AUTOMATIC SKELETONIZATION AND SKIN ATTACHMENT FOR REALISTIC CHARACTER ANIMATION

XIN XIE

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

June 2009

School of Design, Engineering and Computing
& National Centre for Computer Animation
Bournemouth University

AUTOMATIC SKELETONIZATION AND SKIN ATTACHMENT FOR REALISTIC CHARACTER ANIMATION

Xin Xie

ABSTRACT

The realism of character animation is associated with a number of tasks ranging from modelling, skin deformation, motion generation to rendering. In this research we are concerned with two of them: skeletonization and weight assignment for skin deformation. The former is to generate a skeleton, which is placed within the character model and links the motion data to the skin shape of the character. The latter assists the modelling of realistic skin shape when a character is in motion.

In the current animation production practice, the task of skeletonization is primarily undertaken by hand, i.e. the animator produces an appropriate skeleton and binds it with the skin model of a character. This is inevitably very time-consuming and costs a lot of labour. In order to improve this issue, in this thesis we present an automatic skeletonization framework. It aims at producing high-quality animatible skeletons without heavy human involvement while allowing the animator to maintain the overall control of the process.

In the literature, the term skeletonization can have different meanings. Most existing research on skeletonization is in the remit of CAD (Computer Aided Design). Although existing research is of significant reference value to animation, their downside is the skeleton generated is either not appropriate for the particular needs of animation, or the methods are computationally expensive. Although some purpose-build animation skeleton generation techniques exist, unfortunately they rely on complicated post-processing procedures, such as thinning and pruning, which again can be undesirable.

The proposed skeletonization framework makes use of a new geometric entity known as the 3D silhouette that is an ordinary silhouette with its depth information recorded. We extract a *curve skeleton* from two 3D silhouettes of a character detected from its two perpendicular projections. The skeletal joints are identified by down sampling the curve skeleton, leading to the generation of the final *animation skeleton*.

The efficiency and quality are major performance indicators in animation skeleton generation. Our framework achieves the former by providing a 2D solution to the 3D skeletonization problem. Reducing in dimensions brings much faster performances. Experiments and comparisons are carried out to demonstrate the computational simplicity. Its accuracy is also verified via these experiments and comparisons.

To link a skeleton to the skin, accordingly we present a skin attachment framework aiming at automatic and reasonable weight distribution. It differs from the conventional algorithms in taking topological information into account during weight computation. An effective range is defined for a joint. Skin vertices located outside the effective range will not be affected by this joint. By this means, we provide a solution to remove the influence of a topologically distant, hence highly likely irrelevant joint on a vertex. A user-defined parameter is also provided in this algorithm, which allows different deformation effects to be obtained according to user's needs.

Experiments and comparisons prove that the presented framework results in weight distribution of good quality. Thus it frees animators from tedious manual weight editing. Furthermore, it is flexible to be used with various deformation algorithms.

LIST OF CONTENTS

Abstract 2
List of Contents 4
List of Figures 7
List of Tables 9
Acknowledgement 10

1 Introduction 11
1.1 Framework and Applications 15
1.1.1 Skeleton Generation Framework 15
1.1.2 Skin Attachment Framework 16
1.1.3 Applications 16
1.2 Objectives 16
1.3 Thesis Layout 17

2 Related Work 20
2.1 Character Animation 20
2.1.1 Layered Model 21
2.1.2 Rigging 22
2.1.2.1 Skeleton Generation 22
2.1.2.2 Skin Attachment 23
2.1.3 Animating 24
2.1.3.1 Skeleton Animation 25
2.1.3.2 Skin Deformation 25
2.1.4 Summary 25
2.2 Skeletonization 26
2.2.1 Curve Skeleton Extraction 26
2.2.1.1 Volumetric-based Method 27
2.2.1.2 Geometric Method 28
2.2.2 Skeleton Embedding 31
2.2.3 Skeleton Generation by Mesh Decomposition 31
2.2.4 Summary 32
2.3 Deformation 33
2.3.1 Anatomy-based Deformation 33
2.3.2 Example-based Deformation 35
2.3.3 Linear Deformation 35
2.3.4 Dual Quaternion Deformation 38

2.3.5 Other Deformation Methods 40
2.3.6 Summary 41
2.4 Skin Attachment 42
2.4.1 Geometrically-based Attachment 42
2.4.2 Example-based Attachment 45
2.4.3 Other Methods of Attachment 46
2.4.4 Summary 47

3 Automatic Skeletonization 48
3.1 Terminology 48
3.2 Overview 50
3.3 Primary 3D Silhouette Detection 51
3.3.1 Finding the Start Vertex 51
3.3.2 Two-Level Detection Algorithm 51
3.3.2.1 Global Search 52
3.3.2.2 Local Search 54
3.3.3 Detection Algorithm Using Linkage Information 55
3.3.3.1 Initial Algorithm 56
3.3.3.2 Modified Algorithm 62
3.4 Skeleton Extraction 64
3.4.1 Curve Skeleton Extraction 65
3.4.1.1 Constrained Delaunay Triangulation 65
3.4.1.2 Decomposition 68
3.4.1.3 Curve Skeleton Refinement 70
3.4.2 Animation Skeleton Generation 73
3.5 Summary 75

4 Automatic Skin Attachment 77
4.1 Overview 78
4.2 Preliminaries 79
4.2.1 Related Mesh Decomposition Methods 79
4.2.2 Fundamentals of Decomposition 80
4.3 Mesh Decomposition 81
4.3.1 Dividing Points Detection 82
4.3.1.1 Type of Joint Linkage 83
4.3.1.2 Detection Algorithm for Single-Linkage Joint 84
4.3.1.3 Detection Algorithm for Multi-Linkage Joint 87
4.3.2 Dividing Points Modification 89
4.3.3 Boundary Generation 91
4.3.4 Decomposition 91
4.4 Weight Computation 92
4.4.1 Division of Dominant Area and Shadow Area 92
4.4.2 Weight Blending in Shadow Area 94
4.5 Summary 95

5 Results, Comparison and Discussion 96
5.1 Skeletonization 96
5.2 Skin Attachment 102

6 Conclusions and Future Work 111
6.1 Conclusions 111
6.2 Future Work 114
6.2.1 Future Work in Skeletonization 114
6.2.2 Future Work in Skin Attachment 115

References 116

LIST OF FIGURES

Figure 1.1	Thesis Layout	18
Figure 2.1	An illustration of skeleton	23
Figure 2.2	An illustration of candy-wrapper artefact	38
Figure 2.3	An illustration of vertex transformation	39
Figure 2.4	Correcting collapsing joint using 'curve skeleton'	41
Figure 2.5	Determining distance between a vertex and a joint	43
Figure 2.6	Illustration of excess deformation caused by vertices getting irrelevant influences	44
Figure 3.1	Extracting a curve skeleton from two 3D silhouettes	51
Figure 3.2	Illustration of global search	52
Figure 3.3	Discrete 3D silhouette vertices after global search	54
Figure 3.4	Angles formed by $c_1^{'}$ with its connecting vertices	57
Figure 3.5	Curvature at a silhouette vertex $c_k^{'}$	59
Figure 3.6	Removing unsuitable vertices using $\overrightarrow{n_k}$	60
Figure 3.7	An illustration of projection of $c_k^{'}$	63
Figure 3.8	Animation skeleton generation process	65
Figure 3.9	Illustration of extracting 3D medial axis from a 3D silhouette	66
Figure 3.10	Two type of edges after constrained Delaunay triangulation	67
Figure 3.11	Calculating the second projection vector of the k^{th} segment of the coarse curve skeleton	70
Figure 3.12	The second silhouette for six segments of a hand model	71
Figure 3.13	Illustration of curve skeleton refinement	72
Figure 3.14	Comparison between coarse and refined curve skeleton	72
Figure 3.15	Illustration of determining a joint on curve skeleton	74
Figure 4.1	Illustration of the surface change around a joint	81
Figure 4.2	Illustration of regions specified by boundaries	82
Figure 4.3	Illustration of joint J as a root of a skeleton hierarchy	83
Figure 4.4	Illustration of two types of linkage	84
Figure 4.5	Determining the XY searching plane for a single-linkage joint	85
Figure 4.6	Detecting dividing points for a single-linkage joint	86
Figure 4.7	Fitting the XY plane for a multi-linkage joint	88

Figure 4.8	Detecting dividing points for a multi-linkage joint	89
Figure 4.9	Illustration of smoothing sampling distance values	90
Figure 4.10	Illustration of selecting dividing points via second derivation	91
Figure 4.11	Illustration of distance measuring	93
Figure 5.1	Illustration of generating animation skeleton for a hand model	97
Figure 5.2	Animation skeleton generation for different models	99
Figure 5.3	Comparisons of the results produced by different classes of curve skeleton extraction algorithms	101
Figure 5.4	Original mesh with embedded skeleton for skin attachment	102
Figure 5.5	Searching for a dividing point at the lower neck joint	103
Figure 5.6	Decomposition based on skeletal joints	104
Figure 5.7	Comparison of weight computation results	105
Figure 5.8	Comparison of deformation effects, front view	106
Figure 5.9	Comparison of deformation effects, back view	107
Figure 5.10	The original mesh with the skeleton of a cat	108
Figure 5.11	Segmentation using a non-planar boundary	108
Figure 5.12	Weight computation result of the upper leg	109
Figure 5.13	Comparison of deformed results	109

LIST OF TABLES

Table 5.1 Timing statistic for each stage of skeletonization 89

ACKNOWLEDGEMENT

First of all, I would like to express my most sincere appreciation to my supervisor, Prof. Jian Jun Zhang. He offered me the precious opportunity to learn from him. I can not forget how much time and effort he has put into my study and my thesis. Nor can I forget how much help he has given me on all aspects of my life.

I would also like to give my deepest gratitude to my other supervisor, Dr. Xiao Song Yang. I came here knowing almost nothing about computer animation. Dr. Yang tutored me and led me through the way. I can never thank him enough for his help, his encouragement and his extraordinary patience.

My special thanks go to Mr. Jun Jun Pan for not only his wonderful collaboration as a fellow researcher but also his inspiring spirit as a friend.

Dr. Jian Chang has helped me with the academic writing style so that I could start writing the thesis. Prof. Mark Hadfield and Ms. Naomi Bailey have helped me with my English. I would like to thank all of them as well.

Finally, I would like to thank my husband, Jian Hu, for his understanding and support. And I dedicate this thesis with my love to my parents. Without them, I can not be where I am now.

CHAPTER 1
INTRODUCTION

The emergence of character animation can be traced back to the early 1900's when a short film *Gertie the Dinosaur* was on air [Vince 2000]. Ever since the use of computers was extended to the graphics domain, it aroused great interest in bringing a virtual character to life by the aid of computer. During nearly a century, character animation has gone through stages from 2D cartoon to computer aided 2D animation, until true 3D animation. It plays an increasingly significant role nowadays in vast areas such as training and entertaining industries.

Character animation features manipulating a digital figure to produce an illusion of its moving. The figure is usually expressed by a surface mesh. Researchers have explored numerous methods to handle a character model mesh feasibly. For an articulated body, to control it with its skeleton, namely skeleton-driven animation, maintains the dominant method in current systems. The deformation of the surface is rendered through calculating displacements of the skeleton over animation time period.

Magnenat-Thalmann et al. [1988] is credited for putting forth the method, namely

Joint-Dependant Local Deformation (JLD), which is considered to be one of the earliest techniques to derive surface deformation of a 3D articulated object using its skeleton.

Improvements and innovations in techniques have advanced since then during the development of skeleton-driven animation. A great deal of them has already been integrated successfully into commercial software packages during past years. They tackle different issues raised from animation.

User friendly interfaces and tools are provided by these packages to ensure easy access for even novices. With their help, modelling 3D characters has become much simpler than before. However, deforming the surface of a 3D character in a lifelike pattern remains a tricky subject.

An anatomical based approach was proposed by Scheepers et al. [1997], Wilhelms and Gelder [1997], etc. This approach adopts anatomical body configuration to create an animatable character model. It takes into account the internal structure which affects the final outcomes of deformation, thus it is capable of producing robust graphical realism. Scheepers et al. [1997] also concluded a typical anatomical based model as three layers: the skeleton layer, the musculature layer which may includes fatty tissues and the skin layer which is represented by surface mesh.

The modelling process of an anatomical based approach works by establishing the model from inside out, i.e. it requires animators to first set up the skeleton, then lay the muscles and finally spread the skin. The musculature layer has to be described explicitly and precisely to fit the external skin shape. This process is opposite to the conventional animation workflow during which the skin layer is the first to be created. This opposition in a certain extent makes the modelling result unpredictable and requires repeatedly manual amendments to reach desired

goals. Although Pratscher et al. [2005] proposed an outside-in method trying to overcome this inconvenience, it only serves ellipse-shaped muscles bearing very simple movements.

Moreover, the deformation is derived from the overall shape of muscles and fatty tissues. Extra parameters and functions are introduced to express influences taken. This adds up to computational complexity during rendering.

Outstanding graphical effects anatomical-based approach may bring, it suffers from unintuitive modelling process and high computational cost. These drawbacks prevent it from being popularized in animation industries.

An alternative approach regards that the skeleton transformation has a direct impact on its skin. It attempts to resemble skin movements without considering the anatomic in-between layer of muscles and fatty tissues.

Direct deduction of skin deformation from the skeleton has been wide-spread in the animation industry, especially in applications where rendering speed is found crucial, such as in computer games, due to its computational simplicity and efficiency.

During the procedure of this direct deduction, each vertex among the model's surface mesh is attached to a chosen set of joints in the skeletal hierarchy with corresponding weight values for indicating the influence a specified joint has on that vertex. The final motion of a character is taken from the combined transformation of all the related bones or joints, thus appropriate skeleton construction and weight assignment are vital steps to achieve performances which are computationally efficient and visually convincible.

To set up a hierarchy, building up bones or joints in a skeleton is mainly

completed through manual work in existing commercial animation software packages. The structure of the hierarchy is also established by hand based on generally accepted examples. Techniques of generating the skeleton from the surface mesh of a 3D model automatically or semi-automatically have been discussed by many researchers in earlier years, yet they do not serve animation as their primary purpose that either the skeleton they produce is not organized for the convenient controlling of animation or the model mesh needs tedious pre-processing.

Apart from setting up a skeleton manually, allocating weights by hand is another and probably much more burdensome task. Most of the commercially used animation software spread weight values according to the Euclidean distance between the vertex and the particular joint. To select the set of relative joints is also based on the Euclidean distance. This mechanism of delivering initial weight always puts irrelevant influences in consideration. It is left to animators to polish up the distribution to obtain feasible values.

In spite of the efforts traditional modelling and animation techniques have made to offer support to animators, efficiency and quality of the techniques either much depend on professional experience and skills of animators as a major part of the work has to be done manually, or relies on a large number of pre-obtained good-quality examples. Moreover, anatomical knowledge may be needed in some cases. This makes the whole process difficult to grasp and highly time consuming.

To solve the problems mentioned above, in this thesis we propose an automatic skeleton generation and skin attachment framework. Within this framework, we develop effective algorithms for skeletal character animation. Compared to traditional method, our method is easy to use, and the result is of rather good quality. It supplies animators with models possessing considerably accurate body details whilst saving them from the tedious work of locating the joints and

adjusting the skin weights manually. At the same time, this algorithm provides enough flexibility, allowing animators of fine tuning according to their needs.

1.1 Framework and Applications

In this thesis, we will present a framework of automatic skeleton extraction, and a framework of weight distribution.

1.1.1 Skeleton Generation Framework

The traditional means of generating an animatable skeleton is to accomplish the construction by hand. When a mesh of a 3D model is supplied, animators will locate joints and construct the skeleton hierarchy manually. This is a labour-intense process. And the ease of control over the skeletal model depends a great deal on animators' personal experiences.

Other automatic or semi-automatic skeleton generation frameworks primarily concern solving rather 'general' cases, e.g. CAD. The algorithms that they employed to construct the 3D curve skeleton of a model, such as voxelization, thinning and pruning, are computationally expensive. They are not suitable for character animation.

In our framework, we propose a skeleton generation algorithm from a different angle. This framework takes advantage of the fact that common 3D animation characters such as humans and animals are usually composed of a series of segments whose cross-sections are approximately elliptic. Therefore the projection of their 3D curve skeleton on a 2D plane can be approximated with the 2D medial axis of the projection of the original model in the same projection orientation. It suggests that the problem of finding a 3D curve skeleton can be settled in 2D spaces. It allows for developing an efficient and effective purpose-built skeleton generation approach for animating various characters.

1.1.2 Skin Attachment Framework

The most widely adopted method by existing animation software to define weight values is to compute first the distances between a vertex and the joints. A number of joints are then selected as a relevant influence set based on the geometrical closeness. Popular animation software, e.g. Maya, usually chooses 4-5 closest joints as the related joint set. During skin attachment, weights are blended in accordance with its proportion based on its distance. This distribution is quite rough. Careful rectification has to be made manually afterwards in order to get desired deformation.

In this thesis, we come up with a solution which tries to compute skin weights automatically with enhanced usability. We first segment the mesh into different regions of which each is associated with one joint. Then we base the weight distribution on the distance between the vertex and the region boundary. This algorithm associates a vertex with a more realistic influence joints set. Thus the distortions resulting from getting irrelevant influences may be reduced.

1.1.3 Applications

We contribute our framework to realistic skeletal 3D character animation, yet its application is not confined. Techniques derived during the research, e.g. the 3D silhouette detection, might be applied to other fields as well.

1.2 Objectives

The overall objective of the framework proposed here is to develop an automatic skeletonization and skin attachment approach whilst not breaking the traditional animation work pipeline. The framework is expected to possess qualities in the following aspects:

- It should be easy to use. The method proposed should aim at not only professional animators but also novices with little experience in animation.

With the most tedious and skilful part accomplished by computer, character animation would become a more accessible work.

- It should be intuitive and interactive. Fine tunings should be allowed on outcomes during both skeleton generation and skin attachment processes in order to suit different needs in applications.
- It should be example-independent. Pre-obtained example database may facilitate the refinement of the results but is not necessary in both skeleton generation and weight computation process.
- It should produce more accurate results which can be used in animation with much less manual supervision than before. This will save labour and time of animators.
- Most importantly, it should be compatible with the current industrial animation pipeline, which will make it possible to be directly adopted in commercial software.

1.3 Thesis Layout

The structure of the thesis is presented in figure 1.1.

An overview of this thesis is given in Chapter 1. It outlines the objective and basic methodology of our work.

Chapter 2 introduces character animation and some difficulties encountered by skeleton-driven animation on a general background, which explains the motivation of our research. Previously developed techniques regarding both skeleton generation and skin deformation will be reviewed in this chapter as well, which lays the foundation of our research.

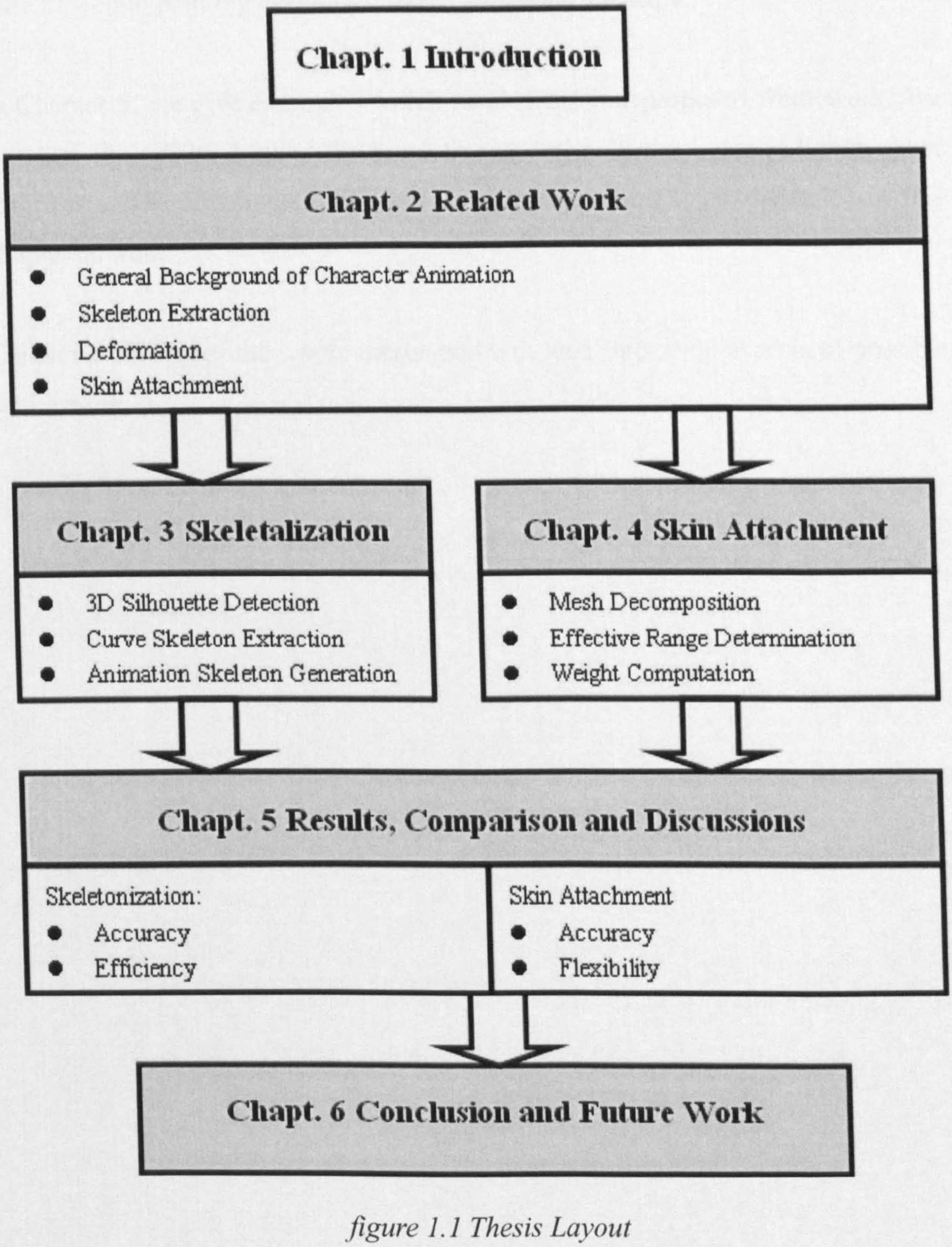


figure 1.1 Thesis Layout

Chapter 3 presents our framework of automatic skeleton extraction which regards character animation as its main purpose.

Chapter 4 constructs a reliable weight computation framework as a solution to

manual weight painting currently used in animation software.

In Chapter 5, we give examples which result from our proposed framework. We compare the results derived from our framework with those derived from other algorithms. The advantage of our framework is examined and discussed in this chapter as well.

Chapter 6 concludes the whole thesis, and will look into some aspects of possible future work.

CHAPTER 2

RELATED WORK

To attain believable effects in computer animation relies on accurate graphical and mathematical representations, and feasible instructions during the animation process which generate and render a motion sequence. Thus developing algorithms and a framework which helps to improve visual realism is one of the issues of primary research interest.

In this chapter, we will first briefly introduce the general background of character animation. Then previously developed techniques which are related to our research, such as rigging and skinning, will be reviewed in detail.

2.1 Character Animation

The main idea of character animation is to associate computer generated figures, either human or non-human, with visually convincible movements [Vince 2000]. In industry pipeline, it undergoes several stages including: story design, which lays out the plot and the guidance of desired characters; modelling, which is concerned with setting up the character's geometry; and animating, which assigns appropriate motion data to the character.

A popular approach of animating an articulated character is skeleton-driven animation. To facilitate the description of our research, a general background to the basics of the skeleton-driven character animation is given in this section. It lays the foundation for better understanding of our skeletonization and skin attachment framework.

2.1.1 Layered Model

In animation practice, every movement of a virtual character is created by manipulating a computer-generated model. A better quality model enhances the sense of reality. At the same time, it usually increases the amount of data to be processed. To get around this issue, animators often use a layered model structure in character animation.

The shape of a character is represented by its surface which is called *skin*. The skin can be approximated by different geometric entities, such as polygons [Botsch et al. 2007], or by parametric surfaces [Eck and Hoppe 1996] [Shen et al. 1994]. The most common description of such a surface is a triangular mesh which is composed of interconnected vertices, for its flexibility in approximating arbitrary topologies.

The skin usually consists of a large amount of vertices, which makes it extremely expensive in terms of computation to be animated directly. Consequently, a lower resolution representation was introduced into animation, which is called *skeleton*. It provides a means of easy control over an animation process: specify the skeleton movements in each animation frame and transformations will be passed to skin as designed.

Skin and skeleton are considered as two requisite elements for a character model used in skeleton-driven animation. In-between layers, such as musculature or fatty

tissues, sometimes are appended to provide extra details for the model [Scheepers et al. 1997] [Yang and Zhang 2006a]. In this case, other methods are demanded to describe their representations and animated effects.

2.1.2 Rigging

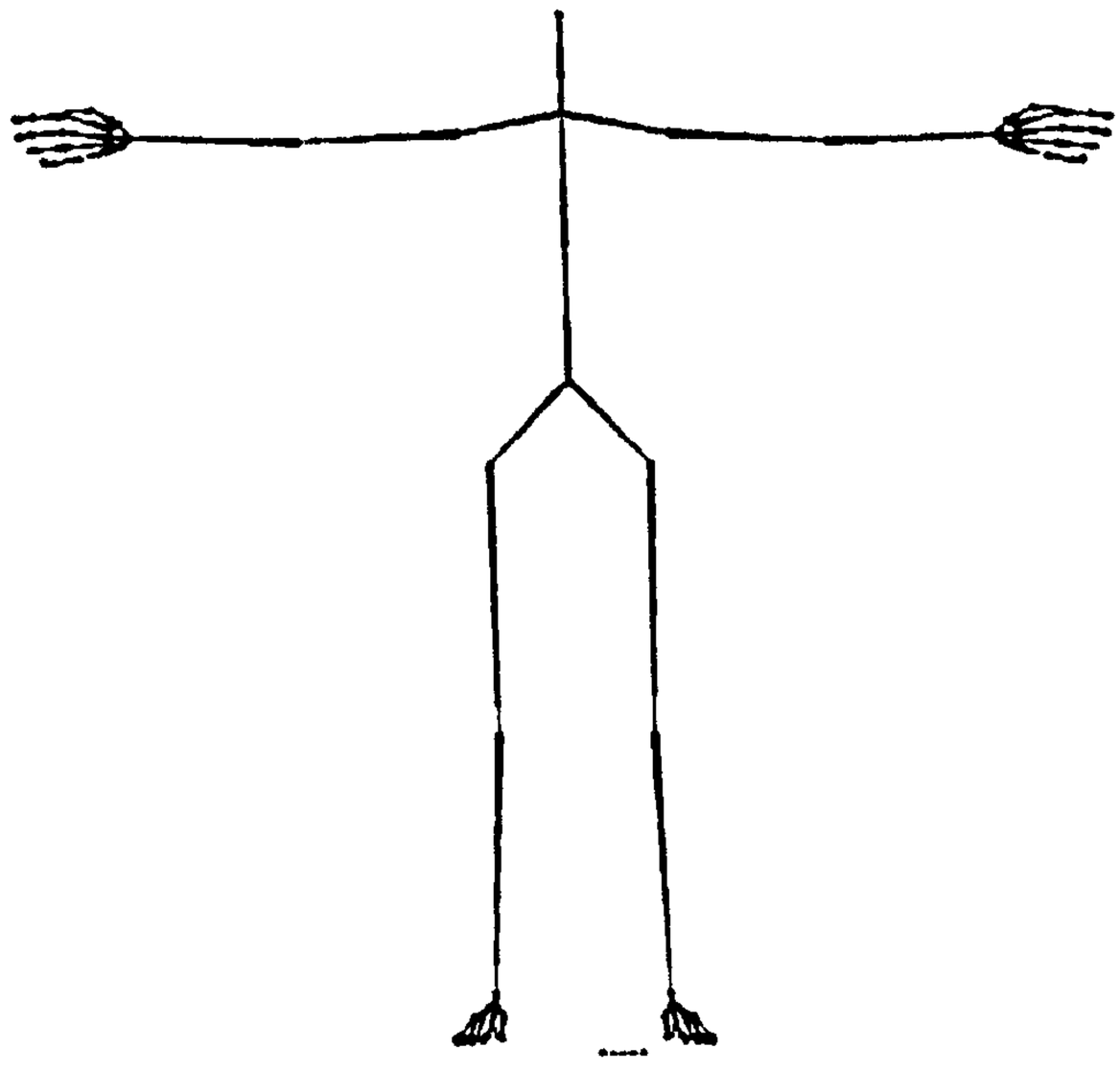
In skeleton-driven animation, *rigging*, which defines how the internal structure (skeleton) affects the movements of the model surface, is the basis of character animation framework.

2.1.2.1 Skeleton Generation

The skeleton used in animation is purposed for controlling the character. Thus it is not necessary that it strictly depict the anatomical skeletal structure of a character. However, in the interest of graphical realism, understanding the anatomy is important for rigging. Poor placement of joints may cause severe distortions.

Figure 2.1(a) and (b) illustrates the anatomical skeleton and the animatible skeleton of a human character.

(a)



(b)

figure 2.1: An illustration of skeleton: (a) an anatomical skeleton; (b) an example of an animatible skeleton.

The animatible skeleton is a set of hierarchically organized joints embedded in the skin mesh [Kavan and Žára 2003]. Each joint is described by variables specifying its position and orientation, and each connecting pair of joints is bound by a parent-child relationship.

With current state-of-the-art commercial software, generating a skeleton is mainly done by hand. Animators have to manually locate each joint inside a character, which is a time-consuming and skilful task.

Techniques on automatic skeleton generation have been produced. Due to different unsuitabilities, they have not been adopted in the modern industrial animation pipeline. The following section will give an overview on their pros and cons.

2.1.2.2 Skin Attachment

Joints in the skeleton will not be rendered to be seen during runtime. They are actually matrices or other mathematical forms which represent transformations. Therefore, after setting up the skeleton hierarchy, it needs to be bound to the skin in order to display the visual effects.

Each vertex in the mesh is associated with the skeletal joint with a scalar value which is referred to as *weight*. Weight reflects how much influence a joint contributes to a vertex. The larger the weight value is, the greater the joint influence on the vertex is.

Basically, skin attachment methods can be categorized into two groups based on the number of joints a vertex is attached to. In the animation software Maya, it is referred to as rigid skinning and smooth skinning. Rigid skinning confines the attachment of a vertex to only one joint, while smooth skinning attaches a vertex to several joints. In the latter case, the movement of the vertex is regarded as transformed by each attached joint respectively and the final position is interpolated by all the transformed positions.

In current animation pipeline, animation software provides an initial weight distribution. However, the quality is often too poor to be used in animation. Again animators have to manually adjust the weights, often referred to as weight painting, to achieve correct skin deformations, which is one of the most tedious processes in character animation. Several algorithms have been proposed which intend to evaluate the influence of a joint on a vertex automatically. However, they failed in compatibility so that they have not been integrated into current animation systems.

2.1.3 Animating

The stage of animating, i.e. bringing a character to life, is the most crucial phase during computer animation procedure.

2.1.3.1 Skeleton Animation

Many techniques can be applied to the skeleton to determine joint movements at each presented frame. The animation of a skeleton yields to its underlying hierarchical structure: Transformations are inherited by a child joint from its parent joint [Clark et al. 2005].

The keyframing technique allows the animator to draw a few 'key' frames, and leaves in-between frames interpolated by the computer. It is adopted by almost all the animation systems. Another technique to define joint movements over time is to modify them using parametric curves. When a prototype of virtual character exists in reality, motion capture techniques can be used as well to obtain realistic motion data [O'Rourke 1998].

2.1.3.2 Skin Deformation

A major topic in character animation is to express changes that made upon shape of its surface with its skeleton, which is often referred to as *skin deformation* or *skinning*. Numerous approaches regarding this area have been explored to cope with model features [MacCracken and Joy 1996] [Singh and Fiume 1998] [Shen et al. 1994] [Hyun et al. 2003].

In skeleton-driven animation, with each vertex in the skin mesh attached to the skeleton, different level of influences are taken from joints according to the weight associated. Various strategies are provided to define the deformation.

2.1.4 Summary

Character animation applies appropriate motion data to computer generated either human or non-human figure models. Given that human perception to distortions occurred on a character is usually sensitive, when visual realism is the purpose, it could be difficult to achieve.

In this section, we have presented a brief introduction to skeleton-driven character animation. Skeleton-driven character animation has become a *de facto* industrial standard, yet with current animation software, to achieve the desired result involves a great deal of manual work. In the following sections, we will review related algorithms and techniques previously developed which aim to achieve realistic results with less manual intervention.

2.2 Skeletonization

As stated earlier, the skeleton plays a fundamental role in skeleton-driven animation. Placing a skeleton is still mainly done by hand in current animation industries, although researches on automatic skeleton generation started a long time ago. Among these automatic skeletoniztion methods, numerous researchers focus on extracting a curve skeleton which roughly resembles a desired animatible skeleton, others aim at obtaining the skeleton of a target model by referring to a template model.

2.2.1 Curve Skeleton Extraction

Although differing in terms of mathematical ways, curve skeleton [Dey and Sun 2006] [Lieutier 2003] and medial axis [Blum 1967] are two closely related concepts. They are both 1D representations which encodes topological information of the original object. The curve skeleton can be represented by or generated from the medial axis in many applications.

Most of curve skeleton extraction algorithms are based on drawing medial axis of the model. An overview of curve skeleton generation has been given by Cornea et al. [2007]. Depending on the mesh type with which is dealt, existing extraction algorithms fall into two main categories: volumetric-based and geometric-based. The former handles volumetric solid meshes, while the latter handles surface meshes which can be regarded as a 2D surface embedded into 3D space.

2.2.1.1 Volumetric-based Method

Volumetric-based method can be further divided into three classes: thinning, distance field based, and general field based.

Thinning methods use templates or masks against which all boundary voxels are tested to determine whether it is a simple point, i.e. a point which can be removed whilst the topology of an object is preserved. Curve skeletons are generated by iteratively removing simple points inward until it reaches the desired thinness. [Ma et al. 2002][Svensson et al. 2002]

The fact that this method relies on removing simple points increases the risk of excessive shortening of curve-skeleton branches since an end-point of curve skeleton itself is a simple point. Furthermore, the curve skeleton produced by thinning algorithms is not always smooth. Wang and Lee [2008] proposed a method to achieve smoother skeleton by applying an iterative least-squares optimization to shrink models before thinning, but the skeleton it produced suffers the potential problem of losing good centrality.

Distance field methods define the smallest distance from an interior point of an object to the boundary as the distance field of that point. They attempt to detect the ridges of the distance field to identify those locally centred voxels within an object as candidates. A curve skeleton is extracted by pruning down the candidate voxel set and then connecting the remaining voxels. [Couprie et al. 2007] [Bitter et al. 2001] [Zhou and Toga 1999]

These methods are most successful in extracting medial surfaces and the computation of a distance field is fast. The drawback is that the candidate voxel set is considerably large in quantity and additional processing methods are required to extract curve skeletons out of medial surfaces.

General field methods [Chuang et al. 2000] make use of functions to create a field other than distance field. The field of an interior point is defined as the sum of potentials regarding the object's boundary which are generated by certain function. Cornea et al. [2005] considered boundary voxels also as points. These methods extract curve skeleton by detecting and connecting local extremes of the field. Grigorishin et al. [1998] presented an electrostatic field function to generate a potential inside a 3D object to extract the curve skeleton, while radial basis functions are adopted by Ma et al. [2003]. Liu et al. [2003] introduced the Newtonian repulsive force to solve the searching problems, which can be regarded as a special case proposed by Chuang et al. [2000].

The main advantage of general field methods is that they are capable of generating nicer curves on medial surfaces than distance field methods do. However, this merit is achieved at the price of sacrificing computation efficiency and numerical stability.

Volumetric-based methods require the model mesh to be volumetrically solid. However, the models provided in animation are usually only surfaces. Pre-processing, such as voxelization, is always required. The voxel size has to be smaller than model features in order to describe details of a model correctly. This adds significant complexity, hence leads to extra computation costs.

2.2.1.2 Geometric Method

Geometric based method is most applicable to models which are represented by discrete points or polygonal surface meshes.

Voronoi diagram is widely used in geometric-based method category for its simplicity and intuitiveness [Brandt and Algazi 1992] [Wu et al. 2006]. The Voronoi diagram can be generated by either vertices of a 3D polygonal mesh or a

set of scattered vertices. It describes a subdivision of a metric space into regions of which each consists of elements closest to a generator element than all the rest elements. Internal faces and edges of these regions are used in extracting the medial surface of a 3D object.

M-reps [Pizer et al. 1999] obtain the medial surface on the basis of generalized *cores* which provides additional information of local shape of the object. The medial surface is modelled with a web formed by connected cores. **Shock scaffold** is similar in basing its extraction method on contact spheres. It extracts the medial surface by a propagation of an initial source along the scaffold [Leymarie and Kimia 2003]. In a later study [Leymarie and Kimia 2007], medial scaffold was proposed which intends to improve the shock scaffold method.

Instead of producing medial axis directly, Voronoi diagram methods, M-reps and scaffold methods all produce approximation of medial surface, which is similar to the methods using distance field functions. Although further processing, such as pruning or curve thinning, can be applied to achieve medial axis [Culver et al. 2004] [Yang et al. 2004], it increases the computational cost.

Reeb graph based methods have attracted much attention in recent years. The Reeb graph is a 1D structure which represents the topology of the model by establishing correspondence between its nodes and critical points of the function defined on the model surface. Reeb graph is not a curve skeleton, yet an embedding of Reeb graph into the original space of an object can be used to define a curve skeleton for the object [Pascucci et al, 2007]. Biasotti et al. [2000] and Xiao et al. [2003] extended the application of Reeb graph to polygonal meshes.

Reeb graph directly generates 1D skeleton from the object. It maintains a comparative insensitivity to boundary noise. However, most Reeb-graph-based

methods require the user to set the boundary conditions explicitly, which is a task of difficulty. And re-sampling on model surface is needed to increase the skeletal nodes which can be obtained.

Mesh contraction method was presented by Au et al. [2008] to extract the skeleton using implicit Laplacian smoothing. The contracted mesh is then converted into a 1D structure curve through a connectivity surgery process. The result of this method is of good quality. However, it shows some obvious disadvantages besides the limitations mentioned by the author. For example, to ensure the collapsed shape preserve the original geometry in contraction process, the weights of attraction and contraction constraints for different vertices must be carefully set, which is a difficult task for animators in practical use. Moreover, the geometry contraction is also a very time-consuming process.

Point cloud was proposed most recently by Tagliasacchi et al. [2009] to extract the skeleton from incomplete point clouds with missing data information. They used a planar cut in search of rotational symmetric axis (ROSA) points which are connected into a curve skeleton. Field function or intermediate surface representation is not required by this method. However, the convergence of the planar cutting is not proved. Moreover, it involves a great deal of optimization and parameterization, which indicates a high level of complexity of this method.

The skeletonization algorithm presented in this thesis belongs to geometric based category. It extracts a curve skeleton using Voronoi diagram generated from constrained Delaunay triangulation. However, compared with other methods which also feature Voronoi diagram, one remarkable advantage of our algorithm is that the result from the two perpendicular silhouettes is the 3D medial axis, not medial surface. It can generate a curve skeleton directly without post-processing such as thinning and pruning, which significantly speeds up the computation.

2.2.2 Skeleton Embedding

Skeleton embedding differs from the above methods which extract curve skeleton as the groundwork for generating an animatible skeleton. Skeleton embedding obtains an animatible skeleton of a given model by applying an existing template to it.

To obtain an optimal skeleton fitting for the given model, Baran and Popović [2007] designed a maximum-margin supervised learning method with a set of hand-constructed penalty functions. They also presented a Laplace's diffusion equation based method for skin attachment. Their method is fast enough, capable of rigging a character model of over 50,000 vertices in less than one minute on a midrange PC. However, it asks for well-proportioned character models and often fails in correctly positioning knees and elbows.

Aujay et al. [2007] presented a method which combines Reeb graph and skeleton embedding. It constructs the Reeb graph of a harmonic function, which gives the overall morphological structure of the model. Then it is refined and embedded to generate the animatible skeleton by the aid of anatomical information. This approach is quite fast as well, yet it needs tuning of parameters and a reference anatomical template of the character during the refining process.

Skeleton embedding method in general requires that a character model has approximately the same proportion and pose as the template. Moreover, every type of character needs a template of its own. For example, in the paper of Baran and Popović [2007], the embedding method was only applied to the human-like characters due to the template they have held. To set up various templates indicates extra labours to be taken.

2.2.3 Skeleton Generation by Mesh Decomposition

Decomposing a model can facilitate skeleton generation, for it provides guidance

for understanding the model structure. Our skeletonization framework benefits from mesh decomposition as well.

Generating the skeleton with mesh segments was mentioned in the study of Katz and Tal [2003]. Their method segments mesh into meaningful components which are in forms of hierarchically structured patches according to region concavity. The skeleton thus could be inferred by associating each patch with a joint. The hierarchy of the skeleton can be deduced as well from the hierarchy of skin patches. However, the complexity involved in mesh segmentation of this method slows down the process of generating skeleton.

In Lien et al. [2006], skeleton is also generated from decomposed mesh. They first generated a simple skeleton from the input model either using centroids, which is easier, or using the principal axis, which is more accurate. The skeleton is checked against some criteria defined by the user. If the quality is not satisfied, the model is further decomposed into finer segments. Skeleton extraction is then performed according to each component and is expected to evolve to a better quality. The mesh decomposition and skeleton extraction are repeated in turn, until the desired result is achieved. Obviously, this approach means a process of tedium, where our skeleton extraction framework intends to improve.

2.2.4 Summary

In section 2.2, we reviewed the techniques for skeleton generation which have been developed previously. Although many algorithms aiming at automation have been proposed, they are primarily for a more general purpose than for character animation as a specific objective. As a result, these methods are typically computational expensive or need significant manual intervention from the user.

A few methods mentioned above regard computer animation as their main concern, but they have their own drawbacks, such as low efficiency, which holds

back their suitability in practice.

Our skeleton generation framework presented in this thesis aims to present an accurate and efficient approach, whose particular purpose is to automatically rig a 3D character for animation.

2.3 Deformation

The movements of a virtual character are expressed visually through positional changes of its skin vertices. It is impractical to give a direct description of every vertex at each animation frame considering the large number of vertices in a model mesh. Thus deformation algorithms are needed to map transformations of the skeleton of a character to its skin, which provides a more manageable means of tracing vertex displacements.

Skeleton-based deformation has been favoured by various researchers for manipulating articulated characters. Burtnyk and Wein [1976] proposed an algorithm of 2D bilinear skeletal deformation. An early 3D skeleton-driven deformation algorithm was addressed by Magnenat-Thalmann et al. [1988]. It has evolved into various styles and algorithms over decades.

Efficiency has always been an important criterion in judging deformation algorithm. Recent studies cast great attention on enhancing visual realism of character animation as well. Therefore, a great deal of research work has been dedicated to both directions.

2.3.1 Anatomy-based Deformation

In terms of realism, anatomy-based deformation demonstrates an ability of the closest resemblance of reality.

By studying artistic anatomy [Goldfinger 1991], factors that affect the surface

form of a character were identified by Scheepers et al. [1997] as skeleton, muscles and fatty tissues. Scheepers et al. [1997] approximated muscles with ellipsoids and presented a general muscle bending model. However, algorithms of skin deformation by means of muscles movements are not mentioned in the paper.

Another muscle model was proposed by Kähler et al. [2001] and adopted by Albrecht et al. [2003]. A hand model was deformed in the study of Albrecht et al. [2003] based on anatomical data and mechanical laws through pseudo-muscles which control the rotation of bones and geometric-muscles which shape the formation of the skin.

The graphical effect of anatomical based deformation is impressive for it works by imitating details of an actual character. However, to handle such a model, a large number of parameters and intricate algorithms are required in order to capture the fine features to create these elegant effects. This inevitably results in prolonged rendering time. Furthermore, it suffers from a profound modelling process, which as well adds its complexity. As for real time interactive applications, such as gaming, anatomy-based deformation quite likely fails to meet their needs.

Trying to overcome these difficulties, Pratscher et al. [2005] provided a simplified anatomical model with an 'outside-in' modelling method. It generates muscles by first segmenting mesh into regions which correspond to bones. Each region is again divided into two sub-regions and a pair of ellipsoids is created to represent the rough geometry of the muscles. The ellipsoid muscle is then reshaped to fit the skin. This method takes off much the complexity of producing a model suitable for anatomy-based deformation. However, it is still expensive in terms of computation for real-time rendering. Furthermore, it is not proved to work well on models.

2.3.2 Example-based Deformation

Example-based approach is considered to be another powerful solution for realistic skin deformation. Its way of deforming skin is to mimic the deformation pattern of training examples.

The method proposed by Wang et al. [2007] aims at capturing extra deformation details, such as twisting and bulging, by predicting deformation gradient D [James and Twigg 2005] from bone transformations q. The mapping between q and D is solved by a training process during which a sequence of existing mappings are provided as examples.

Other example-based deformation approaches can be found in [Mohr and Gleicher 2003], etc.

The common disadvantage of all example-based deformation algorithms is its high dependence on its database. This example database has to be of good quality. If it is not available from real characters, it has to be prepared with other methods, which could mean another tedious process.

2.3.3 Linear Deformation

Within linear deformation framework, transformation of a joint is defined as a 4×4 matrix under homogenous coordinates that incorporates both translation and rotation information. Given initial vertex position v, its new position with regard to the i^{th} joint is deduced by:

$$v_i' = M_i \hat{M}_i^{-1} v \qquad (2.1)$$

where i stands for the i^{th} joint, M_i for the transformation matrix associated with the joint in its current pose, and $\hat{M}_i$ for the transformation matrix in its binding pose.

During animation, the vertex is considered to be transformed by a number of joints. Linear deformation algorithms basically can be regarded as finding the best strategy for blending these transformations.

Among all deformation algorithms, be it linear or non-linear, **Skeleton Subspace Deformation** (SSD) is the most popular one. It is widely spread in real-time animation for its well-known efficiency. SSD has never been published formally, yet its description is available in many literatures under different names, e.g. *Enveloping* or *Linear Blend Skinning* (LBS) [Lewis et al. 2000] [Wang and Phillips 2002] [Merry et al. 2006]. In some commercial animation software, e.g. Maya, it is called smooth skinning.

SSD associates a vertex to each joint with a weight. The resulting position is an interpolation between all the deformed positions with reference to their weighted influences respectively, as shown in equation (2.2)

$$v' = \sum_{i=1}^{n} w_i M_i \widehat{M}_i^{-1} v \qquad (2.2)$$

Considering the simplest example of associating vertex v to two joints J_1 and J_2, the resulting position v' would lie on the line segment connecting v_1' and v_2', as illustrated in figure 2.3(b).

In practice, a vertex is associated with several joints, which gives a bit more freedom to its possible deformation space. Nevertheless, the interpolated position v' is restricted to inside a convex combination of each transformed position of v. The stiffer the combination is, the more restricted the deformation space is.

Therefore, as efficient and simple as SSD may be, it is vulnerable for artefacts such as collapsing-joint and candy-wrapper caused by such stiffness. In search of improvements, one direction is toward increasing flexibility over blending

components during linear blending.

Wang and Phillips [2002] proposed a **Multi-Weight Enveloping** (MWE) technique which launches additional weights for transformation components. That is to say, let transformation matrix be $T = M_i \hat{M}_i^{-1}$, Wang and Phillips [2002] rewrote vertex blending as:

$$v' = \sum_{i=1}^{n} \begin{bmatrix} w_{00_k} m_{00} & w_{01_k} m_{01} & w_{02_k} m_{02} & w_{14_k} m_{14} \\ w_{10_k} m_{10} & w_{11_k} m_{11} & w_{12_k} m_{12} & w_{14_k} m_{14} \\ w_{20_k} m_{20} & w_{21_k} m_{21} & w_{22_k} m_{22} & w_{24_k} m_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet v \quad (2.3)$$

The extra weights allow more degrees of freedom. Additional flexibility is expected to produce non-linear effect to reduce the artefacts. However, tuning weights depends on a range of existing well-established example poses. An increased number of weights lead to a higher risk of overfitting as well. Furthermore, taking into more parameters suggests more memory usage and slower performance.

Merry et al. [2006] tackles the same issue of improving the result of linear blending with a method called **Animation Space**. Let $\hat{v}_i' = w_i \hat{M}_i^{-1} v$, the blending equation of Animation Space is written by them as:

$$v' = \sum_{i=1}^{n} M_i \hat{v}_i' = \begin{bmatrix} M_1 & M_2 & \dots & M_n \end{bmatrix} \bullet \begin{bmatrix} \hat{v}_1' \\ \hat{v}_2' \\ \dots \\ \hat{v}_n' \end{bmatrix} \quad (2.4)$$

This method tries to bypass the distortion problem by providing a more flexible blending as MWE does. It allows a joint to impose independently on every component of a vertex description. In this regard, it also increases the number of weights as MWE does. This drives Animation Space method into a dilemma:

painting weight by hand makes it painstaking as SSD is, while tuning weights by examples makes it face the same problem as MWE does.

2.3.4 Dual Quaternion Deformation

In mathematical terms, changing of orthogonality of the rotational components causes non-rotational factors being introduced into the rotation matrix during pure linear blending. This explains the candy-wrapper artefact encountered in SSD. Should the rotational angle be small, the errors are so trivial that they may not have much influence on the overall effects. As the rotational angle increases, the errors are increased, and distortions become more noticeable, as shown in figure 2.2.

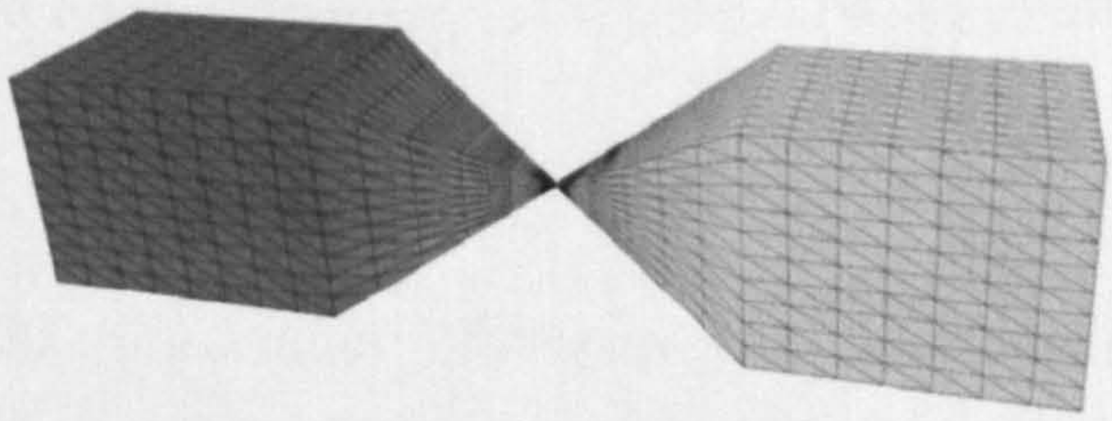


figure 2.2: An illustration of the candy-wrapper artefacts brought by Skeleton Subspace Deformation [Kavan and Žára 2005].

The artefacts mentioned above are inherent for pure linear blending. Therefore, another direction of reducing these distortions is not toward a modified linear blending but toward a totally different way: bringing nonlinearity into the deformation algorithm.

A method called **Spherical Blend Skinning** (SBS) was proposed by Kavan and Žára [2005]. Intuitive-wise, the deformed position would lie on an arc not on a line segment if considering the same simplest example of a vertex affected by two joints, as shown in figure 2.3(c).

SBS splits the transformation into a pair of elements: translation and rotation. Instead of blending transformed positions of a vertex, it interpolates directly the

transformation quaternion which is a replacement of a transformation matrix.

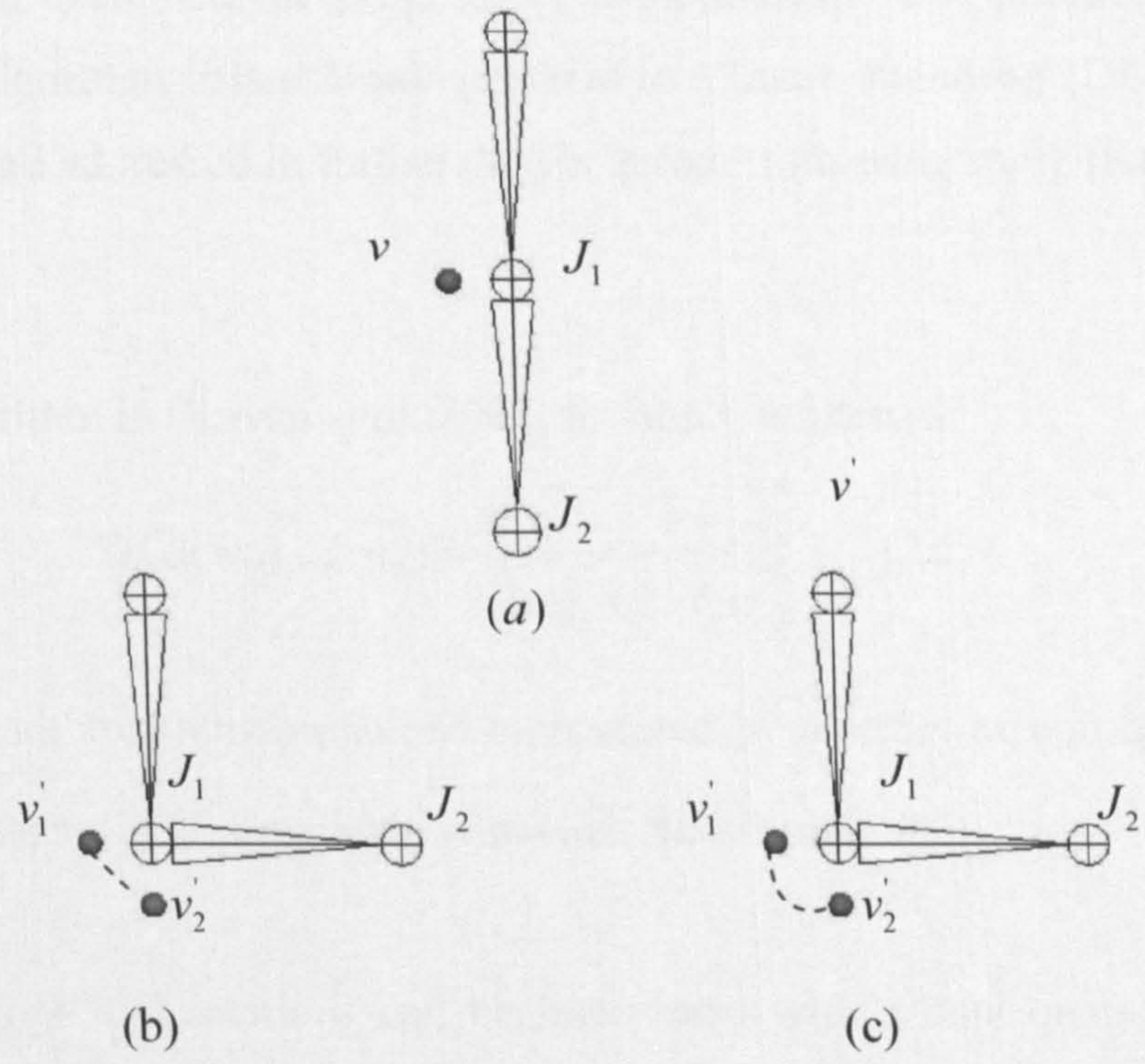


figure 2.3: An illustration of vertex transformation.(a)initial position; (b)blended by SSD; (c)blended by SBS. The actual position is influenced by the ratio between w_1 *and* w_2

Due to the incapability of a regular quaternion to represent translation, one bottleneck of this interpolation is to determine the rotation centre in order to approximate a non-translational transformation as closely as possible. Kavan and Žára [2005] proposed to compute precisely the optimal rotation centre for each individual vertex using singular value decomposition (SVD) but with a high computational cost.

In exchange for efficiency, the runtime algorithm was twisted into caching rotation centre derived for a previous vertex and reusing it for other vertices assigned to the same joints set. This trick potentially causes discontinuity of deformed skin due to discontinuous change of rotation centres.

Regarding this, Kavan et al. [2007] proposed to base the transformation on dual quaternions and examined the proposal both theoretically and practically. They presented an algorithm called **Dual quaternion Linear Blending** (DLB) which was polished and addressed in further details in their following study [Kavan et al. 2008].

The DLB algorithm in [Kavan *et al*, 2007] is simply written as:

$$DLB(w;\vec{q_1},...,\vec{q_n}) = \frac{w_1\vec{q_1}+...+w_n\vec{q_n}}{\left\|w_1\vec{q_1}+...w_n\vec{q_n}\right\|} \qquad (2.5)$$

where $\vec{q_i}$ stands for transformations represented by a series of unit quaternions and w_i stands for weights associated with each transformation.

Both translations and rotations can be interpreted with a dual quaternion. This enables a dual quaternion to represent rotations about an arbitrary axis. In other words, dual quaternion rotation is coordinate-independent. Thus DLB eliminates problems caused by determining rotation centres.

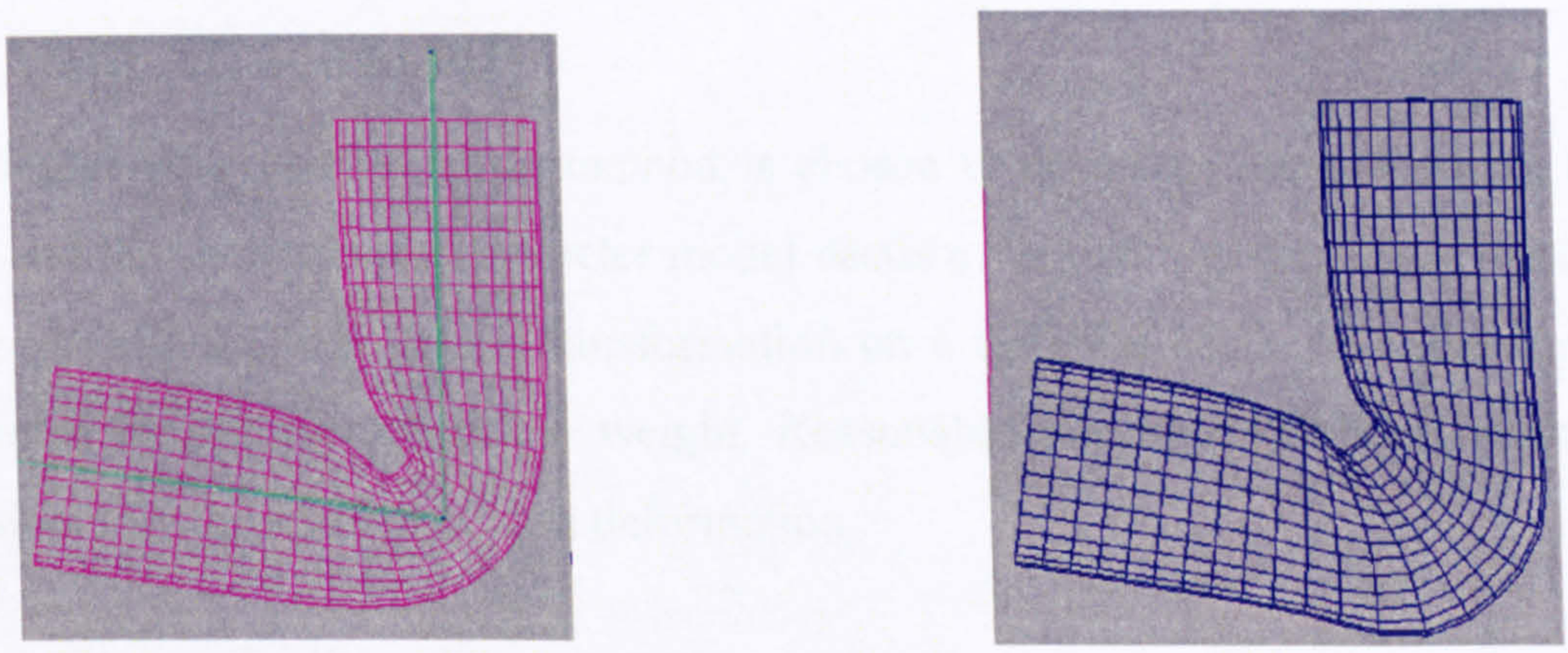
DLB exhibits a fast and robust performance on rigid joint transformations, whereas it demands much longer shader code and more memory usage in runtime than SSD where non-rigid transformations, such as non-uniform scaling to vary the proportion of a character, are involved. In this regard, DLB is not ready yet to take the place of SSD in animation industry.

2.3.5 Other Deformation Methods

The pure linear blending also introduces non-uniform scaling and skewing into transformation components during matrix multiplication, which causes volume loss while bending a joint, as shown in figure 2.4.

Yang et al. [2006] proposed to solve the deformation using 'curve skeleton'. The

name of 'curve skeleton' is different from its traditional concept. It is used in their method as a control curve layered on top of the animation skeleton. A local frame of every skin vertex is defined by associating the vertex with a curve segment. The transformed position of a vertex is calculated via the transformed local frame on the curve. This method corrects the collapsing joint problem caused by SSD, and is compatible with current animation software, e.g. Maya. However, it is not easy to decide the appropriate associating curve segment for a vertex when the bending angle of a joint is large.



(a) (b)

figure 2.4: Correcting collapsing joint using 'curve skeleton'. (a) An illustration of collapsing joint caused by SSD. (b) Correction of collapsing joint using 'curve skeleton'. [Yang et al. 2006]

Similar idea of transforming vertex using associated local frame was adopted by Fortsmann et al. [2007]. They defined spline-based coordinates for a skin vertex and used Free Form Deformation (FFD) rather than SSD when transforming the vertex. Their problem is how to establish the appropriate bounding box required by FFD. Another problem is that new tools and a new data format is demanded, which makes it difficult to be integrated in existing animation system.

2.3.6 Summary

In section 2.3, a review has been given for the different deformation methods. Despite the excellent performance of other deformation algorithms, SSD still

dominates the current animation industry for its simplicity, efficiency, and ability of expressing all types of transformations.

Instead of producing a replacement method of SSD, the skin attachment framework presented in this thesis focuses on automatic weight distribution. This framework aims to provide an excellent weight distribution result as well as reducing the tedium of manual skin weight editing.

2.4 Skin Attachment

No matter which deformation method is chosen in an animation procedure, the skin and the skeleton of a character model needs to be tied together. Each joint or bone contributes to a vertex transformation on a different scale. Its influence is reflected by the corresponding weight. Reasonable weight distribution affected considerably the realism of skin deformation.

2.4.1 Geometrically-Based Attachment

In modern animation industry, SSD is still the prevailing approach for animating an articulated character. Despite its well-known simplicity and efficiency, SSD is unfortunately tedious in terms of defining weight, which is predominately undertaken manually.

Commercial animation software integrated with SSD, e.g. Maya, does make a guess for initial weight distribution based on the distance between a skin vertex and its associated joints. A deduction made on its principles of the guessing was given by Yang and Zhang [2006b].

Let us denote the vertex with v, the associated joint with J_i, and the child joint of J_i in the skeleton hierarchy with J_k. If we project v to the line on which joint link $\overrightarrow{J_iJ_k}$ lies, depending on the position of the projection v', the distance

$D(v,J_i)$ between v and J_i regarding child joint J_k can be defined in three ways as follow:

$$\alpha = \frac{\overrightarrow{J_i v} \bullet \overrightarrow{J_i J_k}}{\left|\overrightarrow{J_i v}\right|^2}; \qquad (2.6)$$

$$D(v,J_i) = \begin{cases} \dfrac{\left|\overrightarrow{J_i v} \times \overrightarrow{J_i J_k}\right|}{\left|\overrightarrow{J_i J_k}\right|}, 0 \le \alpha \le 1 \\ \left|\overrightarrow{J_i v}\right|, \alpha < 0 \\ \left|\overrightarrow{J_k v}\right|, \alpha > 1 \end{cases} \qquad (2.7)$$

Illustration is given below in figure 2.5. Suppose J_i has more than one child, then $D(v,J_i)$ is given as the minimum value regarding its child joints:

$$D(v,J_i) = \min\left(D_k(v,J_i), J_k \in Child(J_i)\right) \qquad (2.8)$$

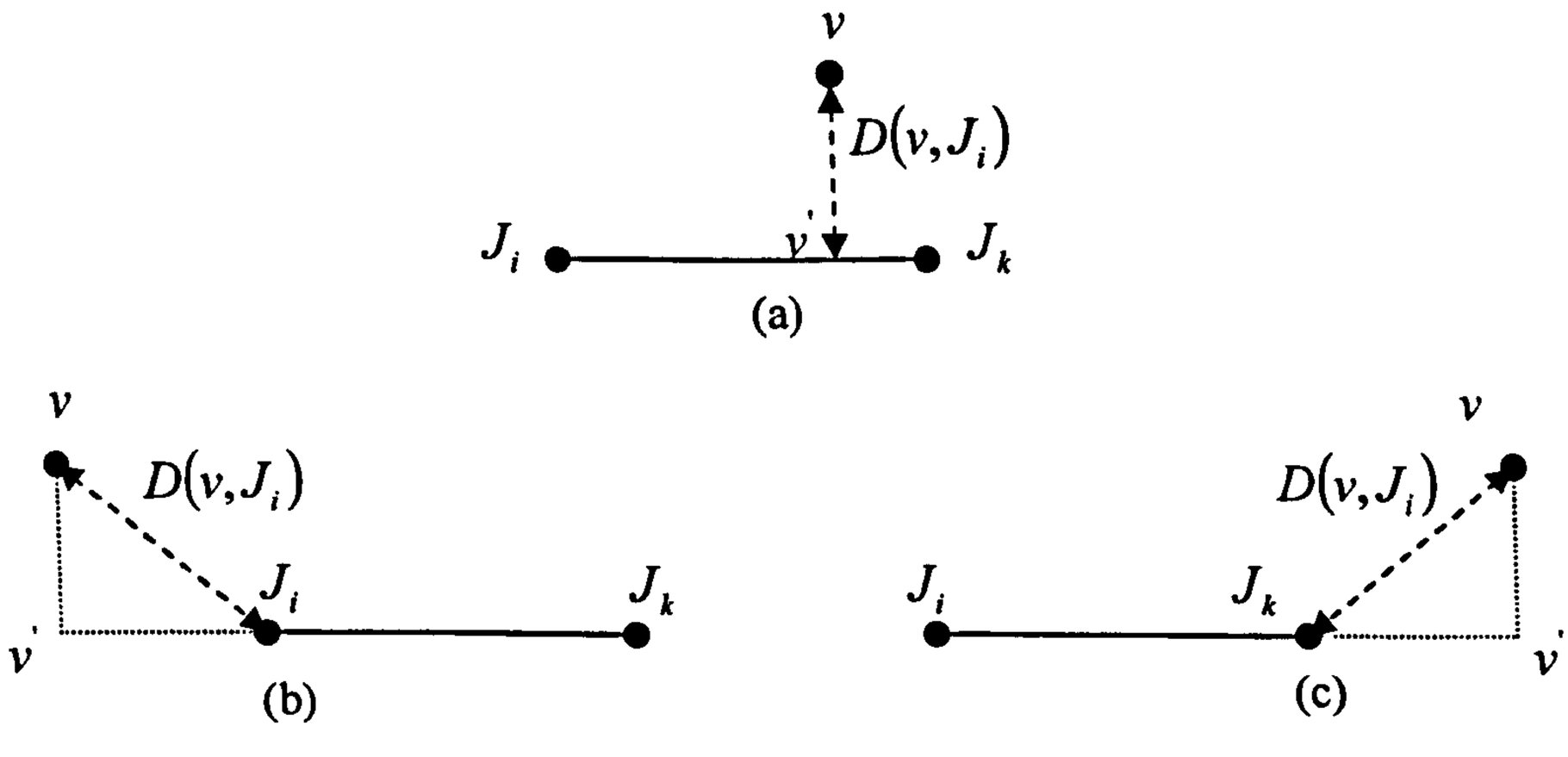


figure 2.5: An illustration of determining distance $D(v,J_i)$ between a vertex and the associated joint. $D(v,J_i)$ is represented by dashed line. (a) $0 \le \alpha \le 1$; (b) $\alpha < 0$; (c) $\alpha > 1$. [Yang and Zhang 2006b]

Therefore, suppose there are n joints in the influence set of a vertex v, for the

weight which is associated with i^{th} joint, Yang and Zhang [2006b] also gave out a computation equation according to the methods proposed in [Shepard, 1968]. In practice, the number of influence joint is chosen as 4 or 5 for good performance.

The result of the automatic guess made by the animation software is far from being 'reasonable' to use. Influences on a vertex brought by joints which are topologically far yet geometrically close result in bizarre effects during deformation runtime. A skin vertex might move while it should not because of redundant influences it gets from a neighbouring joint. A typical example is skin attachment around the armpit area of a human body, as shown in figure 2.6.

figure 2.6: An illustration of excess deformation caused by vertices getting irrelevant influences.

Animators usually have to manually tweak the weights, which is considered to be one of the most tedious and time-consuming tasks in production pipeline. To make it more difficult, an appropriate weight allocating in one pose is not always suitable for another pose. Therefore, the model needs to be exercised in various poses to verify the painted weights.

An algorithm to smooth out the weights automatically was proposed by Weber [2000]. This method establishes cross-sections at both ends of a joint link to

evaluate its local thickness. A threshold based on the local thickness is then defined to specify the maximal weight change allowed between two vertices which are tied to the same joint. This algorithm performs well in preventing radical weight changes between vertices. However, it does not prevent the distortions caused by vertex being affected by irrelevant joints.

The method proposed by Forstmann et al. [2007] based the weight computation on the twisted angle of a joint. It is able to create deformation details, such as wrinkles. However, the weights are computed for certain poses and not transferable for a global weight distribution.

More recently, Rohmer et al. [2008] proposed to compute the weight simply using the ratio of the length of a joint link to the distance between a vertex and the joint link. A ray tracing algorithm is hired in this method to determine influence bones. This algorithm is simple and presumably fast. However, it produces rather rigid transformations since the closest bone is likely to take up all the influences.

2.4.2 Example-based Attachment

A different solution that aims at weight distribution was adopted in [Mohr and Gleicher 2003]. This algorithm determines the weights by learning from examples. During a learning process, the pair of skeleton configuration and its matching skin geometry is fed to the system as an example pose. By solving a least-squares problem, the weights set of a joint are trained to fit the example. The learning process is repeated until the system finds the set of weights which produces the closest fit to the whole example sequences.

A similar idea can be found in the study of James and Twigg [2005] where to estimate weights is also done via least-squares-based approach. James and Twigg [2005] furthermore proposed to compute a weights set by solving the nonnegative least squares problem (NNLS) to reduce overfitting. Their approach was

suggested by Merry et al. [2006] and Wang et al. [2007] in their example-based weight training processes.

Weber et al. [2007] also proposed to fit weights through example-based training. However, the weights are computed by solving a Laplace equation in their methods.

The mutual problem of example-based weight fitting is that their quality depends on a considerable amount of well-established examples. It is unable to achieve its goal where these examples are not available,

2.4.3 Other Methods of Attachment

An interactive method was proposed in the article of Mohr et al. [2003]. It creates a possible deformation range for a vertex, and lets the user decide the exact position according to their needs. The weight is then computed from the given position. This method may produce an ideal result. Nevertheless, it needs a great deal of manual supervision.

In another article [Baran and Popović 2007], heat equation was used to find the ideal weights. The character body is regarded as an insulation within which heat conducts. The temperature is assigned 1 to the bone being dealt with, while other bones are assigned 0. As the heat diffuses over the surface, the temperature equilibrium at a vertex can be calculated. This temperature equilibrium is taken as the weight at the vertex. If a vertex is influenced by several bones, it takes heat contributed by other bones. This algorithm also embedded a distance field to decide whether the shortest line segment between a bone and a vertex is totally contained in the mesh. If not, the heat contributed from this bone is forced to 0 in order to eliminate unwanted influences. This method needs to store a pre-calculated distance field and solve a heat equation for each vertex regarding every bone. It is expensive in terms of both memory usage and computation.

2.4.4 Summary

In section 2.4, we have reviewed different weight computation algorithms developed previously, of which some are labour intense and some are data dependant. Most of them are designed to suit a particular deformation algorithm and non-transferable.

Our algorithm presented in the thesis otherwise provides an automatic and more flexible way for skin attachment, which not only benefits SSD, but also can be used in other deformation algorithms.

CHAPTER 3
AUTOMATIC SKELETONIZATION

Appropriate skeletonization for 3D characters is crucial since it lays the foundation of skeletal animation. Most existing skeleton generation techniques serve rather 'general' purposes other than animation. Some researches targeting animation have been carried out, e.g. by Aujay et al. [2007] and by Schaefer and Yuksel [2007]. Yet they all suffer from either computation overload or intense manual intervention that holds them back from being used directly in animation.

In this chapter, an automatic skeletoniaztion framework aiming at realistic skeletal character animation is discussed. The approach is based on curve skeleton extraction, which has been discussed by other researchers. However, our goal is not to produce a skeleton for general usage, but to present an efficient and easy-to-use technique especially for generating animatible skeleton of good quality. As reflecting the differences in purpose, the framework differs from previous ones in a few aspects.

3.1 Terminology

Our method of extracting an animation skeleton introduces a new geometric entity

called *3D silhouette*. It is an extension of silhouette which is conventionally a 2D concept. Given a polygonal mesh, the 3D silhouette records extra depth information than a 2D silhouette.

We denote a connective vertex set of a meshed model with V, and any vertex that belongs to V with $c(x,y,z)$. Without losing generality, we suppose the appropriate projection orientation with the least occlusion of this mesh model is along the Z axis and the 2D projection of V on the XY plane is P. For P, we can detect its 2D silhouette C' as a set of vertices that lie on the boundary of P. For each element $c'(x,y)$ in set C', we record its Z coordinate z_{c_i} from the corresponding vertex $c_i(x,y,z)$ in V. Here $c_i(x,y,z)$ forms a vertex of the 3D silhouette of V when the projection direction is along the Z axis. So a 3D silhouette C of mesh model V can be defined in mathematical term as follow:

$$C\{c_i(x,y,z) \mid c_i(x,y,z) \in V, c_i'(x,y) \in C', C' \subset P, x_{c_i} = x_{c_i'}, y_{c_i} = y_{c_i'}\} \qquad (3.1)$$

The primary 3D silhouette is defined as the 3D silhouette when the original 3D model is projected from an optimal orientation. To facilitate the generation of an ideal curve skeleton, it is important that the projection direction of the original 3D model should be selected so that it minimises the likelihood of occlusion. This is equivalent to making the model have approximately the maximal projection area. Thus we describe the optimal direction of projection as:

$$\left(\frac{\sum_{i=1}^{m} S_i \left\|\overrightarrow{n_{ix}}\right\|}{\sum_{i=1}^{m} S_i}, \frac{\sum_{i=1}^{m} S_i \left\|\overrightarrow{n_{iy}}\right\|}{\sum_{i=1}^{m} S_i}, \frac{\sum_{i=1}^{m} S_i \left\|\overrightarrow{n_{iz}}\right\|}{\sum_{i=1}^{m} S_i}\right) \qquad (3.2)$$

where m stands for the number of triangles in this mesh model, S_i for the area of the i^{th} triangle and $\overrightarrow{n_i}$ for its normal vector.

Other terms used for describing different objects and processes during the

extraction are explained as follow:

- Animation Skeleton: It is the skeletal structure used in animation, which consists of joints and bones (joint links).
- Curve Skeleton: It is a term often used in CAD (Computer Aided Design), graphical data compression and object topology analysis. No strict definition has been given before. Nevertheless, it might be regarded as a subset of the medial surface, as suggested in the work of Dey and Sun [2006].
- Skeletonization: It is the process of animation / curve skeleton generation.

3.2 Overview

It is known that the cross-sections of a common 3D animation character are approximately elliptic. These character models could be approximated using their 2D representations which have been produced from suitable projection angles. This fact suggests that extracting a 3D skeleton structure could be dealt with on 2D planes. Reducing of dimensions is supposed to provide an easier and faster solution.

Our skeletonization procedure begins with the construction of a 3D silhouette of the target model. Using constrained Delaunay triangulation, a curve skeleton is generated from the 3D silhouette, after which the positions of primary joints are located and the animation skeleton is produced by down sampling the curve skeleton. Therefore, this skeletonization framework can be summarized in 3 steps:

1. 3D silhouette detection
2. Curve skeleton extraction
3. Animation skeleton generation

Each step will be addressed in the following sections. An illustration of extracting a curve skeleton from its 3D silhouette is shown below in figure 3.1.

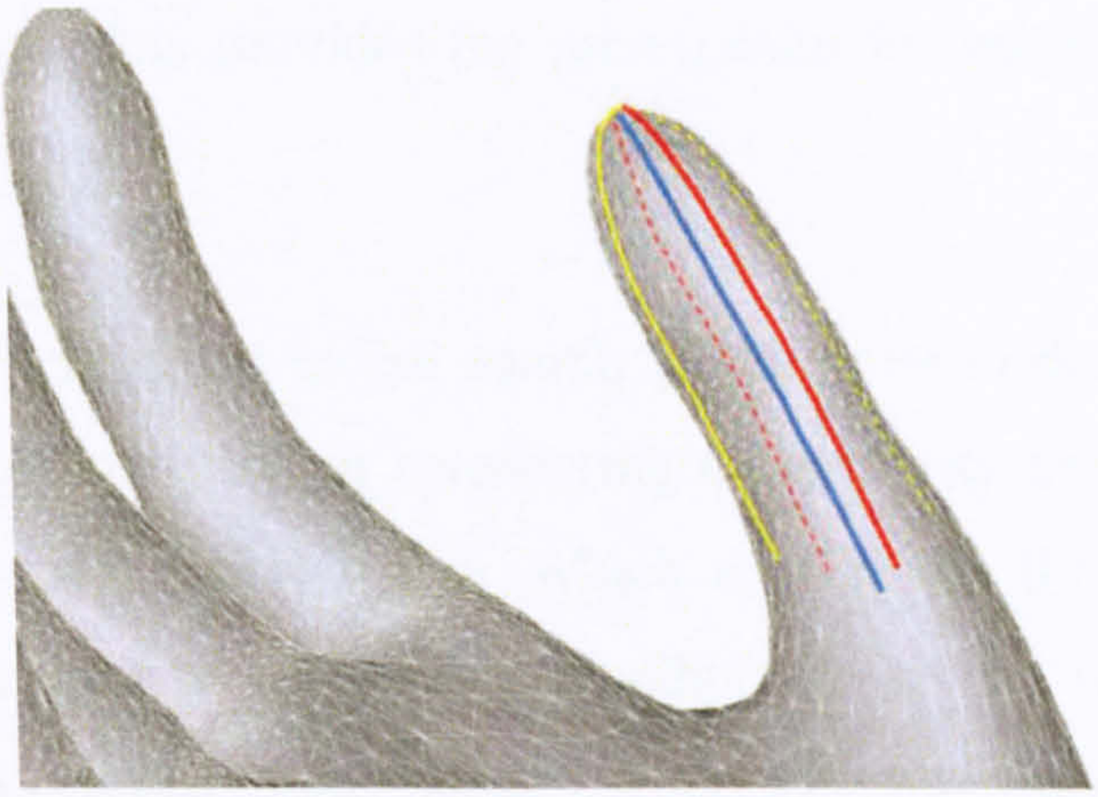


figure 3.1: Two different 3D silhouettes of the thumb are represented by red and yellow curves. Its 3D medial axes are represented by the blue curve.

3.3 Primary 3D Silhouette Detection

The optimal projection orientation demonstrated in this section is supposed to be along the Z axis. In computer animation, the binding pose will normally meet this requirement. If not, other orientations can be dealt with in the same way by rotating the model to make sure the optimal projection orientation align with the Z axis.

3.3.1 Finding the Start Vertex

The silhouette being detect is a closed boundary curve of the model with all the vertices belonging to the model lying inside. On the XY plane, the highest vertex, i.e. the vertex with maximum y value $y_{\max}$, is definitely on the boundary of the model projection. We label it as $c_1^{'}(x, y), c_1^{'} \in P$. This vertex is regarded as the start vertex of the silhouette set.

3.3.2 Two-Level Detection Algorithm

During a practical animation process, a model is usually given at a neutral pose, e.g. commonly-used Da Vinci Pose. In this case, the mesh is roughly extended, i.e.

with little occlusion. This provides the prerequisite for our two-level silhouette detection algorithm.

The Level 1 search is called global search which aims to determine the overall shape of a 3D silhouette without considering connectivity between vertices. The Level 2 search is called local search, which is to refine the 3D silhouette and connect all the vertices which belong to silhouette set C with regard to their connectivity in the original model mesh. For any character model that is deal with in this thesis, its 3D silhouette should result in a simple closed curve.

3.3.2.1 Global Search

The global search aims to find vertices which will decide the rough shape of a 3D silhouette. A further refinement will be given in the local search.

An illustration of a global search is given in figure 3.2. The search algorithm can be broken down to 4 steps.

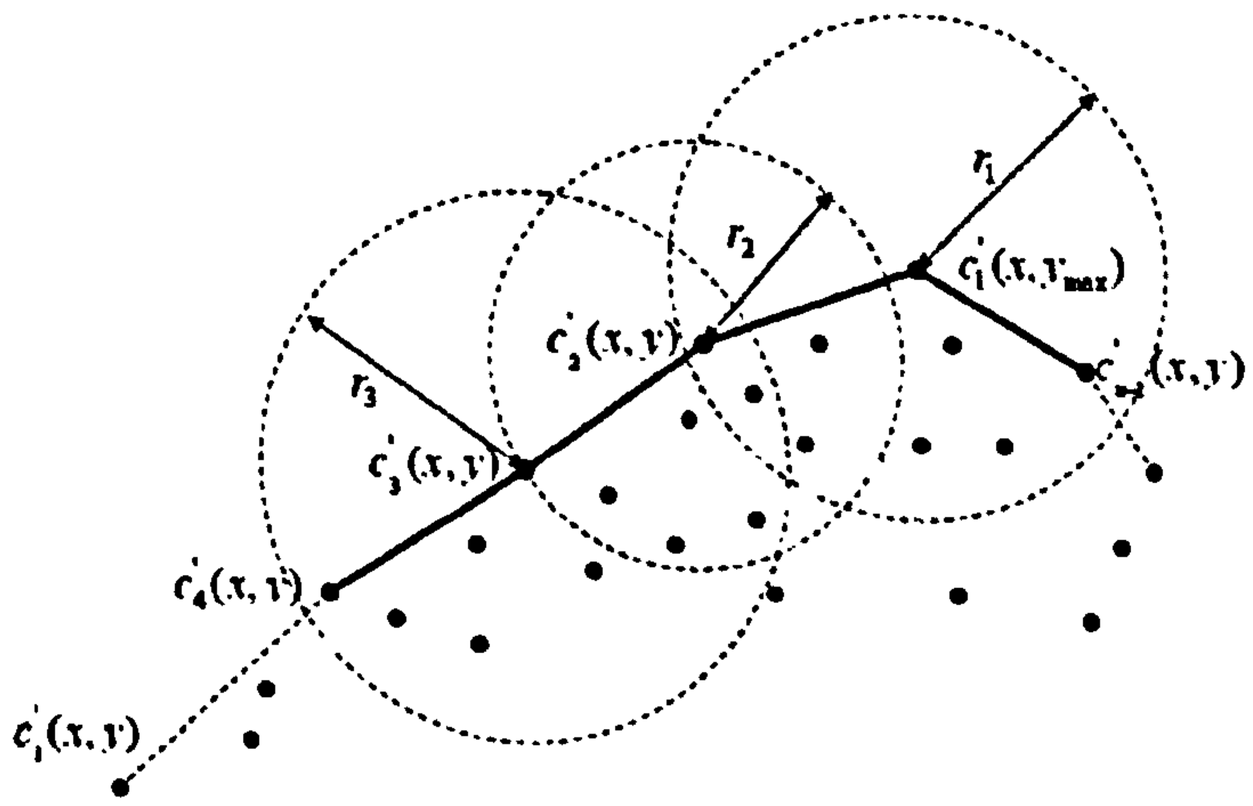


figure 3.2: Illustration of global search

Step 1: Finding the second vertex. Search for all the vertices in P whose distance to $c_1'(x, y)$ is less than a given value r. These vertices are regarded as a candidates set of the second silhouette vertex, recorded as T'. Each vertex is

labelled $t_j^{'}(x,y), t_j^{'} \in T^{'}$. Here all 'distance' mentioned in the global search is referred to as 2D Euclidean distance, and r is chosen as an experiential parameter which has a rather remarkable influence on accuracy and robustness of the global search. The value of r is affected by local vertex density. If the current local density is large, r should be considerably small. It helps to find more boundary vertices in order to express changes of the silhouette as proper as possible, and vice versa, if the current local density is small, increasing r will guarantee the potential silhouette vertex enters the search scope.

For each $t_j^{'}$, it can be regarded as lying on a vector pointing from $c_1^{'}$ to $t_j^{'}$. Regarding $\overrightarrow{OX}$ as the positive orientation of the polar axis, and $c_1^{'}(x,y)$ as the origin, we search for the vertex in $T^{'}$ on the vector with the least angle clockwise. This vertex is the second vertex of the 2D silhouette set, labelled as $c_2^{'}(x,y)$.

Step 3: Finding the third vertex. Search for all the vertices in P whose distance to vertex $c_2^{'}(x,y)$ is within r. Similarly, they are candidates of the third vertex in the silhouette set, recorded as $t_j^{'}(x,y) \in T^{'}$. Here the value of r is evaluated by the local vertex density as well.

$c_2^{'}$ is regarded as the origin, and $\overrightarrow{c_2^{'}c_1^{'}}$ as the positive orientation. Search among $T^{'}$ for the vertex with the maximal value of angle $\angle t_j c_2^{'} c_1^{'}$, calculating clockwise. This vertex is accordingly labelled as $c_3^{'}(x,y)$. It is the third vertex of the 2D silhouette.

Step 4: Finding the rest vertices. Repeat Step 3 until the vertex currently being detected $c_m^{'}(x,y)$ meets the start vertex $c_1^{'}(x,y)$ or their distance is below a given value.

Step 5: Extent 2D silhouette vertices to 3D. After all vertices of 2D silhouette set $C^{'}$ are discovered, we can extend them into 3D vertices in accordance with the definition given in equation (4.1) by acquiring for them the z coordinates, i.e. the depth information. By this means the primary 3D silhouette vertex set $C\{c_i(x,y,z)\big|c_i(x,y,z)\in V, c_i^{'}(x,y)\in C^{'}, x_i = x_i^{'}, y_i = y_i^{'}, z_i\}$ is constructed.

During the global search, an experimental self-adaptive equation is often used to evaluate r, shown below:

$$r = \max\left\|t_j^{'}(x,y) - c_k^{'}(x,y)\right\| \qquad (3.3)$$

where $t_j^{'}(x,y)$ is the neighbour vertex connected to the current vertex $c_k^{'}$.

Figure 3.3 shows an example of a hand model after a global search. The original model is shown in figure 3.8(a).

figure 3.3: Discrete 3D silhouette vertices detected after a global search

3.3.2.2 Local Search

Global search produces a set of discrete vertices of which the 3D silhouette consists. The next level of the algorithm is the local search which is to link all these vertices in accordance with their connectivity in the original model mesh V. As mentioned previously, the 3D silhouette is one closed simple curve. That is to

say, the first and the last vertex of the silhouette set should meet. Moreover, there are no intersections when connecting the vertices together.

The local search can be regarded as searching for the approximate shortest route within the mesh for each vertex pair $c_i(x,y,z)$ and $c_{i+1}(x,y,z)$. Suppose there are n vertices in silhouette vertex set C, then a loop as follow can be used to described the linking process:

For (i from 1 to n)

Connect $(c_i(x,y,z),c_{i+1}(x,y,z))$;

We also break down this *Connect* $(c_i(x,y,z),c_{i+1}(x,y,z))$ operation into steps.

Step 1: Starting from c_i, find its connecting vertices in the original mesh. These vertices are then recorded as a connecting set T.

Step2: Each $t_i \in T$ will be first compared with c_{i+1}. If any t_i is found the same as c_{i+1}, that means c_i and c_{i+1} are connected originally. During local search, they can be connected directly. Otherwise, calculate the each distance between c_{i+1} and every t_i. The vertex closest to c_{i+1} is chosen as the next connecting vertex with c_i. This vertex is labelled as c_{i1}.

Step 3: Start from c_{i1}, find all its connecting vertices in the original mesh and repeat step 2 until a vertex the same as c_{i+1} is found or the distance between one vertex and c_{i+1} is below a given value.

The local search method is effective, yet without the general estimation of the silhouette from global search, it usually falls into an infinite loop.

3.3.3 Detection Algorithms Using Linkage Information

As mentioned previously, in animation practice, the character model is usually given at a roughly extended pose with little occlusion, which will well satisfy the prerequisites for the above presented two-level search algorithm. However, the two-level algorithm may fail in finding the appropriate silhouette in some extreme cases where parts of a model are highly occluded. Therefore, another detection algorithm is presented in this section. This algorithm is capable of dealing with general situations without the restriction of poses. However, it may sacrifice efficiency to achieve this generalization.

3.3.3.1 Initial Algorithm

The character model mesh used in animation is composed of a set of connected vertices. The silhouette vertex set is expected to resemble this connectivity. In the two-level search algorithm above, the rough shape of a silhouette and its connectivity are considered in separate processes. In this section, another detection algorithm will be devised to take into account the connectivity of the silhouette whilst considering its shape. This is achieved by examining those connected to the current silhouette vertex. That is to say, our initial attempt of finding the next silhouette vertex is based on the assumption the next silhouette vertex is among one of the vertices connected to the current one.

Under the assumption given above, the second vertex of the 3D silhouette is one of connecting vertices of $c_1^{'}$. Supposing there are n vertices connected to $c_1^{'}$, we label them as $t_j^{'}, j \in (1,n)$. M is used to mark a point chosen lying on the line $y = y_1^{'}$. This point does not have to really exist in the original model mesh vertices. It only matters that a vector $\overrightarrow{c_1^{'}M}$ which is parallel to the X axis needs to be established and shares the same positive direction as the X axis.

$c_1^{'}$ is chosen by having the largest y coordinate value among the projection vertex set $C^{'}$, which indicates that any other vertices on the projection plane situate

below the line $y = y_1^{'}$, i.e. when $c_1^{'}$ is chosen as the origin, $\overrightarrow{c_1^{'}M}$ as the positive orientation, calculated clockwise, it is impossible for any angle $\alpha_j^{'}$ formed by $\overrightarrow{c_1^{'}M}$ and $\overrightarrow{c_1^{'}t_j^{'}}$ to have a value outside $(0,\pi)$, as illustrated in figure 3.4.

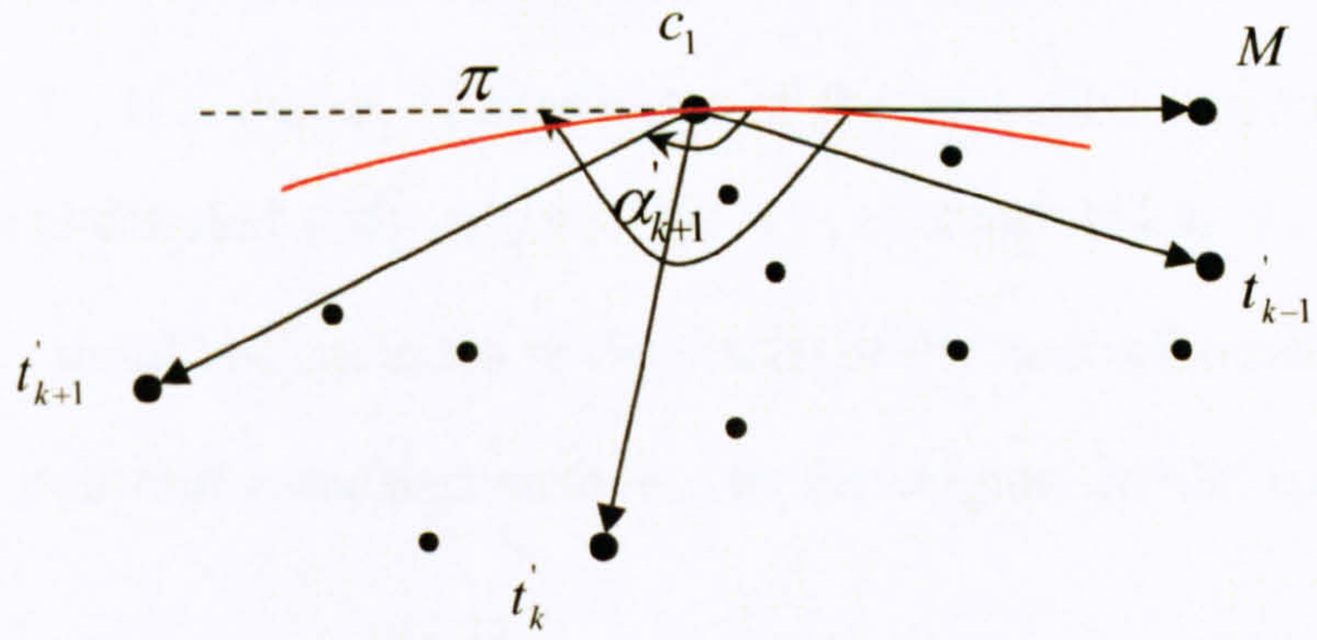


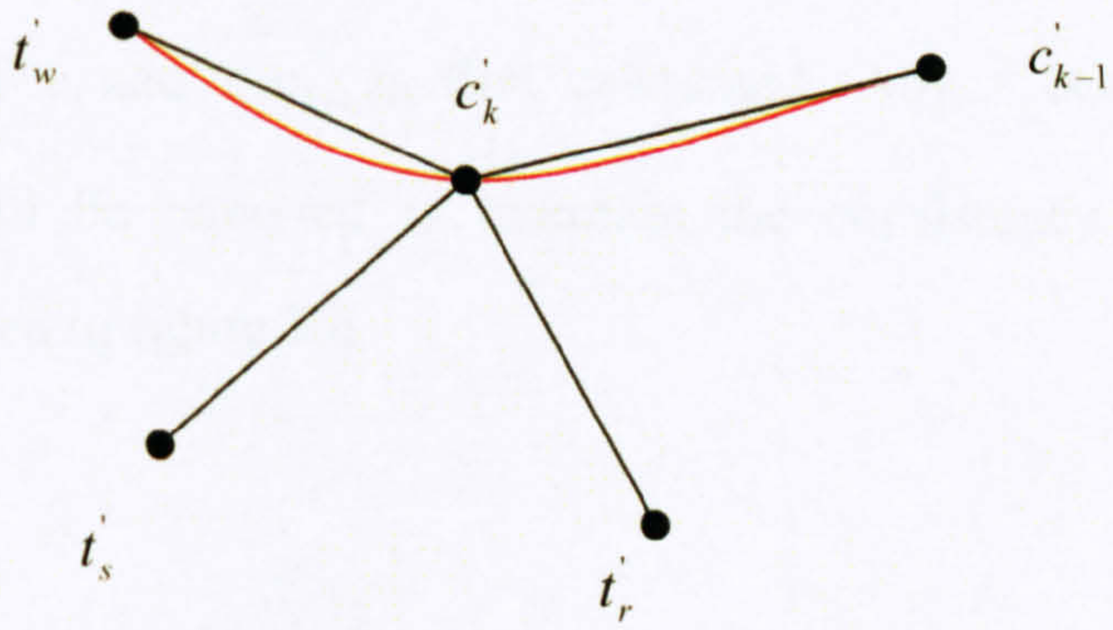
figure 3.4: Angles formed by $c_1^{'}$ with its connecting vertices. The silhouette is represented by the red curve. $c_1^{'}$ is the start vertex of the silhouette set and $t_{k-1}^{'}, t_k^{'}, t_{k+1}^{'}$ are among its connecting vertices. Let $\overrightarrow{c_1^{'}M}$ be the positive orientation and $c_1^{'}$ be the origin, following clockwise, any angle $\angle t_i^{'} c_1^{'} M$ is less than 180 degrees.

With all the rest of the vertices below $c_1^{'}$, the silhouette curve segment at $c_1^{'}$ becomes convex. This is also illustrated in figure 3.4. Every connecting vertices of $c_1^{'}$ can be used to construct a vector $\overrightarrow{c_1^{'}t_j^{'}}$ pointing from $c_1^{'}$ to $t_j^{'}$, the vector which has the largest angle calculated from $\overrightarrow{c_1^{'}M}$ extends to the outmost of the vertex set. The vertex which forms this vector therefore lies on the boundary of the projection, following an anticlockwise direction. Thus we can calculate the angle $\alpha_j^{'}$ between $\overrightarrow{c_1^{'}M}$ and $\overrightarrow{c_1^{'}t_j^{'}}$, and the vertex with the biggest $\alpha_j^{'}$ is recorded as the second silhouette vertex. It is labelled as $c_2^{'}$.

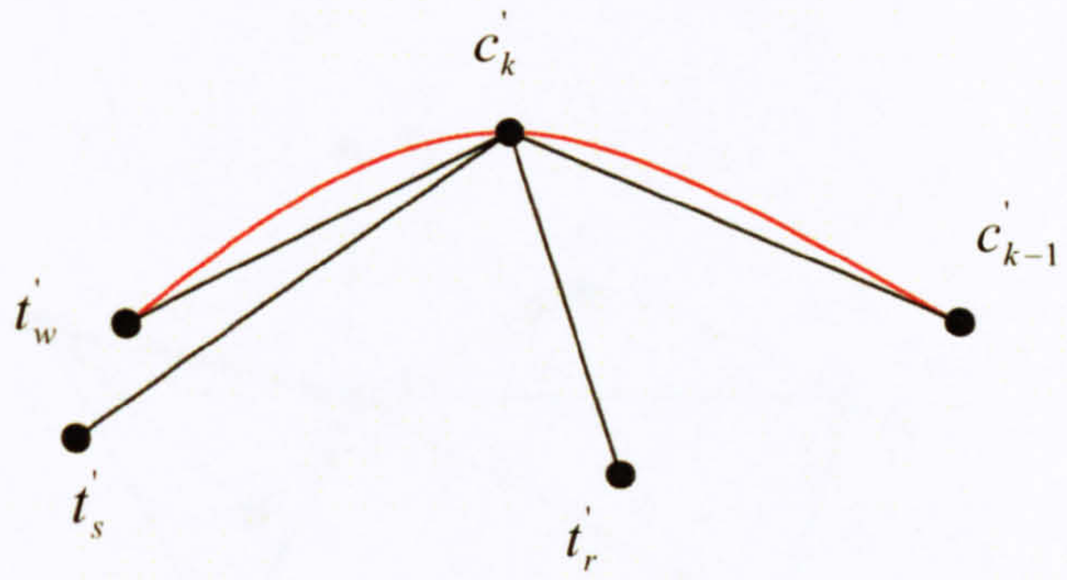
After finding out the first two silhouette vertices, the means of searching for the remaining vertices of the silhouette set is repeatedly the same.

Supposing the current silhouette vertex is $c_k^{'}$, its previous silhouette vertex is $c_{k-1}^{'}$, and there are n vertices connected to $c_k^{'}$. The set of connecting vertices is denoted with $T^{'}$. It is the set of candidates of the next silhouette vertex. Each of those vertices is denoted with $t_j^{'}, j \in (1,n)$. $c_{k-1}^{'}$ is connected to $c_k^{'}$, so $c_{k-1}^{'} \in T^{'}$. However, $c_{k-1}^{'}$ should be excluded in the search of the next silhouette vertex. Thus for each $t_j^{'}$, it is first compared with $c_{k-1}^{'}$ in the original model mesh to remove duplicate.

The initial side is always considered as $\overrightarrow{c_k^{'}c_{k-1}^{'}}$, and the angle between two vectors is always calculated clockwise. As shown below in figure 3.5(a) and (b), either the curvature at $c_k^{'}$ is convex or concave, the vertex lying on the vector which forms the largest angle with $\overrightarrow{c_k^{'}c_{k-1}^{'}}$ is always on the boundary of the projection.



(a)



(b)

figure 3.5: Curvature at a silhouette vertex $c_k^{'}$. The red curve represents the silhouette. (a)The curvature of silhouette at $c_k^{'}$ is concave. (b)The curvature of silhouette at $c_k^{'}$ is convex. $s,r,w \in (1,n)$.

For each connecting vertex $t_j^{'}$, the angle $\alpha_i^{'}$ between the vector $\overrightarrow{c_k^{'}t_j^{'}}$ and vector $\overrightarrow{c_k^{'}c_{k-1}^{'}}$ is calculated clockwise. The vertex lying on the vector which bears the largest angle would be the next silhouette vertex. Moreover, to ensure that the search process as a whole moves anticlockwise, another rule is brought into the algorithm for determining the silhouette vertex. The angle $\alpha_{nk}^{'}$ between the normal vector $\overrightarrow{n_k^{'}}$ of $c_k^{'}$ and $\overrightarrow{c_k^{'}c_{k-1}^{'}}$ is first calculated. Any $t_j^{'}$ bearing an angle $\alpha_i^{'}$ exceeding $\alpha_{nk}^{'}$ will be removed to maintain the consistency of the search direction, as illustrated in figure 3.6.

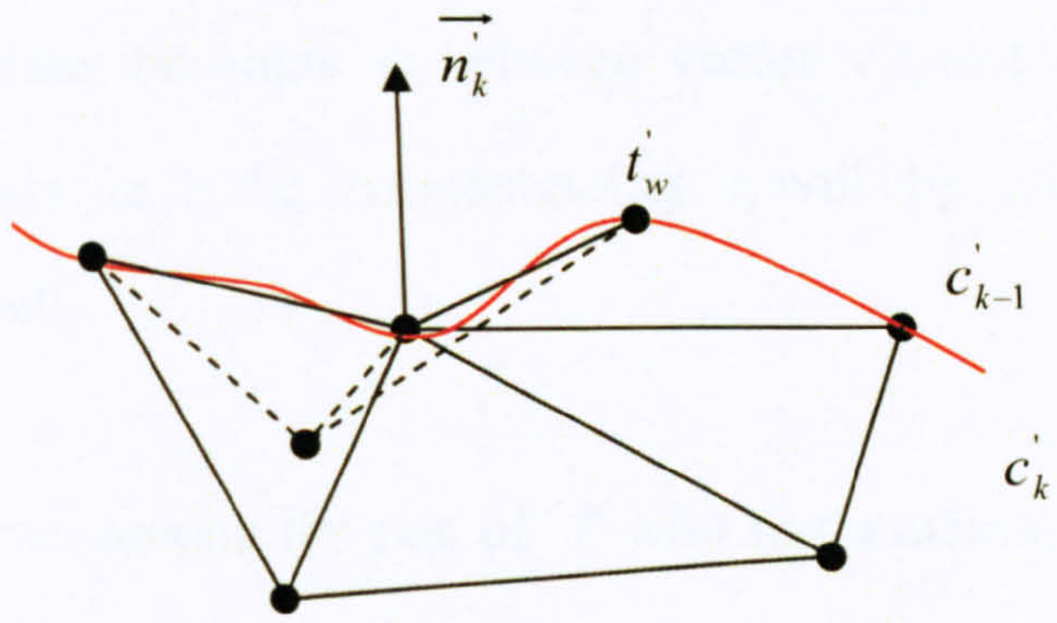


figure 3.6: Removing unsuitable vertices using $\overrightarrow{n_k}$ *. The silhouette is represented by the red curve. During the search process,* t_w *will be removed to maintain a constant search direction*

Therefore, the search algorithm to accomplish a 3D silhouette using its connectivity information can be described in 5 steps as follow:

Step 1: Construct a vector $\overrightarrow{c_1 M}$ which is parallel to X axis. Find in the original model all the vertices connected to c_1. Each vertex t_i in the projection of those connecting vertices is used to form a vector with c_1. Calculate the angles between $\overrightarrow{c_1 M}$ and $\overrightarrow{c_1 t_i}$. The vertex lying on the vector bearing the largest angle with $\overrightarrow{c_1 M}$ is selected as the second silhouette vertex, recorded as c_2.

Step 2: For any vertex $c_k(x, y)$, find in the original model all the vertices connected to c_k. The projected vertex set of those connecting vertices is labelled T as the candidates set of the next silhouette vertex. Compare $t_i \in T$ with the previous silhouette vertex c_{k-1}. The vertex which is found the same as c_{k-1} will be eliminated from the candidates set.

Step 3: Calculate the angle $\alpha_k^{'}$ between the normal vector of $c_k^{'}$ and vector $\overrightarrow{c_k^{'}c_{k-1}^{'}}$. Also calculate the angle $\alpha_i^{'}$ between vector $\overrightarrow{c_1^{'}t_i^{'}}$ and vector $\overrightarrow{c_k^{'}c_{k-1}^{'}}$. For any $\alpha_i^{'}$ that exceeds $\alpha_k^{'}$, the corresponding $t_i^{'}$ will be eliminated from the candidates set as well.

Step 4: Find a vertex among the rest of $T^{'}$ with the maximal angle with $\overrightarrow{c_k^{'}c_{k-1}^{'}}$. This vertex will be selected as the next silhouette vertex, recorded as $c_{k+1}^{'}$.

Step 5: Repeat step 2 and step 3. After each silhouette vertex is discovered, it will be checked against $c_1^{'}$. When this silhouette vertex meets $c_1^{'}$, the detection is completed. The silhouette is joined into a closed curve.

The detection algorithm makes use of the vertex connectivity of the original model mesh. This connectivity does not change along with model postures and it remains constant after projection. Thus it presumably is applicable to any postures without concerning occlusions and the Euclidean distance between vertices may affect the search result.

During implementation, we found that the algorithm works well in many cases with arbitrary projection orientations, yet in other cases it is sensitive to projection orientation. When a model is projected from a certain angle, it produces a false silhouette result, or the implementation crashes and produces no result at all. Though this problem can always be solved by adjusting the projection angle, it weakens the robustness of this algorithm.

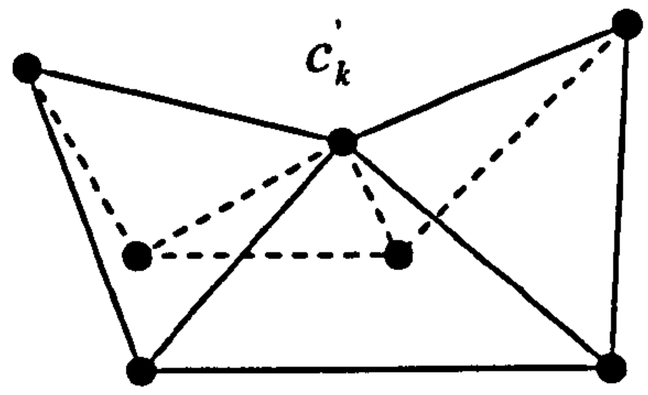
The false results are due to the criterion which employs the angle $\alpha_k^{'}$ between the normal vector $\overrightarrow{n_k^{'}}$ of the current vertex and $\overrightarrow{c_k^{'}c_{k-1}^{'}}$ as the threshold. It is not guaranteed to decide the proper silhouette vertices without exceptions. For

example, if the curvature at $c_k^{'}$ suffers an acute change, the angle with $\overrightarrow{c_k^{'}c_{k-1}^{'}}$ which is formed by a vertex irrelevant to the silhouette set may still be below $\alpha_k^{'}$, yet this vertex will be selected as the next silhouette vertex. Vice versa, if the curvature only suffers an acute change at $c_k^{'}$ while maintaining smooth changes over a large scale, under certain projection orientation, the angle formed by the true silhouette vertex may exceed $\alpha_k^{'}$. In this case, the next true silhouette vertex is removed and a false vertex is chosen. Both situations may bring the search process towards the wrong direction. In other words, the convergence of the initial algorithm is beyond provability.

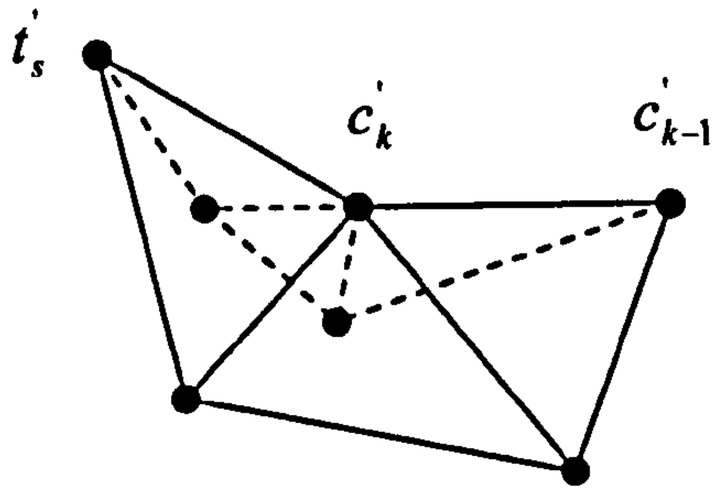
3.3.3.2 Modified Algorithm

The initial algorithm aims to identify the next silhouette vertex $c_{k+1}^{'}$ among the connecting vertices of $c_k^{'}$ by the angles they formed with $\overrightarrow{c_k^{'}c_{k-1}^{'}}$. However, it fails under certain projection orientation. As mentioned above, the reason which accounts for this failure is the insufficiency of its selecting criteria. To overcome this orientation sensitivity, a modified algorithm is developed in this section.

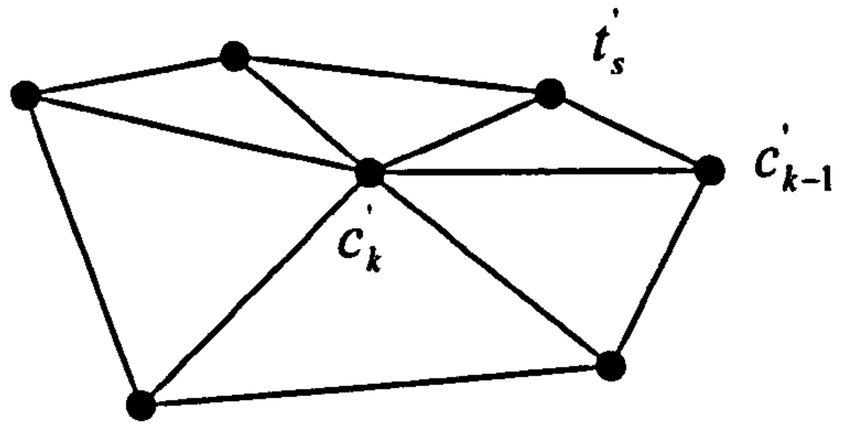
The feasibility of the selecting criterion lies on appropriately determining maximal angle among all the angles formed by the connecting vertices. In the original model mesh, $c_k^{'}$ with its connecting vertices forms a series of non-overlapping triangular facets. Each triangle shares a common side, i.e. two common vertices, with another two triangles respectively, shown below in figure 3.7(a). Regardless of the projection orientation, the connectivity maintains its topological properties, as illustrated in figure 3.7(b) and (c). The modified algorithm is capable of distinguishing between these two situations.



(a)



(b)



(c)

figure 3.7: An illustration of projection of c_k'. (a) connectivity in the original model. (b) c_k' is projected on the boundary. (c) c_k' projected inside.

In figure 3.7(b), t_s' is the desired next silhouette vertex. While in figure 3.7(c), t_s' should be eliminated from the candidate set. The adjustment converts $\angle t_j' c_k' c_{k-1}'$ in order to distinguish these two situations.

The adjustment follows the order of the triangular connectivity. It can be described in 5 steps:

Step 1: Regard c_k' and c_{k-1}' as two vertices of a triangle, according to the linkage information given in the original model mesh, the third vertex that composes this triangle can be found. This vertex is labelled as t_1', the angle formed by $\overrightarrow{c_k' t_1'}$ and $\overrightarrow{c_k' c_{k-1}'}$ is accordingly labelled α_1'.

Step 2: Regarding $c_k^{'}$ and $t_1^{'}$ as two vertices of the next triangle, find the third vertex of the next triangle and label it $t_2^{'}$. Accordingly, $\alpha_2^{'}$ is found and labelled. Repeat this process, until all the connecting vertices and the angles they formed are labelled.

Step 3: Consider the value of $\alpha_1^{'}$. If it is below 180°, the adjusted angle $\alpha_1^{''} = \alpha_1^{'}$, and then move to $\alpha_2^{'}$. If it exceeds 180°, its adjusted angle $\alpha_1^{''} = \alpha_1^{'} - 360^{\circ}$, and then move to $\alpha_2^{'}$.

Step 4: For each $\alpha_i^{'}$, compute the value difference between $\alpha_i^{'}$ and $\alpha_{i-1}^{'}$. If it is less than 180°, this $\alpha_i^{'}$ remains unaltered; if not, it will be replaced with $\alpha_i^{''}, \alpha_i^{''} = \alpha_i^{'} - 360^{\circ}$.

Step 5: Compare all $\alpha_i^{''}$. The corresponding vertex of the largest angle is the next silhouette vertex.

The example of the final detected 3D silhouette of a hand model is shown below in Figure 3.8(b).

3.4 Skeleton Extraction

In our framework, the animation skeleton is generated by down sampling the curve skeleton. Thus we first need to extract a curve skeleton from the 3D silhouette. After determining skeletal joints along the curve skeleton and joining them with line segments, the animation skeleton is then established. The entire process is illustrated in figure 3.8.

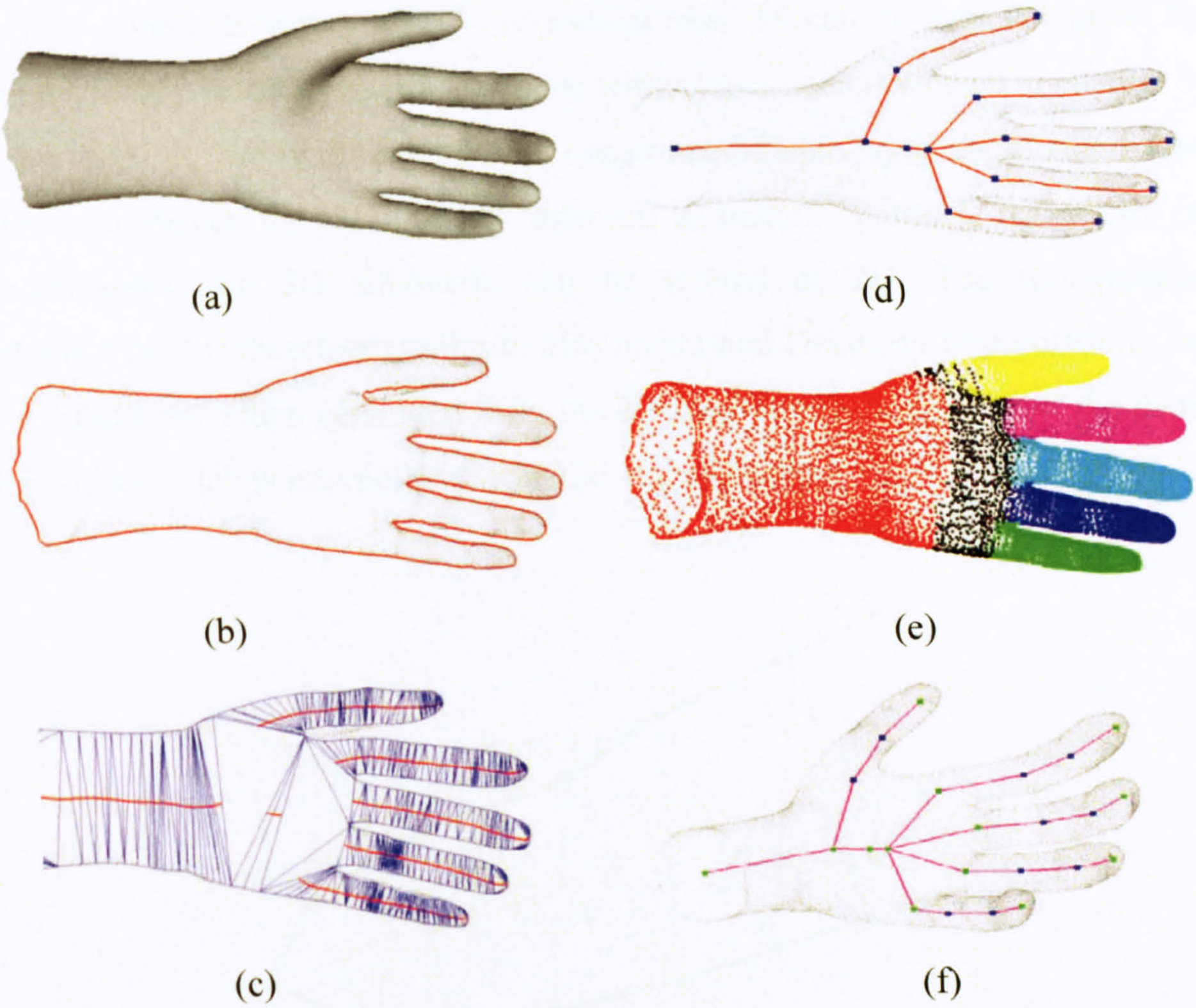


figure 3.8: Animation skeleton generation process.

(a) Original model

(b) Primary 3D silhouette

(c) 3D medial axis of hand by constrained Delaunay triangulation

(d) Coarse curve skeleton and key skeletal joints

(e) Decomposition

(f) Animation skeleton and skeletal joints

3.4.1 Curve Skeleton Extraction

Once a 3D silhouette of the model is drawn, it can be divided into a set of triangles based on constrained Delaunay triangulation. The curve skeleton is then generated by down-sampling the medial axis of the triangulated silhouette.

3.4.1.1 Constrained Delaunay Triangulation

For a 2D silhouette vertex set $C^{'}$, its medial axis $M^{'}$ can be detected easily by several Voronoi-based geometric methods which have been reviewed in chapter 3. In this section, we will extend the constrained-Delaunay-triangulation-based method to detect the 3D medial axis of a model. Without regard to its Z coordinates, the 3D silhouette can be treated as 2D. The triangulation performed on it is therefore similar to 2D constrained Delaunay triangulation. The 3D medial axis is then generated from the 2D medial axis coupled with the depth information which is interpolated from the 3D silhouette.

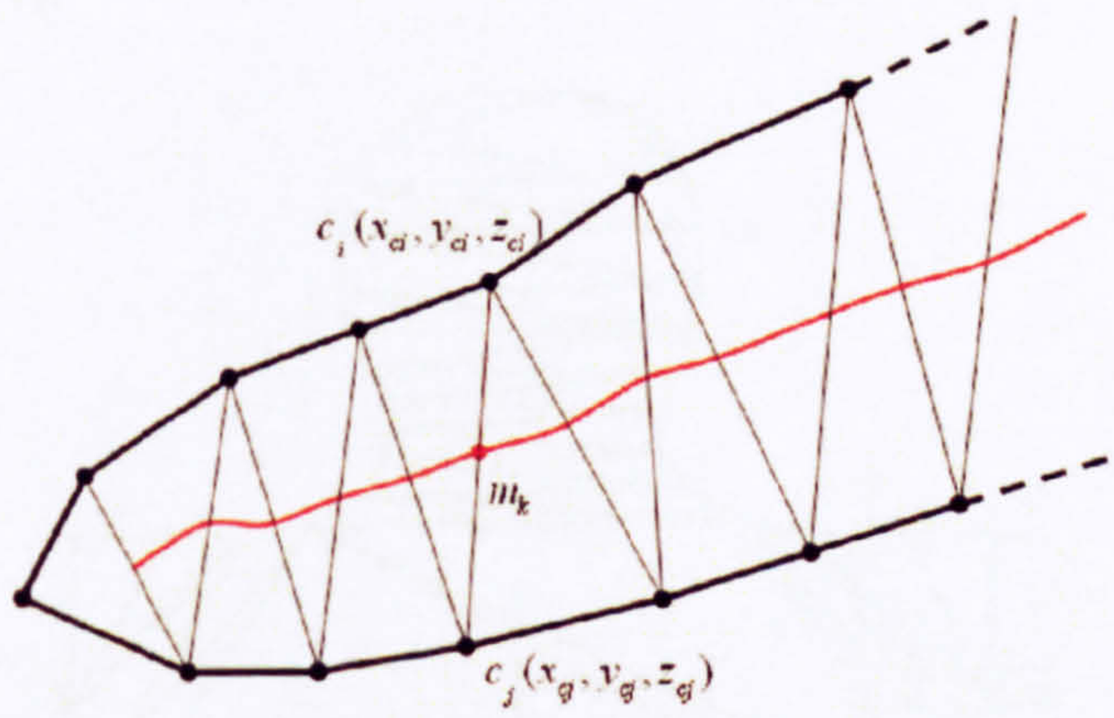


figure 3.9: Illustration of extracting the 3D medial axis from a 3D silhouette.

The triangulation process is illustrated in figure 3.9. Suppose the vertices which consists of a triangle edge are $c_i(x,y,z)$ and $c_j(x,y,z)$, without considering its z coordinates, it is easy to work out the 2D midpoint $m_k^{'}(\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2})$ of the edge $c_i^{'}c_j^{'}$. The interpolated z value is then put back to form a vertex of the 3D medial axis for the 3D silhouette. Therefore, along with the 2D medial axis $M^{'}$, the 3D medial axis M is also discovered. In mathematical terms, the 3D medial axis M follows:

$$M\{m_k(x,y,z) \mid m_k^{'} \in M^{'}, x_k = x_k^{'}, y_k = y_k^{'}, z_k = \frac{z_i + z_j}{2}\} \quad (3.4)$$

Two types of edges can be distinguished after constrained Delaunay triangulation: external edge and internal edge. The triangles are then divided into three categories with regard to their edge type:

1. Terminal Triangles: with two external edges and one internal edge;
2. Sleeve Triangles: with one external edge and two internal edges;
3. Junction Triangles: without external edges.

The different types of edges and triangles they form are illustrated with different colours in figure 3.10:

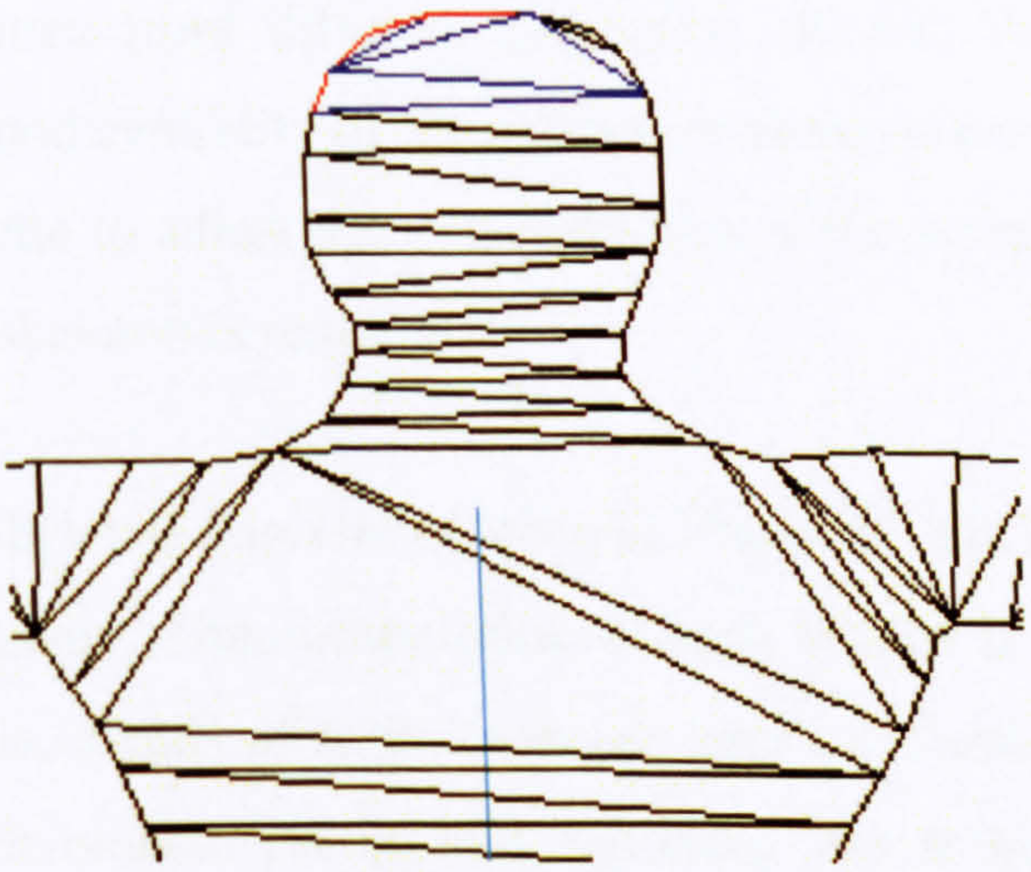


figure 3.10: Two types of edges after constrained Delaunay triangulation are represented by two different colours. The red represent external edges and the blue represent internal edges. They form triangles of three categories.

A medial axis is obtained by connecting the midpoint of the internal edges [Igarashi et al. 1999]. It ends when it meets the large junction triangles. The experimental result of constrained Delaunay triangulation and 3D medial axis detection of a hand model is shown in figure 3.8(c). The medial axis produced from the primary 3D silhouette consists of several segments. We regard it as a coarse curve skeleton.

3.4.1.2 Decomposition

In character animation, the centrality of an animation skeleton, i.e. all joints and bones lie on the central path of the body segments, is of great importance. The coarse curve skeleton is centred in the particular projection plane which it is produced from, yet some part of it may not be centred in other projection directions, since the actual z coordinates of the curve skeleton points are not considered during coarse curve skeleton generation.

As we stated earlier in this chapter, an accurate curve skeleton can be determined by two 3D silhouettes from different projection planes. To centre the curve skeleton to obtain good centrality of the animation skeleton extracted from it later, a second 3D silhouette to adjust the z coordinates of the points for each segment of the coarse curve skeleton is required.

The coarse curve skeleton consists of several branches. Each branch should be centred after adjustment. Thus information of each branch is required. However, projected as a whole, details of some branches may be missed. In order to obtain information of each branch, the second silhouette has to be detected from the model which is decomposed according to its coarse curve skeleton branches.

During decomposition, each vertex of the original mesh is assigned to a certain branch using a simple but effective method which is similar to the nearest-neighbour algorithm in pattern recognition. First of all, the distance between a mesh vertex and every point on each coarse curve skeleton is calculated. For the nearest-neighbour algorithm, the minimum distance value presumably assigns the vertex to the segment which this point belongs to. However, there could be situations that lead to false categorizations. For example, consider the hand model presented in figure 3.8(c). If the arm is too strong, it might cause a vertex that actually belongs to the arm part to have a less distance value to a point

situated on the coarse curve skeleton of the finger than to any points on the coarse skeleton of the arm.

A modification is made to eliminate this error. We calculate the distance between the corresponding point and the 3D primary silhouetted. For each point, its two distances are compared to discover the differences. The minimum value is used to determine which segment the vertex belongs to. In general, this algorithm is formulated as:

$$s(v_i,k)=\min_{j=(1,n)}^{k}\left|\ distance(v_i,m_{k,j}(x,y,z))-l_{k,j}\right| \quad (3.5)$$

where k stands for the sequence number of the coarse curve skeleton branch, v_i for the i^{th} vertex of the model mesh and $m_{k,j}(x,y,z)$ for the j^{th} point of the k^{th} branch. $distance(v_i,m_{k,j}(x,y,z))$ is the Euclidean distance between v_i and $m_{k,j}$. $l_{k,j}$ is the distance between $m_{k,j}$ and the primary silhouette. The vertex v_i will be classified into the k^{th} branch which bears the minimum distance difference. Figure 3.8(e) shows the decomposition result with seven parts of the hand.

A vertex lying on the silhouette is assigned to the branch which it is connected through constrained Delaunay triangulation. Other vertices are presumably to be checked against all the branches of the coarse curve skeleton to find the minimum distance difference. To improve efficiency, an approach which involves fewer calculations is employed. This approach is based on the condition that all vertices in a model mesh are connected. Choosing any vertex in the silhouette set as the root, all vertices can be traversed, and thus categorized, through connectivity.

Suppose v is a classified vertex, b_k is its classified branch, and T is its connecting vertex set. The neighbourhood branches of b_k can be identified by studying the junction triangles at both ends of b_k, and we denote them with p_j.

For any vertex $t_j \in T$, it belongs to either b_k or p_j. Thus we only need to compute its distance to b_k and to p_j for the comparison proposed in equation (3.5).

3.4.1.3 Curve skeleton refinement

After decomposition, we now need to detect the 3D silhouette for each segment of the object in the second projection direction. This detection is for the purpose of fine tuning the coordinates of the coarse curve skeleton.

When the second projection plane is chosen, it is preferred that the projection orientation is perpendicular to the plane from which the primary silhouette is produced. We calculate the corresponding projection orientation of the k^{th} segment of the model using equation (3.6):

$$\overrightarrow{p_k} = \vec{p} \times \overrightarrow{m_{k-end} - m_{k-start}} \qquad (3.6)$$

where $\overrightarrow{p_k}$ stands for the projection vector of the k^{th} segment of the model. $\vec{p}$ is the projection vector of the plane form which the primary 3D silhouette is generated, $\overrightarrow{m_{k-start}}$ and $\overrightarrow{m_{k-end}}$ stand for the start point and the end point of the k^{th} coarse curve skeleton segment.

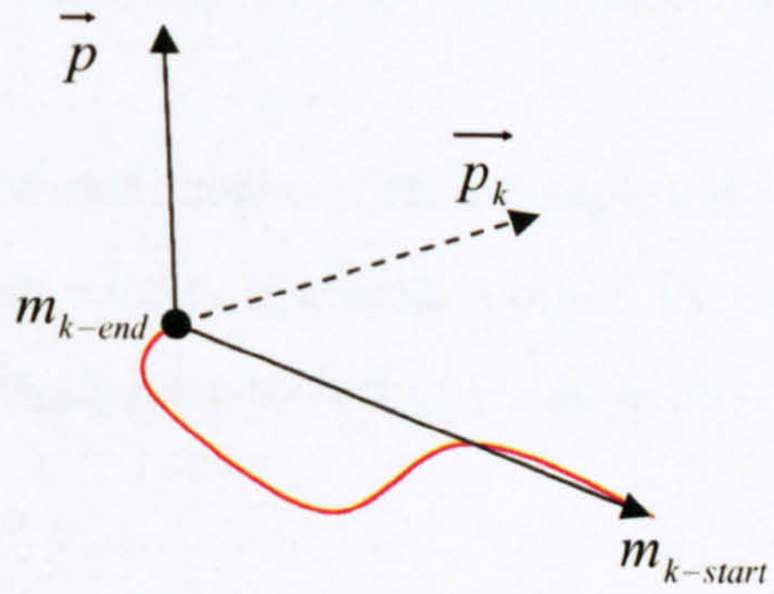


figure 3.11: An illustration of calculating the second projection vector of the k^{th} segment of the coarse curve skeleton. The red curve represents the k^{th} segment.

As illustrated in figure 3.11, the second projection vector is perpendicular to $\overrightarrow{m_{k-end} - m_{k-start}}$ as well. By this means, it is most possible to make the second projection fully extended hence it is most likely to obtain the maximum projection area.

The detection of the 3D silhouette for each decomposed part of the model is the same as what we described in section 4.3. In most cases, the global search is effective enough to produce the desired result; however, the local search or the other detection algorithm is always applicable if necessary. Figure 3.12 illustrates the second 3D silhouette for each segment of a hand model. Each branch is self-closed by simply joining projected vertices lying on the end of the decomposed mesh.

figure 3.12: The second 3D silhouette for six segments of a hand model

The second 3D silhouette for each curve skeleton segment is used to fine tune the z coordinates of the coarse curve skeleton points in order to acquire good centrality. Figure 3.13 illustrates how to refine a coarse curve skeleton through the second 3D silhouette.

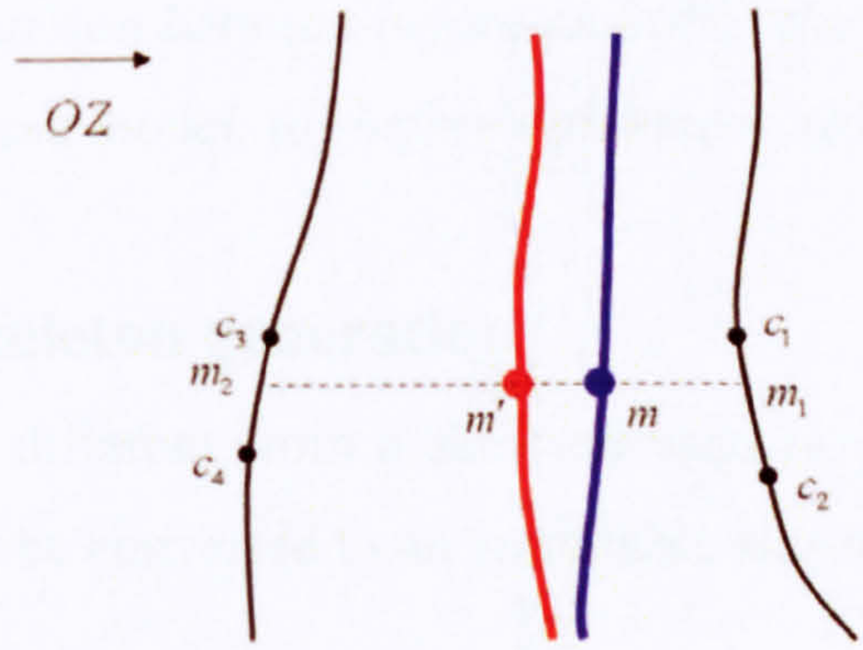
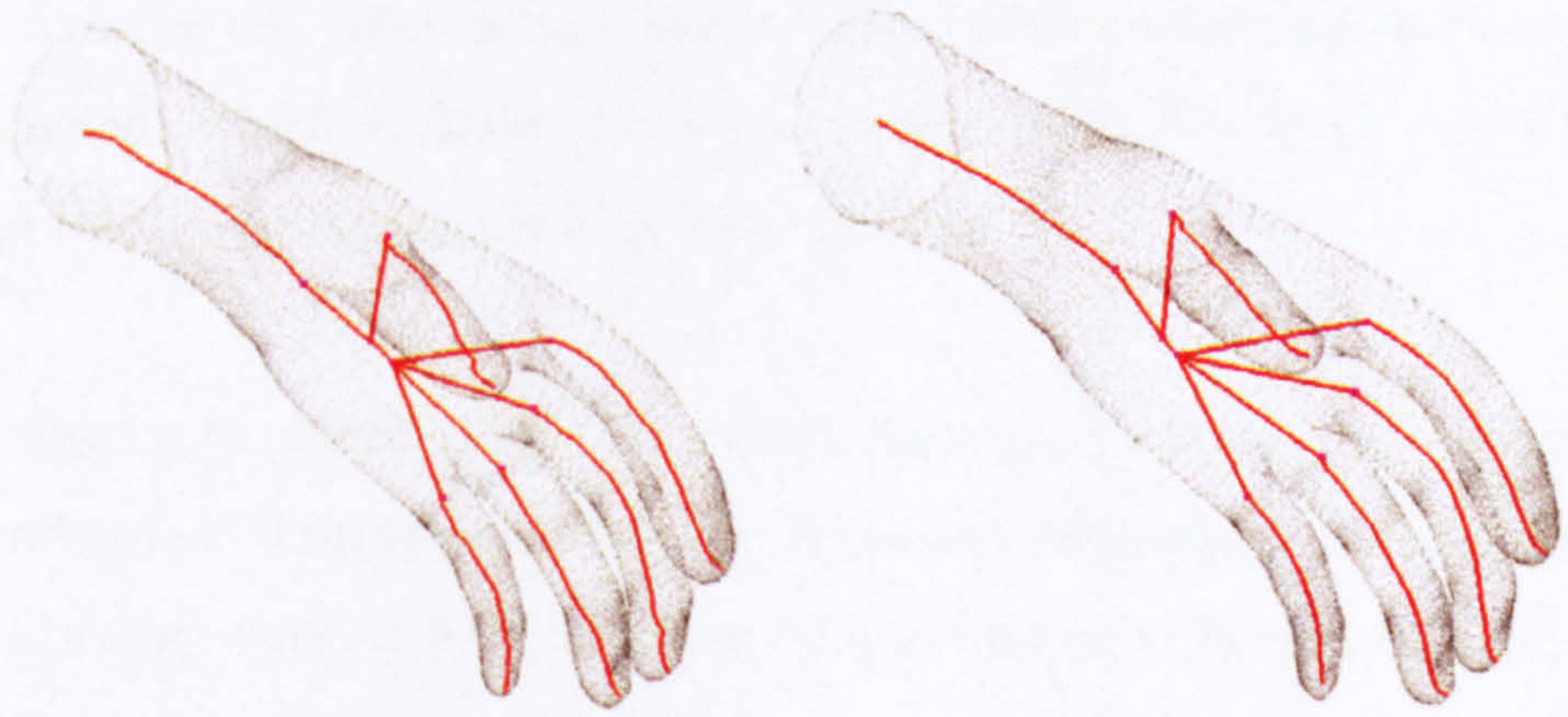


figure 3.13: Illustration of curve skeleton refinement. The coarse curve skeleton is shown with the blue curve, and the refined skeleton is shown with the red curve.

For a given point m on the coarse curve skeleton, the four nearest neighbour vertices c_1, c_2, c_3, c_4 on the second 3D silhouette of this segment can be found. As shown previously in figure 3.8, in this case, m is on the common edge of two adjoin triangles. These four vertices of the two triangles are the four nearest neighbour vertices. Using interpolation methods proposed by Dey and Sun [2006], the z coordinates of m can be adjusted by these four vertices. m is then offset to m' to form a new point of the curve skeleton.

In figure 3.14, the variance of the curve skeleton of the hand model during refinement is demonstrated. The improvement of curve skeleton centrality is clearly shown by comparing (b) with (a).



(a)

(b)

figure 3.14: Comparison between before and after the refining curve skeleton of a hand model. (a) before refinement. (b) after refinement.

3.4.2 Animation skeleton generation

The curve skeleton is different from a skeleton used in character animation. To make it usable, it must be converted to an animitable skeleton.

An animation skeleton consists of a series of connected skeletal joints. Our primary goal here is to locate these skeletal joints properly on the curve skeleton derived in above sections. The final animation skeleton is a set of joint nodes linked with straight line segments.

Observed from figure 3.8(c), a 3D medial axis, i.e. the coarse curve skeleton, is divided into branches when it meets the junction triangles. It ends at the midpoint of an internal edge of the junction triangle. After refinement, these points may change position, yet the new points they form are still regarded as an end point of the curve skeleton branch. Each curve skeleton branch has two ends. These two end points represent the key skeletal joints for the animation skeleton. As shown in figure 3.8(d), they are marked in blue. Thus it delivers the coarse animation skeleton structure.

Taking the hand model as an example, it is divided into seven branches: wrist, palm and five fingers. Accordingly seven medial axes containing fourteen points are constructed, which includes both end points of each branch curve. Those fourteen points are treated as the key skeletal joints.

The next stage is to identify those in-between joints, i.e. the joints located between the two ends of a 3D medial axis branch. A number of methods are practicable for this task. In our work, a method which makes use of a technique called down sampling was adopted.

The constrained Delaunay triangulation produces a set of discrete nodes which are the midpoints of internal edges. These points are considered as potential skeletal joints. Our further deduction is made by assuming that the skeletal link of a rigid body object exhibits rotational displacements to a certain extent around a joint. For each node, we calculate the angle between the line segments that joins it with its two adjacent nodes. A threshold is given to evaluate the bending. If the angle is below the threshold, this node will be determined a skeletal joint. This is shown in figure 3.15.

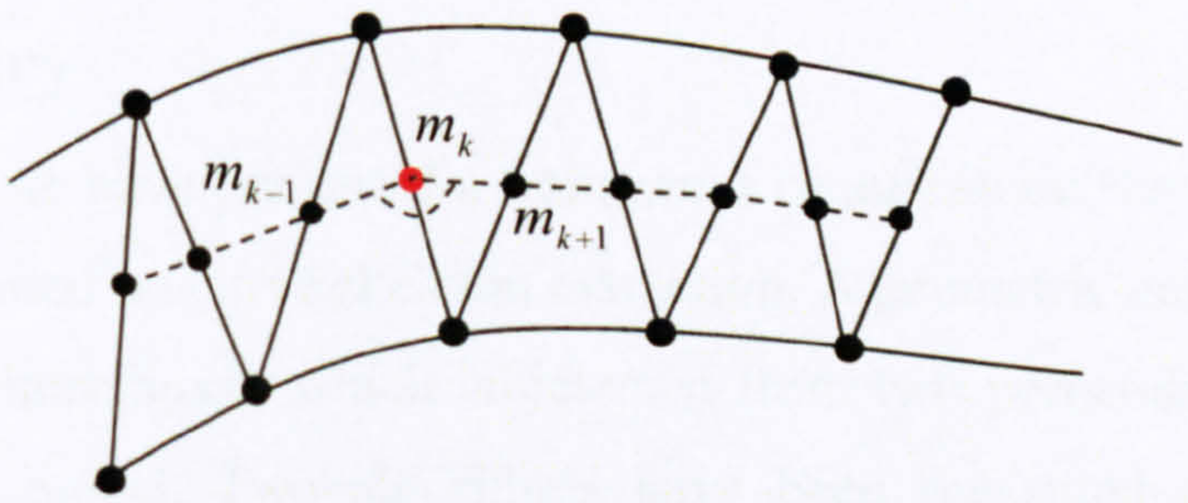


figure 3.15: Illustration of determining a joint. The bending angle $\angle m_{k+1}m_k m_{k-1}$ is below a given threshold. m_k is determined as a skeletal joint.

The advantage of this method is that it is not restricted to suit a certain type of models. It is capable of detect in-between joints of a model which whether it is a human-like character or an odd-shaped monster. The disadvantage is that, if the model is given at a pose where the extracted animation skeleton is absolutely fully extended at some joint, i.e. not bent at all, it is not possible to identify all the joints. Some in-between joints may be missing. For example, the elbow is unlikely to be identified if the arm is extended without even tiny bend around it. However, the joints set can always be adjusted manually if necessary. With major joints having been detected, it is rather straightforward to insert or remove a few in-between joints.

After all the skeletal joints are located, one of them is selected to serve as the root for the skeleton hierarchy. The selecting method is presented in the study of Wade and Parent [2002].

An example of an animation skeleton generated for a hand model is given in figure 3.8(f). The green nodes represent the major joints which are produced directly by end points of curve skeleton branch and the blue nodes represent the in-between joints which are determined by bending angles.

3.5 Summary

In this chapter, we have presented a framework of automatic skeletonization. This framework is based on curve skeleton extraction. A geometric entity called the 3D silhouette was introduced, which is detected from two perpendicular projections of a character model. Two algorithms have been presented to detect the 3D silhouette.

The first algorithm is based on a two-level-search procedure using a user-defined parameter as the searching radius: the global search selects a set of discrete vertices which describe the overall configuration of the silhouette, followed by the local search which connects and refines the result.

The other algorithm is based on the connectivity of vertices in the original mesh. It works by identifying the next silhouette vertex from those vertices connected to the current one, which at the same time connects the silhouette. This algorithm frees the silhouette detection from the assumption required by the two-level-search algorithm that the character model has to be given at an extended pose but at a price of efficiency. Its purpose is to handle the extreme cases since most models will satisfy the assumption in practice.

The curve skeleton can be extracted automatically from the 3D silhouette using constrained Delaunay triangulation, which describes the rough shape of the character's animation skeleton. With the necessary joints identified, it is able to establish an animation skeleton with them.

This automatic skeletonization framework is suitable for dealing with characters of different appearances and topological structures. It only demands that the cross-sections of a model are roughly elliptic.

This automatic skeletonization framework also provides interactive options for animation. Automation is important for relieving human labour. However, it only aims to ease the tediousness in animation, and should not be perceived as replacing completely the craftsmanship of animators. An animator is therefore given the overall control of the final decision. They are allowed to supervise the result, adding or removing joints and adjusting the positions of the joints produced by this automatic algorithm according to their needs.

CHAPTER 4

AUTOMATIC SKIN ATTACHMENT

Due to various advantages of skeleton-driven animation, it is widely spread in modern animation industry for handling articulated character models. This approach, as reviewed in Chapter 2, drives the skin deformation with an embedded skeleton which consists of joints and bones (joint links). Most deformation algorithms attach the skin to the skeleton with a set of weights to evaluate the influence a skin vertex gets from a joint during transformation.

Determining weight appropriately is not easy. The weight distribution algorithm currently in practical use simply computes the distance between a vertex and each joint and chooses a number of geometrically closest joints as the influence joint set. It introduces too many irrelevant influences that adjusting weight values afterwards is recognized as the most tedious task in realistic animation.

In this chapter, we aim to present a new automatic skin attachment framework which distributes weight values of better usability without intense manual intervention. This proposed framework is intended to be compatible with most of the popular skinning approaches.

4.1 Overview

Contrary to the conventional weight computation techniques which calculate weights directly from the distance between a vertex and a joint without considering the model structure, our framework of weight distribution benefits from its ability to determine a set of joints which have substantial influence on the skin shape by taking into account the topological neighbourhood of a skin vertex.

We start with segmenting a model mesh into regions. Each region corresponds to a joint. In accordance with the structure of the skeleton, two regions are described 'adjacent' if their enclosed joints bear a 'parent-child' relationship or share the same parent. For each region of a joint, a shadow area and a dominant area are further described, which reflects the rough level of influence of the enclosed joint.

We then categorize each skin vertex into either a dominant area or a shadow area of a joint region with regard to its distance to the region boundaries. Should a skin vertex fall into a dominant area, it is only attached to the enclosed joint of the region and only affected by this enclosed joint during deformation. Otherwise, it falls into a shadow area, and is affected by the enclosed joint of one or several adjacent regions as well. The weight that associates the vertex with each joint is solved according to the distance from the vertex to the boundary of the region of that joint.

Therefore, the weight computation framework presented in this thesis can be divided into three steps:

1. Boundary generation;
2. Dominant area and shadow area specification;
3. Weight computation.

We will explain each step in detail in the following sections.

4.2 Preliminaries

Our skin attachment framework starts with mesh decomposition. It is assumed that the model prepared for skin attachment is already embedded with an animation skeleton. The skeleton can be either set up by hand, or extracted using the skeletonization algorithm proposed in the previous chapter, or using any other extraction algorithms.

4.2.1 Related Mesh Decomposition Methods

Mesh decomposition is about dividing a mesh into a number of disjoint subsets under preset conditions. A great deal of decomposing criteria and algorithms can be found in literature [Shamir 2008].

In dealing with partitioning an object into meaningful segments, many approaches focused on making use of surface properties, such as convexity, or concavity, or dihedral angles [Kraevoy et al. 2007][Zuckerberger et al. 2002]. Some approaches proposed to decompose the mesh by defining feature points [Katz et al. 2005].

Skeleton and medial axis encode topological information of the object, hence they can be linked to mesh segmentation directly [Lien et al. 2006] [Wu et al. 2006] [Tierny et al. 2007]. In the study of Shapira et al. [2008], the mesh was decomposed via the measurement of local-thickness using their proposed shape-diameter-function. A decomposition method for articulated bodies was proposed by de Goes et al. [2008] by defining a new medial structure of the body. By recursively computing the bijection between a region and its medial structure, the mesh segments are established and refined to the desired standard.

In our framework, the mesh decomposition is based on the structure of the character skeleton. The skeleton-based decomposition methods mentioned above

are able to finely define body segments. However, they are computationally too expensive for our goal of weight computation. We will present another decomposition algorithm which is more explicit in computation, and well suits our needs.

4.2.2 Fundamentals of Decomposition

The representation of a character in animation is its surface configuration, i.e. the skin. In skeleton-driven animation, the deformation of the skin is derived from its embedded skeleton. Transformations of the skeletal joints bring to the character changes on its surface shape.

The mesh decomposition used in this framework aims to define more accurately how the skin vertices are affected by the joints. Therefore, a skeletal joint is a key reference for decomposition. Segmentation can be regarded as a mesh being cut at each joint.

However, the setup of animation skeletal joints does not strictly copy from the anatomical skeleton. Very often, a larger number of anatomical joints can be simplified to a few animation joints. For instance, anatomical joints along the spine are usually represented by three nodes in animation skeleton: neck, chest and pelvis. The joints taken in animation are considered of great importance in motion generation, e.g. elbow or ankle.

It is observed that, in most cases, a joint of this type usually defines a major division. At some point on the surface of the joint area, the curvature exhibits a concave change, as shown in figure 4.1(a). In other cases, this concavity may not be as obvious as others, as shown in figure 4.1(b). Nevertheless, the area around the joint over a large range is unlikely to be 'thicker'.

(a) (b)

figure 4.1: An illustration of the surface change around a joint area. (a) Around the wrist area, the surface appears concave at some point. (b)For a cat tail, no obvious changes are observed.

This observation is made on creatures which exist in reality. However, for a virtual character to appear realistic, it needs to preserve similar anatomical configurations of real ones.

Therefore, the principle of our mesh decomposition algorithm is to divide the model mesh at the potentially concave point where the distance between the surface and the joint is approximately the shortest among its neighbouring area. The boundary of the mesh segments may be detected by curvature examination as proposed in some papers. This is achieved by finding four points on the surface which is approximately the closest to the joint, and a boundary is generated by joining the dividing points together.

4.3 Mesh Decomposition

In order to reduce distortions caused by excess influences from topologically irrelevant joints, it is considered in our framework that the joint affecting a skin vertex should be close to the vertex not only geometrically, but also topologically. We identify the influence joints set for a vertex by first dividing the character mesh into regions. Each region corresponds to a joint. These regions are then extended to create overlapping areas for the purpose of smoother weight blending. The extended region describes an effective range of a joint.

The mesh decomposition is achieved by specifying a boundary between two regions whose enclosed joints are bound by a 'parent-child' relationship. As illustrated in figure 4.2, let J_p be the parent joint of a joint J, let $J_{c_i}, 0 \le i \le m$ be the child of J. We denote the mesh region corresponding to J_p with r_p, the region corresponding to J with r, and the region corresponding to each J_{c_i} with r_{c_i}. Region r is then separated from r_p by boundary B, and from r_{c_i} by boundary B_{c_i}.

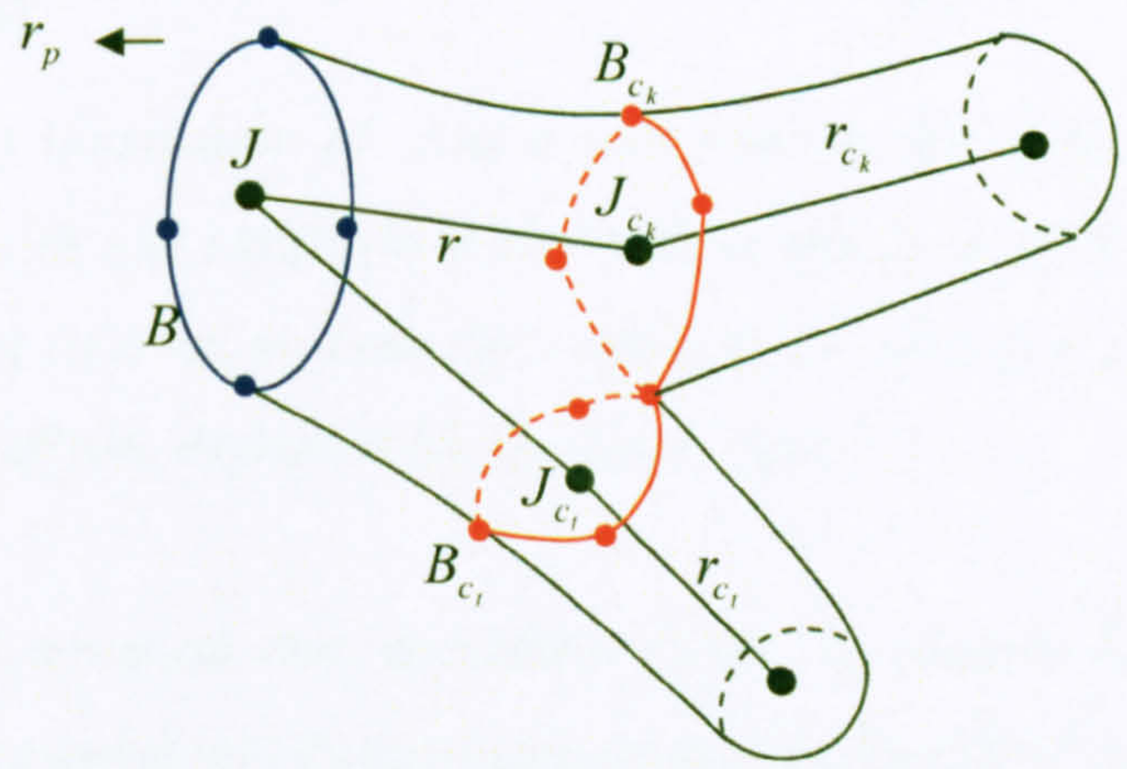


figure 4.2: Illustration of regions specified by boundaries corresponding to joints. Boundary B with its dividing points which separates r from r_p is represented by the blue. Boundary B_c with its dividing points which separates r from r_c is represent by the red. $k,t \in (1,m)$.

A boundary is a closed curve which roughly encircles a joint. In our algorithm, it is determined by four dividing points on the mesh surface, also illustrated in figure 4.2.

4.3.1 Dividing Points Detection

Intuitively speaking, the boundary stops the influence of a joint from entering the domain of its child joint. Thus we would exclude the consideration of root joint,

e.g. the pelvis in a human skeleton, before searching for dividing points of a joint. As shown in figure 4.3(a) and (b), a root joint J is the top node of a skeleton hierarchy. Without influence taken from another higher node, there is nothing to separate from at this joint.

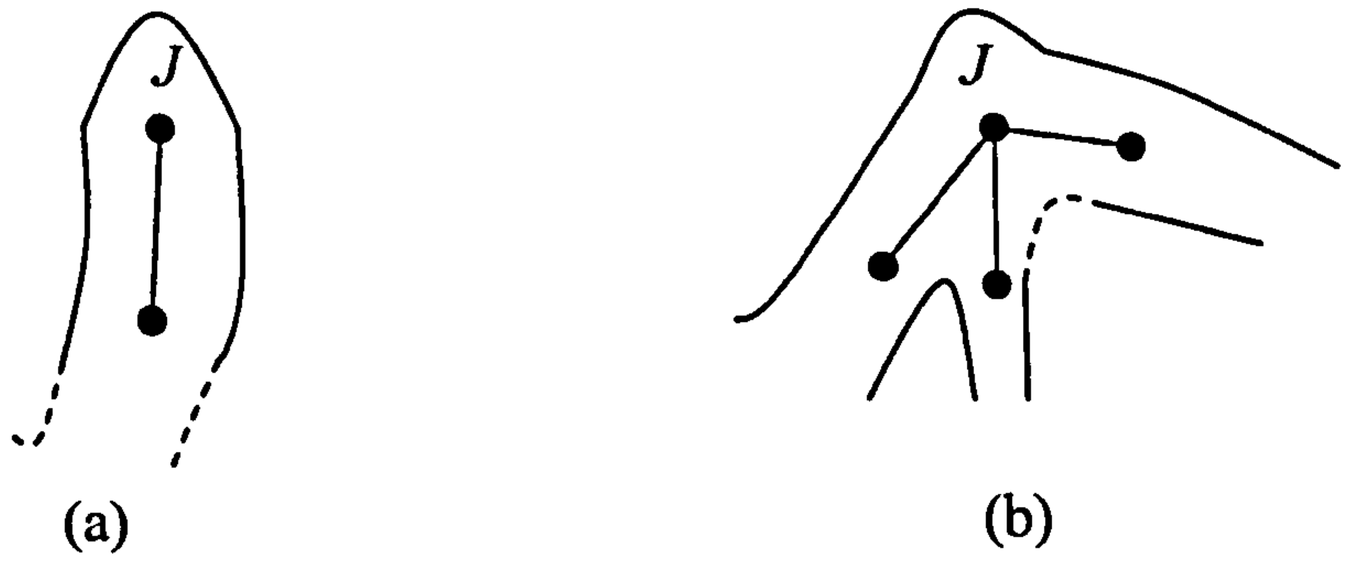


figure 4.3: An illustration of J as a root joint of the skeleton hierarchy: (a) with one child; (b) with multiple children. Without a parent joint of its own, no boundary needs to be specified at J to separating influences from a higher level node.

Based on the observation and deduction stated in section 4.2.2, the distance between a dividing point and its corresponding joint should be a local minimum.

4.3.1.1 Type of Joint Linkage

A boundary B corresponding to a joint J describes the separation of region r of its own from the region r_p of its parent. The approach to find dividing points that determine the boundary at J relies on the number of links formed by J with its child joints.

Also based on the number of child joint links of a joint, we classify joints as two types: single-linkage joint and multi-linkage joint. This categorization is similar to that proposed by Yang et al. [2006].

Single linkage implies a joint J has only one child, as shown in figure 4.4(a). In the case of a joint having no child, i.e. J is the leaf node of a skeleton hierarchy,

it can be regarded as a single-linkage joint as well, with a virtual child placed on the prolongation line of joint link $\overrightarrow{J_pJ}$.

A multi-linkage indicates a joint J has more than one child. Thus its links formed with its children is a one-to-multiple linkage, as shown in figure 4.4(b).

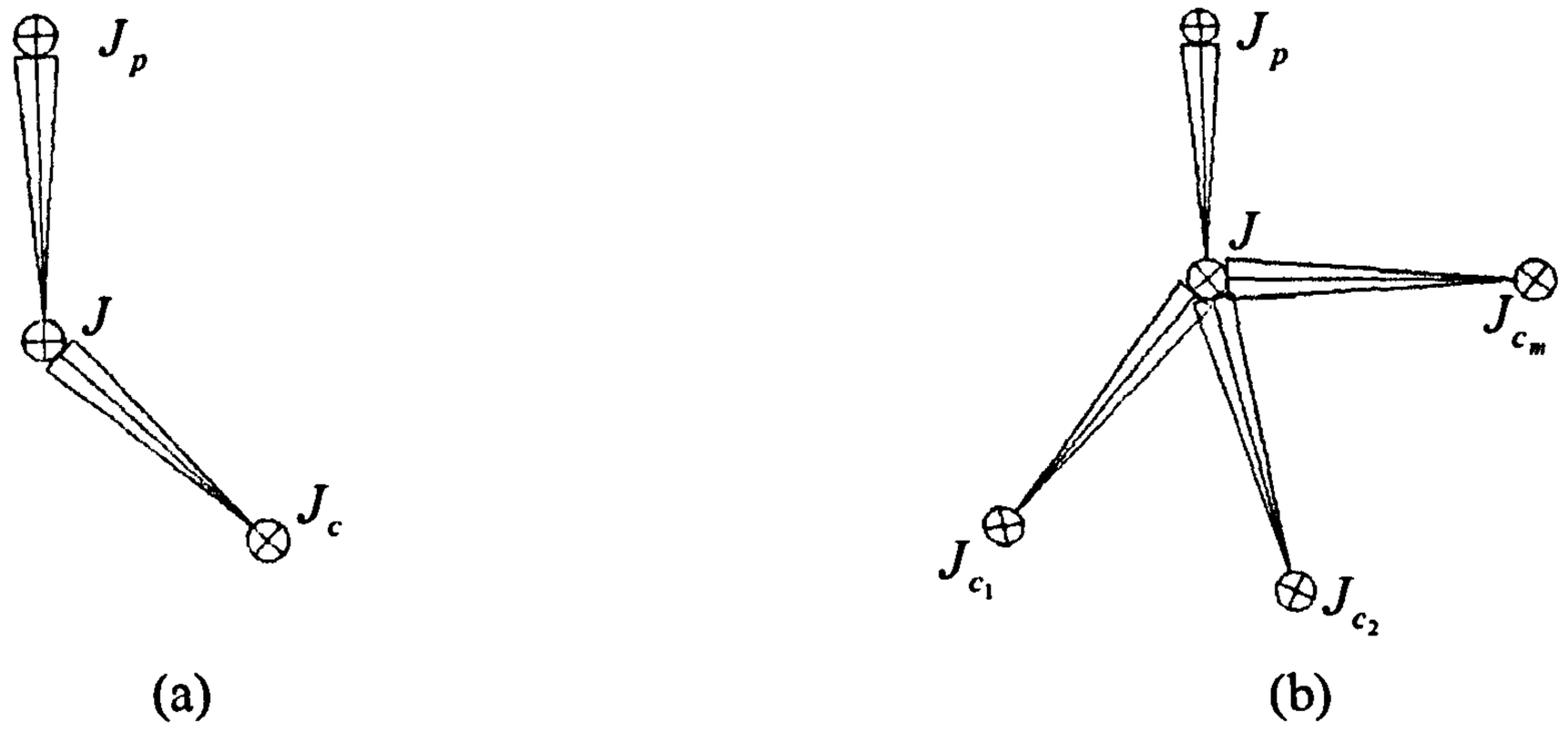


figure 4.4: Illustration of two types of linkage. (a) single-linkage; (b) multi-linkage.

To find dividing points for a joint of either type of linkage is similar in principles, but with a few modifications in practice.

4.3.1.2 Detection Algorithm for Single-Linkage Joint

To search for a dividing point, a searching plane needs to be fixed. For a joint of single-linkage, the dividing points are regarded as two pairs located on two perpendicular planes which can be referred to as XY plane and XZ plane. Here only how to decide the XY plane is described. With the XY plane and the origin being fixed, the XZ plane is also fixed.

In a single-linkage, the link $\overrightarrow{J_pJ}$ which is formed by joint J and its parent is taken as the orientation of positive X axis, with J as its origin. The method of determining the XY plane can be further divided depending on its child link $\overrightarrow{JJ_c}$.

As a plane is determined by two intersecting lines, the XY searching plane can be determined generally by $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_c}$, shown in figure 4.5(a). For a joint whose links $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_c}$ are in the same line, any plane that passes the X axis, i.e. link $\overrightarrow{J_pJ}$, can be chosen as the XY plane, which is shown in figure 4.5(b).

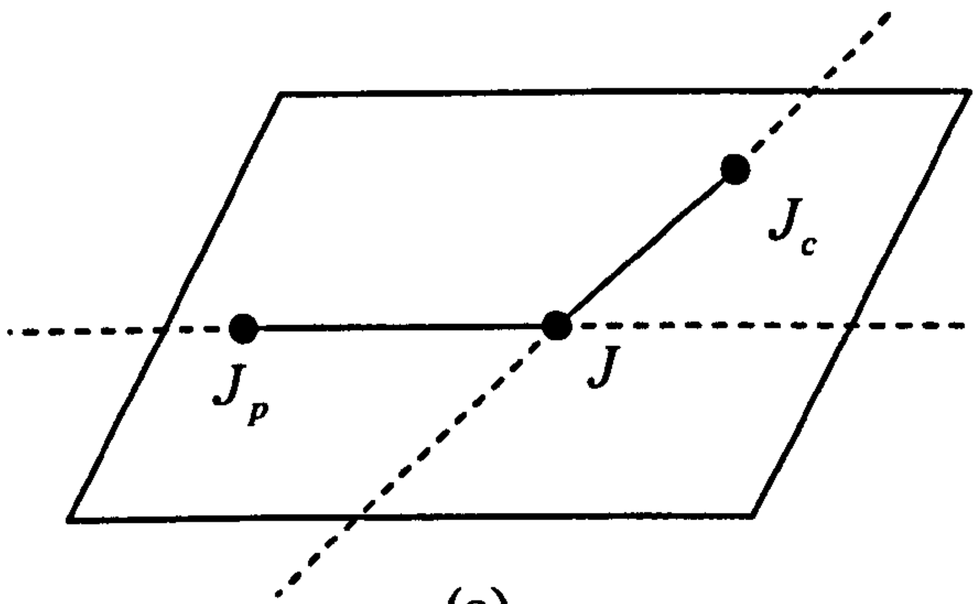


figure 4.5: Determining the XY searching plane for a single-linkage joint. (a)The plane is determined by intersecting joint links $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_c}$; (b) In the case of $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_c}$ in the same line, the XY plane is chosen as any plane passing $\overline{J_pJJ_c}$.

The approach of finding the diving points, i.e. finding the surface points which are approximately the closest, for a joint J of single-linkage type is rather straightforward. A sample of distance value can be obtained by emitting a ray from joint J on the XY plane. When the ray intersects the surface at a point p, the length of line segment $\overline{Jp}$ is recorded as the distance from J to this

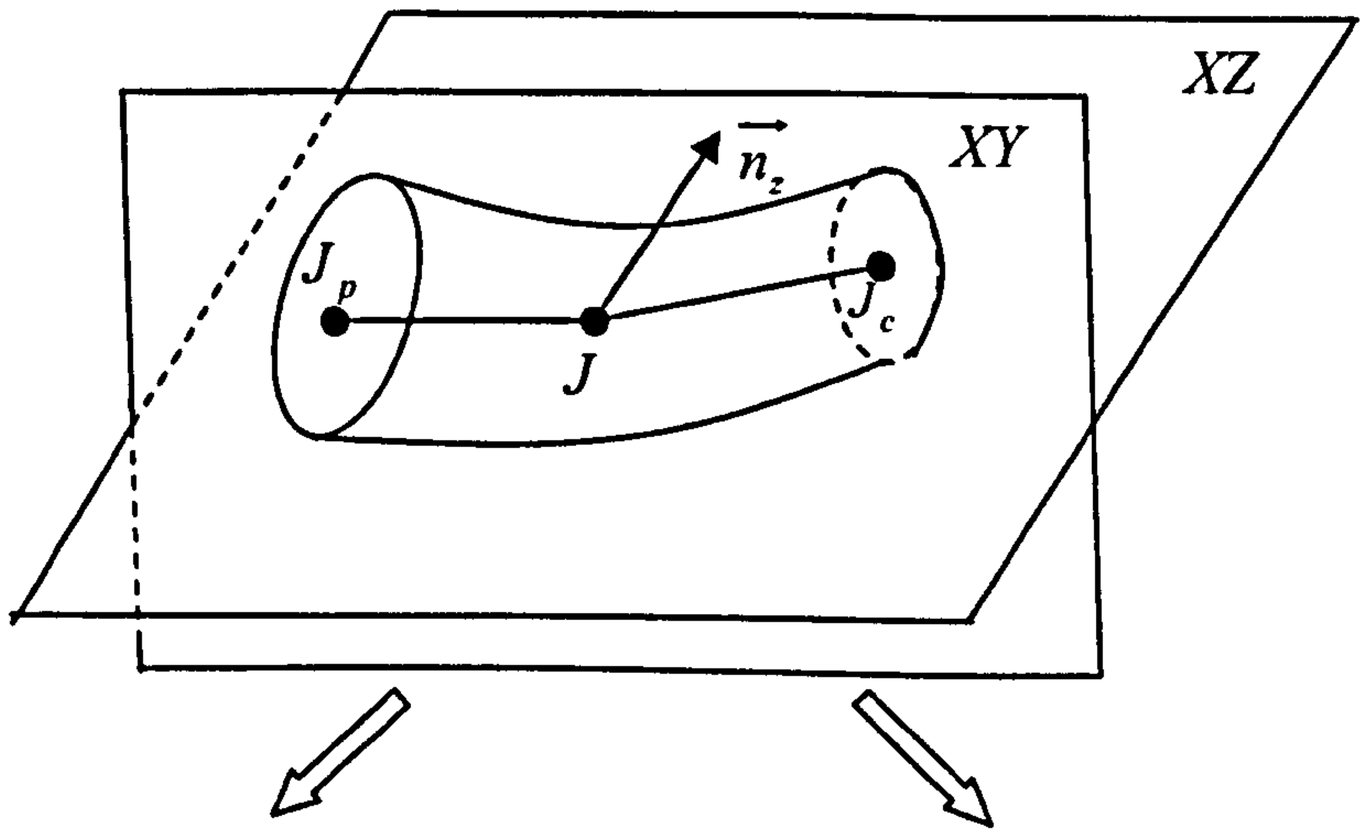
particular surface point.

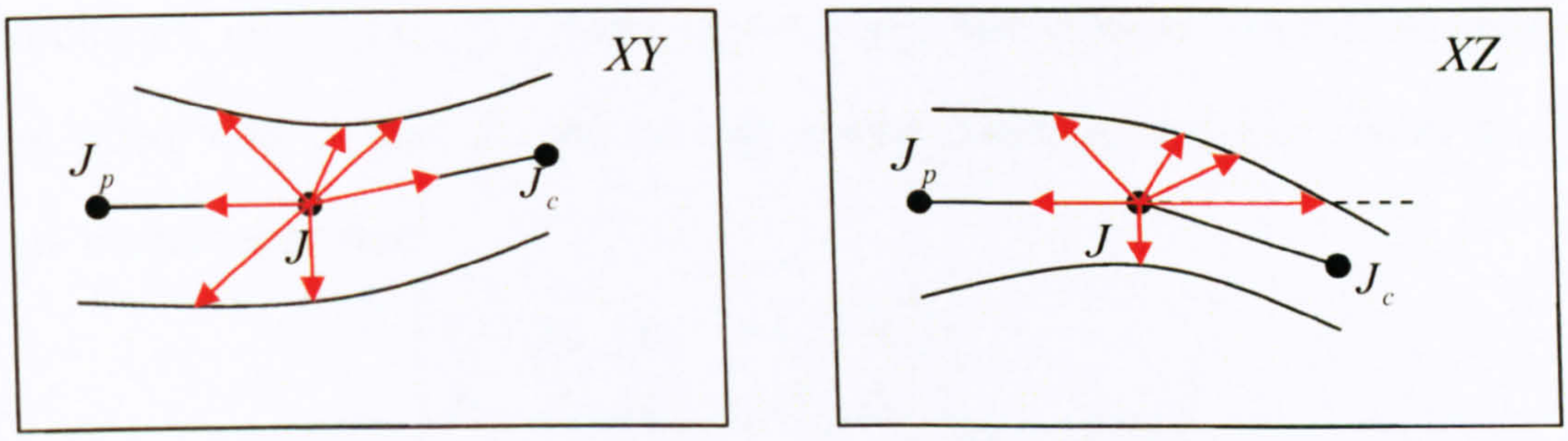
As shown in figure 4.6(a), the XY plane in which joint link $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_c}$ lie is divided into two halves by these two links. Thus we sample the distance all along the surface between J_p and J_c within both sides of link (J_p, J, J_c) by respectively emitting n rays. On either side, the point bearing the shortest distance is selected as one of the dividing points.

The number n affects the density of distance sampling. It can be user-defined. In the interest of both accuracy and efficiency, n is normally chosen between 100 and 150. It gives good performance in practice.

As shown in figure 4.6(b), to determine the other pair of dividing points on XZ plane, a similar approach applies with a slight modification.

Suppose n rays are emitted from joint J on positive Z half plane and negative Z half plane respectively, between positive X axis and negative X axis. After compare the distance obtained via measuring the line segment formed by J and the intersection point, the dividing points on XZ plane is also chosen as the points which bears the shortest length.





(a) (b)

figure 4.6: Illustration of detecting dividing points for a joint of single-linkage by sampling the distance along the surface. (a) on XY plane; (b) on XZ plane. The sampling rays are represented by red, and the blue vector points to the positive Z.

Therefore, the detecting algorithm is formulated as:

$$(p_t, t) = \min_{i=(1,n)}^{t} | D(J, p_i) | \qquad (4.1)$$

where $D(J, p_i)$ stands for the distance between the joint and the i^{th} intersection point, t for the sequence number of the intersection point that bears the shortest length of the intersection line segment.

4.3.1.3 Detection Algorithm for Multi-Linkage Joint

To search for dividing points for a joint of multi-linkage type is a little more complicated. Any three of these joint links are not always on the same plane. Thus we are not able to determine a common plane as a searching plane using all the joint links.

However, in animation practice, the binding pose of a model is usually neutral. At this pose, all joint links are approximately fully extended. That is to say, when projected to a plane, all the joint links do not overlap each other.

Suppose there is one parent joint and m child joints connected to J, their

coordinates, i.e. $J_p(x_p, y_p, z_p)$ and $J_{c_i}(x_i, y_i, z_i)$ are already known. A plane is then fitted with J_p and J_{c_i} by solving a over-determined linear equation with least-squares approach:

$$\begin{bmatrix} x_p & y_p & z_p & -1 \\ x_1 & y_1 & z_1 & -1 \\ \dots & \dots & \dots & \dots \\ x_m & y_m & z_m & -1 \end{bmatrix} \bullet \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0 \qquad (4.2)$$

This plane is then translated to go through J. Thus the XY plane is determined. J is chosen as the origin and the projection $\overrightarrow{J_p'J}$ of $\overrightarrow{J_pJ}$ on this plane is chosen as the orientation of positive X axis, as shown in figure 4.7.

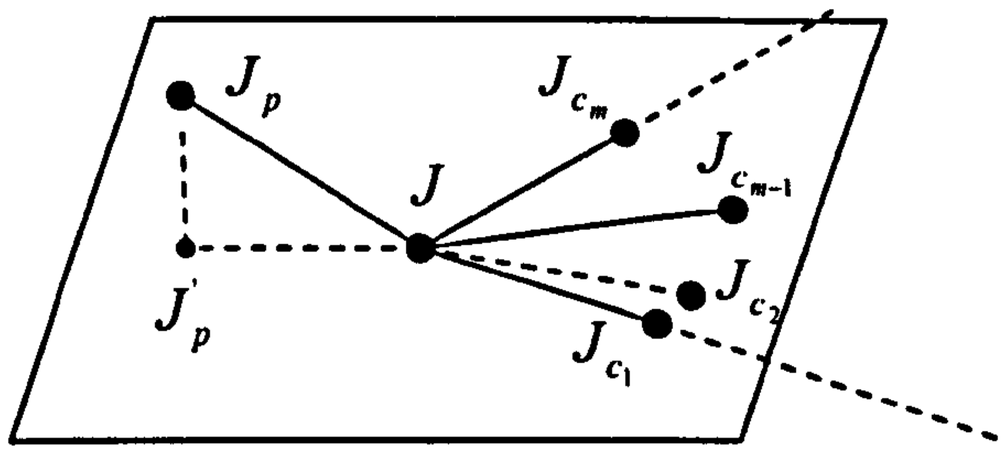


figure 4.7: Fitting the XY plane for a multi-linkage joint using the parent and child joints of the joint.

To find the four dividing points for such a multi-linkage joint, we still consider that two of them lie on the XZ plane which is perpendicular to the XY plane given above. The approach of searching for these two points remains the same as that of searching for the two points on the XZ plane for a single-linkage joint. However, as the children of J may not lie on the XY plane, accordingly, we will not define the other two dividing points on the XY plane.

To find the other two dividing points, we first calculate angle between $\overrightarrow{J_p'J}$ and the projection of each joint link $\overrightarrow{JJ_{c_i}}, i \in (1, m)$ on the XY plane anticlockwise. The child joints are labelled from J_{c_1} to J_{c_m} following the order of angles from

small to large. The two links formed by the child joints with smallest and largest angle respectively, i.e. link $\overrightarrow{JJ_{c_1}}$ and $\overrightarrow{JJ_{c_m}}$, are defined as the 'border' link. The other two dividing points are located respectively on the plane determined by $\overrightarrow{J_pJ}$ with $\overrightarrow{JJ_{c_1}}$ and with $\overrightarrow{JJ_{c_m}}$. As shown in figure 4.8, rays are emitted from joint J, on the plane determined by $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_{c_1}}$, and the plane determined by $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_{c_m}}$ respectively, to sample the distance between the surface and the joint J.

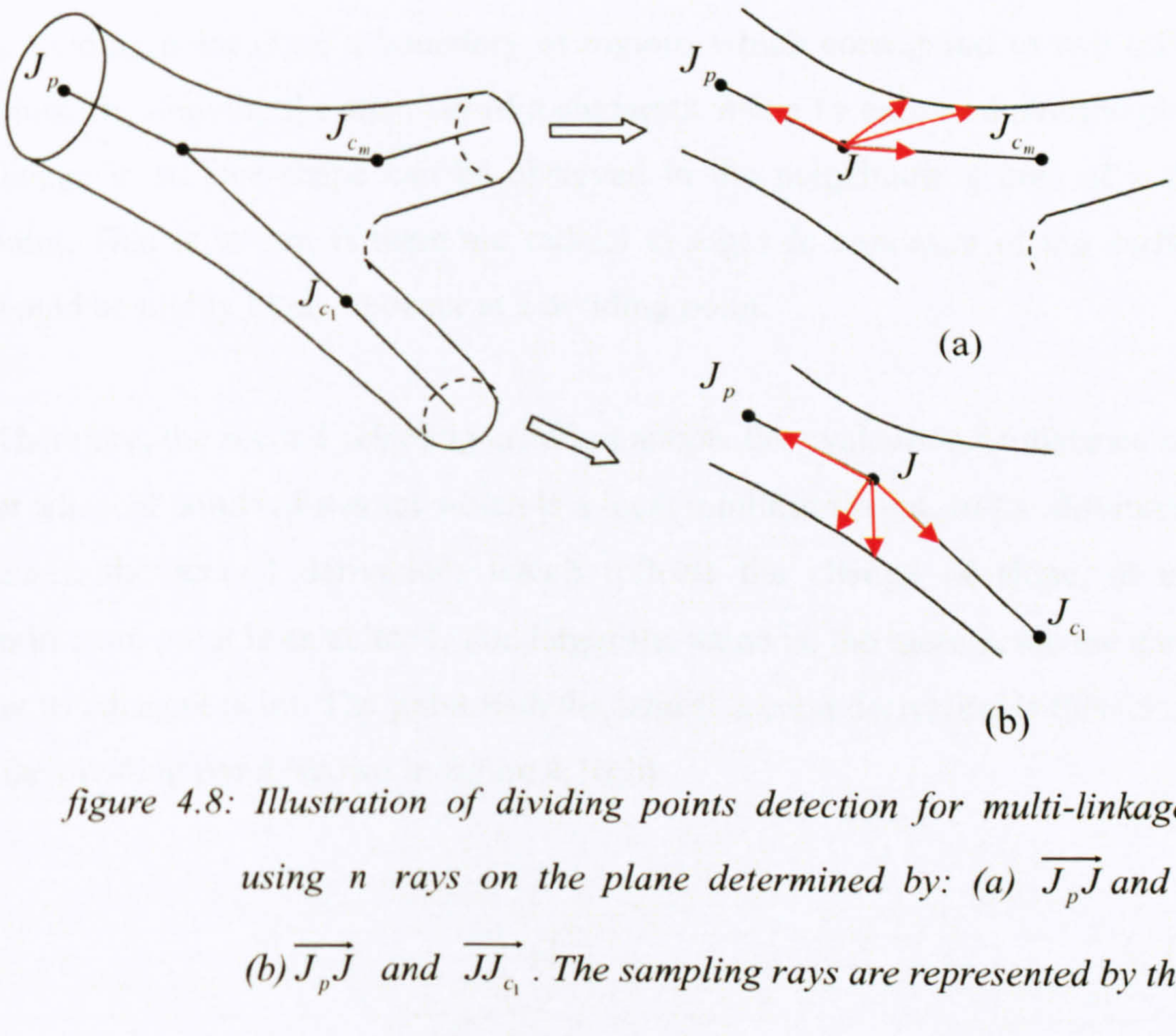


figure 4.8: Illustration of dividing points detection for multi-linkage joint using n rays on the plane determined by: (a) $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_{c_m}}$; (b) $\overrightarrow{J_pJ}$ and $\overrightarrow{JJ_{c_1}}$. The sampling rays are represented by the red.

The dividing points are chosen as the intersection point bearing the shortest distance to J as well, as stated in eqation (4.1).

4.3.2 Dividing Points Modification

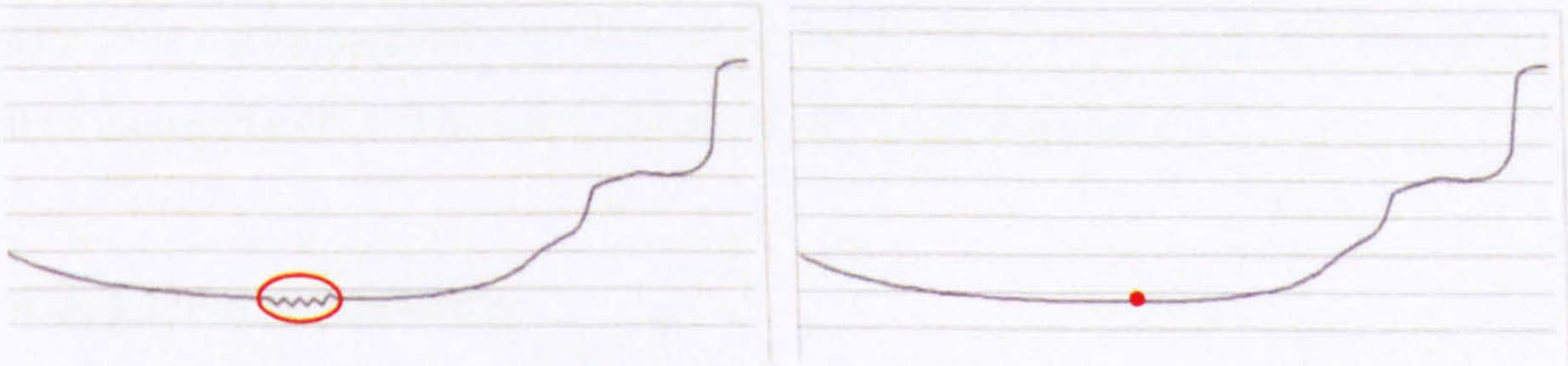
In the process of determining the dividing points via local minimum, some

sampling rays may return fluctuated values within a small range, as shown in figure 4.9(a). Or they may return several local minima which are very close to each other, as shown in figure 4.10(a).

To deal with such noisy circumstances, we need to smooth the sample values before making the final decision. The fluctuated values can be smoothed out by this process, as shown in figure 4.9(b). However, as regards the situation that two or more local minima appear, another selecting criterion is provided.

A dividing point is on a boundary of regions which correspond to two different joints. By studying the anatomy of a character, it can be expected that the obvious change in surface shape can be observed in the neighbouring area of a diving point. That is to say, if there are radical changes in curvature of the surface, it would be highly likely to occur at a dividing point.

Therefore, the second selecting criterion adopts the evaluation of distance change at adjacent points of a point which is a local minimum point. In the distance value chart, the second derivation, which reflects the change of slope, at a local minimum point is calculated. The larger the value is, the more acute the change is at its adjacent point. The point with the largest second derivation is then chosen as the dividing point, shown in figure 4.10(b).



(a) (b)

figure 4.9: Illustration of smoothing the sampling distances. (a) Fluctuated values; (b) After smoothing.

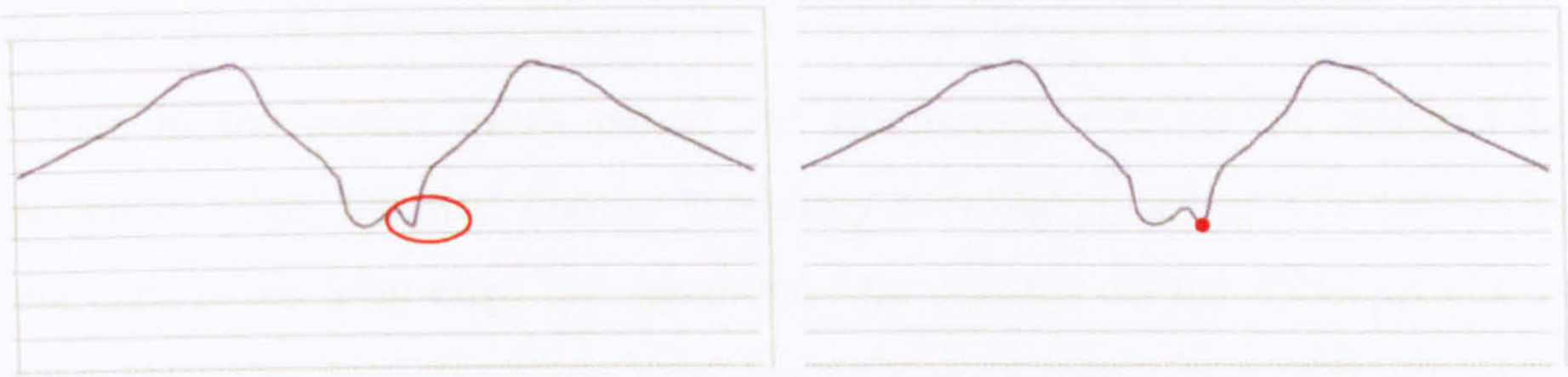


figure 4.10: Illustration of selecting a dividing point from a group of points with a similar distance value to the joint. (a) two local minima with similar value appear; (b) the one with more radical changes at its neighbouring points are determined as the dividing point.

4.3.3 Boundary Generation

After the dividing points for each joint are determined, the next step is to define the boundary B at joint J.

The four dividing points are labelled from b_1 to b_4. The order can be either clockwise or anticlockwise. As shown below in figure 4.11, we use the anticlockwise direction. A plane can be determined by every two adjacent points along with J. Its intersection with the model surface between the two adjacent dividing points forms a curve segment. For an instance, the plane determined by b_4, b_1 and J produces an intersection curve segment (b_4, b_1) which forms a segment of B.

The four curve segments produced by every adjacent pair of dividing points join into a closed curve. Thus the boundary B is generated.

4.3.4 Decomposition

Once the boundaries are created, regions belonging to different joints are separated. Skin vertices are then categorized into each region. The algorithm we use for the mesh decomposition is a region grow algorithm which is described in [Shamir 2008].

Edges in the model mesh which are intersected with B are referred to as a boundary edge. We first choose randomly a boundary edge. The two vertices at the ends of such an edge are separated into regions on the different sides of the boundary. The decomposition then can be started with either of these categorized vertices. Its connecting vertices are visited in turn. All vertices will be categorized into the same region until a vertex on the other side of a boundary is reached. If a vertex is just on the boundary, we categorize it into the adjacent region.

After all vertices in the mesh have been visited, the decomposition is completed.

4.4 Weight Computation

By decomposing a mesh into regions corresponding to a joint, an effective range of a joint is roughly defined. However, the joint region and the effective range is not exactly the same.

A region can be divided into two areas. The area closer to a joint J is the dominant area r_d, where the vertices are regarded as being affected by J alone. The further area is the shadow area r_{s_i}, where the influence of a child joint J_{c_i} starts to blend in.

4.4.1 Division of Dominant Area and Shadow Area

Suppose a vertex v is located in region r which corresponds to joint J, to judge which area of r the vertex v falls into is based on the distance between v and boundary B_{c_i} which separates r from regions that correspond to a child joint J_i of J. Geodesic distance can be used. To achieve good performance whilst accelerating the computation speed, a simplification is adopted in practice.

In boundary generation, boundary B is composed of four curve segments. Here a

simplified boundary can be created by substituting line segments for curve segments, which is also shown in figure 4.11. The distance between v and each line segment is computed, and the shortest one is defined as the distance d_B from v to boundary B. That is to say, d_B is computed by:

$$d_B = \min |dis(v, l_k)| \qquad (4.2)$$

where l_k stands for the line segment which forms the simplified boundary.

Similarly, the distance d_{c_i} between v and boundary B_{c_i} is computed as:

$$d_{c_i} = \min |dis(v, l_{k_i})| \qquad (4.3)$$

We also calculate the distance between the two line segments from which d_B and d_{c_i} is measured. It is recorded as D_i.

The value of ηD_i is given as a threshold to estimate whether or not the vertex v belongs to a shadow area r_{s_i}. If $d_{c_i} < \eta D_i$, the vertex is regarded as belonging to r_{s_i}. Otherwise, it is categorized into dominant area r_d of joint J.

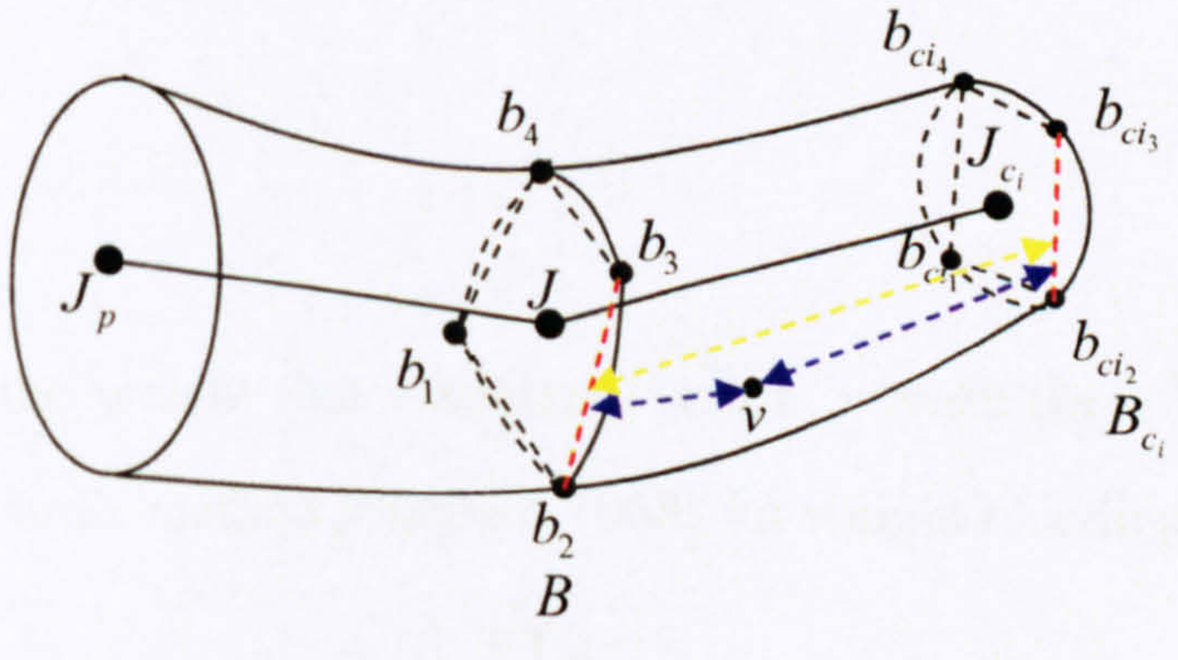


figure 4.11: Illustration of distance measuring. The line segment from which the distance between a vertex v and a boundary is taken is shown in red. The distance D_i between these two line segments is shown in yellow. The distance d_B and d_{c_i} are shown in blue.

The parameter η in the evaluation is user-defined. It affects the rigidness of the deformation effects. Larger η produces smoother deformation and vice versa. The deformation would be totally rigid if $\eta = 0$. Depending on the deformation effects a user wants to present, η can be assigned with different values according to their needs.

4.4.2 Weight Blending in Shadow Area

In this section, we will define the weight which associates a vertex with a joint with consideration to the area the vertex belongs to. For a vertex located in a dominant area, it is only attached to the enclosed joint. Thus the weight associated with joint J is assigned with 1. For a vertex in a shadow area, it is attached to several joints, and the weight associating with each joint is blended with respect to the distance to the boundary of the related region.

Suppose the vertex v is attached to n joints, the weight used in animation needs to satisfy:

1. $w_i(v) \geq 0$;

2. $\sum_{i=1}^{n} w_i(v) = 1$;

3. $w_i(v) \in C^0$.

Here $w_i(v)$ is the weight that associates vertex v with the i^{th} joint. Therefore, we choose Shepard's method [Shepard 1968] for weight blending:

$$w_i(v) = \frac{\prod_{j \neq i, j=1}^{n} d_i^{\alpha}}{\sum_{k=1}^{n} \prod_{j \neq i, j=1}^{n} d_i^{\alpha}} \qquad (4.4)$$

where d_i stands for the distance from v to the boundary that separates the region of i^{th} joint for its parent joint region. α is called the drop rate. The larger α is,

the smoother the effect is.

For a region that belongs to an end joint, i.e. a leaf node in skeleton hierarchy, there is no influence from other adjacent regions to blend in. Therefore, the whole region is regarded as a dominant area, with the weight of a vertex in this region assigned to the enclosed joint with 1.

4.5 Summary

In this chapter, we have presented a framework of automatic skin attachment. This framework is based on mesh decomposition. It determines a reasonable influence joints set for a vertex by taking into account the topological distance between a vertex and a joint.

By determining dividing points, boundaries are specified to separate a mesh into regions which correspond to the joints. Vertices in the mesh are categorized into these regions. Depending on the distance to the region boundary, a vertex may be further categorized into dominant area and shadow area of the region. In dominant area, the vertex is attached to only the enclosed joint of this region, while in shadow area, it is attached to the enclosed joints of some adjacent regions as well. In the latter case, we use Shepard's method to distribute the weights.

To distinguish the shadow area from dominant area, a user-defined parameter is provided. Depending on the smoothness of the deformation effect, this parameter can be adjusted according to their needs.

This framework aims to provide a more accurate weight computation to relieve the manual modification currently required by conventional weight assigning method. Furthermore, it aims to provide a flexible skinning approach which is compatible with various deformation methods, e.g. SSD or SBS mentioned in Chapter 2.

CHAPTER 5

RESULTS, COMPARISON, AND DISCUSSION

In Chapter 3 and Chapter 4, we have presented an automatic skeletonization and skin attachment framework, aiming to provide a more accurate and easy-to-use approach for both animation skeleton generation and skin weight distribution. In this chapter, we will demonstrate the application of the framework and compare its results to the results of previously developed frameworks.

5.1 Skeletonization

The first model we use to demonstrate the skeleton generation is a hand model, shown in figure 5.1(a). Its primary 3D silhouette is shown in figure 5.1(b). Via Delaunay triangulation, a 3D medial axis, i.e. a coarse curve skeleton, is extracted from the primary silhouette by connecting the midpoints of the internal edges of the triangles. Its centrality is refined by the second 3D silhouette which is generated from a decomposed mesh, shown in figure 5.1(c) and (d).

By down sampling the curve skeleton, 22 skeletal nodes are produced. The

animation skeleton is generated by connecting these skeletal joints with line segments, as shown in figure 5.1(e).

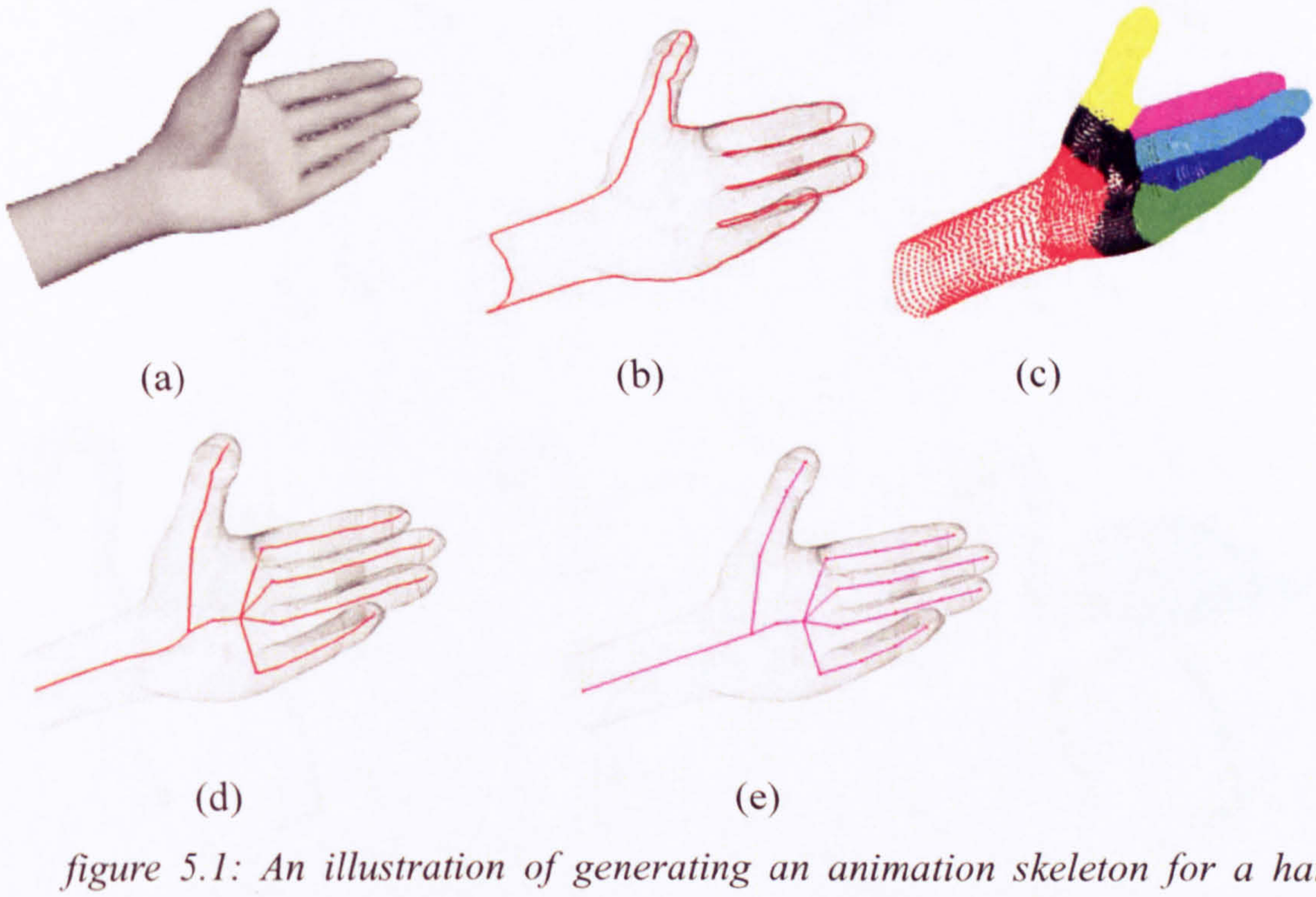
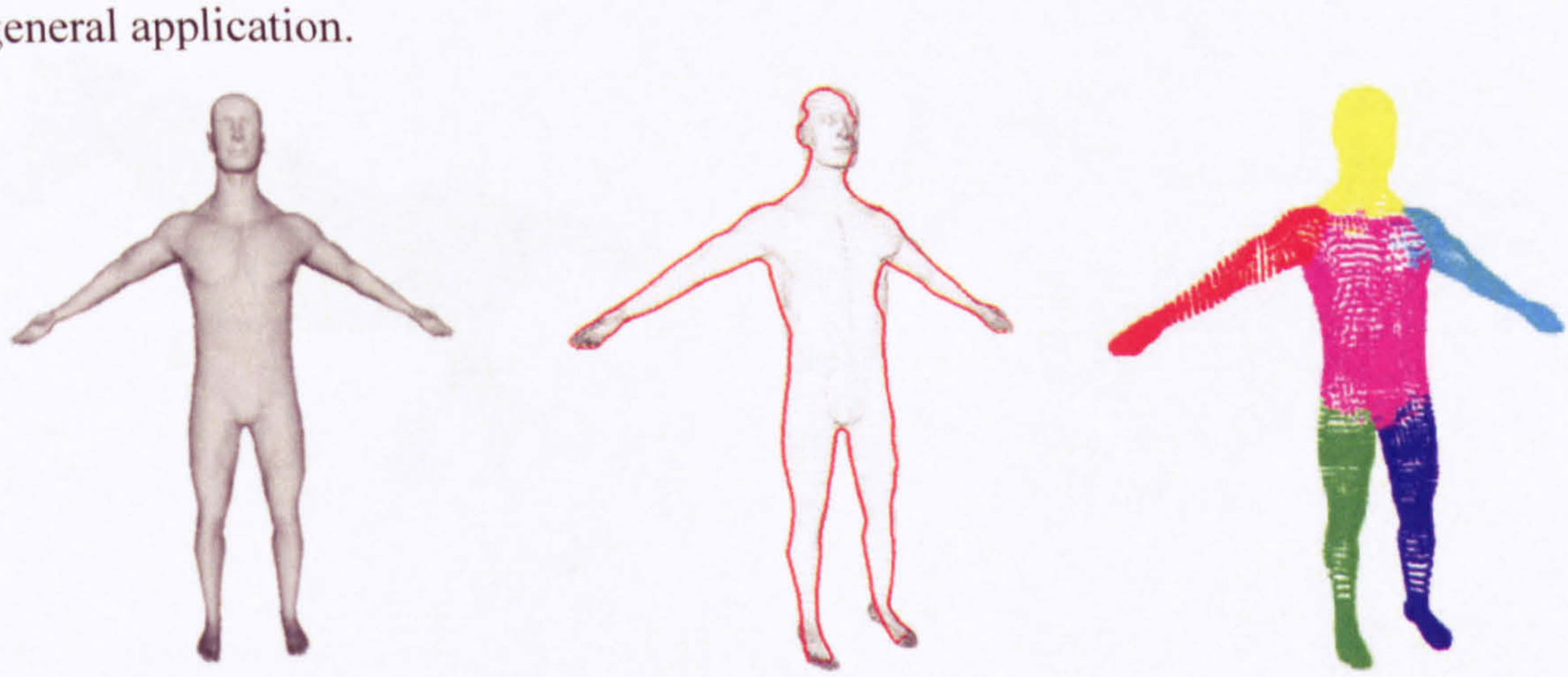
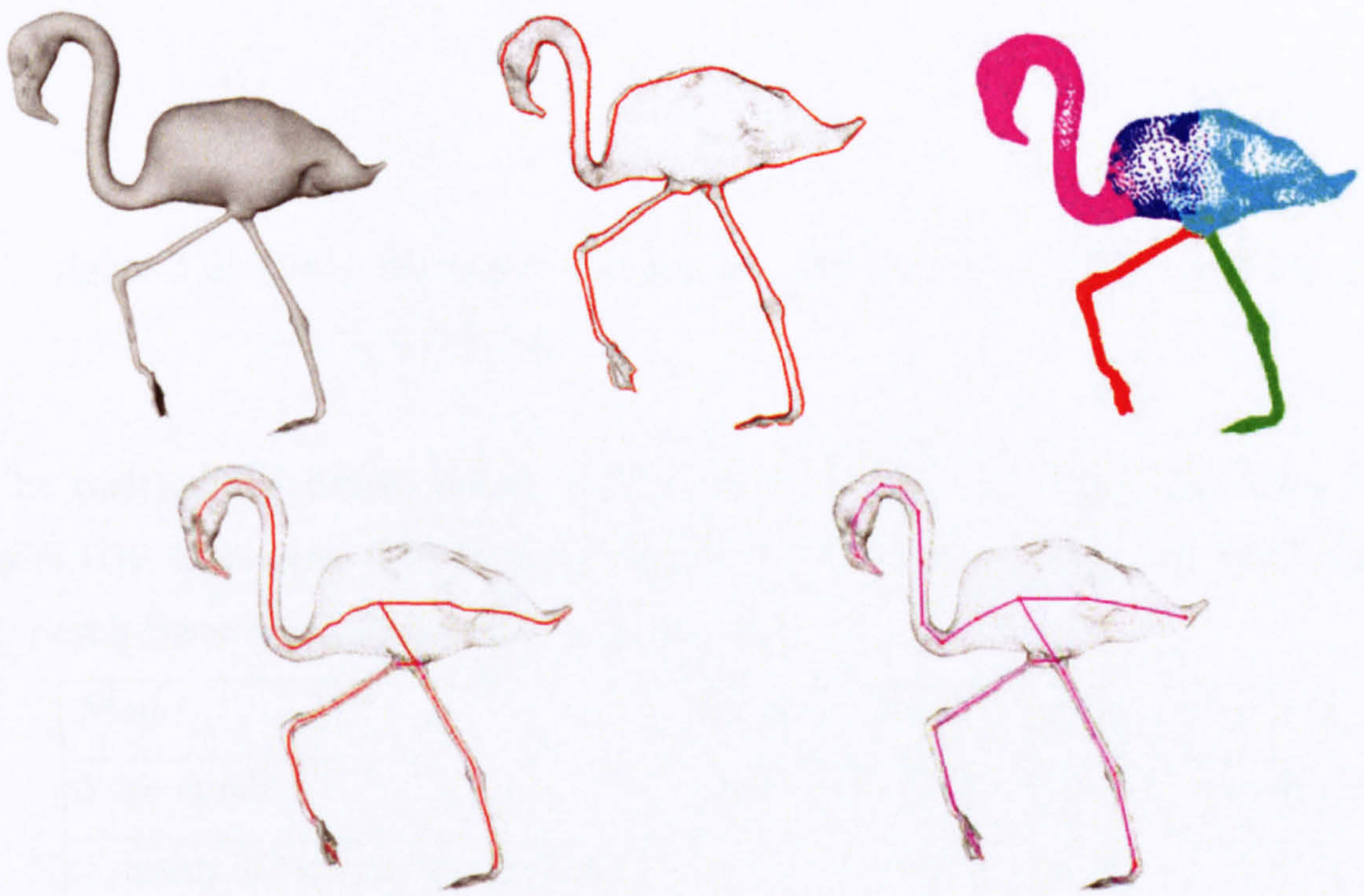


figure 5.1: An illustration of generating an animation skeleton for a hand model. (a) original model; (b) primary 3D silhouette; (c) decomposed mesh; (d) curve skeleton; (e) animation skeleton.

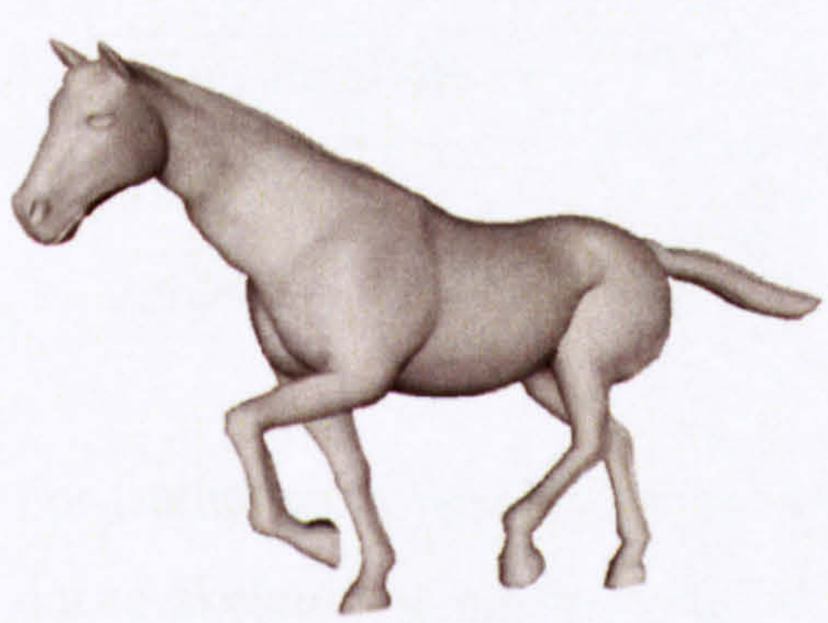
This presented framework is applicable to various character models despite shape and size, as long as the cross-section of the model maintains approximately elliptic. Three other examples are given below in figure 5.2 to demonstrate its general application.

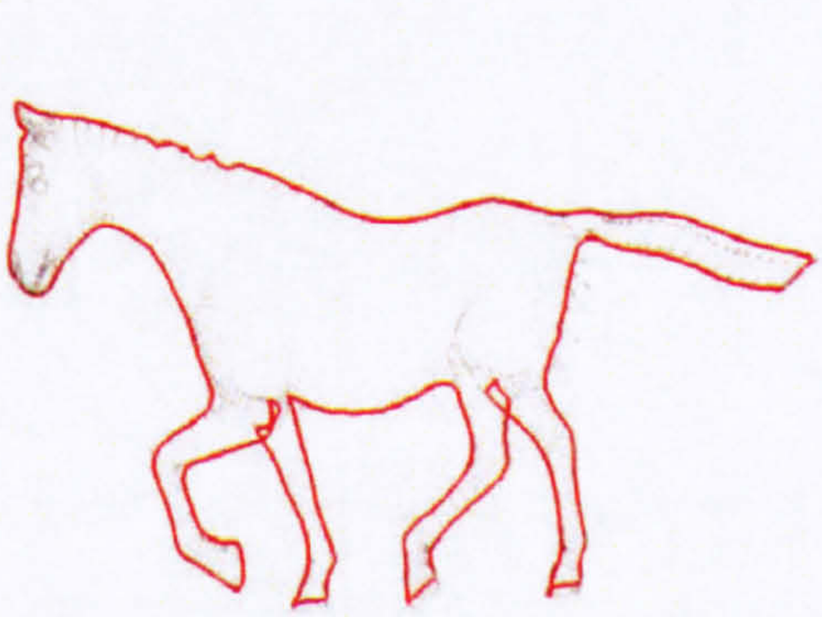


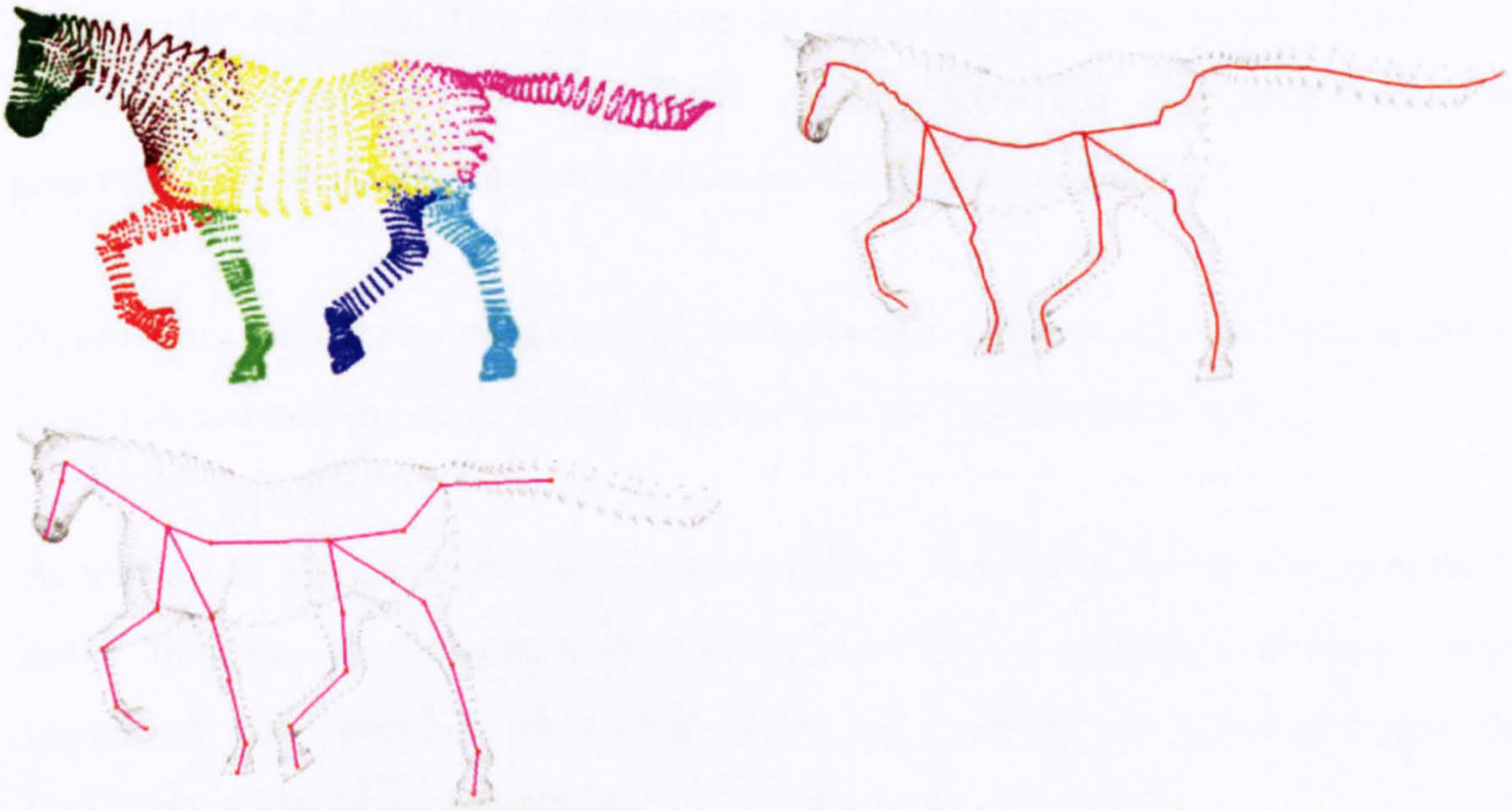
(a)



(b)







(c)

figure 5.2: Other examples of animation skeleton generation. (a) man; (b) ostrich; (c) horse.

The skeleton generation process is performed on a 1.6 GHz Intel Core Dual PC with 1GB of memory. The timing statistics of each stage along with the face count for each character is given below in table 5.1.

Model	hand	man	ostrich	horse
Face number	38016	29216	52895	16843
Primary 3D silhouette detection	4.2	3.7	9.2	2.2
Curve skeleton extraction	2.0	1.9	5.4	1.2
Joints location	0.4	0.3	0.5	0.2
Total time	6.6	5.9	15.1	3.6

Table 5.1: Execution time statistics for four different 3D models (sec.)

For frameworks which generate the animation skeleton from a curve skeleton, the curve skeleton is expected to depict the topology of the animation skeleton as closely as possible. To evaluate the accuracy of the framework proposed in this thesis, we compare the skeleton of a hand generated by our framework to the

skeleton derived from five other popular skeletonization methods. These five methods are medial surface, Reeb Graph, thinning, distance-field and potential-field methods. The comparison is illustrated in figure 5.3.

Figure 5.3(a) illustrates an animation skeleton of a hand model which is manually rigged by animators [Alias 2004]. It is used as the evaluation criterion.

According to this standard animation skeleton, the curve skeleton generated by either thinning or distance-field algorithm is not smooth and contains many unwanted fine branches. Post-processing, e.g. pruning, is demanded and this inevitably undermines the applicability of these two algorithms.

The skeleton generated by Reeb Graph method results in poor centrality which is fatal to the control of animation. The skeleton produced by potential-field and medial surface algorithms yield cleaner and smoother results. However, these two algorithms do not always preserve the connectivity of the model. Furthermore, intense post-processing is also demanded by these two algorithms.

In contrast, the curve skeleton extracted using our method results in a clean, continuous, well-centered and topological-preserving structure. It is the closest resemblance of the standard manually-rigged animation skeleton.

(a)

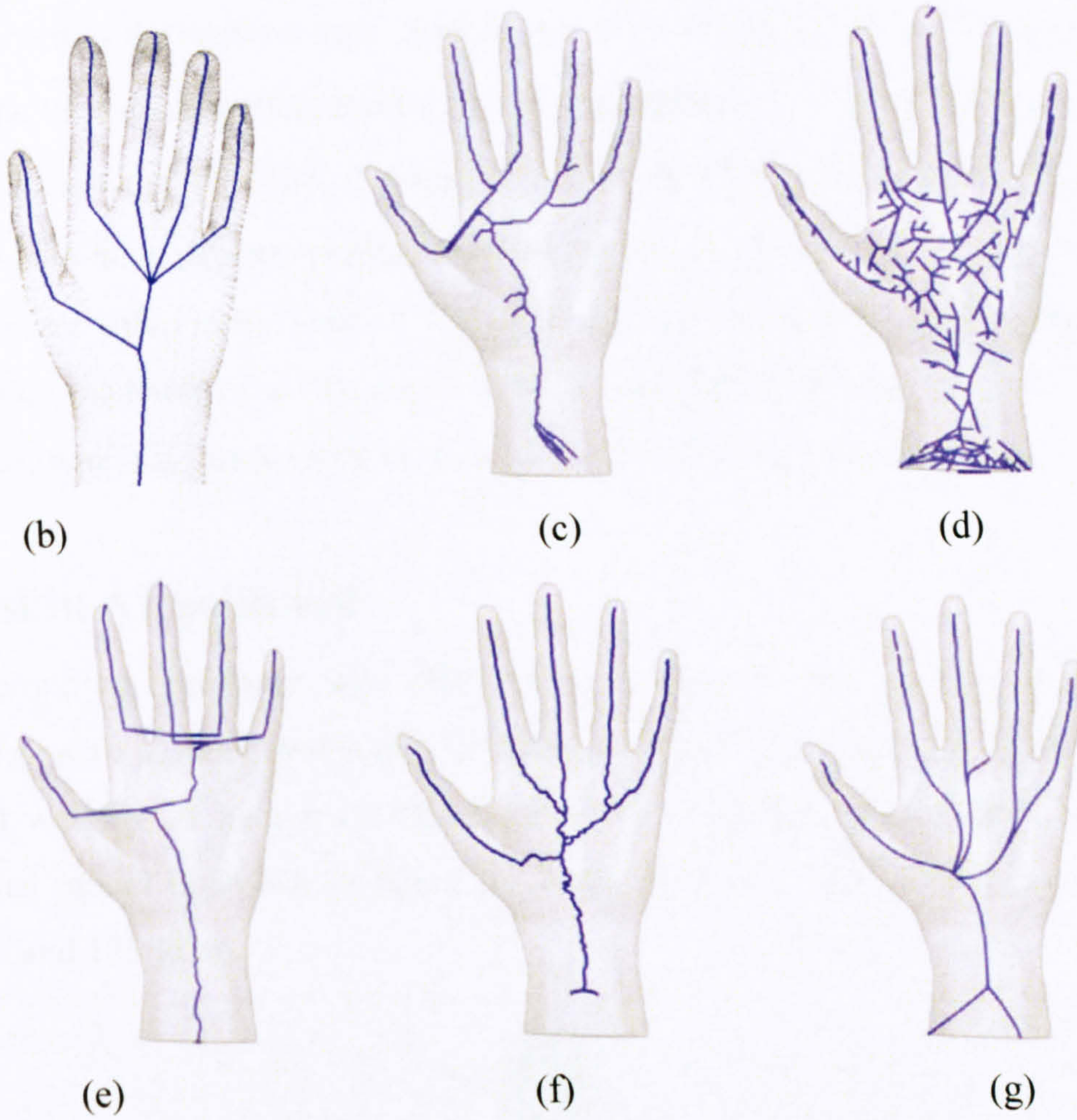


figure 5.3: Comparisons of the results produced by different classes of curve skeleton extraction algorithms. (a) a hand model rigged manually [Alias 2005]; (b) our method; (c) thinning; (d)distance field; (e) Reeb Graph; (f) medial surface; (g) potential field.

To study the efficiency of the presented framework, the computation complexity of this framework and other frameworks are analyzed. Suppose there are n vertices in the model mesh, m vertices in the primary 3D silhouette, and l branches in the curve skeleton extracted from the primary silhouette, the computation complexity of such a model using our framework for primary 3D silhouette detection, curve skeleton extraction, joints locating, skin attachment respectively are $O(m^2)$, $O(m^2/l)$, $O(m)$ and $O(n)$. In a general sense, m can be regarded as the maximal contour length (perimeter) of the model and n can

be regarded as the surface area. This makes $O(m^2)$ approximately equal to $O(n)$. Thus the total computation complexity of the automatic skeletonization algorithm in our framework is $O(n)$. Compared to the $O(n^2)$ computation complexity of typical automatic skeletonization methods [Wu *et al*, 2006], our framework works much more efficiently. Table 1 suggests that on average we can generate the animation skeleton of a 3D model with around 30000 triangles in less than 10 seconds, which is much faster than most skeletonization methods.

5.2 Skin Attachment

The proposed automatic skin attachment framework aims to associate the skin vertices with a more reasonable influence joints set to produce more accurate initial weights. A human model is used in this section for demonstration. The original model is shown in figure 5.4, composed of 14,652 vertices and 29,862 edges and 133 joints.

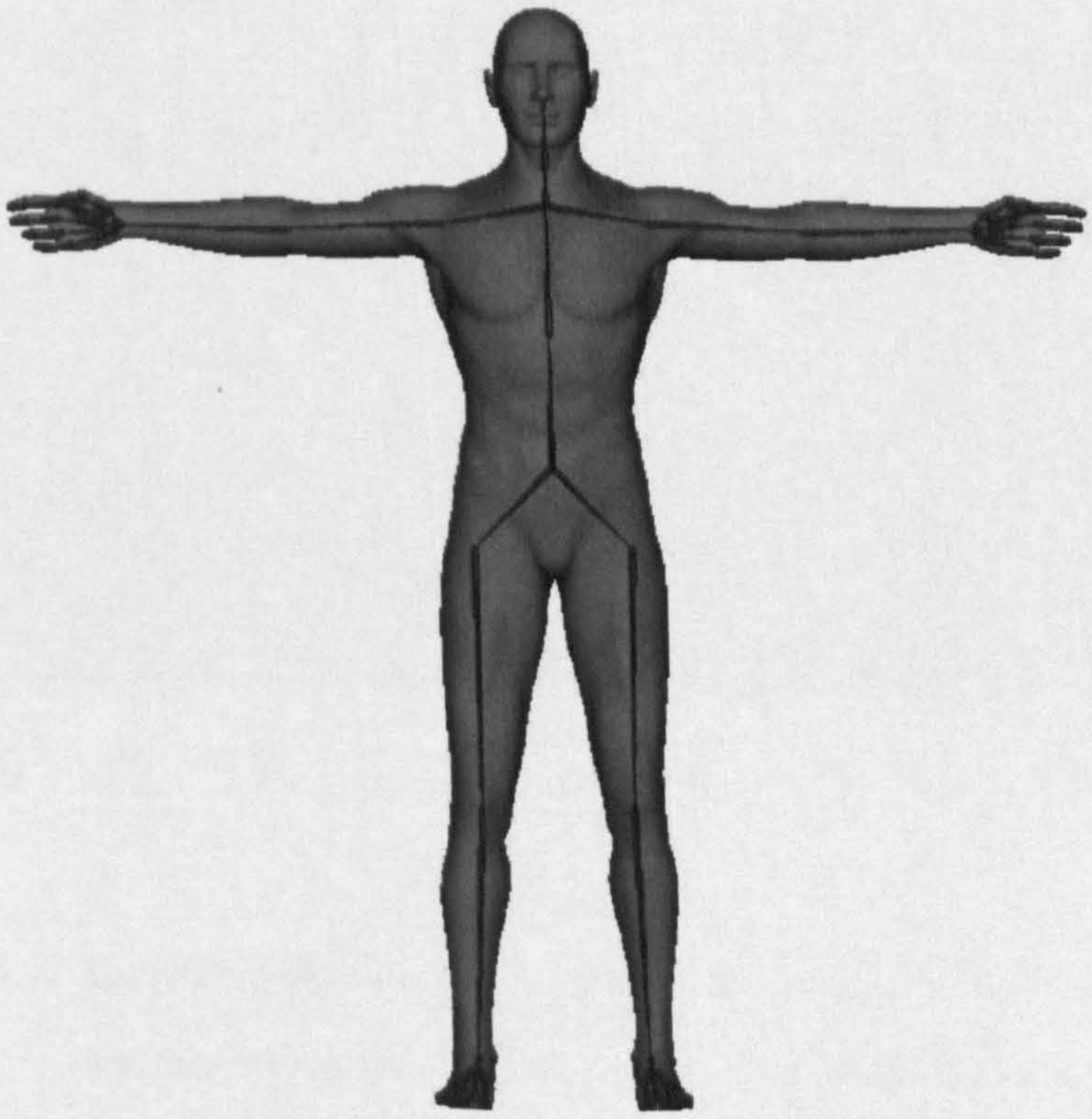
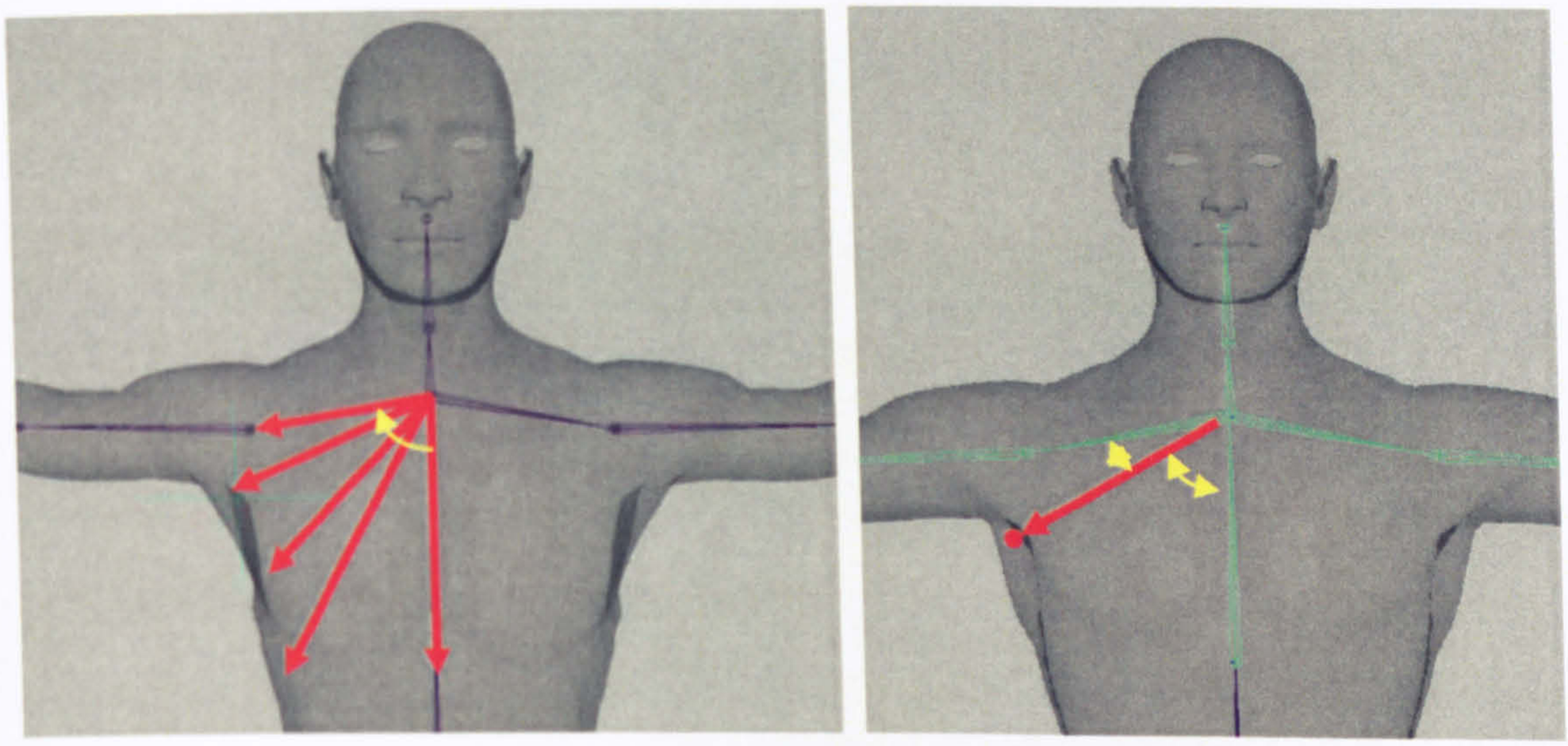


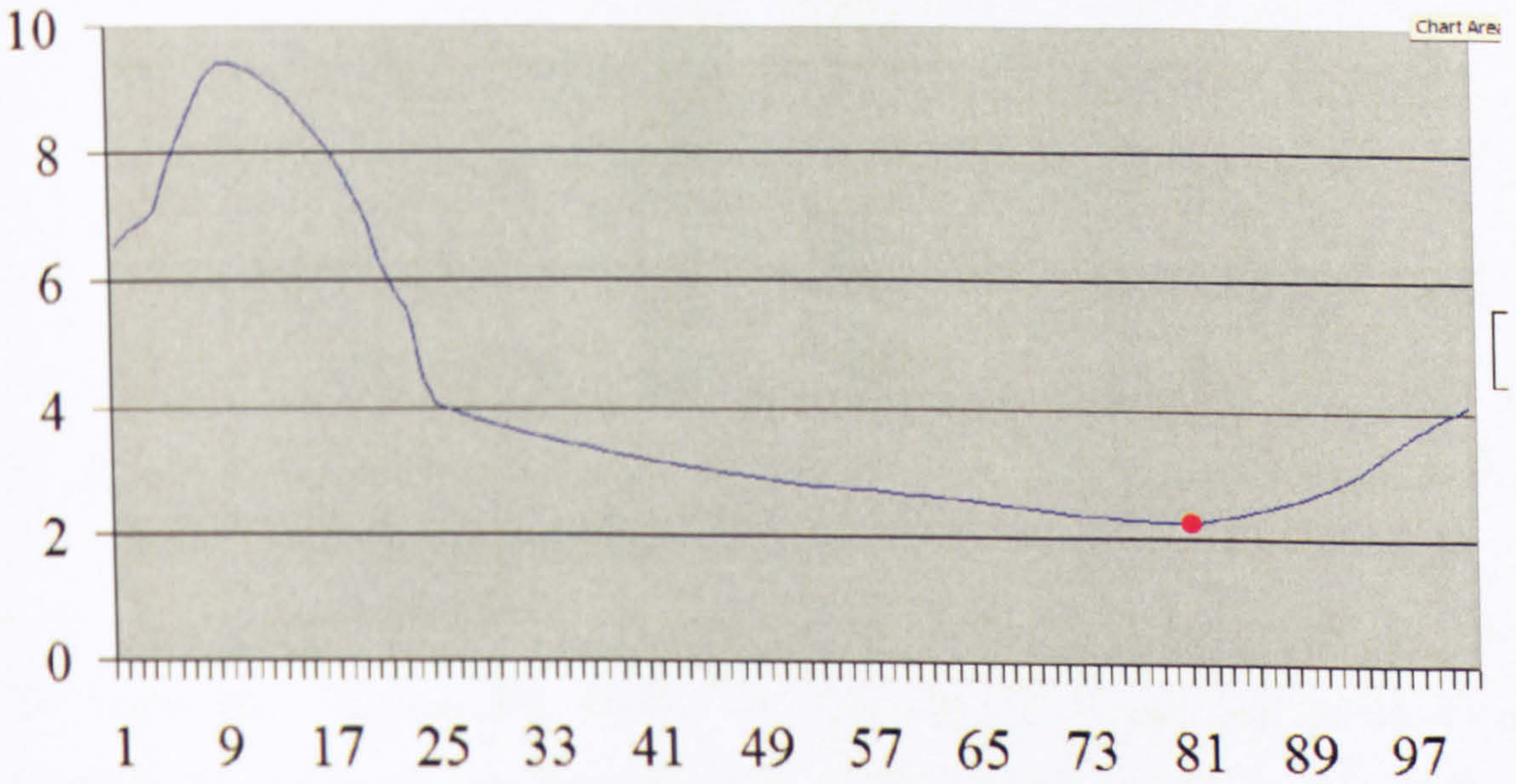
figure 5.4: Original mesh with embedded skeleton for skin attachment

n rays are emitted from a joint to sample the distance between this joint and its surface area. In our implementation, we choose $n = 100$. The dividing points are chosen as the closest points, according to formula (4.1). An illustration of this process is given below in figure 5.5.



(a)

(c)



(b)

figure 5.5: Searching for a dividing point at the lower neck joint. (a) Rays are emitted to sample the distance. (b) The distance values returned by the 100 sampling rays. (c) The closest point to the joint is chosen as a dividing point.

A boundary is generated at each joint by connecting dividing points and the model mesh is then decomposed into regions corresponding to joints, as shown in figure 5.6.

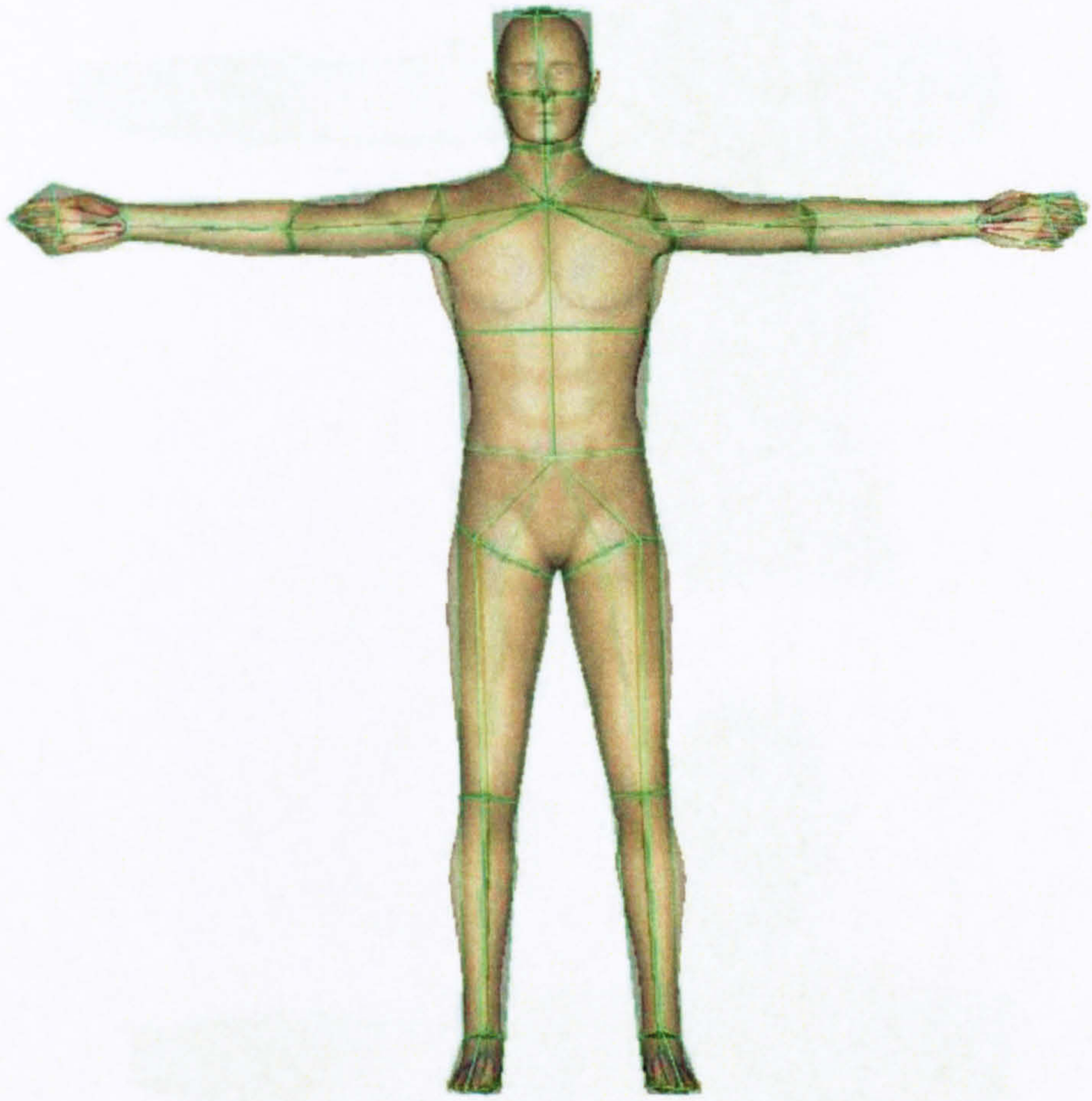
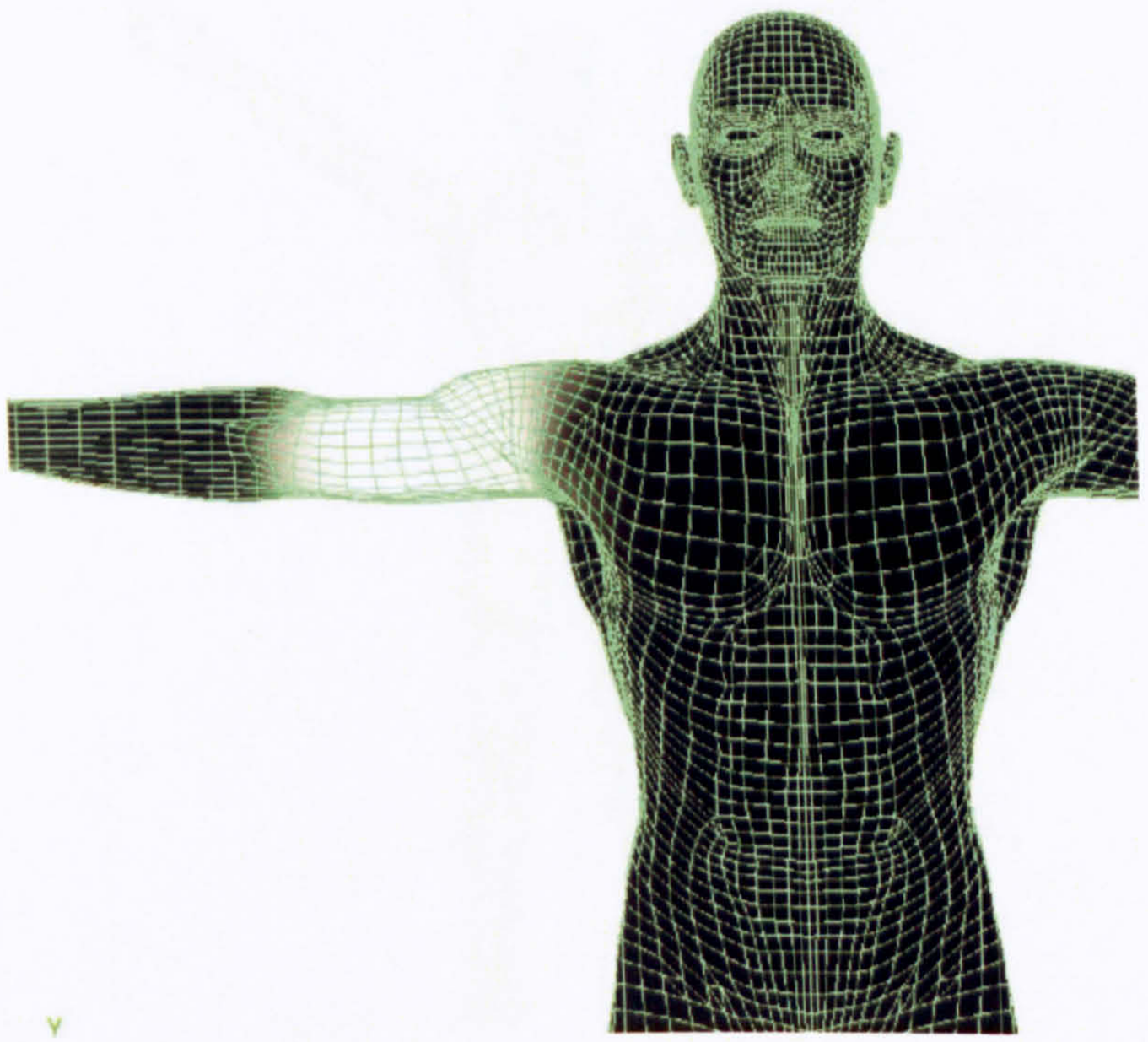
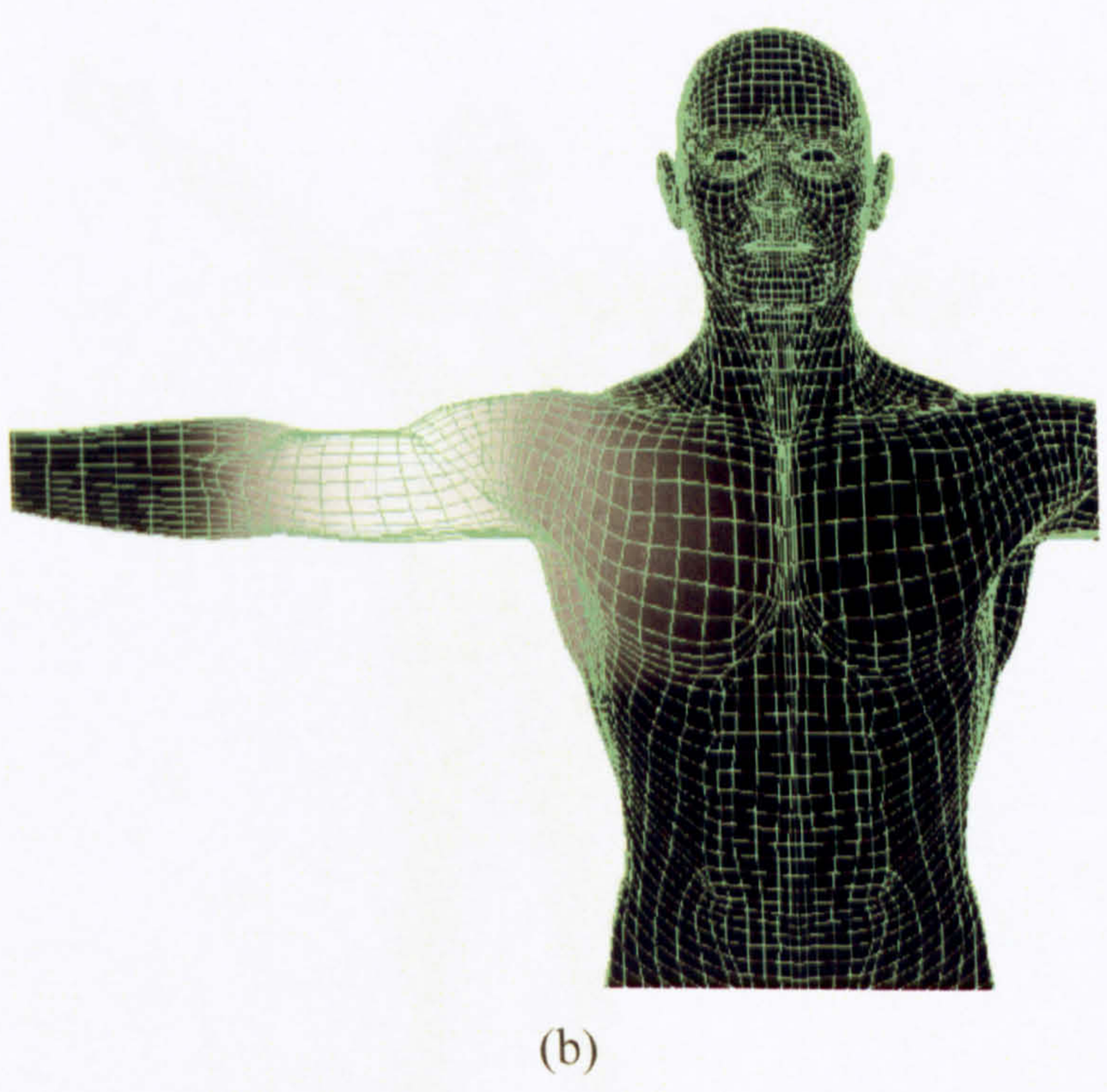


figure5.6: Decomposition based on skeletal joints. A boundary is generated to specify a joint region.

To study this framework's ability of removing irrelevant influences from topologically far joints, we compare its computed weights to the results derived from conventional automatic binding method. An illustration is given in figure 5.7 to demonstrate the weight computation results. In figure 5.8 and figure 5.9, the differences can be seen in the armpit area. The unwanted transformations of vertices in the armpit area as the shoulder moves are eliminated.



(a)



(b)

figure5.7: Comparison of weight computation for vertices in the armpit area using different skin attachment methods. The brightness indicates the change in weight values. (a) our method; (b) Maya smooth binding.

(a)

(b)

figure 5.8: Comparison of deformation effects using weights derived from: (a) our framework; (b) Maya smooth binding. View from front.

(a)

(b)

figure 5.9: Comparison of deformation effects using weights derived from: (a) our framework; (b) Maya smooth binding. View from back.

Another example is to demonstrate the advantage of using a non-planar boundary to segment the mesh. We use a cat model for demonstration.

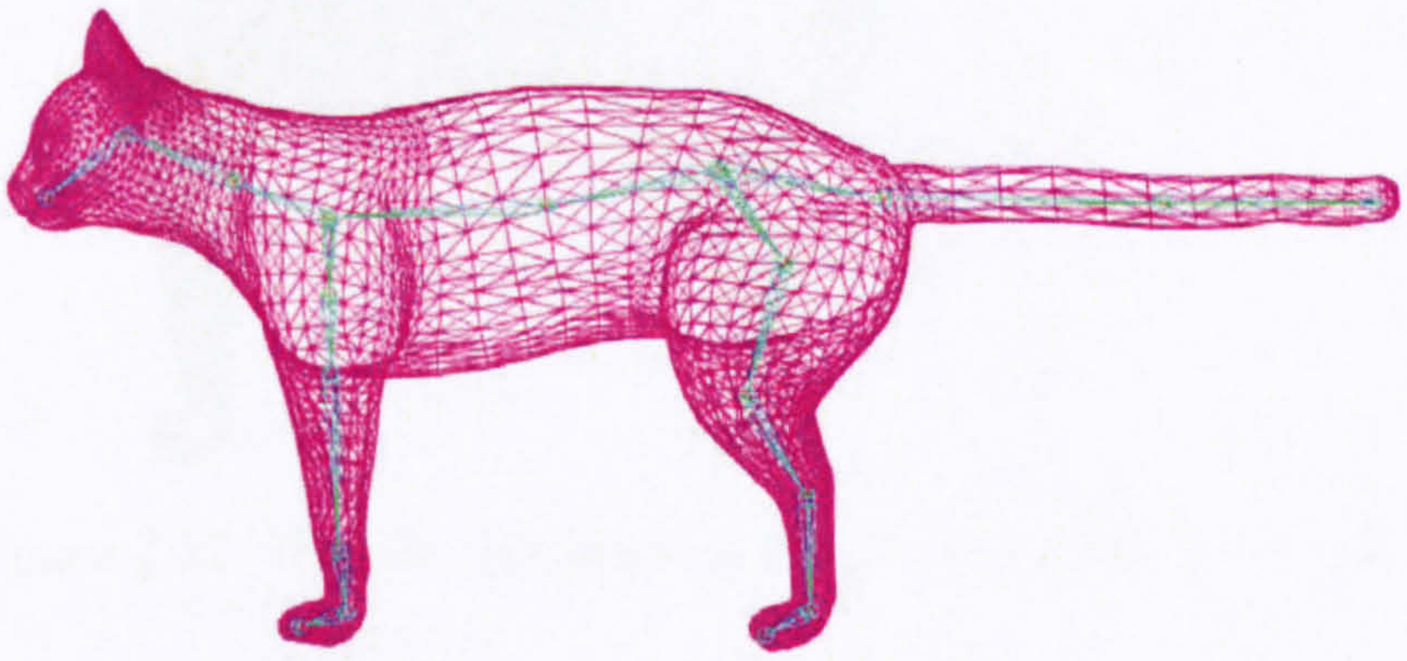


figure 5.10: The original model mesh of a cat. A bulge area can be seen between the leg part and the torso part.

As shown in figure 5.10, if separated by a planar boundary, the bulge part which topologically belongs to the leg would be categorized to a region belongs to the torso. However, using our method, it is capable of classifying this part to 'leg', as illustrated in figure 5.11.

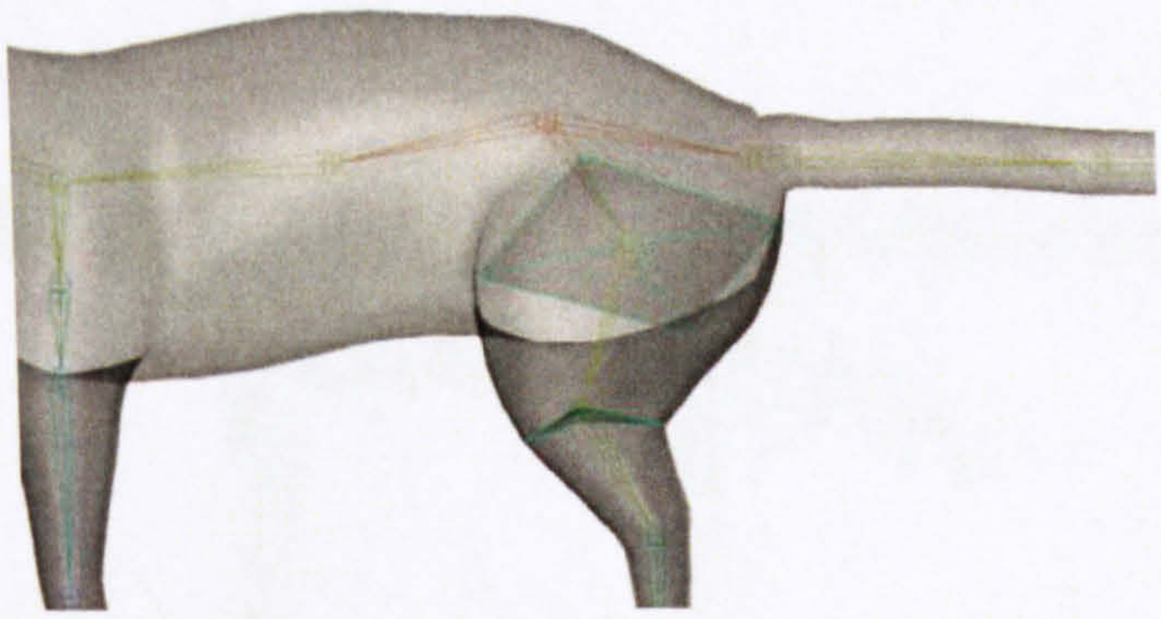


figure 5.11: The region between the leg and the torso of a cat is separated by a non-planar boundary.

The weight computation result is shown in figure 5.12. The deformed effect using weights computed from this segmentation is compared to that using planar

segmentation boundary in figure 5.13.

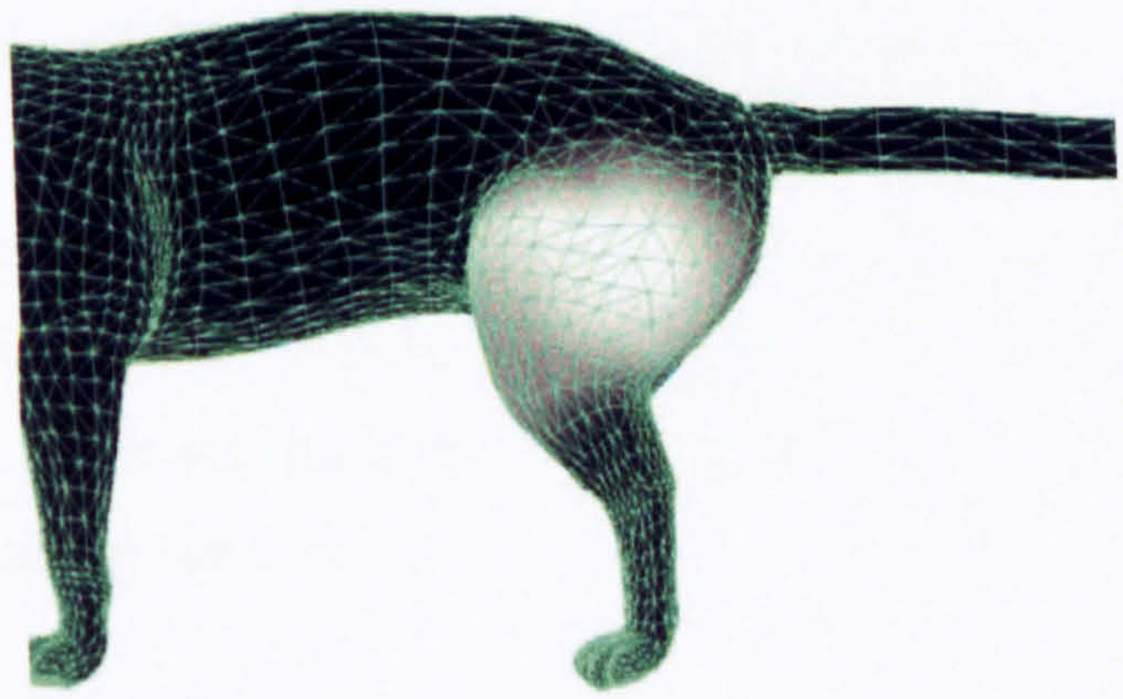
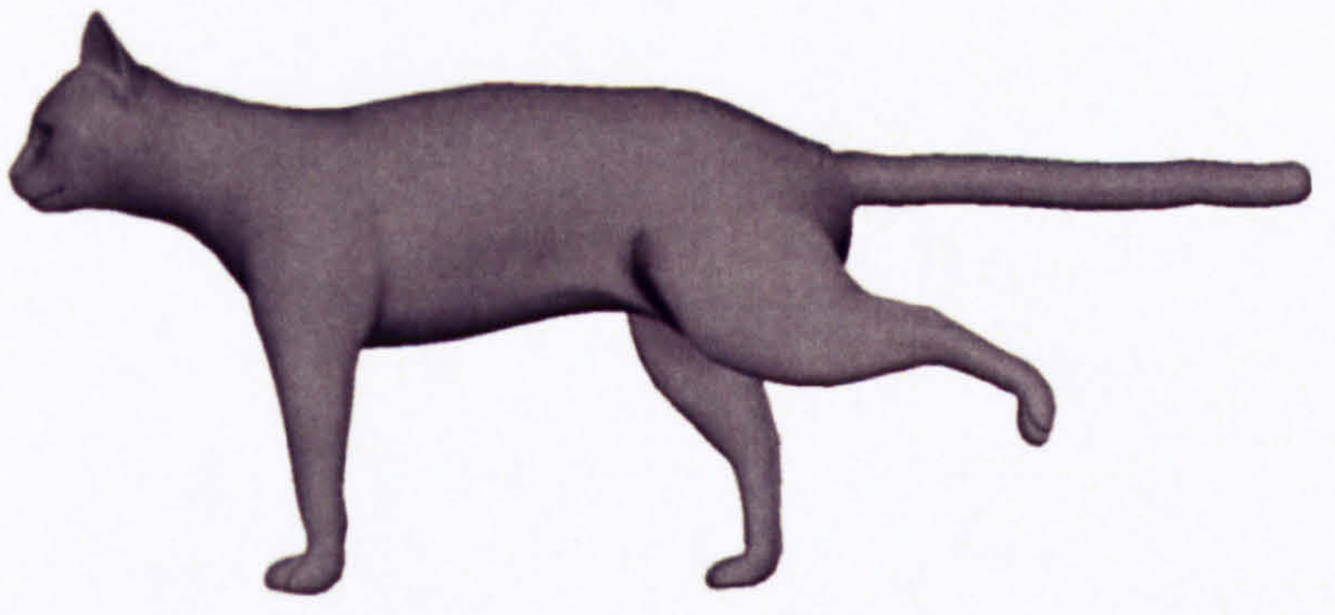
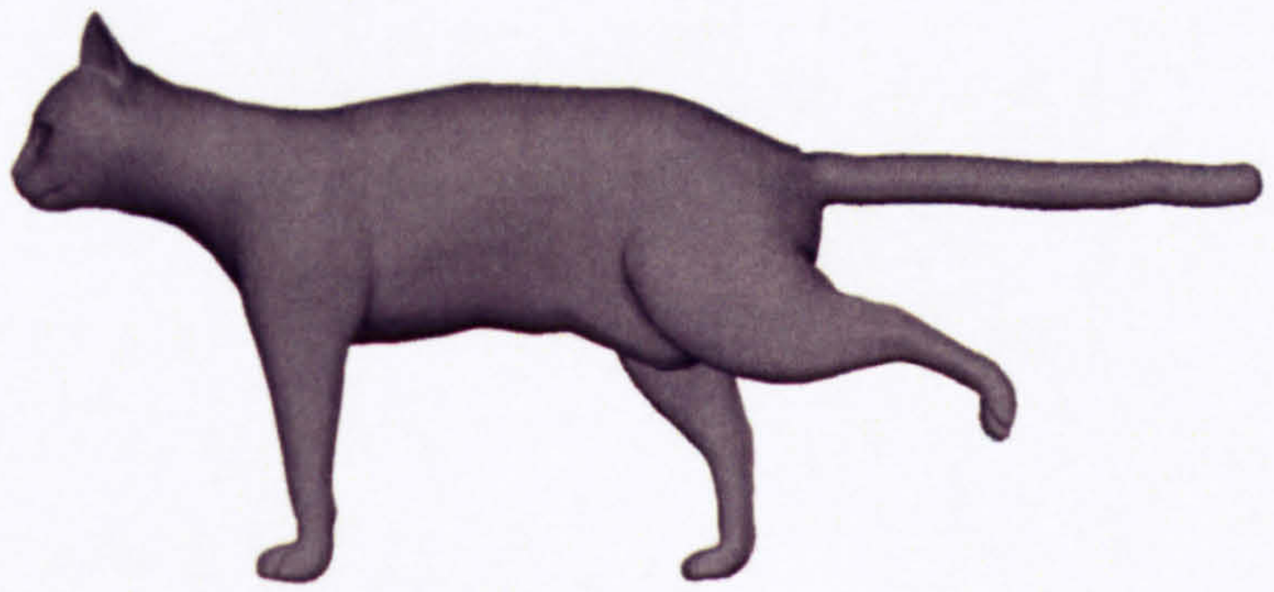


figure 5.12: Weight computation results of the upper leg part.



(a)



(b)

figure 5.13: Comparison of deformed effects by weight computation using: (a) our non-planar boundary in mesh segmentation; (b) conventional planar boundary in mesh segmentation.

The computation involved in our framework is slightly more complicated than conventional automatic binding approach which calculates the distance from a vertex to a joint directly. However, our framework produces far more accurate results that it saves a great deal of time on manual tuning needed by conventional attachment methods and reduces the tedium. Furthermore, this weight computation process is accomplished before animating. It would not affect the rendering speed during run time.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Skeleton generation is of great importance to articulated character animation. Traditionally, this task is done by hand, which is time consuming and experience dependant. Approaches to automatic skeleton generation have been proposed before. However, most of them aim at general use rather than animation so that the skeleton they produce does not suit the needs of animation, others are computationally expensive.

In this thesis, we have presented an automatic skeletonization framework which serves realistic character animation as its particular purpose. This framework is applicable to characters with various shapes and topologies as long as their cross-sections are approximately elliptic.

A new geometric entity, namely 3D silhouette, is introduced in our skeletonization algorithm. A two-level search algorithm is developed to detect the 3D silhouette of a character model. This search algorithm is simple and fast. It handles most character models hence they are usually given at roughly extended poses. Another

algorithm employing connectivity information is presented as well to deal with some extreme circumstance where models are severely occluded.

A coarse curve skeleton is extracted automatically from the 3D silhouette via constrained Delaunay triangulation, which roughly depicts the formation of an animation skeleton. This coarse curve skeleton can be divided into several branches which are used in guiding the decomposition the model. A second 3D silhouette is then extracted from the decomposed mesh, which improves the centrality of the curve skeleton. By identifying necessary joints on the refined curve skeleton, an animation skeleton is generated.

This framework solves the 3D problem of skeletonization in 2D space. Thus it simplifies the computation and is more efficient. The computation complexity of other main skeletonization methods for a model with n vertices is typically $O(n^2)$, while that of our method is $O(n)$. Furthermore, we have compared the skeleton generated using our framework with that using other methods in Chapter 5. The results produced by our method are of much better quality.

Nevertheless, our algorithm has some limitations. The main disadvantage is that the algorithm is restrictive to the shape of the objects. Although it is able to handle most character models, if the model cannot be treated as a series of generalized elliptic columns, e.g. a rock with grotesque shape, the centrality of the generated skeleton is not guaranteed.

Another deficiency is that if the pose of a model is extended to a certain degree that the bending angle of a joint is too small, some joints may not be recognized on the generated curve skeleton. However, this is very rare in practice. At the same time, with key joints being located, to decide a few in-between joints is rather straightforward. After all, the automation aims to facilitate the animation, not to replace animators.

In skeleton-driven animation, to derive the deformation from the skeleton, the skin and the skeleton need to be tied together. *It is a vital step* to attach skin mesh to skeleton with weights which can reasonably reflect the contribution of each joint to a vertex during deformation. The weight distribution made by animation software currently in use is simply based on the geometric distance between a vertex and a joint. It introduces to a vertex many influences which should be considered irrelevant so that manual weight editing is one of the most tedious tasks along the animation pipeline. Researchers have proposed a few automatic weight distribution methods. However, they either do not solve the problem fundamentally, or are designed for a certain deformation algorithm and are not transferable.

In this thesis, a framework has been presented to provide a flexible and more reasonable automatic weight computation algorithm. It starts from segmenting a model mesh into regions of which each corresponds to a joint. These regions are then extended to generate overlapping areas which we call shadow areas, in contrast of the non-overlapping areas which we call dominant areas. Vertices in the skin mesh are classified into these areas. Should a vertex fall into a dominant area of a joint, it is affected only by this joint. Otherwise, it is transformed by all the joints in regions that overlap in this area during deformation. The weight distribution among the associated joints is based on distance from the vertex to the boundary of each related region.

A user-defined parameter is provided in our algorithm to define the size of the shadow area. Depending on the value of the parameter, the blended weights are capable of produce either quasi-rigid or smooth deformation effects.

Examples are provided in Chapter 5 to demonstrate the application of this skin attachment framework. The weight computation results are of better quality than

that produced by conventional method which computes weight directly using the distance between a vertex and a joint. The weights derived from this framework are flexible that they can be used with different deformation methods.

Our skin attachment framework would benefit experienced animators as well as novices. The computation complexity is slightly higher than conventional methods due to an extra step of mesh decomposition. Given that it saves animators from intense labour of manual weight editing, this is considered to be worth it. Fine tuning may still be required. However, our purpose is to accomplish the most tedious part of the task using computers instead of men. Pure automation is not our ultimate goal. As stated earlier, it should not, and never will, replace the craftsmanship of animators.

6.2 Future Work

6.2.1 Future Work in Skeletonization

Regarding the problem of missing joints mentioned above for the proposed skeletonization framework, some methods can be explored in our future work. One direction is to use template matching method for more accurate inbetween joints locating.

Two types of templates are considered for future development. The first template consists of a full-body skeleton of a character where all joints are predefined. The second is a template for a partial skeleton associated with a body segment. For an instance, for an arm of a human model, only middle joint, i.e. the elbow, may need to be specified since the wrist and the shoulder joint are key nodes which can be identified directly from the midpoints of junction triangles after constrained Delaunay triangulation. Using this partial template helps to locate any middle joints by comparing the medial axis branch of the model with the matching template. Such a simplified template may work more effectively than a complete

one.

6.2.2 Future Work in Skin Attachment

A possible improvement of our skin attachment framework in future is its accuracy, although it has outperformed the existing methods. In consideration of computation cost, our current framework uses a simplified boundary line and Euclidean distance in weight computation. In future, the actual boundary line and geodesic distance will be used in implementation. That may bring an even better distribution result.

Another aspect of possible improvements is toward enlarging its range of applicable models. Currently our decomposition method is based on the observation on the anatomical shape of real creatures. Although this observation is correct for most realistic characters, it may not suit some imaginary creature that does not hold an anatomical feasibility. In future, other decomposition methods may be developed to solve this need.

Furthermore, the weight distribution derived from our framework is flexible to be used in various deformation styles, yet we would like to develop a new algorithm in future based on this distribution approach for a more realistic deformation.

REFERENCES

Albrecht, I., Haber, J., Seidel, H.-P., 2003. Construction and Animation of Anatomically Based Human Hand Models. *In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2003, San Diego, USA*. Aire-La-Ville, Switzerland: Eurographics Association, 98-109.

Alias Learning Tools, 2004. *Learning Maya 6: Character Rigging and Animation*. Hoboken, USA: John Wiley & Sons.

Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., Lee, T.-Y., 2008. Skeleton Extraction by Mesh Contraction. *In: International Conference on Computer Graphics and Interactive Techniques, 2008, Los Angeles*. New York: ACM.

Aujay, G., Hetroy, F., Lazarus, F., Depraz, Z., 2007. Harmonic Skeleton for Realistic Character Animation. *In: ACM SIGGRAPH / Eurographics symposiums on Computer Animation, 2007, San Diego, USA*. Aire-La-Ville, Switzerland: Eurographics Association, 151-160.

Baran, I., Popović, J., 2007. Automatic Rigging and Animation of 3D Characters. *In: International Conference on Computer Graphics and Interactive Techniques, 2007, San Diego, USA*. New York: ACM, 72.

Biasotti, S., Falcidieno, B., Spagnuolo, M., 2000. Extended Reeb Graph for Surface Understanding and Description. *In: 9th International Conference on Discrete Geometry for Computer Imagery, 2000.* London: Springer-Verlag, 185-197.

Bitter, I., Kaufman, A. E., Sato, M., 2001. Penalized-Distance Volumetric Skeleton Algorithm. *IEEE Transactions on Visualization and Computer Graphics.* 7(3), 195-206.

Blum, H., 1967. A Transformation for Extraction New Descriptor of Shape. *In: Models for the Perception of Speech and Visual Form, 1967.* MIT Press, 362-380.

Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., Lévy, B., Bischoff, S., Rössl, 2007. Geometric Modelling Based on Polygonal Meshes. *In: International Conference on Computer Graphics and Interactive Techniques, 2007, San Diego, USA.* New York: ACM.

Brandt, J. W., and Algazi, V. R., 1992. Continuous Skeleton Computation by Voronoi Diagram. *CVGIP: Image Understanding.* 55(3), 329-338.

Burtnyk, N., and Wein, M., 1976. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Communications of the ACM.* 19(10), 564 – 569.

Chuang, J., Tsai, C., Min-Chi, K., 2000. Skeletonization of Three-Dimensional Object Using Generalized Potential Field. *IEEE PAMI.* 22(11), 1241.

Clark, B., Hood, J., Harkins, J., 2005. *3D Advanced Rigging and Deformations.* Boston: Thomas Course Technology.

Cornea, N. D., Silver, D., Yuan, X., Balasubramanian, R., 2005. Computing Hierarchical Curve-Skeletons of 3D Objects. *The Visual Computer*. 21(11), 945-955.

Cornea, N. D., Silver, D., Min, P., 2007. Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*. 13(3), 530-548.

Couprie, M., Coeurjolly, D., Zrour, R., 2007. Discrete Bisector Function and Euclidean Skeleton in 2D and 3D. *Image and Vision Computing*. 25(10), 1543-1556.

Culver, T., Keyser, J., Manocha, D., 2004. Exact Computation of the Medial Axis of a Polyhedron. *Computer Aided Geometric Design*. 21(1), 65-98.

Dey, T. K., and Sun, J., 2006. Defining and Computing Curve-Skeletons with Medial Geodesic Function. *In: 4th Eurographics Symposium on Geometry Processing, 2006, Cagliari, Italy*. Aire-La-Ville, Switzerland: Eurographics Association, 143-152.

Eck, M., and Hoppe, H., 1996. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. *In: 23rd annual conference on Computer graphics and interactive techniques, 1996*. New York: ACM, 325-334.

Forstmann, S., Ohya, J., Krohn-Grimberghe, A., McDougall, R., 2007. Deformation Styles for Spline-based Skeletal Aniamtion. *In: ACM SIGGRAPH / Eurographics symposium on Computer animation, 2007, San Diego*. New York: ACM, 141-150.

de Goes, F., Goldenstein, S., Velho, L., 2008. A Hierarchical Segmentation of

Articulated Bodies. *Computer Graphics Forum*. 27(5), 1349-1356.

Goldfinger, E., 1991. *Human Anatomy for Artists: the Elements of Form*. New York: Oxford University Press.

Grigorishin, T., Abdel-Hamid, G., Yang, Y. H., 1998. Skeletonization: an Electrostatic Field-based Approach. *Pattern Analysis and Applications*. 1(3), 163-177.

Hyun, D.-E., Yoon, S.-H., Kim, M.-S., Jüttler, B., 2003. Modelling and Deformation of Arms and Legs Based on Ellipsoidal Sweeping. *In: 11th Pacific Conference on Computer Graphics and Applications, 2003*. Washington: IEEE Computer Society, 204.

Igarashi, T., Matsuoka, S., Tanaka, H., 1999. Teddy: A Sketching Interface for 3D Freeform Design. *In: 26th annual conference on Computer graphics and interactive techniques, 1999*. New York: ACM Press / Addison-Wesley Publishing, 409-416.

James, D. L., and Twigg, C. D., 2005. Skinning Mesh Animation. *In: International Conference on Computer Graphics and Interactive Techniques, 2005, Los Angeles*. New York: ACM, 399-407.

Katz, S., Tal, A., 2003. Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts. In: International Conference on Computer Graphics and Interactive Techniques, *2003, San Diego, USA*. New York: ACM, 954-961.

Katz, S., Leifman, G., Tal, A., 2005. Mesh Segmentation using Feature Point and Core Extraction. *The Visual Computer (Pacific Graphics)*. 21, 649-658.

Kavan, L., and Žára, J., 2003. Real Time Skin Deformation with Bones Blending. *In: WSCG Short Papers Proceedings*.

Kavan, L., and Žára, J., 2005. Spherical Blend Skinning: A Real-time Deformation of Articulated Models. *In: symposium on Interactive 3D graphics and games, 2005, Washington*. New York: ACM, 9-16.

Kavan, L., Collins, S., Žára, J., O'Sullivan, C., 2007. Skinning With Dual Quaternions. *In: symposium on Interactive 3D graphics and games, 2007, Seattle.* New York: ACM, 39-46.

Kavan, L., Collins, S., Žára, J., O'Sullivan, C., 2008. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Transactions on Graphics*. 27(4), No. 105.

Kähler, K., Haber, J., Seidel, H.-P., 2001. Geometry-based Muscle Modeling for Facial Animation. *In: Graphics Interface 2001, Ottawa*. Toronto: Canadian Information Processing Society, 37-46.

Kreavoy, V., Julius, D., Sheffer, A., 2007. Model Composition from Interchangeable Components. *In: 15th Pacific Conference on Computer Graphics and Applications, 2007*. Washington: IEEE Computer Society, 129-138.

Lewis, J. P., Cordner, M., Fong, N., 2000. Pose Space Deformation: a Unified Approach to Shape Interpolation and Skeleton-driven Deformation. *In: 27th annual conference on Computer graphics and interactive techniques, 2000*. New York: ACM Press / Addison-Wesley Publishing, 165-172.

Leymarie, F., and Kimia, B., 2003. Computation of the Shock Scaffold for Unorganized Point Clouds in 3D. *Proc. Conf. Computer Vision and Pattern*

Recognition. 1, 821-827.

Leymarie, F., and Kimia, B., 2007. The Medial Scaffold of 3D Unorganized Point Clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 29(2), 313-330.

Lien, J.-M., Keyserz, J., Amato, N. M., 2006. Simultaneous Shape Decomposition and Skeletonization. *In: ACM symposium on Solid and physical modelling, 2006,Cardiff*. New York: ACM, 219-228.

Lieutier, A., 2003. Any Open Bounded Subset of R^n Has the Same Homotopy Type than Its Medial Axis. *In: 8th ACM Symposium on Solid Modelling and Applications, 2003, Seattle*. New York: ACM, 65-75.

Liu, P., Wu, F., Ma, W., Liang, R., Ouhyoung, M., 2003. Automatic Animation Skeleton Construction Using Repulsive Force Field. *In: 11th Pacific Conference on Computer Graphics and Applications, 2003*. Washington: IEEE Computer Society, 409.

Ma, C.-M., Wan, S.-Y., Chang, H.-K., 2002. Extracting medial curves on 3D images. *Pattern Recognition Letters*. 23(8), 895 – 904.

Ma, W.-C., Wu, F.-C., Ouhyoung, M., 2003. Skeleton Extraction of 3D Objects with Radial Basis Functions. *In: Shape Modelling International 2003*. Washington: IEEE Computer Society, 207.

MacCracken, R., and Joy, K. I., 1996. Free-form Deformations with Lattices of Arbitrary Topology. *In: 23rd annual conference on Computer graphics and interactive techniques, 1996*. New York: ACM, 181-188.

Magenant-Thalmann, N., Laperrière, R., Thalmann, D., 1988. Joint-dependent Local Deformations for Hand Animation and Object Grasping. *In: Graphics interface '88, Edmonton, Canada*. Toronto: Canadian Information Processing Society, 26-33.

Merry, B., Marais, P., Gain, J., 2006. Animation Space: A Truly Linear Framework for Character Animation. *ACM Transactions on Graphics*. 25(4), 1400-1423.

Mohr, A., and Gleicher, M., 2003. Building Efficient, Accurate Character Skins from Examples. *In: International Conference on Computer Graphics and Interactive Techniques, 2003, San Diego*. New York: ACM, 562-568.

Mohr, A., Tokheim, L., Gleicher, M., 2003. Direct Manipulation of Interactive Character Skins. *In: Symposium on Interactive 3D Graphics, 2003, Monterey, USA*. New York: ACM, 27-30.

O'Rourke, M., 1998. *Principles of Three-Dimensional Computer Animation: Modeling, Rendering, and Animating with 3D Computer Graphics*. Revised ed. New York: W. W. Norton & Company.

Pascucci, V., Scorzelli, G., Bremer, P.-T., Mascarenhas, A., 2007. Robust On-line Computation of Reeb Graphs: Simplicity and Speed. *In: International Conference on Computer Graphics and Interactive Techniques, 2007, San Diego, USA*. New York: ACM, (58).

Pizer, S. M., Thall, A. L., Chen, D. T., 1999. *M-reps: a New Object Representation for Graphics*. University of North Carolina at Chapel Hill, TR99-030.

Pratscher, M., Coleman, P., Laszlo, J., Singh, K., 2005. Outside-In Anatomy

Based Character Rigging. *In: ACM SIGGRAPH / Eurographics symposium on Computer animation, 2005, Los Angeles*. New York: ACM, 329-338.

Rohmer, D., Hahmann, S., Cani, M.-P., 2008. Local Volume Preservation for Skinned Characters. *Computer Graphics Forum*. 27(7), 1919-1927.

Schaefer, S., and Yuksel, C., 2007. Example-based Skeleton Extraction. *In: 5th Eurographics symposium on Geometry processing, 2007, Barcelona*. Aire-La-Ville: Eurographics Association, 153-162.

Scheepers, F., Parent, R. E., Carlson, W. E., May, S. F., 1997. Anatomy-Based Modelling of the Human Musculature. *In: 24th annual conference on Computer graphics and interactive techniques, 1997*. NY: ACM Press / Addison-Wesley, 163-172.

Shamir, A., 2008. A Survey on Mesh Segmentation Techniques. *Computer Graphics Forum*. 27(6), 1539-1556.

Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent Mesh Partitioning and Skeletonization Using the Shape Diameter Function. *The Visual Computer: International Journal of Computer Graphics*. 24(4), 249-259.

Shen, J., Magnenat-Thalmann, N., Thalmann, D., 1994. Human Skin Deformation from Cross-Sections. *In: Computer Graphics International, 1994, Melbourne, Australia.*

Shepard, D., 1968. A Two-Dimensional Interpolation Function for Irregularly Spaced-Data. *In: 23rd ACM national conference, 1968.* New York: ACM, 517-524.

Singh, K., and Fiume, E., 1998. Wires: a Geometric Deformation Technique. *In: 25th Annual Conference on Computer Graphics and Interactive Techniques,1998.* New York: ACM, 405-414.

Svensson, S., Nyström, I., Sanniti di Baja, G., 2002. Curve-skeletonization of Surface-like Objects in 3D Images Guided by Voxel Classification. *Pattern Recognition Letters*. 23(12), 1419 – 1426.

Tagliasacchi, A., Zhang, H., Cohen-Or, D., 2009. Curve Skeleton Extraction from Incomplete Point Cloud. *ACM Transactions on Graphics*. 28(3).

Tierny, J., Vandeborre, J.-P., Daoudi, M., 2007. Topology Driven 3D Mesh Hierarchical Segmentation. *In: the IEEE International Conference on Shape Modelling and Applications 2007*. Washington: IEEE Computer Society, 215-220.

Vince, J., 2000. *Essential Computer Animation Fast*. London: Springer-Verlag.

Wade, L., and Parent, R. E., 2002. Automated Generation of Control Skeletons for Use in Animation. *The Visual Computer*. 18(2), 97-110.

Wang, Y.-S., and Lee, T.-Y., 2008. Curve-Skeleton Extraction Using Iterative Least Squares Optimization. *IEEE Transactions on Visualization and Computer Graphics*. 14(4), 926-936.

Wang, X. C., and Phillips, C., 2002. Multi-Weight Enveloping: Least-Squares Approximation Techniques for Skin Animation. *In: ACM SIGGRAPH / Eurographics symposium on Computer animation, 2003, San Antonio, USA*. New York: ACM, 129-138.

Wang, R. Y., Pulli, K., Popović, J., 2007. Real-time Enveloping with Rotational

Regression. *ACM Transactions on Graphics*. 26(3).

Weber, J., 2000. Run-time Skin Deformation. *In: Game Developers Conference.*

Weber, O., Sorkine, O., Lipman, Y., Gotsman, C., 2007. Contex-Aware Skeletal Shape Deformation. *Computer Graphics Forum*. 26(3).

Wilhelms, J., Gelder, A.V., 1997. Anatomically Based Modelling. *In: 24th annual conference on Computer graphics and interactive techniques, 1997*. New York: ACM Press / Addison-Wesley, 173-180.

Wu, F.-C., Ma, W.-C., Liang, R.-H., Chen, B.-Y., Ouhyoung, M., 2006. Domain Connected Graph: the Skeleton of a Closed 3D Shape for Animation. *The Visual Computer: International Journal of Computer Graphics*. 22(2), 117-135.

Xiao, Y., Siebert, P., Werghi, N., 2003. A Discrete Reeb Graph Approach for the Segmentation of Human Body Scans. *In: 4th International Conference on 3D Digital Imaging and Modelling, 2003*. 378-385.

Yang, Y., Brock, O., Moll, R. N., 2004. Efficient and Robust Computation of an Approximated Medial Axis. *In: 9th ACM symposium on Solid modelling and applications, 2004, Genoa, Italy*. Aire-La-Ville, Switzerland: Eurographics, 15-24.

Yang, X., Somasekharan, A., Zhang, J. J., 2006. Curve Skeleton Skinning for Human and Creature Characters. *Computer Animation and Virtual Worlds*. 17(3-4), 281-292.

Yang, X., and Zhang, J. J., 2006a. Automatic muscle generation for character skin deformation: Research Articles. *Computer Animation and Virtual Worlds*. 17(3-4),

293-303.

Yang, X., and Zhang, J. J., 2006b. Stretch It - Realistic Smooth Skinning. *In: International Conference on Computer Graphics, Imaging and Visualisation, 2006.* Washington: IEEE Computer Society, 323-328.

Zhou, Y., and Toga, A.W., 1999. Efficient Skeletonization of Volumetric Objects. *IEEE Transactions on Visualization and Computer Graphics*. 5(3), 196 – 209.

Zuckerberger, E., Tal, A., Shlafman, S., 2002. Polyhedral Surface Decomposition with Applications. *Computers and Graphics*. 26(5), 733-743.