initial joints.
- In Chapter 4, a method to parameterize the example poses is investigated. Pose parameters are defined as a set of Euler angles and scaling values. An optimization scheme is formulated to find these parameters.
- Inverse Pose Space Deformation (IPSD) is analyzed in Chapter 5. A practical optimization scheme is proposed for production environment.
- In Chapter 6, Spherical Blend Skinning (SBS) is modified to replace the conventional method of skeletal subspace deformation. Performing PSD upon SBS as a further step is taken to increase visual qualities.

CHAPTER 1. INTRODUCTION

- In Chapter 7, example meshes are reconstructed using augmented deformation sensitive decimation. Together with parameterization results from Chapter 4, PSD is performed again upon progressive decimated examples to obtain scalable animations.
- Conclusions and Future works are generalized in Chapter 8.
- In Appendix A, implementing issues are presented including several factors such as software framework, dependent libraries, pseudo code and system integration.

Chapter 2

Literature Review

To achieve character animation using computers, a series of techniques will be reviewed in this chapter. First, a character has to be represented by a specific form defined either discretely or analytically. Shape representation is a classic topic which is studied throughout the advancement of computer graphics. Related research is still conducted actively even today. Second, given a shape, spatial changes present another wide research field. For 3D shapes, deformation methods need to follow some geometric or physical laws when inducing changes or movements. Conventionally, deforming a shape is mainly achieved by manipulating some handlers or controllers attached to that shape. How to build the relationship between shapes and their associated controllers for articulated characters is called skinning (rigging or enveloping). Skinning is reviewed in the third part in this dissertation. What follow are animation techniques which are practiced and explored both in industries and academia.

2.1 Shape Representation

Boundary representation (a.k.a B-rep) is introduced in Solid Modeling, which is used to represent shapes by using boundaries to identify the solid and non-solid part. In this dissertation, this concept is borrowed to describe the property of surface models. Regardless of solid part, the boundary of a shape is equivalent to a surface.

Figure 2.1: From the Left to the Right: Polygonal Mesh, NURBS Surface, Subdivision Surface

2.1.1 Local Boundary Representations

The two categories, point clouds [75, 85] and meshes [52, 70] belong to local boundary representations. By local it means that this kind of representations provides no priori global information, such as the figural hierarchy of an object [154]. Nevertheless they are used widely, since the control and manipulation are easy and straightforward. Polygonal mesh comprises geometries together with topology to describe a complete shape. Geometries consist of a set of vertices with Cartesian coordinates (x, y, z) denoting their spatial positions, while topologies are encoded in a set of faces, each of which has a list of vertex indices. Triangle mesh is a subset of polygonal mesh just because each face is a triangle. This type of mesh presents distinctive advantages as each face comprising only three vertices is co-planar, thus algorithms on representing and manipulating are easier to implement and more robust. In contrast to the advantages of meshes such as flexible, easy and robust, this representation tends to have too many redundant faces, say, millions, which is far more than necessary.

2.1.2 Global Boundary Representations

This kind of representations provides a mechanism to represent global structure for objects. By global it means that a succinct mathematic formulation can present a complete

CHAPTER 2. LITERATURE REVIEW

shape. This type is further divided into explicit and implicit forms.

In explicit forms, each surface point has an explicit mathematic expression such as B-spline or NURBS surface [109]. In contrast to explicit models, implicit representations involve an implicit equation to locate the surface points [50]. Different equations produce different models, such as algebraic surfaces [138, 139], superquadrics [23], hyperquadrics [43, 67]. B-Spline or NURBS surface present a set of advantages in computer-aided design applications. They can represent smooth shapes by nature, and are efficient to evaluate. A group of geometric operations are supported as well for the convenience of designers. Explicit representation also provides the ease for texture mapping. Since explicit form is actually a linear sum, local property is well maintained such that manipulating one control point will not influence other points. However, since the topology of a single NURBS surface is equivalent to rectangle sheet, modeling complex shapes is achieved by sewing together a set of individual NURBS surfaces. This operation will dramatically increase the difficulty of NURBS modeling because the smoothness of boundaries of different patches is hard to maintain and seams may occur without carefully designed procedures. Recently, Sederberg et al. propose T-spline surfaces in which control grids permit T-junctions [126]. In contrast, implicit surface can be used to represent complicated shapes such as a tube with multiple branches. It is also much easier to evaluate the inside or outside of implicit shape for a bounded shape. But since adjusting a single parameter of an implicit function may result in a completely different shape, implicit representation usually lacks local properties and employing deformation is therefore difficult. However, some specific forms such as A-patches [21], which present a Bernstein-Bezier form within a tetrahedron, is able to provide local controls.

2.1.3 Subdivision Surfaces

Subdivision surfaces bridge the flexibility of polygon mesh and the smoothness of parametric patches [55]. There are two parts of a subdivision surface: a polygonal mesh as a

CHAPTER 2. LITERATURE REVIEW

base shape and a set of subdivision rules indicating how the base shape should be subdivided. Applying a step of subdivision to the mesh will incur a smoother form through inserting new vertices by predefined rules. Theoretically, unlimited steps of subdivisions should yield a smooth shape. Therefore, subdivision surfaces present nice features from both of polygonal mesh and parametric surfaces (B-spline or NURBS). Flexible manipulations can be applied to the base mesh, and the subdivision operations maintain the smoothness. The limit surface can provide parameterization for texture mapping. Some of the above mentioned representation methods are illustrated in Figure 2.1

2.1.4 Surface Reconstruction Methods

Many surface reconstruction methods addressing different applications have been crafted based on different additional knowledge about the input data. Normally, we can classify this area as follows. First, parametric reconstruction methods dedicate to embedding reconstructed surface $f(S) \subset R^3$ ($S \subset R^2$) in a 2-dimensional parameter domain. Bolle et al. discuss reconstruction of surfaces by a topological embedding $f(S)$ of a plane region S into R^3 [30]. Brinkley considers the reconstruction of surfaces that are slightly deformed spheres [34]. Schmitt et al. choose cylinders [124] while Goshtasby works with cylinders and tori [62]. Second, function reconstruction methods mainly involve building a function f , for a surface S , a set $\{x_i \in D1\}$ and a set $\{y_i \in D2\}$ ($D1 \subset R^3, D2 \subset R^3$), determine $f : D1 \rightarrow D2$, such that $f(x_i) \approx y_i$. Carr et al. use globally supported radial basis functions to fit data points by solving a large dense linear system [36]. Another approach in this direction defines the distance field as the distance to a locally fitted surface. Hoppe et al. define the signed distance field as the distance to a locally fitted tangent plane [76]. Third, a group of approaches is through Delaunay triangulation. The surface is a subset of the faces of the Delaunay triangulation on the data points. Nina et al. present the Power Crust algorithm, which employs the medial axis transformation [11, 12].

2.1.5 Multiresolution Representation

Figure 2.2: Multiresolution Representation of Bunny. From the front to the back: 34834 vertices, 5357 vertices, 1216 vertices, 284 vertices.

The aim of multiresolution representation is to distinguish between high-frequency detail information that has to be preserved and the low-frequency shape that users want to edit [31]. Given a shape with detailed geometries, constructing a multiresolution representation can be achieved by progressive decimation, which leads to a series of surfaces $S_{m-1}, \dots, S_0$ with decreasing level of geometric details. A surface with scale of details which can be specified by users can be found in [116] or by some rules in [73]. Multiresolution representation can also be built from a coarse shape by applying a set of subdivision steps. The resulting shape is smooth but lacks geometric details. Static mesh decimation has been well studied and documented. Luebke provides a survey in the field of polygonal simplification from a developer's point of view [101]. Garland and Heckbert propose the QSlim decimation scheme to perform progressive decimation on polygonal mesh [61]. In this method, if two vertices are to be collapsed, an optimal vertex is found by minimizing the sum of its distances to ingoing planes of the original two vertices, of which adjacent vertices will be re-connected to the newly found vertex.

CHAPTER 2. LITERATURE REVIEW

That sum of distances can be treated as the contraction cost for one edge. Initially, this cost should be zero (no contraction exists for the original mesh). Costs for all edges will be arranged in a priority queue, which will be updated immediately after applying one contraction. In summary, edge contraction will be performed iteratively until the given resolution is reached. Figure 2.2 presents the Bunny with varied resolutions. Mohr and Gleicher proposes deformation sensitive decimation (DSD) by summing corresponding contraction costs from all examples [106]. The contraction strategy is the same as [61], but the order governed by the priority queue is determined by the costs from all associated examples. The value of each component in that queue is a linear sum of contraction costs for corresponding edges in all examples. This work contributes to the scenario that a group of meshes with the same topology is considered, and during decimating the topology should be maintained. In this dissertation, DSD is augmented by adopting a linear sum of squared edge costs as key values for the priority queue. This strategy will nicely preserve large decimation in some particular examples while maintaining topology correspondence.

Kircher and Garland propose a multiresolution representation for deforming surfaces, in which the connectivity is adapted for building suitable hierarchy in each frame [86]. DeCoro and Rusinkiewicz address simplification problems on skeletal models [51]. Huang et al. extend DSD and combine static reference connectivity and dynamic updating to approximate simplified deformations [78]. All these methods cannot maintain topology correspondence and therefore are not suitable for blending example shapes.

2.2 Mesh Deformation and Editing

A single visible shape may not be sufficiently useful in computer graphic applications. Users would like to make manipulations or editions. Deformation is thus a family of methodologies in manipulating and editing shapes as users desired. Angelidis and Singh

Figure 2.3: Barr-Style Deformations (Image Courtesy of [134])

generalize techniques in space deformations for modeling [14]. Deforming shapes is usually achieved by manipulating pre-designed handles attached to the shape. The handles are either visible or non-visible although designing a flexible and easy-to-use handle is a non-trivial task. Given initial shapes and associated handles, a function which defines the relationship between the handles and shapes is what deformation techniques usually address. The function can be either linear or non-linear. Mario Botsch and Olga Sorkine provide a nice survey on recent advances in linear deformations normally constrained by non-skeleton handles [32]. Milliron et al. provide a general treatment for varied free-form deformation techniques [105].

In the following, deformation techniques are reviewed according to the varied handles. Nevertheless, there are deformation techniques in which no user controls are involved such as the geometric transformations proposed by Barr [24]. These deformations, space tapering, twisting and bending etc. (see the Figure 2.3), are generalized as functions of coordinates of initial shapes. Function values yield new coordinates in deformed positions, while the Jacobian matrix of the function helps in evaluating tangent and normal vectors of deformed shapes.

2.2.1 Point / Parameter Handles

An object shape is usually modelled through a mathematic function of a set of parameter control points. Handles can be defined either as object points or as parameter control

CHAPTER 2. LITERATURE REVIEW

points. Deformation is performed by manipulating these points. Manipulating control parameters requires users to have sufficient understanding of the underlying mathematics, while direct tuning of object points is more intuitive. Hsu et al proposed a frame work for deformations with object point handles [77]. In this work object shape is still represented as a function of control points while the deformation is conducted by directly manipulating object points, which would induce an alteration to the control points. Finding a suitable alteration is achieved through a least squares approach such that the deformed object shape reaches the desired destination while the deformation is spread smoothly over the object shape. Besides the intuitive interface, direct manipulation of project points can also yield exact deformation as desired while it is hard to be reached by tuning the control points.

2.2.2 Curve Handles

Figure 2.4: Free-Form Deformation based on Lattice Handles (Image Courtesy of [134])

CHAPTER 2. LITERATURE REVIEW

Singh and Fiume propose a geometric deformation technique by defining wires handle, which is a set of curves whose manipulation deforms the surface of an associated object near the curve [131]. The wire curves give the definition of an object shape and outline its deformable features. This technique mimics the usage of armatures for sculptors and is widely practiced in industries [2]. Lazarus et al. propose axial deformations, in which a 3D axis (represented as a 3D curve) is built and positioned by users for a shape. This axis is then manipulated interactively using conventional curve-editing algorithms, and the induced deformations are passed on to the object shape automatically [93]. Passing the deformations to the object is achieved by the following steps. First, each vertex of the object shape is attached to a point of the axis curve. The attachment is simply performed by finding the closest point in 3D space, although some special cares must be taken when multiple closest points exist. Second, a local coordinate frame is set up for each point on the curve and the local coordinate values of associated vertices of the object are computed. Finally, the deformation of the curve is evaluated and the vertex positions of the object are evaluated through associated local frame.

2.2.3 Lattice Handles

Free form deformation (FFD) is originated from [125], in which solid geometric models are embedded in parallelepiped lattices. Vertices in these lattices serve as control points, on which deformation functions are defined as a tensor product trivariate Bernstein polynomial, which can be extended to trivariate B-Splines [63]. Such formulations inherit the nice properties of parameter functions such as smoothness and local controls. A typical deformation using lattice handles is shown in Figure 2.4. Coquillart extends the free-form deformation to allow arbitrarily shaped deformations by introducing non-parallelepipedical lattices [45], which are created by either designing or editing existing lattices.

CHAPTER 2. LITERATURE REVIEW

MacCracken and Joy use an extension of Catmull-Clark subdivision methodology that refines a 3D lattice into a sequence of lattices that converge uniformly to a region of 3D space [102]. This technique increases the capabilities of lattice deformation by allowing arbitrary topology. Coquillart and Jancene describe an animation technique by employing lattice-based FFD [46]. Chua and Neumann provide a general implementation on GPU for FFD, which reduces the need for software deformation engines and execution time penalty [42].

2.3 Skinning

Figure 2.5: A Typical Working Scene for Character Skinning (Image Courtesy of [41])

CHAPTER 2. LITERATURE REVIEW

Skeletal skinning can be regarded as constrained shape deformation in a more specific context, which usually refers to character animation. Skeletons are manipulated by users, and associated transformations serve as constraints to the skinning method to induce reasonable deformations for skinned shapes. Non-skeletal skinning can be interpreted similarly except that the constraints are formed not from skeleton but something else such as grid forms, sparse markers or isoline handles.

Character animation is a process of creating an animated person or animal. It usually involves creating thought, emotions as well as physical actions, which distinguish character animation from other animation forms. In this dissertation, characters which consist of articulate skeleton structure are considered while non-articulate objects such as cloths, soft tissue, or particles will not be investigated. A typical work scene for skinning a character is illustrated in Figure 2.5.

2.3.1 Skeletal Skinning

Skeletal subspace deformation (SSD) is the most widely used skinning scheme in 3D animation. During runtime animation, the movement of 3D character surface is driven by a weighted linear sum of transformations of associated skeletal joints. SSD is widely used in animation applications due to its simplicity and ease in implementations. Actually, most hardware vendors provide graphic cards which support SSD with at most 4 influencing joints per each vertex. However, due to the simplicity of SSD, its inability in capturing more faithful deformations is also notorious. Underlying reasons are well documented in [84, 96, 143]. Kavan and Zara propose spherical blend skinning (SBS) to remove artifacts induced by SSD [84]. In this method, the rotation parts are first extracted from transformation matrices and further converted into quaternions. A new quaternion induced by weighted linear blending of the above quaternions is converted back to a matrix to serve as a rotation transformation, which together with translation part deform the surface from the rest pose to the desired animated pose.

CHAPTER 2. LITERATURE REVIEW

Example based skinning approaches can be categorized into two types. One involves the runtime interpolation [88, 90, 96, 132] which has names such as pose space deformation (PSD) and shape by examples (SBE). PSD takes each example as a pair of geometric shape and a well-configured skeletal structure as input. Skeletal configurations from examples provide a pose space in which radial basis interpolation function (RBF) is employed to correspond geometries. At runtime, a new shape is synthesized by interpolating example geometries. For those deformations that SSD fails to capture, an interpolation of a set of example shapes can bring an ideal correction at runtime. Different from PSD, Sloan et al. form the interpolation functions by combining RBF with a linear hyperplane and thus it brings constant changes between examples and allows extrapolations [132]. SBE also forms a concept of “shape” which forms a continuous range of meshes across examples. This concept provides a convenient methodology for reparameterization and interactive manipulating the runtime example interpolations. Kry et al. took account of the independence of abstract space dimensions and applied principal component analysis (PCA) to joint support displacements [88]. This strategy thus enforced a limitation of the number of examples at each joint to a few, and guaranteed the runtime performance through vertex programming even the total number of examples is more than one hundred. But in this work the influence of each vertex was constrained by only one joint. Kurihara and Miyata augmented PSD by introducing extra weight values [90]. Rhee et al. implement weighted PSD on GPU [121].

The other category takes examples to augment the underlying parameter-based skinning method [106, 143]. Wang and Phillips extend SSD from one weight per transformation to twelve weights per transformation entry. This method requires sufficient examples and presents some over-fitting problems [143]. On the other hand, Mohr and Gleicher reduce the artifacts by learning examples to improve skeletal structure [106]. New joints are inserted in the place where large deformation occurs. Wang et al. proposed a method by

CHAPTER 2. LITERATURE REVIEW

building a map between skeleton transformation and gradient deformations from examples [142]. Given a new configuration pose, corresponding mesh could be reconstructed from gradient deformations predicated by the mapping.

Figure 2.6: Deformation Style is Reused (Image Courtesy of [60])

2.3.2 Curve Skinning

Curve-skeletons are thinned 1D representations of 3D objects useful for many visualization tasks including virtual navigation, reduced-model formulation, visualization improvement and animation, etc. Cornea et al. provide a comprehensive review on the properties, applications and related algorithms for curve skeleton [47, 48]. Based on the general definition of the curve skeleton there are many works on extraction algorithms which are reviewed in the following paragraphs. Recently, a few works are dedicated to skinning methods using the curve skeleton. Yang et al. propose curve skeleton skinning

CHAPTER 2. LITERATURE REVIEW

based on the observation that most artifacts from the traditional skeletal skinning (SSD) come from linear relationship between the skeleton and the character surface. Therefore they enhance the skinning method by representing the relationship in a non-linear continuous fashion using curve skeleton. Besides, they keep the traditional discrete skeleton to retain the current animation production pattern [152]. Forstmann et al. present a skeletal skinning animation system based on spline-aligned deformations [60]. They introduce deformation styles, which are demonstrated by two sweep-based FFD variants. These two deformation variants are further driven by three textures and three curves which are designed by users. The flexible usage of deformation style is a desired feature for animators as illustrated in Figure 2.6.

Figure 2.7: Isolines Handles (Image Courtesy of [17])

2.3.3 Non-skeletal Skinning

Instead of using a centered skeleton as controlling object to drive character animation, many other alternatives are practiced in productions. One popular choice is free form deformation (FFD), which was first introduced by Sederberg and Parry [125]. The general concept and methodology of FFD are reviewed in Section 2.2.3. Here only its usage in character skinning is highlighted. Character skin is embedded into a box shaped lattice with control points and a mapping function is formulated between control points and original skin points. Deformations are achieved by manipulating the control points. Chadwick et al. propose a layered framework by combining the influence of skeleton

CHAPTER 2. LITERATURE REVIEW

and free form lattices [37]. In this framework, some physical deformation effects such as muscle and fatty tissues can be loosely represented in FFD lattice. Another hybrid method which combines skeleton and free-form deformations is proposed [153]. A voronoi-based skeleton is first extracted, and then modified by free-form deformations.

Figure 2.8: Mesh-Based Inverse Kinematics (Image Courtesy of [137])

Very recently, a group of methods is designed to deform characters by manipulating a subset of vertices on the surface mesh [54, 136, 137], in which a group of example meshes with the same connectivity structure is represented using deformation gradient vectors [135]. Gradient vectors encode the relation of a surface vertex to its neighbors. To generate a new shape which is supposed to be the most representative to given examples, an optimization scheme is first built to evaluate interpolation parameters while treating positions of edited vertices as constraints. In addition to gradient vectors, these parameters form the feature space. And a new shape is generated as close to this space as possible. Compared to [137], in which interpolation parameters are dependent on geometric complexities, [54] explores the correlations of adjacent vertices and solves the parameters independent of geometric complexities and allows interactive editing for extremely detailed geometries. The above methods perform skinning without skeleton but bear some shortcomings in common. First there must be enough examples to train the model. Second, the synthesized mesh presents strong dependence on examples [144]. Weber et al. first build an underlying skeletal skinning scheme similar to [100, 155] and

CHAPTER 2. LITERATURE REVIEW

then apply runtime interpolation on examples which are represented as gradient vectors [144].

Au et al. adopt vertex handles to produce a set of isolines computed from a harmonic scalar field representing the deformation propagation from those handles [17]. In other words, a handle corresponds to a set of isolines, which are intelligently distributed across the surface for that handle and associated transformations are actually control variables. The set of isolines can also be treated as a mimic of skeletons, which employs the skinning by solving a non-linear system iteratively. This scheme is demonstrated in Figure 2.7.

2.3.4 Anatomy based Skinning

Physical or anatomical simulation can produce extremely realistic skin deformation at high computational costs. Anatomical models for character animation have been explored extensively. Chen and Zeltzer develop a computational model of skeletal muscle [38]. Wilhelms and Gelder provide a framework to model and animate underlying muscles, bones, and tissues for animals [146]. Scheepers et al. design a muscle model that is able to simulate muscle deformations according to the changes of an underlying articulated skeleton [123]. Aubel and Thalmann generalize an interactive tool for modelling and animating realistic human skin by building a multi-layered framework [19, 20]. Pratscher et al. take outermost skin surface as input to generate an anatomical model for realistic skin deformation [119]. Pollard and Zordan extract a controller from example motion data to physically simulate hand grasping [117].

2.3.5 Skeleton Extraction

Although extracting skeletons from 3D objects is beyond the scope of this dissertation, skeletons are important and critical for character animation. Recently, research works on skeleton extraction which are particularly used for character animation are actively

CHAPTER 2. LITERATURE REVIEW

conducted [18, 6, 68, 122, 140]. In this subsection, a selected review is done regarding to applications in character animation. Besides animation, skeletonization from 3D shapes is also widely explored for segmentation [99, 98], collision detection [33, 79], shape description and analysis [71, 127, 128], and navigation [97]. For character animation, the skeleton is represented as a hierarchical structure that is conventionally represented by a tree data structure. The overall methodology is divided into transformation-based [112], clustering-based [122, 82], and semantics-based [53]. Recently, Schaefer and Yuksel propose a novel method to extract hierarchical, rigid skeletons from example poses [122]. Aujay et al. presented the harmonic skeleton in which anatomical information is used to enhance the skeletons [6]. Au et al. perform skeleton extraction by mesh contraction [18].

2.3.6 Harmonic One-Forms and Reeb Graph

A scalar function $f : M \rightarrow \mathbb{R}$ on manifold M is harmonic if f minimizes the harmonic energy $\int_M |\nabla f|^2$. The harmonic one-form of f defined as the gradient ∇f is also harmonic on M . Harmonic 1-form plays a critical role in many applications of geometry processing. Gu and Yau pioneered global conformal parameterization using holomorphic 1-forms, which can be decomposed to real and imaginary part and both parts are real harmonic 1-forms [65]. Holomorphic 1-form is also used to compute the affine structure of a given manifold, which is the key to construct manifold splines [66]. Guo et al. demonstrated that global surface conformal parameterization can be generalized to a more general setting of point based geometry by computing the holomorphic 1-form in a meshless manner [4]. Ni et al. used the harmonic Morse function to extract the topological structure of a surface [110]. Dong et al. [59] and Tong et al. [95] applied harmonic 1-forms in quadrilateral remeshing.

Reeb graph is a powerful tool to analyze the topology of a shape. It has wide applications in computer graphics, such as removing topological noise [147], skeleton extraction

[29], shape abstraction and understanding [27, 28]. Yoshihisa and Kunii proposed an algorithm to compute the Reeb graph in time $O(n^2)$ [128], where n is the number of edges of the mesh. Cole-McLaughlin solved the problem in $O(n \log n)$ time [44]. Recently, Pascucci et al. presented a robust online algorithm to compute Reeb graph for extremely large dataset [113].

2.3.7 Recovering Articulated Skinning Model

A conventional skinning model for articulated character comprises at least a shape surface, an articulated skeleton, a binding (attaching) method that maps the surface vertices to the skeleton. Typically in SSD, this mapping generates a set of weights for each vertex, which indicates which skeletal joints will influence this vertex and by how much. To recover a skinning model, the above information needs to be evaluated.

Lu et al. [99] compute joint weights using mesh decomposition. In this method a segmentation is first performed based on associated skeletal joints, and then overlapping area is generated from these segmentations by estimating geodesic distances from surface vertices to each boundary curve. Finally a cubic function is designed to compute joint weights for vertices in overlapping areas. Baran and Popovic used an analogy to heat equilibrium to find joint weights [22]. James and Twigg [80] built a linear system for joint weights from given examples, and the system is solved by using non-negative least squares to fit the joint weights and enforce the non-negativity properties of them. Wang and Phillips [143] proposed a multi-weight scheme in which each entry of affine transformation matrix was associated to one weight value. This approach could improve the skinning quality given sufficient examples but may incur over-fitting effects.

Given a geometric shape (mesh, parametric patch, etc), skinning parameters usually consist of a skeleton structure represented by a tree with joint nodes; joint-vertex weights defining how a surface vertex is influenced by associated joint nodes; pose values denoting

CHAPTER 2. LITERATURE REVIEW

where the skeleton should be corresponded to extra example geometries. Pose estimation referred in the context of example-based approaches usually means skeletal parameters (joint rotation, translation and scaling etc.) corresponding to given example meshes. This can be generalized as an inverse kinematic problem that is widely explored in motion capture and robot applications. Grochow et al. described a system which set up a probability distribution function (PDF) model with parameters learned from given pose data [64]. Users can provide new constraints to this model to create new pose.

Two distinctive problems occur in skinning the range scan data. Firstly, example data from scanning have varied sampling and therefore are hard to correspond. Secondly, due to occlusion or other technical difficulties, holes would occur frequently in scanned mesh. Allen et al. address the above problems by generating consistent animation for articulated human body [9]. In their work, a set of markers are attached to the scanned subject and the marker positions of each scanning example are identified automatically. But labeling markers is done manually, which builds the correspondence between markers throughout the examples. Further, a skeleton with joints is drawn manually. With a set of constraints for joint parameters together with marker positions, an optimization scheme is used to find the pose parameters for each example scanning. To overcome the problems of varied sampling and holes, a template surface is constructed based on the underlying skeleton. Displacements can be generated from the template and example scanings. Given these displacements and corresponded pose parameters, runtime shape blending is achieved by employing k -nearest neighbors interpolation. They extend their works by fitting high-resolution template meshes to detailed human body scan data [10]. This fitting is achieved by formulating an optimization problem with respect to an affine transformation at each template vertex. Angelov et al present a method to recover the articulated body from range scanned data [15]. They used unsupervised non-rigid technique to register all mesh data and then perform segmentation by identifying the rigid part with spatial contiguity.

2.4 Animation

Once a character has a specific representation and is paired with controllers using a specific skinning method, it is ready for animation. Animating a character is the process of setting the values of the rigging controls over time in order to create the illusion of movement [134]. A more traditional method is key framing, in which animators set the values of controllers at key moments in time. A computer program is responsible to generate interpolated values for in-between frames. Where to set key frames and which interpolation method should be adopted are more related to artistic skills. As such key framing provides artists with full creative control over the resulting animation. However, to produce faithful and vivid animation, a great deal of efforts and time are needed by skilled animators. Motion Capture (a.k.a MoCap) takes the movement of real actors to produce realistic animation. In physical-based animation, a character's movement is governed by physical laws so that the generated animation is physically accurate. Both methods (MoCap and physical-based) intend to ease the generation of animations but are difficult to be manipulated by animators especially when a particular exaggerated effect is desired. Much of the research in these areas focuses on regaining the lost control without sacrificing the benefits.

2.4.1 Key Framing

Key framing is an extension of the traditional 2D key drawing. In this process, animators draw sequences of animation on sheets of paper perforated to fit the peg bars on the light box. A light box, also called the trace box, holds a fluorescent light fixture that shines up through a rotatable disk of frosted glass or milky-white plastic. This allows the animator to see through the sheets and easily trace images on a sheet of paper below the one he is working on. The peg bar on the disk helps to register the sheets with corresponding holes [39]. A keyframe animator will draw the keyframes in a scene, using the character

Figure 2.9: Bouncing Ball

layouts as a guide . They draw enough frames to get across the major points of the action. Once the key frames have been established, the ones in between are drawn in a process called as “inbetweening” [81]. Automated generation of in-betweens using a computer program is a challenging problem [40]. An analogous procedure is followed in 3D animation. The values for the control handlers are set at key frames in time and the values in between are computed by the animation software. Lasseter describes the basic principles of traditional 2D hand drawn animation and their application to 3D computer animation [91]. Timing is a critical technique that conveys weight, size, speed, resistance, and force, permits anticipation, follow through, overlap, and reaction, and even suggests mood [145]. Figure 2.9 presents a path and deforming shape for a bouncing ball, which is a typical starting point for studying key framing techniques.

2.4.2 Motion Capture

Motion capture (also known as motion tracking, mocap or performance capture), refers to the technique of recording the actions of human actors, and using that information to animate digital character models in 3D animation. The information recorded from real actors are normally spatial positions of the reflective markers that are attached to actors.

Figure 2.10: Motion Capture (Image Courtesy of [87])

These raw data are further converted or mapped to kinematic parameters (trajectories and joint angles). The mapping is resolved by solving a constrained fitting problem. The generated animation is faithful since the nuances and style from the performer can be exactly captured (See Figure 2.10). Active research in this area focuses on mapping captured data to varied character surfaces, editing motion paths to generate new animation and combining separate motion sections to generate a longer and complex actions.

In this dissertation, the problem of finding reasonable kinematic (pose) configurations given a skinned mesh and its associated example meshes is addressed. This problem is similar to mapping the captured raw data (cartesian coordinates of markers) to kinematic parameters, but also presents obvious difference because example meshes provide much more information than marker positions. Silaghi et al. suggest a two-phase fitting to obtain articulated skeleton from optical motion data [130]. In this method, skeleton is first recovered locally by estimating joint positions, and the generated skeleton is then augmented by adjusting all parameters of the skeleton simultaneously. O'Brien et al. recover the joint parameters by performing a linear least squares fit [111]. Zordan et al.

CHAPTER 2. LITERATURE REVIEW

develop a physically based model to find the whole body posture including joint angles and root body estimates [156].

Chapter 3

Skeletonization from Example Meshes

In this chapter, building a skeleton for a character from a set of example meshes in different poses is demonstrated. The key idea of the proposed approach is to first compute harmonic 1-forms of the reference and example poses. Due to the small changes of the metric, the isocurves of harmonic 1-forms across various poses are highly consistent. Then the isocurve based representations are used to extract the skeleton-like Reeb graphs of the harmonic functions. Next, by examining the changes of the mean curvatures, the initial locations of joints are identified. Finally the joint locations are refined by solving a constrained optimization problem.

3.1 Differential Geometry

This section provides several concepts in differential geometry which serves as the theoretic basis of our skeletonization applications. Instead of thorough and strict descriptions, the related discussions are rather intuitive and confined to our familiar two-manifolds in Euclidean $\mathbb{R}^3$ space. For more comprehensive and generalized concepts and analysis, readers are suggested to refer [35, 133].

3.1.1 Metrics and Isometric Mapping

A key target of differential geometry is to discover local or global invariants, which are a set of measures independent of coordinates or geometries. Informally, a metric can be defined as a measure that describes the “distance” between neighboring points of a given space [103]. For instance, given two points $P1(x1, y1, z1)$ and $P2(x2, y2, z2)$ in the Euclidean space $\mathbb{R}^3$, the measure

$$d(P1, P2) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2}$$

gives the distance metric between $P1$ and $P2$ in $\mathbb{R}^3$. A widely used metric is the arc length (geodesic distance) between two points along a given surface. Isometric mapping is a mapping that preserves the metric measures, which therefore has another name “length-preserving” mapping [72]. In the discrete setting, if the given mesh is finer, more information would be available to measure the values of length, area or volume etc. providing a better approximation of the continuous metric.

3.1.2 Curvatures

Consider a plane curve $\tau(s)$, where s is the parameter (i.e. arc length). The curvature of the curve is defined as $\kappa(s) = \|\tau''(s)\|$. Given a point on the curve, the curvature of this point measures how the curve deviates from the initial direction (the tangent). If the curve keeps close to the same direction, the vector changes very little and the curvature is small while if the curve undergoes a tight turn, the curvature is large. The curvature of a straight line is zero and the curvature of a circle is a constant. For a smooth surface in 3D space, intuitively, curvatures describe how the surface is curved within a local area. Given a point P on the surface, there are many curves passing through this point along this surface. Each curve is associated with a curvature. Among all associated curvatures, there is one minimum κ_1 and one maximum κ_2 , which are defined as principal curvatures

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

of P . The Gaussian curvature is defined as $G = \kappa_1 * \kappa_2$ and the mean curvature is defined as $H = \frac{\kappa_1 + \kappa_2}{2}$. Gaussian curvature remains invariant when the surface undergoes arbitrary bending. A bending is defined as a deformation for which the arc length is left invariant [72]. In other words, if a deformation occurs only by bending the surface without stretching it, this deformation can be defined as an isometric deformation and the Gaussian curvature along the surface will remain unchanged. For instance a cylinder or a cone has $G = 0$, since both can be obtained by bending a flat plane. While from the definition of the mean curvature, it is not an intrinsic notion and will change if it undergoes an isometric deformation i.e. a bending.

3.1.3 Harmonic Scalar function

Harmonic scalar function is defined as f such that

$$\Delta f = \nabla^2 f = 0 \quad (\text{Eq. 3.1})$$

where Δ is the Laplace operator, subject to the Dirichlet boundary conditions that vertices in the set C of constrained vertices take on the prescribed values [59]:

$$f_i = c_i \forall i \in C$$

In discrete setting, the commonly used Laplacian operator on a polygon mesh is

$$\Delta f_i = \sum_{j \in N_i} \omega_{ij} (f_j - f_i) = 0 \quad (\text{Eq. 3.2})$$

where N_i is the set of vertices adjacent to vertex i and ω_{ij} is a scalar weight assigned to the edge (i, j) .

From Eq. 3.2, it can be derived as

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

$$\sum_{j \in N_i} \omega_{ij} f_j + \left(\sum_{j \in N_i} \omega_{ij} \right) f_i = 0$$

If the weights satisfy the condition $\sum_{j \in N_i} \omega_{ij} = 1$, the above equation can be written as

$$\sum_{j \in N_i} \omega_{ij} f_j + f_i = 0$$

such that

$$f_i = - \sum_{j \in N_i} \omega_{ij} f_j$$

It is now clear that the harmonic scalar function f_i is a weighted combination of the function values at its neighboring vertices. Thus f_i is guaranteed that it is not a local extremum provided that (1) $\sum_{j \in N_i} \omega_{ij} = 1$ and (2) $\omega_{ij} > 0$ [110]. Given these two conditions, we are free to choose the weights. A natural choice is $\omega_{ij} = \frac{1}{N_{ij}}$. Another popular choice is the discrete harmonic weight

$$\omega_{ij} = -\frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) \quad (\text{Eq. 3.3})$$

where α_i and β_{ij} are the angles opposite the edge. These choices of weights guarantee for f that the energy $\nabla f = 0$ [114].

If we present the function f by a column vector of its values at all vertices $\mathbf{f} = [\mathbf{f}_1 \mathbf{f}_2 \dots \mathbf{f}_n]^T$, the Laplacian operator on the entire mesh is given as a matrix function

$$\Delta \mathbf{f} = -\mathbf{L} \mathbf{f} \quad (\text{Eq. 3.4})$$

where the matrix $\mathbf{L}$ has entries:

$$L_{ij} = \begin{cases} 1, & \text{if } i = j \\ -\omega_{ij} & \text{if } j \in N_i \\ 0, & \text{otherwise.} \end{cases}$$

Then the scalar function can be obtained by solving the linear system

$$\mathbf{A}\mathbf{f} = \mathbf{b} \quad (\text{Eq. 3.5})$$

where

$$A_{ij} = \begin{cases} \delta_{ij}, & \text{if } i \in C \\ L_{ij} & \text{otherwise.} \end{cases}$$

$$b_i = \begin{cases} c_i, & \text{if } i \in C \\ 0 & \text{otherwise.} \end{cases}$$

where δ_{ij} is the Kronecker delta function [59]. This is a sparse system and there are several sparse linear solvers available for this system [110].

3.1.4 Harmonic One-Forms

By choosing appropriate constraints (source vertices), a harmonic scalar function built as described in the previous subsection provides a natural measure of geodesic distance on the mesh. In other words, if the mesh undergoes isometric deformation such as bending, the corresponding function value of a deformed vertex will not change compared with the value of non-deformed one. Another important property of the harmonicity of a harmonic function f is that it has global extreme values only at constrained vertices and flows smoothly everywhere else. However if the mesh is topologically complex, there are critical points existing on the mesh where the smooth property does not hold. Therefore we seek Harmonic One-forms, which is loosely defined as the gradient field of the f . According to the theory of harmonic function, the gradient of a harmonic function is still harmonic. In a discrete setting, when each vertex of a mesh is associated with a harmonic function value f_i , the harmonic one-form of f is defined on each edge $[i, j]$ and denoted as $\nabla f_{i,j} = f_i - f_j$.

Given the concepts and properties of Harmonic function and Harmonic one-forms, a set of advantages of Harmonic one-forms are generalized as follows:

Figure 3.1: Harmonic 1-form on genus two surface. The red line in (a) is the boundary.

- Harmonic 1-form is determined by the metric and is invariant under isometric transformation.
- Harmonic 1-form is independent of surface representation. The surface with different resolution and triangulation has similar harmonic 1-form.
- Harmonic 1-form is computationally efficient and robust as we only need to solve a linear system.
- Symmetric harmonic 1-forms can be easily constructed on symmetric shapes.

3.2 Algorithm Overview

The proposed method takes several poses of a given model as input. These models are allowed to have very different triangulations and resolutions. The algorithm runs in five successive stages which are illustrated in Figure 3.2.

- (i) The user specifies a source point on the reference pose. A registration algorithm is used to map this source point to the other poses.
- (ii) Harmonic 1-forms are computed using the user-specified source point as constraints.

Figure 3.2: Algorithm Pipeline for extracting the skeleton using harmonic one forms. (a) Input examples with user specified source point. (b) Harmonic one form. (c) Reeb graph. (d) Joints are identified by evaluating the mean curvature of isolines. (e) Joints are optimized

- (iii) A skeleton-like Reeb graph of the harmonic 1-form is extracted.
- (iv) Isolines of the harmonic 1-form are extracted with their associated mean curvature. By comparing the mean curvature for all poses, joints can be identified, which are closely related to the isolines whose mean curvatures are changed significantly.
- (v) Joint locations and bone lengths are optimized by minimizing the distance between the skeletons and the Reeb graphs.

3.3 Reeb Graph Extraction

Reeb graph is a powerful tool when dealing with topological skeleton. For a real-valued smooth function $f : M \mapsto \mathbb{R}$ on M , the points whose derivatives of f vanishes are called critical points of f . The Reeb graph of f is a graph whose nodes corresponds to these critical points and encodes the connectivity between them.

- **Building the graph.** Reeb graph can be constructed by contracting the connected components of the isocurves (level sets) of f to a point. Let n be the user-specified number of samples of the isovalues (200 in our experiments). We first normalize the function value of f to the unit interval $[0, 1]$ and then uniformly sample isovalues, $f_i = \frac{i}{n}, i = 1, \dots, n$. Next we scan every critical point p of $f, \nabla f(p) = 0$. If $f(p)$ is not sampled, we insert $f(p)$ into the isovalue set. Denote $c_k(f_i)$ as the isocurve with isovalue f_i on k -th pose. $k = 0$ refers to the reference pose. (Note that each isovalue f_i may have more than one isocurve.) For each isocurve, we compute its center as the representative. Then we use the sweep algorithm to connect these representatives and form the Reeb graph [44].
- **Topological filter.** Once the Reeb graphs of reference and example poses are constructed, we first identify the key nodes (valence $\neq 2$). Due to numerical error and metric changes, multiple key nodes may occur in locations where there should only be a single key node. Thus, we cluster key nodes such that the Reeb graphs of the reference and example poses have the same topological structure, i.e. the same number of key nodes.
- **Eliminating the ambiguity.** Note that due to the symmetry of many articulated models, such as human and animals, two or more isocurves with similar geometry may have the same isovalue, e.g., the Armadillo's elbows. Thus, we then need to distinguish these isocurves with the same isovalues. In this work, the advantages of the articulated models which usually have some key points in the Reeb graph are explored to eliminate the ambiguity. Since the user picks the source point on the head, the first key point connecting to the source is the neck. So we can identify the spine easily. To further classify a branch into left and right arms (legs), we compute the sum of the signed distance from the points on the branch to the

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

symmetry plane passing the spine. If the sum is positive, the branch is identified as a left arm or leg depending on the distance to the source point. Otherwise, the branch is identified as the right arm or leg. Note that the sign determining left and right is not important. The user can freely change the sign to swap left arm/leg and right arm/leg.

- **Finding the correspondence among poses.** Next we want to find the one-to-one correspondence between reference pose and example pose. Note that we do not need to find such a mapping between the vertices of two poses which are usually in different triangulation. Instead we will find the mapping for isocurves. Isocurves are the level sets of the harmonic function f , which are independent of the resolution and embedding. Therefore, even though the reference and example poses have very different resolutions, their isocurves are highly consistent if the metrics are similar.

3.4 Evaluations of Joint Positions

3.4.1 Identifying the Joints

As explained above, many real-world deformations of articulated models are isometry or near isometry. It is well known that Gaussian curvature is completely determined by the metric, therefore, the Gaussian curvature of the reference and example poses remain almost unchanged. Mean curvature, however, is related to the embedding, and could be used to identify the joints. We observe that compared to the reference poses, the embedding of the joint-related skins could vary significantly in example poses. Thus, we use the average mean curvature

$$D(i) = \max\left\{\left|\frac{\int_{c_k^i} k_H dl}{\int_{c_k^i} dl} - \frac{\int_{c_0^i} k_H dl}{\int_{c_0^i} dl}\right| \mid k=1,2,\dots\right\} \quad (\text{Eq. 3.6})$$

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

to measure the difference of the embedding of isocurve c^i between the reference pose and example poses.

3.4.2 Optimizing the Joint Locations

The above subsection identifies the number of joints in the reference and example poses. However, the joints just locate on the Reeb graph. Although encoding the topological structures nicely, the skeleton-like Reeb graphs are not the anatomical skeletons. Therefore, we need to estimate the joint locations using examples.

Assume m joints are identified using the above mentioned. Denote J_k^j the j -th joint, and B_k^l the length of the l -th bone in k -th pose. We enforce that the bone lengths do not change in the deformations. For an isocurve c_k^i in the k -th pose, denote $d(c_k^i, J_k^1, \dots, J_k^m)$ the sum of distances between the point on the isocurve and the skeleton defined by the joints $J_{k,j=1}^m$. Then, we also require that such distances are also invariant in all poses too. These constraints lead to the following constrained optimization problem:

$$\operatorname{argmin} \sum_{k \geq 1} \sum_i |d(c_k^i, J_k^1, \dots, J_k^m) - d(c_0^i, J_0^1, \dots, J_0^m)|^2 \quad (\text{Eq. 3.7})$$

The objective function ensures that the relationship between the skin and the bone is as rigid as possible. The constraints ensure that the bone length remains unchanged in all poses.

3.5 Comparison with Alternative Methods

Automatically recovering the skeleton from 3D shapes has been explored extensively. Most of the existing works focus on extracting skeletons from a static pose of a model. Mortara and Patané defined an affine-invariant skeletal representation for 3D shape matching [108]. Katz and Tal used fuzzy clustering method to decompose a shape and then extract the skeleton [82]. Lien et al. proposed an efficient and robust approach

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

that simultaneously generates hierarchical shape decomposition and a corresponding set of multi-resolution skeletons [98]. Theobalt et al. took a sequences of volume data to estimate skeleton through volume decomposition of motion data [49]. Oda et al. presented a framework of extracting skeleton interactively using geodesic distances. Ma et al. constructed a RBF level set for 3D model and found the maximum of gradient descent for each point and then formed the skeleton [112]. Dellas et al. presented an automatic method, which relies on an a-priori knowledge of the human anatomy [53], to extract the animation control skeleton of virtual humans. Baran and Popovic presented an automatic method to embed a generic skeleton to a wide range of articulated 3D characters [22].

Since it is usually difficult to estimate the dynamic behavior using the static pose alone, example-based techniques gain popularity in computer graphics in recent years [115, 122, 135]. In [80] James and Twigg demonstrated that the conventional skinning techniques can be extended to automatically skin deformable mesh animations. Instead of specifying the hierarchical kinematic skeleton, they estimated proxy bone transformations and vertex weights for deformable shape sequences.

Recently, Schaefer and Yuksel proposed a novel method to extract hierarchical, rigid skeletons from example poses[122]. They present a similar motivation but an entirely different methodology. They first defined “Rigid Error Functions” to find the best rigid transformation and used these error functions to estimate the transformations of bones in the example poses. Then they skinned the mesh by solving for vertex weights using a constrained optimization and bone influence maps. Finally, they determined the connectivity of the skeleton and the joint locations. This approach is capable to estimate the complete set of parameters for skeletal animation including bone transformation, skeletal hierarchy, joint location and vertex weights. However, their method requires that the reference and example pose have the same triangulation, in other words, this method

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

is highly dependent on the geometries of the given meshes and does not explore the invariants of the shapes.

Aujay et al. presented the harmonic skeleton in which anatomical information is used to enhance the skeletons [6]. Reeb graph is also used to build an initial structure for skeletons, but a set of heuristics has to be explored to assist the refinement of the harmonic skeleton, i.e. source points are manually picked to ensure that the spine is near the back instead center of the body. Compared to the above existing works, the advantages of the proposed method include:

- There is no restriction on the connectivity of the input meshes, i.e., the example poses could have very different triangulations and resolutions.
- This method is automatic except that the user needs to choose a source point on the reference pose.
- The number and locations of the joints and bones are totally determined by the dynamic geometry of the reference and example poses.
- Since symmetric Dirichlet boundary values are set for symmetric shapes, the extracted skeletons also reflect the geometric symmetry of the input model.

3.6 Results and Discussions

Harmonic skeleton is proposed in [6], in which a single mesh is considered with several anatomical heuristics assumed. Example based skeletonization is also explored such as [122], which is purely dependent on the geometric imbedding. In this dissertation, skeletonization from examples by exploring harmonic one-forms is first proposed and successfully demonstrated in [68, 69]. The methodology is based on the observation that many deformations in real world applications are isometric or near isometric. By taking

CHAPTER 3. SKELETONIZATION FROM EXAMPLE MESHES

Figure 3.3: Skeletons at the reference poses

advantage of the intrinsic property of harmonic 1-form, i.e, it is determined by the metric and independent of the resolution and embedding, the method can easily find a consistent mapping between the reference and example poses. A set of skeleton-like Reeb graphs of a harmonic function is first constructed. Then by examining the changes of mean curvatures, initial location of joints are identified. The locations are refined by solving a constrained optimization problem. The result can be seen in Figure 3.3.

Chapter 4

Parameter-based Skeletal Skinning from Examples

Most works related to example based approaches assume that skeletal (pose) configurations are given as input [84, 88, 90, 96, 132, 143]. If there are only geometric example meshes, building pose configurations for each example is a challenging and also tedious work. In this chapter, we address this problem with only geometric example meshes available as input. The framework is illustrated in Figure 4.1, in which a set of example meshes together with a skeleton in rest pose will be taken as input (Figure 4.1 (a)). A rough skinning model as one instance of skeletal subspace deformation (SSD) is assumed. Positions of skeletal joints in associated example poses are found by formulating a minimization problem (Figure 4.1 (b)). These positions will be parameterized as Euler angles and scaling factors of each joint as these parameters are more semantically meaningful to animators (Figure 4.1 (c)). Besides, SSD will be thoroughly investigated at the beginning since this underlying skinning model serves as the building block for the research works in this dissertation.

4.1 Skeletal Subspace Deformation

A skeleton should be drawn and skinned to a character surface beforehand, roughly based on the anatomy of the character and kinetic rules. The pose in which the skeleton is rigged

Figure 4.1: Illustration of parameterization scheme.

is normally referred to as the rest pose. The basic relation between surfaces and skeletons is defined in the rest pose, and all motions of the character will be influenced thereafter. If SSD is adopted to define this relation, each vertex or control point of the character surface is provided with a list of joints, that will influence it, along with the weight indicating the amount of influence. When a character is animated, the position of a vertex in the animated pose is the result of weighted linear blending of its transformations by associated joint. In the following, SSD will be investigated with respect to its formulation, limitation and existing solutions for improvements.

4.1.1 Formulation

In this subsection, a description of skeletal subspace deformation (SSD) is given. (SSD) is the most widely used skinning scheme in 3D animation. During runtime animation, the movement of 3D character surface is driven by transformations of associated skeletal joints. Geometric surface will be bound to skeletons in the “rest” or “stand” pose. When a joint J is transformed, a series of transformation matrices associated with the

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

Figure 4.2: Skeletal Subspace Deformation

coordinate frame system of J can be generated. These matrices include a matrix M_l^i denoting a transformation from the world space to the local space in rest pose; a matrix M_a^i denoting animated transformation that normally refers to a combination of rotation, translation and scale; and a matrix M_w^i which is the transformation matrix from the local space to the world space in animated pose. For a vertex in the animated pose, its position in the world space relative to joint J frame should be

$$v = M_w * M_a * M_l * v^r \quad (\text{Eq. 4.1})$$

where v^r is its position in the rest pose.

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

In Figure 4.2, a simple skeleton consisting of only two joints is considered: P for parent joint and C for child joint. For each joint, a coordinate frame is specified respectively. Now how SSD performs skinning for a single vertex v^r is illustrated. The above matrices for parent joint are denoted as M_{pl} , M_{pa} and M_{pw} . For the child joint, they are M_{cl} , M_{ca} and M_{cw} . Thus, at an arbitrary animated pose (middle row of Figure. 4.2), vertex positions for v^r at the parent and child coordinate frames are v_p and v_c respectively computed from Eq. 4.1.

In SSD, each vertex is first assigned a set of joints which will exert influence to it. In Figure 4.2, v^r is influenced by joint P and C . We call the amount of this influence as joint weight ω . The typical value of joint weight is between 0.0 and 1.0, and the sum of all associated joint weights for one vertex should equal to 1.0. The core of SSD algorithm can be stated as: In any animated pose, each joint is associated with a transformation, which transforms a vertex to a position in this animated pose. A weighted sum of positions from those joints with non-zero influence on this vertex generates the final position in this animated pose. Therefore in Figure 4.2, the final position for v^r equals $v = \omega_p * v_p + \omega_c * v_c$. In general, if the number of influencing joints can be arbitrary, say, N_j , we have:

$$v^p = \sum_{i=1}^{N_j} \omega_i * M_i^p * v^r = \sum_{i=1}^{N_j} \omega_i * v_i \quad (\text{Eq. 4.2})$$

where v^p is the position at pose p , M_i^p is the accumulation of M_{il} , M_{ia} and M_{iw} for joint J_i at pose p , v_i is corresponded position transformed by each joint j , and ω_i is associated joint weight from joint i to vertex v^r . In most applications and commercial packages, the number of joints can be adjusted by users. (A common choice is 3 or 4.) Joint weights are computed based on distances from vertices and associated joints in rest pose. But users are also given some parameters to initialize joint weights such as a “fall-off”, which control how fast joint weights decrease along with the increase of distances

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

from vertices to joints. In some packages like Autodesk Maya, animators are able to fine tune the joint weights by using a tool called “weight painter”.

Due to the succinct formulation of SSD, it presents a couple of advantages either in academic research or in industrial practice. It is relatively easier to be implemented and thus is commonly supported by most graphic cards. Besides, tuning weight parameters can provide a space of flexibilities and creativities to users. However, there are a set of typical shortcomings in SSD, which is investigated in the following subsection.

4.1.2 Limitations

Figure 4.3: Limitations of SSD: left: collapse elbow; right: candy-wrapper.

SSD is notorious for its “collapse elbow” (left of Figure 4.3) and “candy-wrapper” (right of Figure. 4.3) effects. The reasons for those drawbacks from SSD have been well documented [84, 96, 143]. Taking a closer look at the Figure 4.2 in the animated pose, if the joint C is rotated further, the surface part around C will collapse to a very small part or even worse, to a single point. Analyzing Eq. 4.2, it can be noticed that multiplying transformation matrices cannot express the correct combination of separate deformations and may lead to singular results in some circumstances. Still taking Figure 4.2 as an extreme example, if all elements of the weighted sum $\sum_{i=1}^{N_j} \omega_i * M_i^p$ become

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

zero, all vertices associated to joint $i (i = 1, \dots, N_j)$ would be transformed to one single position $(0, 0, 0)$.

4.1.3 Existing Solutions

Targeting the limitations discussed above, there are many works dedicated to improve the skeletal skinning methodology. The first category is so called “parameter based”, in which the model is enhanced by adding more parameters such as the number of weights [143] or the number of joints [106]. Another category enhances the quality by introducing more examples, in which a runtime interpolation compensates deformations for those collapsed parts [96][132]. In this dissertation, these two categories will be combined to improve the underlying parameter-based model (by replacing skeletal subspace deformation with spherical blend skinng) as well as to guarantee skinning qualities by runtime interpolation(through inverse pose space deformation). In addition, how to parameterize the given geometric examples with respect to pose vectors is explored in this chapter to facilitate our proposed framework.

4.2 Optimal Joint Positions in Example Poses

Figure 4.4: Joint skin and bone skin segmentation.

4.2.1 Binary Segmentation

We observe that distances between a joint position and its adjacent surface vertices are near invariant during articulate deformation. Hence, to obtain joint positions for example

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

meshes, an optimization problem is formulated in terms of an objective function which measures the difference of distances between surface vertices and associated joints in the rest pose and example poses respectively. The formulation details are discussed in the next section. Generally speaking, this objective function is minimized with respect to the joint positions in example poses. However, not all surface vertices are involved in solving this minimization problem. Only vertices that are very close to their associated joints are counted. We call these vertices “joint skin” that should be almost equally influenced by two joints in hierarchical adjacency. Weights of these two joints are therefore close to 0.5. Other vertices exclusively influenced by a single joint, can be categorized as “bone skin” and normally have a larger weight value close to 1.0. This binary segmentation (illustrated as Figure 4.4) is quite coarse nevertheless it is sufficient for estimating optimal joint positions in example poses. In our experiments, we use a threshold parameter 0.8. Any vertex with a weight value bigger than it will be assigned to “bone skin”, while other vertices will automatically go to “joint skin”. Joint weights used for segmentation can be roughly approximated as a function of vertex-joint function. We will describe how to refine joint weights from example meshes in section 4.4.2.

Figure 4.5: Undesirable weight distribution will not influence the evaluation of optimal joint positions by tuning threshold parameter. Left: non-connected weight distribution; Right: joint skin set filtered by weight threshold.

Note that “joint skin” set has to be a simple connected region, otherwise joint position may not be correctly evaluated. To guarantee this simple connection, people have to

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

solve a surface segmentation problem [95]. In this dissertation, we start with a rough weight assignment as shown in Figure 4.5 by introducing a lower threshold for “joint skin”. Although the distribution of weighted vertices shown in the left of the figure 4.5 is undesirable, those unconnected vertices always have a very small weight values (artists will employ a so called “weight painting” to correct this problem). After introducing a lower threshold (0.2 in our experiments), we can obtain the joint skin set shown in the right of Figure 4.5. To summarize, the upper threshold to categorize “bone skin” is 0.8, and the lower threshold to mark “joint skin” is 0.2. Figure 4.6 shows the distribution of “joint skin” and “bone skin”. Joints of body part have more complicated connectivity structure and are placed closer than those of limb part. Therefore, the joint skins of body joints may overlap each other.

Figure 4.6: Binary segmentation for horse and camel. Red: joint skin; Green: bone skin.

4.2.2 Conjugate Gradient Method

In this subsection, a novel method is proposed to evaluate skeletal joint positions in example poses by formulating a minimization problem. An object function is built with respect to joint positions. This function measures the difference of the squared distance from a vertex to its associated joint in the example pose against the distance in the rest pose.

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

Figure 4.7: A non-centered joint is circled, which is resulted by a local minimum in solving Eq. 4.3.

For a joint J , denote $\mathbf{p}$ as its position in world space. The objective function for J is formulated as:

$$f(\mathbf{p}) = \sum_{i=1}^M \|\mathbf{d}\mathbf{0}_i^2 - \mathbf{d}_i^2(\mathbf{p})\|^2 \quad (\text{Eq. 4.3})$$

where $\mathbf{d}\mathbf{0}_i$ is a constant scalar value denoting the distance from vertex $\mathbf{v}_i$ to J in the rest pose. $\mathbf{d}_i(\mathbf{p})$ is the distance from the vertex i to the joint J in one example pose: $d_i(\mathbf{p}) = \|\mathbf{v}_i^e - \mathbf{p}\|$. M are the “joint skins” identified in the previous section. Actually $f(\mathbf{p})$ is a mapping function:

$$f(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R} \quad (\text{Eq. 4.4})$$

The derivative of $f(\mathbf{p})$ is discretized as the following:

$$\frac{\partial f}{\partial p_x} = 4 * (d0_i^2 - d_i^2) * (v_x^e - p_x)$$

$$\frac{\partial f}{\partial p_y} = 4 * (d0_i^2 - d_i^2) * (v_y^e - p_y)$$

$$\frac{\partial f}{\partial p_z} = 4 * (d0_i^2 - d_i^2) * (v_z^e - p_z)$$

Given a vertex set as “joint skin”, an object function and its derivative, we use conjugate gradient methods [120] to solve Eq. 4.3 for each joint. Our investigation shows that joint positions are well centered at example meshes (shown as Figure 4.11). In some rare circumstances, minimization may end up with local minimum, and resulting joint is not correctly placed (shown as Figure 4.7). Therefore we need to choose an alternative initial value for minimization. In terms of the implementation, the solving starts from the root node, and for the joint whose minimization residual is not near zero, optimization will be re-performed again by setting the initial value as the optimal position of its parent joint. Since finding optimal value for root is trivial, there is always an optimal parent available for solving child joint.

4.3 Semantic Parameters for Example Pose Space

4.3.1 Euler Angles

Euler angles can be represented as three rotation coordinates in sequence [58]. Consider the triple rotations in which the first rotation is an angle ψ about the x-axis, the second rotation is an angle θ about the y-axis, and the third rotation is an angle ϕ about the z-axis. The function that maps an Euler angle vector to its corresponding rotation matrix, $R_{xyz} : \mathbb{R}^3 \longrightarrow SO(3)$, is

$$R_{xyz}(\psi, \theta, \phi) := R_x(\psi)R_y(\theta)R_z(\phi) \quad (\text{Eq. 4.5})$$

The order of the sequence is very critical because rotations do not commute [129]. Some disadvantages on Euler angles are identified by many researchers. First, “converting between rotation matrices and angle coordinates is difficult and expensive, involving arbitrary assumptions” [129]. Second, this parametrization bears a singularity known as “gimbal lock” which is demonstrated in Section 6.1. Nevertheless, Euler angles are favored by animators since they are flexible to manipulate and relatively easy to be understood. A good animation system should provide convenient interface to users and hide the abstract representation and implementation from them. That is why a parametrization of Euler angles is desired while another one with better behavior can be used to perform the underlying skinning method described in Chapter 6.

4.3.2 From Joint Positions to Euler Angles

Figure 4.8: Estimating pose space parameters: JP: parent joint; JCr: child joint in rest pose; JCe: child joint in example pose; q: quaternion rotation; s: scaling value.

In the previous section, only the positions of skeleton joints in example poses are obtained. To employ the skeletons in real productions, it is desirable to compute the joint transformations as parameters of example poses. Although an example mesh can be expressed as a reference mesh with a group of affine transformations [80], animators prefer

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

more meaningful parameters such as joint rotations, translations or scaling values for flexible manipulation. In most example-based approaches, these parameters are manually built or represented as a vector of 4×4 transformation matrices which are not semantically meaningful. In this section, Euler angles of skeletal joints, as one type of semantic pose parameters, in each example are recovered. We first align the source joint of reference mesh to one example mesh. And then starting from the root joint, rotation angles and scaling values for each joint will be recursively computed in a width first search manner. Given joint positions of a father-child pair in reference pose and one example pose, two bones in the respective reference and example pose form a quaternion that rotates the reference bone into the example bone.

In Figure 4.8, for a specific bone with a parent joint and child joint, JCr is the position of the child joint in rest pose, and JCe is its position in one example pose. The positions of parent joint in the rest pose and example pose are aligned to JP . Two vectors are thus generated as $\mathbf{v}_{JP-JCr}$ and $\mathbf{v}_{JP-JCe}$. Based on the Euler's Theorem that any two independent orthonormal coordinate frames can be related by a sequence of rotations [89], rotating $\mathbf{v}_{JP-JCr}$ to $\mathbf{v}_{JP-JCe}$ brings out a quaternion. Quaternion is widely used in animations, which is investigated in details in Chapter 6. Given a quaternion $\mathbf{q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4]$, the Euler angles are:

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{pmatrix}$$

These Euler angles can be used as pose parameters for key framing, can serve as interpolation values for pose space deformation, and can be tuned by animators for other special effects.

Figure 4.9: Illustration on bone scaling and translating

4.3.3 Scaling Joints

In the previous section, Euler angles are recovered by rotating a bone from the rest pose to one example pose. But there is a disparity in the lengths of two vectors as shown in Figure 4.8. This disparity will serve as a scaling value as another type of pose parameters. Instead of computing pure rotations [94] or rotations and translations [104], example pose space is parameterized as rotations and scaling values based on the observation that scaling joints can capture more accurate deformations than translations. This strategy can also be justified by the fact that articulate deformation is more likely resulted from muscle flexion while the bone length should be fixed. In Figure 4.9, the joint which is manipulated is circled in orange color. In the left is a human arm in rest pose. In the middle, the circled joint is translated along x-axis by 0.6, while it is scaled still along x-axis by 1.6 in the right picture. It is obvious that the translation brings more artifacts than the scaling, and further, scaling will not change the position relative to the parent joint.

Given the recovered Euler angles ϕ, θ, ψ and scaling values s for a joint, the transformation matrix which is used to evaluate M_a in Eq. 4.1 is given by:

$$M_a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix}$$

However, those recovered parameters are not simply used to evaluate the SSD model,

but also for fitting the joint weights to given examples and further to serve as interpolation variables for PSD, which is discussed in Chapter 5.

4.4 Fitting Joint Weights

4.4.1 Least Squares Problems

In computer graphics, Least Squares (LS) can be used to solve a wide range of problems, which are usually viewed as finding the best set of parameters for a model given some data [5]. For instance, in inverse kinematic problem, joint angles can be determined by minimizing the disparities between specified effector positions and evaluated effector positions. Generally speaking, in 1-dimension case, given two variables a and b , x is used to describe the relationship between them:

$$b = xa \quad (\text{Eq. 4.6})$$

Since this relationship (the linear model) may not be exact for all data pairs, evaluated data can be written as:

$$\hat{b} = xa \quad (\text{Eq. 4.7})$$

Then the LS problem can be generalized as given a set of data (a_i, b_i) ($i = 1, \dots, n$), find a parameter x such that

$$e = \sum_{i=1}^n (b_i - \hat{b}_i)^2 = \sum_{i=1}^n (b_i - xa_i)^2 \quad (\text{Eq. 4.8})$$

is minimized. In this case, evaluating joint weights from examples, SSD can be used as the underlying model to describe the relationship between the weights and the example meshes. Other parameters (i.e. skeletal structure, pose parameters for examples, mapping between joints and influenced joints, etc) for evaluating deformed meshes through SSD are recovered already. So we have SSD as the model:

$$\hat{\mathbf{e}} = \text{SSD}(\omega) \quad (\text{Eq. 4.9})$$

then, LS can be used to obtain ω by minimizing $|\mathbf{e} - \hat{\mathbf{e}}|^2$. Because the joint weights should be positive, then Non-Negative Least Squares (NNLS) is applied as discussed in the next subsection.

4.4.2 Fitting joint weights through Non-Negative Least Squares

Normally, joint weights based on SSD are roughly initialized through either “closest distance” or “closest in hierarchy”, and then carefully tweaked by artists to obtain better deformations. This task known as “weight painting” requires much animating experiences and tedious efforts as well. Careless weighting will result in ambiguous attachment that some surface vertices are assigned to wrong joints (illustrated as Figure 4.5). Deriving joint weights automatically from examples by formulating a data fitting problem is adopted in recent research works.

Figure 4.10: Joint Weight Fitting: (a) A tube is skinned through SSD; (b) One of five examples from manually re-sculpting; (c) fitted tube corresponding to (b).

The method of least squares, also known as regression analysis, is used to model numerical data obtained from observations by adjusting the parameters of a model so as to get an optimal fit of the data. The best fit is that instance of the model for which the sum of squared residuals has its least value, a residual being the difference between an observed value and the value given by the model. The model that is going to be fitted is Eq. 4.2, in which joint weights ω_i are refined according to example meshes in the manner of least squares.

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

Given skinning formula Eq. 4.2 and coordinates in example pose space from the previous sections, for a single vertex v at one specific example pose, we can obtain transformed vertices induced by influencing joints $v_k^p = M_k^p v^r$ where $k = 1, \dots, J$ indicating indices of influencing joints. We stack these transformed vertices from all joints and all examples:

$$\begin{pmatrix} v_1^1 & v_2^1 & \dots & v_J^1 \\ v_1^2 & \dots & \dots & v_J^2 \\ \dots & \dots & v_p^k & \dots \\ v_1^N & \dots & \dots & v_J^N \end{pmatrix} \begin{pmatrix} \omega_1 \\ \dots \\ \omega_k \\ \dots \\ \omega_J \end{pmatrix} = \begin{pmatrix} v^1 \\ \dots \\ v^i \\ \dots \\ v^N \end{pmatrix}$$

where N is the number of examples and v^i is the example vertex corresponded to v . v_p^k means the vertex position for v , which is transformed by k th joint to example pose p . ω_k means influencing weight from joint k . This is a constrained linear system in which $\omega \geq 0$ and $\sum_{k=1}^J \omega_k = 1$. To meet these constraints, the above linear system is revised:

$$\begin{pmatrix} v_1^1 & v_2^1 & \dots & v_J^1 \\ v_1^2 & \dots & \dots & v_J^2 \\ \dots & \dots & v_p^k & \dots \\ v_1^N & \dots & \dots & v_J^N \\ 1 & \dots & \dots & 1 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \dots \\ \omega_k \\ \dots \\ \omega_J \end{pmatrix} = \begin{pmatrix} v^1 \\ \dots \\ v^i \\ \dots \\ v^N \\ 1 \end{pmatrix}$$

We solve this system using nonnegative least squares (NNLS) [92] to obtain joint weights from examples [80]. Solving this linear system using NNLS can efficiently avoid over fitting problem [143]. The fitting result is demonstrated in (c) of Fig. 4.10, in which a cylinder is fitted to five examples. Compared with the conventional SSD skinning result (a), fitting joint weights from examples present obvious advantages.

4.5 Results and Discussions

Skeletal subspace deformation (SSD) has been proposed and applied in real productions for many years. As for example-based approaches (i.e. PSD), which intend to improve SSD in many ways, have been seen actively practised from 2000. However, a gap that

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

associates each example mesh to its corresponding pose always exists and almost all example-based approaches assume that the associated poses are already available, which actually requires lots of tedious manual works. This chapter is to fill this gap by proposing a pose parameterization scheme, based on which PSD can be applied upon pure geometries by building corresponding pose information. The work described in this chapter is published in [68, 151]. Optimized joint positions for example poses are demonstrated in Figure 4.11. Based on the recovered pose parameters, the underlying model skeletal subspace deformation (SSD) is refined to fit the given examples. New poses can be generated as shown in Figure 4.12. The contributions made in this chapter include, first, to obtain the positions of skeletal joints in example poses, an optimization scheme is proposed by identifying “joint skin”, distances of which to the associated joints are supposed to be near invariant during non-rigid deformation. Second, a parameterization scheme of example poses is proposed for each skeletal joint. These parameters provide a flexible and semantically meaningful interface for animators. Third, the recovered pose parameters are used to fit joint weights by solving a non-negative least squares problem. However, the mesh deformed by SSD in the example poses shows some artifacts compared with the given example meshes. Pose space deformation is then applied to remove these artifacts which is described in Chapter 5.

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

Figure 4.11: Estimating world positions of joints in example poses.

CHAPTER 4. PARAMETER-BASED SKELETAL SKINNING FROM EXAMPLES

Figure 4.12: New poses.

Chapter 5

An Optimization Scheme for Example Based Approaches

In this chapter, a layered framework that incorporates example-based skinning algorithms such as Pose Space Deformation into an existing character animation system is proposed. The most challenging aspect in designing an example-based skinning in an existing system is that it is generally believed that the interpolation of the examples is best performed before doing other skinning deformations (although there has been no analysis as to why this is the case), whereas the examples are specified by the user after the other deformations are performed. It is therefore necessary to *invert* the operation of these skinning and deformation operators. Existing systems typically allow layering of both basic skinning methods such as Skeleton Subspace Deformation (SSD) and other deformations such as lattices, etc., and commercial systems may further allow additional proprietary deformation algorithms as part of the character skinning. Unfortunately, understanding and accessing their various parameters can be laborious at best, and the algorithms in the case of commercial packages are usually not accessible to users. The contributions of this Chapter are 1) a detailed analysis showing how inverting the skinning operations leads to better example interpolation, and 2) a demonstration that the black-box inverse can be accomplished in practise using Powell optimization, resulting in an improved example-based skinning capability for existing large animation systems.

5.1 Scattered Data Interpolation

Scattered data interpolation problems arise in many different ways such as gravitational field of earth, color film processing, implicitly defined surfaces, oil exploration etc. [8]. The problem can be formulated as that given data points $(x_i, y_i) \in \Omega \times \mathbb{R}$, find $s \in S(\Omega)$ such that

$$s(x_i) = y_i, \quad i=1,2,\dots,N \quad (\text{Eq. 5.1})$$

A popular choice to solve the above problem is to use Radial Basis Functions (RBF) as discussed in the following subsection. Typical works using RBF interpolation for example skinning are also investigated in this section.

5.1.1 Radial Basis Functions

Figure 5.1: 1D Radial Basis Function Interpolation

Radial basis functions (RBF) [118, 36] are frequently employed for scattered data interpolation in computer graphics. They are characterized by having a value that is determined solely by one parameter, the distance from a center point in the multi-dimensional

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

abstract space [132]. Related applications in computer graphics can be found in image warping [16] and 3D interpolation [141]

Suppose there is a set of data points $\{(x_i, y_i)\}$, where $i = 1, \dots, N$. An interpolated point from RBF is given by

$$y = F_{rbf}(x) = \sum_{i=1}^N \omega_i \phi(\|x - x_i\|) \quad (\text{Eq. 5.2})$$

The values of ω_i from Eq. 5.2 are obtained by solving a least square system such that

$$\sum_{i=1}^N \|y_i - F_{rbf}(x_i)\| \quad (\text{Eq. 5.3})$$

is minimized. Any function that satisfies the property $\phi(x) = \phi(\|x\|)$ will interpolate the data ¹ [96], and in this dissertation Gaussian function $\phi_i(x) = \exp(\frac{-\|x-x_i\|^2}{2\sigma^2})$ will be used. To minimize Eq. 5.3, a set of basis vectors ϕ_i will be formed:

Figure 5.2: Scattered Interpolation. Left: Shepard's method; Right: RBF interpolation. (Image Courtesy of [96])

$$\phi_i = \begin{bmatrix} \phi(\|x_1 - x_i\|) \\ \vdots \\ \phi(\|x_j - x_i\|) \\ \vdots \\ \phi(\|x_N - x_i\|) \end{bmatrix}$$

Denote $\phi(\|x_j - x_i\|)$ as ϕ_{ji} , thus a $N \times N$ matrix Φ is written as:

¹http://en.wikipedia.org/wiki/Radial_basis_function

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \dots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \dots & \dots & \phi_{2N} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \phi_{j1} & \dots & \phi_{ji} & \dots & \phi_{jN} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \dots & \dots & \phi_{NN} \end{bmatrix}$$

The least square system Eq. 5.3 is then equivalent to

$$\Phi \mathbf{w} = \mathbf{y} \quad (\text{Eq. 5.4})$$

where $\mathbf{w} = (\omega_1, \dots, \omega_N)$. Since Φ is a symmetric positive, a unique solution should exist for building the interpolation scheme Eq. 5.2. In the Gaussian function, parameter σ is used to control the “fall-off”. In Figure 5.1, we use Gaussian Radial Basis functions to interpolate 3 points. The red and blue curves represent $\sigma = 1.0$ and $\sigma = 2.0$ respectively. Other basis functions also can be candidates.

RBF interpolation presents several advantages. First of all it will result in a smooth interpolant. Further, it extrapolates to zero and shows a better quality in the first derivative at the data points compared with Shepard’s method [26] (shown as Figure 5.2).

5.1.2 Shape by Example

Shape by Example (SBE) is proposed in [132], which defines a *shape* as a continuous range of forms generated from given examples on the fly. Each example is annotated by a value in an abstract (pose) space designed by users. To interpolate the examples, a linear approximation is first used to find a hyperplane in the abstract space that comes closest to approximating the example values. What follows is RBF interpolation which is performed upon the residuals between examples and the results from the linear approximation. The linear part provides an overall approximation to the space defined

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

by the examples and permits extrapolation outside the convex hull of the locations of examples. To generalize, the basis functions used in this work is:

$$\omega_{i_1}(x) = \sum_{i_2=1}^N r_{i_2 i_1} F_{i_2}(x) + \sum_{l=0}^D a_{i_1 l} A_l(x) \quad (\text{Eq. 5.5})$$

where $\omega_{i_1}(x)$ are the weights for example i_1 , $r_{i_2 i_1}$ are the radial basis function weights, $F_{i_2}(x)$ are the radial basis functions themselves, the $a_{i_1 l}$ and A_l are the linear coefficients and linear bases. The subscripts i_1 and i_2 are both example indices. Thus at runtime for the j th vertex, its position is obtained by

$$p_j(x) = \sum_{i_1=1}^N \omega_{i_1}(x) p_{i_1 j} \quad (\text{Eq. 5.6})$$

where N is the number of examples and $x_{i_1 j}$ is the vertex position of the j th vertex in example i_1 .

5.1.3 EigenSkin

EigenSkin [88] also involves runtime interpolation from given examples. SSD is first built and the displacements are computed by comparing the SSD results against given examples in corresponding poses. One contribution made in this work is that instead of directly interpolating the displacements, Principle Component Analysis (PCA) is first conducted to obtain eigen vectors of displacements per each joint. This strategy is based on the fact that significant redundancy exists in the pose displacements when treating large amount of input simulated data. The detailed technique is to construct a rectangular matrix A_j of size $3|S_j| \times |P_j|$ for each joint j . $|S_j|$ is the set of vertex indices supported by j and P_j is the pose indices used to compute S_j . Using Singular Value Decomposition (SVD), $A_j = U_j D_j V_j^T$, the matrix U_j has the same size as A_j and consists of columns of eigendisplacements for those vertices influenced by j , which has the same block format that used to build A_j . The singular values, in the diagonal matrix

D_j , identify the importance that each eigendisplacement has in reproducing the observed poses. The combination of the matrix V_j and the singular values gives the coordinates of the observed displacements in the eigendisplacement basis.

Given a threshold (determined by the hardware limits), each observed eigendisplacement basis, using the above technique, can be truncated with the error minimized in the least squares sense. At runtime computing the eigendisplacement coordinates in arbitrary configurations still involves solving a scattered data interpolation problem using radial basis functions. Note that, in this work, a direct inverse operation of SSD skinning model is performed to form the observed displacements which is implemented simply as a matrix inverse operation. In this dissertation, SSD is treated as an unknown black box and an optimization problem is formulated to allow the usage of any skinning model. Besides, the formulated strategy can handle some singular poses (i.e. a singular transformation matrix) which are not treated in [88].

Compared with SBE and EigenSkin, one distinction presented in this dissertation is that only pure geometries of example meshes are known. Extracting skeletons and parameterizing pose parameters are nontrivial as discussed in the previous chapters. In the following, improved PSD in production environment is investigated, analyzed, and implemented based on the skeletonization and parametrization results.

5.2 Pose Space Deformation

5.2.1 Formulating Pose Space

Pose space comprises a set of parameters from controlling objects such as rotation angles, translation displacement vectors, scaling values of joints, etc, or parameters indicating character features such as age, sex, stature etc. [132], or some other values displaying facial status or even mental emotions (See the Figure 5.3) [96]. These parameters not only serve as the spaces for interpolation, but also provide convenient interface to animators

Figure 5.3: Pose Space Formulation (Image Courtesy of [96])

for tuning animations freely. Therefore, pose space parameters should be semantically meaningful. Specifically, a rotation angle, a translation or scaling value can be used as pose parameters but a 4×4 affine transformation should be hidden from users.

In previous literature on PSD, users build pose spaces for examples and thus PSD is performed by assuming that all pose spaces are available and naturally suitable for associated surface geometries. However, in many circumstances, example geometries come without associated pose parameters, and thus a parameterization is necessary. In the previous chapter an intuitive scheme to find joint parameters for example pose is proposed. In this Chapter, pose parameters are known and are used as input for PSD scheme.

5.2.2 Algorithm Overview

The famous SSD problem of “collapsed elbow” is recognized in [96] as being due to the fact that deformation is limited to a linear subspace. Because of this limitation SSD cannot synthesize many parts of a character skin involving complicated joint structures.

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Building on the SSD scheme, the Pose Space Deformation (PSD) is proposed by [96] as a combination of SSD and shape blending which provides a nice solution to above mentioned problems. PSD can be performed in the following steps:

- Example pose setup: move the character to problematic poses, sculpt and store pose information (joint configuration x) including the amount of movement of each sculpted vertex (delta values d).
- In the inverse PSD approach, transforming example models (d) to the rest pose (d^r): this step is trivial if the basic skinning, say SSD, is explicit. In our proposed framework, this step will be replaced by an optimization routine.
- Solving a linear system: we set up an interpolation scheme for delta $\Phi(x)\omega = d$ and solve it to obtain weights ω for all example poses.
- Realtime synthesis: for an intermediate pose, we obtain the delta in the rest pose by interpolating example poses at runtime. We add this delta to the original character surface and then let SSD or any other skinning scheme finish the final transformation.

For a vertex v , sculpted in N example poses, there are N delta $d^i, i = 0, \dots, N - 1$ corresponding to each pose $x^i, i = 0, \dots, N - 1$. These are converted to rest pose displacements using $d^{ri} = SSD^{-1}(d^i)$. The Gaussian Radial Basis functions are used to interpolate d^{ri} . First a $N * N$ matrix Φ is built with the $(i, j)th$ element as $\phi(\|x^i - x^j\|)$, where $\|x^i - x^j\|$ means the Euclidean distance between pose x^i and pose x^j , then we have a linear system:

$$W = \Phi^{-1}D^r \quad (\text{Eq. 5.7})$$

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Here W and D^r are column vectors with i th element ω_i and d^{ri} respectively. In the synthesis phase, for an intermediate pose x , we can obtain the delta d for this vertex by:

$$d = \sum_{i=0}^{N-1} \omega_i \phi(\|x^i - x\|) \quad (\text{Eq. 5.8})$$

Although PSD and improved example-based schemes have been discussed in many publications [88, 90, 132], the reason why the inverse should be performed is still ambiguous. In the next section this issue will be analyzed to demonstrate why inverting the SSD (and other deformations) in order to interpolate the examples in the rest pose is the right choice.

5.2.3 PSD in Practical Applications

PSD had been applied successfully in movie industry [96]. As for its implementation, a set of issues must be addressed. First, one must carefully define a pose vector for a particular vertex. That is to answer which controls or parameters and how many of them consist of this pose vector. If the dimension of this vector is not sufficient, a singularity in RBF interpolation will occur. This singularity can be described by a particular scenario that if a joint which should be incorporated in the pose vector is not counted as an effective controller, and if there are displacements of the vertex resulted by this joint at two different key poses, a set of interpolant data, which has two different RBF function values from the same variable, will be generated. The interpolation of this data set is ill defined. This can be explained from Eq. 5.2 as the following:

$$y_i = f_{rbf}(x), y_j = f_{rbf}(x), \text{ for } y_i \neq y_j \quad (\text{Eq. 5.9})$$

On the other hand, if the dimension is more than necessary, animators will be surprised by the undesired deformations from those redundant controllers. For instance, rotating a leg knee will result in a bulge in an arm elbow. To avoid the above problems,

defining an appropriate pose vector can be handed over to animators, which is the case in SoftImage system [3]. Animators first define a set of joints or controllers, and then the influenced surface vertices, set the value of controllers at key poses, sculpt influenced areas, define the key poses, and let the system do the interpolation. The implementation in this dissertation is simplified. The pose vector is defined by the SSD scheme, in which only those joints with non-zero weights will be counted in pose vector. Setting a weight value as zero can erase unwanted joints. As for the singularity, our system simply flags an exception to users.

5.3 Improvement of PSD by Inverse Operation

This section will describe the implementation of our inverse algorithm and why it is an improvement. SSD is used as the underlying skinning, since an explicit form of basic skinning can help simplify our task of explanation. We call the PSD scheme without the inverse operation the “forward PSD”, and comparison to it will be used to demonstrate the superiority of the inverse method.

5.3.1 Deformation Direction Analysis

Figure 5.4: (a)Rest pose; (b)Example pose with rotation of 90 degrees

Figure 5.5: Deformation direction

In this section the deforming effects of both forward PSD and inverse PSD are analyzed and compared, and why the inverse method is superior is explained.

Given two examples as shown in Figure 5.4 (a) and (b) respectively, vertex v with the position v^r in the rest pose (0 degree) is sculpted to a “target position” v_i^t in an example pose (90 degrees). The delta value in the first pose is zero. Then we apply forward and inverse PSD respectively to interpolate these two poses. For an intermediate pose x , we have two distinct deforming vertices resulting from these two algorithms, as illustrated in Figure 5.5, $v_{ssd}^x, v_p^x, v_{Ip}^x$ are the deformed positions from SSD, forward and inverse PSD in an intermediate pose x . We use two angles α_p and α_{Ip} to analyze how directions of a deformed vertex change with the pose. In the forward case, α_p is formed by the vector (v_{ssd}^x, v_p^x) and the line $y = Y_{v_{ssd}^x}$, where $Y_{v_{ssd}^x}$ is the y coordinate of v_{ssd}^x . For the two examples shown in Figure 5.4 (a) and (b), we have delta values $\mathbf{d}_1 = [d_{1x}, d_{1y}]$ and $\mathbf{d}_2 = [d_{2x}, d_{2y}]$. From Eq. 5.7:

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

$$\begin{vmatrix} \omega_{1x} & \omega_{1y} \\ \omega_{2x} & \omega_{2y} \end{vmatrix} = \begin{vmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{vmatrix}^{-1} \begin{vmatrix} d_{1x} & d_{1y} \\ d_{2x} & d_{2y} \end{vmatrix}$$

Because in the rest pose, no movement for v is generated, then $\mathbf{d}_1 = [d_{1x}, d_{1y}] = [0, 0]$. Taking the model in the rest pose as an example is a common practice when applying shape interpolation, since interpolating effects from other examples should not change the original model in rest pose. Therefore, by solving the above equation we have:

$$\begin{aligned} \omega_{1x} &= \phi_{11}^{-1}d_{1x} + \phi_{12}^{-1}d_{2x} = \phi_{12}^{-1}d_{2x} \\ \omega_{2x} &= \phi_{21}^{-1}d_{1x} + \phi_{22}^{-1}d_{2x} = \phi_{22}^{-1}d_{2x} = d_{2x} \\ \omega_{1y} &= \phi_{11}^{-1}d_{1y} + \phi_{12}^{-1}d_{2y} = \phi_{12}^{-1}d_{2y} \\ \omega_{2y} &= \phi_{21}^{-1}d_{1y} + \phi_{22}^{-1}d_{2y} = \phi_{22}^{-1}d_{2y} = d_{2y} \end{aligned}$$

where ϕ_{ij}^{-1} is the (i, j) th element of Φ^{-1} , and if $i = j$, $\phi_{ij}^{-1} = 1$. Then in an intermediate pose x for α_p , we have $\tan \alpha_p = \frac{d_y}{d_x}$. d_y and d_x are delta values in x, y coordinates computed from equation Eq. 5.8. With the above weight values, we have:

$$\begin{aligned} \tan \alpha_p = \frac{d_y}{d_x} &= \frac{\omega_{1y}\phi(x - x_1) + \omega_{2y}\phi(x - x_2)}{\omega_{1x}\phi(x - x_1) + \omega_{2x}\phi(x - x_2)} \\ &= \frac{\phi_{12}^{-1}d_{2y}\phi(x - x_1) + d_{2y}\phi(x - x_2)}{\phi_{12}^{-1}d_{2x}\phi(x - x_1) + d_{2x}\phi(x - x_2)} \\ &= \frac{d_{2y}}{d_{2x}} \end{aligned}$$

We can see this angle α_p is a constant and depends only on the value of delta in the second pose $\mathbf{d}_2 = [d_{2x}, d_{2y}]$. Now take a look at α_{Ip} in the inverse case. Two examples are transformed to rest pose to obtain delta values: $\mathbf{d}_1^r = [d'_{1x}, d'_{1y}] = [0, 0]$ and $\mathbf{d}_2^r = [d'_{2x}, d'_{2y}]$. Since only the second joint is rotating, SSD is simplified as a rotation transformation and

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

ignoring other issues such as accumulating effects from the first joint:

$$SSD_{\theta} = \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where θ is the rotation angle of the second joint. Then for the vertex $v^r = [v_{0x}, v_{0y}]$, $SSD_{\theta}(v^r)$ transforms v from the rest pose to $[v_{SSD_x}, v_{SSD_y}] = [v_{0x} \cos \theta - v_{0y} \sin \theta, v_{0x} \sin \theta + v_{0y} \cos \theta]$. In an intermediate pose x , its corresponding rest position is obtained as $v_x^r = [v_{0x} + d'_x, v_{0y} + d'_y]$, and here the $[d'_x, d'_y]$ is the interpolated result from equation Eq. 5.8. We just apply the simplified SSD to v_x^r to obtain v_{Ip}^x : $v_{Inp_x} = (v_{0x} + d'_x) \cos \theta - (v_{0y} + d'_y) \sin \theta$ and $v_{Inp_y} = (v_{0x} + d'_x) \sin \theta + (v_{0y} + d'_y) \cos \theta$. Similarly, the tangent of α_{Ip} is computed as:

$$\begin{aligned} \tan \alpha_{Ip} &= -\frac{v_{Inp_y} - v_{SSD_y}}{v_{Inp_x} - v_{SSD_x}} \\ &= -\frac{d'_x \sin \theta + d'_y \cos \theta}{d'_x \cos \theta - d'_y \sin \theta} = -\tan(\beta + \theta) \end{aligned}$$

where $\tan \beta = \frac{d'_y}{d'_x} = \frac{d_{2y}}{d_{2x}}$. Then we can see $\alpha_{Ip} = -(\theta + \beta)$, which is linearly proportional to the pose rotation θ .

Now we take a look at a real cylinder model with one vertex sculpted in the second pose, shown in the Figure 5.6. The Forward PSD and the corresponding inverse PSD in the same poses (30, 45 and 60 degrees of one rotated joint) are illustrated respectively in Figure 5.7. We can see that in the forward case, the direction of deformed vertex always remains the same with the example cylinder (figure 5.6). For the inverse PSD however, that direction is changed along with the rotation of the joint. The case described above is quite common in practice when animating shoulder, elbow, knee, hip-bone, neck, etc. All these parts would rotate from the rest pose with some angle to other poses. On the other hand, as a matter of experience, PSD is supposed to be a method of “local”

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

correction, which means pose space should not be extended to the whole space that has to incorporate all influenced objects. Otherwise, large amount of unnecessary works of building examples will be required, and the distance between different poses is also meaningless. For example how to measure the distance between differing poses such as “lying down” and “pitching”?

Figure 5.6: A simple test case: two example poses with one vertex sculpted

5.3.2 Direction Set (Powell’s) Methods for Minimization

For a minimization problem, there are many candidate algorithms according to the form of function, knowledge of the derivative, computing capacity, and requirements for the rate of convergence, etc. In our situation, the function form is not explicit, and the computing burden increases with the number of example poses. We will adopt Powell’s method as the solution to this minimization problem.

One advantage of Powell’s classic method is that it does not need explicit computation of the function’s gradient [120]. Because we are treating the skinning operations as a “black box”, their gradient is not available, so Powell’s method is suitable. Minimizing the function $f(d')$ in a particular direction is actually minimization problem of one variable, which is also called *line minimization*. Powell’s idea is trying to find each minimum of the function $f(d')$ in different directions until $f(d')$ stops decreasing. How to choose the next direction is the main concern of Powell’s method, and it has been proven that after

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Figure 5.7: Interpolated Poses: the rotated joints are at 30, 45 and 60 degree. Left column: PSD; Right column: inverse PSD.

repeated cycles of M line minimizations on conjugate directions, the optimization will in due course converge to the minimum [120]. M is the dimensionality of the function $f()$.

To solve this minimization problem, conjugate gradient method can also provide an optimal solution. However, it will involve computing the first derivative of an objective function which will infer the calling API functions of production tools (i.e. Maya, 3DS Max, SoftImage etc.). Such evaluation will cause extra computation burden and we therefore adopt the Powell method that has a slower converge rate but it needs less time

in calling objective function compared to conjugate gradient method.

5.3.3 A Unified Framework for Inverse Skinning Model

In this sub-section, how inverse PSD works is explained. For N examples, a vertex v is first transformed from the rest pose by SSD to positions $v_i, i = 0, \dots, N - 1$. Then animators move it to example positions to obtain delta values $d_i, i = 0, \dots, N - 1$. The final positions of v in the example poses are $v_i + d_i, i = 0, \dots, N - 1$, and we call them target positions v_i^t . The above operations are summarized as:

$$v_i^t = v_i + d_i = SSD_i(v^r) + d_i$$

where v^r means the rest position of v and $SSD_i(*)$ represents Eq. 4.1. The “forward PSD” approach then concludes by interpolating d_i as a function of pose.

In the inverse approach we instead apply the inverse of $SSD_i(*)$ to v_i^t to obtain a modified rest pose vertex v_i^r . The difference of v_i^r and v^r produces new delta value d_i^r , which will be the input of the linear system (equation Eq. 5.7) introduced in the previous section.

$$d_i^r = SSD_i^{-1}(v_i^t) - v^r$$

In this step we need to implement the inverse skinning operator SSD^{-1} . Since SSD is a 3D transformation, SSD^{-1} is simply the inverse transformation matrix generated by SSD. For the situation where other unknown skinning operations are adopted, we propose a unified framework which will be discussed in the next section. Next we build a new delta vector D^r with the i th element as d_i^r , and replace D in equation Eq. 5.7 with d_i^r to get a new weight vector W^r .

$$W^r = \Phi^{-1} D^r \quad (\text{Eq. 5.10})$$

In the synthesis phase, for an intermediate pose x we have:

$$d_x^r = \sum_{i=0}^{N-1} \omega_i^r \phi(\|x_i - x\|) \quad (\text{Eq. 5.11})$$

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

then we add this d_x^r to v^r and let SSD finish the rest of the job:

$$v_x = SSD_x(v^r + d_x^r)$$

where v_x represents the final position of vertex v in pose x .

The above discussions assume that the basic skinning algorithm is SSD , but in many circumstances, other deformation schemes may be adopted [102, 131], most of which have been implemented in animation packages. Therefore we propose a unified framework in which no explicit inverse operation is necessitated.

Given a basic skinning method supported by any animation package we can deform the original character model from rest pose to another specific pose. In the more general case, we need to replace SSD with $SKINNING$ in the inverse skinning algorithm as such for equation Eq. 5.10:

$$v_i^t = v_i + d_i = SKINNING_i(v^r) + d_i$$

But this time we do not implement the inverse of $SKINNING$ as in equation Eq. 5.10.

To find delta d_i' in the rest pose:

$$v_i = SKINNING_i(v^r) + d_i = SKINNING_i(v^r + d_i')$$

we can set up a minimization problem to minimize the function:

$$f(d_i') = \|v_i - SKINNING_i(v^r + d_i')\|^2 \quad (\text{Eq. 5.12})$$

This function can be given to the Powell's method to find d_i' at the minimum of $f(d')$. For each example pose P_i , we have a d_i' . We then apply radial basis function to d_i' ($i = 0 \dots n - 1$) in the pose space to obtain ω_i ($i = 0 \dots n - 1$). In synthesis phase, a d_x' in an intermediate pose x can be computed by equation Eq. 5.8 based on its position x in pose space $d_x' = \sum_{i=0}^{n-1} \omega_i \phi(\|x - x_i\|)$. Then we have the final synthesis result:

$$v_x = SKINNING(v^r + d_x') \quad (\text{Eq. 5.13})$$

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Figure 5.8: System Pipeline for Unified Inverse Skinning.

This unified approach is implemented as a Maya plug-in. In Maya, “tweaking” is a procedure adding delta values to original surface vertices before any deformations. It is actually Maya’s form of rest-pose editing for their built-in deformation operators. As presented in Figure 5.8, the whole system is divided into two phases. The first phase is to find each delta in the rest pose corresponding to each example pose. Basic skinning provided by Maya is called in the loop of minimization scheme.

The output of the first phase, the delta in the rest pose, is input to the second phase

that is a linear system performing RBF interpolation to obtain the PSD weights. In the synthesis process, for an intermediate pose x , a delta d'_x (or d''_x) is synthesized by equation Eq. 5.11. The final deformed vertex is computed by Maya skinning as in Eq. 5.13.

5.3.4 Singular Poses

If the SSD transformation in Eq. 4.2 is singular, the inverse PSD deformation will not be possible. Deformation that lies in the null space of the SSD matrix will be ignored. Although singular cases (one example is a joint rotated by 180° degrees with equal $\frac{1}{2}, \frac{1}{2}$ weights on the two joint frames, which is an unrealistic case of self-intersection) are rare, it is possible to handle these cases with a small rearrangement of the inverse PSD approach. We reformulate the problem as

$$f(y_i) = \|v_i + w_i - SKINNING_i(v^r + d'_i)\|^2 + \lambda \|w_i\|^2$$

where y_i is a concatenated vector $y_i = [d'_i, w_i]$ and λ is an arbitrary small number. The final synthesis is then

$$v_x = SKINNING(v^r + d'_x) + w_x$$

where w_x is interpolated after SKINNING by applying the same RBF scheme as used for d'_x (thus, only minimal code changes are required). The idea here is that, since w_i is being minimized, it will generally be zero, and will be non-zero only if it is not possible to obtain the desired deformation v_i using $SKINNING_i(v^r + d'_i)$.

In the case where the SSD transform is near-singular, the solved d'_i can be much larger than other d'_k , which can result in poorly posed interpolation. To address this case, we further modify the objective function as

$$f(y_i) = \|v_i + w_i - SKINNING_i(v^r + d'_i)\|^2 + \lambda \|w_i\|^2 + \mu \|d'_i\|^2$$

to prevent very large d'_i where 0.0001 is a sufficient value for both λ and μ .

5.4 Results and Discussions

When PSD was first introduced in [96], it was suggested that the interpolation should be performed in the rest pose and some follow up works such as [88] did the same by multiplying the inverse transformation of SSD. However, there are no explanations on why this is a better practice and no further studies on how to apply the inverse operation when the underlying skinning scheme is not SSD or when it is an entirely unknown scheme, which is the case in most real production environment where the underlying skinning is a combination of many schemes and there is no uniform mathematic model available for inverse operation. In this chapter, details on why and how the inverse operation can improve the interpolation results are provided. To our knowledge, this provides for the first time a clear theoretical reasoning why inverse operation is required. Editing geometry is formulated in rest pose as an optimization problem and a unified framework which can be implemented on high-end commercial packages while allowing any proprietary skinning operators to be incorporated is proposed. The main work is published in [148].

Figure 5.9: **toad**: closeup of circled part from figure 9. left: PSD; right: inverse PSD.

Inverse skinning integrates SSD and shape interpolation better than its forward rival. It is demonstrated that the direction of a deformed vertex in inverse skinning is linearly

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Figure 5.10: **toad**: a sequence of animated frames. left: PSD; right: inverse PSD.

proportional to joint rotations in a simplified example, while the forward PSD does not incorporate the direction information. Therefore the inverse approach presents better performance and more consistent interpolation (Figure 5.10 to Figure 5.12).

By formulating the inverse process as a minimization problem a unified model is proposed not only for SSD but also for other skinning schemes, into which shape interpolation can be incorporated. But the minimizing process will introduce more cost. This cost depends on the size of deformed character, parameters of minimization methods (Powell) such as convergence precision, and the number of example poses. In addition the cost of the animation software must be considered (for example, the Maya API implements a run-time type interpretation system on all operations). The cost of the inverse

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

operation is not critical, however, since it is a one time “setup” cost, and the computing time is insignificant compared to the human time required to sculpt the desired deformations. Once the linear system is solved, the synthesis is potentially realtime since no extra computing is involved in this process compared to the forward PSD. This unified example-based approach is implemented as a Maya plugin. It interoperates with Maya’s closed-source “Smooth Skinning” deformation [84].

CHAPTER 5. AN OPTIMIZATION SCHEME FOR EXAMPLE BASED APPROACHES

Figure 5.11: PSD vs. Inverse PSD: with two example poses, one in rest pose and the other is sculpted when the thumb is rotated down 66.8 degrees. SSD shows an obvious collapse. The intermediate poses from SSD, PSD and inverse PSD are shown in the second (45 degrees) and third (60 degrees) rows respectively.

Figure 5.12: human arm

Chapter 6

Skinning Examples upon Spherical Blending

In this chapter, a skinning framework by layering example based approach upon the spherical blend skinning (SBS) is proposed. The prototype skeleton subspace deformation (SSD) method is replaced by SBS, which partly removes notorious artifacts such as collapsing-elbows and candy-wrappers induced by SSD while maintaining comparable efficiency with SSD. Further, example based approach will be built by extending deformation using spherical space instead of linear space to provide more convincing effects.

The aim is to seek a balance between the deformation quality and runtime speed. It is known that by providing more examples, the desired effects can be captured better, but it is at the expense of performance efficiency. SBS is able to significantly reduce artifacts induced by SSD while still maintaining comparable efficiency. Example shapes can provide more artistic deformation effects that cannot be captured through mathematical solutions. Therefore, we are motivated to combine SBS and example based approach without introducing too many examples while still maintaining reasonable speed for real productions.

To summarize, in this chapter, spherical blend skinning (SBS) is adopted as the underlying scheme. Example based approach is further layered upon SBS to capture more deformations from those examples. Fig. 6.1 gives a simple illustration. When the joint

Figure 6.1: Illustration for SSD, SBS and PSD

J2 rotates, each area with different grey level represents deformed effects from SSD, SBS and PSD respectively.

6.1 Interpolation in Spherical Space

Joint orientation is treated as a transformation matrix in ad hoc skinning algorithm skeletal subspace deformation, in which rotations are defined as Euler angles [25]. Dealing with orientations using Euler angles, “gimbal lock” may result when two axes effectively line up, resulting in a temporary loss of a degree of freedom (shown in Figure 6.2). There is another representation for rotations which is more favored in animation field: quaternions. In this section, quaternions and their usage in animation are investigated.

6.1.1 Quaternion

In mathematics, quaternions are a non-commutative extension of complex numbers, which are normally written as the combination of a scalar value s and a vector value

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

Figure 6.2: Gimbal Lock

$\mathbf{v}$ such as:

$$\mathbf{q} = \langle \mathbf{s}, \mathbf{v} \rangle \quad (\text{Eq. 6.1})$$

where $s = q_0$ and $\mathbf{v} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3]$. A quaternion can represent a rotation by an angle θ around a unit axis $\mathbf{a}$:

$$\mathbf{q} = \left[\cos \frac{\theta}{2}, \mathbf{a}_x \sin \frac{\theta}{2}, \mathbf{a}_y \sin \frac{\theta}{2}, \mathbf{a}_z \sin \frac{\theta}{2} \right] \quad (\text{Eq. 6.2})$$

or

$$\mathbf{q} = \left\langle \cos \frac{\theta}{2}, \mathbf{a} \sin \frac{\theta}{2} \right\rangle \quad (\text{Eq. 6.3})$$

A rotation matrix can be obtained by a quaternion as follows:

$$\begin{pmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{pmatrix}$$

An inverse of a quaternion is given by:

$$\mathbf{q}^{-1} = \frac{1}{\|\mathbf{q}\|^2} \langle \mathbf{s}, \mathbf{v} \rangle \quad (\text{Eq. 6.4})$$

and a vector rotated by a quaternion is:

$$\mathbf{v}' = \mathbf{q}^{-1} \mathbf{v} \mathbf{q} \quad (\text{Eq. 6.5})$$

Shoemake demonstrates that quaternions present better behaviors in rotating orientations [129]. Thus he suggests that interpolation in key framing techniques should be treated using quaternions instead of Euler angles.

Figure 6.3: SLERP and QLERP (Image Courtesy of [84])

6.1.2 Interpolating using Quaternions

In computer graphics, SLERP is shorthand for spherical linear interpolation introduced by Ken Shoemake in the context of quaternion interpolation for the purpose of animating 3D rotation. It refers to a constant speed motion along a unit radius great circle arc, given the ends and an interpolation parameter between 0 and 1 (illustrated in Figure 6.3). SLERP is formulated as:

$$s(t; \mathbf{p}, \mathbf{q}) = \frac{\sin((1-t)\theta)\mathbf{p} + \sin(t\theta)\mathbf{q}}{\sin \theta} \quad (\text{Eq. 6.6})$$

Instead of interpolating quaternions on a circular arc, Kavan and Zara propose to perform interpolation along the shortest segment [84], which is called QLERP as shown in Figure 6.3:

$$l(t; \mathbf{p}, \mathbf{q}) = \frac{(1-t)\mathbf{p} + t\mathbf{q}}{\|(1-t)\mathbf{p} + t\mathbf{q}\|} \quad (\text{Eq. 6.7})$$

Although this slight change does not result in uniform interpolation on the arc, it is sufficient for skinning in which blending a set of joint rotations will be involved. [84] proves that the difference between SLERP and QLERP is strictly constrained in 0.143 radians.

6.2 Modified Spherical Blend Skinning

Figure 6.4: Skeleton Subspace Deformation vs. Spherical Blend Skinning

It has been observed that deforming shapes should be decomposed into rigid and elastic parts. “The rigid part performs the general positional changes, while the fine details are gradually changed by the elastic part” [7]. The reason that sometimes SSD fails can be reinterpreted as blending being performed by mixing rigid part and elastic part, which should be handled separately. This is the idea that SBS is based on: each transformation matrix of joint will be decomposed into a pure rotation (elastic) part and a translation (rigid) part, and then be blended respectively.

Given the blending method in Eq. 4.2, the core of SBS is to extract the rotation part of transformation matrix M_i^p , and converts them into quaternions, which are further blended. The algorithm is given as:

$$v^p = SBS_p(v^r) = Q^p(v^r - r^p) + \sum_{i=1}^J \omega_i M_i^p r^p \quad (\text{Eq. 6.8})$$

Compared with equation Eq. 4.2, new terms Q^p (blended transformation of pure rotation part) and r^p (rotation center) are introduced. On how to formulate the above algorithm, please refer to [84]. In this section we will fully explore how to evaluate those terms.

For M_i^p in Equation Eq. 4.2, we denote each entry as $m_{h,k}$. Then M_i^p can be expanded

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

as:

$$M_i^p = \begin{pmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,3} \\ m_{1,0} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,0} & m_{2,1} & m_{2,2} & m_{2,3} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_i^p & l_i^p \\ 0 & 1 \end{pmatrix}$$

where R_i^p and l_i^p are rotation and translation parts of transformation M_i^p respectively. R_i^p will be converted to quaternions and used to determine the rotation center together with l_i^p .

6.2.1 Singular Value Decomposition

In linear algebra, the singular value decomposition (SVD) is an important factorization of a rectangular real or complex matrix, with several applications in signal processing and statistics applications which employ the SVD include computing the pseudoinverse, least squares fitting of data, matrix approximation, and determining the rank, range and null space of a matrix [1]. It is a method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships among the original data. It is also an approach for identifying and ordering the dimensions along which data points exhibit the most variation. And thus it is possible to find the best approximation of the original data points using fewer dimensions. In other words, SVD can be seen as a method for data reduction as shown in subsection 5.1.3. In the following the details of finding rotation center using SVD are investigated.

6.2.2 Determine Rotation Center

The rotation center is defined as the point whose positions transformed by the assigned joints are as close as possible. That is:

$$M_i^p c^p = M_j^p c^p \quad (\text{Eq. 6.9})$$

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

where $i, j = 1, \dots, m$. We use $(c^p, 1)'$ as the homogeneous representation of the center c^p , then Equation Eq. 6.9 can be written as:

$$\begin{pmatrix} R_i^p & l_i^p \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c^p \\ 1 \end{pmatrix} = \begin{pmatrix} R_j^p & l_j^p \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_p \\ 1 \end{pmatrix}$$

such that:

$$\begin{aligned} R_i^p c^p + l_i^p &= R_j^p c^p + l_j^p \\ (R_i^p - R_j^p) c^p &= l_j^p - l_i^p \end{aligned}$$

Stacking all above equations for i, j we have:

$$\mathbf{R} c^p = \mathbf{l}$$

where $\mathbf{R}$ is a $3m \times 3$ matrix and $\mathbf{l}$ is a $3m$ column vector. A robust singular value decomposition (SVD) method is adopted to solve for rotation center c^p . It is not necessary to perform SVD for each vertex since the computation involved is only related to joint transformation. As stated in [84], the center position can be evaluated for each joint set and further cached for look-up when handling each vertex.

6.2.3 Quaternion Rotations Blending

We need to expand R_{pk} to homogeneous matrices and then convert them into quaternions, which are trivial works. However blending quaternions are somewhat complicated even though a straightforward linear blending is performed in [2]. The algorithm is described as following:

- R_k^p are extended to homogeneous matrices:

$$R_k^p = \begin{pmatrix} t_{0,0} & t_{0,1} & t_{0,2} & 0 \\ t_{1,0} & t_{1,1} & t_{1,2} & 0 \\ t_{2,0} & t_{2,1} & t_{2,2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- R_k^p are further converted to quaternions q_k^p .
- Choose a pivot from q_k^p . Here we adopt a strategy slightly different from that of [84]. For a group of quaternions $q_k (k = 1, \dots, m)$ (ignoring the subscript p), we will choose a particular quaternion q_{pivot} with the maximum value of $|q_k.x + q_k.y + q_k.z|$ ($k = 1, \dots, m$) as a pivot. For each of the other quaternions $q_k (k \neq pivot)$, we compute the distances d_+ from q_{pivot} to q_k , and d_- from q_{pivot} to $-q_k$ respectively. If $d_+ < d_-$, q_k remains unchanged. Otherwise, we replace q_k with its inverse $-q_k$.
- Obtain q^p by linearly blending q_k^p : $q^p = \sum_{k=1}^m \omega_k q_k^p$
- Convert q^p to matrix Q^p .

By now we have obtained Q_p and c_p , both of which are run-time parameters that have to be evaluated for each frame. As stated earlier, we compute them only for each joint set and not necessary for each vertex. Fig. 6.4 demonstrates the superiority of SBS over SSD when a human elbow twists.

6.3 Pose Space Deformation on Spherical Blended Models

SBS performs deformations in the spherical space. This is an apparent advantage over SSD, of which the linear blending results in many obvious artifacts [96]. A skinning framework by layering example based approach upon SBS is proposed. SSD will be replaced by spherical skinning, and further, the example based approach provides more convincing deformation by extending from spherical space instead of linear space.

Heuristically, blending transformations in spherical space naturally inherits the properties of articulate animations. Artifacts resulted from linear blending such as candy wrapper effects and collapsing joints can be partially removed. We say “partially” because detailed deformations still cannot be captured by simply performing SBS. Example

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

Figure 6.5: Example based skinning strategies: (a) an example at the rest pose; (b) an example at the pose of “elbow-twist” (rotated -130 degrees along x axis); (c) Blended shape by PSD being layered on SSD (rotated -110 degrees along x axis); (d) Blended shape by PSD being layered on SBS (rotated -110 degrees along x axis).

shapes in varied poses are necessary to provide more convincing animation effects. In other words, deformations for articulated characters should neither be limited to linear space nor spherical space. However, the detailed deformations are achieved at the expense of runtime speed. With more examples, the runtime synthesis speed is slower. Compared with conventional example based methods which are built upon SSD, our framework shows apparent advantages. First, the number of example shapes can be reduced due to the fact that SBS is already capable of removing some artifacts. Second, displacement values obtained by subtracting problematic shapes from examples will be smaller than those from conventional methods. Similarly, in practice, re-sculpting efforts to obtain the same example shape are reduced, which is particularly useful in productions since re-sculpting badly collapsed shapes is difficult and tedious.

Therefore we propose to layer pose space deformation upon spherical blend skinning.

The system flowchart is illustrated in the Figure 6.6.

Figure 6.6: System Flowchart

6.4 Results and Discussions

SBS presents a better blending on joint transformations compared with SSD, and furthermore it does not introduce too much run-time cost. But the blending is still constrained in spherical space and the quality is far from being satisfied in production point of view. Combining PSD with SSD seems a nice solution and is already widely practised in productions, however there are some artifacts that this strategy is not capable of dealing with (one case is shown in Figure 6.5). This is the motivation for the work of this chapter, in which a layered skinning framework by combining example based approach and SBS

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

is proposed. It blends rotation transforms in spherical space and consequently removes typical artifacts resulted from linear blending of SSD. The advantage of our framework over the traditional method PSD is shown by the experiment of “elbow twist” (Figure 6.5).

This work is published in [150]. The one-time setup computation involves the joint weight ω_i discussed in Section 4.4 and the pose weight W in Eq. 5.4. Other elements such

Figure 6.7: Spherical Skinning from examples: Arm; Upper row shows 4 examples; bottom row shows interpolated models from examples

as Q^p, r^p, M_i^p, d^p have to be evaluated at runtime. [84] also shows one strategy to speed up the runtime performance. Since [121] demonstrates that the real-time performance on weighted PSD by implementing the vertex transformation on GPU is possible, our method has the potential for improvement in speed. The method is implemented as a Maya plug-in which can be incorporated into the production pipeline immediately. Figure 6.6 shows the flow chart of this layered framework. Figure 6.7 gives two interpolation results of a human arm from four examples. Fig. 6.8 shows interpolation results from

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

two examples of a human leg. Fig. 6.9 shows a group of shapes of human hand. All tests are conducted on a PC with 3.2G CPU and 2G RAM and the related data are listed in the table 6.1.

Models	Arm	Leg	Hand
Verts	7276	6458	3600
Faces	7224	6440	3600
Number of joints	6	4	24
Number of examples	4	2	5
Average FPS	3-4	2-3	2-3

Table 6.1: Statistics for SBS from examples

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

Figure 6.8: Spherical Skinning from examples: Leg; Upper row shows 2 examples; bottom row shows interpolated models from examples

CHAPTER 6. SKINNING EXAMPLES UPON SPHERICAL BLENDING

Figure 6.9: Spherical Skinning from examples: Hand; Upper row shows 5 examples; bottom row shows interpolated models from examples

Chapter 7

Skinning on Progressive Decimated Examples

In this chapter, augmented deformation sensitive decimation (ADSD) is proposed to represent a set of example meshes. This method provides progressive multiresolution forms for examples, which allow scalable animations and maintain the connectivity of all examples during arbitrary sequences of decimating operations. Meanwhile, compared with deformation sensitive decimation (DSD), ADSD respects the large deformations in examples, which presents a more balanced decimation throughout all examples. Skinning existing animation frames or examples is exploited actively to find the most suitable scheme that is capable of capturing deformations from given mesh sequences. It is necessary to approximate joint transformations using a fitting algorithm that usually involves solving a large scale linear system. ADSD can significantly decrease the dimension of this system. On the other hand, the results of the previous chapters are used with this multiresolution form, to build a more general layered framework to produce smooth and scalable animations.

7.1 Introduction

More and more animation skinning schemes seek the help of example models to obtain the best visual quality, either through interpolations on the fly [96, 132] or by approximating

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

associated parameters [107, 143] that may be the most representative to given models. Examples can be obtained by artistic sculpting, 3D scanning or sampling from existing animation sequences. How to build a skinning framework which is consistent with given examples raises many problems in related fields.

The very initial examples have only geometric information such as those from 3D scanning. Registrations address the problem of building local topology (connectivities) through all examples. Constructing global topology such as pose configurations for example meshes involves many works in motion capture, in which approximating transformation parameters is an initial step. For those examples with correspondence (i.e. each vertex in one example should be associated with a particular vertex in other examples [104]) and also being paired with skeletal configurations, further refinement is needed on fitting skinning parameters (i.e. joint weights) to given examples. The above tasks are performed off-line compared to runtime interpolations, in which off-set displacements between examples and corresponding skinned geometries are evaluated at runtime.

There are extensive works on static mesh decimation, which can be applied to many tasks requiring level of details, animation compression and transferring. Recently a few works have been involved in decimating deforming meshes. In this dissertation, registration problem will not be tackled and it is assumed that examples are corresponded. Other preliminaries include a skeletal structure at the rest pose and a group of joints for each vertex indicating influence relationship. A coarse “weight-painting” (i.e., a practice to fine tune joint weights) to initialize the amount of such influences will be conducted.

Based on the above preliminary works, example meshes are first reconstructed as multiresolution models using augmented deformation sensitive decimation (ADSD). This progressive representation is able to handle large deformations while maintaining connectivity throughout all examples. Furthermore, for approximating joint transformations through fitting algorithm, decimated meshes from the above representation can be used to

improve computational performance substantially by reducing system dimensions while preserving sufficient visual quality. It will be demonstrated that joint association as well as joint weights can be propagated throughout progressive models from detailed meshes to decimated ones. Thus existing fine tuned animations can be down scaled as required for other applications directly. Finally, recovered pose parameters from Section 4.3 will be applied to perform PSD on progressive decimated examples.

7.2 Decimating Example Meshes

7.2.1 QSlim Algorithm

Garland and Heckbert introduce the QSlim algorithm by minimizing a set of quadric error metric (QEM) functions to perform simplifications on static models [61]. One function of that set is defined as a sum of squared distance between vertex v to its adjacent planes. A plane can be represented as $[n_x, n_y, n_z, -n \cdot a]'$, in which a is a point through the plane, and n is the plane normal vector. The distance function is then given as:

$$\begin{aligned} d(v) &= \sum_{p \in \text{planes}(v)} (p' \cdot v)^2 \\ &= \sum_{p \in \text{planes}(v)} v' (pp') v \\ &= v' \left(\sum_{p \in \text{planes}(v)} pp' \right) v \end{aligned}$$

The quadric associated to a vertex v can be generated as:

$$Q_v = \sum_{p \in \text{planes}(v)} pp' \quad (\text{Eq. 7.1})$$

and the distance function becomes:

$$d(v) = v' Q_v v \quad (\text{Eq. 7.2})$$

When contracting an edge formed by vertices v_1 and v_2 , an optimal vertex v_{opt} will be found out by minimizing $d(w) = w' Q_{1,2} w$, where $Q_{1,2} = Q_1 + Q_2$. $d(v_{opt})$ is contraction

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

cost of the edge formed by v_1 and v_2 . All contraction costs will be placed into a min priority queue, which gives the order of edge contractions in the process of simplification.

7.2.2 Augmented Deformation Sensitive Decimation (ADSD)

Figure 7.1: Deformation Sensitive Decimation on two cylinder examples with 70% decimations. Left: DSD; Right: Augmented DSD.

Deformation sensitive decimation (DSD) is proposed to decimate a group of examples while maintaining connectivities through all meshes [107]. Compared with QSlim for one single static mesh, DSD utilizes the same QEM scheme to find an optimal vertex for each contraction but places the sum of corresponding costs of all examples into the min priority queue. Therefore each heap key becomes $\sum_{i=1}^N d(v_{opt}^i)$, where N is the number of examples.

Instead of simply summing the cost of each example, we propose the augmented DSD (ADSD) by employing a sum of the squared costs as a heap key. Intuitively, for those parts with larger deformations (i.e. bending elbows), it is better to perform fewer contractions to maintain those deformed details. Therefore, we have heap keys as:

$$k = \sum_{i=1}^N d^2(v_{opt}^i) \quad (\text{Eq. 7.3})$$

Compared with decimating a single static mesh, either in DSD or ADSD, the priority

Figure 7.2: An example of ADSD

queue is only used to manage contraction orders. Thus we can adjust those orders as needed, such as using a weighted linear sum, in which contractions with larger weights will be placed behind those with smaller weights in the priority queue. In this dissertation, a simple sum of squares will do. By the manner of squaring, penalties employed on highly deformed parts are much more severe than those from a single linear sum. In other words, ADSD performs decimation in favor of those parts without many deformations throughout all examples, and leaves highly deformed parts with low priority to contract. Thus, deformed details will be preserved given sufficient resolutions. Figure 7.1 demonstrates how ADSD is advantageous to DSD.

7.3 Skinning on Progressive Models

In this section, with the progressive multiresolution mesh examples in place, we will exploit this representation to assist automatic updating of weight map through meshes with varied resolutions and approximating joint transformations for example meshes.

Figure 7.3: Mapping from progressive models; From left to right, edge contractions are performed progressively. Large greyed points are corresponded in mapping through multiresolution representations.

7.3.1 Mapping between Progressive Meshes

Given a progressive model, we denote two meshes S_i and S_j with resolutions as R_i and R_j respectively, where $0.0 < R_j < R_i \leq 1.0$. Our mapping scheme maintains an index list ind_list for S_j , each entry of which, say $lin_list[v]$, records the index of a point in S_i that is closest to a point v in S_j . To find the closest point in S_i , we need not loop through the whole geometry, because associated candidates from S_i can be found through a sequence of *vsplit* operations [74] and this group is much smaller than that of whole S_i . A simple illustration on index association is shown in Figure 7.3, and Figure 7.4 shows typical edge collapse (*ecol*) and vertex split (*vsplit*) operations.

Figure 7.4: vertex split. Image courtesy of Hugues Hoppe [74]

7.3.2 Automatic Joint Weight Updating

As mentioned earlier, initial pose configurations including skeletal joint structure, joint-vertex associations and joint weights are manually built. But this procedure is performed only as a one-time setup for each model at the rest pose. Either “weight painting” or solving should be performed on the original model. Thus, given the index list from the previous discussions, for any mesh with lower resolution, assigning the weight information of the original component indicated by the index list directly will suffice. Figure 7.5 demonstrates joint weight updating results for a progressive decimated horse model.

Figure 7.5: Joint Weight Mapping: (a) joint weight is painted for a mesh with resolution of 30%; (b) mesh of 5% is skinned by automatic mapping from the model on the left.

7.3.3 Fitting through Decimated Models

Approximating joint transformations is commonly a fitting procedure by minimizing least square errors between fitted geometries and given examples. As mentioned earlier, it is assumed that joint hierarchy, influence associations and weights are already given. Joint transformations are then the only unknown variables which can be solved through a linear system. This linear system is derived from a problem of minimizing least square errors, which is essentially to find best transformations that are able to obtain a mesh

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

Figure 7.6: Two example results from Augmented DSD on 11 horse poses

most closest to a given example. We build this linear system in a very similar manner to [83], in which:

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{x}$ is the unknown vector with the dimension of $12J$, and J is the number of joints. $\mathbf{b}$ is a $3M$ vector with M equal to the number of vertices of a given example. $\mathbf{A}$

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

is a $3M \times 12J$ matrix constructed from vertex weights, rest-pose vertex positions and influencing joint sets.

Figure 7.7: Fitting examples to approximate joint transformations using $R_i = 0.3$, $R_j = 0.1$ and $M_i = 2535$, $M_j = 850$; (a) decimated examples; (b) fitted examples.

Different from [83], we will not use all vertices from example geometries to build the linear system. Instead, we use a reference mesh as a facility, which is obtained from a decimated model with a specific resolution normally much lower than that of the original. If two meshes from one progressive model have resolutions as R_i and R_j ($0.0 < R_j < R_i \leq 1.0$), we denote the number of vertices of these two meshes as M_i and M_j respectively ($M_j < M_i$). An index list *ind_list* for M_i and M_j is then built as discussed in 7.3.1. To build a joint transformation for M_i , we simply sample those points listed in *ind_list*. Then the linear system has the same form but with a smaller dimension:

$$\mathbf{A}'\mathbf{x} = \mathbf{b}'$$

where $\mathbf{A}'$ becomes $3M_j \times 12J$ and $\mathbf{b}'$ is a $3M_j$ vector. Dimensions of the linear system are decimated by $3(M_i - M_j) \times 12J$. Take the horse model as an example. If $R_i = 1.0$, $R_j = 0.1$, then the space for storing $\mathbf{A}$ will be reduced by almost 20 mega bytes (assuming each entry is a float). Although we do not need such space since $\mathbf{A}$ is a sparse matrix with many zero entries, the reduced dimensions are still significant.

Figure 7.8: A flowchart of PSD on decimated examples.

7.4 Pose Space Deformation on Progressive Decimated Examples

In this section, pose space deformation (PSD) is applied on reconstructed examples to generate interpolated shapes. These new shapes in turn, can be represented in multiresolution form. In this section, we provide a framework catering to the scenario that only a few geometric example meshes are given and expressive animation is expected to be the output. The proposed ADSD (Section 7.2.2) is used for tackling the complex geometry issues and pose parameters (Section 4.3) served to build the SSD model. Finally we apply PSD on these examples to produce expressive animations.

A flowchart of our framework is shown in Fig. 7.8. The input to our system is a group of N example meshes. These example meshes comprise only vertex positions and connectivity information. The framework will output a sequence of frames as an expressive animation for the given examples.

7.5 Results and Discussions

Resolution	Verts	Faces	Fitting Time	Memory	Iterations	Residuals
10%	850	1685	5s	5.3m	616	0.017
20%	1693	3368	6s	10.56m	418	0.016
30%	2535	5052	10s	15.8m	414	0.014
40%	3378	6738	12s	21.07m	399	0.013
50%	4220	8422	15s	26.33m	376	0.015
60%	5062	10106	17s	31.68m	367	0.012
80%	6746	13474	24s	44.22m	400	0.01
100%	8431	16843	50s	52.76m	560	0.39

Table 7.1: Statistics for fitting a decimated horse model: fitting time and required memory is proportional to decimating resolutions

The main objective of the entire dissertation is about how to deal with the given examples to generate animations. Although the skeletonization method from Chapter 3 is independent of the geometries of given examples, the complexities of them still impose a high computational cost. There are many works dedicated to decimating a single mesh, but very few of them consider a group of example meshes or take into account of varying deformations presented in different examples. The target of this chapter is thus to provide a representation scheme for the proposed methods in the previous chapters. The classic decimation scheme used for a group of examples is improved to deal with character examples that usually contain very large deformations.

The work in this chapter is published in [149, 151]. The implementation is performed on a PC with Dual Core 2.66 GHz, 2G RAM and a Quadro FX 560 Graphic Card. Statistics of approximating joint transformations for a progressive horse model at one single specific pose with varied resolutions are given in Table 7.1. Note that fitting time and required memory are proportional to decimating resolutions. Fitting time referred here is the only cost for solving LSQR problem excluding other transforming and displaying operations. We adopt the same error metric for measuring fitting accuracy as

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

in [80]. 10 example poses are fitted given a reference mesh with resolutions varied from 0.9 to 0.05. The error statistics are shown in the Figure 7.9, in which we can see that using a reference mesh with a resolution as low as 0.1, we can still obtain almost the same qualities as a resolution of 0.9. But fitting time and required memory for solving the linear system are almost reduced by 90%. Figure 7.10 presents more experiments with models fitted through decimated reference meshes.

Contributions made in this chapter include 1) ADSD is proposed to represent a set of example meshes to provide a multiresolution form while maintaining the connectivities throughout the decimation. At the same time, decimation is performed in a manner that the part heavily deformed in examples will be withheld to be decimated while other part with less deformation will be decimated beforehand. 2) Decimated examples with correspondence can assist the fitting procedure for approximating transformation matrices by decreasing the dimension of the linear system. 3).By integrating PSD and ADSD, as well as the results from the previous chapters such as skeletonization, parameterization, weights augmentation, a system taking a set of example meshes to produce visually pleasing, scalable and pose consistent animation is formulated and achieved.

Table 7.5 provides statistics of experiments on pose space deformation on progressive decimated meshes. PSD on decimated mesh shows a significant gain on performance by trading off some visual qualities as shown in Fig. 7.11, Fig. 7.12 and Fig.7.13.

horse (3 poses)			camel (4 poses)		
Res.	#Face.	FPS	Res.	#Face.	FPS
10%	1684	60	10%	4382	49
40%	6738	45	40%	17526	23
80%	13474	30	80%	35052	12

Table 7.2: Runtime performance for PSD on decimated models.

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

Figure 7.9: Fitting horse model with 10 examples: error of fitting for increasing decimating resolutions. Even for resolution lowered to 0.1, the fitting quality is almost as accurate as resolution of 0.9.

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

Figure 7.10: Fitting example using decimated reference meshes. (a) Two decimated reference meshes: Lion(1000 verts) and Cat(723 verts); (b) Fitted examples with original resolutions: Lion (5000 verts) and Cat (7207 verts).

Figure 7.11: Horse: original mesh has 16843 faces. 3 example meshes are used; Interpolated Poses: Left: 1684 Faces (10%); Middle: 6738 faces (40%); Right: 13474 faces (80%).

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

Figure 7.12: Camel: original mesh has 43814 faces. 4 example meshes are used; Interpolated Poses: Left: 4382 faces (10%); Middle: 17526 faces (40%) ; Right: 35052 faces (80%).

CHAPTER 7. SKINNING ON PROGRESSIVE DECIMATED EXAMPLES

Figure 7.13: Armadillo: original model has 331904 faces. 2 examples are used; Upper row: models with 1% of resolution; Bottom row: models with 30% resolution; Left and middle columns: example poses; Right column: interpolated poses.

Chapter 8

Conclusions

Animating articulated characters to tell a story provides abundant application in education, communication, scientific research and entertainments. Current procedure of character animations requires considerable amount of labor and time from skilled and talented artists. Exploring potentials in a character shape together with a set of example meshes in varied poses to enhance the animation methodology will definitely help in producing animations efficiently. However, many available example meshes are too complex to use directly for animation production. Furthermore, most of those examples are pure geometry meshes and lack sufficient pose information, which is critical in enhancing the underlying skinning model. Encompassing the above problems, in this dissertation, a set of methods is proposed and investigated to facilitate animation production.

8.1 Contributions

Character skeletons are extracted automatically from examples using harmonic one-forms. The extracted skeletons are well centered, anatomically structured and positioned. The extraction methodology is independent of the geometries and topologies of given examples, which means it is well suited to 3D scanning where example meshes in varied poses usually present very different sampled vertices.

CHAPTER 8. CONCLUSIONS

Semantic pose parameters are recovered from example geometries to enhance the underlying skinning model and serve as interpolation variables for pose space deformation (PSD). With these parameters, joint-vertex weights from the underlying skinning model can be fitted to given example meshes. These parameters, including Euler angles and scaling values of skeletal joints, also provide a flexible interface to animators and allow them to tune key framing to produce animations as desired.

PSD is improved by introducing an inverse operator which is analyzed and demonstrated to present better visual qualities in shape interpolation. An optimization scheme is formulated providing a practical framework in real productions.

Spherical blend skinning is revised to replace with the conventional skinning model. PSD is layered upon spherical blend skinning to produce better visual qualities. This combination can capture some deformations (i.e. “elbow twist”) that are difficult to produce by simply applying PSD.

Augmented deformation sensitive decimation (ADSD) is proposed to present a set of example meshes, which is an ideal choice that detailed example meshes can be used as any desired resolution while vertices are still corresponded throughout all examples. Strict correspondences are critical to performing PSD since a vertex correspondence throughout examples are required. Besides, highly decimated examples can decrease the dimension of the linear system to solve pose transformations while accuracy is still maintained. Finally, ADSD shows nice properties on respecting large deformations in the original example meshes compared with DSD.

8.2 Future Works

An immediate direction is to optimize the recovered pose parameters by fitting the underlying skinning model to given examples. Converting from Euler angles and scalars of skeletal joints to transformation matrices is a non-linear procedure, and thus formulating

CHAPTER 8. CONCLUSIONS

and solving a non-linear optimization problem is challenging. Although local minimum exists for a non-linear problem, the recovered parameters presented in this dissertation can be used as a good initial value. Effective constraints for the parameters also need to be developed carefully. For instance, these parameters cannot be negative, joint scalars cannot be larger than corresponding bone length and Euler angles must be less than 180 degrees etc. A carefully designed optimization scheme will improve the pose parameters significantly. The aim for optimizing pose parameters is to refine the underlying skinning model. In this dissertation, joint-vertex weights are refined based on recovered pose parameters, which if optimized will definitely improve the fitted weights. The refined skinning model will decrease the disparity between deformed mesh and example mesh. Large disparities or displacements will result in vibrating effects when PSD is applied.

Figure 8.1: Multiresolution Skinning. Left: a rough decimated mesh is deformed by a skeleton through skinning method; Right: a detailed mesh is recovered by propagating deformations through the rough mesh.

Another direction to be developed is a multiresolution skinning system based on the proposed augmented deformation sensitive decimation. In this system, characters are represented by multi-layered resolutions, and deformations are propagated along these layers starting from the underlying skeleton, passing through progressive decimated meshes, and finally reaching to the detailed mesh with original resolution (See the Figure 8.1). This

CHAPTER 8. CONCLUSIONS

framework mimics the real production procedures in an inverse direction, in which a rough character model is built and skinned. That rough model is refined to produce a smooth model. The deformation produced by skeleton is transferred to the smooth model through the rough model. The rough model is generated through progressively decimating the detailed model. The deformation propagation will recover the mesh details while building from rough models in the real productions cannot produce any detail features but only bring out a smooth version of that rough model. The challenge in developing this framework is to define a relation between the decimated mesh and the original mesh, and that is how to propagate the deformations generated by skeletons. This strategy presents a valuable advantage that skinning a rough model is much easier than skinning a detailed one since the mapping from skeletons to a detailed model is difficult to design. That is the reason why a rough-to-smooth scheme is adopted in real productions. There is a related work, which builds affine transformation between a rough model to a detailed model [136], that may perform this task.

Figure 8.2: Pose Adapted Skinning. Left: a heavy bending arm demanding more vertices to display details; Right: a rest arm with fewer vertices.

Another possible direction is to design a scheme to perform resolution adapted skinning, in which a character will adapt its resolution automatically according to its pose.

CHAPTER 8. CONCLUSIONS

The pose parameters present the amount of deformation. For instance, a larger Euler angle may indicate a heavily bending elbow. In this case more vertices are required to display the deformed details (See the Figure 8.2). On the other hand, a smaller deformation demands fewer vertices. This scheme is similar to the work in [73], in which the resolution is adapted based on viewing distances and directions. However, designing such a scheme is valuable or not to applications needs to be carefully evaluated, since the workload of adjusting resolutions may be penalized at the expense of the runtime performance.

Besides the above research directions, several issues are being considered to refine our framework.

- **Refine the Mapping Scheme.** Skinning is a procedure that maps surface vertices to skeletal joints. In this dissertation, only joint weights are fitted which actually measure the “how much” in the mapping. Another critical issue is “which should be mapped to which?”. This problem is usually addressed in segmentation methodology. In this dissertation, a simple proximity is adopted as the criterion to assign vertices to associated joints. This criterion works in most situations, but fails sometimes and results in a wrong assignment that some vertices are assigned to a joint which is spatially near but hierarchically far to those vertices (illustrated in Figure 4.5). Adopting geometric proximity may improve the mapping significantly like in [99], which is our ongoing works.
- **Hardware Implementations.** GPU implementations on SSD will improve the performance of the entire pipeline. The implementations of SBS [13] and PSD [121] using GPU, which should be realized for a practical system ready for real productions are also reported recently.

CHAPTER 8. CONCLUSIONS

- **Weighted PSD.** [121, 90] produce nicer deformation results compared with traditional PSD at some expense of efficiency. In the original SBS method, rotation parts are extracted from transformation matrices, and then converted to quaternions that are further linearly blended by joint weights. We believe this operation can provide another suitable pose space for deformations, since the weight blending performed in WPSD has been implemented in the similar manner. We can also incorporate this feature without much implementation efforts.
- **Analysis on Spherical Space.** [84] still builds SBS based on the scheme of SSD, leaving some potential features of spherical space unexplored. Further, joint weights are also solved by assuming SSD is the underlying deformation scheme. How to fully explore the spherical nature of transformation blending is our ongoing works.

Appendix A

Implementation

Detailed description on implementation and system design is discussed in this Appendix. Maya API is chosen as the implementation tool and a set of plug-ins are delivered as the prototype for this dissertation. In this chapter, an overview on Maya API is first presented with the reasons why it is an ideal tool for delivering animation prototypes. What follows is the specific description on implementation details for the works of this dissertation. In the end, an integrated prototype and work flow issues are presented.

A.1 Software and Libraries

A.1.1 Maya Architecture and Programmability

Maya is a powerful tool for modeling and animation, and is widely used in areas which need to produce stunning visual effects. Besides being a content generating tool, it is also designed as an open, flexible and extensible product¹. The core part of Maya architecture is a very efficient database which is called **Dependency Graph** (DG). Information in this database is organized in structures called **nodes**, configurable properties of which can be accessed through **attributes**. Attributes with similar types can be connected, and Maya will handle the data flow among different nodes. Upon the DG core, a highly customizable user interface is built using **Maya Embedded Language** (MEL), which

¹Autodesk Maya API — White Paper

CHAPTER A. IMPLEMENTATION

Figure A.1: Maya Interfaces

can be used to automate and integrate work flow or to implement some algorithms without critical requirements on runtime performance. For very complex algorithms, implementations with DG level through C++ are required. Figure A.1 presents a set of interfaces that are quite frequently referred when a developing task is involved. 3D objects are usually displayed in a perspective view panel (Area 1). Area 2 is the Plug-in Manager which links plug-ins (.mll file in windows and .so file in linux) to Maya. Area 3 is the Script Editor, in which MEL scripts can be edited and executed. Area 4 is the window called Hypergraph that helps users to investigate the structures and connections of dependent nodes.

Maya provides a set of generic classes with different functions in DG level. For instance, “MPxCommand” is to create a command which can be executed in Maya command line. A deformation algorithm working on a mesh or NURBS surface can be implemented by instanting the class “MPxDeformerNode”. Customized object shape can

CHAPTER A. IMPLEMENTATION

be obtained by instantiating “MPxSurfaceShape”, etc. If a new type of node is implemented and inserted into DG, extra cares must be taken since it only works with other nodes. The connection and attribute relationships determine when and how to evaluate these nodes. In other words, the input and output of the customized node have to be carefully designed and implemented.

Attributes are presented as “MObject” in Maya API, and usually defined as static members of an instanced class. Input and output attributes are implicitly built through “attributeAffects” functions, which is a built-in function in “MPxNode”. Another built-in function “compute”, taking an input to calculate an output, is the core of a node. The data-flow is then set up within the customized node. When the output attribute is explicitly required in DG, the node will be evaluated exactly as executing the function “compute”. More interesting details and materials can be found in [56, 57].

Maya API provides an interface of the fundamental building blocks of arbitrary 3D tasks, and therefore it is very flexible to implement algorithms. Furthermore, Maya itself provides good data structures for 3D shape representations (NURBS surface, subdivision surfaces, meshes ect.), related functions (interpolations, intersections etc.) and powerful features (key-framing, vertex editing, rendering etc.). Thus graphics related ideas or algorithms can be quickly implemented with convincing demonstrations.

A.1.2 Dependent Libraries

Besides Maya API, several libraries are adopted to provide desirable features such as generic data structures and robust numerical implementations.

- **STL/BOOST** Standard Template Library (STL) is heavily used for this dissertation including generic containers (vector, list and map etc.) and related algorithms. A class with an assignment operator function is valid for STL containers. Almost

CHAPTER A. IMPLEMENTATION

all objects in Maya API have built-in assignment operators and therefore applicable to STL containers. Smart pointers in BOOST are also adopted to deal with memory issues which are quite common in C++ programs.

- **TNT/JAMA** The Template Numerical Toolkit (TNT) is a collection of interfaces and implementations of numerical objects useful for scientific computing in C++². In this dissertation, 2D vector from TNT is used as a matrix representation. Several linear operations such as singular value decomposition (SVD) etc. are also provided by TNT. Another advantage is that TNT does not require to be compiled and installed, and including a set of header files is all that is needed to get TNT/JAMA work.
- **QSLIM** QSlim³, a mesh simplification method, is proposed and implemented by Michael Garland [61]. Basically, for this dissertation, the heap structures of a polygon mesh and the quadratic structure for each vertex with adjacent planes are borrowed for progressive skinning.
- **Optimization** An optimization class which provides “Powell” and “Conjugate Gradient” solvers is implemented. This class uses the vector structures provided by TNT and produces the results that have data structures consistent with Maya API.
- **RBF Interpolation** RBF solving procedures are wrapped into a single class, which mainly involves computing the Eulerian distances of different poses and a linear matrix solver for obtaining weights from key poses.

²<http://math.nist.gov/tnt/index.html>

³<http://mgarland.org/software/qslim.html>

A.2 Implementation Details

A.2.1 Skeleton Subspace Deformation

In Maya, SSD is a built-in function with the name “smooth skinning”. To use it, simply provide an object (NURBS surface or mesh) with a set of structured skeleton joints. Maya will deal with how to apply SSD for them through “smooth skinning”. However Maya only provides a default setting, and users need to tweak some parameters or even assign vertex-joint weight manually (through a tool called “weight painter”) to obtain better skinning effects. In this dissertation, SSD has to be explicitly implemented since spherical blend skinning is implemented based on SSD. A highly simplified pseudocode of algorithm SSD is presented in Algorithm A.2.1. It takes a new “pose” of a skeleton as an input and computes the positions of associated mesh vertices (or NURBS CVs). The calculation is a linear blending of the transformations from a set of skeletal joints. The core of SSD is a recursive method that computes each joint transformation through the skeletal structure (Algorithm A.2.2).

Algorithm A.2.1: SSD(*pose*)

comment: $jointSet_i$: a set of joints that has influences to $vertex_i$
comment: All value of joints are given from input pose
comment: w_{ij} : weight between $joint_j$ and $vertex_i$
for each $vertex_i \in mesh$
 do for each $joint_j \in jointSet_i$
 do $\begin{cases} m_{ij} \leftarrow TRANSFORMATION(j) \\ vertex_i \leftarrow vertex_i + w_{ij} * m_{ij} \end{cases}$

Algorithm A.2.2: TRANSFORMATION(*j*)

if $Parent(j) = \text{true}$
 then $m* = TRANSFORMATION(parent)$
 else return (*m*)

CHAPTER A. IMPLEMENTATION

It is straightforward to implement SSD on GPU since only the vertex transformation and blending are involved.

A.2.2 Example Skeletons by Optimization

Given a skeleton structure and a set of example meshes, finding the pose information for each example is to solve an optimization problem that is explained in Chapter 4. The core method is outlined in Algorithm A.2.3, which is to set up an objective function that is further minimized by Conjugate Gradient methods. The value of the variable x of which the function value is minimized provides the optimal position of a joint in example pose.

Algorithm A.2.3: OBJECTIVEFUNCTION(Sr, Se, Jr, x)

comment: Jr: a joint in the rest position

comment: Sr: a set of vertices influenced by Jr in the rest position

comment: Se: a same set as Sr but in an example position

comment: x: function variable (joint position in example pose)

$d \leftarrow 0$

for each $vr_i \in Sr, ve_i \in Se$

do $d \leftarrow d + (vr_i - Jr)^2 - (ve_i - x)^2$

return $(d)^2$

A.2.3 Paramerizaiton of Poses

The results for skeletal joints from the Algorithm A.2.3 only give the absolute 3D positions in world space. We still need to parameterize the pose for example meshes. A pose, from the implementation point of view, is a tree structure, each node of which represents a joint. Given a character with a skeleton, the structure of its pose should remain unchanged. While in an articulate motion, a new (or different) pose means that the skeletal structure is having a specific configuration of nodes. In other words, a pose is an

CHAPTER A. IMPLEMENTATION

instance of tree-structure with specific node values. Here the values could be anything that is capable of describing the relative motions in the skeletal structure. A 4×4 transformation matrix is widely used [80, 106]. However, a matrix is not intuitive for animators especially when a pose is exposed to be tuned or tweaked. Chapter 4 introduces the method on using rotation and scaling factors to parameterize joint nodes. This section presents the implementation details, which are relatively simple since required data structures and functions are provided by Maya API. The following data types are heavily used:

Maya API:

MQuaternion

MVector

MEulerRotation

Besides the data structures, essential operations related to them are also provided by the API. Consider a bone rotation described in Chapter 4 (See Figure. 4.8), to obtain the Euler rotations, just apply the following code:

Code List:

```
MVector v1, v2;
MQuaternion q = v1.rotateTo( v2 );
MEulerRotation e = q.asEulerRotation();
```

Through **MEulerRotation**, we can get euler rotations in radian directly, specify the rotation order, get rotation matrix or bound the rotation angle by indicating constraints.

A.2.4 Spheric Blend Skinning

The core methods in Spheric Blend Skinning are outlined in Algorithm A.2.4. Several routines are required such as computing the center of a set of joints, converting a matrix to a quaternion or vice versa. These details are discussed in Chapter 6. From the

CHAPTER A. IMPLEMENTATION

Algorithm A.2.4, we can see that SBS bears a similar structure with SSD, differences are rotation centers are calculated every frame and the transformation matrices are decomposed to rotation part and converted to quaternions. So SBS presents comparable runtime performance to SSD.

Algorithm A.2.4: SBS(*pose*)

comment: $jointSet_i$: a set of joints that has influences to $vertex_i$
comment: All value of joints are given from input pose
comment: w_{ij} : weight between $joint_j$ and $vertex_i$
for each $joint_j \in jointSet_i$
 do $c_j \leftarrow RotationCenter(joint_j)$
for each $vertex_i \in mesh$
 do $\left\{ \begin{array}{l} \text{for each } joint_j \in jointSet_i \\ \quad \text{do } \left\{ \begin{array}{l} m_{ij} \leftarrow TRANSFORMATION(j) \\ q \leftarrow q + w_{ij} * MATRIX2QUATERNION(m_{ij}) \\ M \leftarrow M + w_{ij} * m_{ij} \end{array} \right. \\ Q = QUATERNION2MATRIX(q) \\ vertex_i \leftarrow Q * (vertex_i - c_j) + M * vertex_i \end{array} \right.$

A.2.5 Inverse PSD

The idea of Inverse PSD is to calculate delta values in the rest pose before applying the general skinning method (SSD in this dissertation). Algorithm A.2.5 generally presents this idea. The pipeline is almost the same as the Algorithm A.2.1, except that each vertex is added by a delta value before blended transformation. The delta value is computed by Algorithms A.2.6 and A.2.7, while Algorithm A.2.7 is one time setup once examples are given and Algorithm A.2.6 is called in runtime during the animation. In Algorithm A.2.7, the delta value for each example pose is computed by an optimization method which is discussed in Chapter 5.

CHAPTER A. IMPLEMENTATION

Algorithm A.2.5: INVERSEPSD()**comment:** $jointSet_i$: a set of joints that has influences to $vertex_i$ **comment:** w_{ij} : weight between $joint_j$ and $vertex_i$ **for each** $vertex_i \in mesh$

$$\begin{cases} \delta_i \leftarrow RBFINTERPOLATE(pose) \\ vertex_i \leftarrow vertex_i + \delta_i \end{cases}$$
do $\begin{cases} \textbf{for each } joint_j \in jointSet_i \\ \textbf{do } \begin{cases} m_{ij} \leftarrow TRANSFORMATION(j) \\ vertex_i \leftarrow vertex_i + w_{ij} * m_{ij} \end{cases} \end{cases}$
Algorithm A.2.6: RBFINTERPOLATE($pose$) $\Omega = RBFCORE(examples)$ **for each** $e_i \in examples, \omega_i \in \Omega$ **do** $\begin{cases} d_i \leftarrow Gaussian(pose, e_i) \\ \delta \leftarrow \delta + \omega_i * d_i \end{cases}$ **return** (δ)**Algorithm A.2.7:** RBFCORE($examples$)**comment:** N : number of examples**comment:** M : a $N \times N$ matrix**comment:** $m_{i,j}$: a component in M **comment:** e_i : an example including mesh, and pose**comment:** Ω : PSD pose weight**for each** $e_i \in examples$ **do** $\delta_i = OPTIMIZE(e_i)$ **for each** $e_i \in examples$ **do** $\begin{cases} \textbf{for each } e_j \in examples \\ \textbf{do } m_{i,j} \leftarrow Gaussian(e_i, e_j) \end{cases}$ $\Omega = M^{-1} * d$ **return** (Ω)

Implementing the entire work flow of inverse PSD as a plug-in is challenging. One reason is that although Maya API does provide a generic class (MPxDeformerNode)

CHAPTER A. IMPLEMENTATION

Figure A.2: Dependent Graph Node for Inverse PSD

to implement deformation methods on meshes, the way that the deformation works is different from our proposed method. The deformation is supposed to be applied **after** SSD, while in our case, we must add the deformation (delta values) **before** SSD starts to work. Therefore to implement inverse SSD, **MPxDeformerNode** is not adopted instead another more generic class **MPxNode** is instanced. The instanced class which is supposed to deal with adding delta values before SSD is called “tweakManager” (see Figure A.2). The reason for this strategy is that Maya provides a tweaking function that is able to adjust the mesh geometries in the rest pose.

A.2.6 Skinning on Progressive Decimating Examples

Algorithm A.2.8 outlines the core method of ADSD, which takes the sum of squared quadratics of each example vertex as the heap key. The detailed discussions are presented in Chapter 7. On the performance side, the plug-in will take a few seconds (or minutes for a set of very detailed meshes) to calculate the heap structures, while the toggle among different resolutions is relatively faster.

CHAPTER A. IMPLEMENTATION

Algorithm A.2.8: ADSD(*examples*)**comment:** e_i : an example including mesh, and pose**for each** $e_i \in \text{examples}$

do { **for each** $\text{vertex}_j \in \text{meshe}_i$

do { $q_{ij} \leftarrow \text{QUADRATICS}(\text{vertex}_j, e_i)$

$\text{HeapKey}_j \leftarrow \text{HeapKey}_j + q_{ij}^2$

$\text{Heapify}(e_i)$

A.3 System Integration

Figure A.3: A set of plug-in files

The prototypes for this dissertation are delivered in form of a set of plug-ins (shown in Figure A.3), which are several “.mll” files working simultaneously under Maya. Separating the modules as different plug-in files maintains the independency of each proposed

CHAPTER A. IMPLEMENTATION

Figure A.4: User Interface of Character Skinning from Examples

method and thus all of them are flexible to be changed and improved, while integrating them into a unified system is a matter of system design or a taste for a specific work flow. Given those plug-in files, users can work with them by executing commands from Script Editor or building DG relations through Hypergraph window. However, a user friendly interface is more convenient for practical work flow. For this dissertation, one interface is implemented using MEL scripts to integrate these plug-ins (see Figure A.4). And a typical working scene is presented in Figure A.5.

CHAPTER A. IMPLEMENTATION

Figure A.5: A Working Scene under Maya

Through the interface (shown in Figure A.4), users can import a set of examples, apply ADSD progressive decimation, and display them with a specific resolution. The poses for different examples can be set up and parameterized, and then inverse PSD can work on them to generate smooth animation. The underlying skinning can be chosen as SSD or SBS, which presents a better quality with little lost of runtime performance. Other built-in functions from Maya can help to generate a better demonstration. For instance after a set of key poses are built, it is straightforward to apply key-frame animation and there is no need to implement another interpolation scheme to generate in-between frames, which can be played back directly within Maya. Some artifacts can be detected immediately. The animated sequences can be rendered as high quality images to produce a complete video clip.

References

- [1] http://en.wikipedia.org/wiki/Singular_value_decomposition.
- [2] <http://usa.autodesk.com/adsk/servlet/index?id=7635018&siteID=123112>.
- [3] <http://www.softimage.com/>.
- [4] Meshless thin-shell simulation based on global conformal parameterization. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):375–385, 2006. X.H. Guo and X. Li and Y.F. Bao and X.F. Gu and H. Qin.
- [5] Course 11: Practical least-squares for computer graphics, 2007.
- [6] Grégoire A., Franck H., L. Francis, and D. Christine. Harmonic skeleton for realistic character animation. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*. ACM-Siggraph/Eurographics, 2007.
- [7] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [8] P. Alfeld. Scattered data interpolation in three or more variables. pages 1–33, 1989.
- [9] B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. *ACM Trans. Graph.*, 21(3):612–619, 2002.
- [10] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 587–594, New York, NY, USA, 2003. ACM.

REFERENCES

- [11] N. Amenta, S. Choi, and R. K. Kolluri. The Power Crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–260, 2001.
- [12] N. Amenta, S. Choi, and R. K. Kolluri. The Power Crust, Unions of Balls, and the Medial Axis Transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.
- [13] K.A. Andersen. Spherical blend skinning on gpu. Technical report, University of Copenhagen, 2007.
- [14] A. Angelidis and K. Singh. Space deformations and their application to shape modeling. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 10–29, New York, NY, USA, 2006. ACM.
- [15] D. Anguelov, D. Koller, H.C. Pang, P. Srinivasan, and S. Thrun. Recovering articulated object models from 3d range data. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 18–26, Arlington, Virginia, United States, 2004. AUAI Press.
- [16] N. Arad, N. Dyn, D. Reissfeld, and Y. Yeshurun. Image warping by radial basis functions: applications to facial expressions. *CVGIP: Graph. Models Image Process.*, 56(2):161–172, 1994.
- [17] O.K.C. Au, H.B. Fu, C.L. Tai, and D. Cohen-Or. Handle-aware isolines for scalable shape editing. *ACM Trans. Graph.*, 26(3):83, 2007.
- [18] 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*, 27(3), 2008.
- [19] A. Aubel and D. Thalmann. Realistic deformation of human body shapes. In *Proceedings of Computer Animation and simulation*, pages 125–135, 2000.
- [20] A. Aubel and D. Thalmann. Interactive modeling of the human musculature. In *Proceedings of Computer Animation, 2001. The Fourteenth Conference on Computer Animation*, pages 167–255, Seoul, South Korea, 2001.

REFERENCES

-
- [21] C. L. Bajaj, J. Chen, and G.L. Xu. Modeling with cubic a-patches. *ACM Trans. Graph.*, 14(2):103–133, 1995.
 - [22] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.
 - [23] E. Bardinet, L. Cohen, and N. Ayache. Tracking and motion analysis of the left ventricle with deformable superquadrics. *Medical Image Analysis*, 1(2):129–149, 1996.
 - [24] A. H. Barr. Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph.*, 18(3):21–30, 1984.
 - [25] E. B.Dam, M. Koch, and M. Lillholm. Quaternions, interpolation and animation. Technical report, University of Copenhagen, 1998.
 - [26] T. Beier and S. Neely. Feature-based image metamorphosis. *SIGGRAPH Comput. Graph.*, 26(2):35–42, 1992.
 - [27] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended reeb graphs for surface understanding and description. In *DGCI '00: Proceedings of the 9th International Conference on Discrete Geometry for Computer Imagery*, pages 185–197, London, UK, 2000. Springer-Verlag.
 - [28] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Shape abstraction using computational topology techniques. pages 209–222, 2002.
 - [29] S. Biasotti, S. Marini, M. Mortara, and G. Patané. An overview on properties and efficacy of topological skeletons in shape modelling. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 245, Washington, DC, USA, 2003. IEEE Computer Society.
 - [30] R. M. Bolle and B. C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):1–13, 1991.

REFERENCES

- [31] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff, and C. Rössl. Geometric modeling based on polygonal meshes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 1, New York, NY, USA, 2007. ACM.
- [32] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.
- [33] G. Bradshaw and C. O’Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–40, New York, NY, USA, 2002. ACM.
- [34] J. F. Brinkley. Knowledge-driven ultrasonic three-dimensional organ modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):431–441, 1985.
- [35] M.D. Carmo. *Differential Geometry of Curves and Surfaces*.
- [36] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM Press.
- [37] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1989. ACM Press.
- [38] D. T. Chen and D. Zeltzer. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 89–98, New York, NY, USA, 1992. ACM.
- [39] Q. Chen. *Computer-Assisted Inbetween Generation for Cel Animation*. PhD thesis, Nanyang Technological University, 2007.

REFERENCES

- [40] Q. Chen, F. Tian, H.S. Seah, Z.K. Wu, J. Qiu, and M. Konstantin. Dbsc-based animation enhanced with feature and motion. *Comput. Animat. Virtual Worlds*, 17(3-4):189–198, 2006.
- [41] J. Choi. *Maya Character Animation*. Sybex Books, 2002.
- [42] C. Chua and U. Neumann. Hardware-accelerated free-form deformation. In *HWWS '00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 33–39, New York, NY, USA, 2000. ACM.
- [43] I. Cohen and L. D. Cohen. A hybrid hperquadric model for 2-d and 3-d data fitting. *Computer Vision and Image Understanding*, 63(3):527–541, 1996.
- [44] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. *Discrete Comput. Geom.*, 32(2):231–244, 2004.
- [45] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1990. ACM.
- [46] S. Coquillart and P. Jancne. Animated free-form deformation: an interactive animation technique. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 23–26, New York, NY, USA, 1991. ACM.
- [47] N. D. Cornea and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007. Member-Deborah Silver.
- [48] N.D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21(11):945–955, 2005.
- [49] C.Theobalt, E. de Aguiar, M. A. Magnor, H. Theisel, and H.P. Seidel. Marker-free kinematic skeleton estimation from sequences of volume data. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 57–64, New York, NY, USA, 2004. ACM.

REFERENCES

- [50] M. de G. Malheiros and S.T. Wu. A hierarchical skeleton-based implicit model. *Computer Graphics and Image Processing*, pages 65–70, 1997.
- [51] C. DeCoro and S. Rusinkiewicz. Pose-independent simplification of articulated meshes. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 17–24, New York, NY, USA, 2005. ACM.
- [52] H. Delingette. Simplex meshes: a general representaion for 3d shape reconstruction. In *Conference on Computer Vision and Pattern Recognition*, pages 856–859, 1994.
- [53] F. Dellas, L. Mocozet, N. Magnenat-Thalmann, M. Mortara, G. Patan"e, M. Spagnuolo, and B. Falcidieno. Knowledge-based extraction of control skeletons for animation. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 51–60, Washington, DC, USA, 2007. IEEE Computer Society.
- [54] K. G. Der, R. W. Sumner, and J. Popović. Inverse kinematics for reduced deformable models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1174–1179, New York, NY, USA, 2006. ACM.
- [55] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM.
- [56] D.Gould. *Complete Maya Programming Volume I: An Extensive Guide to MEL and the C++ API*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
- [57] D.Gould. *Complete Maya Programming Volume II: An In-depth Guide to 3D Fundamentals, Geometry, and Modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
- [58] J. Diebel. Representing attitude: Euler angles, quaternions, and rotation vectors. Technical report, Stanford University, 2006.

REFERENCES

-
- [59] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des.*, 22(5):392–423, 2005.
 - [60] S. Forstmann, J.O., A. Krohn-Grimberghe, and R. McDougall. Deformation styles for spline-based skeletal animation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–150, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
 - [61] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
 - [62] A. Goshtasby. Surface reconstruction from scattered measurements. In J. D. Warren, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 1830 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 247–256, November 1992.
 - [63] J. GREISSMAIR and W. PURGATHOFER. Deformation of solids with trivariate b-splines. In *EuroGraphics 89*, pages 137–148, 1989.
 - [64] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.
 - [65] X.F. Gu and S.T. Yau. Global conformal surface parameterization. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 127–137, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
 - [66] X.F. Gu, Y.He, and H. Qin. Manifold splines. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 27–38, New York, NY, USA, 2005. ACM.
 - [67] K. B. S. Han and D. B. Goldgof. Using hyperquadrics for shape recovery from range data. In *International Conference on Computer Vision*, pages 492–496, Berlin, Germany, May 1993.

REFERENCES

-
- [68] Y. He, X. Xiao, and H.S. Seah. Example based skeletonization using harmonic one-forms. In *IEEE International Conference on Shape Modeling and Applications, 2008. SMI 2008.*, pages 53–61, Stony Brook, USA, 2008.
 - [69] Y. He, X.Xiao, and H.S. Seah. Harmonic 1-form based skeleton extraction from examples. *Graph. Models*, 71(2):49–62, 2009.
 - [70] D. Hearn and M. P. Baker. *Computer Graphics*. Prentice Hall, 1998.
 - [71] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM.
 - [72] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*, page 260. American Mathematical Society.
 - [73] H. Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
 - [74] H. Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.
 - [75] H. Hoppe, T. DeRose, and T. Duchamp. Surface Reconstruction from Unorganized Points. *Computer Graphics*, 26:71–78, July 1992.
 - [76] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302, July 1994.
 - [77] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 177–184, New York, NY, USA, 1992. ACM.

REFERENCES

- [78] F.C. Huang, B.Y. Chen, and Y.Y. Chuang. Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 53–62, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [79] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, 1996.
- [80] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, 2005.
- [81] O. Johnston and F. Thomas. *The Illusion of Life: Disney Animation*. Disney Editions, 1995.
- [82] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, New York, NY, USA, 2003. ACM.
- [83] L. Kavan, R. McDonnell, S. Dobbryn, J. Vol. rí Vol. Zára, and C. O'Sullivan. Skinning arbitrary deformations. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 53–60, New York, NY, USA, 2007. ACM.
- [84] L. Kavan and J. Zara. Spherical blend skinning: a real-time deformation of articulated models. In *I3D 2005: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, New York, NY, USA, 2005. ACM Press.
- [85] D. G. Kendall. Shape manifolds, procrustean metrics and complex projective spaces. *Bull. London Math. Soc.*, 16:81–121, 1984.
- [86] S. Kircher and M. Garland. Progressive multiresolution meshes for deforming surfaces. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 191–200, New York, NY, USA, 2005. ACM.
- [87] A. Kirk, J. F. O'Brien, and D. A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 29, New York, NY, USA, 2004. ACM.

REFERENCES

- [88] P.G. Kry, D.L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–159, New York, NY, USA, 2002. ACM Press.
- [89] J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1998.
- [90] T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 355–363, New York, NY, USA, 2004. ACM Press.
- [91] J. Lasseter. Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 35–44, New York, NY, USA, 1987. ACM.
- [92] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Society for Industrial Mathematics, 1987.
- [93] F. Lazarus, S. Coquillart, and P. Jancne. Axial deformations: an intuitive deformation technique. *Computer-Aided Design*, 26(8):607–613, 1994.
- [94] T.Y. Lee, C.H. Lin, H.K. Chu, Y.S. Wang, S.W. Yen, and C.R. Tsai. Mesh pose-editing using examples. *Comput. Animat. Virtual Worlds*, 18(4-5):235–245, 2007.
- [95] T.Y. Lee, Y.S. Wang, and T.G. Chen. Segmenting a deforming mesh into near-rigid components. *Vis. Comput.*, 22(9):729–739, 2006.
- [96] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

REFERENCES

-
- [97] T.Y. Li, J.M. Lien, S.Y. Chiu, and T.H. Yu. Automatically generating virtual guided tours. In *CA '99: Proceedings of the Computer Animation*, page 99, Washington, DC, USA, 1999. IEEE Computer Society.
 - [98] J.M. Lien, J. Keyser, and N. M. Amato. Simultaneous shape decomposition and skeletonization. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 219–228, New York, NY, USA, 2006. ACM.
 - [99] L. Lin, Franck Hetroy, Cédric Géro, and Boris Thibert. Atlas-based character skinning with automatic mesh decomposition. Research Report 6406, INRIA, jan 2008.
 - [100] Y. Lipman, D. Cohen-Or, R. Gal, and D. Levin. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph.*, 26(1):5, 2007.
 - [101] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35, 2001.
 - [102] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH 96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1996. ACM Press.
 - [103] M.Desbrun, E. Kanso, and YY Tong. Discrete differential forms for computational modeling. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–17, New York, NY, USA, 2008. ACM.
 - [104] B. Merry, P. Marais, and J. Gain. Animation space: A truly linear framework for character animation. *ACM Trans. Graph.*, 25(4):1400–1423, 2006.
 - [105] T. Milliron, R. J. Jensen, R. Barzel, and A. Finkelstein. A framework for geometric warps and deformations. *ACM Trans. Graph.*, 21(1):20–51, 2002.
 - [106] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22(3):562–568, 2003.

REFERENCES

- [107] A. Mohr and M. Gleicher. Deformation sensitive decimation. Technical report, University of Wisconsin, 2003.
- [108] M. Mortara and G. Patané. Affine-invariant skeleton of 3d shapes. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, page 245, Washington, DC, USA, 2002. IEEE Computer Society.
- [109] M. E. Mortensen. *Geometric Modelling*. Wiley, 1997.
- [110] X.L. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 613–622, New York, NY, USA, 2004. ACM.
- [111] J. F. O'Brien, R. E. Bodenheimer, G. J. Brostow, and J. K. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface 2000*, pages 53–60, May 2000.
- [112] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. Interactive skeleton extraction for 3d animation using geodesic distances. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters*, page 9, New York, NY, USA, 2006. ACM.
- [113] V. Pascucci, G. Scorzelli, P.T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.
- [114] U. Pinkall, S. D. Juni, and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.
- [115] P. Merrell. Example-based model synthesis. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 105–112, New York, NY, USA, 2007. ACM.
- [116] E. Pojar and D. Schmalstieg. User-controlled creation of multiresolution meshes. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 127–130, New York, NY, USA, 2003. ACM.

REFERENCES

- [117] N. S. Pollard and V. B. Zordan. Physically based grasping control from example. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 311–318, New York, NY, USA, 2005. ACM.
- [118] M. J. D. Powell. *Radial basis functions for multivariable interpolation: a review*. Clarendon Press, New York, NY, USA, 1987.
- [119] M. Pratscher, P. Coleman, J. Laszlo, and K. Singh. Outside-in anatomy based character rigging. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 329–338, New York, NY, USA, 2005. ACM.
- [120] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C++: the art of scientific computing*. 2002.
- [121] T. Rhee, J.P. Lewis, and U. Neumann. Real-time weighted pose-space deformation on the gpu. In *Computer Graphics Forum (Eurographics 2006)*, volume 25, 2006.
- [122] S. Schaefer and C. Yuksel. Example-based skeleton extraction. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 153–162, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
- [123] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 163–172, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [124] F. Schmitt, B. A. Barsky, and W. Du. An adaptive subdivision method for surface-fitting from sampled data. *Computer Graphics (SIGGRAPH 86 Proceedings)*, 20(4):179–188, 1986.
- [125] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.
- [126] T. W. Sederberg, J.M. Zheng, A. Bakenov, and A. Nasri. T-splines and t-nurccs. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 477–484, New York, NY, USA, 2003. ACM.

REFERENCES

-
- [127] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Shape description by medial surface construction. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):62–72, 1996.
 - [128] Y. Shinagawa and T. L. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Comput. Graph. Appl.*, 11(6):44–51, 1991.
 - [129] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.
 - [130] M.C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and global skeleton fitting techniques for optical motion capture. In *CAPTECH '98: Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, London, UK, 1998. Springer-Verlag.
 - [131] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *SIGGRAPH 98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414, New York, NY, USA, 1998. ACM Press.
 - [132] P. J. Sloan, III C.F. Rose, and M.F. Cohen. Shape by example. In *SI3D 2001: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM Press.
 - [133] M. Spivak. *A Comprehensive Introduction to Differential Geometry*.
 - [134] R. W. Sumner. *Mesh Modification Using Deformation Gradients*. PhD thesis, Massachusetts Institute of Technology, 2005.
 - [135] R. W. Sumner and J Popović. Deformation transfer for triangle meshes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 399–405, New York, NY, USA, 2004. ACM.
 - [136] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80, 2007.
 - [137] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.

REFERENCES

- [138] G. Taubin. Distance approximation for rasterizing implicit curves. *ACM Transactions on Graphics*, 13(1):3–42, 1994.
- [139] G. Taubin, C. Fernando, S. Sullivan, J. Ponce, and D. J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):287–303, 1994.
- [140] M. Teichmann and S. Teller. Assisted articulation of closed polygonal models. In *SIGGRAPH '98: ACM SIGGRAPH 98 Conference abstracts and applications*, page 254, New York, NY, USA, 1998. ACM.
- [141] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [142] R. Y. Wang, K. Pulli, and J. Popović. Real-time enveloping with rotational regression. *ACM Trans. Graph.*, 26(3):73, 2007.
- [143] X. C. Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 129–138, New York, NY, USA, 2002. ACM Press.
- [144] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3), 2007.
- [145] H. Whitaker and J. Halas. *Timing for Animation*. Focal Press, 2002.
- [146] J. Wilhelms and A. V. Gelder. Anatomically based modeling. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 173–180, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [147] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004.

REFERENCES

- [148] X. Xiao, J.P. Lewis, H.S. Seah, N. Fong, and F. Tian. A powell optimization approach for example-based skinning in a production animation environment. In *CASA 2006, Computer Animation and Social Agents*, pages 141–150, 2006.
- [149] X.Xiao and H.S. Seah amd F.Tian. Skinning on progressive decimated models. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 1054–1058, Las Vegas, NV, USA, 2008.
- [150] X.Xiao, H.S. Seah, and F.Tian. Spherical skinning from examples. In *International Workshop on Advanced Image Technology, 2007 (IWAIT 2007)*, Bangkok, Thailand, 2007.
- [151] X.Xiao, H.S. Seah, F. Tian, and Y. He. Pose space deformation on progressive decimated example meshes. In *Computer Graphics International, 2008. CGI 2008.*, pages 53–61, Istanbul, Turkey, 2008.
- [152] X.S. Yang, A. Somasekharan, and J. J. Zhang. Curve skeleton skinning for human and creature characters: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):281–292, 2006.
- [153] S. Yoshizawa, A. Belyaev, and H.P. Seidel. Skeleton-based variational mesh deformations. *Computer Graphics Forum*, 26(3):255–264, September 2007.
- [154] P. Yushkevich, P. Thomas Fletcher, S. Joshi, A. Thall, and S. M. Pizer. Continuous medial representations for geometric object modelling in 2d and 3d. *Image and Vision Computing*, 21:17–27, 2003.
- [155] R. Zayer, C. Rossl, Z. Karni, and H.P. Seidel. Harmonic guidance for surface deformation. *Computer Graphics Forum*, 24(3):Computer Graphics Forum, 2005.
- [156] V. B. Zordan and N. C. Van Der Horst. Mapping optical motion capture data to skeletal motion using a physical model. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 245–250, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

Publication

1. Xiao Xian, Seah Hock Soon, Tian Feng, He Ying. Pose Space Deformation on Progressive Decimated Example Meshes. in *Computer Graphics International Conference (CGI2008)* , Pages 188-195, Istanbul, Turkey. June 9-11, 2008.
2. Xiao Xian, Seah Hock Soon, Tian Feng. Skinning on Progressive Decimated Models. in *IEEE Consumer Communications & Networking Conference: Workshop on Digital Entertainment and Creative Technology (DECT 2008)*, Pages 1054–1058, Las Vegas, USA, 10-12 Jan. 2008.
3. Xiao Xian, Seah Hock Soon, Tian Feng. Spherical Skinning from Examples. in *International Workshop on Advanced Image Technology (IWAIT 2007)*, Bangkok, Thailand, 8-9 Jan. 2007.
4. Xiao Xian, John Lewis, Seah Hock Soon, Nickson Fong, and Tian Feng. A powell optimization approach for example-based skinning in a production animation environment. In *CASA 2006, Computer Animation and Social Agents*, pages 141–150, Geneva, Switzerland, 2006. Computer Graphics Society.
5. Xiao Xian, Seah Hock Soon, Wu Zhongke, Tian Feng and Xie Xuexiang. Interactive Free Hand Drawing and In-between Generation with Disk B-spline Curves. in *Multimedia Art Asia-Pacific (MAAP) International Conference on New Media Art, Technology and Education*, Singapore, October, 2004
6. He Ying, Xiao Xian, Seah Hook Soon. Harmonic 1-form based skeleton extraction from examples in *Graph. Models*, volume 7, number 2, pages 49-62, 2009.

REFERENCES

7. He Ying, Xiao Xian, Seah Hook Soon. Example based skeleton extraction using harmonic 1-forms. in *IEEE International Conference on Shape Modeling and Applications*, Stony Brook, USA, 4-6 June. 2008.
8. Seah Hock Soon, Wu Zhongke, Tian Feng, Xie Xuexiang, Lu Yixiang and Xiao Xian Modeling and Visualization of Plants with Ball B-Splines. In International Workshop on Advanced Image Technology (IWAIT 2007), Bangkok, Jan, 8-9 2007..
9. Wu Zhongke, Seah Hock Soon, Tian Feng, Xiao Xian, Xie Xuexiang. Simulating Artistic Brushstrokes Using Disk B-spline Curves. in *Multimedia Art Asia-Pacific (MAAP) International Conference on New Media Art, Technology and Education*, Singapore, October, 2004
10. Seah Hock Soon, Wu Zhongke, Tian Feng, Xiao Xian, Xie Boya. Artistic Brush-stroke Representation and Animation with Disk B-Spline Curve. in *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE)*. Spain, 2005