# Sketching-based Skeleton Extraction

Qingzheng Zheng

Submitted for the degree of Doctor of Philosophy

School of Engineering and Computing Science

Durham University

July 2011

## Abstract

Articulated character animation can be performed by manually creating and rigging a skeleton into an unfolded 3D mesh model. Such tasks are not trivial, as they require a substantial amount of training and practice. Although methods have been proposed to help automatic extraction of skeleton structure, they may not guarantee that the resulting skeleton can help to produce animations according to user manipulation. We present a sketching-based skeleton extraction method to create a user desired skeleton structure which is used in 3D model animation. This method takes user sketching as an input, and based on the mesh segmentation result of a 3D mesh model, generates a skeleton for articulated character animation.

In our system, we assume that a user will properly sketch bones by roughly following the mesh model structure. The user is expected to sketch independently on different regions of a mesh model for creating separate bones. For each sketched stroke, we project it into the mesh model so that it becomes the medial axis of its corresponding mesh model region from the current viewer perspective. We call this projected stroke a “sketched bone”. After pre-processing user sketched bones, we cluster them into groups. This process is critical as user sketching can be done from any orientation of a mesh model. To specify the topology feature for different mesh parts, a user can sketch strokes from different orientations of a mesh model, as there may be duplicate strokes from different orientations for the same mesh part. We need a clustering process to merge similar sketched bones into one bone, which we call a “reference bone”. The clustering process is based on three criteria: **orientation**, **overlapping** and **locality**.

Given the reference bones as the input, we adopt a mesh segmentation process to assist our skeleton extraction method. To be specific, we apply the reference bones and the seed triangles to segment the input mesh model into meaningful segments using a multiple-region growing

mechanism. The seed triangles, which are collected from the reference bones, are used as the initial seeds in the mesh segmentation process. We have designed a new segmentation metric [1] to form a better segmentation criterion. Then we compute the Level Set Diagrams (LSDs) on each mesh part to extract bones and joints. To construct the final skeleton, we connect bones extracted from all mesh parts together into a single structure.

There are three major steps involved: optimizing and smoothing bones, generating joints and forming the skeleton structure. After constructing the skeleton model, we have proposed a new method, which utilizes the Linear Blend Skinning (LBS) technique and the Laplacian mesh deformation technique together to perform skeleton-driven animation. Traditional LBS techniques may have self-intersection problem in regions around segmentation boundaries. Laplacian mesh deformation can preserve the local surface details, which can eliminate the self-intersection problem. In this case, we make use of LBS result as the positional constraint to perform a Laplacian mesh deformation. By using the Laplacian mesh deformation method, we maintain the surface details in segmentation boundary regions.

This thesis outlines a novel approach to construct a 3D skeleton model interactively, which can also be used in 3D animation and 3D model matching area. The work is motivated by the observation that either most of the existing automatic skeleton extraction methods lack well-positioned joints specification or the manually generated methods require too much professional training to create a good skeleton structure. We dedicate a novel approach to create 3D model skeleton based on user sketching which specifies articulated skeleton with joints. The experimental results show that our method can produce better skeletons in terms of joint positions and topological structure.

## Declaration

The work in this thesis is based on research carried out in the School of Engineering and Computing Sciences, University of Durham, England, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text. My contribution is that we have designed a new skeleton extraction system which makes use of user sketching as input to construct the articulated skeleton.

Selected Publications:

Gary K. L. Tam, Qingzheng Zheng, Mark Corbyn, Rynson W. H. Lau, "Motion Retrieval Based on Energy Morphing," Ninth IEEE International Symposium on Multimedia (ISM 2007), pp. 210-220, 2007

Q.Z., Zheng, F.W.B., Li, R.W.H, Lau, "Sketching-Based Skeleton Extraction," Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference, pp. 179-186, 2010.

Q.Z., Zheng, F.W.B., Li, "A Mobile Environment for Sketching-based Skeleton Extraction", Journal in World Wide Web :Internet and Web Information Systems, 2010.

# Acknowledgements

With my deepest respect, I am especially indebted to my supervisor Dr. Frederick Li and Prof. Rynson Lau. Especially thanks for Dr. Frederick Li due to his patient on my research guidance. Their insightful remarks and guidance have expanded my view to my research study. This thesis would not be completed without their patience and endless support. I would also like to thank the two examiners Prof. JianJun Zhang and Dr. Ioannis Ivrissimtzis for their valuable and constructive comments on my thesis.

I would thank Mr Christopher Watson for his help on my thesis proofreading. Besides, I would thank Dr. Yonghong Xiang for his help in teaching me how to do the proofreading on my thesis. I would thank Dr. Gary Kwok-Leung Tam for his friendship and guidance at the beginning of my research study. Also thanks to Dr. Wei Lu, Dr. Yan Zhuang and Miss Fan Yang for their friendship.

Most of all, my wholehearted thanks go to my father, my mother, my brother for their unconditional love and support. And my deepest thanks go to my wife Ms Hongwei Li for her love, encouragement and understanding.

# Table of Contents

## List of Figures

# Chapter 1

## 1. Introduction

Nowadays, the 3D mesh model can be obtained from the Internet freely. Thus, it is interesting convenient to make use of those models to teach amateurs the animation concept with the help of skeleton-driven manipulations. Thus, given some existing mesh models which contains lot number of triangles and vertices, how to create skeletons using those models is an interesting research topic. In computer graphics, a mesh is a representation of 3D models. Mesh $M$ [2] is defined as a pair of $(K,V)$, where $K$ is a simplicial complex representing the connectivity of the vertices, edges, and faces, thus determines the topological type of the mesh; $V=\{v_1,...,v_n\}$ describes the geometric position of vertices in $R^3$.

Free Form Deformation (FFD) technique [3] is a representative mesh deformation method in early days. Later, multi-resolution editing approaches [4, 5] have been developed for either detail-preserving based purpose or reconstructed mesh optimization purpose. In the past few years, differential coordinates mesh deformation techniques [6-11] have been developed for detail-preserving based deformation purpose.

Skeleton-driven based methods [12-15] manipulate the skeleton (either created manually or automatically from some skeleton extraction methods) instead of the mesh model. A skeleton contains many bones which are connected by joints. Each bone is associated with a set of surface vertices from the mesh model. The vertices position is controlled by their associated bones. To perform animations, some mature software like Maya or 3D studio has been developed. However, a user has to be trained for a long time before they can construct well-behaved skeletons. Then one can make use of the generated skeleton to produce animations. Also the animation quality relies on the skeleton quality. Researchers have developed many skeleton extraction methods [16-27] to produce skeleton for mesh models. Some methods require a user to create a skeleton manually, which require a user to be trained, such as methods in Maya software. To liberate a user from long term training, we propose an easy way to generate skeletons which have well-positioned joints specifications.

## 1.1. Motivation

There are two types of skeleton extraction methods: manual construction methods and automatic extraction methods. Manual skeleton construction methods are widely adopted in professional software like Maya or 3D Studio. The manual construction methods build up skeleton based on a user's manipulations. However, the user has to be trained before performing skeleton construction. Also there are lots of post-work to be done in skeleton-driven animation, like skinning and weight setting for animation, which are tedious and time-consuming. For these reasons, automatic skeleton extraction has been well studied in recent years. But most of the exiting works focus on 1D curve skeleton extraction which does not consider calculating the joint positions. This limitation narrowed the application the skeleton can be applied.

We want to combine the advantages of manual construction methods and automatic extraction methods together to build a semi-automatic skeleton extraction system. In the system, we allow a user to draw freely on a mesh model. Then we perform a mesh segmentation process based on the sketched lines as input. Finally we extract bones from each mesh part and connect them together to construct the final skeleton. The users do not need to be trained to use our system.

In summary we have the following objectives:

1. Simplify the skeleton generation process. Traditional manually creation methods require a user to be trained before performing skeleton creation tasks. Our method requires only simple sketches from the user.
2. Construct skeleton model based on the user's specification. The skeleton should contain joint specification, and can also be used to distinguish different parts of the mesh model.
3. Evaluate the quality of our skeleton from two aspects: the topology features and the skin mapping quality.
4. Build a real-time animation framework to let a non-trained user perform skeleton-driven animations.

## 1.2. Basic Definition and Notations

In this thesis, we proposed a novel approach to generate the skeleton from a given mesh model and user's sketching activities. We target at amateur users such as children or students which are interested in the animation related work.

In our work a bone is defined as a set of sequential nodes, in which the two end nodes are called joints. The bone is represented as:

$$B = \{J_0, d_1, ..., d_n, J_1\} \tag{1.2.1}$$

where $J_0$ and $J_1$ are two joints and $d_i$ ($1 \le i \le n$) is the node inside a bone. $n$ indicates the number of nodes within bone $B$. $\{d_1, ..., d_n\}$ represents a ordered set of nodes, we also use a vector $plist$ to store those nodes. So, we also use $B = \{J_0, plist, J_1\}$ to describe the data structure of bone.

The node is the basic element which describes a geometric location of a point. The joint is a special node which is associated with one or more bones whereas the node is associated with one bone only. To visualize the bones in our system, we connect the sequential nodes and joints together to form a poly-line. And our skeleton is a bone graph, which is connected by joints.

## 1.3. Research Problems

To achieve the above objectives, we have to solve the following problems:

1. Sketching analysis problem. A user may have duplicate drawings, or a sketched line may contain great curvature changes, which means that this sketched line needs to be divided into several parts. We need a method to clarify a user's intention by analyzing his/her drawings.
2. Skeleton extraction problem. Most of the existing 1D curve skeleton extraction methods do not specify well-positioned joints. We need a method to generate well-positioned joints which connect two or more bones in a skeleton. Also to distinguish different parts of a mesh model, we expect the bones inside a skeleton should have a mapping between mesh model parts and their correspondent bones.

To extract a user's intention from his/her drawing, we need to solve two issues: ***multiple sketched lines matching,*** which is used to cluster similar sketched lines into one group and ***single sketched line analysis,*** which is used to analyze the topology feature of a single sketched line. By investigating the existing skeleton extraction methods, we notice there are several issues which haven't been solved. Some of the methods are sensitive to the surface details, such as the Medial Axis Transform (MAT) base methods [28]. Some of the methods can only produce shrunken skeleton, such as the thinning method in [24]. Some of the skeleton extraction methods make use of the geometric information to construct the skeleton, which do not consider the semantic meaning of the topology. Thus the result from those methods may contain wrong topological structure from the user's point of view.

To obtain a semantically meaningful skeleton structure, some work [29] has introduced the feature point concept to construct a skeleton manually. Most of the feature points are the starting points to extract the skeleton. But the existing methods do not specify well-positioned joints in their skeleton due to the lack of measurements to detect those joints.  Although some of methods [17, 25] can generate good results with regard to the topological representation, the existing methods focus on 1D curve skeleton extraction and do not consider the importance of joints in animation applications. Besides the joint issue in the existing work, there is no benchmark to evaluate the correctness of the skeleton. We investigate the property of the animation based skeleton, and design a metric to evaluate the skeleton's quality.

## 1.4. Proposed solutions

To extract a user's intention from his/her drawing, we need to solve two problems: single sketched line analysis and multiple sketched lines matching. In the single sketched line checking process, we need a method to check the topology feature for any new sketched line. If a sketched line contains large curvature changes, we need to divide the sketched line into several parts. In multiple sketched lines matching process, we need to identify whether a new sketched line represents a new topological feature or a duplicate sketching, which is similar to the existing sketched lines.

With regards to the skeleton extraction issue, we have the following requirements:

1. The skeleton should contain important joints specifications, which are used to distinguish different components of a skeleton model.
2. For the usage of animation, a skeleton should contain a direct mapping between bones and its correspondent mesh parts.

Based on these requirements, we adopt a segmentation based skeleton extraction method in our system. Our method is inspired by the concept of minima rule [30, 31], which is widely used in several part-type segmentation methods. The minima rule states that human vision tends to define areas of minimum negative curvatures, i.e., concave shape areas, as interfaces separating between mesh model parts [1]. This is a good criterion in mesh decomposition, where the centers of decomposition boundaries define the natural location of well-positioned joints. However, the minima rule based methods can only be used in the concave region based decomposition. For models like tube-bar structure, it is hard to define the final decomposition positions using minima rule strategy only. Thus we designed a new segmentation criterion, which satisfies both the minima rule and the definition from user sketching. We adopted the Level Set Diagram (LSD) method [32, 33] to extract bones from each segmentation part. Finally we combined those bones together to construct the skeleton for a given mesh model.

## 1.5. Thesis Organization

This thesis is organized as follows: In Chapter 2, we give a brief literature review in sketching related work and skeleton extraction methods. We then discuss our research goals, the challenges and proposed solutions in Chapter 3. In Chapter 4, we discuss the sketching analysis method that we used in our system to extract user’s intention from their sketched lines. In Chapter 5, we talk about the segmentation method that we proposed to divide the mesh into several meaningful parts. In Chapter 6, we discuss the skeleton extraction method that we have adopted in our skeleton extraction system. In Chapter 7, we give a conclusion on our research work and discuss the possible future work.

# Chapter 2

## 2. Literature Review

### 2.1. Introduction

Our target is to create an interactive skeleton extraction platform, which considers user sketched lines as input and creates a skeleton for a given mesh model. To achieve this target, we studied the following areas: interactive sketching, skeleton extraction and skeleton evaluation. We allow users to draw freely on the screen and our system depicts the topology definition of their desired skeleton from their drawings. With regards to the skeleton extraction, we have the following requirements:

1. The skeleton should have well-positioned joint specifications. The well-positioned joints are used to distinguish different parts of a mesh model.
2. The skeleton should represent the topology features of its corresponding mesh model.
3. The skeleton should contain the mapping between bones and mesh surface elements (i.e. triangles or vertices on the mesh model).

A lot of skeleton extraction methods have been proposed in recent years. But none of them seem to fit in all the requirements that we have listed above. In this chapter, we will discuss the existing methods which are closely related to our requirements. We divide the skeleton extraction problem into mesh segmentation problem (see Chapter 5) and skeleton extraction problem (see Chapter 6).

To understand the existing techniques used in sketching analysis, we studied the existing works which make use of sketched lines as input in the 3D model creation and manipulation areas. We will give a survey in section 2.2. With regards to skeleton extraction, there are two major methodologies: voxel-based methods and surface-type methods. Voxel is the basic element in volume-based models. “Voxel” is short for “Volumetric Pixel”, or more correctly, “Volumetric Picture Element”, which is a volume element, representing a value on a regular grid in 3D space. On the other hand, surface-type models make use of polygon mesh to represent the model structure. Usually, a polygon mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral mesh model in 3D computer graphics. There are many representations of faces, in which triangles and quadrilaterals are the most two common representations. In our work, we make use of triangle-based models only.

Volumetric-based methods make use of different shrinking or thinning strategies to extract the topology feature from a voxel-based model. Most of the existing methods are divided into two classes: thinning based methods and distance transform based methods. A thinning algorithm is a kind of “boundary peeling” process, where the voxel-based model is iteratively peeled layer-by-layer. Distance field is used as the constraint of the thinning process. Most of existing thinning processes make use of different distance metrics to check the priority of a voxel during the boundary peeling process. Surface-type methods contain many different ways to construct the skeleton. We classified the surface-type skeleton extraction methods as:

1. Mesh contraction based method,
2. Interior point computation based method,
3. Surface feature point based method ,
4. Segmentation based skeleton extraction method.

Mesh contraction methods perform iterative shrinking process to the inverse of a mesh model's surface normal direction while preserving some feature point positions to maintain the whole topological structure. Interior point computation based methods compute some interior points which have different radius to measure the volume of a mesh model as the skeleton points. Surface feature point based methods make use of some feature points as the starting points to extract a skeleton. Usually such a method makes use of geometric distance to compute a set of topology contour lines to construct the skeleton. Segmentation methods decompose a mesh model into several parts and then extract bones for each part. The segmentation boundaries are used to connect all parts together to form the final skeleton. We summarized the first three types of methods in section 2.4, whereas we isolate segmentation based methods in section 2.5. In the end, we also give a survey on Linear Blend Skinning method as well as the Laplacian mesh deformation technique in section 2.6. We have combined those two methods to perform our skeleton-driven animation.

## 2.2. Related Work in Sketching Analysis

Freehand sketching is a natural and efficient way of expressing certain ideas. This technique has been widely used in modeling and editing approaches [34-36]. Some 3D modeling tools [37, 38] make use of user's interactive sketching as input to create 3D polygonal surfaces or parametric surfaces. Besides the model construction applications, sketching can also be used as manipulators in different sorts of areas. For example sketch-based deformation tools [9, 39, 40] utilize sketched lines as manipulators to deform 3D models into different poses.

We focus on "user-centered" techniques to provide more intuitive interfaces and interactive tools which allow a user to create a skeleton easily and intuitively without too much professional knowledge of 3D background. We allow users to sketch freely on the surface of the mesh model.

And the system will generate their desired skeleton model from their drawing. Based on this expectation, we explore the most recent work and try to find an ideal solution to fit in our need. Most of the existing sketch-based related works assume the sketched lines are correct. However, when a user sketches on the screen, it is hard to construct the 3D structure for a given 2D sketched line, as there are many possible solutions. Also the user's drawing may not be accurate and may even contain mistakes. For example a user might draw a single line which contains large topology changes or a user might draw some duplicate lines within the same region. We need a method to eliminate those mistakes before using those sketched lines in our skeleton extraction system.

### 2.2.1. Interactive Sketching technique in 3D Modeling

A typical procedure for geometric modeling is to start with a simple primitive such as a cube or a sphere, and gradually build a more complex model through successive transformations or a combination of multiple primitives. Also people make use of several deformation techniques to create a wide variety of smooth and complex models by interactively manipulating control points or 3D widgets. The sketch-based methods are used to identify some built-in model construction commands, which will be performed in a rapid model construction process.

Zeleznik [41] defines a set of primary gestures. He proposed an application [41] to create 3D scenes by defining primary gestures to perform the 3D model construction. The built-in gestures are a set of sequential strokes which correspond to important visual features. The strokes are those sketched lines which are projected on the screen. For example drawings of three non-collinear line segments which meet at a point imply a corner. By retrieving the sketched gesture a system is able to identify the task to be performed in the model construction process. By designing some “inflation” operations which converted a closed curve in the plane into a 3D shape whose silhouette from the current view was that curve on the view plane, Igarashi built an intuitive model construction system in [38]. Karpenko [42] made use of variational implicit surfaces as a representation in a free-form 3D model construction. Rather than considering sketched lines as silhouette, Das [43] made use of the connected network lines as the 3D shape outline to construct a 3D model. Such a 3D modeling development was based on the use of computer vision, human perception and new interactive techniques (such as different gesture operations).

### 2.2.2. Sketching in Model Editing and Deformation

On top of sketch-based applications user sketching can support various mesh model editing operations: such as cutting, extrusion and erasing. To allow more complicated shape editing, Nealen et al. let a user to specify the region of interest (ROI) on a mesh model and express how to modify the shape of this ROI by sketching [9]. They then applied the mesh Laplacian operator on the ROI which could create the deformed mesh model whilst preserving the local mesh features. By introducing the sketching activity a user can specify the region where he wants to modify directly.

In fact sketching is a typical way for animators to express mesh model motions quickly [44]. Kara [45] proposed an application, which made use of sketched lines and template model to modify and design new models. Davis [46] proposed a method to take user sketched 2D skeleton model as input and create some possible character poses for animation generation. Chang [47] proposed an interface between 3D meshes and 2D sketching with the inference of two sketched lines which were marked as reference curve and target curve. The reference curve was used to define the ROI region to be deformed, whereas the target curve defined the constraints for the ROI, which gave guidance on how the ROI will be deformed. The sketch-based method can easily specify the ROI. Also by considering the sketched lines as target curve, one can easily deform the shape of ROI into the new pose as the target curve defined. Similar to the facial animation, Kho and Garland adopted sketched lines as manipulators in [40]. In their system, a user has to draw two lines to perform deformation task. One line is considered as a reference curve which defines the region where the user was interested in. Another line is considered as a target curve, which defines the deformation constraints, such as the rotation matrix for vertices inside the ROI. The sketching activities in sketch-based deformation light up the motivation of our research work on sketch-based skeleton extraction. To our knowledge, there is no published work to allow users to create skeleton based on sketching.

## 2.3. Skeleton Extraction from Volumetric Models

Volume-based skeletonization promises to be an efficient method for compact shape description. Figure 2-1 shows an example of a voxel-based model, as is illustrated in Wolfire's Blog [48] .

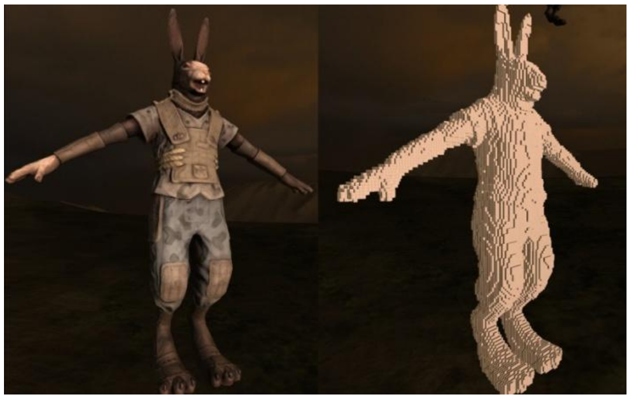

**Figure 2-1 An Example of Voxel-based Model from [48]**

A voxel is the smallest unit cube in the volume, with its eight vertices taking values of zero or one. A voxel is defined as an inside voxel if all its vertices take a value of one. If all its vertices take a value of zero, we call it an outside voxel (background voxel). Otherwise, the voxel is considered as a boundary voxel. Both boundary voxels and inside voxels are called mesh model voxels. Given a voxelized representation, to produce a skeleton, one may remove boundary voxels [49] or pull interior voxels together [24] while maintaining the mesh model topology to shrink the volumetric model. The 3D voxel based model is represented by a 3D array $F = \{ f_{ijk} \}$, when $i$, $j$ and $k$ are integers such that $1 \le i \le M$, $1 \le j \le N$, $1 \le k \le L$ ($M$, $N$ and $L$ are the dimension size for the 3D array), and $f_{ijk}$ represents a density value at the i-th row, j-th column and the k-th plane. If $f_{ijk} = 1$, the voxel is a known as an object voxel (also called black voxel), which is visible, if $f_{ijk} = 0$, the voxel is known as a background voxel (also called white voxel), which is invisible.

### 2.3.1. Thinning Process

Thinning method was first used in 2D image processing, such as fingerprints recognition, scanned-in letters or DNA[1] structures and human organs etc. Figure 2-2 is an example of thinning procedure which comes from [50].

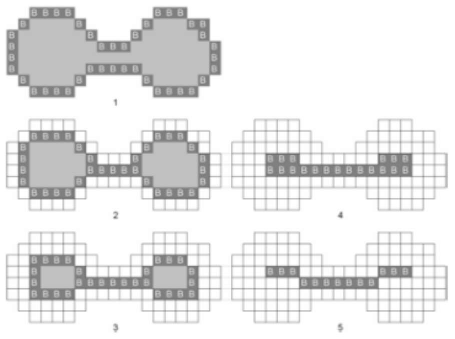

**Figure 2-2 Thinning Method from [50]**

[1] As described in Wikipedia, DNA is short for Deoxyribonucleic acid which is a nucleic acid that contains genetic instructions used in development and functioning of all known living organisms.

A thinning algorithm is a kind of "boundary peeling" process, where the voxel-based model is iteratively peeled layer-by-layer. During the peeling process, each boundary voxel is tested against a set of topology preserving conditions and possibly removed. The algorithm only needs to preserve those voxels which are the key elements on the topology presentation of the model. One widely used condition is the connectivity check among voxels. Shrinking is a processing that replaces a deletable inside voxel by a boundary voxel while preserving topology. In a shrinking algorithm, all deletable voxels are deleted while in a thinning algorithm only parts of deletable voxels are really deleted [51].

One example of the topological property check is denoted as the connectivity of voxels. Given a voxel $p$, there are three types of neighborhood between $p$ and its adjacent voxels. If two voxels share one face, the neighborhood type is defined as 6-neighborhood (6-n), which is denoted as $N_6(p)$. Similarly, if two voxels share one face or one edge, we call 18-neighborhod (18-n), which is denoted as $N_{18}(p)$, if two voxels share one point or one edge or one face, we call 26-neighborhood (26-n), which is denoted as $N_{26}(p)$. According to the definition from [52], two voxels $P$ and $Q$ are said to be 6-connected (6-c), (18-connected(18-c), 26-connected(26-c)), if a sequence of voxels $P_0 = P$, $P_1, \ldots, P_n = Q$ exists, such that each $P_i$ is in the 6-n (18-n, 26-n) of $P_{i-1}$ ($1 \le i \le n$) and all $P_i$'s have the same density value as $P$ and $Q$.

Morgenthaler described the "simple point" in [53], which was widely used in many thinning algorithms. A simple point is a mesh model voxel that can be removed without changing the topology of the mesh model (see [54] for a complete review of digital topology). By monitoring the Euler number of the voxel model, one can detect the topological change during the voxel removal process (see [52] for details). One of the most properties of simple points is that they can be identified locally. Namely, one can determine if a voxel is a simple point or not by inspecting its local neighborhood, which makes the thinning algorithm more efficient in skeleton extraction.

Based on the selection of simple points, Cornea categorized the thinning algorithms into three types: directional thinning methods, subfield sequential thinning methods and full parallel algorithms in [50]. Directional thinning methods remove boundary voxels only from one particular direction (i.e. up, down, east, west). These methods are sensitive to the order in which the different directions are processed, and the resulting skeleton may not be located in the center of the mesh model. Subfield sequential thinning methods divide the discrete space into subfield. At each sub-iteration, only voxels belonging to one of subfields are considered for deletion. Full paralleled algorithms consider all the boundary points for deletion in a single thinning iteration. To preserve skeleton topology a voxel's neighborhood must be inspected to decide whether the voxel is deletable.

The boundary elimination process cannot provide accurate topology description without adding more constraints to the peeling process. Wang introduced a shrinking process [24] to provide such constraints which makes the result more smooth and accurate on the shape structure. Their shrinking process was an iterative least square optimization process. In their work, they defined two energy terms: boundary constraint $E_b$ and edge contraction $E_Q$. The boundary constraint was used to preserve the model’s geometry, whereas the edge contraction was used to maintain the shape of the mesh model. They attempt to reduce the edge lengths to shrink a model. To maintain the shape, the contraction amount for different edges varies. Wang made use of the boundary constraints to find the contraction amount for boundary edges.

Wang defined two energy terms as:

$$\begin{cases} E_b = \sum | v_b^{t+1} - v_b^t |^2 \\ E_Q = \sum_{(i,j)\in E} | (v_i^{t+1} - v_j^{t+1}) - (p_{ij}^t)^2 (v_i^t - v_j^t) |^2 , \ \mathrm{v} \in \mathrm{V} \end{cases} \tag{2.3.1}$$

where $v_b$ is the boundary voxel and $t$ indicates the steps of thinning iteration. $p_{ij}^t = l_{ij}^t / l_{ij}^{t-1}$ is the contraction ratio and $l_{ij}^t, l_{ij}^{t-1}$ are the length of $v_i^t - v_j^t$ and $v_i^{t-1} - v_j^{t-1}$ respectively, $v_i^t - v_j^t$ defines an edge for two adjacent voxels $v_i^t$ and $v_j^t$.

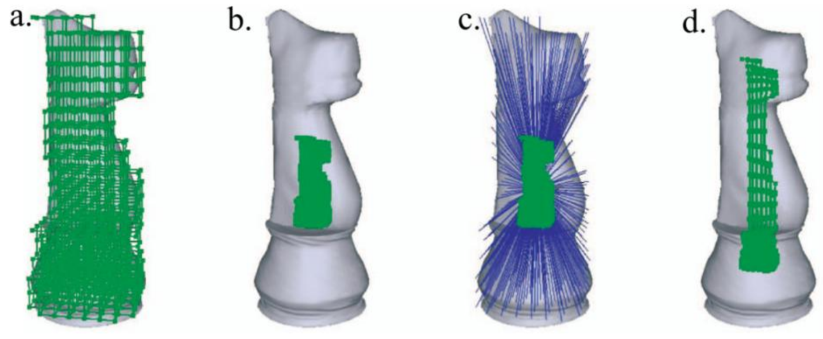

**Figure 2-3 Shrinking Process from Wang's Work from [24]**

Figure 2-3 is a demonstration of the shrinking concept from [24]. Figure 2-3.a is the result of the voxelization process. Figure 2-3.b is an example of homogeneous shrinking result without boundary constraints. Figure 2-3.c is the illustration of the boundary constraints which are used to preserve the original geometry of the model. And Figure 2-3.d is the result of Wang's method which uses boundary constraints as the forcement to preserve the geometry of the model.

The shrinking pre-processing makes the skeleton result more accurate compared to those thinning processes which have no pre-processing. However, the shrinking process makes the skeleton result even smaller than the real skeleton model size (see Figure 2-3.d as illustration). This is because the thinning algorithm cannot guarantee the full topology preservation. Also the created skeleton has no mapping relationship with the existing model, which further limited the application usage of the thinning based method in deformation related regions.

## 2.3.2. Distance Transform Based Methods

Besides the thinning algorithms, researchers notice that they are able to reduce the iteration process and get a smooth skeleton by using some distance functions. Distance field [27] may be applied to define a distance transform for each interior point of a mesh model and detect ridges of the field to obtain candidate voxels for connecting them to generate a skeleton. Such method is not robust due to the "noise"[2] data appeared on the mesh model. In 3D model processing area, lots of metrics (such as the normal variation or curvature changes) have been designed to find out some feature points. Those feature points are critical in many applications, such as mesh segmentation, skeleton extraction etc.

For each interior point $P$ of a 3D mesh model $O$ , the distance field is defined as the smallest distance from that point to the boundary $B(O)$ of the mesh model:

$$\underset{P \in O}{D(P)} = \underset{Q \in B(O)}{\min(d(P,Q))} \tag{2.3.2}$$

where $Q$ is the boundary mesh model of $O$ , and $d(P,Q)$ is a distance function.

Most of the distance field-based methods use the following steps to extract the final skeleton:

1. Find out ridge points (i.e. local maxima, saddles points).

[2] During the feature extraction process, some other points may be captured as the feature points but would result in bad outcomes.

2. Pruning remains voxels.
3. Connect the voxels to build up the final skeleton.

Distance field was used as the constraint of the thinning process. Most of existing thinning processes made use of different distance metrics to check the priority of a voxel during the boundary peeling process. For example some methods [55-57] make use of Euclidean distance to define the priority of the voxels candidates selection. Gagvani and Silver proposed a parameter controlled thinning mechanism [58] to determine the priority of removable voxels, which involved comparison between voxel and the average distance field of its neighborhoods. Similar to this concept, Bitteret proposed a gradient searching algorithm [59] to detect boundary voxels for removal, which involved detecting neighborhoods of non-uniform gradient. Bouix made use of divergence calculation as the priority function together with a user-defined threshold to find the removal voxel candidates in [60]. Combined with a distance-from-a-source field, Zhou proposed a voxel coding approach [27] to create a set of critical voxels.

After running the thinning process, the size of remaining voxels is still large. To reduce the size of the voxels, several methods have been applied to optimize the final skeleton, such as sphere coverage of a path tree in [23], boundary visibility from candidate voxels in [61], thinning criteria in [33] or clustering criteria in [62].

After the pruning step, the remaining voxels are usually disconnected and the final step involves reconnecting them to product the final 1D curve skeleton. Most algorithms make use of minimum spanning trees [62, 63], shortest paths [23, 33, 61] or other graph algorithm to connect the remaining voxels.

The distance field based methods can also be used to create the medial surface. But they cannot generate a 1D curve skeleton without additional techniques to prune the medial surface.

The main advantage of the distance field based method is the low computation cost in the algorithm.

### 2.3.3. Summary

Volumetric based skeleton extraction methods cannot guarantee the smoothness of a skeleton. Such methods produce 1D curve skeleton which has no joints been specified. Usually the smoothness of a curve skeleton is defined as the variation of the curve tangent direction as one move along the curve skeleton. More precisely, the smoothness can be measured as the angles variation between tangent directions at successive locations along the curve skeleton. During the extraction process, the extracted skeleton is smaller than the real topology features [24] or it contains undesired branches which require pruning to eliminate those branches. Thus the thinning or shrinking processes cannot preserve the exact geometry information which makes the extracted skeleton is always smaller than the real model. In general volumetric methods may not guarantee producing good results due to information loss from discretization.

Also the extracted skeleton from volumetric based methods is only 1D curve skeleton which only describes the topology features. Those 1D curve skeletons can be used in mesh model matching, feature tracking or even model reconstruction. However, such skeleton does not suitable for animation related purpose. There are two problems which prevent us from using such kind of methods. First, the skeleton contains no well-positioned joints specification, which is useful in articulated animation area. Second, the thinning algorithm destroyed the structure of model. To make use of skeleton in animation, one has to build up correspondent mapping between skeleton and the original model.

## 2.4. Skeleton Extraction for Surface-type Models

In our research work, we make use of triangulated surface model as the benchmark model. Figure 2-4 is an example of triangulated surface model.

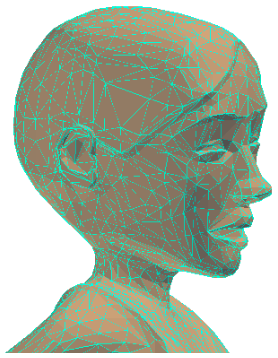

**Figure 2-4 One of the Mesh Model example in Our Experiment**

In this section, we will discuss the following three types of skeleton extraction methods:

1. Mesh contraction based methods
2. Interior points computation based methods
3. Surface feature points extraction based methods

### 2.4.1. Mesh Contraction

Model contraction based method [17] tried to extract skeleton from an iterative contraction process by adding some position constraints to prevent the homogeneous shrinking. Interior distance function based methods [28, 64] also tried to simulate the shrinking process, which made use of gradient flow and surface normal to estimate the interior skeleton positions. The radial basis function [64] and the MAT technique [28] were the most two common examples for the interior distance function based methods. Surface distance function based methods [22, 29, 32] tried to construct contour lines from some feature points on the surface of a mesh model. Reeb graph or Level Set Diagram was the traditional technique using this strategy. Some other methods [65-67] tried to segment the mesh model first and then construct the skeleton from the segmentation result, which is also the choice of this thesis.

Au et al. contracted the mesh geometry into a zero-volume skeletal shape using implicit Laplacian smoothing algorithm with global positional constraints in [17]. Then by applying face removal operations via edge collapse operations, the contracted mesh was converted into 1D curve skeleton. The mesh model was contracted via a discrete Laplacian equation solver, which was introduced to mesh editing area by Sorkine et al. in [6]. In order to achieve mesh contraction while preserving the topological features of the mesh model, Au proposed two types of constraints: contraction constraints and attraction constraints. The contraction constraints were used to shrink the mesh model whereas the attraction constraints were used to preserve the geometric features for some feature points such as the tips of fingers or tips of feet etc.. The contraction constraint was defined by applying Laplacian smoothing operations on mesh vertices, which forced the vertices of the mesh model moving to its curvature flow normal (inward) direction. The attraction constraints were defined via selected positional constraints, which were described as anchors in [68].

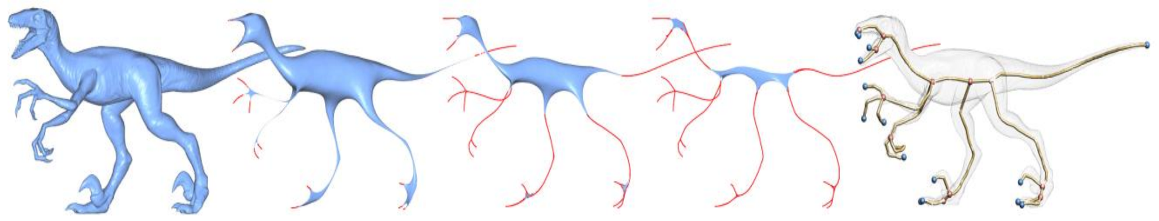

**Figure 2-5 Mesh Contraction Method from Au [17]**

Figure 2-5 is an example from Au's work. This method creates 1D curve skeleton. In addition to producing a curve skeleton, the method produces other valuable information about the model's geometry. For example all the collapsed vertices can be used for segmentation purpose.

### 2.4.2. Interior Point Computation

There are two types of interior point computation strategies: radial basis function based method and MAT based method. Radial basis function based method can be considered as one special case of distance function. In this function a 3D mesh model can be transformed into an implicit surface, which has the following form:

$$f(x) = h \tag{2.4.1}$$

where $x$ is a point in $R^3$ and $h$ is a level value of a designated surface. Usually, $h = 0$ indicates the mesh model surface. It will return a higher value when the input data x gets deeper away from the surface. Combine with the techniques such as gradient descend and active contour model, one can extract the skeleton from implicit surface of a 3D model.

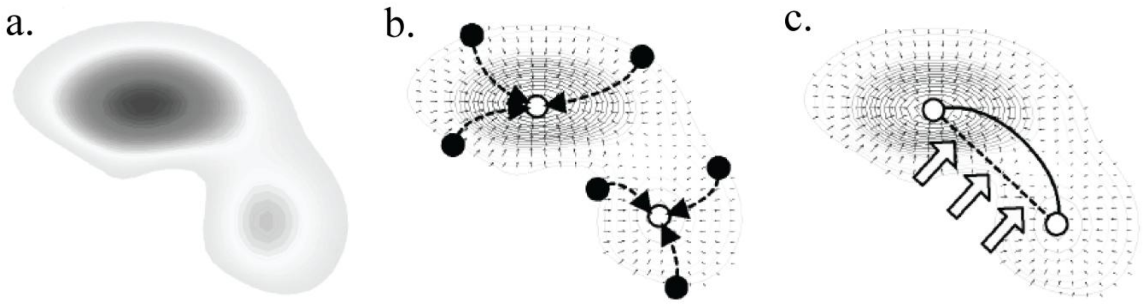

**Figure 2-6 Radio Basis Function based Method from [64]**

The skeleton extraction process using radial basis function can be revealed from the following description: Let $\Omega_o$ be a connected bounded domain in $R^3$. A point $p$ on the surface $S$ can be denoted as $p \to S$. If there is an existing distance function $D_s(x)$ that calculates the distance between point $x$ and surface $S$, a gradient operator $\nabla D_s(x)$ can be applied and for $p \to S$ we can apply the gradient descent $G(p): p^0 = p, p^{n+1} = p^n + \nabla D_s(p^n)$. If

$\nabla D_s(p^n)=0$, $p^{n+1}$ is a local maximum and $G(p)=p^{n+1}$. The local maximum $m_i \in M$ is a representative of the set $p(m_i)=\{p_j \mid G(p_j)=m_i\}$. The skeleton is constructed by linking local maximum pair $m_i$ and $m_j$ if $p(m_i)$ and $p(m_j)$ are adjacent [64].

Figure 2-6 is the demonstration of the radial basis function based skeleton extraction method. It includes three steps: construct implicit surface (as shown in Figure 2-6.a), compute the interior point using gradient descent method (as shown in Figure 2-6.b) and link adjacent interior points (as shown in Figure 2-6.c).

MAT technique is another interior point based skeleton extraction method. It is widely used in image skeletonization process. The medial axis of a mesh model is the set of all points which have more than one closest point on the mesh model's boundary. Originally, it was introduced by Blum [69] as a tool for biological shape recognition. In mathematics the closure of the medial axis is known as the cut locus.

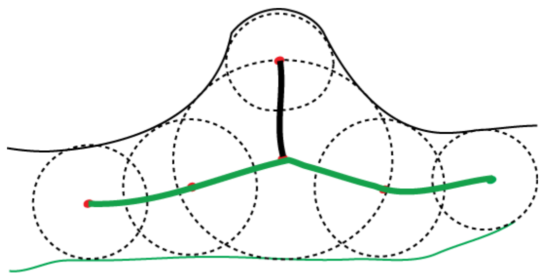

**Figure 2-7 MAT based Method**

The medial axis together with the associated radius function of the maximally inscribed circle is called the MAT. MAT in 3D is the locus of the inscribe spheres inside a model. Each point of a MAT based skeleton is associated with a radius function, which is the radius of the maximal ball around any given point on the skeleton. MAT based skeleton can describe the mesh model surface precisely and is widely used in surface reconstruction based applications.

Figure 2-7 is an example of MAT based skeleton extraction method. The circles represents the radius function for each point on the skeleton, whereas the green line and black line are the skeletons inside a mesh part.

### 2.4.3. Surface Feature Point Extraction based methods

Geometric computation based methods apply a function directly over a mesh model surface to extract and link critical points inside the mesh model to form a skeleton structure. Most of the geometric based approaches study the properties of real valued functions which are computed over triangulated surfaces. And those functions are provided by the application context, such as scientific data analysis [70, 71].

Morse and Reeb graph theories [72, 73] based skeleton extraction methods can preserve the topological properties of the model with the help of some predefined critical feature points (maxima, minima and saddle points). Morse theory can be thought of as a generalization of the classical theory of critical points of smooth functions on Euclidean spaces. Morse theory states that for a generic function defined on a closed compact manifold (e.g. a closed surface) the nature of its critical points determines the topology of the manifold. Reeb graph represents the configuration of critical points and their relationship and provides a way to understand the intrinsic topological structure of a model. Thus it has been used in many applications such as shape matching [74], shape coding[75] and surface description and compression[21, 48].

Reeb graph Definition: Let $r(x)$ be a real-valued function on a compact manifold $M$. The Reeb graph of $r$ is the quotient space of the graph of $r$ in $M$ by equivalence relation "$=$" defined as $(x_1, r(x_1)) = (x_2, r(x_2))$, if $r(x_1) = r(x_2)$ and $x_1$, $x_2$ are in the same connected component of $r^{-1}(r(x))$.

The two pairs $(x_1, r(x_1))$ and $(x_2, r(x_2))$ are represented as the same element in the Reeb graph is the function values are the same and if they belong to the same connected component of the inverse image of $r(x_1)$ or $r(x_2)$. Normally the real value function $r(x)$ can be considered as a height function, thus points which have the same height values are located in the same contour line. By sampling the height values, one can obtain several contour lines for a mesh model. By connecting the center of each contour line, one can obtain a curve skeleton.

#### 2.4.3.1. Mapping Function

Mapping function definition depends on what is expected to be revealed [22]. For example researchers notice that height functions [51, 75] will present critical points over hills and valleys, providing consequently an appropriate topological description for terrain modeling. They use the height function (the z-coordinate), relatively to a given orientation of the mesh model to define their diagram. By defining the critical points from a given model, they can construct the Reeb graph as shown in Figure 2-8.

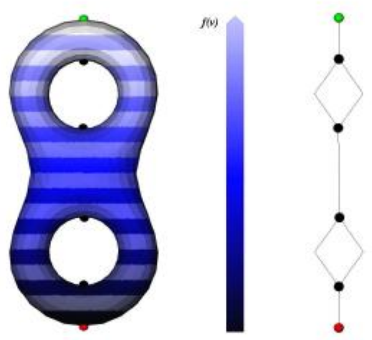

**Figure 2-8 Example of Reeb Graph from [22]**

However, the height function based methods are not rotation invariant, which make the user have to set up a proper coordinate before starting the skeleton extraction. Lazarus and Verroust

[32] introduced a geodesic distance function, which was used to compute the shortest path between vertices, from a source vertex to any other vertex in the mesh model. Compared to the height function based method, the geodesic distance function based method has two advantages. First, the algorithm is invariant to geometrical transformations. Second, it is robust against variations in model pose. Nevertheless, as the Morse theory described, the skeleton structure is defined by the critical points on the mesh model. In order to get an accurate topological structure, the Reeb graph based method requires the user to specify the critical points manually. Otherwise, the noise points, which have similar curvature and normal features within some local regions, may affect the final skeleton structure.

Dey proposed a medial geodesic function to extract the skeleton for a mesh model, which can describe the intrinsic property of the mesh model surface as well as its embedding in 3D space in [76]. The Voronoi diagram was also used to enhance both the computational performance and the robustness. The skeleton from Reeb graph is in 1D structure, which nodes are critical points of a real-value function defined on the mesh model surface, for encoding the topology of a mesh model [77]. The choices of the real-value functions may also include geodesic function [74] and harmonic function [18]. After obtaining a Reeb graph, coordinates information should be determined for all nodes to turn a Reeb graph into a skeleton. In general, geometric methods are sensitive to noise appearing over the mesh model surface. Additional treatment, such as re-meshing, is needed for generating a better skeleton.

### 2.4.3.2. Critical Point Selection

Critical points are mesh vertices, which are located on extremities of prominent components [65]. Mortara [78] proposed to use Gaussian curvature as the metric to define the critical points. In their work, they predefined a threshold to collect vertices whose Gaussian curvature exceeds the threshold. Tierny proposed differential topology tools to extract feature points in [29].

With regards to the critical points extraction, one major problem is to eliminate the “noise” vertices. The “noise” vertices are those vertices that have similar curvature features but are located in local regions. Figure 2-9 is an example of the critical points in a hand model. The green points on the finger tips are the critical points that we need in our Reeb graph construction, whereas the red point is the source point to compute the distance used in a Reeb graph. However, in some local regions, “noise” vertices have similar curvature features. In [29], they proposed two geodesic based mapping functions to remove the “noise” vertices by performing cross analysis on the two geodesic based mapping functions. However, the cross analysis cannot guarantee the rest features vertices have no “noise” vertices, especially for those “noise” vertices which are very close to the feature points that we need.

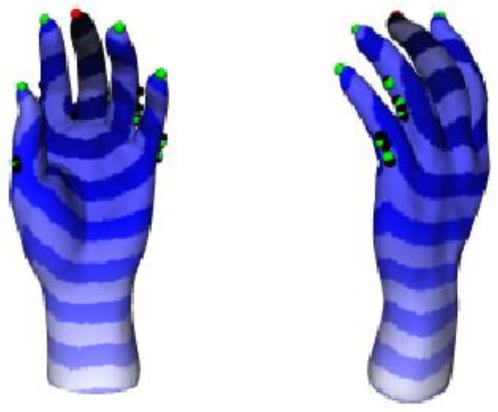

**Figure 2-9 Critical Points in a Reeb Graph from [22]**

### 2.4.4. Summary

By fixing some feature point positions, mesh contraction method can produce 1D curve skeleton. However, such a method requires high computation to extract the final skeleton. Also the skeleton contains no well-positioned joint information, which is important in animation related tasks. Besides, this method destroys the mesh model after the contraction process, which makes it is impossible to use the skeleton directly to the mesh model.

The topological feature of a skeleton from using the radial basis function is insensitive to the surface noise. But it is a time consuming process due to an iterative computation of gradient descent. Also this type of skeleton contains no joint information either. The MAT skeleton always lies in the center of mesh model. However, the skeleton result is too sensitive to the surface details, which makes the MAT based skeleton contains too many undesired branches (see the dark line in Figure 2-7). To deduce a reasonable skeleton structure, the users have to remove the noisy branches by using some pruning methods.

Reeb graph (or LSD) based method extracts a skeleton from some critical feature points, such as the end of an articulated branches. However, such a method suffers from the “noise” data problem, which means the skeleton structure depends on the quality of feature point selection.

## 2.5. Segmentation based Skeleton Extraction Methods

The skeleton from some automatic methods [17, 25] is insensitive to the surface noise. However, the skeleton generated from those automatic methods has no joint specification. The joint specification of a given skeleton describes the topological structure from semantic point of view. Those joints can be used in animation purpose, i.e. the applications in Inverse Kinematic solvers, which make use of joint to compute different rotation or transformation matrix for bones. Some researchers notice that segmentation results can be used to construct skeletons. For example Katz and Tal proposed a control skeleton extraction algorithm by using their segmentation method in [66]. The boundary between each pair of segmentation parts can be considered as the joint positions. This is the initial concept of segmentation based skeleton methods.

### 2.5.1. Introduction

Mesh segmentation has become an important research problem in computer graphics. Many applications have the need to segment a 3D model. For example in modeling [79], compression [80], simplification [81], 3D shape retrieval [67], collision detection [82], texture mapping [83, 84], geometry-image creation [85] and skeleton extraction[86].

There are two main types of methods: surface-type and part-type methods. Surface-type methods use geometric properties, such as planarity or curvature, to create surface patches. In general, surface patches are topologically equivalent to a disk and do not necessarily possess any semantic meaning. Our work adopts mesh segmentation to construct bones and joints for a skeleton structure, which relies on the semantic meaning of each mesh parts. Hence, surface-type methods do not fulfill our need. In this chapter, we only discuss the related research work in part-type based segmentation methods.

More generally, the segmentation can be interpreted either in a purely geometric sense, where some distance measures (i.e. Euclidean distance or geodesic distance ) are used as segmentation metric, or semantic-oriented manner. For the first type, a mesh is segmented into parts with respect to some uniform features, like curvature or distance to a fitting plane. On the other hand, semantic-oriented methods focus on the identification of meaningful parts which describes the local feature of the mesh model.

Semantic-oriented methods are rooted in the study of human perception for producing semantically meaningful mesh parts. Most of the semantic-orientated methods are based on the minima rule (see description in section 1.4) concept, which makes use of curvature variation to distinguish different parts of a mesh model.

To segment a mesh model into meaningful pieces, there are two import issues to be solved:

- How to define the clustering criteria to generate meaningful parts?
- How to define the number of meaningful parts?

The first question reveals the requirement for the segmentation criterion which is able to produce geometrical meaningful result such as the meaningful boundary definition. The second question indicates the requirement to define the semantically meaningful parts from the segmentation process. We analyzed the most widely used techniques in segmentation area with their strengths and weaknesses. And then based on our requirements, we propose a new solution to segment a mesh model into meaningful parts. We will discuss the details of our solution in Chapter 5.

### 2.5.2. Geometric Distance Based Methods

#### 2.5.2.1. Fitting Primitives based Segmentation

Attene et al. proposed a clustering scheme in [67] which made use of different primitives as metric to decompose a mesh model into parts. The algorithm was a variation of the hierarchical face clustering (HFC) method which was fully described in [87]. The HFC algorithm was used to cluster the polygonal surface in a hierarchical structure. By introducing different sorts of predefined shape primitives (i.e. fitting plane, fitting sphere, fitting cylinder), Attene's algorithm divided the mesh model into several parts.

The general procedures of HFC algorithm are described as follows:

1. Constructing the dual graph of a mesh model. In the dual graph, each node will correspond to a face cluster, which is a connected set of faces that have been clustered

together. For the initial dual graph, each cluster contains a single face of the mesh model. And these clusters will form the leaves of the hierarchy.

2. Performing an edge contraction in the dual graph (see Figure 2-10) which merges two dual nodes into one. Repeat this step until only one node left, which can be viewed as the root the hierarchical structure. In order to construct a complete hierarchy, each dual edge is assigned with a "cost" of contraction. And the system will iteratively contract the dual edges of least cost.

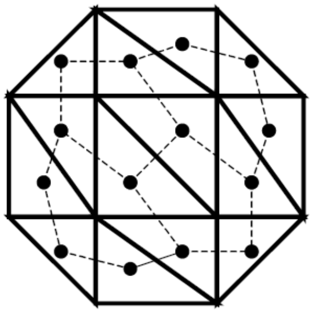

**Figure 2-10 Dual Graph**

There are many different HFC algorithms which vary in different edge contraction metrics. Garland et al. adopted planarity as the primary criterion in [87]. They proposed three distance-metrics, which are Plane Fitting metric, Orientation Bias metric and Compact Shape Bias metric, to define the criteria in the dual edge contraction process.

First, they made use of Least Squares Fitting[3] to find out the best fit plane for a given set of cluster. Assume the best fit plane for a cluster set is represented as $(n, d)$, where $n$ indicates the unit normal of the plane and $d$ is a scalar value. The fit error metric is defined as:

[3] Least Squares Fitting is a mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets of the points from the curve.

$$E_{fit} = \frac{1}{k}\sum_{i=1}^{k}(n^T v_i + d) \tag{2.5.1}$$

where $n^T v_i + d$ indicates the distance from point $v_i$ to the plane.

In order to further improve the clustering accuracy, Garland et al. proposed an Orientation Bias metric to distinguish local surface area, which may have dramatic normal change within a local region. The Orientation Bias metric was designed as:

$$E_{dir} = \frac{1}{w}\sum_{i} w_i (1 - n^T n_i)^2 \tag{2.5.2}$$

where $w_i$ was the area of faces $f_i$, and $w = \sum_i w_i$ was the total area of the face cluster, $n_i$ indicates the unit normal of each face. This metric was used to measure the average deviation of the plane normal $n$ from the surface normal $n_i$.

They also proposed a Compact Shape Bias metric to reduce the irregularity of the clustered shape, which aimed to improve the cluster to be as nearly circular as possible. Given a cluster with area $w$ and perimeter $\rho$, the irregularity $\gamma$ of the cluster was defined as a ratio of its squared perimeter $\rho^2$ to its area $w$:

$$\gamma = \frac{\rho^2}{4\pi w} \tag{2.5.3}$$

The irregularity of a circle is $\gamma = 1$; larger value of $\gamma$ correspond to more irregular regions. The definition of irregularity is widely used in image processing [79]. Given two set of clusters $c_1$ and $c_2$ to be merged together, the Compact Shape Bias metric is defined as:

$$E_{shape} = 1 - \frac{\max(\gamma_1, \gamma_2)}{\gamma} \tag{2.5.4}$$

where $\gamma$ is the irregularity of the new cluster which is merged from $c_1$ and $c_2$. $\gamma_1$ and $\gamma_2$ are the two irregularities of $c_1$ and $c_2$ respectively.

Finally the distance metric used in HFC algorithm is defined as:

$$E = E_{fit} + \alpha_1 E_{dir} + \alpha_2 E_{shape} \tag{2.5.5}$$

where $\alpha_1$ and $\alpha_2$ are constants which are used to evaluate the importance of different aspects. These two values must be chosen by the user.

In general the metrics used in different HFC algorithms can only produce patch-based segmentation results, rather than part-type segmentation results. Surface patch has no volume information which cannot be used to define meaningful parts. Attene et al. further introduced several fitting primitives (fitting sphere, fitting cylinders) as references in part-type segmentation. However, there are some limitations in part-type segmentation using the predefined fitting primitives. Firstly, the fitting primitive based method cannot determine the exact number of meaningful parts. There is no such detection process in HFC related work. Second, although some new fitting primitives were introduced, such as spheres or cylinders, the segmentation boundary is normally irregular which does not fit in the shape structure in the local regions.

#### 2.5.2.2. Fuzzy Clustering

Katz et al. proposed the fuzzy clustering algorithm in [66], which produced hierarchical tree structure segmentation result from coarse to fine. Each node in the hierarchy tree was associated with one part of the mesh, which was constructed by several smaller patches. And the root node was associated with the whole mesh model.

A key idea of the algorithm is to find out meaningful components while keeping the boundaries between the components fuzzy. Then the algorithm focuses on the small fuzzy areas to determine the exact boundaries which go along the features of the mesh model. In the recognizing of fuzzy components process, the condition that every face should belong to exactly one patch is relaxed, and fuzzy membership is allowed. In essence, this is equivalent to assigning each face a probability of belonging to each patch.

The fuzzy clustering algorithm contains four stages:

1. Constructing a dual graph (see Figure 2-10) from the existing triangle faces and computing the distance for all pair of faces in the mesh.
2. Finding out the initial representative faces which define the number of parts to be segmented. And then computing the probability for each face that belongs to each part.
3. Computing a fuzzy decomposition by refining the probability values using an iterative clustering scheme.
4. Constructing the exact boundaries between each part.

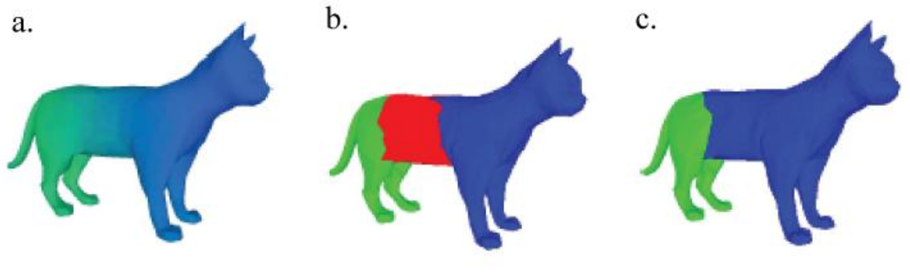

**Figure 2-11 Fuzzy Clustering Method Example from [66]**

Figure 2-11 is an example of fuzzy clustering algorithm. Figure 2-11.a displays the probability distribution over a cat model. Figure 2-11.b is the fuzzy decomposition result, where the red region is uncertainty region. Figure 2-11.c is the final result after region merging and boundary extraction.

Katz made use of geodesic distance as metric to compute the probability for each face. The geodesic distance between any two adjacent faces $Dist(f_i, f_j)$ was defined as the geodesic distance of the centers of mass in two faces. Regards to the dual graph of the mesh model, which was constructed in the first step of Katz's algorithm, the geodesic distance between two adjacent faces can be considered as the arc which connected the dual vertices (two adjacent faces in the mesh model) in the dual graph.

Assume there are two representative faces: $REP_A$ and $REP_B$, which define the number of patches. By defining the initial representative faces in the second step of the algorithm, one can compute two geodesic distances $a_{f_i} = Dist(f_i, REP_A)$ and $b_{f_i} = Dist(f_i, REP_B)$, for each face $f_i$. The probability of $f_i \in \mathrm{REP_A}$ is defined as:

$$P_A(f_i) = \frac{a_{f_i}}{a_{f_i} + b_{f_i}} = \frac{Dist(f_i, REP_A)}{Dist(f_i, REP_A) + Dist(f_i, REP_B)} \tag{2.5.6}$$

For two patches only based clustering, the probability that face $f_i \in REP_B$ is $1 - P_A(f_i)$.

It is difficult to find out how many parts that a mesh model should be decomposed to. Katz et al. made use of derivative variation to detect the number of parts for mesh decomposition. The method was proposed as follows:

1. Step 1: For each face $f_i$ in the mesh, assign a distance value $d_i = \sum Dist(f_j, f_i)$, where $f_j$ is the rest face candidate in the mesh model.

2. Step 2: At the beginning, take the face $f_i$, which has the minimum distance value, as the representative face. This is done in order to represent the main "body" of the mesh model.

3. After step 2, the algorithm runs iteratively to add new faces as the new representatives. But only the face which has the largest distance value is added.

A new function $G(f_k) = \min_{i<k}(Dist(REP_k, REP_i))$ was proposed to detect the minimum distance between the new added representative and the previous located representatives. Obviously, this function decreases as more representative faces are added. By monitoring the derivative variation of the function $G(f_k)$, Katz observed that the number k is the proper position when the correspondent $G(f_k)$ has the maximum derivative value.

The fuzzy clustering algorithm tries to decompose a mesh model hierarchically. However, this method only divides the mesh model based on the geometric information. It cannot define the semantically meaningful parts by using this mechanism. Take the human leg segmentation For example considering the geodesic distance only cannot guarantee the segmentation boundary lies in the foot ankle region. The time complexity for the shortest path between any two faces $f_i$ and $f_j$ computation is $O(n^3)$. The preprocessing takes too much time which makes the algorithm is inefficient for large models.

### 2.5.3. Semantic-Oriented Methods

To get meaningful pieces segmentation result, the concept of minima rule is used as the criteria in semantic-oriented methods [30, 88, 89]. The minima rule states that human vision tends to define areas of minimum negative curvatures, i.e., concave shape areas, as interfaces separating between mesh model parts [31]. Based on this concept, surface curvature is used as one property of some distance metric in clustering based segmentation.

Many research works have been done to produce meaningful segmentation result, such as region growing methods [70, 88, 89], iterative clustering method [90], spectral clustering [71] and feature point-based clustering [65, 72]. Semantic-oriented methods take special care on the variation of the surface curvature change.

Probability metric was widely used in many applications. Lai proposed a clustering scheme [89] which was extended from image processing [73], in his random walks algorithm. For a given face $f_i$, and its neighbors $f_{i,k}$ where $k = [1, 2, 3]$, a function:

$$d_1(f_i, f_{i,k}) = \eta(1 - \cos(diheddral(f_i, f_{i,k}))) \tag{2.5.7}$$

is used to measure the curvature between two triangles. The overall difference between $f_i$ and $f_{i,k}$ is computed as:

$$d(f_i, f_{i,k}) = \frac{d_1(f_i, f_{i,k})}{\bar{d}} \tag{2.5.8}$$

where $\bar{d}$ is the average edge length of face $f_i$.

Given this difference function on hand, the probability distribution is computed as:

$$p_{i,k} = |e_{i,k}| * \exp\left\{-\frac{d(f_i, f_{i,k})}{\sigma}\right\} \tag{2.5.9}$$

where $|e_{i,k}|$ is edge length of the correspondent shared edge between face $f_i$ and $f_{i,k}$. And $\sigma$ is a threshold to control the variation of probability via the difference function. The difference function $d(f_i, f_{i,k})$ can be considered as the implementation of minima rule.

Figure 2-12 is a segmentation result example from [89]. The black dots in each part are the source points which are used to compute the probability for each part. This picture shows that the probability metric is useful in part-type segmentation. However, regards to the implementation of minima rule, the result still need further improvement. For example the boundaries among different legs are not properly defined (see Figure 2-12). Because the segmentation result does not follow the curvature property of the mesh model, it is very hard to use this segmentation result to generate skeleton due to the inaccuracy problem. Also in Lai's work, the convex and concave distance is manually assigned with different weights, which is not a natural way to define the difference between two adjacent faces. Besides, according to the segmentation results comparison, we notice that the segmentation result is sensitive to the location selection of seed points.

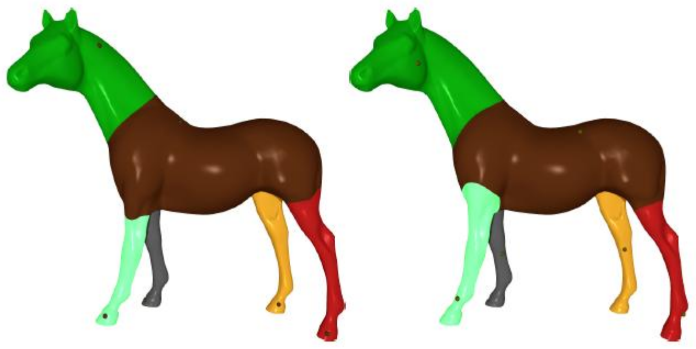

**Figure 2-12 Segmentation Result using Random Walk method [89]**

Pottmann introduced the isophotic metric in [1]. The isophotic metric was widely used in surface-type segmentation. This is because the length of a surface curve is not just dependent on the curve itself, but also on the variation of the surface normal along it. Pottmann considered the field of unit normal vectors $n(x)$ attached to the surface points $x \in S$ as a vector-valued image defined on the surface. One can map each surface point $x$ to a point $x_f = (x, wn)$ in $R^6$ where $w$ denotes a non-negative weight, whose magnitude regulates the amount of feature sensitivity and the scale on which one wants to respect features [88]. Thus $S$ is associated with a 2-dimensional surface $S_f \subset R^6$.

According to Pottmann's definition, the isophotic metric distance between two points $p$ and $q$ on a surface is depended on the path $\Gamma$, which connects these two points on the surface. And the isophotic metric is defined as the arc length differential as:

$$d_\Gamma(p,q) = \int_\Gamma ds + w\int_{\Gamma^*} ds^* \tag{2.5.10}$$

where $ds$ is the arc element of $\Gamma$, $ds^*$ is the arc element of $\Gamma^*$,which is the Gaussian image of $\Gamma$, and $w > 0$ is the weight of the isophotic components.

Ji [88] further extended the isophotic metric as:

$$d_\Gamma(p,q) = \int_\Gamma ds + w\int_{\Gamma^*} ds^* + w^*\int_\Gamma f(k_D)ds \tag{2.5.11}$$

where $k_D$ is the normal-section curvature in the direction tangent to the path $\Gamma$, $f$ is a function of curvature and $w^*$ is the weight of the curvature metric. The curvature component in Ji's metric can be considered as an implementation of minima rule over the surface domain.

Based on the sketched line guidance, Ji [88] made use of this metric to segment the mesh model into two parts. He defined a segmentation algorithm to divide a mesh model into two parts. One part was considered as the background, whereas another was considered as foreground. More precisely, it is a clustering process, which measures the surface curvature changes to decide whether a surface triangle is selected into foreground group or background group. Figure 2-13 is an example of the segmentation result. As shown in Figure 2-13.a, Ji sketched two lines: a green line, which is a foreground reference, and a red line, which is a background reference. The finger, which is painted in purple, is the isolated foreground part. Also Ji applied a snake algorithm to smooth the boundary, which is shown in Figure 2-13.b. The yellow curve is a reference, which is used to indicate the region where the snake algorithm will be used. Figure 2-13.c is the final segmentation result of a hand model.

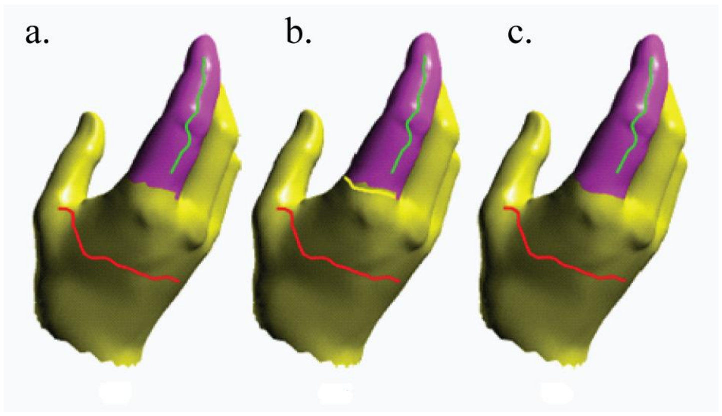

**Figure 2-13 Segmentation Result using Isophotic Metric [88]**

### 2.5.4. Summary

Segmentation based methods are used to decompose a mesh model into meaningful parts. Geometric distance based segmentation methods, such as [67] or [66], do not consider the surface curvature features, which may result in undesired segmentation results. Later, the minima rule is introduced in surface-type segmentation [30]. But those segmentation methods are too sensitive to the surface curvatures, which may result in over-segmentation problem too. To obtain meaningful parts, Lai et al. proposed a part-type segmentation method in [89]. By considering the surface curvature features, such as the dihedral angle between surface triangles, Lai et al. obtained meaningful segmentation parts. We notice that the segmentation result is sensitive to the selection of seeds.  Ji et al. also proposed a surface segmentation method, which utilize the minima rule to decompose surface mesh. In order to obtain meaningful segmentation parts, which are used to construct skeleton, we need to come up with a segmentation method which can avoid over-segmentation issue and also be able to generate smooth segmentation boundaries.

## 2.6. Introduction to animation related work

### 2.6.1. Linear Blend Skinning Method

Traditional hand-drawn animation requires drawing each frame of an animation explicitly. Computers may be used to reduce the labor work by providing a degree of automation. Animating characters, such as human or animal models, is a particularly demanding area. This is because that most of the existing movies or cartoons require lots of human or animal motions to describe a story. A convenient way of specifying the motion of characters is through the movement of an internal articulated skeleton from which the movement of surrounding polygon mesh may then be deduced. With regards to the animation framework, there are two major technologies:

1. Linear skinning techniques,
2. Mesh deformation techniques

There are many different linear skinning techniques, such as Skeletal-Subspace Deformation (SSD) [13], Animation Space [15] and Multi-Weight Enveloping (MWE) [91]. Jacka et al. [14] proposed a survey to discuss the difference among those methods. In our implementation, we just adopted the original Linear Blend Skinning (LBS) method to perform skeleton-driven animation.

LBS technique is the simplest and most widely used method for calculating animations in real-time. It was not originally published but is described in papers [13, 15, 92, 93] that look to extend and improve it.  The LBS method defines the final position of a point in a mesh model as a weighted linear combination of the initial state of the point projected into several moving coordinates frames, one frame for each bone[94]. The position of a point $p'$ after deformation can be written as:

$$p' = \sum_{k=1}^{n} w_k p M_k \tag{2.6.1}$$

where $p$ is the initial position of a point, $M_k$ is a transformation matrix that transforms bone $k$ from its initial position to current new position, $w_k$ is the weight of point $p$ relative to bone $k$, and $n$ is the number of bones which are associated with point $p$. To obtain reliable deformation result, careful choice of weights $w_k$ is needed; otherwise, self-intersections might occur near joints located regions.

### 2.6.2. Laplacian Mesh Deformation

The LBS technique suffers from the self-intersection problem. We adopted a differential coordinates based deformation system to ease the self-intersection problem. The differential coordinates system is an approach for detail-preserving deformation. The differential coordinates are defined by a linear transformation of the vertices in the mesh. This allows the reconstruction of the edited surface by solving a linear system that satisfies the reconstruction of the local details in least square sense. There are many approaches based on differential coordinates system. One approach is by using Laplacian coordinates [6, 7]; one is based on Poisson equations [8]. Also harmonic field analysis been proposed in [95] for surface deformation. Here is a general definition of differential setting:

$$\nabla^2 u = f \tag{2.6.1}$$

If $f \equiv 0$, the equation is Laplace equation; If $f \neq 0$, the equation above is called Poisson equation.

Differential surface deformation was inspired by gradient domain image manipulation. It has been noticed that the gradients of image intensity function contains important visual information that humans are sensitive to; many image techniques exploit this fact by applying certain manipulations to the input image gradients $g = \nabla I$, and then reconstruct the resulting image by a global optimization process that looks for an image $I'$ whose gradients are as close as possible to the modified gradients $g'$:

$$I' = \arg\min_{I'} \int_{\Omega} \left\| \nabla I' - g' \right\|^2 dxdy \tag{2.6.2}$$

where $\Omega$ denotes the domain of the image manipulation.

The simplest form of differential coordinates is the Laplacian coordinates. The powerful properties of local frames have been exploited in recent years. Taubin [96] derives a discrete mesh fairing operator that is applied to model smooth surfaces. Karni and Gotsman [76] take advantage of this extension of spectral theory to arbitrary 3D mesh structures for progressive and compressed geometry coding. Lipman [7] proposed a Laplacian coordinates representation of mesh, which lead to efficient, interactive and intuitive shape modeling include local control and detail preservation.

The basic definition of the mesh Laplacian operator is described as follows: Let $M=(V,E,F)$ be a given triangle mesh with $n$ vertices, where $V$ denotes the set of vertices, $E$ denotes the set of edges and $F$ denotes the set of faces. Each vertex $i \in M$ is conventionally represented using absolute Cartesian coordinates, denoted by $V_i=(x_i, y_i, z_i)$.

First, the differential or $\delta - coordinates$ of $v_i$ is defined to be the difference between the absolute coordinates of $v_i$ and the center of mass of its immediate neighbors in the mesh.

$$\delta_i = (\delta_i^x, \delta_i^y, \delta_i^z) = v_i - \frac{1}{d_i} \sum_{j \in N(i)} v_i \tag{2.6.3}$$

where $N_i = \{ j \mid (i,j) \in E \}$ and $d_i = \left| N(i) \right|$ is the number of immediate neighbors of $i$ (the degree of valence of $i$). See Figure 2-14 for illustration.

In Figure 2-14, the vector of the differential coordinates at a vertex approximates the local shape characteristics of the surface: the normal direction and the mean curvature. $\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (v_i - v_j)$ is a discretization of the curvilinear integral: $\frac{1}{|\gamma|} \int_{v \in \gamma} (v_i - v) dl(v)$ where $\gamma$ is a closed simple surface curvature around $v$ (as shown in the circled region in the right side of Figure 2-14) . $v_i$ is the red point as shown in the left side of Figure 2-14, and $v_j$ are vertex which share one edge with vertex $v_i$.

According to Sorkine's explanation in [68], it is known from differential geometry that

$$\lim_{|\gamma| \to 0} \frac{1}{|\gamma|} \int_{v \in \gamma} (v_i - v) dl(v) = -H(v_i) n_i \tag{2.6.4}$$

where $H(v_i)$ is the mean curvature at $v_i$ and $n_i$ is the surface normal. Therefore, the direction of the differential coordinate vector (the left yellow arrow in Figure 2-14) approximates the local normal direction (the right yellow arrow in Figure 2-14). And the magnitude approximates a quality proportional to the local mean curvature. Thus, the differential coordinates encapsulate the local surface shape.

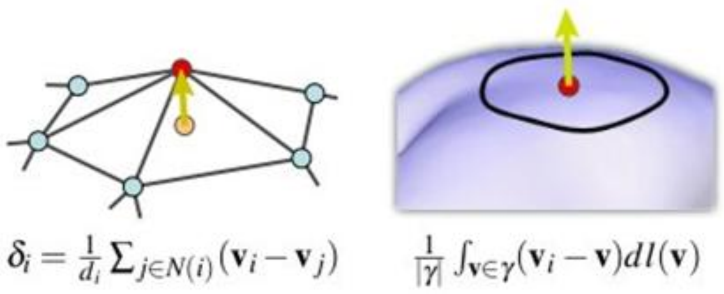
$\delta_i = \frac{1}{d_i}\sum_{j\in N(i)}(\mathbf{v}_i - \mathbf{v}_j)$
$\frac{1}{|\gamma|}\int_{\mathbf{v}\in\gamma}(\mathbf{v}_i - \mathbf{v})dl(\mathbf{v})$

**Figure 2-14 Differential Coordinates from [6]**

Given the differential coordinates $\delta^x, \delta^y, \delta^z$ of the mesh, the absolute coordinates of the mesh geometry can be reconstructed by solving the system $Mx = \delta^x$ (the same goes for y and z coordinates). The procedure of using Laplacian operator to achieve mesh deformation can be classified into the following steps:

1. The Laplacian coordinates are constructed through Eq-(2.6.3). According to the mesh connectivity, adjacent matrix $A$ and diagonal matrix $D$ are constructed first, where

$$A_{ij} = \begin{cases} 1, & \text{(i,j)} \in \text{E} \\ 0, & \text{otherwise} \end{cases} \tag{2.6.5}$$

And $D_{ii}$ is the valence of each vertex $i$ in the mesh. We get relative coordinates from the following formula $L = D^{-1} - A$ or $Ls = D - A$.

2. When we get $L$ or $Ls$ matrix, we compute the $\delta - coordinates$ for x, y, z respectively, through the following formula:

$$Lx = \delta^{(x)} \tag{2.6.6}$$

3. Since we get the $\delta - coordinates$, we add constrained vertices positions into $L$ matrix, and deformed value for x, y, z to the right side of $\delta$ vector. The computation of x, y, and z is carried on through the following formula:

$$\left\{ \frac{L}{\omega I_{m \times m} \mid 0} \right\} = \left\{ \frac{\delta^{(x)}}{\omega C_{1:m}} \right\} \tag{2.6.7}$$

The weight $\omega$ is used to tweak the importance of the positional constraints. $C_{1:m}$ indicates the handle vertices value in corresponding direction (i.e. x, y or z).

### 2.6.3. Summary

In this section, we introduce two deformation methods that we will adopt in our skeleton extraction system. The LBS based method is fast and efficient, which is widely used in real-time animation applications. However, the deformation results may have self-intersection problem. To handle this issue, we further introduce a Laplacian mesh deformation framework to perform the deformation tasks. The Laplacian mesh deformation method preserves the surface details in the deformation process. However, such a method cannot preserve volume information during the deformation process. Thus to maintain rigid deformation result, we make use of LBS result as the positional constraints to get our desire deformation results.

# Chapter 3

## 3. Research Goals, Challenges and Proposed Solutions

### 3.1. Goals and Challenges

Our target is to design an interactive skeleton extraction system. Within this system, a user can draw freely on a mesh model, and the system will generate the skeleton automatically according to his/her sketched lines. After the skeleton extraction, we also allow the user to make use of sketching as manipulators to perform skeleton-driven animation task. The key idea is to let non-professional users produce animation easily.

To achieve those goals that we mentioned above, there are several challenges that we need to overcome. First, we need to extract user's intention from their drawings. As we allow a user to draw freely on the mesh model, it is not a good idea to make use of those sketched lines directly. The reason is that a user's drawing may contain duplicate sketched lines or the sketched lines have unclear topology specification, such as a sketched line may contain great curvature changes. If a sketched line contains some large curvature changes, we need to divide this line into several line segments from those positions where large curvature changes occurred.  In general, we need an algorithm to analyze the user's input and clarify the user's intention. With regards to the user's intention extraction, we have to solve the following problems:

1. Sketched line projection issue: To construct a sketched line in 3D, we need to detect the intersection regions between an orthogonal projection line and the mesh parts. However the intersection regions are sensitive to the selection of view point. Inappropriate view

point may result in wrong intersection region selections. We need a method to guarantee the projection quality.

2. Similarity check for duplicated drawings: If the new sketched line is similar to some existing sketched lines, we cluster the new sketched line with the existing similar lines together. If we do not merge similar sketched lines together, and make use of the triangles as the independent seed triangles, we cannot obtain correct segmentation results.
3. Topology check for sketched lines: If the sketched lines contain large curvature changes, we need to break apart the sketched lines. Large curvature changes indicate the topological changes for a sketched line. If we do not divide the sketched lines into parts, we cannot get the right segmentation result. Thus our skeleton may lack joint specifications in the regions where this sketched line is located.

Second, we need to find out a solution to create a skeleton from both a user's drawing and the mesh model itself. The skeleton should contain meaningful topological structure and well-positioned joint position specification. So, in our skeleton extraction solution, we need to find a way to calculate the joint positions. After the skeleton extraction, we also need to verify the correctness of the skeleton. We need to find out a metric to measure the correctness of the skeleton with the help of its correspondent mesh model.

In our method, we have proposed two metrics to measure the quality of a skeleton: the smoothness of all bones, which reflects the curvature quality of the skeleton; the quality of skin mapping, which is used to check the quality of a skeleton in the animation aspects.

## 3.2. Proposed Solutions

This thesis divides the whole problem into three major categories:

1. Sketching Analysis, which is used to analyze the user's input and find out the user's intension from his drawing.
2. Skeleton Extraction, which is used to create the skeleton.
3. Skeleton Validation, which is used to evaluate our result.

We allow the user to sketch freely on the mesh model. Thus the user's drawings are inevitable to contain mistakes, such as some lines may contain large curvature change, which should be divided into several shorter lines. We need a system to reduce those mistakes and collect the user's real intention on the topological definition of the skeleton. We proposed to handle those issues using the Douglas-Peuck (DP) algorithm. We use this algorithm to detect the curvature change and break sketched lines into several parts.

In our system, we assume that a user will properly sketch lines by roughly following the mesh model structure. The user is expected to sketch independently on different regions of a mesh model for creating separate bones. For each sketched stroke, we project it onto the mesh model so that it locates inside the correspondent mesh region from the current viewer perspective. We call this projected stroke a sketched bone.

Skeleton extraction is much more complicated than the sketching analysis part. We need to create a skeleton based on the guidance of the user's drawing. The skeleton should contain well-positioned joints specification, which defines the topological bone connection relationship. Furthermore, we want to specify the geometric position of the joints clearly via a segmentation preprocessing. To extract the topology feature of a mesh model, it is worth to divide the mesh model into several meaningful pieces. Also we notice that the minima rule has already been applied in several surface-type segmentation methods [30, 88]. The minima rule states that human divides 2D shapes into parts at negative minima of the principal curvatures; it divides 3D shapes into parts at negative minima of the principal curvatures. The boundaries from minima rule based segmentation methods locate at those regions which have negative principal curvatures. Those boundaries divide a mesh model into meaningful pieces, and we can use the center of those boundaries as the joints positions of our skeleton model.

A sketched bone provides important information: the triangles, which come from the sketched bone associated surface, and the topology feature of the sketched bones. We divide the sketched bones, which contain large curvature changes, into several line segments; we merge the sketched bones, which have similar topology features, to be one single line. We will give full description in Chapter 4. After these processing, the rest of the sketched bones are called reference bones. The number of reference bones defines the number of parts to be decomposed. The triangles from each reference bone define the initial seeds for mesh segmentation purpose. Also the reference bone is used to guide the clustering process.

To construct a skeleton for a mesh model, we need to solve two problems:

1. Mesh segmentation
2. Bone Extraction

We propose to segment a mesh model into meaningful parts. Then we extract bones from each part and combine those bones to construct the final skeleton for a given mesh model. To segment the mesh model into meaningful parts, we adopt a region growing based algorithm, in which the sketched bones and its associated surface information are considered as input in our algorithm. We extract the bone structure for each part by using LSD method. Finally we merge the bones from each part and smooth the skeleton to finalize our result.

We use Figure 3-1 to describe the overview of our proposed solutions. The whole skeleton extraction process contains 6 steps which are described as follows:

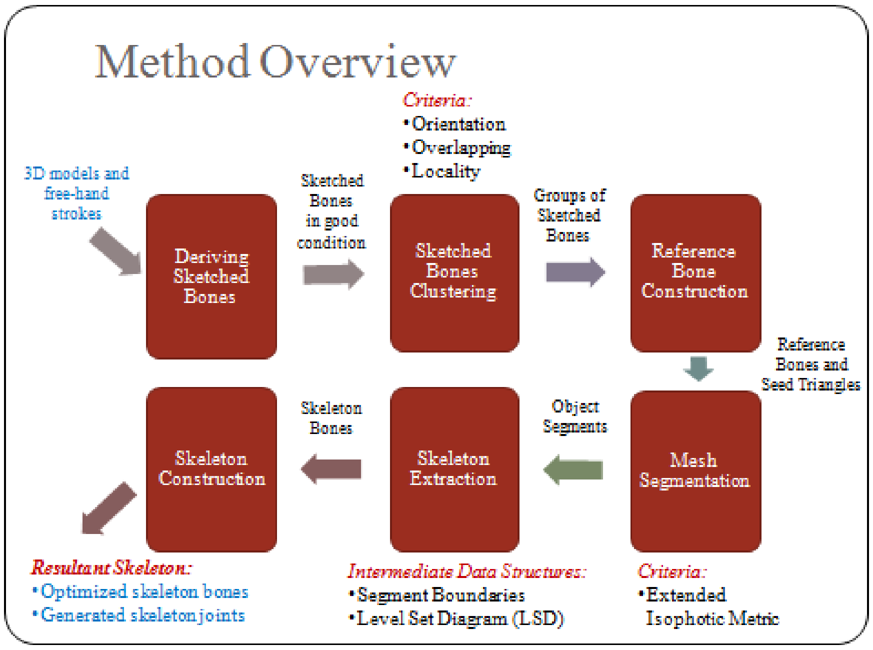

**Figure 3-1 Method Overview**

1. Deriving sketched bones – We allow a user to sketch freely on a mesh model. When a user sketches on the screen, we are able to select the projected[4] mesh triangles from the surface by using the function provided by OpenGL.

2. Sketched bones clustering – The sketched bones may contain repeat drawing, or need to be divided into several straight bones. We need a process to analyze the sketched bones. Based on the orientation, locality and overlapping properties among sketched bones, we need to cluster them into different groups.

3. Reference bone construction – We propose to divide a mesh model into several meaningful parts, the clustered bones need to be merged into one reference bone. During this process, we need to keep two important factors: the topology feature of

[4] The selection may involve redundant surface data. For example, the project function may pick more than two set of data during the penetration based selection. We only need the first one or two set of surface triangles. This can be done by filtering the triangles with some depth value, in which the depth value are recorded from the screen to the selected triangles.

reference bone and the seed triangles which are associated with all the clustered sketched bones.

4. Mesh segmentation – We take the reference bones and seed triangles as the input and perform a multiple-region growing segmentation algorithm to segment a mesh model into parts. Each region is defined by a reference bone and its associated seed triangles.
5. Skeleton extraction – We make use of LSD method to extract bones for each part. Each segmentation part contains a set of segmentation boundaries. The center of those boundaries is the ideal position for a joint. For each boundary, we use the center of a boundary as the starting point to compute a set of level set diagrams. By linking the center of those level set diagrams together, we construct a bone from one boundary.
6. Skeleton construction – We connect all the bones together if any two bones share the same boundary. And the joints are those which are located in the boundary regions. The bone that we extracted from using LSD method may contain distortions due to the surface details variation. We also need to smooth the extracted bones. If there are many boundaries in a segmentation part, we need to merge bones together to construct the final skeleton structure within one segmentation part.

We make use of the following picture to describe the process of our system:

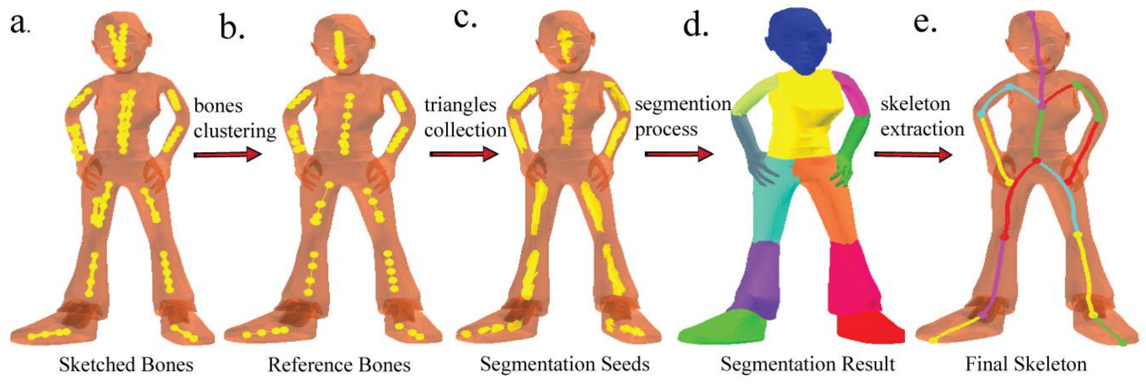

**Figure 3-2 Skeleton Generation Process**

When a user sketches several stokes on the mesh model, our system will first cluster the sketched bones and merge each group to construct reference bones. Each reference bone is marked with one unique index number. From each reference bone, we collect a set of triangles as segmentation seeds, those triangles are indexed with different group number which is the same as their correspondent reference bone number. We perform region growing algorithm from those seed triangles. After the segmentation process, we extract bones from each segmentation part and connect them together to create the final skeleton. During the skeleton extraction process, we also include bone smoothing operations to eliminate the distortions from the LSD algorithm.

## 3.3. Evaluation Method

Cornea et al. described a set of desirable curve-skeleton properties in [50]. They have considered the following properties: invariant under isometric transformations, reconstruction, thinness, centeredness, reliability, smoothness, component-wise differentiation, robustness, efficient to compute and hierarchic. Most of those properties are analyzed based on discrete 3D model. And these properties are summarized from a number of different applications of curve-skeletons in computer graphics and visualization.

We try to evaluate our method by doing some quality and quantity analysis on our skeleton results. As we mentioned, Cornea et al. have summarized 11 types of properties in the evaluation of discrete model based curve skeleton. Our skeleton model is constructed from segmentation result which has the well-positioned joint specifications. We want to evaluate the properties which are highly related to animations. We expect to evaluate the correctness of a skeleton from the following aspects: Smoothness of bones and Skin mapping

**Smoothness**: The smoothness of a bone will affect the skeleton-driven animation result. For example an unsmoothed bone will result in large distortion in a bending process.

**Skin mapping**: this property reflects the relationship between bones and mesh model surface information. Irregular skinning association will result in bad animation result because of the undesired distortion on the mesh model. We build up a mapping relationship between surface data (triangles or vertices) and the joints inside a bone by clustering vertices or triangles to their nearest joints.

There are some properties that we can analyze, such as the centeredness. However, such properties are hard to represent in mathematical manner. We can still discuss those features of our skeleton. But more importantly, we are interested in the quality analysis of the animation relation aspects. Also there are some other properties that we do not consider in our skeleton evaluation criteria. This is because that some of the criterions are defined for special purposes, such as the criteria for reconstruction purpose; some of the criterions are defined to method dependent such as the criteria to check the thinning quality some of the features have no problem in surface mesh based skeleton extraction methods. For example by using LSD method with geometric distance computation, our skeleton is always invariant under isometric transformations.

# Chapter 4

## 4. Sketching Analysis

### 4.1. Introduction

In this chapter, we will discuss our sketching related work. It includes three major processes:

1. Deriving sketched bones
2. Sketched bones clustering
3. Reference bone construction

It is difficult to allow a user to draw 3D lines directly in a model. A user can only sketch on a 2D screen. We make use of projection technique to derive the 3D sketched bones from his 2D sketched lines. However, the user’s drawing may contain duplicated sketched bones or some sketched bones may need to be divided into several parts. To clarify the user’s intention, we need an algorithm to cluster similar sketched bones into one group. Within a sketched bone group, we create a reference bone, which is a combination of all the sketched bones in the group. Also we need to collect all the surface associated data (we use triangles in our experiment) for each group. We will use those triangles as seeds in our segmentation process, which will be discussed in Chapter 5.

We allow the user to draw freely, which is used to express the topological structure from the user's point of view. We provide skeleton modeling and real time editing during the sketching process. The major contributions of this part of the thesis are listed as follows:

1. We introduced a new algorithm to derive sketched bones in 3D space. By analyzing the

surface normal variation of sketched region and the distance variation between projections, we can create stable sketched bones.

2. We proposed a smoothing algorithm to eliminate the distortions from both user sketching and incorrect surface projection.
3. We proposed an algorithm to detect the user's drawing topology. This algorithm can detect whether we need to break down the sketched bone into several bones due to the topology property change.
4. We made use of three properties to cluster the sketched bones into different groups. They are the locality, the orientation and the overlapping. Based on those properties, we clustered the sketched bones into different groups.

## 4.2. Deriving Sketched Bones

Our work is challenging because we take non-trained user sketching as input. Hence, the input may contain mistakes. In addition, although many methods have been developed to produce suitable joints when performing automatic skeleton extraction, a perfect solution is still not available. A typical way to address this problem is by applying local minima together with certain distance constraints [21] to build the skeleton, but the result is sensitive to local features or noise. Recently, [18] adopts the Harmonic function to produce better joints. However, it relies heavily on the existence of mesh model symmetry.

### 4.2.1. Problems Analysis in Deriving Sketched Bones

A sketched bone represents the topology feature within the mesh part that the sketched bone is located at. Thus a sketched bone should be smooth, and roughly like a straight line segments. To get such a sketched bone, we need to solve the following problems:

1. Projection problem
2. Topology problem
3. Smoothness problem

A user can roughly draw some strokes on a 2D screen. In order to get 3D sketched bones, we need to project those 2D strokes into 3D space. Also some strokes may contain large curvature changes; we need to check those sketched bones whether we need to divide those bones into several parts. To eliminate the distortions from projection process, we also need to perform smooth operations to the final sketched bones.

We project user's strokes into a mesh model to construct sketched bones. An example of such method can be found in [47], in which sketched bones were created from two projected layers of a given mesh model. The sketched bone was created by linking an ordered set of points which were the center points of back and front layers as shown in Figure 4-1.

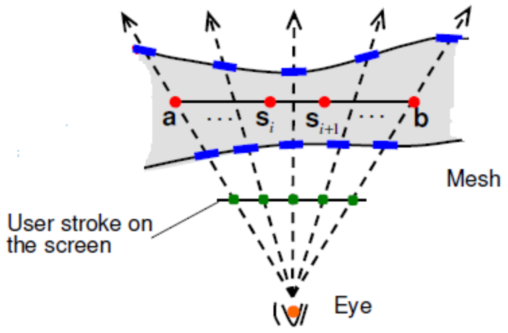

**Figure 4-1 Sketched Bone Projection**

Figure 4-1 describes the way to create a sketched bone inside a mesh model. From the viewpoint, a user sketched a stroke on the screen, which is indicated using green dots. During the projection, each green point will pass through a mesh model with two interaction area, which is marked as the blue line segments. For each projection, we can obtain a central point $s_i$ . By linking those ordered points $\{s_i \mid i = 1, ..., n\}$, we finally create a sketched bone inside the mesh model. Point $a$ and point $b$ are the two end points for a sketched bone.

We make use of orthogonal projection to construct sketched bones. Figure 4-2 is an example of using orthogonal projection method to construct a sketched bone. A user sketches a 2D line (the red line at the bottom of Figure 4-2). This line contains a set of points, which is marked as green dots. We project each green dot inside the mesh part. Each projection contains two intersection regions, which are marked by blue line segments. The middle point (red points) of these two intersection regions is one of the joints which are used to construct a sketched bone. However, we cannot guarantee the result due to some regions like the projected region is irregular or the viewpoint is not set properly. Figure 4-3 is an example of unstable result, where the sketched bone is created using two interaction areas. Thus for each projection, we also store the distance $d$ between two intersection regions. This distance is used to monitor the quality of the projection. Given a list of $d$, if the value contains great change in some position, we need to remove those projections. The reason is obvious, if the projection contains large distance change, it means that the topology has great change within that region. We use Figure 4-3 to illustrate this situation.

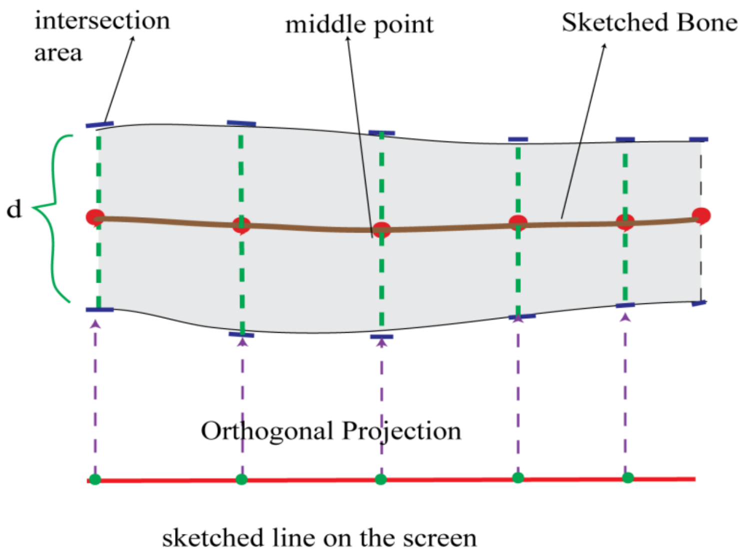

**Figure 4-2 Our Projection Method**

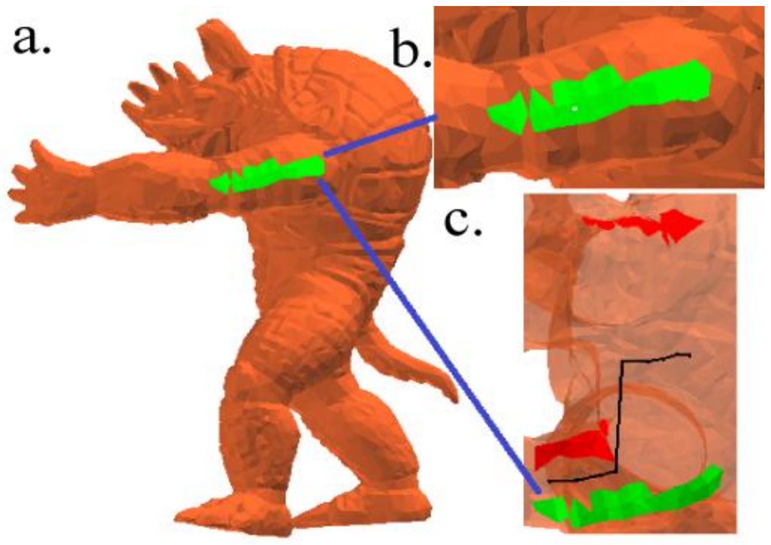

**Figure 4-3 Projection Problem**

Figure 4-3.a displays the projection result for a single stroke. The green color indicates the intersection region on the surface. And Figure 4-3.b is the enlarged picture for the front intersection region. The red region in Figure 4-3.c is another intersection region, which is behind the view of a user. To display the intersection region behind the view, we view the intersection result from the viewpoint above the shoulder. The black line is constructed by using a set of ordered points (see the red points in Figure 4-2). The point is the center of two intersection regions (see the blue line segments in Figure 4-2). If we do not monitor the distance (see the dotted green line in Figure 4-2) between the two intersection regions, we may get a wrong projection as shown in the black line in Figure 4-3.c.

### 4.2.2. Solutions to Derive Sketched Bones

To handle the problem as illustrated in Figure 4-3.c, we designed an algorithm to eliminate wrong projections in the sketched bone construction process. For each point $p$ in a sketched bone, we store a distance $d$ (see the virtual green lines in Figure 4-4) between the two intersection regions (see the blue line segments at the end of each dotted green line in Figure 4-2) which are used to create $p$. We compute an average distance $\bar{d}$ from all the points in a sketched bone[5]. Given a point $p$, if the distance $d > \bar{d}$, we remove the back-layer (the red line segment in Figure 4-4) from point $p$, and recomputed the location by using the front layer and the distance $\bar{d}$.

[5] We summarize the length of all the virtual green lines in Figure 4-4 and then divide by the number of points in the sketched bone to obtain the average distance length $\bar{d}$.

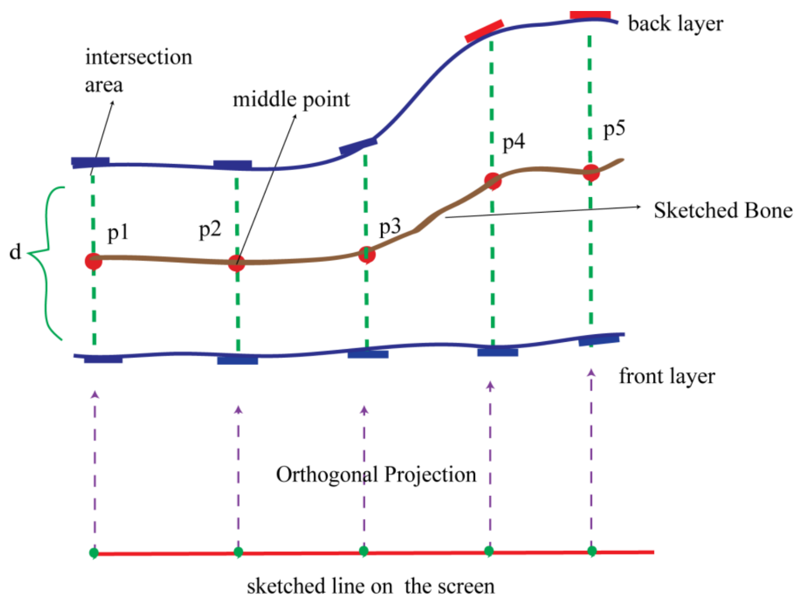

**Figure 4-4 Projection Optimization**

We depict the 3D drawings from the following steps:

1. First, we let the user to choose a suitable view coordinate to display the mesh model. The "suitable view coordinate" that we described here is the one which can display the maximum topology information in the view windows. See Figure 4-5.a for illustration.
2. Second, we propose an algorithm to compute the sketched bones in 3D. When a user sketches on the 2D screen, the sketched line contains a set of ordered points (see the green dots in the bottom of Figure 4-4). For each green dot, we use orthogonal projection to find a proper position (the red dots in Figure 4-4) inside the mesh model. After collecting the surface information (the intersection regions which are marked by blue line segments) from each projection, we compute the middle point of the surface information which makes the created point lies in the center of two intersection regions. We connect all the points in sketched order to construct the final 3D line (the brown line in Figure 4-4).
3. The sketched bone that we construct from last step cannot guarantee the correctness of

the sketched bone in 3D (see Figure 4-3). The projection selection from one view point coordinates may not reflect the real shape structure. In our system, we developed an algorithm to solve the problem. We monitor the distance variation between the two intersection regions which are collected from each projection (see the blue line segments in Figure 4-2). We remove the second layer from those projections whose distance between two layers is larger than average distance value (see the red line segments in Figure 4-4).

The sketch projection is sensitive to the viewpoint. We assume a user can find out the most suitable viewpoint to display the mesh model. Figure 4-5.a is a result of suitable viewpoint, whereas Figure 4-5.b is a result of non-suitable viewpoint. We assume the user can choose a suitable viewpoint which can display full topology features for a given mesh model.

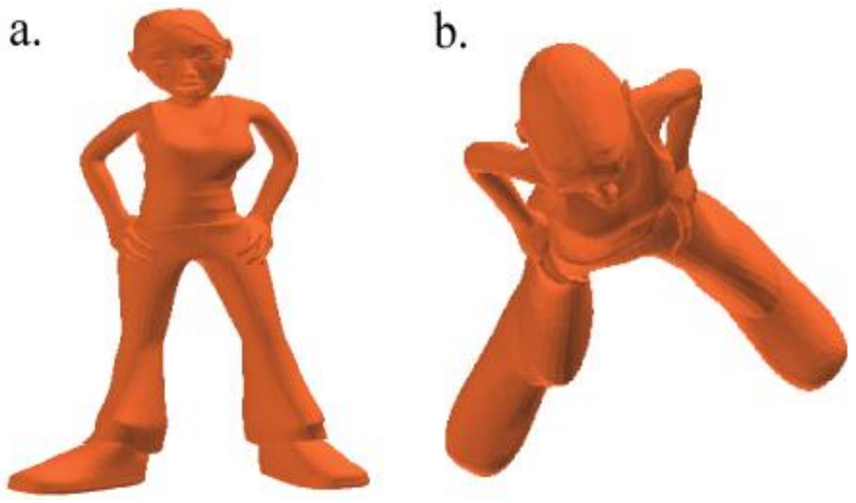

**Figure 4-5 Viewpoint Examples**

We use the following steps to find a suitable distance:

1. Given a list of distance value $ds[n]$, we initialize a list $ref[n]$, which is used to record the number of similar distance for each value $ds[i]$, $0 \le i < n$.

2. For each $ds[i]$, we first set a reference value $d_{ref}^{i} = \delta \cdot ds[i]$, where $\delta$ is a threshold to represent the error tolerance in the distance picking. if $M_{ij} < \delta \cdot d_{ref}^{i}$, we increase the number of $ref[i]$ .

3. By checking the index of the largest value in $ref[n]$, we find out a suitable distance to eliminate the error projections.

### 4.2.3. Sketched Bone Smoothing

Our sketched bone has the following data structure:

$$L_s = \{p_h, plist, p_t\} \tag{4.2.1}$$

where $plist = \{p_1, ..., p_n\}$ is an ordered list of points, $p_h$ , $p_t$ are the two ends of the sketched bone $L_s$ . We make use of an iterative way to smooth the extracted bones. During the smoothing process, we use two features, namely angle and distance, to measure the smoothness at position $i$ .

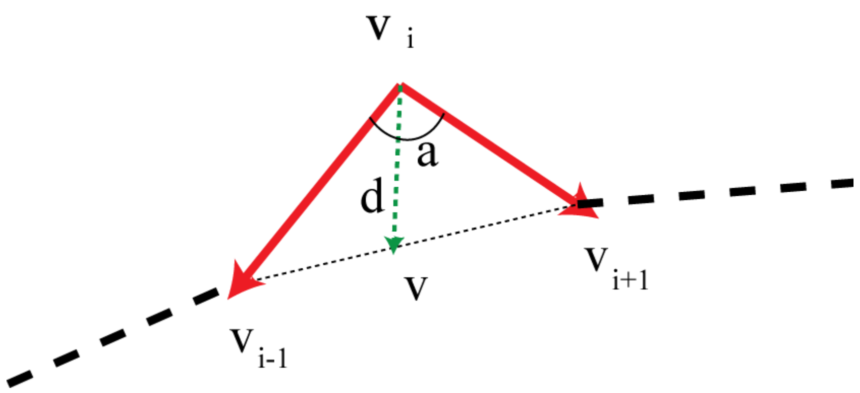

**Figure 4-6 Smooth Operation**

Figure 4-6 describes the basic concept of our smoothing algorithm. For each point $v_i$ in the $plist$ of $L_s$, we measure the angle $a$, which is constructed by the red lines in Figure 4-6, against a predefined angle tolerance threshold $\theta$. If $a < \theta$, we need to move point $v_i$ along the direction of the dotted green line (as shown in Figure 4-6) with a distance $\delta \cdot d$, where $\delta$ is a scalar value to determine how much a point $v_i$ will be moved, and $d$ is the Euclidean distance from $v_i$ to line $\overline{v_{i-1}v_{i+1}}$. Besides the angle constraint, we also considered the absolute distance constraint in our smoothing operation. To eliminate local distortions, we check the distance $d$ against a predefined threshold $\varepsilon$. For each point $v_i$, if $d > \varepsilon$, we perform the smoothing operation at point $v_i$ again. To avoid unnecessary computation, we also limit the smoothing iteration to 400 times.

## 4.3. Topology Check by Using Douglas-Peucker Algorithm

Our sketching analysis algorithm contains two independent processes: topology checking process and similarity checking process. The topology checking process is used to verify the sketched bone itself whereas the similarity checking process is used to make comparison with existing sketched bones.

Although a single freehand stroke generally corresponds to one sketched bone, as we allow a mesh model to be in any arbitrary posture, user sketching sometimes may become ambiguous. Particularly, a user may sometime sketch a roughly straight stroke on a bended mesh model region to create a sketched bone as this region may appear straight to the user from his/her view. In this case, the resultant sketched bone will be bent and should correspond to more than one piece of bone. On the other hand, as we tolerate imprecise sketching, we accept that a user may draw a single stroke across some bended regions of the mesh model to express more than one piece of bone. Hence, we further process the derived sketched bones to resolve the ambiguity problem. At this early stage, we do not intend to analyze the mesh model structure. We merely take user sketching as the prime factor to trace out separate bones. By doing this, we may also generate more accurate seeds to the mesh segmentation process to be described in Chapter 5.

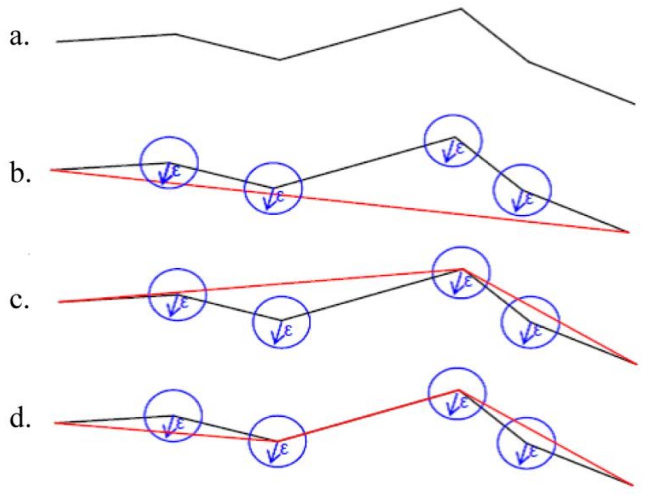

**Figure 4-7 Douglas-Peucker Algorithm Procedures [97]**

We run the Douglas-Peucker (DP) algorithm [98] to process a sketched bone to detect major curvature changes on the bone. Given that a sketched bone is a polyline with two end points $p_s$ and $p_e$, DP identifies point $p$ of the sketched bone which is located furthest from line $\overline{p_s p_e}$ and measure angle $\angle p_s p p_e$. If this angle is smaller than a predefined threshold $ang$, it divides the sketched bone into two parts at point $p$. DP is then run recursively on these two parts. This method is good at identifying major global curvature changes at any points of a line while minor curvature changes are ignored. Our objective here is to identify topologically meaningful separate bones while withstanding minor user errors. We have noted that a user may perceive a mesh model region comprising separate parts if it bends at an angle roughly equal to or less than $90$ degree. To keep this part of preprocessing simple, we use $ang = 100$ as the curvature threshold to run the DP algorithm. From our experiments, this threshold appears to work well on our tested mesh models. Figure 4-7 is an example of running DP algorithm, which comes from [97]. The idea of using DP algorithm is to detect great curvature change globally, which provides suggestion on whether we need to divide the sketched bones.

Despite the usage in sketched bone checking, we can use this method in our skeleton optimization process. If one bone contains large curvature change, we can use this method to divide one bone into several parts.

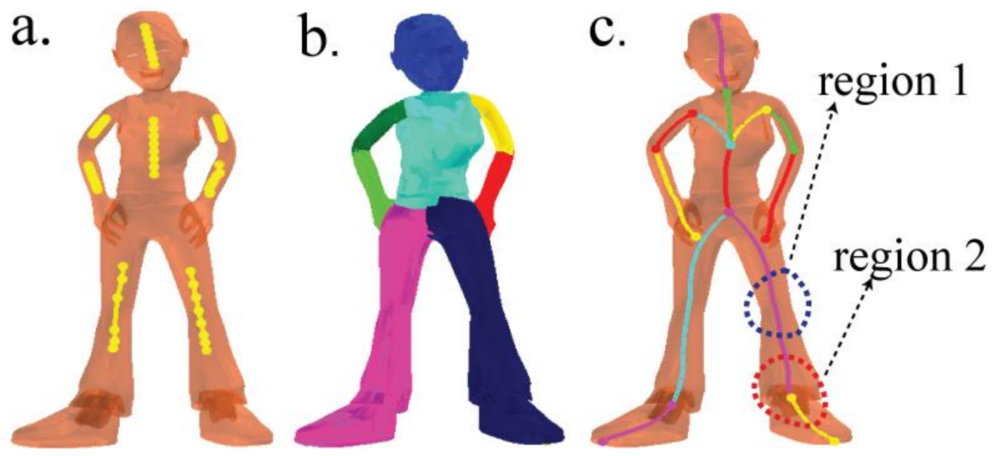

**Figure 4-8 DP Algorithm in Skeleton Construction**

We use Figure 4-8 to describe the functionality of our DP algorithm. Figure 4-8.a describes the user sketching process, where the yellow lines are depicted from user's drawings. Base on the reference bones (yellow lines in Figure 4-8.a), we segment[6] a mesh model into several parts. By running the DP algorithm, we detect that the bones in ankle regions (see Figure 4-8.c region 2, which is located in red circle region) should be divided into two parts, which is shown in Figure 4-8.c. However, the DP algorithm cannot be used to detect semantic changes such as the problem in region 1 part of Figure 4-8.c, which is located in blue circle region.

[6] The segmentation process will be discussed in Chapter 5.

## 4.4. Problem Analysis on Reference Bone Construction

We designed a sketching analysis algorithm to extract the user's intention from his/her drawing. In contrast to most of the existing sketch-based platforms, which provide button-based interface to perform editing process, we allow a user to modify their drawing by placing a duplicated drawing. And our system will merge those duplicated drawings to be one sketched bone.

Based on the location of sketched bones, we classified sketched bones into three different types: adjacent neighborhood, overlapping neighborhood and fake adjacent neighborhood. The adjacent neighborhood type refers to sketched bones which define a connection relationship (see Figure 4-9.a). The overlapping neighborhood refers to sketched bones which indicate the same topology definition within the same region (see Figure 4-9.b). The fake adjacent neighborhood type refers to sketched bones which are close to each other in geometric locations but logically indicate two different regions (see Figure 4-9.c).

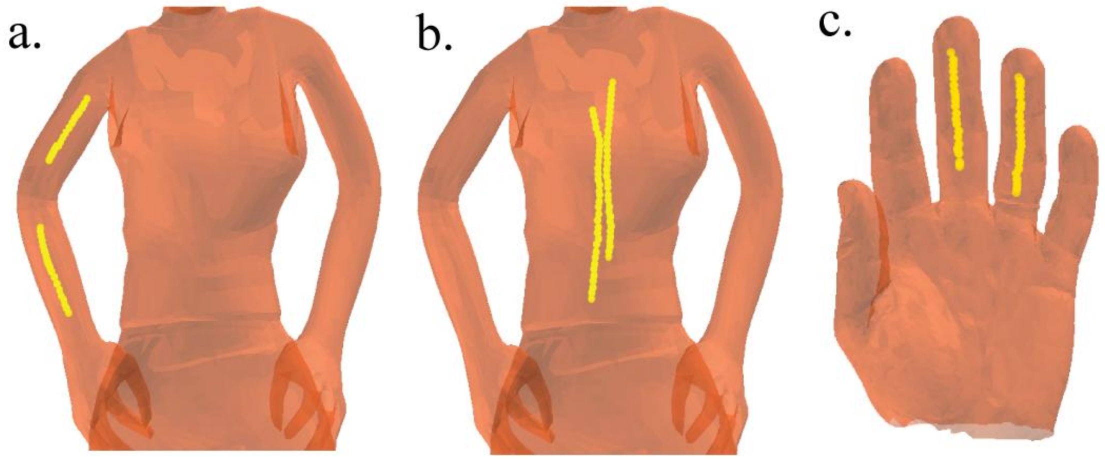

**Figure 4-9 Different Relationship among Sketched bones**

Our sketching analysis algorithm is critical as user sketching can be done from any orientation of a mesh model. Such flexibility is provided to allow user sketching to be done on a mesh model with any arbitrary posture, as the user may perceive the mesh model much better by altering the mesh model orientation, e.g., revealing an occluded mesh model part. In addition, sketched bones are prone to user mistakes. For example a user may produce multiple sketched bones, which may in some cases overlap each other, to represent the skeleton bone for a particular region of the mesh model.

The clustering process is done based on three criteria: **orientation**, **overlapping** and **locality,** which are described as follows:

- **Orientation**: We first cluster sketched bones into the same group if they form an acute angle (i.e. $\theta \leq 45^{\circ}$) with each other. This can be done by computing their dot products. In our experiment, we connect the two joints to form a straight line, and the orientation difference is measured by the angles between such two straight lines.

- **Overlapping**: For each cluster, we check if any sketched bones overlap each other from certain orientation in the 3D space. The exact evaluation function may involve complicated transformation operations, which are expensive. To simplify the computation, given two sketched bones, we treat one as the base bone, and the other one as the active bone. We then identify the closest points on the base bone to the two ending-points of the active line. If the distance between these two closest points is equal to or larger than 50% of the length of either line, then we consider the two lines overlapped. This 50% threshold is set based on our experiments to avoid either over- or under-estimation of overlapped sketched bones.

- **Locality**: This criterion is a measure to determine whether certain sketched bones logically correspond to the same region of the mesh model. Here, a simple distance metric is not applicable, as two closely located sketched bones may correspond to different regions of the mesh model. To determine the locality for each pair of overlapped sketched bones, we first extract the middle point of the projected overlapping segment of each of the two sketched bones. For each sketched bone, we create a cutting plane based on this middle point and the sketched bone direction. By evaluating the intersection between the cutting plane and the mesh model, we collect a list of neighboring mesh triangles on the mesh model to the middle point. If the triangle-lists of the two sketched bones have triangles in common, the two sketched bones are assigned to the same group.

## 4.5. Summary

In this chapter, we have discussed our method of extracting the reference bones from user drawings. It involves the followings steps:

1. Deriving sketched bones, which is used to project 2D sketched lines into 3D space;
2. Topology analysis, which is used to check the topology feature of a sketched bone;
3. Reference bone construction, which is used to cluster sketched bones into different groups.

In deriving sketched bones process, we proposed a projection refinement method to eliminate the projection errors. To clarify the topology feature of a sketched bone, we also designed a line smoothing method. As we allow a user to sketch freely on the mesh model surface, it is inevitable to get some sketched bones which contain large curvature change. We adopted a DP algorithm to detect the global curvature change for a sketched bone. If there is some large curvature change in the sketched bone, our method divides the sketched bones into several parts.

To extract a user's intention from his/her drawings, we also proposed a clustering method to collect sketched bones which have similar topology features together. We have defined three features to distinguish different sketched bones. They are namely: **orientation**, **overlapping** and **locality.** By checking these three features, we are able to cluster the sketched bones into different groups. Thus for each group, we can merge those bones together to construct a reference bone. Given a group of sketched bones, we pick up two bones for merging which result into a new bone, and then we pick up one bone from the rest of sketched bones to merge, until only one bone left in the group. The merging step is straightforward. Assume there are two bones $b_1$ and $b_2$ to be merged. We find out the overlapping regions $ov_1$ and $ov_2$ for each bone. For each node insides the overlapping region $ov_1$, we compute the nearest node in $ov_2$. We construct a new node by using the average position of such two nodes. We replace the old two nodes with is new one and merge the rest of nodes together to form a new sketched bone.

The major objective in sketching analysis part is to clarify user's intention from their drawings. In this step, we proposed a set of procedures to construct the reference bones, which are used in our segmentation process. There are two important elements: the seed triangles and the geometric locations of reference bones. The seed triangles are those triangles which are associated with the reference bones. Each reference bone contains a set of triangles which are collected in the sketching process (see the green and red regions of Figure 4-3). Those seed triangles are the initial triangles in our segmentation algorithm. The geometric locations of the reference bones are used to define the distance constraint for our segmentation algorithm.

# Chapter 5

## 5. Mesh Segmentation

### 5.1. Introduction

With regards to the animation purpose, a skeleton should have two major features: well-positioned joint specifications among different components of a mesh model and reasonable mapping relationship between joints and surface data (triangles or vertices). The well-positioned joint specifications can be used to create reliable skeleton-driven animations. For example joint positions can be used to compute different rotation matrix or translation matrix, which are used in LBS (Linear Blend Skinning) based animation system. However wrong mapping relationship between bones and surface data could produce undesired animation results. To avoid undesired animation results which contain distortions, we need to make sure the mapping relationship between joints and surface data are correct.

Based on the requirements that we have discussed above, a segmentation based method fits our needs. First, by running a segmentation process, a mesh model can be decomposed into several parts. We can extract one bone from a segmentation part such as the leg or arm parts. Or we can extract multiple bones from one part such as the main body of a human model. The joints are located at the center of the segmentation boundaries. Because each bone is extracted from one part only, the mapping relationship between joints and surface data is limited within a segmentation part. The only problem is to design a segmentation metric which has the ability to generate meaningful parts.

In this chapter, we will discuss the mesh segmentation method that we have adopted in our skeleton extraction platform. Our segmentation method takes the reference bones, which are deduced from the algorithm that we have discussed in Chapter 4, as input to decompose a mesh model.

In order to extract articulated skeleton, we adopt a multiple-region growing segmentation method to segment a mesh model into parts. Then we use the segmentation boundaries to extract bones. In this multiple-region growing segmentation method, we make use of reference bones and the triangles from each reference bone as the input to segment a mesh model into meaningful parts. Minima rule is a criterion to decompose a mesh model into meaningful parts. By computing the curvature change between any adjacent triangles, we are able to find the segmentation boundaries which are used to divide the mesh model into meaningful pieces. However, there are two problems in the existing minima rule implementations. First, the minima rule is based on the surface curvature features, which means that the minima rule based segmentation method is sensitive to the surface curvatures. As a result, the segmentation result may be incorrect due surface curvature distortion. Second, for those regions which do not have concave features, it is hard to get the correct segmentation result. This is because there is no criterion to segment a mesh model in the regions which have no concave features. In this chapter, we will discuss our segmentation method. The major contribution of our work is that we designed a new segmentation metric which can produce high quality segmentation results. Thus we can make use of those segmentation results to extract a well-positioned bone structure for a mesh model.

## 5.2. Background and related work

Mesh segmentation is a technique for partitioning a mesh model into mesh parts. There are two types of methods: surface-type and part-type methods. Surface-type methods make use of geometric properties, such as planarity or curvature, to create surface patches, i.e. segments, from a mesh model. Applications include texture mapping [84], building charts [83] and geometry-image creation [85]. In general, surface patches created are topologically equivalent to a disc and do not necessarily possess any semantic meaning.

Part-type methods are rooted in the study of human perception for producing semantically meaningful mesh parts. A typical method is region growing [99], which uses geometric criteria to cluster mesh polygons around some selected seed polygons locally. The segmentation results from such method are significantly affected by the local feature of a mesh model and the choice of the seeds. To offer better results, hierarchical clustering method [87] treats each mesh polygon as a separate cluster initially and merges neighboring clusters based on certain criteria recursively. Alternatively, interactive clustering method [90] searches for the best segmentation for a given number of segments by allowing iterative seed updating to be performed based on the changes in cluster characteristics during clustering construction. This explicitly addresses the seed selection problem. In general, to allow a part-type method to generate much meaningful mesh parts, the selection of segmentation seeds and segmentation criteria must be done very carefully.

Katz et al. proposed a hierarchical based segmentation method [66] to segment mesh model into parts. The hierarchical method decomposes the mesh model by computing the probability of a vertex to different patches which are predefined in their system and then decomposing a mesh model based on some fixed thresholds. The thresholds are used to define fuzzy regions among different parts. Based on some predefined distance metric, the fuzzy regions are further divided into their nearest patches. So the segmentation result depends on the threshold which is used in their decomposition process.

Lee et al. proposed a segmentation method in [30] which was based on minima rule and part salience. This method performs well on surface-type segmentation. However, large mesh models usually contain lots of details on the mesh surface, which also known as local features. The segmentation result using minima rule and salience curvature is highly sensitive to the surface details. Thus it cannot guarantee the result is semantically meaningful.

Similar to Lee's work, Ji et al. made use of minima rule to perform surface-type segmentation in their easy mesh cutting framework [88]. As we notice, there are two important factors which affect the segmentation result. The first one is the reference seed selection, and another is the distance metric design. The reference seeds are used to define the number of parts for a mesh model whereas the distance metric defines the priority for all the segmentation elements (such as triangles or vertices) in a segmentation process. Ji's method can only be used in surface-type segmentation. Rather than performing surface-type segmentation, we prefer to segment a mesh model into parts.

## 5.3. Segmentation Metric

Pottmann et al. introduced the isophotic metric [1]. The isophotic metric distance between two points $p$ and $q$ is defined as:

$$d_{\Gamma}(p,q) = \int_{\Gamma} ds + w\int_{\Gamma^*} ds^* \tag{5.3.1}$$

where $\Gamma$ is the path between point $p$ and $q$ on the surface, and $\Gamma^*$ is the Gaussian image of $\Gamma$, and $ds^*$ is the arc element of $\Gamma^*$ whereas $ds$ is the arc element of $\Gamma$. $w$ is the weight for the Gaussian image of $\Gamma$. A segmentation result, which makes use of this isophotic metric, is shown in Figure 5-1.

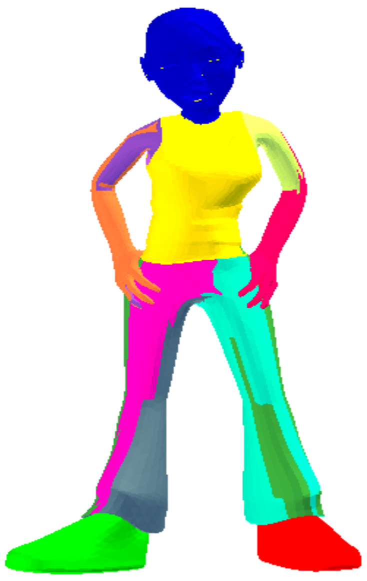

**Figure 5-1 Segmentation Result from Pottmann's Method**

Pottmann's metric can distinguish the boundaries which have large curvature changes, such as the neck region and the waist region shown in Figure 5-1. However, there are some over-segmentation problems (see segmentation result for leg regions). So, one requirement in the part-type segmentation is to avoid over-segmentation. We need to limit the segmentation process within some predefined regions, such as the femur region or fibula region in Figure 5-1.

By considering the curvature in the direction tangent on the path $\Gamma$, Ji et al. proposed an improved isophotic metric as:

$$d_{\Gamma}(p,q) = \int_{\Gamma} ds + w\int_{\Gamma^*} ds^* + w^*\int_{\Gamma} f(k_D)ds \qquad (5.3.2)$$

where the $k_D$ is the normal-section curvature in the direction tangent to the path $\Gamma$. $f$ is a function of curvature, and $w^*, w$ are the weights for the curvature metric and normal metric.

As the minima rule specified, all negative minima of the principal curvatures (along their associated lines of curvature) form boundaries between perceptual parts. Ji et al. created an augment function $f(k_D)$ to enlarge the effect of negative curvature in the improved isophotic metric. The function is defined as:

$$f(k_D) = \begin{cases} k_D, & \mathrm{k}_D \geq 0, \\ g(|k_D|), & \mathrm{k}_D < 0, \end{cases} \tag{5.3.3}$$

where $g(x)$ is an augmentation function (i.e. $g(x) = x^2$ or $g(x) = e^x$).

The improved isophotic metric from Ji et al. makes use of normal curvature and tangent curvature information to detect the segmentation boundaries of a mesh model. By augmenting the negative curvatures, the improved isophotic metric can distinguish the concave regions clearly. However, it only suitable for surface-type model isolation, where a model is divided into two parts, based on the surface curvature features. When we use the existing isophotic metric to handle multiple-region based segmentation, we notice that the existing isophotic metric cannot produce reasonable segmentation results. There are two problems in using the above metric:

- Local curvature noise, which may affect the segmentation results.
- Uncontrollable result to smooth regions which have no concave curvatures.

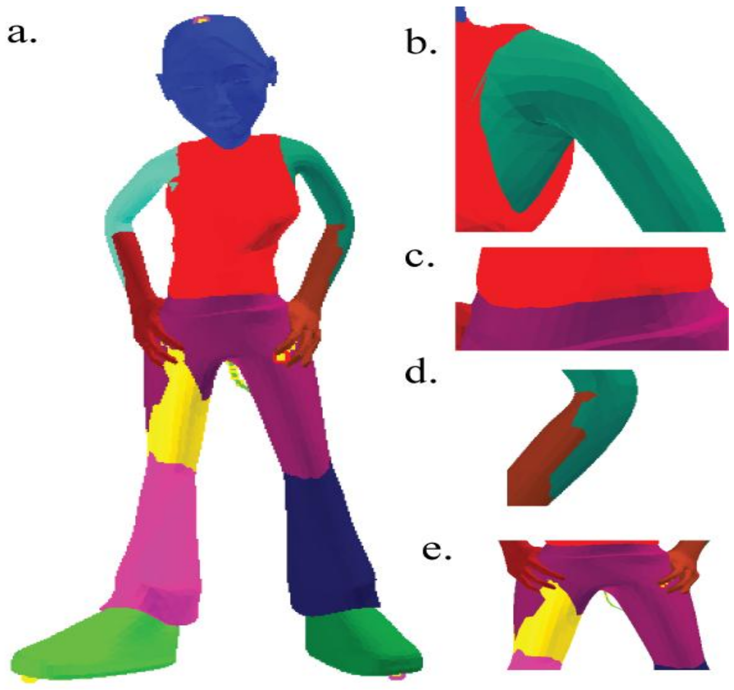

**Figure 5-2 Segmentation Result from Ji's Method**

The segmentation result[7] from Ji's metric has some improvement with regards to the over-segmentation problem. But the over-segmentation issue is not solved properly. From Figure 5-2.b and Figure 5-2.c, we can see that Ji's metric distinguishes the parts clearly on the region where large curvature variation occurs. However, if the region between any two mesh parts contains smooth or flat feature (such as the region between two hands in Figure 5-2.e), the segmentation boundary is irregular and uncontrollable. This is because those smooth regions will result in over-segmentation results due to the lack of criteria to limit the segmentation process. From the segmentation result, we notice another requirement for part-type segmentation metric: the boundaries for each part should be reliable. If there are concave regions, the boundary should

[7] In the experiment, to make a fair comparison, we have selected a set of proper parameters. (In Ji's work, we set the weights for normal and curvature to 0.2 and 0.8).

locate in the concave regions. If there is no great curvature change between any two adjacent parts, the boundary should be reliable.

## 5.4. Our Segmentation Method

We designed our segmentation metric based on two requirements:

1. The segmentation metric should avoid the over-segmentation issue.
2. The segmentation metric should generate smooth and reliable boundaries.

To avoid the over-segmentation problem, we have let the user to define the number of mesh parts via his/her sketched bones. We also introduce a distance metric to limit the segmentation process within some regions, so that we can obtain reliable mesh parts. We make use of the reference bones for each region as the reference positions to control the segmentation process. However, the boundary for each mesh part may be not smooth or even contains large variations for some mesh parts which have smooth regions between any two adjacent regions. Thus to obtain reliable boundaries, we also consider the boundary curvature change in our distance metric design.

### 5.4.1. Curvature and Distance based Segmentation

Before introducing our segmentation metric, let us recall the basic steps of our segmentation scheme:

1. Defining the source points. Each source point defines a mesh part, which finally becomes one segmentation part.

2. Collecting candidate points and computing the distance for each candidate to the source points, choose the mesh part which has the minimum distance between each candidate to its source point as the target mesh part.
3. Sorting the candidates based on the distance value, and performing the segmentation operation for the candidate which has the minimum distance.
4. Repeat Step 2 and Step3 until all points are processed.

In our experiments, we notice that Ji's metric cannot generate reasonable result for flat regions which have multiple branches, such as the waist region of a human model. This is because there is no criteria to define the segmentation for those regions which have multiple branches.

On the other hand, the boundary of some semantically distinct segments of a mesh model may not possess any clear trait to allow them to be distinguished. An example is the thighs and the legs of the human model used in our experiments, where the knees located between these two limb parts do not possess clear traits to differentiate the connecting thighs and legs. Consequently, a direct adoption of the isophotic metric may cause either over- or under-segmentation, leading to inaccurate skeleton bone extraction. This is because that the metric does not consider the segmentation situation between two smooth regions (see Figure 5-2.d and Figure 5-2.e).

To address this problem, we modify the isophotic metric [86] as follows:

$$d(\Gamma, l) = w_1 \int_{\Gamma} ds + w_2 \int_{\Gamma} D_N ds + w_3 \int_{\Gamma} C ds + w_4 \int_{\Gamma} D_B(l) ds \tag{5.4.1}$$

where $D_N$ is a function to compute the normal vector variation on the surface. $C$ is a function to evaluate the curvature change along the tangent direction of the surface, which is guided by region growing direction. $D_B(l)$ is a function to measure the distance from the reference bone $l$, which helps determine the relevance of a mesh model vertex to a user sketched bone. $w_i$ are application dependent weights to adjust the contribution of each function in Eq-(5.4.1) where $\sum_{i=1}^{4} w_i = 1$. $\int_\Gamma ds$ defines the distance between the source region and the element to be clustered.

In our segmentation algorithm, we use triangles as the basic element. Assume $S_i$ is one mesh part, and $t$ is a triangle which is one of the boundary triangles to be clustered into $S_i$. $\int_\Gamma ds$ defines the distance between triangle $t$ and segmentation set $S_i$. It is clear that, to obtain a reasonable segmentation result, we will include the triangle $t$ which has the minimum distance to its neighbor cluster.

Figure 5-3 is a segmentation result comparison between Pottmann's method and Ji's method. Figure 5-3.a is the result of Pottmann's method, which only considers $\int_\Gamma ds$ and $\int_\Gamma D_N ds$[8]. Figure 5-3.b is the result of Ji's method[9], in which a horizontal curvature constraint is added. Ji's method has some improvements on the over-segmentation issue. However there are two problems which have not been solved properly. First, over-segmentation issue is still not solved which makes it is impossible to get a reasonable skeleton from the segmentation result. Secondly, the boundaries within a high curvature change regions may not correct (see Figure 5-2.b for illustration, where the boundary is highly affected by the curvature change on the mesh surface).

[8] According to our experiment, we set the weights for $\int_\Gamma ds$ and $\int_\Gamma D_N ds$ as 0.3 and 0.7 respectively to get a better segmentation result. However, the segmentation result is still unacceptable.

[9] To get reasonable result, we set the weights for normal variation and curvature variation as 0.2 and 0.8 respectively.

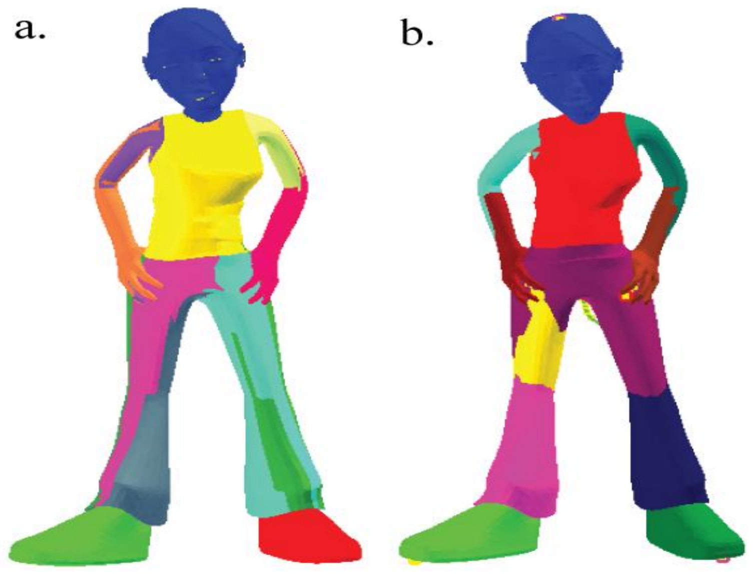

**Figure 5-3 Pottmann's Result vs Ji's Result**

Ji et al. put more emphasis on the curvature issue, so that the segmentation can distinguish the boundaries which have large curvature changes. However, for smooth regions, where no sudden curvature changes, there is no criterion to define the clustering orders. As a result, the segmentation result is uncontrollable (see the arm and leg regions in Figure 5-2). To define the clustering order in such a situation, we defined a distance constraint $D_B(l)$. If two mesh parts meet in a smooth region, the curvature and normal changes for the elements to be clustered are quite small. The distance constraint $D_B(l)$ defines the priority of the segmentation for each element to be clustered into different groups. The element, which has the shortest distance, has the highest priority in the clustering process. We can see the effect of adding this term in Figure 5-4.a.

We consider the boundary problem as the most important issue. To address this problem, we apply a modified isophotic metric on the mesh triangle level using the following discrete form of the equation:

$$d(p,q,l) = w_1 \,|\, p - q \,| + w_2 Aug(|\, n_p - n_q \,|) + w_3 Aug(C(p,q)) + w_4 D_B(q,l) \qquad (5.4.2)$$

In the equation, we have two triangles $p$ and $q$. $p$ is a triangle inside one mesh part $S$, whereas $q$ is a boundary triangle outside of $S$. $p$ and $q$ share one edge. $|\, p - q \,|$ denotes the Euclidean distance between $p$ and $q$, which is computed from the center of those two adjacent triangles. $|\, n_p - n_q \,|$ is the normal vector different between $p$ and $q$. $w_i$[10] $(1 \le i \le 4)$ is the weight for different components, and $\sum_{i=1}^{4} w_i = 1$. $Aug(x) = e^x$ is an augment function, which is applied on top of the normal vector variation and the curvature change functions to exaggerate their effects, since these two factors are critical to our method in order to obtain a better segmentation result. $C(p,q)$ computes the curvature change between triangles $p$ and $q$, considering if these triangles are located in a concave or a salient region. If they are located in a concave region, the normal vector of either triangle will form an angle of less than $90$ degree with the line connecting the two triangles. In contrast, such an angle will be larger than $90$ degree if it is located in a salient region. $C(p,q)$ is computed as:

$$C(p,q) = n_p \cdot \overrightarrow{p_c q_c} \qquad (5.4.3)$$

where $n_p$ is the normal vector of triangle $p$. $p_c$ and $q_c$ are the centers of triangles $p$ and $q$ respectively. $\overrightarrow{p_c q_c}$ is the normalized vector from $p_c$ and $q_c$. We use $D_B(q,l)$ to measure the shortest distance between triangle $q$ and reference bone $l$.

[10] In our experiment, the weights are set as $w_1 = w_3 = w_4 = 0.2$ and $w_2 = 0.4$.

Figure 5-4 is a result of Eq-(5.4.1). Figure 5-4.a is an overview of the segmentation result. Although the boundaries look fine, especially in the regions around leg joints and arm joints there are still some problems. Figure 5-4.b and Figure 5-4.c are two examples of these problems. In Figure 5-4.b, the segmentation boundary of the arm is located in the region where large curvature variation occurs. The segmentation result is highly affected by the surface curvature. In Figure 5-4.c, the boundary between two legs is irregular, that's why Ji et al. proposed a snake algorithm to smooth the boundary. By adding the distance constraint to reference bones we are able to segment the region which multiple branches, such as the waist region in Figure 5-4.c. However, the segmentation boundary still contains large curvature changes. This is because that we do not have a criterion to monitor the smoothness of the boundaries during the segmentation process.

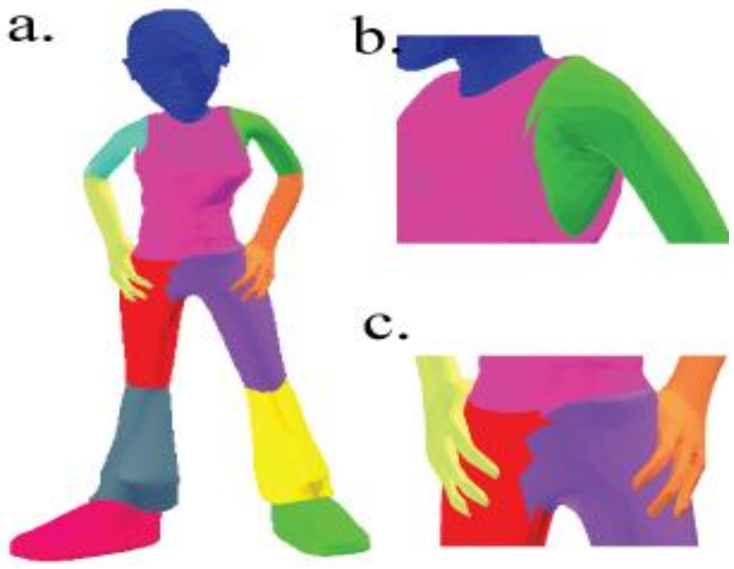

**Figure 5-4 Example of Adding Sketched Bone Constraint**

We use different models to evaluate the correctness of our proposed metric. We test our segmentation metric with three different groups:

1. Simple mesh models (see Figure 5-5) – In this group we use two low resolution models to perform our segmentation task. The rhino model (Figure 5-5.a(i)) contains 2835

vertices and 6012 triangles; the hand model (Figure 5-5.a(ii)) contains 349 vertices and 679 triangles.

2. Fine mesh models (see Figure 5-6) – In this group we use two models which have high resolutions on the surface to perform our segmentation task. The camel model contains 16984 vertices and 33964 triangles, whereas the horse model contains 5229 vertices and 10454 triangles. These two models contain fine level of local surface details, such as the concave regions around the upper leg regions in Figure 5-6.a(i) and Figure 5-6.a(ii).
3. Models (see Figure 5-7) which have lots of surface details, or complex poses. Figure 5-7.a(i) is an armadillo monster which has lots of surface details. Also some regions have no curvature features to distinguish two parts of the mesh model, such as the region between an arm and the main body region which is shown in Figure 5-7.b(i). Figure 5-7.a(ii) is another example which has complex pose. Such a model may hide the curvature feature between two parts of a mesh model, such as the region between the upper arm and the main body region as shown Figure 5-7.d(ii).

For mesh models which have low resolution, it is hard to segment them into low-level parts. For example there is no concave curvature in each finger in Figure 5-5.a(i) or Figure 5-5.a(ii) . We can only segment them into obvious parts as shown in different colors of fingers. In Figure 5-5.a(i), we cannot distinguish the upper and lower legs due to the low-resolution problem in the Rhino model. It is obvious that a mesh model is easy to be decomposed into several meaningful parts (as shown in Figure 5-5.a(i), Figure 5-5.b(i) and Figure 5-5.c(i)) by using the curvature and distance constraint from reference bones. There is no over-segmentation issue in this low-level group. This is because we include a distance constraint (distance from surface triangles to reference bones) to limit the segmentation process.

We notice that the horse model in Figure 5-6 is divided into reasonable parts despite the unsmoothed boundaries as shown in the blue mesh part of Figure 5-6.b(i). The Camel model in Figure 5-6.a(i) has an over-segmentation issue as shown in the blue leg region. According to our investigation, most of the over-segmentation occurs at a region which contains smooth surface features. Figure 5-7.b(i) shows the problem again, where the boundary between the upper arm region and the main body region of the monster model is inaccurate. This is because that there is no constraint to limit the segmentation within a region in the segmentation process**.**

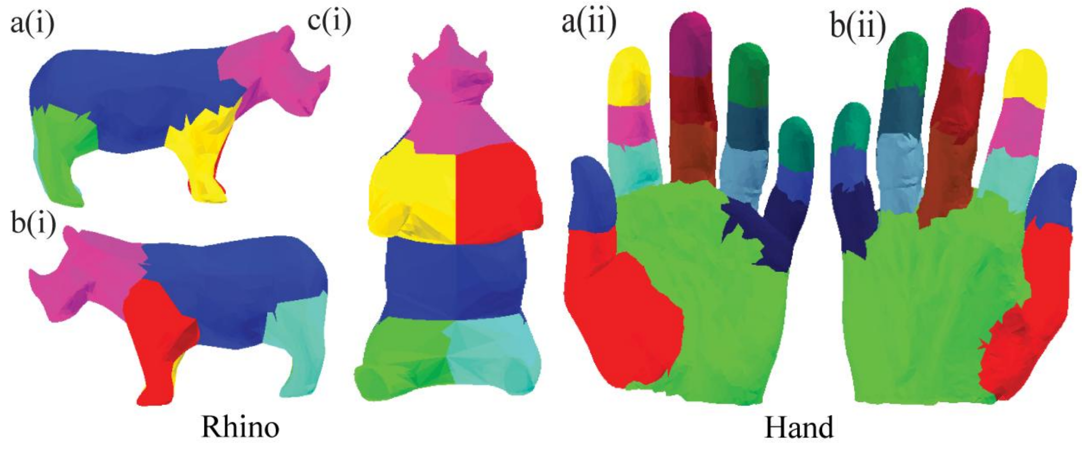

**Figure 5-5 Level 1 - Segmentation Results for Simple Mesh Models**

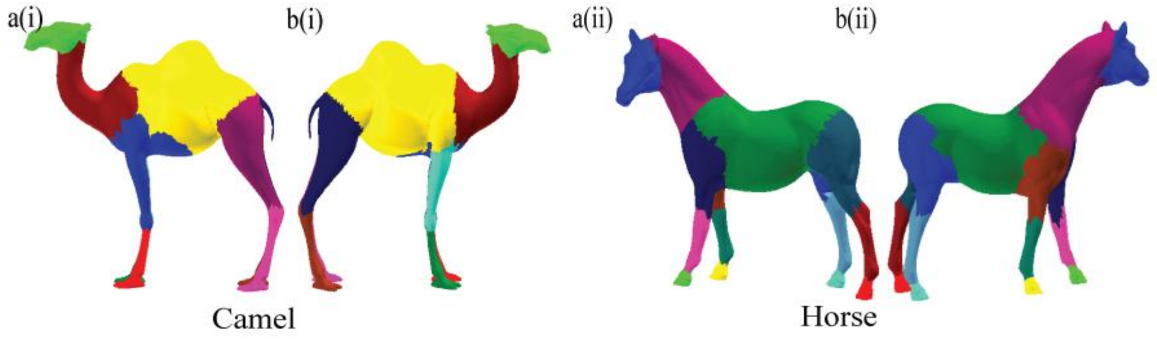

**Figure 5-6 Level 2 - Segmentation Results from Fine Mesh Model**

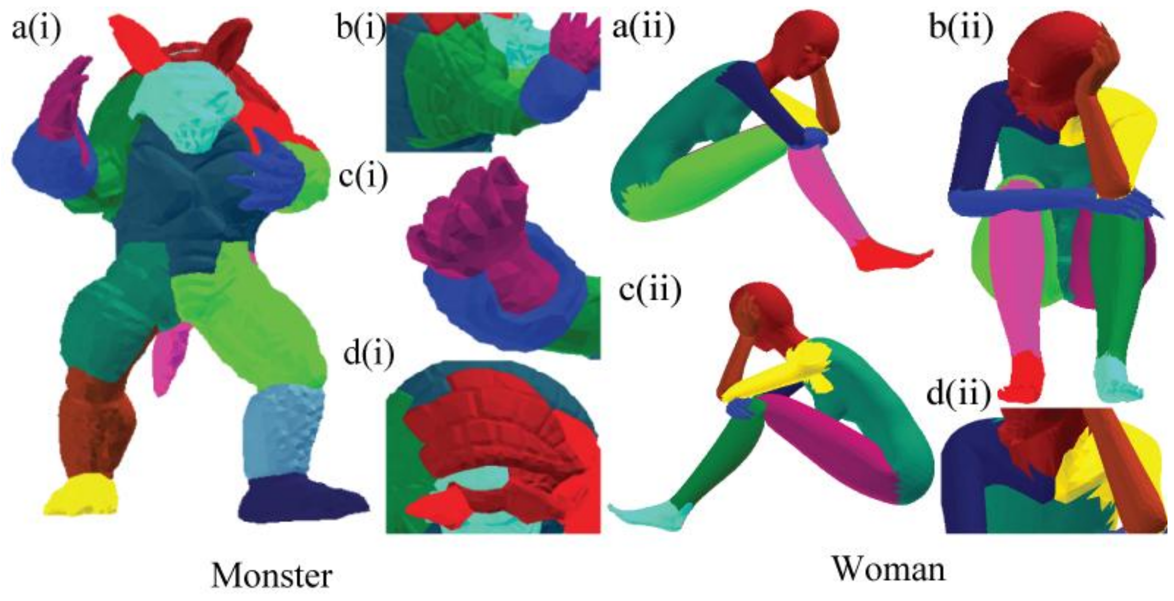

**Figure 5-7 Level 3 – Segmentation Results for Complex Models**

We tested our segmentation metric against two models (See Figure 5-7): an Armadillo monster which has lots of irregular surface details and a woman which has a complex pose. The main problem is that the metric from Eq-(5.4.1) cannot solve over-segmentation issue for models which have complex poses such as Figure 5-7.a(ii), or models which have lots of curvature variations on the surface, as shown in Figure 5-7.a(i). As shown in Figure 5-7.b(i) , Figure 5-7.c(i) and Figure 5-7.d(i), there are some over-segmentation results in those figures. Again, the same problem occurred in Figure 5-7.b(ii), Figure 5-7.c(ii) and Figure 5-7.d(ii).

As we notice that there is no over-segmentation issue in the elbow region or knee region of the above two models in Figure 5-7. The results in those two regions indicate that our distance constraint, which is measured by the Euclidean distance between a triangle and a reference bone, performs well in the regions which have tube-bar shape. However, we also notice that there are some over-segmentation results in the shoulder joints regions as shown in the two models of Figure 5-7. The over-segmentation problem indicates that our distance metric cannot handle the over-segmentation issues in such regions. The major reason is that the Euclidean distance cannot be used to determine the semantic segmentation in such regions.

### 5.4.2. Segmentation Analysis and Optimizations

By adding a new distance constraint, which is illustrated in Eq-(5.4.1), we solved the over-segmentation issue. But this only works for smooth mesh models which have topological specification. When it comes to complex model, such as the two models in Figure 5-7, the segmentation result is not as good as we expected. The major problem is that some of the semantic topology features are not clear and the distance constraint does not work well in such regions. An example of such a situation can be found in Figure 5-7.b(i). Also the segmentation result is highly affected by the curvature changes on the surface. An example is illustrated in Figure 5-4.b, where the boundary between arm and the main body is affected by the local curvature. To get a better segmentation result, we further explored the factors which may affect the final segmentation result.

The normal variation is used to measure the curvature change perpendicular to the surface. But our segmentation process is a multiple-region growing segmentation process, in which we only need to monitor the surface variations which are tangent to the surface of the model. And the curvature change $\int_{\Gamma} Cds$ can distinguish both the concave region and the convex regions clearly. Theoretically, the neighbor distance length $\int_{\Gamma} ds$ computes the distance between the region source point and the destination points. However, in our multiple-region growing segmentation process, the region source points are the boundary points. The boundary points are dynamically changed during the segmentation process, and the length is not an accumulative length[11] which is computed from the source point to the boundary point. In this case, we removed two terms: the normal variation $\int_{\Gamma} D_N ds$ and the neighbor distance length $\int_{\Gamma} ds$.

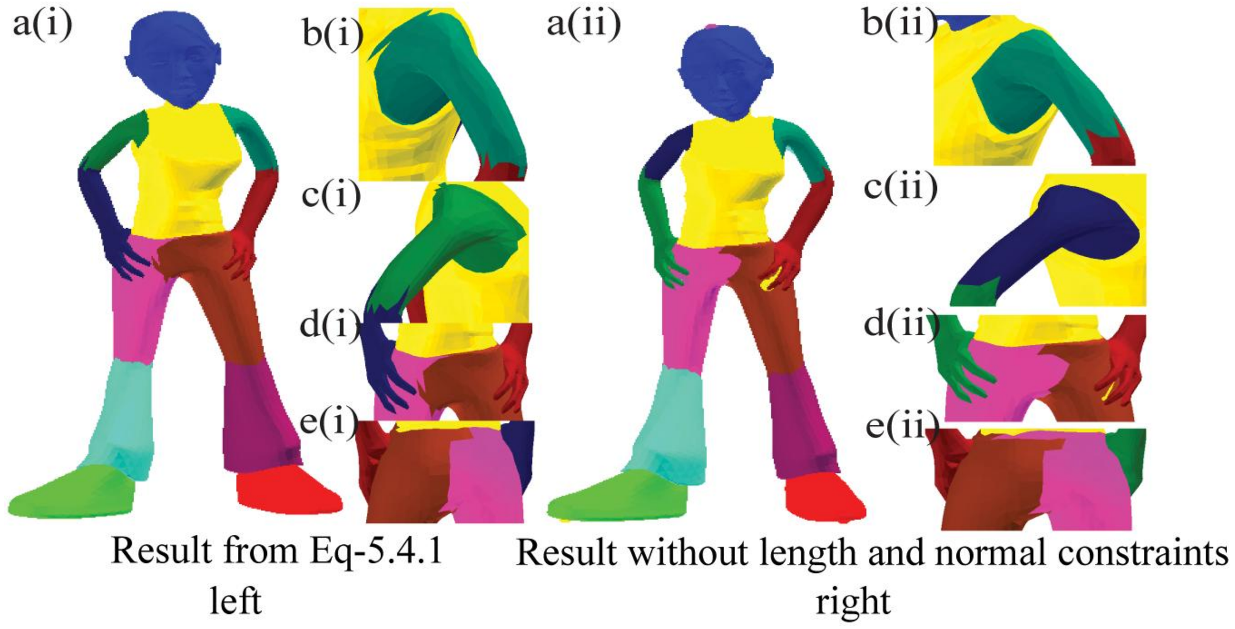

**Figure 5-8 Importance of Length and Normal Constraints**

Figure 5-8 describes the segmentation result comparison between metric using Eq-(5.4.1) and metric without length and normal constraints. Figure 5-8.a(i) describes the segmentation result from Eq-(5.4.1), whereas Figure 5-8.a(ii) describes the segmentation result without length and normal constraints. Some boundaries are smooth and reasonable such as the boundaries in the knees region, or the boundaries around elbow regions. Some of the boundaries are affected by the local curvatures, such as the boundary shoulder regions (as shown Figure 5-8.b(i) and Figure

[11] This is because that every time, the distance is computed only based on two points: one is boundary point and another is the candidate point to be clustered.

5-8.c(i)). We also obtained a well-defined boundary in the hip region as shown in Figure 5-8.d(i) and Figure 5-8.e(i).

According to our experiments, most of the fine mesh models have very small distance variation between any two adjacent triangles. Figure 5-8.a(ii) is a segmentation result which does not consider the adjacent triangle length and the normal variation. The normal variation cannot distinguish the difference between concave and convex change. However, the curvature variation can distinguish both. So, the normal term which is used to detect the surface variation, actually decreases the sensitivity of the curvature changes. The segmentation boundary in Figure 5-8.b(ii) and Figure 5-8.c(ii) are similar to the results in Figure 5-8.b(i), c(i). The segmentation results from hip region are also similar (see Figure 5-8.d(i), e(i) and d(ii) and e(ii)).

#### *5.4.2.1. The Segmentation Direction*

We notice that some segmentation results are highly affected by the surface curvatures, which may result in wrong segmentation, such as the example in Figure 5-10.c. Figure 5-10.c is the segmentation result in the upper arm region. The region within a yellow circle is segmented to the wrong group, which means it should belong to the main body part rather than the upper arm part. It is easy to imagine that for tube-shape model, the skeleton which lines in the central axis of such tube-shape should perpendicular to most of the surface triangles, despites the two end regions. We use Figure 5-9 to describe this concept. In Figure 5-9, we use a dotted red line to represent a skeleton whereas the dotted blue line with an arrow is the surface normal for a triangle on the cylinder mesh model.

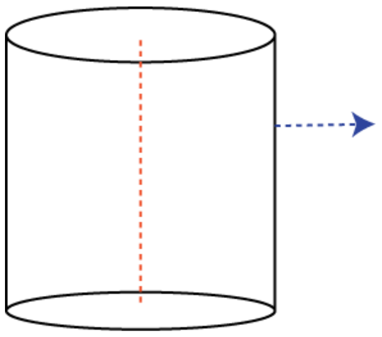

**Figure 5-9 Orientation Relationship between Bone and Surface Normal**

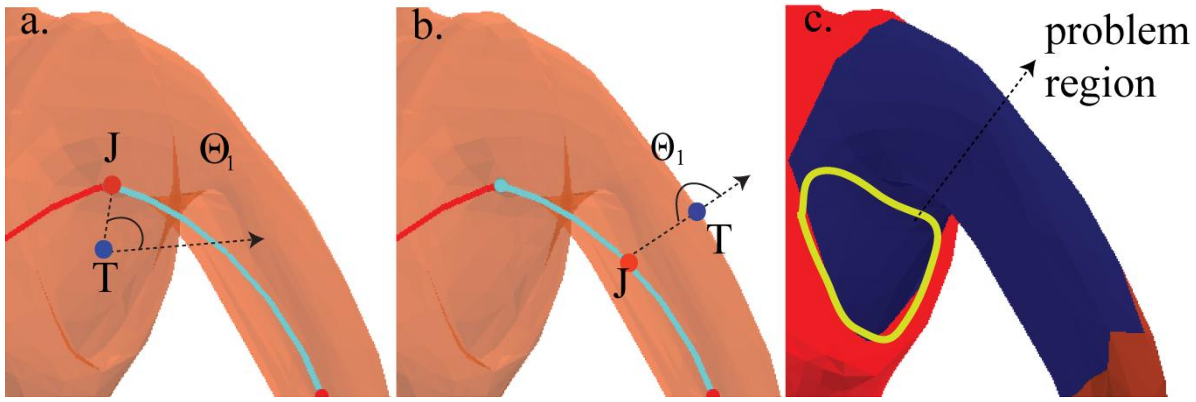

**Figure 5-10 Segmentation Directions**

Given a triangle $T$ on a mesh surface, we use an angle $\theta_1$ (see Figure 5-10.a) to measure the correctness of the segmentation. In Figure 5-10.a, the blue dot is a triangle $T$ in the mesh surface, and the dotted line with an arrow is the normal vector which is perpendicular to the surface. For each triangle $T$, we find the nearest joint $J$, we link them together to construct another dotted line which is used to measure the angle $\theta_1$. If the angle $\theta_1$ is around $180$ degree (see the figure in Figure 5-10.b), the triangle $T$ is segmented in the right group. Otherwise, if the angle $\theta_1$ is around $90$ degree (see the figure in Figure 5-10.a), which means the segmentation result is incorrect, such as the yellow circle region in Figure 5-10.c.

To eliminate the wrong segmentation, we set up the segmentation priority based on the following equation:

$$f_1 = aug(\cos(\theta_1)) \tag{5.4.4}$$

where $\theta_1$ is an angle between the normal of a triangle $\Gamma$, which is to be clustered into a mesh part, and the line which links $\Gamma$ to the nearest joint in the reference bone $l$, as shown in Figure 5-10.a.

### 5.4.2.2. The Local Boundary Smoothness in Each Segmentation Step

Our segmentation algorithm is performed on triangles. When we add a new triangle to one group, the boundary of the current group is modified due to the new added triangle. To reduce the irregularity of the final boundary, we introduced a new metric to evaluate the boundary angles for each triangle.

Every time when we add a new triangle to one group, we either modify the curvature of an existing boundary vertex, or we change the boundary curvature by introducing a new vertex. We notice that there are three types of curvature changes, which are shown in Figure 5-11. In Figure 5-11, the blue region indicates the existing mesh part. The triangle, which is marked as red, is the candidate triangle to be clustered. The orange triangles are the neighbors of red triangle, which have already been clustered into the blue region group. We use an angle to represent the curvature change on vertex $V$. When we add a new triangle, we check the angle for vertex $V$, which is marked as green point in Figure 5-11. There are three kinds of changes:

1. New vertex added (see Figure 5-11.a).
2. Curvature change on the existing vertex by merging one neighbor triangle (see Figure 5-11.b).

3. Curvature change on the existing vertex by merging two neighbor triangles (see Figure 5-11.c).

Based on the cosine value, we set up the segmentation priority for the triangles to be clustered. The triangle angle represents the smoothness of the boundary. Large angle indicates that the boundary is not sharp, which will finally result in smooth boundary after the segmentation process. The triangle containing the smallest value for the angle has the highest segmentation priority. To enlarge the effect, we make use of an augmentation function to set the priority value:

$$f_2 = aug(\cos(\theta)) \tag{5.4.5}$$

where $\theta$ is the angle of a boundary vertex v, which is shown in Figure 5-11.

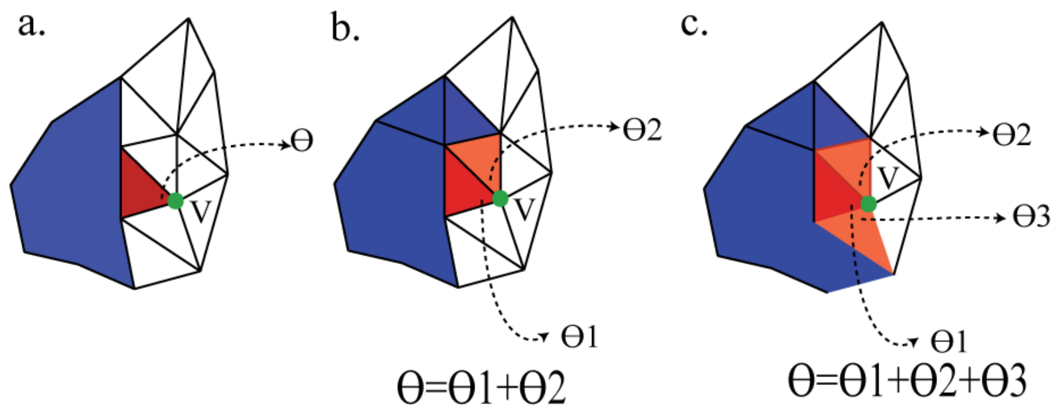

**Figure 5-11 Local Boundary Curvature Check**

Finally our segmentation metric is defined as:

$$d(\Gamma, l) = w_1 \int_{\Gamma} C ds + w_2 \int_{\Gamma} D_B(l) ds + w_3 f_1 + w_4 f_2 \tag{5.4.6}$$

where $C$ is a function to evaluate the curvature change along the tangent direction of the surface, which is guided by region growing direction. $D_B(l)$ is a function to measure the distance from the reference bone $l$, which helps determine the relevance of a mesh model vertex to a user sketched bone. $f_1$ and $f_2$ are two priory functions to determine a triangle $\Gamma$ to be clustered. $w_i$ [12] are application dependent weights to adjust the contribution of each function in Eq-(5.4.6), where $\sum_{i=1}^{4} w_i = 1$.

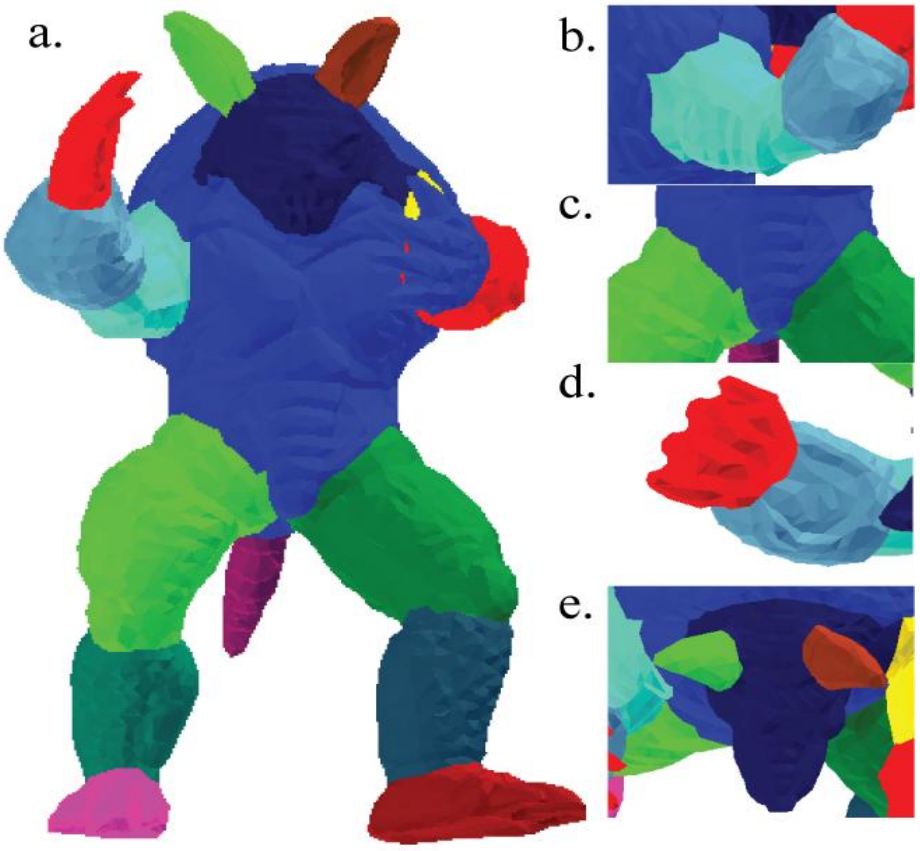

**Figure 5-12 Segmentation Result from Eq-(5.4.6)**

[12] In this experiment, our weights are adjusted as $w_1 = 0.4$, $w_2 = w_3 = w_4 = 0.2$.

Figure 5-12 is the segmentation result using our new segmentation metric as shown in Eq-(5.4.6). In the overview, Figure 5-12.a decomposes the monster model into meaningful pieces, especially in arm regions and the leg regions. Figure 5-12.b describes the segmentation details in the arm region, where no over-segmentation occurred. Figure 5-12.c describes the segmentation details in the leg region, where the segmentation boundaries isolate two legs in the right positions. We also do not have the over-segmentation issue in the hand region. The ear regions also have no over-segmentation issue as shown in Figure 5-12.e.

### 5.4.2.3. Segmentation Algorithm

In mesh segmentation, we adopt a multiple-region growing segmentation method[13], in which we apply both the reference bones and the seed triangles as mentioned in Chapter 4 as the seeds of the initial mesh parts.

We summarized the steps of our approach as follows:

**Input:** Triangular mesh $M$ , a set of reference bones $B_n$ .

**Output:** Segmented mesh, which has $n$ number of parts.

**Step 1:** For each triangle[14] $t$ which is associated with one reference bone $b_i$ , where $0 \leq i < n$, mark $t$ as selected and store in selected triangle seeds set $S$ . Also mark the rest of triangles in the mesh model as unselected.

[13] In our algorithm, the new metric from Eq-5.4.6 was used and the weighting scheme is the same as discussed in this equation.

[14] Each reference bone contains a set of triangles which are collected in the sketching process.

**Step 2:** Pick out the triangles from $S$, whose neighbors contains unselected triangles and store in a new set $S_e$. For each $t \in S_e$, store the neighbor triangles, which are unselected, in a set $S_c$.

**Step 3:** Based on our segmentation metric, we compute a distance for each triangle in $S_c$ and sort the triangles based on this distance.

**Step 4:** Pick the triangle $t$ which has the minimum distance and mark $t$ as selected. The segmentation distance is computed between $t$ and $t_{root}$. $t_{root}$ is the neighbor of $t$ which is marked as selected. We assign $t$ to the same segmentation set as $t_{root}$ belongs to. If $t$ has unselected neighbor, save $t$ in $S_e$ and compute the distance for $t$'s neighbors and insert its neighbors into $S_c$.

**Step 5:** Repeat **Step4**, until there is no unselected triangles left.

## 5.5. Experiment Results

Here are some segmentation results.

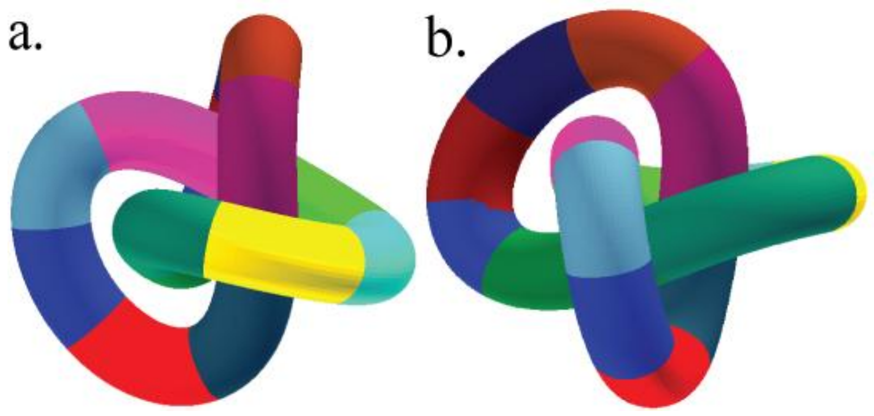

**Figure 5-13 Donut Tube Shape**

Figure 5-13 describes the advantage of our new segmentation method. Figure 5-13.a and Figure 5-13.b are the same model from two different views. Compared to methods which make use of curvature and geodesic distance to compute the segmentation, our method considered two extra terms (as shown in Eq-(5.4.4) and Eq-(5.4.5)) to perform the segmentation task. As you can see from Figure 5-13, the boundaries, which are directly obtained from our segmentation method without any extra smoothing operations, are reliable and smooth.

Even with complex pose model, our method still gives reliable result. Figure 5-17.a is the front view of our segmentation result, whereas Figure 5-17.b and Figure 5-17.c are the results captured from left and right side of the Woman model. The segmentation boundaries are fit into the right positions so that our skeleton maintains well-positioned bone structure. See the root of the arm regions in Figure 5-17.d and Figure 5-17.e and the waist region as shown in Figure 5-17.b and Figure 5-17.c. This is because the two new terms that we introduced in Eq-(5.4.6), controlled the segmentation process by checking both the segmentation boundary angles (see Eq-(5.4.5)) and the orientation of boundary triangles (see Eq-(5.4.4)).

To specify the difference between Eq-(5.4.1) and Eq-(5.4.6), we look deeply into the local details as shown in Figure 5-18. As we can see, the segmentation metric from Eq-(5.4.6) performs better than the one from Eq-(5.4.1).  This is because we have two major improvements in this metric. First, we add a new constraint to reduce the over-segmentation issue (see Eq-(5.4.4)) in some regions which have no clear topology changes. Second, we introduced a boundary smoothness constraint, which is used to reduce the irregularity of the segmentation boundary (see Eq-(5.4.5)). For example by monitoring the angle change (see Figure 5-10) between surface triangles and reference bone’s joints, we eliminate the over-segmentation issue (see the comparison in Figure 5-18.b(i) and Figure 5-18.b(ii)). We avoid the over-segmentation issue in the head region (see the comparison pictures in Figure 5-18.e(i) and Figure 5-18.e(ii)). Besides the angle constraints as illustrated in Eq-(5.4.4), we use a segmentation boundary angle constraint (see Figure 5-11) to limit the segmentation process, which also produces better segmentation results (see the comparison pictures in Figure 5-18.c(i) and c(ii) ).

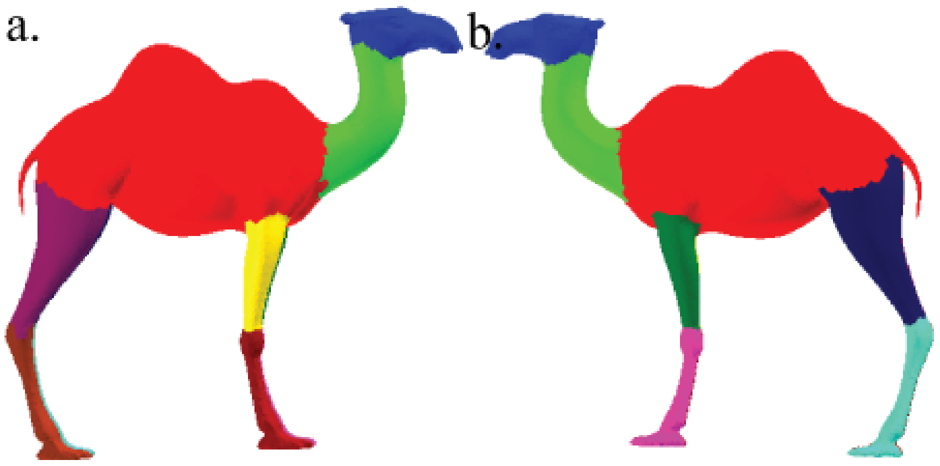

**Figure 5-14 Camel**

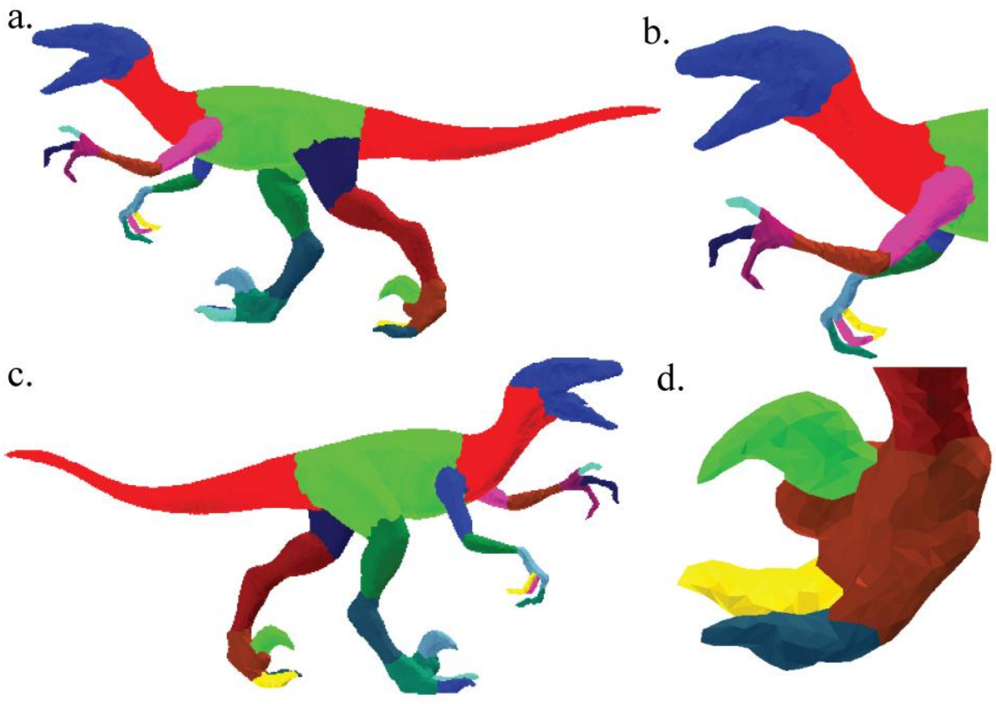

**Figure 5-15 Raptor**

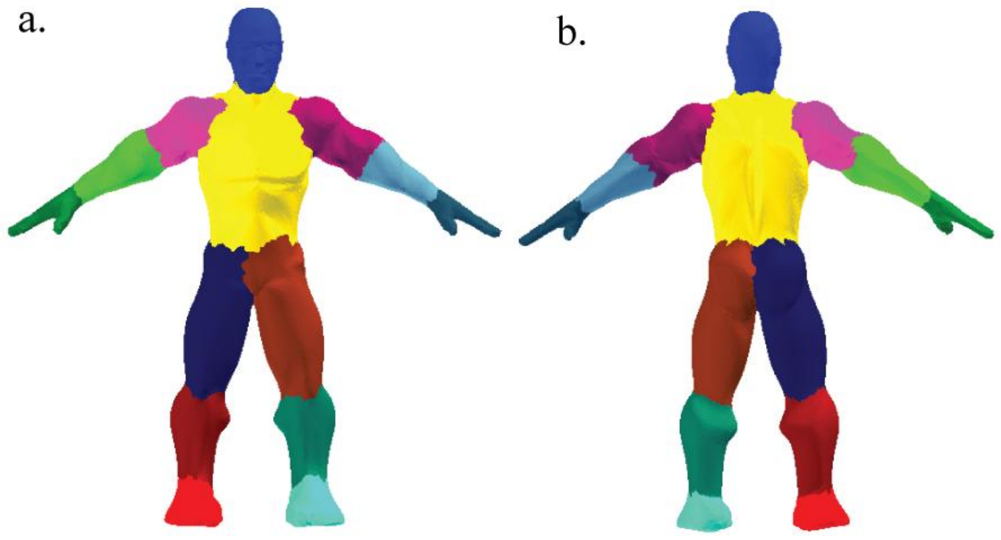

**Figure 5-16 Man**

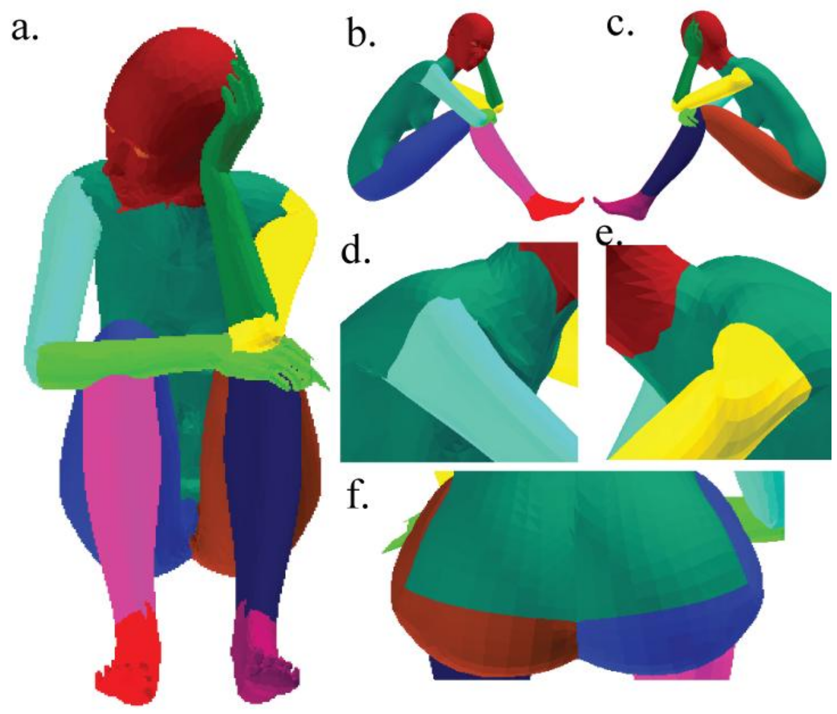

**Figure 5-17 Woman**

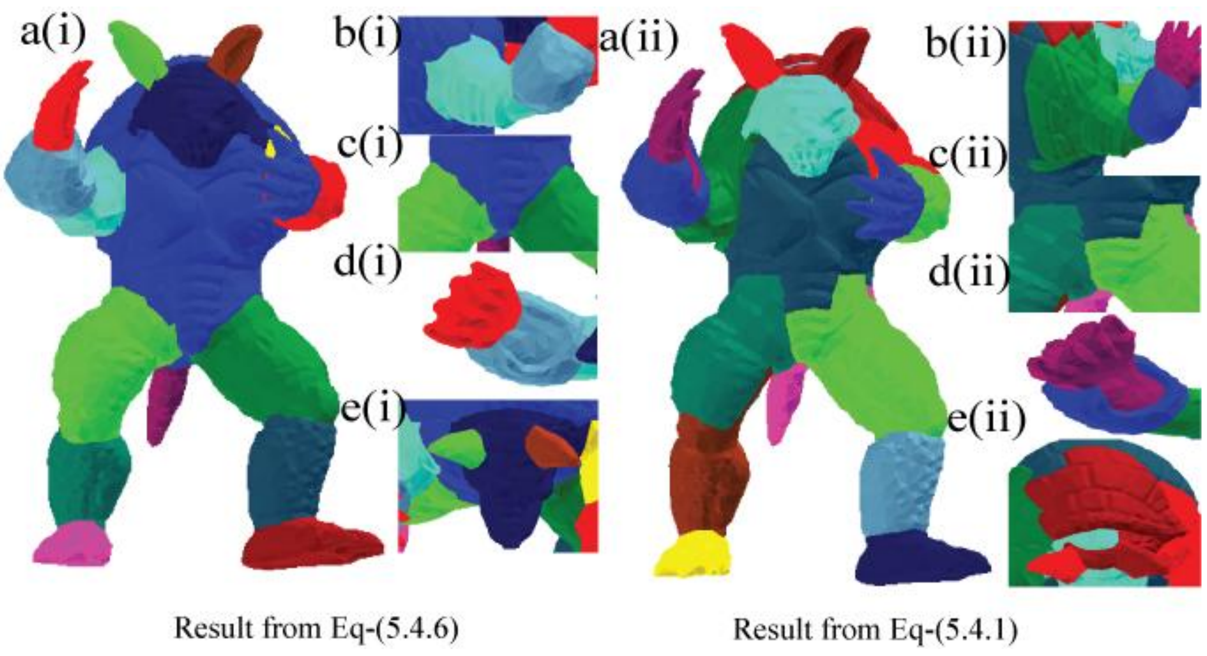

**Figure 5-18 Result Comparison of Eq-(5.4.1) and Eq-(5.4.6)**

## 5.6. Discussion and Summary

Our segmentation method plays an important role in our skeleton extraction system. From our segmentation process, we obtained two important factors which are closely related to our skeleton extraction. They are namely the segmentation boundaries and the segmentation parts. We make use of the segmentation boundaries to create well-positioned joints for our skeleton. Thus to guarantee the quality of our skeleton, we need to find a segmentation method which can produce reliable segmentation results. The data from segmentation parts is used to build the mapping relationship between bones and surface data in the skin mapping process, which also plays an important role in the animation related work.

Literally, there are many part-type segmentation methods, such as [65, 67, 89, 90, 100, 101]. However, there are some limitations for those existing methods, which prohibit us from using the existing methods directly. The K-means method [90] made use of accumulative distance to segment a mesh model into the predefined number $K$ parts, which cannot preserve the shape features. Attene [67] et al. tried to decompose mesh into parts based on some predefined primitives such as cylinder, plane or sphere etc. This method works in mesh parts which have large topology changes. However, it cannot guarantee the reliability of the boundary which is used to distinguish different parts. For example with regards to a leg region, it is easy to isolate the whole leg as a cylinder shape. However, because that they have similar shape features, it is hard to distinguish the difference between upper leg and lower leg.

Katz et al. [65] proposed a hierarchical decomposition algorithm which transfers the mesh vertices into a pose-insensitive representation using multidimensional scaling method. And then by extracting some feature points and the core part, Katz et al. decomposed a mesh model hierarchically into several parts. The advantage of transferring a mesh model into a pose-insensitive representation is that the result gives clear topology feature specification. However, such process also destroys the curvature details of the original mesh. The segmentation result cannot be guaranteed due to the distortion of those curvatures on the mesh surface.

Shapira et al. [101] proposed a "Shape Diameter Function" (SDF) to segment a mesh model, which was a measure of the diameter of a mesh model's volume in the neighborhood of a point on the surface. However, we notice that there are two limitations: the partition number problem and the segmentation problem. First, Shapira et al. made use of mathematical analysis to define the number of final parts for the decomposition which might be different from user's expectation. Second, the segmentation parts are affected by its part volume, such as one part which contains large volume change (i.e. legs like a cone shape) may be divided as two parts.

Lai et al. proposed the Random walk method [89] to segment a mesh model into a user specified number of parts. For a mesh model, a user needs to specify a number of faces as the initial seeds. Each seed defines a region that one part belongs to. For the rest of triangles, each triangle is assigned with a probability to the nearest seed. In [89], surface dihedral angle is used to determine such probability. The result of this method depends on the selection of seeds.

Rather than using some existing methods, we proposed a new solution. This is because we have two special requirements: reliable segmentation boundary and reliable components recognition. Most of them make use of surface geometry attributes, such as curvature [88, 99], geodesic distance [102, 103], planarity and normal direction [67, 81], to segment a mesh model. However, those segmentation methods, which rely on those surface geometry attributes only, may generate wrong segmentation results due to local surface feature changes or model pose changes.

By analyzing the possible factors which may have great impact in the segmentation process, we notice the basic four elements: the curvature, the distance constraint to the referenced bone, the boundary curvature and the segmentation orientation. According to the experiment, we show that the above four elements have great impact on the segmentation results.

Also we do find some limitations on this research work. First, the user's drawing will affect the segmentation result. This is because that our multiple-region growing segmentation method requires some pre-defined seeds. In our experiment, we adopt the selected triangles which come from the reference bone as the seeds. If the seeds are not defined correct, the result is not guaranteed. To avoid this, we have defined the alignment process which is described in Chapter 4. As we introduced a distance constraint, which is related to the reference bone, our segmentation results rely on the quality of this reference bone.

# Chapter 6

# 6. Skeleton Extraction

## 6.1. Introduction

Skeleton can be created either manually or automatically. In typical commercial modeling software, such as Maya or 3D studio Max, skeleton construction requires a user to define joints precisely and bones as well as their hierarchical relationship on an unfolded mesh model. Such task is time consuming and requires user to be trained. To alleviate this problem, research has been conducted to develop methods for automatic skeleton construction.

Our object is to create skeleton for a given mesh model, so that the skeleton can be used for animation purpose. We prefer to extract the skeleton from mesh model directly without modifying the mesh model. A skeleton should have well-positioned joints specification, in which the joint positions define the connection relationship between bones. The joint position will affect the quality of the mesh animation. For example given a leg model, if the knee joint position is not well-defined, the deformation result will contain large distortion due to the inappropriate joint position setting. However, most of the existing skeleton extraction methods produce skeleton in 1D curve format only, which have no joint information at all. Although they can specify joint positions manually after creating the skeleton, they have to use post-processing to map surface vertices to bones or joints. This mapping process requires large computation time.

To get the specification of joint positions, and the connection relationship among different mesh parts, we first segment a mesh model into different parts. If we can segment a mesh model into meaningful parts, we can obtain reasonable joint positions which are located in the center of the segmentation boundaries from each part. These joints also define a better skeleton structure with regards to the joint positions. Given the mesh parts that we have obtained from our mesh segmentation algorithm which has been discussed in Chapter 5, we use the LSD method (see discussion in section **2.4.3**) to create bones for each part of the mesh model. After extracting the bones from each mesh part, we connect them together to construct the final skeleton.

## 6.2. Skeleton Extraction in Our Work

### 6.2.1. Method Overview

Our skeleton extraction system contains three major steps:

1. Bones extraction operation
2. Bones smoothing operation
3. Mapping between bones and mesh model

We propose a LSD algorithm to extract bones from each mesh part which is obtained after the segmentation process. The center of a segmentation boundary is the starting point of a bone. Starting from this central point, we compute the geometric distance from all vertices in the mesh part to the central point. We sample the longest distance with a predefined number $n$ that defines the number of joints within the extracted bone, reflecting the quality of topology features for an extracted bone. After the sampling process, we have $n$ groups of surface vertices. We create a joint for each surface vertices group. The joint is central point of a surface vertices group. We also build up the initial mapping relationship between joint and surface vertices group from the sampling process. Each joint is mapped with the surface vertices group which is used to compute the joint itself. We will give full description in section 6.2.2. The bones that we extracted from each part contain distortions. This is because that the skeleton that our bone extraction method is highly sensitive to the surface details. We proposed a line smoothing algorithm to remove the distortions from the surface details.

Besides the bone smoothing operations, we also introduce a mapping process to reconstruct the mapping relationship between joints and the mesh model. The existing mapping relationship between joints and mesh model is inaccurate due to the surface details distortions.

## 6.2.2. Bone Extraction Methodology

Our bone structure is the same as the reference bone structure which is discussed in section 4.2.3. The bone structure is defined as:

$$B = \{J_h, plist, J_t\} \tag{6.2.1}$$

where $J_h$ and $J_t$ are two well-positioned joints and $plist$ contains a set of ordered nodes. The two well-positioned joints $J_h$ and $J_t$ are used to connect more than one bone, whereas the nodes in $plist$ are used to describe the geometric positions for different parts of a bone. The joints and nodes in a bone are similar in the data structure, which both contain a set of surface vertices. The major difference between joint and node is that a joint is used to connect one or more bones whereas a node is used to describe the geometric position for part of a bone.

The joints ($J_h, J_t$) are located at the two ends of a bone, which are used to connect more than one bone. We call those joints as tip-joints. The tip-joints are important in skeleton-driven animations. Many people make use of this kind of joints to compute rotation or translation matrix. The nodes are located inside the $plist$ of a bone, which are used to represent the geometric positions for different parts of a bone.

After mesh segmentation, the mesh model is divided into semantically meaningful segments. We can now extract a bone from one mesh part in two steps:

1. Identify the segment boundaries of each mesh part. According to the number of boundaries found, we determine the type of each mesh part. Note that a segment boundary is a boundary between two connected mesh parts.
2. Compute a LSD for each segment boundary of a mesh part and extract the medial axis from the LSD of a mesh part to form one bone. If there is more than one segment boundary within a mesh part, we need to merge those bones from all the boundaries together and construct the skeleton structure.

There are three main operations involved in the above two steps. They are explained as follows:

**Mesh part types**: We define three types of mesh parts. As shown in Figure 6-1, a 1-end, 2-end and multiple-end mesh parts comprise one, two or multiple segment open-end boundaries, respectively.

In order to determine the number of boundaries attached to a mesh part, all triangles forming the mesh part are checked, such that triangles that connect to either an open-end or another mesh part are isolated. Figure 6-2 shows the segment and open-end boundaries obtained from the human model used in our experiment. The grouping operation explicitly tells the number of boundaries that a mesh part has, as well as the types of these boundaries. Note that the implementation of the above process is easy and it is efficient, as when we segment the input mesh model, we store an identifier to each mesh model triangle to indicate which mesh part that it will be assigned to. Hence, the whole process becomes a simple table look up and a grouping operation.

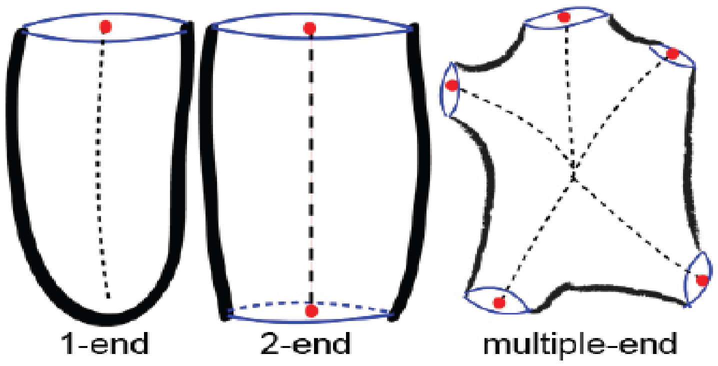

**Figure 6-1 Mesh Part Types**

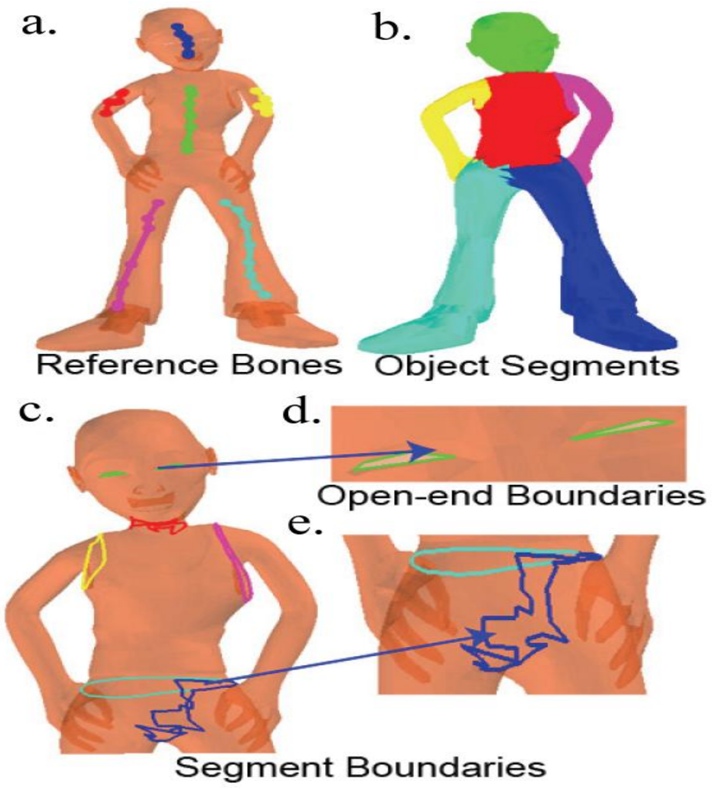

**Figure 6-2 Segmentation Boundaries**

Also to avoid the distortion from the mesh model itself, we check the perimeter of an open-end boundary against a pre-defined threshold. If the perimeter is smaller than a given threshold, we will ignore this boundary. If we do not eliminate small open-end boundaries, the topology structure will be affected. Figure 6-3 describes the difference in the skeleton extraction. Figure

6-3.a describes the skeleton result which does not consider the open-end boundary issue. Figure 6-3.b is the bone structure for the head part. There are two extra bones within the head part. This is because the eyes regions have two open-end boundaries. During the skeleton extraction process, those boundaries are considered as starting point of bones. Figure 6-3.d is another skeleton result, which ignored the open-end boundaries. Figure 6-3.c is the head region, which has a clear topology feature for head part.

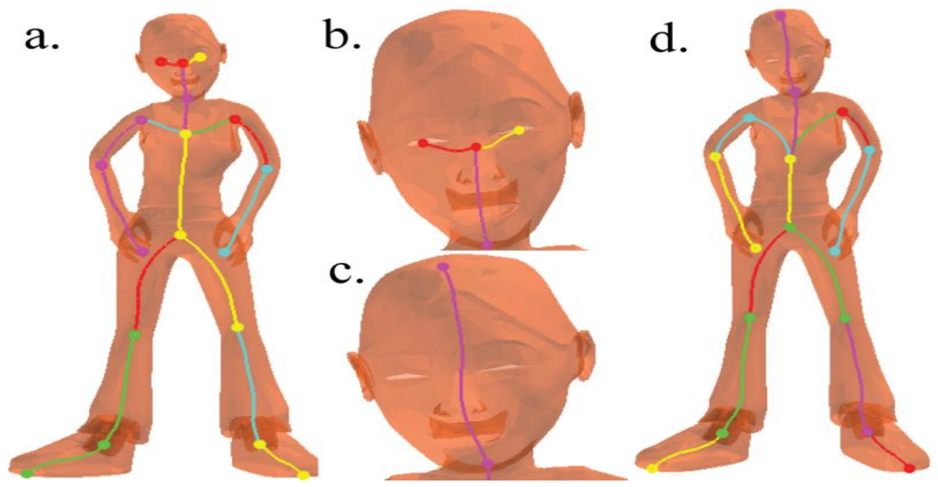

**Figure 6-3 Open-end Boundary Problem**

**Bones**: To extract bones from a mesh part, a source point is first assigned to the center position of each segment boundary shown in Figure 6-1. Based on this source point, we build up a set of contour lines by evaluating the shortest distance of each mesh vertex from the source point as shown in Figure 6-4. We use the shortest distance to obtain the LSDs because it is insensitive to noise or fine local mesh features but can describe the topological structure of a mesh part well. The LSDs generated from source points of different mesh part types are illustrated by the contour lines as shown in Figure 6-4. We then take the medial axis of each contour line (depicted by the dotted color lines in Figure 6-4) to form a skeleton bone, which comprises a list of connected vertices.

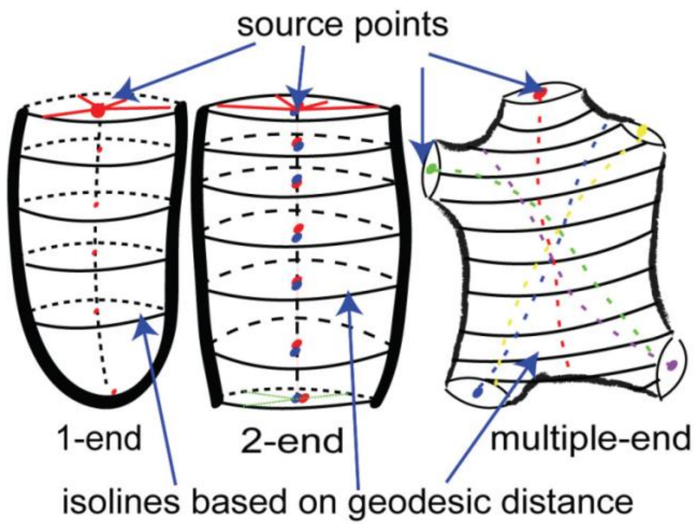

**Figure 6-4 LSD Method**

**Junction joint**: Multiple bones may be extracted from a multiple-end mesh part. We create a joint at the junction joint of the bones, which has the minimum distance to all the bones, to connect all the bones within the mesh part. To connect a skeleton bone to this joint, we assign the segment of the bone, which is located between the source point of the skeleton bone and the junction joint, to this joint, and remove the unassigned bone segment. After we have connected all bones of the mesh part to the junction joint, we obtain a net of bones, where the joint forms the center of the net.

### 6.2.3. Skeleton Construction

To construct the final skeleton, we connect bones extracted from all mesh parts together into a single structure. There are three major steps involved:

1. Optimizing and smooth bones
2. Generating joints and nodes
3. Forming the skeleton structure

We use the same technique as we did in section 4.2 to smooth the bones. The joints are those final joints which connect the bones into graph.

**Optimizing and smoothing bones**: As the bones are evaluated based on the mesh geometry of a mesh model, it is unavoidable for a skeleton bone to be distorted by local mesh features of its corresponding mesh part. We address this problem in two stages. First, we applied the line smoothing operation, which is similar to the one that we have adopted in section 4.2.3, to eliminate the distortions caused by minor local mesh features. The difference between this smoothing algorithm and the one in section 4.2.3 is that we add one constraint to limit the smoothing operations. Each joint/node is associated with one closed contour line (see Figure 6-4). So during the smoothing operation, we need to make sure that the position of a joint/node after bone smoothing should not be relocated outside of the mesh model. The reason to add such a constraint is that the bone should be embedded inside a mesh model. This is a kind of centeredness requirement for a bone. An example of such an effect is shown in Figure 6-5. Figure 6-5.a(i) shows this problem while Figure 6-5.a(ii) shows the result of the smoothing operation. Figure 6-5.c(i) describes the extracted bone which contains large distortions. The main reason is that the LSD based method is sensitive to surface details (see the ankle region of the foot in Figure 6-5.c(i)). After the smoothing operation, we obtain a smooth bone in which the distortions are eliminated (see Figure 6-5.a(ii)). Second, we detect major curvature changes of a skeleton bone as mentioned in section 4.3 to partition the skeleton bone into topologically separate bones and insert new joints between each pair of these new bones. After the above operations, the resultant bones will be more suitable for use in animation.

**Generating joints**: In our work, we have two different types of joints: tip-joint and node. A tip-joint is located at one end of a skeleton bone which is used to connect one or more bones. It corresponds to the source point that we have generated for each segment boundary on a mesh part.

A node is located inside a mesh part, which is used to describe the geometric position for part of a bone.

**Forming the skeleton structure**: we connect bones from different mesh parts at their corresponding joints to form the final skeleton. Figure 6-6 illustrates the whole process for skeleton construction. We first allow the user to sketch bones on the input human model. As shown in Figure 6-6.a, the sketched bones can be very rough. The model is then divided into mesh parts, and bones and joints are generated. By connecting them, we obtain the final skeleton. Note that skeleton joint generation is still an open problem. In our work, we adopt user sketching and employ our modified isophotic metric. It allows us not only to generate joints at typical model traits, such as joint 1 in Figure 6-6.c, where they can be identified by typical minima rules and part salience rules. We can also generate semantically meaningful joints (e.g., joint2 at a knee of the human model), where those rules cannot help.

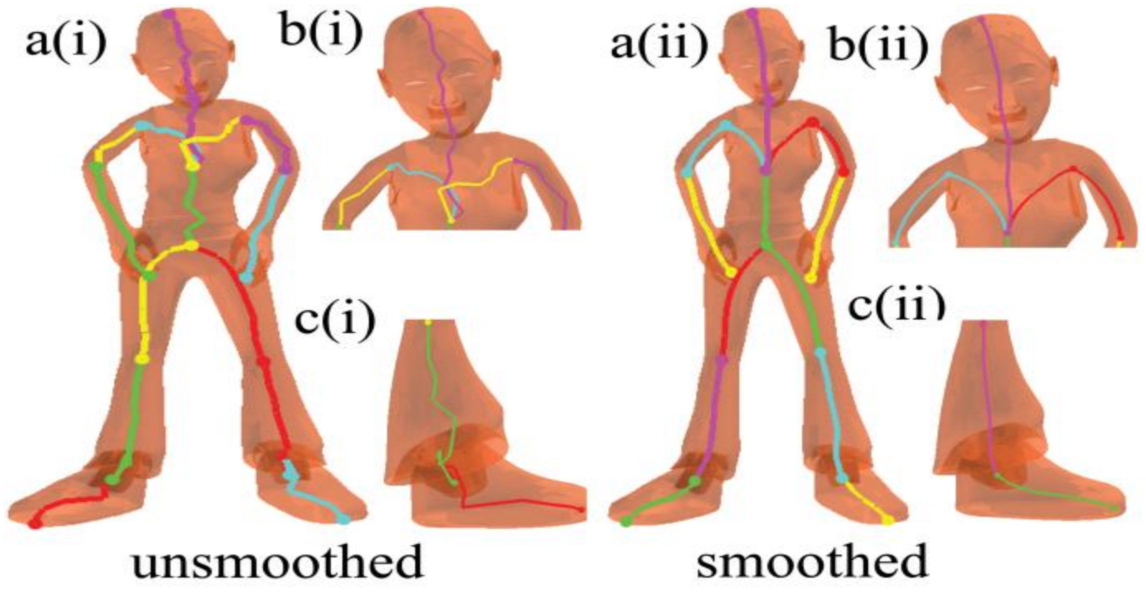

**Figure 6-5 Skeleton Smoothing**

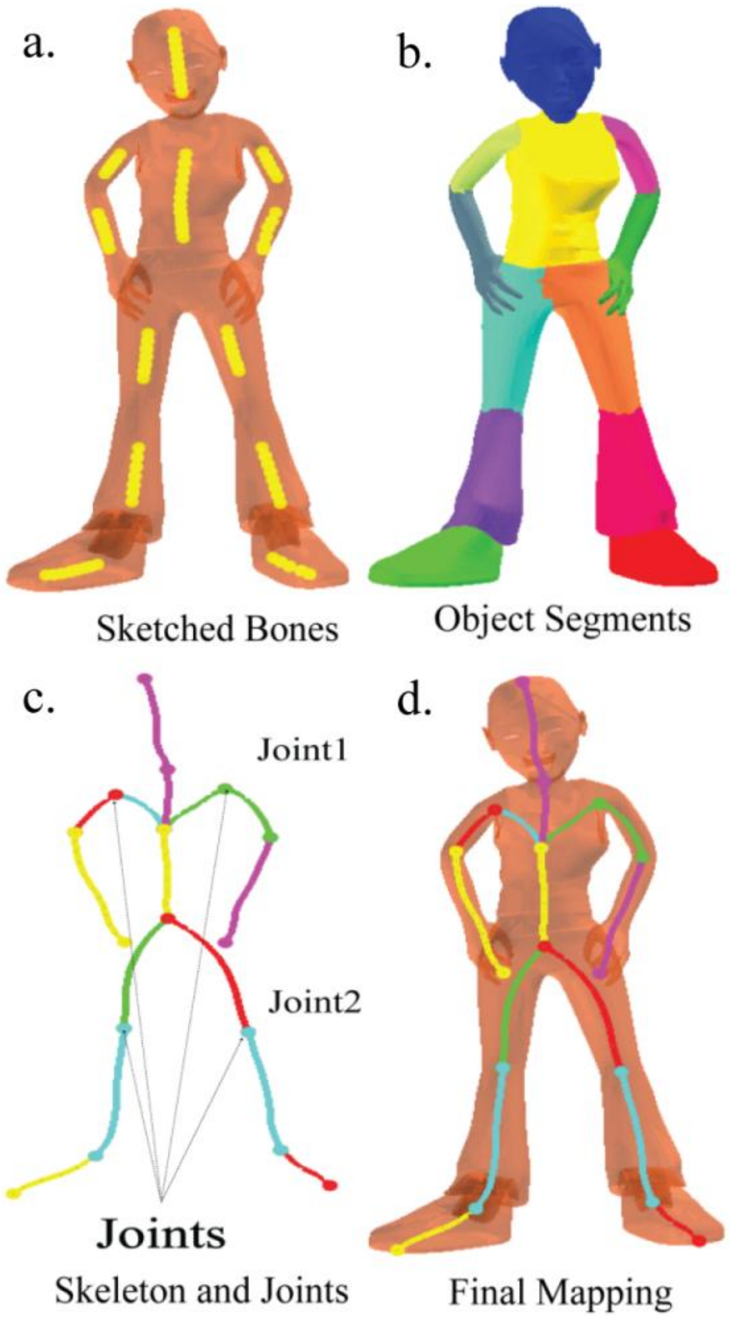

**Figure 6-6 Skeleton Construction Process**

### 6.2.4. Skin Mapping

During the bone extraction process, we have set up the mapping relationship between joints/nodes and surface vertices within a mesh part. In our bone extraction process, the mapping relationship is defined from each segmentation boundary, and the geometric distance to the boundaries, which is affected by the local features. The mapping relationship between a bone and the surface vertices are defined as a collection of all the joint/node mapping relationship within a bone. In multiple-end part, the mapping relationship between bones and the mesh model contains errors. This is because each bone is mapped with the whole multiple-end part, which may result in the wrong mapping relationship. We also used a combination step to merge skeletons which lie in the same mesh part together, which further increased the mapping problem. We make use of

Figure 6-7 to describe the mapping problem in multiple-end part.

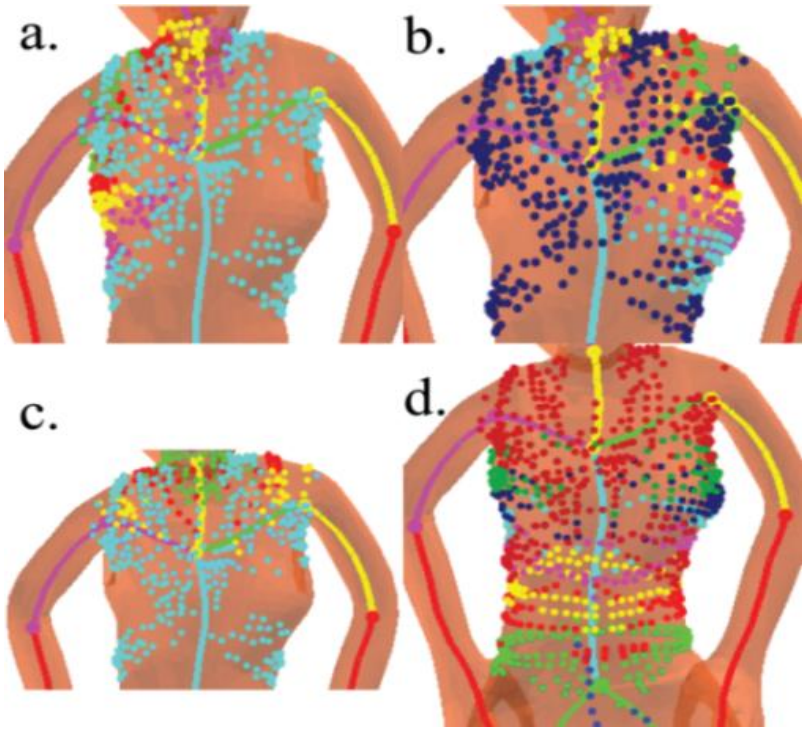

Figure 6-7 Skin Mapping in Multiple-end Part

In Figure 6-7, we have four pictures to describe the skin mapping result within a multiple-end type mesh part. Each picture describes one skin mapping result between one bone and the mesh part. Different colors are used to indicate the skin mapping results for joints/nodes within a bone and surface vertices. Those points with different colors are the surface vertices. The surface vertices, which have the same color, belong to a single joint/node within a bone. Figure 6-7.a and Figure 6-7.b describe the skin mapping results for a purple bone and a green bone within a multipe-end type mesh part respectively, whereas Figure 6-7.c and Figure 6-7.d describe the skin mapping results for the yellow bone and the light blue bone respectively. However, the skinning mapping result between a bone and the mesh part is incomplete. Especially, in Figure 6-7.a and Figure 6-7.b and Figure 6-7.c, there are many light/dark blue vertices which have the same color and distributed everywhere over a mesh part (the same happens to the red vertices in Figure 6-7.d). This is because that those vertices are mapped to the junction joint inside the mesh part. This junction joint connects four bones, and the vertices come from all those four bones.

Those vertices with the same color reflect the skin mapping result between a junction joint and the surface vertices. For a multiple-end type mesh part, when we merge all the bones together after the bone extraction process, we also merged the mapping results for the junction joint. That is why even in this dispersed situation, those vertices are still belonging to one joint only. Obviously, it is not correct. This is because that after extracting bones from a multiple-end type mesh part, we have to combine all bones within this mesh part, in which we have to find out a junction joint to connect all bones together and remove the bone segment between the junction joint and the end joint $J_e$ of a bone $B$. When we create a junction joint, we need to merge all the skin mapping results from a joint which has the minimum distance to this junction joint for each bone inside a mesh part, which result in the wrong mapping results for this junction joint. To address this problem, we make use of the Euclidean distance to reset the mapping relationship between bones and surface vertices. Our solution is designed as follows:

1. We collect all the joints/nodes within a mesh part and stored in a list $js[n]$, where $n$ is the number of joints/nodes inside the mesh part. And we collect all the vertices within a

mesh part and stored in a list $vs[m]$, where $m$ is the number of vertices inside the mesh part.

2. For each vertex $v_i \in vs[m]$, we compute the distance $d_{ij} = |v_i - J_j|$, where $J_j$ is a joint/node in $js[n]$ and $0 \le j < n$, $0 \le i < m$ and $m$ is the total number of vertices within this mesh part. We store the joint/node $J_j$, which has the minimum distance to $v_i$, for vertex $v_i$.

We have discussed the mapping problem in multiple-end type mesh part. There is no mapping problem in one-end type mesh part. This is because that there is no junction regions in one-end type mesh part. There is a tiny mapping problem in two-end type mesh part, which is occurred in the junction joint region (remember that we need to merge bones from all the segmentation boundaries to construct a net of bones.). The time complexity of the skin mapping process is $\mathrm{O}(m \cdot n)$. To speed up the computing, we make use of multiple-thread technique to compute the distance for each vertex $v_i$ within one mesh part. For each vertex within a mesh part, we need to identify the nearest joint/node, which is used to set up the mapping relationship. The computation process can be performed simultaneously, which gives us the hint to use multiple-thread technique to speed up the mapping process.

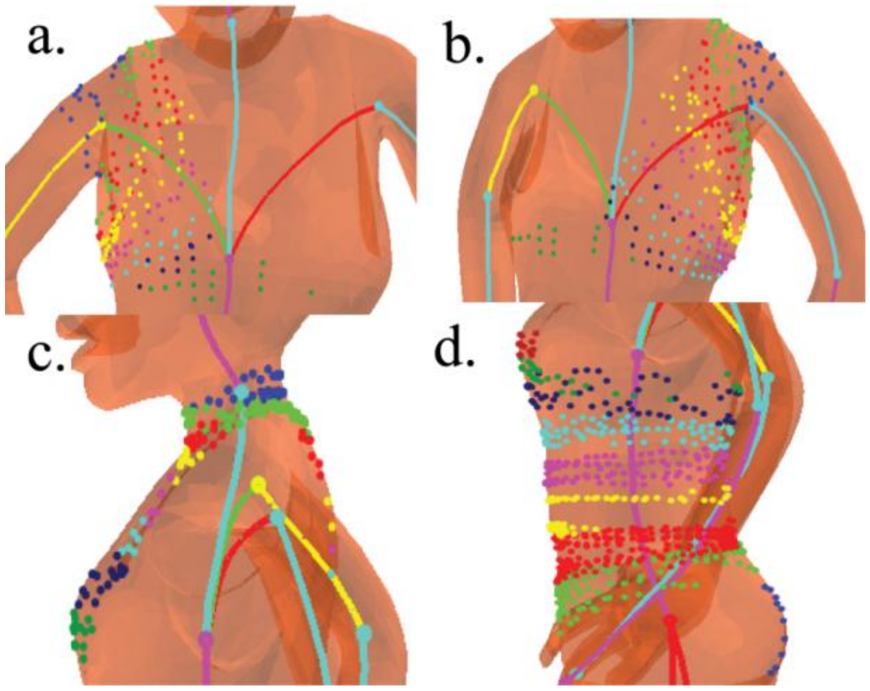

**Figure 6-8 Skin Mapping Result**

Figure 6-8 describes the mapping result of our method. Figure 6-8.a and Figure 6-8.b describe the green bone mapping results and red bone mapping results respectively. Figure 6-8.c and Figure 6-8.d describe the light blue bone mapping result and purple bone mapping result within this multiple-end type mesh part. We make use of different color to represent the skin mapping results for joints/nodes within a bone. Each color represents one mapping relationship between a set of surface vertices and a joint/node inside a bone.

Traditionally the skin mapping is defined by the mapping relationship between surface vertices and bones inside a skeleton. Each bone is associated with a group of surface vertices from the mesh model. Each vertex can also be associated with multiple bones and have a blend weight for each bone. The easiest way to compute such weights is by computing the distance between a surface vertex and its associated bones. And based on the distance to its associated bones, one can set the weights for each bone. When the bones are transformed, each bone transformation is applied to the vertex position, scaled by its corresponding weights. This kind of algorithm is also called matrix palette skinning, because a set of bone transformations form a palette for the vertex to choose from. However, such kind of mapping cannot be used to achieve soft bending effect, such as the snake movement for tails.

To obtain robust control on skeleton-driven animations, we make use of mapping between joints/nodes and surface vertices instead of mapping between bones and surface vertices. We limit the number of vertices which have multiple bone association by using the joint/node mapping instead of bone mapping. In our skin mapping design, each surface vertex is associated with one joint/node only. However, each joint may connected to one bone or more bones. We can still perform matrix palette skinning for those vertices which are connected with more than one bone. For the rest of vertices, each vertex is only associated with one joint/node which belongs to one bone only. Thus we reduced the computation of weight for each vertex in the matrix palette skinning algorithm. This is one advantage of our skin mapping method.

Also the existing matrix palette skinning cannot be used to achieve realist muscle movements, skin motions or soft bending effects for special bones such as animal tail movement. In our skin mapping design, we could achieve robust control over a bone by assigning a transformation matrix for each joint/node. Thus the LBS equation is changed to:

$$p' = \sum_{j=1}^{n} w_j p M_j \tag{6.2.2}$$

where $n$ is the number of neighbors for node $j_p$ ,which is the associated joint of vertex $p$ . For a node inside the $plist$ of a bone, there are only two neighbor nodes: the succeed node and the predecessor node. $w_j$ is the weight for each neighbor joint and $M_j$ is the rotation matrix for each neighbor joint.

## 6.3. Experiment Results and Discussions

We use different models to test the quality of our skeleton extraction algorithm. In the experiment, we test our algorithm using 13 different models which have different topology structures. The results show both the advantages and disadvantages of method. We divide our results into three groups to demonstrate different properties of our method.

**Table 6-1 Model Groups**

| | |
|---|---|
| Group1 (perfect results) | Donut, Rhino, Frog , Man, Girl, Raptor, Hand |
| Group2 (results contain minor errors) | Dog, Camel, Woman and Armadillo Monster |
| Group3 (bad results) | Dragon and Stanford Bunny |

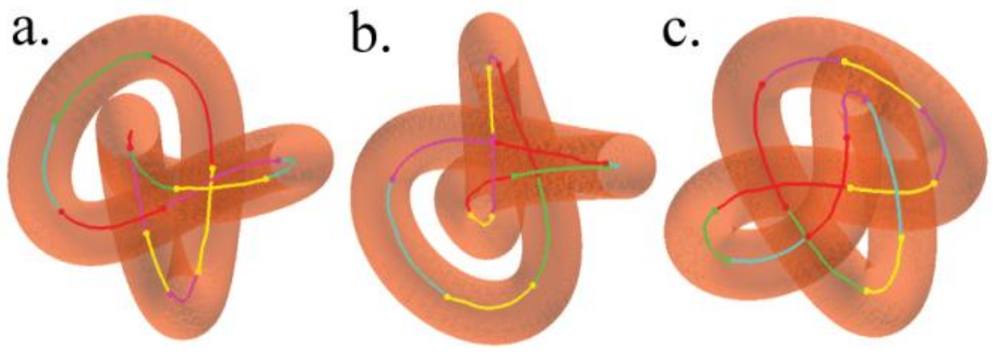

**Figure 6-9 3D Donut**

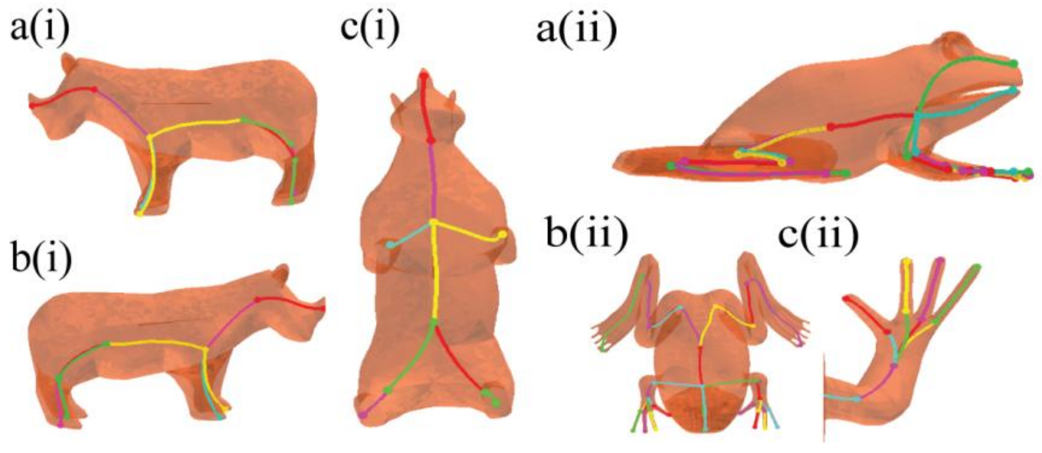

**Figure 6-10 Rhino & Frog**

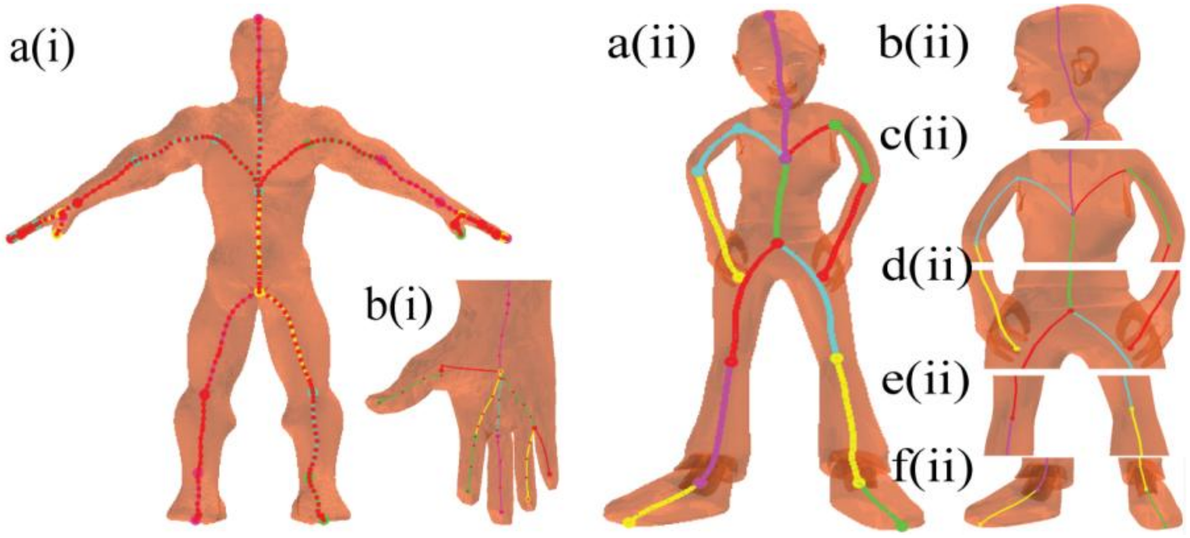

**Figure 6-11 Man & Girl**

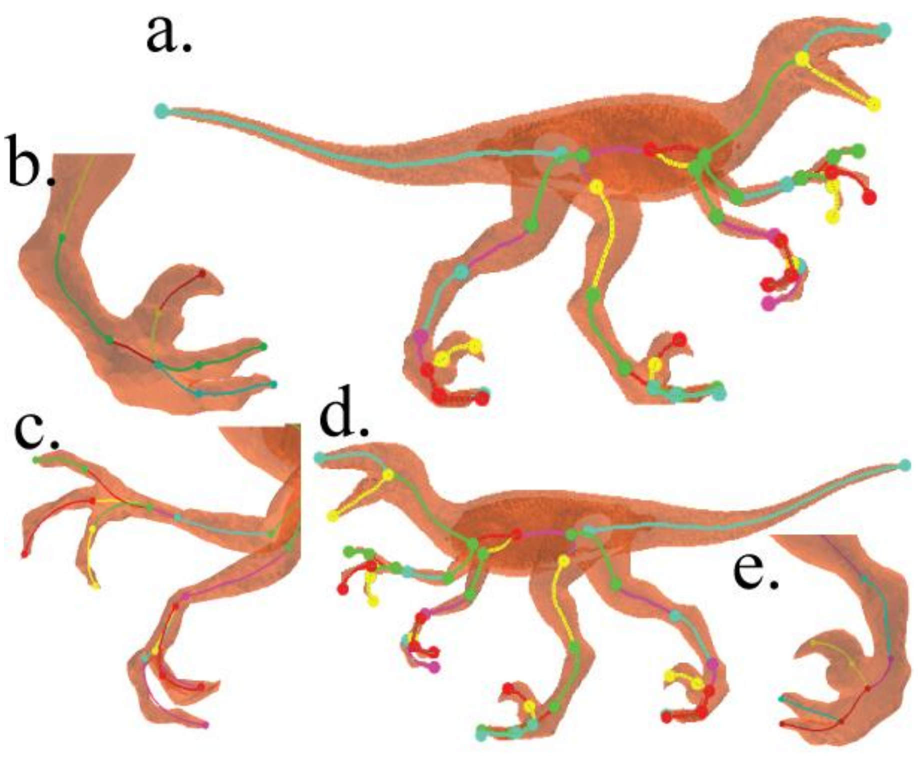

**Figure 6-12 Raptor**

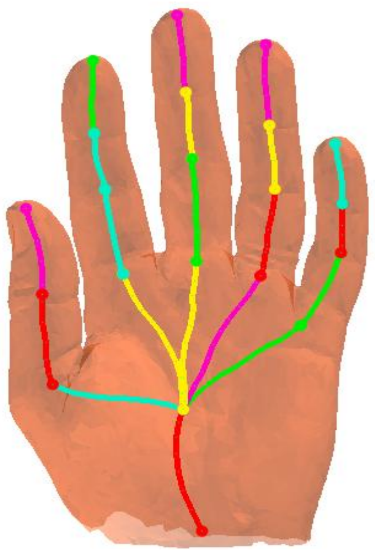

**Figure 6-13 Hand**

The above 7 models belong to group 1, which all give good skeleton results. For a mesh model which has articulated structure (i.e. human body), our method can produce skeleton which has well-positioned joint specification, such as the skeleton in Figure 6-11, both Figure 6-11.a(i) and Figure 6-11.a(ii) have meaningful well-positioned joints. Especially, we make use of Figure 6-13 to illustrate the advantage of our skeleton generation method. Most of the existing work can generate 1D curve skeletons for a hand without joint specification, but our method can generate well-positioned joints in a hand skeleton. The joint positions are specified from the boundaries of our segmentation result. By considering the surface curvature change, user's guidance and region boundaries'[15] curvature changes, we obtained such reasonable segmentation boundaries to define our joint positions.

[15] Our segmentation method is multiple-region growing method; each region contains one or more boundaries.

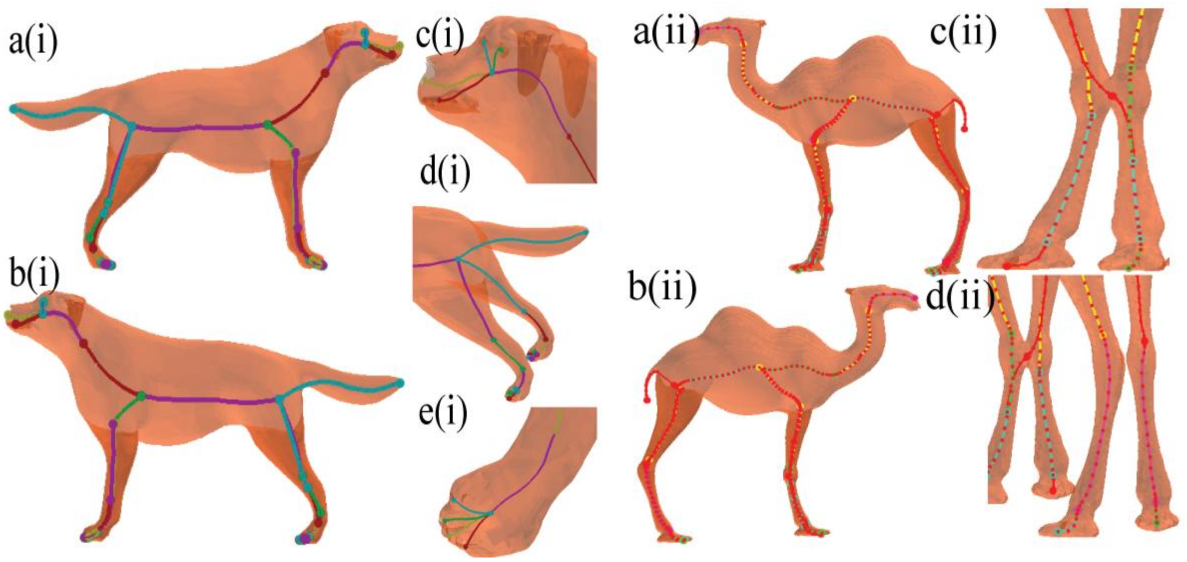

**Figure 6-14 Dog & Camel**

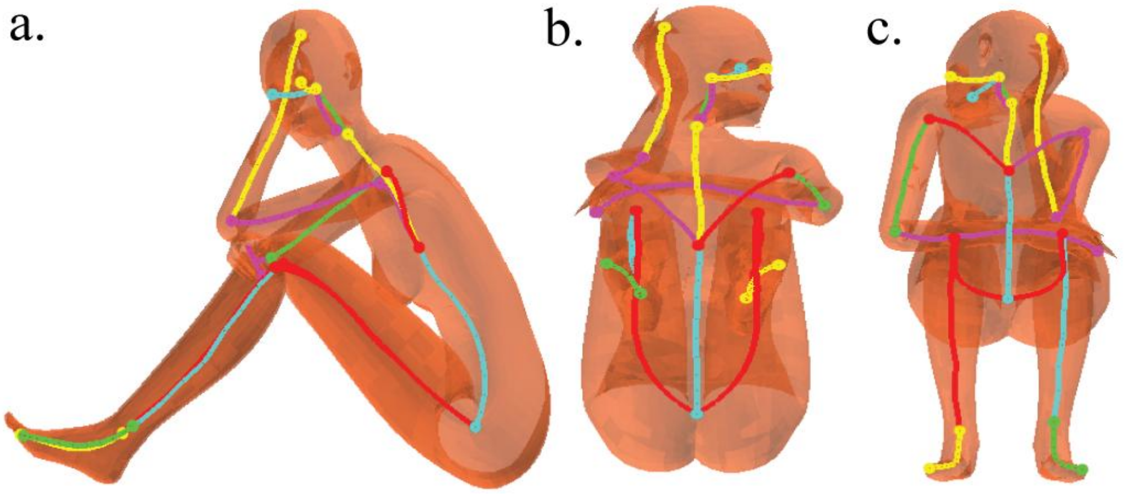

**Figure 6-15 Woman**

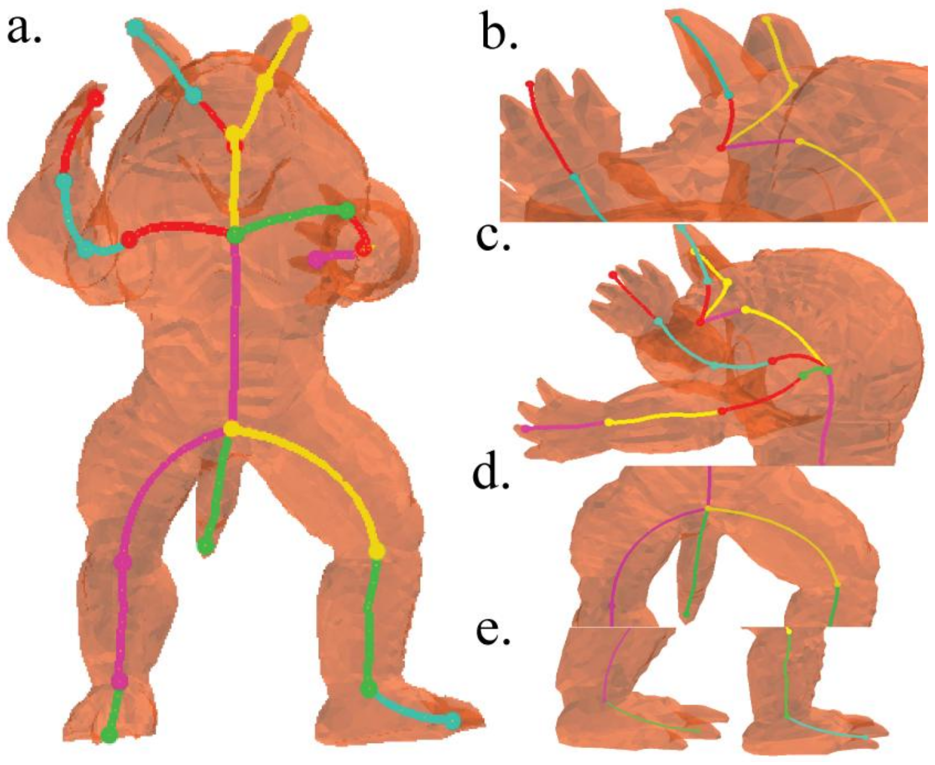

**Figure 6-16 Armadillo Monster**

The above four models belong to the second group. Each model contains a minor error in the skeleton structure, such as the multiple-branch error in the head or foot region of the dog model (knee region of the camel model) in Figure 6-14, or the head region of the woman model in Figure 6-15 or the missing topology feature in the head region of the Armadillo model in Figure 6-16. By analyzing those minor errors, we notice that there are two problems which are not solved in our skeleton extraction system.

First, our method is sensitive to the mesh model structure. If a mesh model has open-end boundaries, our method will construct a skeleton from those boundaries. This feature has both advantages and disadvantages. Figure 6-13 shows an example of the advantage of considering open-end boundary as guidance in our skeleton extraction. By considering the open-end boundary at the bottom of palm region, we constructed an extra skeleton (the red line in the palm region of Figure 6-13) to specify the full topology structure of the hand model. However, some of the open-end boundary may destroy the skeleton structure, such as the loop problem in Figure 6-17-dragon.b.

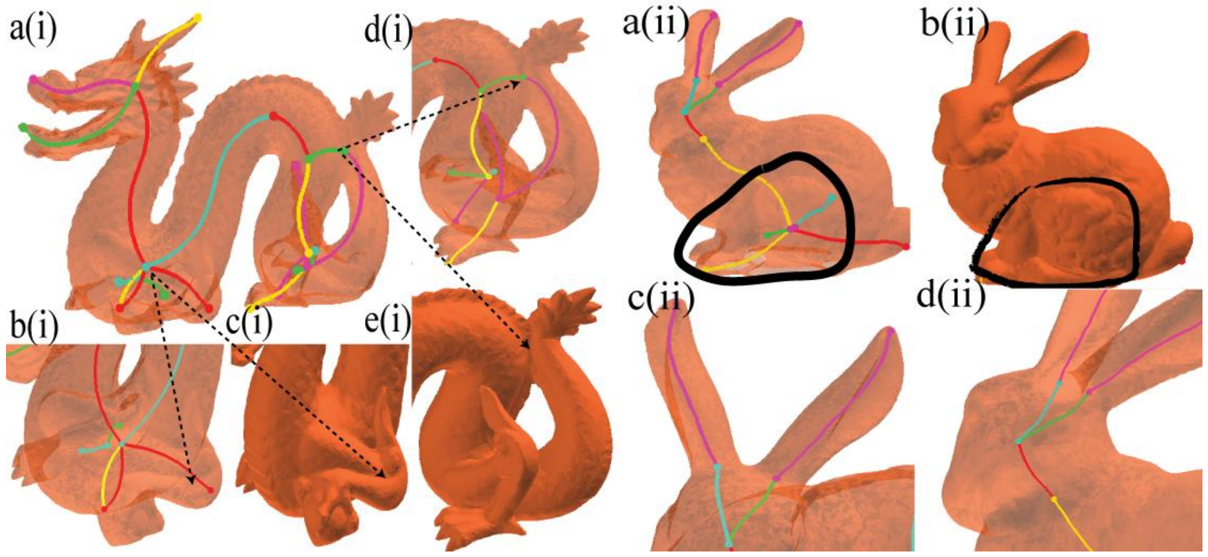

**Figure 6-17 Dragon & Stanford Bunny**

Second, for some parts which have complex shapes, our skeleton extraction method may ignore some major topology feature. The head region of Figure 6-16 demonstrates this problem. As you can, the topology feature for nose part is ignored in the Armadillo Monster model.

The above two pictures in Figure 6-17 are the last group, which illustrate another problem that our skeleton extraction method cannot handle. Our method cannot create correct skeleton structure for any model which contains squeezed regions, such as the dragon leg region in Figure 6-17.b(i), or the main body region in the Stanford Bunny model in Figure 6-17.b(ii). This is a limitation of using boundary to extract the skeleton structure. In the future, we need to solve this problem.

**Table 6-2 Average Running Time In Different Models**

| Model Name | Vertex Size | Triangle Size | Bone Size | Segmentation Time (sec.) | Skeleton Extraction Time (sec.) | Skinning Redistribution Time (sec.) |
|---|---|---|---|---|---|---|
| 3D Donut | 23040 | 46080 | 11 | 20.20 | 34.61 | 275.77 |
| Rhino | 2835 | 6012 | 7 | 2.81 | 2.70 | 11.12 |
| Frog | 17480 | 34956 | 23 | 12.21 | 20.23 | 198.05 |
| Man | 20000 | 39994 | 14 | 14.65 | 28.33 | 195.84 |
| Girl | 6012 | 12016 | 12 | 4.31 | 7.12 | 36.92 |
| Raptor | 9882 | 19604 | 27 | 11.32 | 16.44 | 86.32 |
| Hand | 4976 | 9884 | 15 | 3.59 | 10.22 | 33.55 |
| Dog | 10561 | 20808 | 16 | 7.55 | 25.33 | 88.04 |
| Camel | 16984 | 33964 | 16 | 11.70 | 27.88 | 160.83 |
| Woman | 7578 | 14999 | 12 | 5.55 | 13.07 | 47.61 |
| Monster | 5229 | 10454 | 15 | 3.83 | 6.08 | 29.02 |

Table 6-2 describes the run time for segmentation, skeleton extraction and skinning redistribution steps in different models. All the experiments are performed on a Desktop PC with Pentium 4 CPU (3.0 GHZ) and 2 GB RAM.

## 6.4. Discussion in the Skeleton Evaluation

### 6.4.1. Related Work

Most of the existing skeleton extraction methods focus on 1D curve skeleton generation. We focus specifically on well-positioned skeleton extraction which is used to perform animation tasks. The skeletons are used as manipulators to create different animation postures. In this case, we want to evaluate the quality of the skeletons for the animation related usage.

Cornea et al. analyzed the properties of curve skeletons in [50]. The major contribution of that work is the systematic analysis of the existing skeleton extraction methods, where they give a taxonomy of those methods based on a list of properties from many existing applications. The properties summarized by Cornea et al. are listed as: "*Homotopic, invariant under isometric, transformations, reconstruction, thinness, centeredness, reliability, smoothness, component-wise* *differentiation, robustness, efficient to compute and hierarchic*.” These properties are semantically summarized from different applications.

We did not consider all those properties in our skeleton evaluation criteria. This is because that some of the criteria only suitable for discrete 3D model based process, such as the reconstruction, reliable and thinning properties; some of the features have no problem in surface mesh based skeleton extraction methods. For example by using LSD method with geometric distance computation, our skeleton is always invariant under isometric transformations. We summarized the importance factors for a skeleton as follows:

- Joints location and its associated boundaries
- Centeredness of bones
- Smoothness of bones
- Skin mapping

**Joints location and its associated boundaries**: This aspect is inspired by the skeleton result from [86] (see Figure 6-18). This property refers to the correctness of the segmentation result of a mesh model. The segmentation boundaries define the joint positions of bones. However, some of mesh boundaries are affected by local curvatures (as shown in Figure 6-18.b). To get well-positioned joint positions, we have to improve the quality of our segmentation method. We have done the improvement in section 5.4.2.

**Centeredness of bones:** A bone does not need to be located exactly in the center of a mesh parts. MAT method can guarantee the centeredness of a bone. But such a method also suffers from the distortion of mesh surface details. By using LSD method, we also guarantee the centeredness of the bones. However, the smoothing process, which is used to eliminate the surface distortions, destroys the centeredness of the bones. We think the centeredness property does not need to be embedded as the center of locus of a mesh model but do need to be embedded inside the mesh model. Thus we considered this requirement as the constraint of our bone smoothing algorithm.

**Smoothness of bones**: There are two aspects in the smoothness that we need to take care: curvature changes of the bone and the joints distribution. The smoothness of the curvatures will affect the deformation result. For example if we use one curve bone to represent the human vertebral column, unsmoothed bone bending will result large distortion on the mesh model surface. Also the joints distribution reflects the skin mapping of the bone, which also has influence to the deformation result.

**Skin mapping**: this property reflects the relationship between joints and mesh model surface information. Irregular skinning association will result in bad animation result because of the undesired distortion on the mesh model. Generally, each bone of a skeleton describes the topology feature for a mesh part. A bone should perpendicular to its associated surface, despite the two end joints associated surface region. We need an error metric to measure the difference between bones orientation and its associated surface information.

We use Figure 6-18 to describe the problems in the existing work. Figure 6-18.a is a skeleton example from [86]. Figure 6-18.b describes the skin mapping problem, where some parts of surface is clustered into the wrong parts, such as some parts of blue region should be clustered into yellow region (Figure 6-18.b root of arm region). We had solved this problem by using a new segmentation metric, which had been discussed in section 5.4.2. Figure 6-18.c describes the smoothness problem for articulated bones. Figure 6-18.d is another example of the smoothness problem, where the joints are not distributed evenly. This is because the joint position is defined from the surface of the mesh, which may result in distortions due to the irregularity of the surface details.

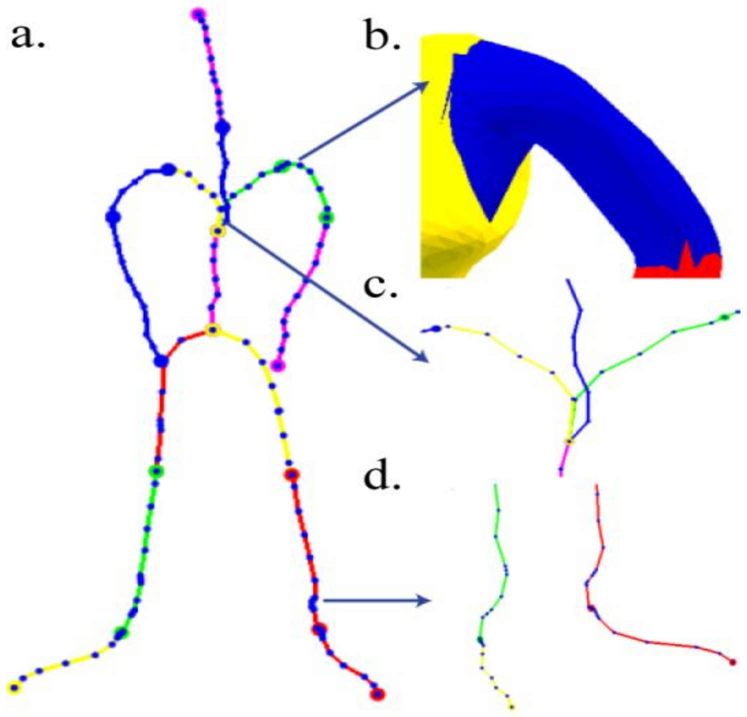

**Figure 6-18 Skeleton Issue from [86]**

## 6.4.2. Method in Skeleton Evaluation

To evaluate the quality of a skeleton, we implement two metrics to validate the correctness of a skeleton from several aspects. We measure the smoothness of a skeleton by monitoring the curvature change for each joint in a bone. The smoothness of a skeleton plays an important role in skeleton-driven animation. Unsmoothed skeleton may result in large distortions in a deformation process. Also the skin mapping quality has important impact to the deformation process. So, we proposed a metric to evaluate the quality of the skin mapping.

### 6.4.2.1. Smoothness Error Metric

To measure the smoothness of a skeleton, we compute the curvature value for each joint inside a bone. To detect large curvature change, we use an augment function to enlarge the curvature values. Given a bone, our smoothness error metric is defined as:

$$E_{smo} = \frac{\sum_{i=1}^{n} | f(\cos(\pi) - f(\cos(\theta_i)) |}{n} \qquad (6.4.1)$$

where $f$ is an augment function to enlarge the curvature values and $\theta_i$ is an angle between two line segments $\overrightarrow{j_i j_{i-1}}$ and $\overrightarrow{j_i j_{i+1}}$. $j_i$ is a joint inside a bone and $i$ is a index of joint $j_i$ inside the $plist$ [16] of bone. $\pi$ indicates the ideal straightness situation where the angle between $\overrightarrow{j_i j_{i-1}}$ and $\overrightarrow{j_i j_{i+1}}$ is $180^\circ$.

**Table 6-3 Smoothness Error Results**

| Model | Error Result before Smoothing | Error Result after Smoothing |
|---|---|---|
| Girl | 0.335535 | 0.004439 |
| Horse | 0.101076 | 0.004240 |
| Camel | 0.095717 | 0.004356 |
| 3D Donut | 0.138169 | 0.003622 |
| Frog | 0.216922 | 0.004448 |
| Monster | 0.184602 | 0.004592 |
| Man | 0.111819 | 0.004147 |
| Woman | 0.248719 | 0.004626 |
| Hand | 0.297429 | 0.004053 |
| Rhino | 0.222870 | 0.004858 |
| Raptor | 0.114235 | 0.004332 |

[16] For the definition of $plist$, please see section 4.2.3 or section 6.2

Based on this metric, we defined our skeleton smooth algorithm to optimize the skeleton structure. We have discussed this issue in Figure 6-5.

### 6.4.2.2. Skin mapping Error Metric

The quality of skin mapping is also one aspect in skeleton evaluation. This is because the skin mapping will affect the animation result, especially for LBS based method. We make use two vectors to measure the quality of skin mapping. The concept of our method can be revealed using Figure 5-9, in which a bone should perpendicular to most of its associated surface vertices. Our bone is constructed by a set of ordered joints. So the difference between joint's orientation and the central axis of its associated surface vertices can be used to measure the distribution quality of surface vertices.

Our metric is designed as follows:

$$E_{ori} = \frac{1}{n}\sum_{i=1}^{n} \| \overrightarrow{mv_{ori}^{i}} - \overrightarrow{mv_{ori\_s}^{i}} \| \qquad (6.4.2)$$

where $\overrightarrow{mv_{ori}^{i}}$ is the bone curvature at joint $i$, which is computed as the average math vector of the two adjacent bone segments connected by joint $i$. $\overrightarrow{mv_{ori\_s}^{i}}$ is the orientation that we computed from the surface of the mesh model which is associated with joint $i$. We compute the central axis $\overrightarrow{mv_{ori\_s}^{i}}$ of joint $i$ from its associated surface. Each joint $i$ is associated with a set of triangles which is collected from LSD based skeleton extraction. We take this vector as the candidate to measure the correctness of bone orientations at joint $i$.

The axis of joint $i$ should perpendicular to its associated vertices on the surface. To compute this central axis $X$, we have to solve the following least square problem: Given a set of vertices $S_v$, each vertex $v_i$ has an average normal $N_i$ which is an outgoing normal perpendicular to the mesh surface. We need to find out the norm $X$ which is perpendicular to most of those normal $N_i$. Thus we have the following equation:

$$f(X) = \sum_{i=1}^{n} \| N_i \cdot X \|^2 \tag{6.4.3}$$

where $N_i$ is the average normal for each vertex $v_i$ , and $X$ is the joint normal .

In order to reflect the skin mapping quality, we also introduced another function:

$$f(X) = \sum_{i=1}^{n} \| N_i^{'} \cdot X \|^2 \tag{6.4.4}$$

where $N_i^{'}$ is a normal vector which starts from joint $i$ to the surface vertex $v_i$. If $v_i$ is tight around joint $i$ , the normal $N_i^{'}$ should also perpendicular to the joint normal $X$ .

We combine the above two equations together to compute $X$ . We solve these equations with constraint $\| X \| = 1$ , which is used to indicate the unit length of the central vector of joint $i$ , and $f(X) = 0$ to represent the least square sense.

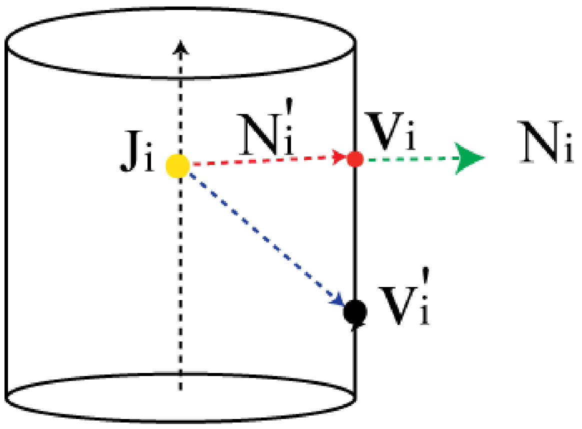

**Figure 6-19 Concept of Bone Evaluation**

Figure 6-19 is an example to explain our skin mapping error metric. In this figure, the yellow dot $J_i$ is a joint inside a bone. The red dot $V_i$ is a vertex on the surface, in which the green line $N_i$ with arrow is the average normal of $V_i$. The red line $N_i^{'}$ is another normal vector which has the direction from joint $J_i$ to vertex $V_i$. If a vertex is far away from joint $J_i$, such as the block dot $V_i^{'}$, we cannot get the right normal direction (see the dotted black line near the joint) for joint $J_i$.

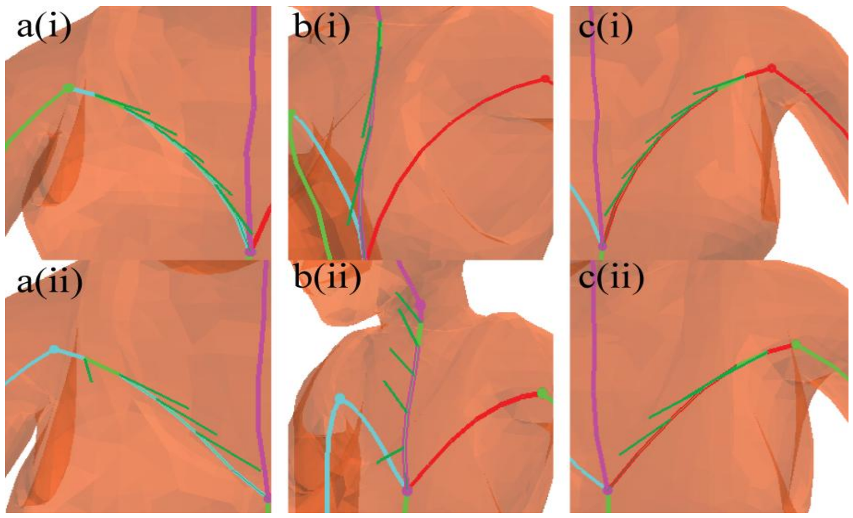

**Figure 6-20 Skeleton Evaluation Results Comparison**

Figure 6-20 describes the comparison of our skin mapping[17] quality. In this figure, there are two groups of bones. Group a is our skeleton with skin remapping process, whereas group b is the skeleton result without skin remapping process. The short green lines are the normal vectors for each joint inside a bone. As we can see, in Figure 6-20.a(i), b(i) and c(i), the joint normal vectors are similar to the bone orientation. However, in Figure 6-20.a(ii), b(ii) and c(ii), there are large difference between bone orientation and joint normal vectors. These large differences indicate that there are skin mapping problem.

## 6.5. Comparison with the Existing Skeleton Extraction Method

Au et al. [17] proposed a mesh contraction based skeleton extraction method which extracts the skeleton automatically from a given mesh model. This method is a representative automatic skeleton generation method in recent years. To evaluate our method, we compared this method with our method. Figure 6-21 displays the resulted skeleton from [17] and our method. The skeleton in Figure 6-21a(i) is generated from mesh contraction method [17] whereas the skeleton in a(ii) is the result from our method. Traditional skeleton extraction methods [17, 29] need to compute feature points before extracting the skeleton from a mesh model. Those feature points are the starting points for skeleton extractions. For example, by computing the normal differentiation, one can identify the finger tips as feature points. Thus, thin and tiny bones like finger skeleton can be extracted from those feature points. However such a step is also prone to the "noise" problem, which may lead to incorrect topological structure for skeleton generation. This is because some non-featured local model details may be identified as features. An example of such a case can be found in Figure 2-9.

[17] See Figure 6-7 and Figure 6-8 for details.

Au's method may generate incorrect skeleton due the "noise" point selection problem. The problem can be eliminated by manually selecting feature points before performing a skeleton extraction task. Figure 6-21 b(i) is an example of such a situation due to the inappropriate selection of feature points. There are several branches in b(i), which is obviously incorrect to the skeleton structure. Our skeleton structure is constructed from a group of model segments. We select the center of each segmentation boundary as feature points to eliminate the "noise" point selection problem. This can maintain correct topological and geometrical features of the generated skeleton, since the segmentation process helps define semantically meaningful skeleton parts for the mesh model and the segmentation boundaries define the connection relationship for different parts. For example, the head part has only one segmentation boundary, thus we have one single bone in the head region as shown in b(ii). Figure 6-21 d(i) illustrates another example of the "noise" problem at the foot region. Our method does not have such a problem (see d(ii)).

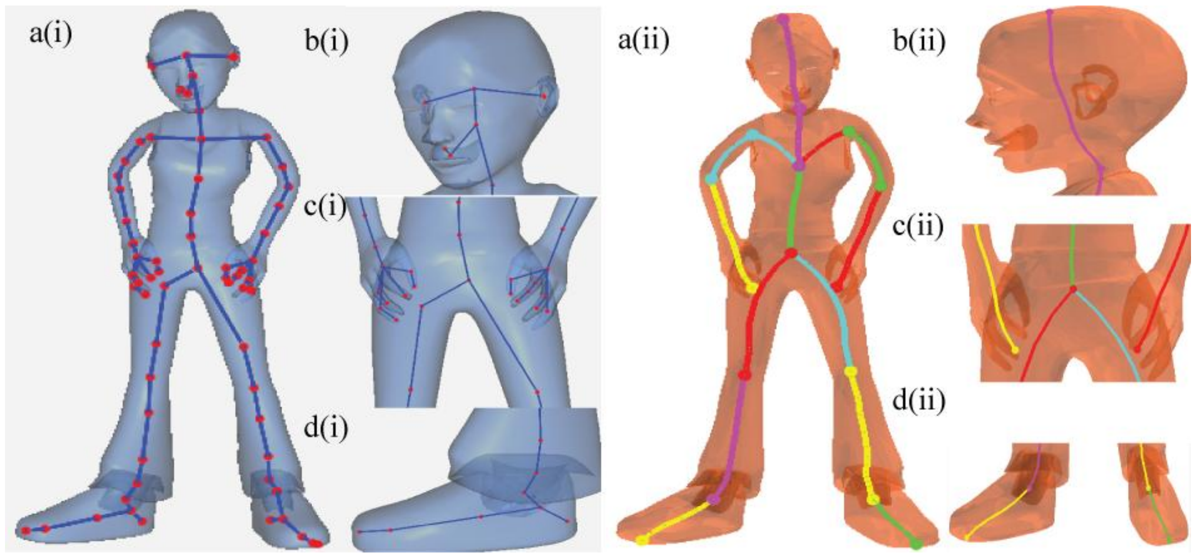

**Figure 6-21 Resulted Skeleton from the Girl Model**

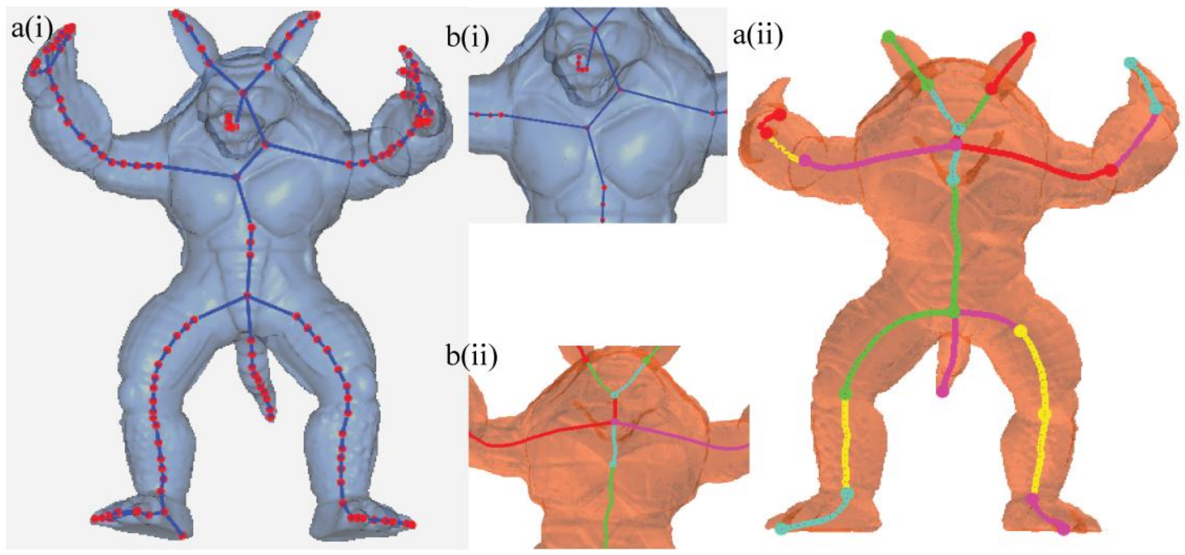

**Figure 6-22 Resulted Skeleton from the Armadillo Model**

Besides the “noise” problem, the joint positions may not be placed correctly. Figure 6-21c(i) shows this problem. As we can see the joint in the waist region which is connected by three bones are not placed correctly. On the contrast, we have a better position for this kind of joint (see c(ii)). Also, the bones are not located at the center of mesh body (see the right leg of c(i)). This is because that the centeredness of the skeletons cannot be guaranteed during the mesh contraction process.

Figure 6-22 is another comparison using the Armadillo model. Figure 6-22 a(i) displays the resulted skeleton from Au’s method and a(ii) displays the resulted skeleton from our method. The major difference is that our method generates better result than Au’s method (see Figure 6-22 b(i) and b(ii) for comparison) in terms of topology structure. In the major body part (see b(ii)), we have a star-type skeleton in which one central joint is used to connect bones for two arms and one neck and one waist region. However from Au’s result (as shown in b(i)), the neck bone is connected with the right arm which is inappropriate to our skeleton recognition. This is because that our method did not modify the mesh model during our skeleton extraction process, whereas Au’s method changes the vertices’ positions during the mesh contraction process. Such kind of modification would destroy the connection relationship among different parts (see the result in b(i)).

MAT based method [25] guaranteed the centeredness of the skeleton which provides smooth 1D skeleton curve. However, the skeleton result is highly affected by surface local details. An example of such situation is discussed in section 2.4.2 with Figure 2-7, where unnecessary branches may occur in salience regions. Compare with the existing methods [22, 25, 29, 64, 75], our method is able to generate well-positioned joints. This is because we have adopted a new segmentation method, which divides a mesh model into meaningful pieces, in our skeleton generation system. Our segmentation method generates reliable segmentation boundaries for skeleton construction usage.

Aujay et al. [18] applied Reeb graph to capture the topological structure of a mesh model, where mesh model symmetry and joint type heuristics were applied additionally to align the bones and joints more correctly. This method emphasizes on generating topologically adequate skeleton structure. However, it may only be applicable to mesh models with reliable topological structures; and more importantly, mesh model symmetry exists. Besides, the treatment of this method is restricted, as the Reeb graph generation is merely with a single direction from a user chosen (or automatically generated) source point. If complicated or irregular mesh model structure exists around such a direction, the topology of the generated skeleton structure may be problematic. In contrast, our work can handle mesh models with irregular structures. This relates to our way of skeleton structure generation, in which we consider all connecting boundaries on a mesh part and generate bones in different directions from these boundaries. In addition, [18] may generate meaningful joints if one is connecting to multiple bones due to joint type heuristics are applied. Our work does not have such a restriction, as we generate joints based on topological parts' boundaries and skeleton bone traits.

Tierny [29] et al. adopted a Reeb graph (which is also called LSD) construction method to construct mesh skeleton. This is similar to our method, where LSD is adopted in our skeleton construction process.  Rather than using the center of segmentation boundaries as the starting points, Tierny et al. proposed a feature extraction method to selection feature points on the surface of a mesh model as the starting points. However, this method suffers from three problems. First, the feature points from mesh surface may contains noise due to the local details variation, which will affect the skeleton construction. Second, skeleton constructed from using feature points may lack of joint specifications in some articulated regions, such as the knee regions. Last, the connection relationship is not well defined in regions which have multiple branches. By considering the problems that we mentioned above, we utilize the segmentation boundaries to produce joints first and then apply LSD method to construction bones for each part.

Wu et al. [25] proposed a method to identify main joints from an input mesh model, including some from major medial axes and some prong-feature points, which were then connected to form a temporary skeleton structure. A snake model was thereafter applied iteratively to each segment of the skeleton structure to modify its shape for matching with the mesh model's topological structure. Extra joints were therefore introduced. Although this method may produce a topologically pleasing skeleton structure, its quality cannot be guaranteed; as running the snake model is too time consuming that the method has to limit the number of iterations and cannot impose any quality control. In addition, the joints generated are too many and do not necessarily possess any semantically meaning.  Similar to [25], our work aims to match the mesh model's topological structure. We satisfy this criterion by using user sketching to lead the skeleton structure generation process; and use our segmentation metric to interpret such topological structure more accurately. In addition, as we generate joints based on topological parts' boundaries and skeleton bone traits, our resultant joints are semantically more plausible.

We also implemented two metrics to evaluate the quality of our skeleton. The first metric is smoothness metric. Based on this guidance of this metric, we have designed a skeleton smoothing method to optimize our skeleton. We also proposed a metric to evaluate the quality of skin mapping. Rather than using data to tell the difference, we use a Figure 6-20 to describe the effect of our metric. To describe the advantages of our method, we compared with the mesh contraction based skeleton extraction method [17].

## 6.6. Summary

In this chapter, we proposed a skeleton extraction method which makes use of the center of segmentation boundaries as the starting points to extract bones. By analyzing different skeleton results from previous section, we have found out the both the strengths and weaknesses of our skeleton extraction method.

**Strengths**: Our skeleton contains well-positioned joints. Our segmentation metric defines the segmentation boundaries from four aspects to produce reasonable segmentation results. They are namely, the horizontal surface curvature variation, the distance to reference bones, the region boundary curvature variation and the normal difference between a reference bone and the triangles' normal. Besides, our skeleton has reasonable mapping between the surface vertices and joints (or bones). This property is very useful in skeleton-driven animation.

**Weaknesses**: Using boundary as the starting point to extract a skeleton is not always correct. Especially, if a boundary is a wrong structure from the mesh model itself (such as the open-end boundary as described in Figure 6-3) rather than been generated from our segmentation method, we cannot deduce a good result. Also even the boundary comes from our segmentation method; we still cannot guarantee the correctness of our skeleton structure. Normally, such a situation occurs when we extract a skeleton from a squeezed model, such as the models in Figure 6-17.

# Chapter 7

## 7. Conclusions

### 7.1. Applications

We use MViewer [104], which is open source mesh display interface, as our GUI to display 3D models. We adopted the LBS technique in our system to perform skeleton-driven animation tasks. A user needs to select bones and then sketch on the screen to define different poses. Because each bone contains the mapping between joints and vertices on the mesh model, it is easy to apply LBS algorithm to achieve skeleton-driven animations.

Figure 7-1 describes the animation result which comes from our skeleton-driven method. In our system, we allow a user to change to skeleton pose by manipulating any bones inside the skeleton. When we select one bone, we define two joints: the root joint (the green point) and the tip-joint (the yellow point). The root joint is a fixed joint whereas the tip-joint is used to rotate around the root joint. When we select one bone, the root joint divides all the bones into two parts: fixed bones and dynamic bones. Fixed bones keep unchange whereas the dynamic bones change the positions when a user rotation the tip-joint. Each dynamic bone contains a rotation matrix. We make use of those matrices to perform a LBS algorithm.

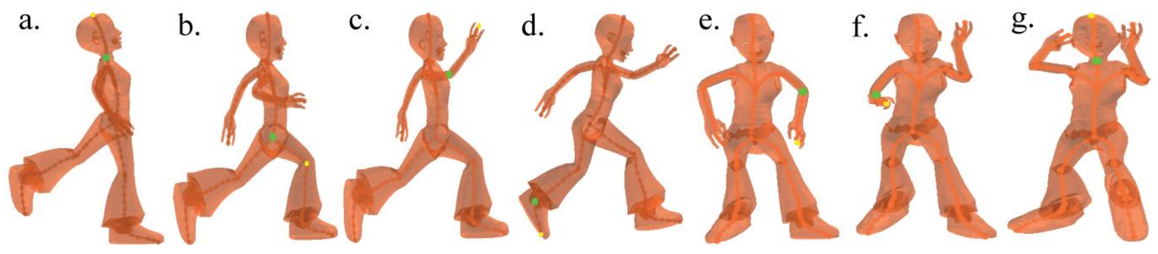

**Figure 7-1 Kung Fu Girl**

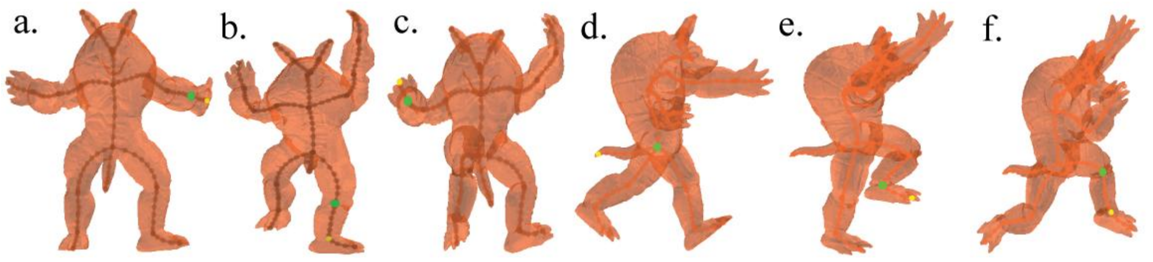

**Figure 7-2 Armadillo Monster**

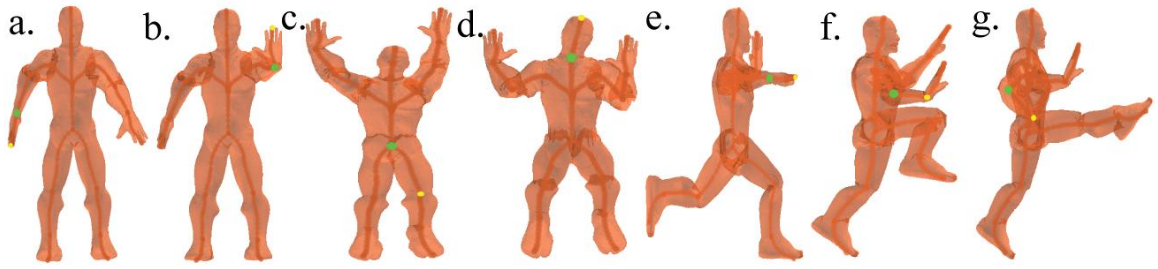

**Figure 7-3 Kung Fu Man**

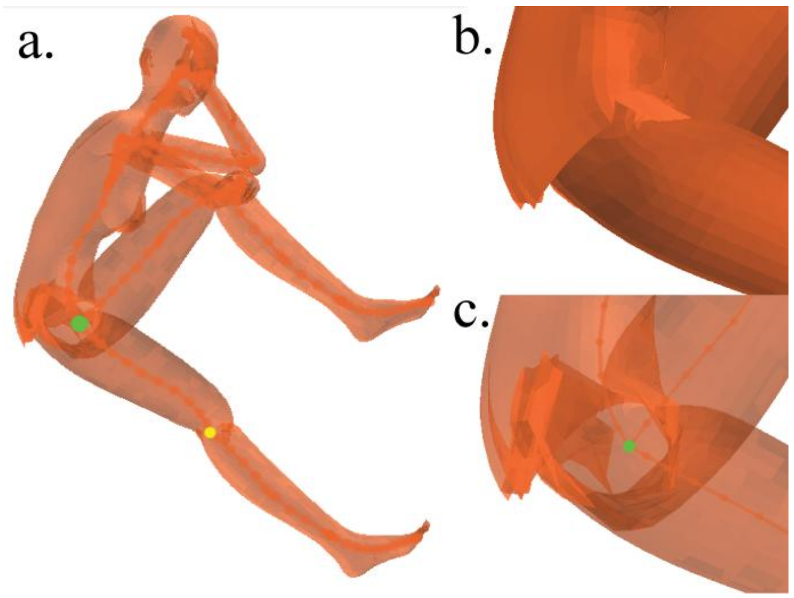

**Figure 7-4 Self-Intersection Problem**

Figure 7-2 and Figure 7-3 are the animation results for another two different models. Our skeleton-driven animation is intuitive and easy to use. However, there are some disadvantages of using LBS method, such as the self-intersection problem in large rotation regions. We use Figure

7-4 to illustrate this problem. In Figure 7-4.a, the green point is the root point whereas the yellow point is the manipulator point. When we rotate the selected bone with a large angle, some region may have serious self-intersection problem (as shown in Figure 7-4.b, and Figure 7-4.c is the transparent display of Figure 7-4.b region).

To avoid such kind of self-intersection problem, we merged the traditional LBS method with Laplacian surface mesh deformation [6] to create reasonable animation results. The reason that we merge those two methods is that LBS method is fast and result in rigid shape transformation and Laplacian surface deformation can preserve local details which will also reduce the self-intersection problem from using the traditional LBS method.

When we select a bone to perform skeleton-driven animation, our system will first create a Laplacian matrix by using all the vertices within a selected Region of Interest (ROI). In our implementation, all the transformed vertices are considered as positional constraints. And we make use of CHOLMOD [105, 106], which is a set of ANSI C routines for sparse Cholesky factorization,  to compute the updated positions for all the vertices in the ROI.

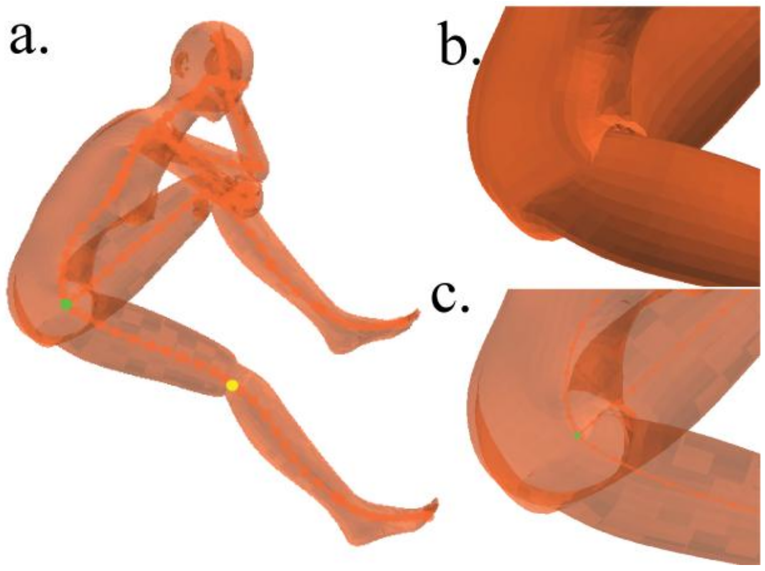

**Figure 7-5  Laplacian Deformation with LBS result as constraints**

Figure 7-5 shows the result of using Laplacian mesh deformation technique in our animation process. The self-intersection problem is eliminated by using Laplacian mesh deformation technique. Laplacian mesh deformation technique is able to preserve local details, which means that the local surface curvature is preserved. Especially, for those segmentation boundary regions, by preserving the local details, we can eliminate the self-intersection problem.

## 7.2. Conclusion

### 7.2.1. Our Method

Our research work contains two major aspects: the analysis of user sketching and the skeleton extraction. In the sketching analysis process, we proposed two methods to extract reference bones, which are used in the skeleton extraction process, from user sketched bones. The first algorithm is used to check each sketched bone from user's sketching. We proposed a DP algorithm to detect large curvature changes for each sketched bone. If there are some large curvature changes, we will divide the sketched bone into several parts. The second algorithm is used to cluster similar sketched bones into one group, so that we can create a reference bone by combining all the bones in the same group together. Our purpose is to clarify a user's intention from his/her sketching. By merging similar sketched bones or dividing sketched bones which contain large curvature, we divide the sketched bones into several groups and construct a reference bone for each group. The number of reference bones defines the number of mesh parts.

We divide the skeleton extraction process into three parts: mesh segmentation, bone extraction and skeleton combination. In mesh segmentation process, we adopted the reference bones as input to perform a multiple-region growing based segmentation process. To obtain meaningful parts and avoid over-segmentation issue, we designed a new segmentation metric. Based on this metric, we can divide a mesh model into several parts which have reliable segmentation boundaries. This is because we have designed four different factors to limit the segmentation process within a specified region[18]. Despite the minima rule that we adopted in our segmentation metric, we also introduced the distance constraint to reference bones, and the normal difference between a surface triangle and the reference bone. Then we utilize the segmentation boundaries to perform bone extraction process. We proposed a LSD method to extract bones for each part. Starting from the center of each boundary, we computed the geometric distance for each vertex in a mesh part. Based on this distance, we created a set of contour lines to construct a bone. Afterwards, we designed line smoothing algorithm to eliminate the distance distortions due to the local details variations. Finally we connected those bones which shared the same boundaries to construct the final skeleton.

We also introduced a skin remapping process to eliminate the wrong mapping result from the LSD method[19]. To evaluate the quality of our skeleton, we proposed two set of metrics to evaluate the smoothness of a skeleton and the skin mapping.

### 7.2.2. Advantages

Compared with the existing methods, our method has the following advantages:

1. Easy to use – we allow a user to sketch freely on the mesh model, which means that the user does not need to be trained to use the skeleton extraction system.

[18] The specified regions are those surface patches which are associated with each reference bone.

[19] During the LSD based bone extraction process, each joint is associated with a set of surface vertices. So, the correspondence between the surface vertices and the joint defines the mapping relationship.

2. Well-positioned joints specification – we use segmentation method to define the joint positions. To get reasonable segmentation boundaries, we designed a new segmentation metric.
3. Well-defined skin mapping – the skeleton from our method contains reasonable mapping relationship between bones and surface data (such as triangles or vertices).

### 7.2.3. Disadvantages

However, we do notice that there are some problems in our research work.  We summarized those problems as follows:

1. The quality of our skeleton depends on the segmentation result. It is hard to obtain a reasonable skeleton structure from mesh models which have no clear topology specifications, such as the Stanford bunny model etc..
2. We introduce a global distance constraint in our segmentation metric. The global distance metric is defined by using the Euclidean distance between surface vertices and the reference bones. Thus the problem is that the segmentation result is affected by the position of user sketched bones.
3. Besides, if a user missed some parts in the sketching process, our segmentation method cannot generate the skeleton with well-positioned joints. To correct this problem, we have introduced a DP algorithm to divide those bones into several parts. By monitoring the curvature change of a bone, we can identify some critical joints which have large curvature changes. Those critical joints divide a bone into several new bones, which also divide the mesh part into several new mesh parts (each new mesh part contains one new bone). However, such method only works on tube based shape. It does not work on mesh parts which have multiple connection relationship regions.

## 7.3. Future Work

First, we need to solve the problem that we have mentioned in section 6.3. With regards to models without clear topology specifications, the segmentation boundaries are not suitable for skeleton extraction. We need a method to approximate the topology features for those mesh parts. In this case, we want to utilize the existing automatic methods which can recognize the structure of a mesh part to deduce a better skeleton result in those regions which have no clear topology features.

We have proposed two metrics to evaluate the quality of a skeleton from two aspects: smoothness and skin mapping. Cornea et al. have discussed several properties for the 1D curve skeleton evaluation, such as the thinness, centeredness etc.. Most of properties were analyzed from their correspondent methods features. And there is no unique metric to evaluate those features mathematically. Although we have implement two metrics to evaluate the smoothness, centeredness and skin mapping features, there are lots of work to be done in the skeleton evaluation area.

We can make use of our skeleton result to deduce different poses for an articulated model. We have shown example in Figure 7-1, Figure 7-2 and Figure 7-3. But the animation work needs to be done manually. In the future, we want to achieve automatic skeleton-driven animation results with some existing motion data. This involves some skeleton retrieval [97] and skeleton projection issue for two different skeleton structures. Also because we have built the mapping relationship between joint and surface vertices, such as vertices or triangles, we can utilize this feature to perform skeleton-driven model editing tasks.

Another possible research direction is the automatic skeleton-driven animation process. For example in action movies or games, fighting motions are the most common scenarios. There are lots of motion capture data, such as a person plays Kung Fu. We can utilize those existing motion data as the input to create some correspondent motion sequences, so that the fighting sequence needs not to be completed with two actors. Besides, there is a lot of existing motion data with some predefined skeleton structure on the Internet. In order to utilize the existing motion data, we need to convert our skeleton structure to the predefined skeleton structure. So how to adapt our skeleton structure to some similar skeleton structure is an interesting field to be explored.

# References

[1] H. Pottmann*, et al.*, *The isophotic metric and its application to feature sensitive morphology on surfaces* vol. 3024: Springer, 2004.

[2] H. Hoppe*, et al.*, "Mesh optimization," presented at the Proceedings of the 20th annual conference on Computer graphics and interactive techniques, Anaheim, CA, 1993.

[3] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.,* vol. 20, pp. 151-160, 1986.

[4] D. Zorin*, et al.*, "Interactive multiresolution mesh editing," presented at the Proceedings of the 24th annual conference on Computer graphics and interactive techniques, 1997.

[5] L. Kobbelt*, et al.*, "Interactive multi-resolution modeling on arbitrary meshes," presented at the Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998.

[6] O. Sorkine*, et al.*, "Laplacian surface editing," in *Proceedings of the Eurographics 2004/ACM SIGGRAPH symposium on Geometry processing*, Nice, France, 2004, pp. 175-184.

[7] Y. Lipman*, et al.*, "Differential Coordinates for Interactive Mesh Editing," in *Proceedings of the Shape Modeling International 2004*, 2004, pp. 181-190.

[8] Y. Yu*, et al.*, "Mesh editing with poisson-based gradient field manipulation," *ACM Trans. Graph.,* vol. 23, pp. 644-651, 2004.

[9] A. Nealen*, et al.*, "A Sketch-based Interface for Detail-preserving Mesh Editing," presented at the ACM SIGGRAPH 2005 Papers, Los Angeles, California, 2005.

[10] J. Huang*, et al.*, "Subspace gradient domain mesh deformation," *ACM Trans. Graph.,* vol. 25, pp. 1126-1134, 2006.

[11] K. Zhou*, et al.*, "Direct manipulation of subdivision surfaces on GPUs," *ACM Trans. Graph.,* vol. 26, p. 91, 2007.

[12] S. Capell*, et al.*, "Interactive Skeleton-driven Dynamic Deformations," in *Proceedings of the ACM SIGGRAPH 2002*, 2002, pp. 586-593.

[13] J. P. Lewis*, et al.*, "Pose Space Deformation: a unified approach to shape interpolation and skeleton-driven deformation," presented at the Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000.

[14] D. Jacka*, et al.*, "A comparison of linear skinning techniques for character animation," presented at the Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, Grahamstown, South Africa, 2007.

[15] B. Merry*, et al.*, "Animation space: A truly linear framework for character animation," *ACM Trans. Graph.,* vol. 25, pp. 1400-1423, 2006.

[16] Andrei Sharf*, et al.*, "On-the-fly Curve-skeleton Computation for 3D Shapes," *Computer Graphics Forum,* vol. 26, pp. 323-328, 2007.

[17] O. K.-C. Au*, et al.*, "Skeleton extraction by mesh contraction," *ACM Trans. Graph.,* vol. 27, pp. 1-10, 2008.

[18] G. Aujay*, et al.*, "Harmonic Skeleton for Realistic Character Animation," presented at the Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Diego, California, 2007.

[19] J.-H. Chuang*, et al.*, "Skeletonization of Three-Dimensional Object Using Generalized Potential Field," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 22, pp. 1241-1251, 2000.

[20] L. Kobbelt*, et al.*, "Skeleton Extraction of 3D Objects with Visible Repulsive Force," presented at the In Computer Graphics Workshop 2003, 2003.

[21] P.-C. Liu*, et al.*, "Automatic Animation Skeleton Construction Using Repulsive Force Field," presented at the Proceedings of the 11th Pacific Conference on Computer Graphics and Applications, 2003.

[22] J. Tierny*, et al.*, "Enhancing 3D mesh topological skeletons with discrete contour constrictions," *Vis. Comput.,* vol. 24, pp. 155-172, 2008.

[23] L. Wade and R. E. Parent, "Automated generation of control skeletons for use in animation," *The Visual Computer,* vol. 18, pp. 97-110, 2002.

[24] Y.-S. Wang and T.-Y. Lee, "Curve-Skeleton Extraction Using Iterative Least Squares Optimization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, pp. 926-936, 2008.

[25] F.-C. Wu*, et al.*, "Domain connected graph: the skeleton of a closed 3D shape for animation," *Vis. Comput.,* vol. 22, pp. 117-135, 2006.

[26] Y. Zhou*, et al.*, "Three-dimensional skeleton and centerline generation based on an approximate minimum distance field," *The Visual Computer,* vol. 14, pp. 303-314, 1998.

[27] Y. Zhou and A. W. Toga, "Efficient skeletonization of volumetric objects," *Visualization and Computer Graphics, IEEE Transactions on,* vol. 5, pp. 196-209, 1999.

[28] N. Amenta*, et al.*, "The power crust," presented at the Proceedings of the sixth ACM symposium on Solid modeling and applications, Ann Arbor, Michigan, United States, 2001.

[29] J. Tierny*, et al.*, "3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses," in *14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006)*, 2006, pp. 85-94.

[30] Y. Lee*, et al.*, "Mesh scissoring with minima rule and part salience," *Comput. Aided Geom. Des.,* vol. 22, pp. 444-465, 2005.

[31] D. D. Hoffman and W. A. Richards, "Parts of recognition," *Cognition,* vol. 18, pp. 65-96, 1984.

[32] F. Lazarus and A. Verroust, "Level set diagrams of polyhedral objects," presented at the Proceedings of the fifth ACM symposium on Solid modeling and applications, Ann Arbor, Michigan, United States, 1999.

[33] A. Telea and A. Vilanova, "A robust level-set algorithm for centerline extraction," presented at the Proceedings of the symposium on Data visualisation 2003, Grenoble, France, 2003.

[34] Curvy3D, "http://www.curvy3d.com/," ed.

[35] SketchUp, "http://sketchup.google.com/," ed.

[36] VRMesh, "http://www.vrmesh.com/," ed.

[37] S. Owada*, et al.*, "A sketching interface for modeling the internal structures of 3D shapes," presented at the ACM SIGGRAPH 2006 Courses, Boston, Massachusetts, 2006.

[38] T. Igarashi*, et al.*, "Teddy: a sketching interface for 3D freeform design," presented at the Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999.

[39] A. Nealen*, et al.*, "FiberMesh: designing freeform surfaces with 3D curves," *ACM Trans. Graph.,* vol. 26, p. 41, 2007.

[40] Y. Kho and M. Garland, "Sketching mesh deformations," presented at the Proceedings of the 2005 symposium on Interactive 3D graphics and games, Washington, District of Columbia, 2005.

[41] R. C. Zeleznik*, et al.*, "SKETCH: an interface for sketching 3D scenes," presented at the Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[42] O. Karpenko*, et al.*, "Free-form Sketching with Variational Implicit Surfaces," ed, 2002.

[43] K. Das*, et al.*, "Sketching Free-form Surfaces Using Network of Curves," presented at the EUROGRAPH-ICS Workshop on Sketch-based Interfaces and Modeling, 2005.

[44] P. Blair, *Cartoon Animation* Walter Foster Publishing, 1994.

[45] L. B. Kara*, et al.*, "Pen-based styling design of 3D geometry using concept sketches and template models," presented at the Proceedings of the 2006 ACM symposium on Solid and physical modeling, Cardiff, Wales, United Kingdom, 2006.

[46] J. Davis*, et al.*, "A sketching interface for articulated figure animation," presented at the Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Diego, California, 2003.

[47] E. Chang and O. C. Jenkins, "Sketching Articulation And Pose For Facial Animation," presented at the Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Vienna, Austria, 2006.

[48] Wolfire, "http://blog.wolfire.com/2009/11/Triangle-mesh-voxelization," in *Wolfire Blog*, ed.

[49] C.-M. Ma*, et al.*, "Three-Dimensional Topology Preserving Reduction on the 4-Subfields," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 24, pp. 1594-1605, 2002.

[50] N. D. Cornea*, et al.*, "Curve-Skeleton Properties, Applications, and Algorithms," *Visualization and Computer Graphics, IEEE Transactions on,* vol. 13, pp. 530-548, 2007.

[51] J. Toriwaki and H. Yoshida, *Fundamentals of Three-Dimensional Digital Image Processing*, 2009.

[52] J. Toriwaki and T. Yonekura, "Euler Number and Connectivity Indexes of a Three Dimensional Digital Picture," *Forma,* vol. 17, pp. 183-209, 2002.

[53] D. G. Morgenthaler and A. Rosenfeld, "Surfaces in three-dimensional digital images," *Information and Control,* vol. 51, pp. 227-247, 1981.

[54] T. Y. Kong and A. Rosenfeld, "Digital topology: introduction and survey," *Comput. Vision Graph. Image Process.,* vol. 48, pp. 357-393, 1989.

[55] M. Couprie and R. Zrour, "Discrete Bisector Function and Euclidean Skeleton," in *Discrete Geometry for Computer Imagery*. vol. 3429, E. Andres*, et al.*, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 216-227.

[56] N. Gagvani and D. Silver, "Animating volumetric models," *Graph. Models,* vol. 63, pp. 443-458, 2001.

[57] C. Pudney, "Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images," *Comput. Vis. Image Underst.,* vol. 72, pp. 404-413, 1998.

[58] N. Gagvani and D. Silver, "Parameter-controlled volume thinning," *CVGIP: Graph. Models Image Process.,* vol. 61, pp. 149-164, 1999.

[59] I. Bitter*, et al.*, "Penalized-Distance Volumetric Skeleton Algorithm," *IEEE Transactions on Visualization and Computer Graphics,* vol. 7, pp. 195-206, 2001.

[60] S. Bouix and K. Siddiqi, "Divergence-Based Medial Surfaces," in *Computer Vision - ECCV 2000*. vol. 1842, ed: Springer Berlin / Heidelberg, 2000, pp. 603-618.

[61] T. He*, et al.*, "Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs," *IEEE Transactions on Visualization and Computer Graphics,* vol. 7, pp. 333-342, 2001.

[62] H. Sundar*, et al.*, "Skeleton Based Shape Matching and Retrieval," presented at the Proceedings of the Shape Modeling International 2003, 2003.

[63] M. Wan*, et al.*, "Distance-field based skeletons for virtual navigation," presented at the Proceedings of the conference on Visualization '01, San Diego, California, 2001.

[64] W.-C. Ma*, et al.*, "Skeleton Extraction of 3D Objects with Radial Basis Functions," presented at the Proceedings of the Shape Modeling International 2003, 2003.

[65] S. Katz*, et al.*, "Mesh segmentation using feature point and core extraction," *The Visual Computer,* vol. 21, pp. 649-658, 2005.

[66] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Trans. Graph.,* vol. 22, pp. 954-961, 2003.

[67] M. Attene*, et al.*, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer: International Journal of Computer Graphics* vol. 22, pp. 181-193, 2006.

[68] O. Sorkine, "Differential Representations for Mesh Processing," *Computer Graphics Forum,* vol. 25, pp. 789-807, 2006.

[69] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Forms*, ed: MIT Press, 1967, pp. 362--380.

[70] A. P. Mangan and R. T. Whitaker, "Partitioning 3D Surface Meshes Using Watershed Segmentation," *IEEE Transactions on Visualization and Computer Graphics,* vol. 5, pp. 308-321, 1999.

[71] R. Liu and H. Zhang, "Segmentation of 3D Meshes through Spectral Clustering," presented at the Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, 2004.

[72] Y. Zhou and Z. Huang, "Decomposing Polygon Meshes by Means of Critical Points," presented at the Proceedings of the 10th International Multimedia Modelling Conference, 2004.

[73] L. Grady, "Random Walks for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 28, pp. 1768-1783, 2006.

[74] W. L. Winston, *Introduction to Mathematical Programming: Applications and Algorithms*, 2 ed.: International Thomson Publishing, 1995.

[75] Y. Shinagawa and T. L. Kunii, "Constructing a Reeb graph automatically from cross sections," *IEEE Comput. Graph. Appl.,* vol. 11, pp. 44-51, 1991.

[76] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," presented at the Proceedings of the fourth Eurographics symposium on Geometry processing, Cagliari, Sardinia, Italy, 2006.

[77] V. Pascucci*, et al.*, "Robust on-line computation of Reeb graphs: simplicity and speed," *ACM Trans. Graph.,* vol. 26, p. 58, 2007.

[78] M. Mortara*, et al.*, "Affine-Invariant Skeleton of 3D Shapes," presented at the Proceedings of the Shape Modeling International 2002 (SMI'02), 2002.

[79] Y. Rubner*, et al.*, "A Metric for Distributions with Applications to Image Databases," presented at the Proceedings of the Sixth International Conference on Computer Vision, 1998.

[80] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," presented at the Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000.

[81] D. Cohen-Steiner*, et al.*, "Variational shape approximation," *ACM Trans. Graph.,* vol. 23, pp. 905-914, 2004.

[82] X. Li*, et al.*, "Decomposing polygon meshes for interactive applications," presented at the Proceedings of the 2001 symposium on Interactive 3D graphics, 2001.

[83] B. Lévy*, et al.*, "Least squares conformal maps for automatic texture atlas generation," *ACM Trans. Graph.,* vol. 21, pp. 362-371, 2002.

[84] E. Zhang*, et al.*, "Feature-based surface parameterization and texture mapping," *ACM Trans. Graph.,* vol. 24, pp. 1-27, 2005.

[85] P. V. Sander*, et al.*, "Multi-chart geometry images," presented at the Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Aachen, Germany, 2003.

[86] Q. Zheng*, et al.*, "Sketching-Based Skeleton Generation," *Proceedings of IEEE International Conference on Ubi-media Computing,* pp. 179-186, 2010.

[87] M. Garland*, et al.*, "Hierarchical face clustering on polygonal surfaces," presented at the Proceedings of the 2001 symposium on Interactive 3D graphics, 2001.

[88] Z. Ji*, et al.*, "Easy Mesh Cutting," *Computer Graphics Forum,* vol. 25, pp. 283-291, 2006.

[89] Y.-K. Lai*, et al.*, "Fast mesh segmentation using random walks," presented at the Proceedings of the 2008 ACM symposium on Solid and physical modeling, Stony Brook, New York, 2008.

[90] S. Shlafman*, et al.*, "Metamorphosis of Polyhedral Surfaces using Decomposition," *Computer Graphics Forum,* vol. 21, pp. 219-228, 2002.

[91] X. C. Wang and C. Phillips, "Multi-weight enveloping: least-squares approximation techniques for skin animation," presented at the Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Antonio, Texas, 2002.

[92] P.-P. J. Sloan*, et al.*, "Shape by example," presented at the Proceedings of the 2001 symposium on Interactive 3D graphics, 2001.

[93] A. Mohr*, et al.*, "Direct manipulation of interactive character skins," presented at the Proceedings of the 2003 symposium on Interactive 3D graphics, Monterey, California, 2003.

[94] H.-B. Yan*, et al.*, "Shape Deformation Using a Skeleton to Drive Simplex Transformations," *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, pp. 693-706, 2008.

[95] R. Zayer*, et al.*, "Harmonic Guidance for Surface Deformation," *Computer Graphics Forum,* vol. 24, pp. 601-609, 2005.

[96] G. Taubin, "A signal processing approach to fair surface design," presented at the Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 1995.

[97] G. K. L. Tam*, et al.*, "Motion Retrieval Based on Energy Morphing," presented at the Proceedings of the Ninth IEEE International Symposium on Multimedia, 2007.

[98] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization,* vol. 10, pp. pp. 112-122, December 1973 1973.

[99] G. Lavoué*, et al.*, "A new CAD mesh segmentation method, based on curvature tensor analysis," *Comput. Aided Des.,* vol. 37, pp. 975-987, 2005.

[100] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3D mesh analysis," *ACM Trans. Graph.,* vol. 27, pp. 1-12, 2008.

[101] L. Shapira*, et al.*, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *Vis. Comput.,* vol. 24, pp. 249-259, 2008.

[102] Y. Xiao*, et al.*, "Topological Segmentation of Discrete Human Body Shapes in Various Postures Based on Geodesic Distance," presented at the Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03, 2004.

[103] Y. Xiao*, et al.*, "A Discrete Reeb Graph Approach for the Segmentation of Human Boday Scans," in *Fourth International Conference on 3D Digital Image and Modeling*, Banff, Alberta, Canada, 2003, pp. pp. 378-385.

[104] H. Cantzler and T. Breckon. http://mview.sourceforge.net/.

[105] Y. Chen*, et al.*, "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate," *ACM Trans. Math. Softw.,* vol. 35, pp. 1-14, 2008.

[106] T. A. Davis and W. W. Hager, "Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves," *ACM Trans. Math. Softw.,* vol. 35, pp. 1-23, 2009.