國立交通大學

資訊科學與工程研究所

碩 士 論 文

網格分割轉換

Mesh Segmentation Transfer

研 究 生：楊詔安

指導教授：莊榮宏　教授

中 華 民 國　一百 年 二 月

網格分割轉換
Mesh Segmentation Transfer

研　究　生：楊詔安　　　　Student：Jau-An Yang

指導教授：莊榮宏　　　　Advisor：Jung-Hong Chuang

國　立　交　通　大　學
資　訊　科　學　與　工　程　研　究　所
碩　士　論　文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

February 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年二月

網格分割轉換

研究生 ： 楊詔安　　　　　　指導教授 ： 莊榮宏 博士

國立交通大學

資訊學院資訊科學與工程研究所

摘 要

我們提出了一個網格分割轉換演算法，將來源網格的分割結果一致地對應到目標網格上。首先我們提出一個新的骨架對應方法，可以同時對應來源網格和目標網格之間的骨架節點以及連結，接著在來源網格上將每個分割邊緣線對應到所屬的骨架連結，最後將每個分割邊緣線轉換到目標網格所對應的骨架連結上。我們的骨架對應演算法對骨架的連接性較不敏感，而且利用了骨架連結的語義性群聚以及空間配置的資訊。我們的分割轉換演算法致力於在目標網格上產生分割邊緣線，使其特徵以及局部幾何細節和來源網格對應的分割邊緣線能夠一致。最後我們在各種類型的物體上展示了多樣的分割轉換結果，並和目前最新的相關研究作比較。

Mesh Segmentation Transfer

Student: Jau-An Yang Advisor: Dr. Jung-Hong Chuang

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

We present a novel segmentation transfer that maps consistent segmentation of a source mesh to a target mesh. We first propose a skeleton correspondence method to match the skeleton links and nodes between two skeletons of the source and target meshes, then associate each segment boundary with a skeleton link of the source mesh, and finally map each boundary to the corresponding skeleton link of the target mesh. Our skeleton correspondence computation is less sensitive to the skeleton connectivity and employs the clustering of semantically similar links and spatial information. The boundary transfer procedure aims to produce the segment boundaries of the target mesh whose locations and geometric properties are consistent to the ones of source. We demonstrate the segmentation transfer results on various meshes and compare with the state-of-the-art techniques.

Acknowledgments

I would like to thank my advisors, Professor Jung-Hong Chuang and Sai-Keung Wong, for their guidances, inspirations, and encouragements. I especially thank to my senior colleague, Tan-Chi Ho, for generously sharing his researches works, such as PMLab and MSP skeletons, and giving me useful advices and suggestions. I thank to all my colleagues in CGGM lab: Hong-Xuan Ji, Tsung-Shian Huang , Cheng-Min Liu and Ta-En Chen for their helps and discuss with me on research issues, my senior colleagues : Yueh-Tse Chen, Jyh-Shyang Chang, Ying-Tsung Li, Hsin-Hsiao Lin and Kuang-Wei Fu for their technical supports and useful information, and my junior colleagues: Ying-Ti Chou, Chieb Yun, Chia-Ming Liu, Yu-Chen Wu and Chien-Kai Lo for their kind assistances and encouragements. Lastly, I would like to thank my parents for their love, and support. Without their love, I could not pass all the frustrations and pain in my life.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Mesh segmentation aims to partition a mesh into several disjoint subpatches or parts, It is a fundamental problem to many graphics applications such as collision detection, parameterization, and skeletonization.

Many segmentation methods have been proposed in the last decade and most of them aimed at generating functional parts that meet the human perception. However, in order to obtain good segmentation results, most of the methods still required careful adjustment of parameters for each type of objects. According to the segmentation benchmark report of Chen et al. [CGF09], the segmentation ground truths produced by human still outperform the results produced by recent segmentation methods. By setting the ground truths as the source segmentation and transferring the segmentation to other objects, the segmentations that are similar to the source can be expected. Moreover, some applications require consistent segmentation results for a set of similar objects, which is in practice hard to obtain even by applying same segmentation algorithm to the objects. Due to the above observations, we propose a novel segmentation scheme called mesh segmentation transfer. User provides a source mesh, a segmentation result of source mesh, and a target mesh. We aim to transfer the segmentation on the source mesh to the target mesh such that two segmentations are consistent.

Our approach first derives the shape correspondence between source and target meshes via skeleton matching and then maps each segment boundary on the source mesh to the target based on the link-to-link correspondence. We employ the minimum slice perimeter(MSP) curve skeleton proposed in [HJWC], which is a high resolution curve skeleton. Each skeleton node possesses its corresponding surface information. We generate the skeletons of source and target mesh respectively. Then we match these two skeletons to establish the correspondence between the skeleton links. Our skeleton matching scheme employs the clustering of semantically similar links and spatial information, which is less sensitive to the skeleton connectivity and more robust than previous methods. For mapping the segment boundaries of the source to the target mesh, we take into account not only the global locations on the parts of the source mesh, but also the local surface feature of the segment boundaries, aiming to obtain a segmentation on the target mesh that is as consistent as possible to the source segmentation. First, we apply the MSP function proposed in [HJWC] to the source and target skeletons. The MSP function is a scalar function that represents the local volume information of the object. It is help for analyzing the global locations of the segment boundaries. We first associate the MSP variation with the both sides of links for source and target skeletons to determine the locations of boundaries on the target links then the points of the boundaries are mapped to the target through the local parameterizations, and finally the boundaries on target mesh are refined by considering the shape and local surface feature of the source segment boundaries.

1.1 Contribution

The main contributions of this work are as follows:

- The skeleton matching algorithm we propose for matching two curve skeletons has some advantages over the previous methods, and produces better matching results.

- We propose a novel segmentation transfer algorithm aiming at producing the segmentation on target mesh that is consistent to the segmentation on the source mesh, which involves local volumetric analysis, boundary mapping through parameterization, and boundary refinement.

1.2 Organization of the thesis

The following chapters are organized as follows. Chapter 2 reviews the previous researches including shape correspondence and mesh segmentation, and discusses the consistent mesh segmentation methods in section 2.2.1. Chapter 3 introduces the MSP function, MSP gradient function and MSP skeleton which are used in our approach. Chapter 4 and 5 describe our skeleton correspondence and segmentation transfer algorithms. Chapter 6 shows the experimental results, timing information, and the comparison to the Electors voting skeleton correspondence and the segmentation benchmark. Finally, Chapter 7 gives a summary of our algorithm and discusses the limitations and future works.

CHAPTER 2

Related Work

We review the related work in the areas of mesh segmentation and shape correspondence.

2.1 Shape correspondence

Shape correspondence establishes a meaningful correspondence between geometric elements of given objects. The types of elements includes meshs primitives, feature points on the surface, skeletal features, or local parts of the shape, and the types of corresponding relations may be one-to-one, one-to-many, or many-to-many. Figure 2.1 shows an example of one-to-one correspondence between the surface feature points of the two objects. Since the objects may have large difference in poses, shapes, and surface details, establishing the correspondence based on the similarities of local geometric information is uneffective. We must need higher-level descriptors which can describe the global structures of the objects and semantics of the parts to handle this problem.

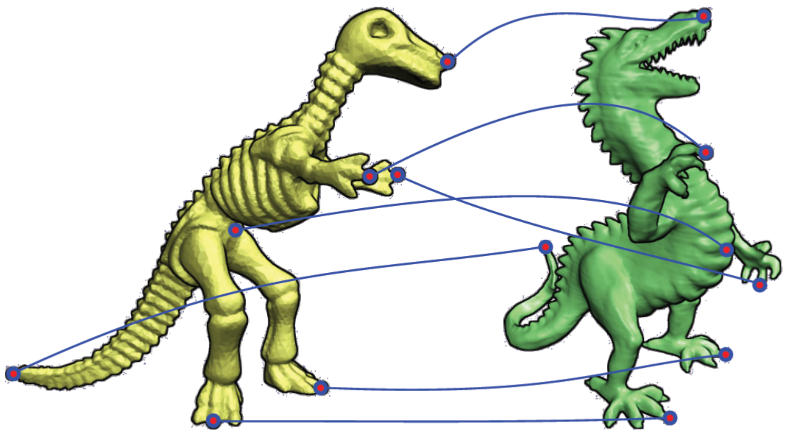


Figure 2.1: An example of shape correspondence [vKZHCO10].

According to the survey presented by Kaick et al. [vKZHCO10], there are several different categories for shape correspondence problems, for example, similarity-based correspondence, rigid alignment, non-rigid alignment, and time-varying registration. The skeleton matching problem is a part of the similarity-based correspondence. In this section we review the previous work on similarity-based correspondence.

Graph matching approaches Graphs can high-levelly describe the global structure of the objects. Graph nodes represent the specific parts of the objects and graph edges represent the connective relations between the parts. The graph matching methods used to establish the correspondence of the objects are intuitive and effective. Our skeleton matching method also adopts this strategy. In the following we briefly introduce graph matching problem.

Given two graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, where N_1, N_2 denote the graph nodes and E_1, E_2 denote the graph edges. The exact graph matching of the G_1 and G_2 is to find a isomorphism $f : N_1 \rightarrow N_2$, such that $(a, b) \in E_1$ iff $(f(a), f(b)) \in E_2$, where $a, b \in N_1$ and $f(a), f(b) \in N_2$. This problem has a solution (f exists) only when the two graphs have the same number of nodes and edges, which is almost impossible in practice due to the different number of parts between the objects or the different setting of graph extraction scheme. So the inexact graph matching is introduced aiming to find the best node correspondence to the nodes between two graphs. A node of one graph may map to a node, a null node (which represents it has no correspondence), or a group of nodes of another graph. Some feature information can be associated with nodes and edges to form attributed graphs. And the best correspondence means that it optimizes a specific objective function that estimates the similarity between the two attributed graphs.

In recent years there are some shape correspondence methods using inexact attributed graph matching techniques. The common graphs used for representing the objects are usually obtained by the skeleton extraction [SSGD03, XWLB09, ATCO$^+$10] or Reeb graph construction [HSKK01, TS04, BMSF06, TVD09].

Hilaga et al. [HSKK01] constructed the multi-resolution Reeb graph and attached AGD value and local region area to each graph node. Shape correspondence is obtained by maximizing geometric similarities between two graphs. Teng et al. [TS04] extended the framework of [HSKK01] by joining more geometric attributes and using spatial configuration constructed by principal component analysis (PCA). It is not robust since the local coordinate established by PCA are often not consistent. Sundar et al. [SSGD03] performed greedy and depth-first search scheme to find bipartite graph between the two acyclic skeletons which has maximum cardinality, minimum weight. Biasotti et al. [BMSF06] constructed the partial correspondence by finding the maximum common sub-graph between the two Reeb graphs.

The above methods suffer from some common problems: (1) Sensitiveness of the topological difference. That is, the matching performance strongly depends locally on the graph connectivity, and can easily produce incorrect match in the presence of noise and the topological difference at junction nodes. (2) Symmetry switching problem [ZSCO$^+$08]. Some nodes which represent the same semantic parts have similar geometric information, such as left/right hands in a human model. These parts are often symmetric. Lacking spatial information easily cause reverse correspondences. Figure 2.2 shows an example of incorrect matching due to the above two problem.

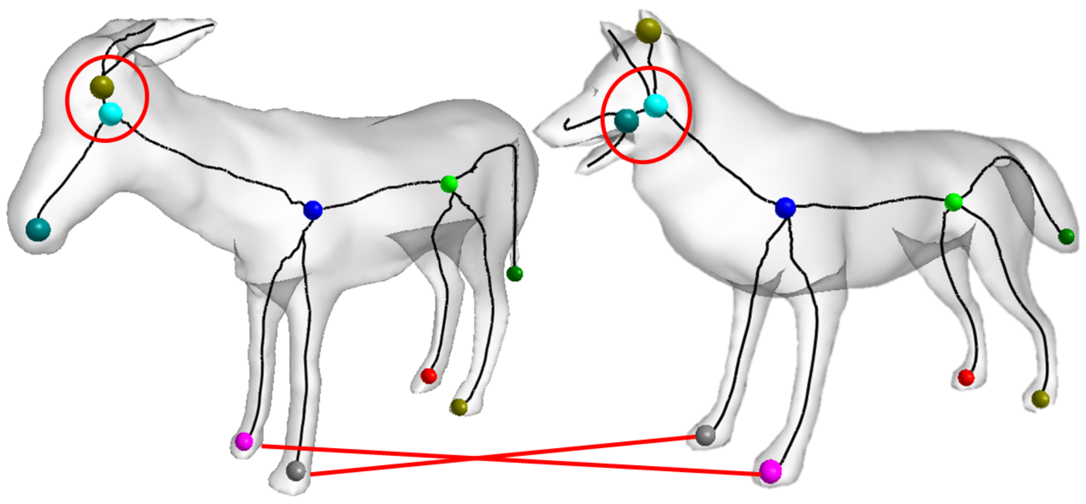


Figure 2.2: A skeleton correspondence example for two four-legged animals, which has symmetry switching problem on the front legs and mismatches on the heads due to the topological difference.

Some methods have been proposed to solve these issues. Xu et al. [XWLB09] observed that the junction nodes often have topological difference (see figure **??**) and applied the merging scheme to each individual skeletons before the matching process. However, it is still not robust since the merging scheme does not take into account the information of the other skeletons. Tierny et al. [TVD09] presented the idea of Reeb patterns and designed a structural signature to describe the terminal and junction nodes without considering the sequence of bifurcations. This can reduce the topological difference problem near the extreme parts of the objects. Au et al. [ATCO$^+$10] proposed a skeleton matching method which handles both topological difference and symmetry switching problems. Instead of traversing the graph along the connectivity, they employed the combinatorial search and local topology consistency test to filter out poor correspondences which have large topological difference. For symmetry switching problem, they transformed the skeletons into pose-normalized space using multidimensional scaling [EK03], and then checked whether the correspondence occurs global flipping in the spatial configurations.

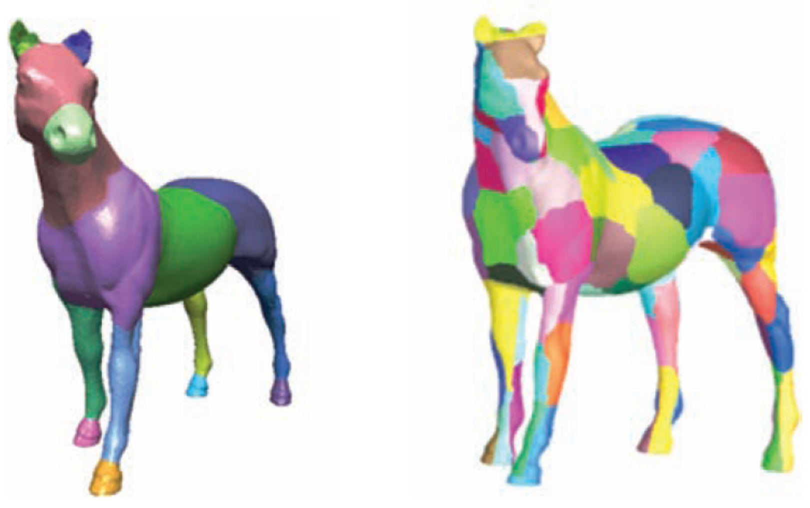
Deformation-driven approaches Zhang et al. [ZSCO$^+$08] proposed a deformation-driven combinatorial search algorithm for a correspondence between shape extremities. For each possible correspondence, one model is deformed so as to align the corresponding feature pairs and the deformation distortion is measured. This method is robust and it can deal with symmetry switching problem. However, it is time consuming due to the costly deformation computation.

Voting approaches Recently, voting schemes have been developed to establish shape correspondence [ATCO$^+$10, LF09]. In the voting scheme of Au et al. [ATCO$^+$10], electors are found by a combinatorial search equipped with a cascade of pruning tests to filter out different types of bad correspondence, and then vote on individual feature-to-feature matching to establish the final correspondence. Lipman and Funkhouse [LF09] proposed a Mobius voting correspondence algorithm for nearly-isometric shapes. The algorithm supports shapes with different poses and with large variation, but suffers from the high cost of computation.

The motivation for using the voting scheme is that it is hardly to design an objective function which ensures the best correspondence can be found. So the combinatorial search and pruning test are performed to list several possible correspondences. Since the best matching pairs must appear many times among the most of the possible correspondences, selecting the pairs with high occurrences to synthesize the final correspondence result is reasonable and effective.

2.2 Mesh segmentation

Mesh segmentation algorithms can be classified as two types: surface-type and part-type, as show in Figure 2.3. Surface-type algorithms aim to segment a given mesh into several patches that satisfy properties, such as planarity and disc-like. Part-type algorithms, on the other hand, segment the mesh into meaningful or semantic components. This is similar to what human perception tends to do. See [AKM$^+$06, APP$^+$07, Sha08, CGF09] for comprehensive surveys and comparisons.



(a) Part-type segmentation. (b) Surface-type segmentation.

Figure 2.3: Two types of mesh segmentation [Sha08].

Since our mesh segmentation transfer mainly considers part-type segmentation, we focus on this type of segmentation. These algorithms can be classified into: (1) Region growing approaches, (2) Hierarchical clustering approaches, (3) Iterative clustering approaches, (4) Spectral analysis approaches, (5) Skeleton-based approaches, (6) Boundary extraction approaches, (7) Critical points approaches. Some algorithms which are hardly to be classified to above categories will be introduced in other approaches section. After that, we discuss the algorithms which can produce consistent mesh segmentation in section 2.2.2.

2.2.1 Traditional mesh segmentation

Region growing approaches Region growing scheme is a greedy algorithm maintained by a priority queue. It starts from a source face, calculates the costs for merging the adjacent faces, and inserts them into priority queue. For each step the region grows increasingly by adding face which has the lowest cost, until priority queue is empty or specified stop conditions are met. A common variation of this scheme is multiple source region growing which uses multiple source faces and grows multiple regions simultaneously. Page et al. [PKA03] proposed a segmentation algorithm based on multiple source region growing and minima-rule [HR84]. It formed boundaries at the locations with low curvatures. This method is sensitive to surface noise and often produces jagged boundaries.

Hierarchical clustering approaches Hierarchical clustering scheme first regards each face of the mesh as an individual cluster. For each iteration it selects the adjacent cluster pair that has lowest merged cost to merge until a specified number of clusters or stop conditions are reached. The overall process can be described as a binary tree constructed in the bottom up scheme.

Attene et al. [AFS06] applied hierarchical clustering algorithm to a polygon mesh in order to dividing it into a set of clusters that are fitted with some specify primitives including planes, spheres and cylinders. Gelfand et al. [GG04] proposed a novel method to segment object into slippable components which are rotationally and translationally symmetrical. The above two methods are especially suitable for handling CAD industrial models. On the contrary they produce poor results for complex, general computer graphics models.

Iterative clustering approaches Iterative clustering schemes iteratively search the best representative points of clusters to find the optimal clustering set which minimize the specified distance metric. The basic of the approach is the K-means algorithm [Llo82] in which user must specify the number of clusters.

Shlafman et al. [STK02] used K-means algorithm to produce mesh segmentation. The distance of two arbitrary polygons was defined as the shortest path distance computed by summing up the geodesic and angular distance in the dual graph. Katz et al. [KT03] extended the framework of [STK02] and developed a hierarchical segmentation scheme. They introduced the concept of probability to construct the fuzzy regions between parts, and used graph cut [BK04] to extract boundaries following the surface concave features.

Spectral analysis approaches Spectral analysis approaches use graphical concepts to describe to connections between the elements of an object (vertices or faces) and record them in a Laplacian matrix. The object is embedded into the space R^k using first k eigenvectors. According to Polarization theory [BH03], in this embedding space points with high affinities will be grouped closer. On the other hand, they will be repulsed further if they have low affinities.

Liu et al. [LZ04] constructed the affinity matrix according to the distance metric defined by [KT03]. They achieved segmentation by employing K-means algorithm to cluster faces which have closing coordinates in the embedding space. Zhang et al. [ZL05] improved this method and developed a mesh segmentation algorithm via recursive bisection. A novel sample mechanism and Nystroms method [FBCM04] helped to largely reduce the time and space complexity needed for constructing the affinity matrix. Recently, Liu et al. [LZ07] extended above methods and proposed a segmentation algorithm by analyzing the outer contours of the object. Two different operators, called structural and geometrical operator, were used for spectral projection. Comparing to [ZL05], this method was much easier to select two faces which belong to different parts.

Skeleton-based approaches Skeletons are 1-D structures that can describe the global shape and topological structure of the objects. We can employ skeletons to develop effective mesh segmentation algorithms.

Li et al. [LWTH01] proposed a skeleton-based segmentation method based on volume variation analysis. For each end of the skeleton link, they defined a sweeping plane vertical to local link direction and move it along the link. The cutting section formed by plane can be regarded as local volume information of objects. They generated boundaries at the locations which has significant volume change. Reniers and Telea [RT07, RT08] observed that skeletons have junction nodes at the locations of the intersection between parts and presented a method to separate these parts using geodesic contours.

The qualities of this type of segmentations are mainly determined by their skeletons. The inappropriate skeletons which have small noise links or inaccurate link directions usually influence the segmentation results dramatically.

Boundary extraction approaches Lee et al. [LLS$^+$04, LLS$^+$05] proposed a method to identify candidate boundaries of the objects, and segmentation results were deduced from these boundaries. According to the minima-rule [HR84], they first collected regions whose curvatures are under a given threshold. After shrinking, branches deletion and extrapolation they obtained several candidate segment boundaries. Then the part salience criteria [HS97] was used to exclude boundaries which produce insignificant object parts.

Critical points approaches This type of approach analyzes geometry and shape of the object, places critical points on each prominent location of the object, and processes segmentation such that each prominent part contains only one critical point.

Katz et al. [KLT05] proposed a hierarchical segmentation algorithm based on core extraction, which can produce pose and proportion invariant segmentation. The critical points which correspond to the prominent parts were selected by using multi-dimensional scaling (MDS) [EK03] transformation. Lin et al. [LLL07] proposed another approach that locates critical points which have local maximum average geodesic distance (AGD) values. The main problem of these methods is that they can only be applied to objects that has significant core portion. Hence its not suitable for objects with overall convex shape, such as human head or CAD models.

Other approaches Some other approaches that are hardly classified into the above categories are also important, thus we describe them briefly.

Mortara et al. [MPS$^+$04] proposed a special segmentation algorithm which can detect the tubular components of the object. By identifying all these characteristics the core parts of the object are revealed.

Lai et al. [LHMR08] proposed a segmentation method based on random walk algorithm which computes the probabilities of one face reaching to each start faces by solving a linear system. The segmentation was obtained by assigning the label with highest probability to each face. Similar to iterative clustering, the qualities of the segmentation results are easily influenced by the start faces.

Golovinskiy et al. [GF08] proposed a new surface function called partition function which describes the probabilities of one edge lied on the segment boundaries. The partition function was computed by accumulating multiple randomized segmentation results. The method is robust to different poses, small surface noises, and tessellations.

Chen et al. [CGF09] provided the Princeton segmentation benchmark which contains a date set with large segmentation results generated by humans and some recent segmentation algorithms. Four metrics were proposed to evaluate how consistent the given two segmentations are by analyzing the properties of the boundaries and segments. They provided comprehensive reports which judge the degree of similarities between the human ground truth and the results of state-of-the-art methods.

Kraevoy et al. [KJS07] developed a modeling tool called Shuffler which produces a new model by exchanging the components between the two models. They used convex decomposition to divide the objects into several meaningful, interchangeable components. Shapira et al. [SSCO08] proposed the shape diameter function (SDF) and applied it to mesh segmentation. SDF is a scalar function which describes the local volumetric thickness for each inward normal direction of vertex. Since the SDF values are almost consistent between a set of same objects with difference pose or same class of objects, it is much easier to achieve consistent segmentation.

2.2.2 Consistent mesh segmentation

There are some approaches which demonstrate consistent segmentation for a set of objects. These methods are most relative to our segmentation transfer algorithm.

Golovinskiy and Funkhouser [GF09] proposed a method which simultaneously produces consistent segmentation results on a set of similar objects. They used the rigid alignment [BM92] to establish the face correspondence between the set of objects, then achieved high-quality consistent segmentations by employing a hierarchical clustering scheme that considers both the geometric features of individual objects and the similarity of clustering results between the set of objects. The method encounters problems for objects that are largely different in shape or poses since the rigid alignment is not able to correctly align the objects.

Kalogerakis et al. [KHS10] proposed a novel approach that simultaneously processes segmentation and labeling on a set of objects. The labeling problem was stated as a Conditional Random Field model which optimizes an objective function. The objective function consists of unary terms and pairwise terms that measures the consistency of faces with labels. The objective function was learned from a set of labeled training objects and then applyied to the other same type of objects to achieve consistent segmentations. This approach employs a large number of features commonly used for segmentation cues, so that it can handle various segmentations for a wide range of objects. But the training process is time consuming, and it still can not well handle the objects that have large topological difference.

CHAPTER 3

Introduction of MSP function

Our skeletons are generated based on the MSP function. We briefly review MSP function, MSP gradient function and MSP skeleton in this section. For more details, please refer to [HJWC].

3.1 MSP function

The MSP function is a scalar function defined on the mesh surface for measuring the local volume information. For a surface point p, the MSP value is computed as the minimum perimeter of the planar slices passing through p and parallel to the normal of p. The minimum slice perimeter can approximate the local volume at the point p. Figure 3.1 shows an example for computing the MSP function. To trade off accuracy for speed, the planar slices are computed in a discrete manner. We compute the MSP value for each vertex on the mesh. Figure 3.2(a) shows the results of MSP function for some objects. With the increasing of MSP value, the surface is colored from blue, green, to red. We observe that the parts with different volumes can be distinguished easily according to the MSP values.

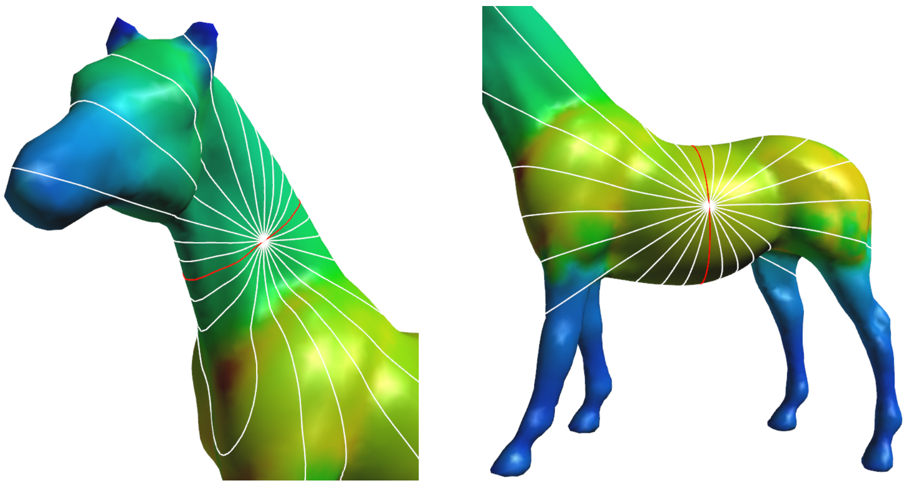

(a) (b)

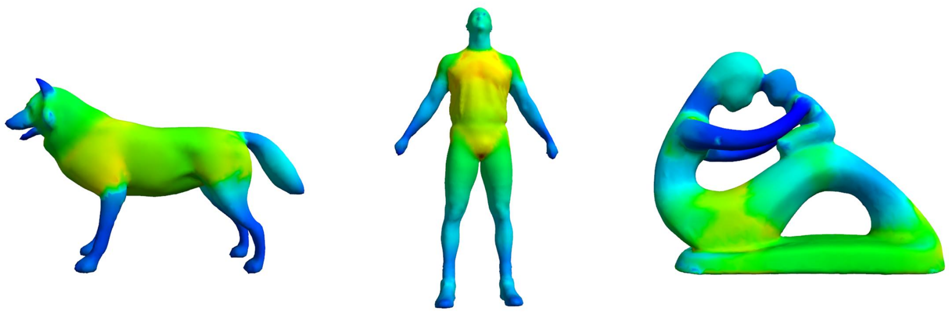
Figure 3.1: MSP function computation. The planar slices are computed in a discrete manner. Minimum slice is colored to red.

3.2 MSP gradient function

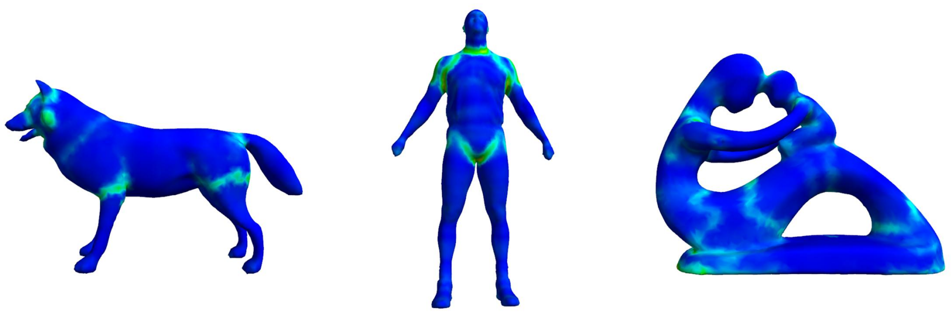
The MSP gradient function represents the change rate of the local volume(MSP) in the neighboring region. For a surface point p, the MSP gradient value is computed similarly to the method by Alliez et al. about the evaluation of curvature tensor [ACSD+03]:

$$\nabla MSP(p) = \frac{1}{|B|}||\sum_{e \cap B}(MSP(e_i) - MSP(e_j))\vec{e}||$$

where B is the neighboring region of p which is specified by user, $e \cap B$ is the part of an edge e inside B, e_i and e_j are end vertices of the edge e, and $\vec{e}$ is a normalized vector along e. Figure 3.2(b) shows the results of MSP gradient function for some objects. We observe that the MSP gradient values raise at the places where MSP values change largely. These places are usually the junctions of the parts.



(a)



(b)

Figure 3.2: (a)The MSP function results, (b)The MSP gradient function results.

3.3 MSP skeleton

The MSP skeletons which are extracted directly from 3D models based on MSP function and a new skeletonization framework. First, The MSP value is used to transform all the vertices toward the interior of object in order to obtain a nearly zero-volume shrunk mesh. This shrinking mesh preserves the same edge connectivity as the origin mesh. Then perform a greedy LOD skeletonization framework to degenerate the topology of the shrunk mesh until the 1D curved skeleton is formed. Instead of using the traditional edge-collapsed operator, the edge swap operator is used for changing the edge connectivity without simplifying the geometry, with a compact error metric defined to measure the deviation of the shrunk mesh vertex, a curved skeleton can be yielded with dense nodes which corresponding to a set of shrunk vertices. In order to obtain high-quality curved skeleton, the smoothing operation is performed to skeleton links and then small skeleton branches due to local surface details are removed.

CHAPTER 4

Skeleton correspondence

In this chapter, we present the skeleton correspondence algorithm for the source and target skeletons, which establishes the relation between their skeleton nodes and skeleton links.

4.1 Overview

Our skeleton correspondence algorithm is based on graph-matching which addresses the (1)sensitiveness of the topological difference and (2)symmetry switching problem mentioned in section 2, and builds the meaningful mappings between the nodes and links of the source and target skeletons. The mapping relation of the nodes and links may be one-to-one, one-to-many or many-to-many. This is different from most of the previous works [HSKK01, BMSF06, ATCO$^+$10] which merely build the one-to-one mapping between the nodes.

The matching process of the two skeletons can be expressed as a correspondence tree, as shown in figure 4.1. The root represents the initial state which has no matched pairs. In the beginning, we select a junction node pair in two skeletons with minimal cost, denote it as start node pair, and generate a tree node attached to the root of the correspondence tree to represent this matching state. We will traverse the skeleton graphs and expand the matched ranges from the start node pair. If there are several junction node pairs with similar low costs, currently we are not sure which one form the best correspondence result, so the graph-matching is processed respectively for each node pair(there are multiple tree nodes attached to the root). The definition of the node pair cost is described in section 4.2. Later, we process the link correspondence described in section 4.3, to match the links connecting to the start node pair. The link clustering and spatial orientations help to avoid the symmetry switching of the link correspondences. For each matched link pair a new matched node pair is derived for graph traverse. Instead of directly choosing the other end nodes of the links to form the matched node pair, we trace along both sides of links to find the best next node pair to address the problem of local topological difference between the two skeletons. When the link correspondence is complete, a new tree node is generated to represent the current matching state and attaches to the previous tree node. If there are several plausible link correspondences, the correspondence tree occurs branching. We traverse the skeletal graphs in depth-first search(DFS) manner. Each tree node has a stack which contains the unprocessed node pairs. After the link correspondence we obtain some new node pairs and push them into the stack. For each round we pick a tree node whose stack is not empty, pop a node pair and recursively perform the above matching steps. After completing the matching process(all the stacks of the leaf node are empty), each leaf represents a candidate final correspondence result. We choose the leaf node with minimal cost as the final correspondence result of the two skeletons. The details are presented in section 4.4.

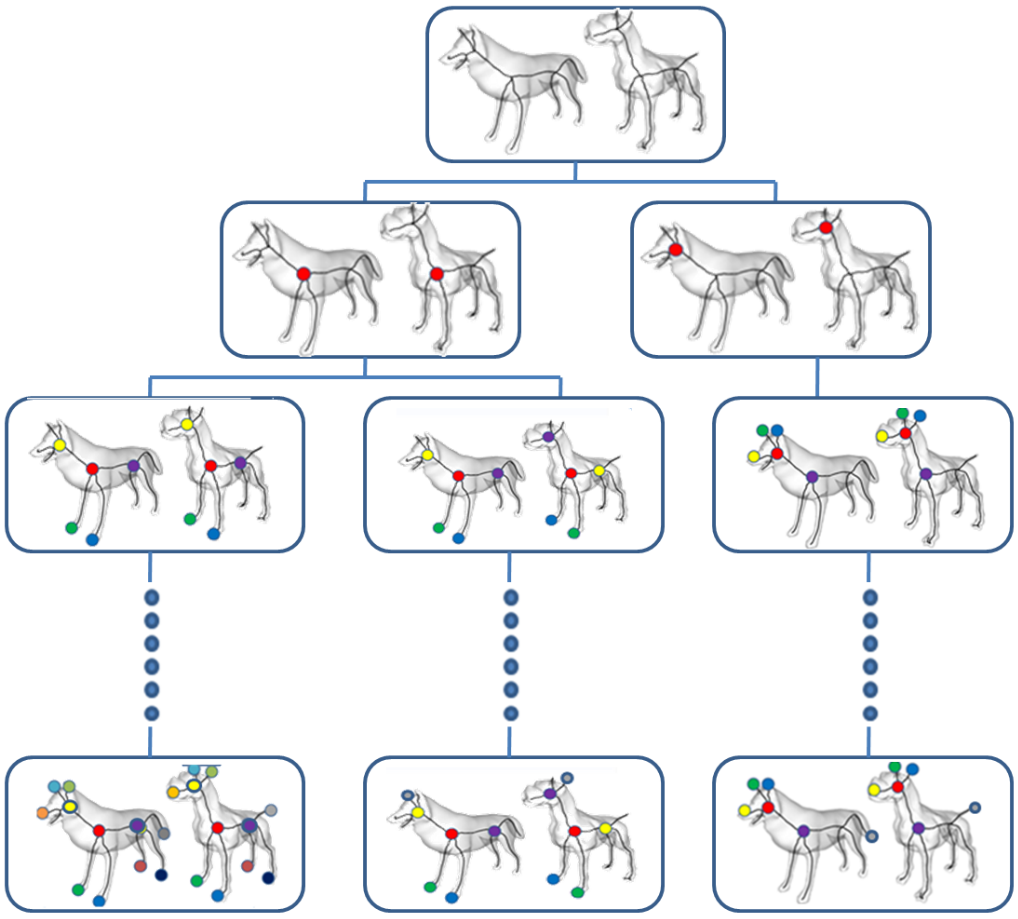


Figure 4.1: Correspondence tree.

4.2 Node pair correspondence

Consider a node pair (n_s, n_t), where n_s and n_t are one of the nodes of source and target skeletons, respectively. Assume n_s has n connecting links and n_t has m connecting links. Denote $L_s = \{s_1, s_2, ..., s_n\}$ and $L_t = \{t_1, t_2, ..., t_m\}$ as the lists of links which connect n_s and n_t. We define the C_{node} which combines two different types of features to estimate the dissimilarity of a node pairs as the following equation:

$$C_{node} = C_{center}(n_s, n_t) * C_{struct}(L_s, L_t) \tag{4.1}$$

The C_{center} is defined as:

$$C_{center}(n_s, n_t) = |AGD(n_s) - AGD(n_t)| \tag{4.2}$$

where $AGD(n_i)$ is the average geodesic distance of the n_i computed by averaging the AGD values to the corresponding vertices. The AGD values of the mesh vertices are first normalized to [0,1]. $AGD(n_i)$ is the geometric feature and represents the centricity of a node n_i.

Denote $SG(L_s) = \{g_1, g_2, ..., g_n\}$ as the size of the sub-skeleton list corresponding the L_s. It represents the structural feature of n_s. A sub-skeleton starts from a connected link of the node and is built by traversing the skeletal graph in BFS manner. Figure 4.2(a) shows the sub-skeletons of node n_i that each sub-skeleton is drawn a different color. Each $g_i \in [0, 1]$ of the $SG(L_s)$ is the total length of the links contained in the sub-skeleton and are normalized by the total length of all the links in the skeleton. If there are some different links which form the same sub-skeleton, then the sub-skeleton must has loops, as shown in figure 4.2(b). In this case g_i is divided by the number the links which share this sub-skeleton.

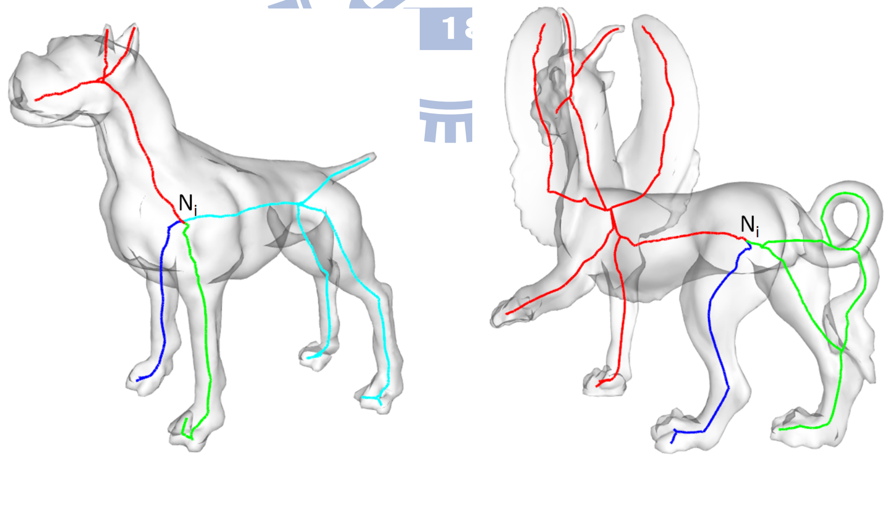


Figure 4.2: The sub-skeleton of a node n_i:(a)without loop, (b)with loop.

We define $SG(L_t) = \{h_1, h_2, ..., h_m\}$ in the similar way to $SG(L_s)$. The items of the $SG(L_s)$ and $SG(L_t)$ are sorted in the decreasing order. Assume $n \geq m$, then the C_{struct} term of the equation 4.1 is defined as the difference between $SG(L_s)$ and $SG(L_t)$:

$$C_{struct}(L_s, L_t) = \sum_{i=1}^{m} (g_i - h_i)^2 + \sum_{i=m+1}^{n} (g_i)^2 + \epsilon \tag{4.3}$$

If $n < m$, we switch the roles of n_s and n_t. We add a small constant ϵ to avoid the zero value(e.g., the c_{struct} of two terminal node is zero, it causes the c_{node} also be zero), ϵ is typically set to 0.01.

We denote a threshold t_{in} for choosing the start node pair. If a node pair whose C_{node} is smaller than t_{in}, we set it as the start node pair. A small tin is sufficient for generating correct skeleton correspondence, which is typically set from 0.02 to 0.03.

4.3 Link correspondence for a node pair

To compute the link correspondence for the links connecting to the node pair (n_s, n_t), we use the information induced from the set of links at the node pair, including clusters of links, their spatial information and relative orientation between links.

4.3.1 Link clustering

Different parts of objects may have the same semantic meanings, such as front/back legs and pairs of ears. These parts have similar shapes and geometric features. In our method, we rely on the volume characteristic of parts for clustering them. Later on, when the matching is performed, using clusters of links not only improves the correspondence results but also simplifies the process.

For each link connecting to the node n_i, we first compute the longest path starting from this link in its sub-skeleton. Then each path is divided into k pieces, where k is given by a user. A typical value of k is from 5 to 10. For each piece the average MSP value is evaluated. These k values are then assembled as a k-dimensional vector, namely volume descriptor of the link. After that, the error of two links with volume descriptors, ℓ_i and ℓ_j, is computed as

$\|\ell_i - \ell_j\|_2$ (L2-norm). We use hierarchical clustering to cluster the links connecting to the node n_i. figure 4.3 shows an example of link clustering. The links having similar volume descriptors are clustered together and are drawn the same color.

The link clustering are processed to n_s and n_t of the node pair, respectively.

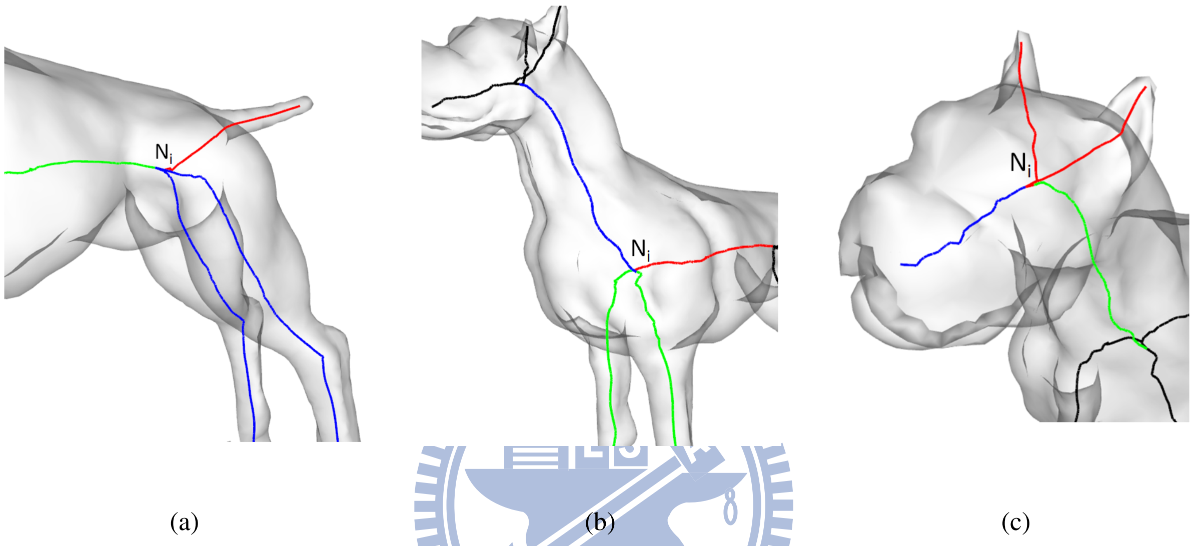
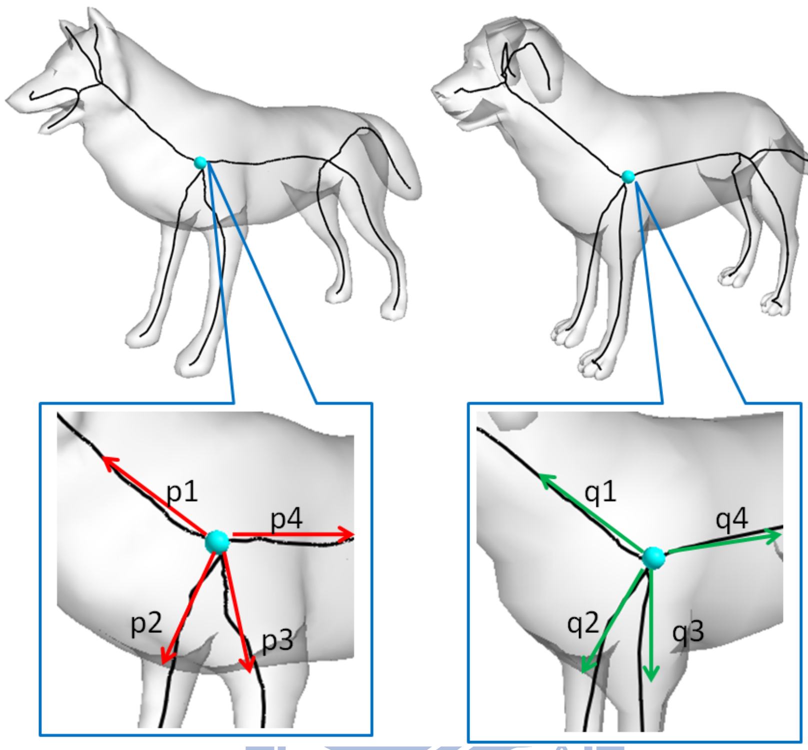


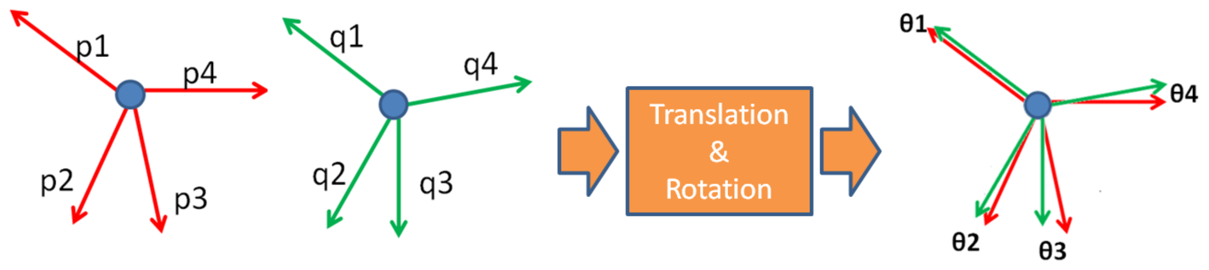
Figure 4.3: The results of link clustering.

4.3.2 Link correspondence

We observe that for two correctly matched node pair, the orientation of links at the local regions of both node pair is similar to each other. Based on the observation, we exploit the spatial information to design the link matching algorithm. Consider an example in figure 4.4. Now we have a matched node pair (n_s, n_t) at the chests of wolf and dog. There are four links connecting to n_s and n_t which belong to trunks, necks, front left legs and right legs. So there are $4! = 24$ possible correspondences for these links. Our goal is to construct the vectors which best fit the local directions of the links, and then find the best transformation to minimize the sum of the angle difference for each corresponding link pair, as shown in figure 4.4(b).



(a)



(b)

Figure 4.4: Link correspondence at a node pair. The local orientations of links are similar: (a)generate local vectors for each link, (b)find the transformation which minimizes the angle difference for the all link pairs.

To compute the pose-insensitively link vectors, We had adopted the least square multidimensional scaling [EK03] to perform pose-normalize for the object. It straightens the bent links. But we find it has some drawbacks: (1) the normalized directions are mainly guided by the beginning directions of the original skeleton links. So the noise around the start node of the links will disturbed the normalized results, (2) the normalized links which far from the center of the object easily form bunches. In conclusion, the original spatial orders and relative orientations may not be retained after processing the multi-dimensional scaling.

We observe that the directions of links are not influenced much for different poses within the local region of the node. We use the idea of inscribed ball to define the valid range of the links. The average distance of the mesh vertices corresponding to the node is computed as the approximate radius of inscribed ball. And the local vectors of the links are constructed by cutting the links at the range of radius.

After constructing the local vectors of the source and target, we use the shape matching technique [MHTG05] to find the rigid transformation which minimize the distance between the corresponding vector pairs. We briefly describe this as follows: Denote $P = \{p_1, p_2, ..., p_n\}$ as the link vectors of source and $Q = \{q_1, q_2, ..., q_n\}$ as the corresponding link vectors of target which p_i matches with q_i, $1 \leq i \leq n$. find the rotation matrix R which minimize:

$$\sum_{i=1}^{n} ||Rq_i - p_i||^2$$

Refer from [MHTG05], first the optimal linear transformation matrix A is computed:

$$A = (\sum_{i=1}^{n} p_i q_i^T)(\sum_{i=1}^{n} q_i q_i^T)^{-1} = A_{pq} A_{qq}$$

The second term A_{qq} is a symmetric matrix which represents only scaling and can be ignored. And the optimal rotation matrix R is the rotational part of the A_{pq} which can be computed by polar decomposition $A_{pq} = RS$, where $S = \sqrt{A_{pq}^T A_{pq}}$. Therefore the optimal rotation is $R = A_{pq} S^{-1}$.

We compute the rotation matrix R for each possible link correspondences (for the example of wolf and dog in figure 4.4. There are 24 possible link correspondences and rotation matrixes). Due to the spatial orientations of both sides are similar, correct link correspondence must has a small angle differences between each link pair. Recall $P = \{p_1, p_2, ..., p_n\}$ as the link vectors of source and $Q = \{q_1, q_2, ..., q_n\}$ as the corresponding link vectors of target which p_i matches with q_i, $1 \leq i \leq n$. Then the angle difference between a link pair (p_i, q_i) is defined as:

$$ang(p_i, q_i) = acos(dot(p_i, Rq_i))/\pi \tag{4.4}$$

where $dot(p, q)$ is the dot product of the vectors p and q. The spatial cost of this link correspondence is defined as the sum of the angle difference for each link pair:

$$C_{spatial}(P, Q) = \sum_{i=1}^{n} (ang(p_i, q_i))^2 \tag{4.5}$$

The bad link correspondences cause large spatial difference since it is hard to well fit all of the link pairs using only a rotational transformation.

Now we formally define the link corresponding cost. Assume that n_s, n_t have n, m connecting links and $n \leq m$. Denote $L_s = \{s_1, s_2, ..., s_n\}$ as the connecting links of the n_s and $L_t = \{t_1, t_2, ..., t_n\}$ as the connecting links of n_t such that s_i maps to t_i, $1 \leq i \leq n$. $P = \{p_1, p_2, ..., p_n\}$,$Q = \{q_1, q_2, ..., q_n\}$ are the corresponding local vectors of L_s and L_t. The cost for this link correspondence is defined as:

$$C_{link} = C_{struct}(L_s, L_t) * C_{spatial}(P, Q) \tag{4.6}$$

The cost of the link correspondence considers not only the local spatial configurations of the links, but also the more global information of skeletal structures. Note that here the sub-skeleton of L_s and L_t do not need to be sorted. It is different to the cost computing for node pair mentioned in section 4.2.

Then we describe how to use the link clustering information described in section 4.3.1 to filter out most of the bad link correspondences. Consider the case in figure 4.4, the wolf has a cluster of links to its front left and right legs, since they are semantic pairs which have similar volumetric distributions along the links. And the dog also has a cluster of links to its front legs. So we can first match the links contained in the two clusters, and then match the links belonging to necks and trucks. According to this way, the possible number of link correspondences is reduced from 4!=24 to 2!2! = 4. In practice, it may contain different number of clusters on the n_s and n_t. Due to lack of the knowledge about the parts of the objects, currently we cannot guarantee the match of the link clusters will be correct. We argue that for the same category of models, they should have the parts with same semantic meaning on the correctly matched node pair. So this strategy works well for most of our test results in practice.

Besides the link clustering information, the rotation matrix R computed in the above step also help to filter out the bad link correspondences. That is, the correspondence with symmetry switching problem. If the determinant of the R is negative(-1), it represents that the rotation matrix involves a reflection, which form a global flipping correspondence.

4.3.3 Next node pair searching

For each link pair, a new node pair is derived for graph traverse. A simple way is to choose the other end nodes of the links. But it is not robust since the local topological difference may occurr. We denote the other end nodes of the link pair as (n'_s, n'_t) and use equation 4.1 to compute its node pair cost. And a threshold t_n is set by the user. If the cost is larger than t_n, (n'_s, n'_t) will be considered to have significant error, which is not suitable to form the matched node pair. Therefore we adopt the following strategies to address this issue:

Node clustering Due to MSP skeleton has high resolution, some close junction nodes connected with short links that indicate slight difference in topology. If we take the more global view on the whole skeleton, these close nodes represent the same part of the object and can be clustered as a virtual single node. We define a distance threshold t_d as the 5% of the diagonal of the bounding box to the mesh. A junction node which can be clustered to n'_s or n'_t if the distance to n'_s or n'_t is less than t_d.

Our original approach processed node clustering to (n'_s, n'_t) only when its cost is less than t_n, but we found out that it could not correctly handle some cases such as in figure 4.5. Although (n'_s, n'_t) has low cost (both of them have similar AGD and sub-skeleton sizes), but one connects the left limb and the other connects the right limb. If we directly set them as matched node pair, they will cause wrong link correspondence. To address this situation, no matter what the original cost of the (n'_s, n'_t) is, we cluster all the junction nodes to n'_s or n'_t if the distances are less than t_d, regard them as a single node pair and compute the node pair cost. If the cost is less than t_n, we set them as matched cluster pair. The correspondence relations of the internal nodes in the clusters are many-to-many, and the internal links are set to unmatched since we imagine these links have been collapsed. In the following matching processes the cluster pair is regarded as a single node pair. The local vector construction (section 4.3.2) uses the average position of the internal nodes as the origin.

On the other hand, If both the costs of the original (n'_s, n'_t) and the cluster pair are larger than t_n, we will discard the clusters and process the link extension.

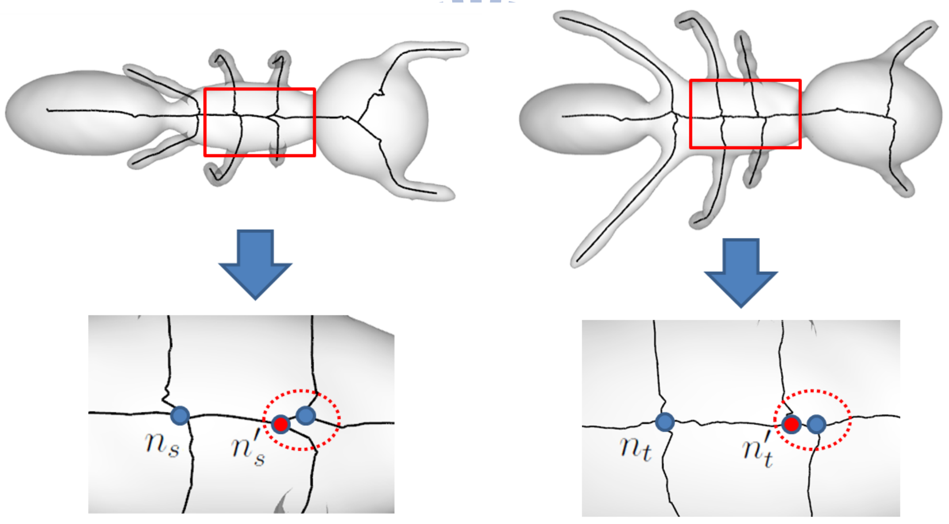


Figure 4.5: Node clustering. (n_s, n_t) is the current node pair and (n'_s, n'_t) colored red is the other end nodes of a link pair. We need to perform the node clustering indicated as the red dotted circles to address the topological difference.

Link extension If the costs of the (n'_s, n'_t) and its cluster are large than t_n, we search the both sides of the sub-skeletons to find the next node pair. For the current node pair (n_s, n_t) and link pair, we define the distance of a node pair in their sub-skeletons as the sum of length of the geodesic paths to the (n_s, n_t), the path length is normalized by the total length of the links of the skeleton. According to the distance in ascending order, we perform the node clustering for each node pair of the sub-skeletons in sequence until encounter a node pair or a cluster pair with cost less than t_n. We denote it as (n_{s0}, n_{t0}). Figure 4.6 shows an example. After the process, the link paths from (n_s, n_t) to the (n_{s0}, n_{t0}) are correspondence to each other. The nodes among the link paths (e.g., n_s in figure 4.6) and their branches are set to be unmatched.

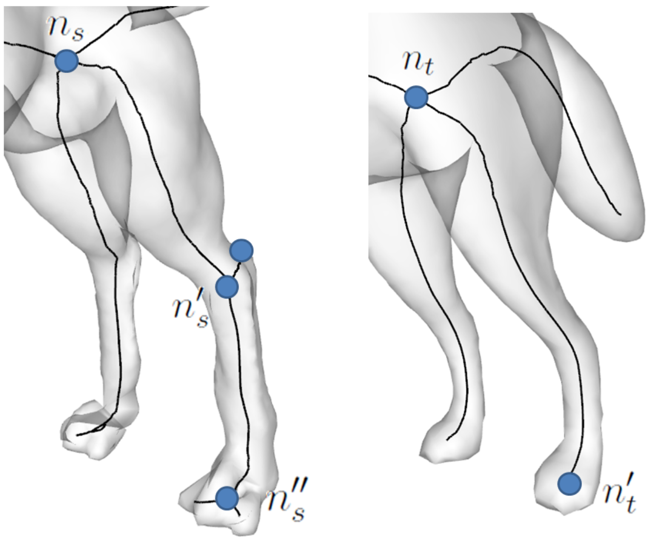


Figure 4.6: Link extension. (n'_s, n'_t) is the end nodes of a link pair, which has large cost due to the topological difference. After the searching, (n''_s, n'_t) is found as the suitable matched node pair.

4.4 Final correspondence result

After completing the correspondence tree searching, each leaf node represents a candidate correspondence of the two skeletons. We choose the best one among them as the final correspondence. Due to the multiple start node pairs, there may have some redundant leaf nodes which represent the same correspondence. We remove these redundancies to make each leaf node represent a unique correspondence.

We select the final correspondence result by using the node pair costs C_{node} and link costs

C_{link}. During the correspondence tree searching, these costs are computed and recorded on the tree nodes. Each leaf node contains a complete list of C_{node} and C_{link} for its correspondence results. Since the lower costs represent the better local correspondences. An intuitive method is to accumulate all these costs and select the leaf node with minimum. But this method fails when a bad correspondence which only has few matched node and link pairs. Although each cost is large, the sum of these costs is still smaller than the good correspondence which has more matched pairs. To avoid this problem, we keep track the maximum C_{node} and C_{link} during the correspondence processes, denote them as $C_{maxnode}$ and $C_{maxlink}$, and normalize each cost to [0,1] by dividing $C_{maxnode}$ or $C_{maxlink}$. We denote the lists of C_{node} and C_{link} of a leaf node as $\{cn_1, cn_2, ..., cn_n\}$ and $\{cl_1, cl_2, ..., cl_m\}$, then we define the cost of a leaf node as the product of these cost:

$$C_{leaf} = \Pi_{i=1}^{n} cn_i * \Pi_{i=1}^{m} cl_i \tag{4.7}$$

The leaf node has small C_{leaf} if its cost is small, or it has more matched pairs. We set the one with minimum cost as the final correspondence result of skeletons.

CHAPTER 5

Segmentation Transfer

In this section, we propose the segmentation transfer algorithm which maps the boundaries on the source mesh to the target mesh, to form the consistent segmentations.

5.1 Overview

After skeleton correspondence is established, we have fully correspondences of links between source and target skeletons. Since each link represents a specific part of the mesh, associating each boundary of source mesh with a link is an easy task. For each iteration we focus on one of the source link. If at least one boundary associated with this link, we transfer these boundaries to the part of target mesh which belongs to the corresponding target link. Once all the links have been processed, the segmentation transfer between source and target meshes is complete.

We first associate each boundary with the link. Then for each link to be processed, our segmentation transfer algorithm can be divided into four steps. First, in step one we utilize the MSP gradient to analyze the volume varying distributions along the source and target links, in order to locate the approximate positions of boundaries on the target mesh. In step two, we build cutting regions on the source and target meshes which the boundaries are expected belong within. In step three, we apply the cylindrical parameterization to these cutting regions, and map source boundaries to the target cutting regions in the parameterization domains. Finally, in step four we use the active contour models(snakes) to evolve the target boundaries locally, so that the geometric features of the source and target boundaries can be as similar as possible.

5.2 Associating boundaries with links

We use the method proposed in [AFS06] which is based on Principal Component Analysis to construct the fitting plane of each boundary. Then compute the intersection between the links and the plane. There would be one link intersecting with the plane. If there are more than one link intersecting with the plane, we select the link with the minimum distance to the centroid of the boundary. Therefore, each boundary would be associated with one link. On the other hand, a link would be associated with zero, one or more boundaries.

5.3 Volumetric analysis

After the link-boundary crossing test described in section 5.2, for each boundary an intersection point p is obtained on the link. We denote a normalized link ratio value which describe the position of p relative to the whole link as : $LR(p) = len(p_s, p)/len(p_s, p_t) \in [0, 1]$, where p_s and p_t are start and end point of the link, and $len(s, t)$ is the geodesic distance between s and t along the link. An intuitive idea is to use $LR(p)$ to produce boundary at the corresponding position of the target link. But it is not an effective way since the source and target meshes may have large differences. So we need some additional geometric properties which help to characterize the source boundaries.

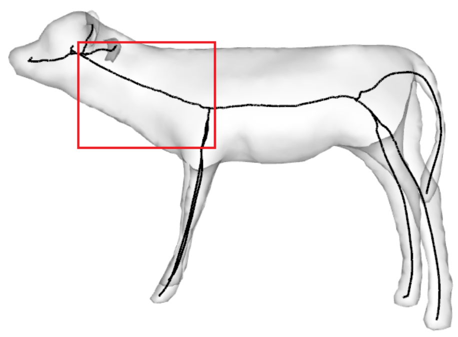
For the common geometric properties used in mesh segmentation methods, global properties such as average geodesic distance(AGD) are suitable for analyzing the whole meshes, and local properties such as curvature, dihedral angle are suitable for analyzing the local surface details. Skeleton links usually represent the object parts, so an intermediate level property which different to above is needed to analyze them. Volume information meets our requirements since it is insensitive to pose. And based on the observation of part-type segmentation, the boundaries are often located on junctions of two parts which has large volumetric variation.

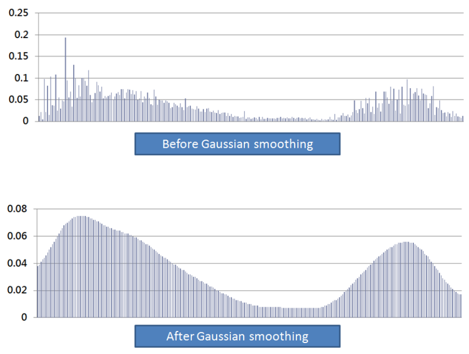
In this step, our goal is to find the positions (LR value) for producing boundaries on the target link. Our strategy is to first generate part-type boundaries based on volume analysis, set them as reference boundaries and compute the differences to the input source boundaries, and finally infer the most suitable positions on the target link.

We use MSP gradient function to describe the volumetric variation of the object parts. Due to each skeleton node associates with some mesh vertices, we can easily construct the distribution of MSP gradient along the source links. The MSP gradient along a link is smoothen by using Gaussian smoothing technique so as to eliminate redundant details and noise. Figure 5.1 shows an example of deer's neck. Figure 5.1(b) are the MSP gradient distributions before and after the Gaussian smoothing. Each local maximum value represents the maximum change of volume locally at the particular position on the link. We adopt the similar strategy to Katz's method [KT03]. For each local maximum a corresponding fuzzy region is constructed and a boundary is extracted from it. We collect the one-ring faces of all such vertices for the local maximum node to form the fuzzy region. If it does not form a ring, we gradually pick the nodes which are adjacent to the local maximum, and add the corresponding faces to the fuzzy region. This is repeated until the fuzzy region form a ring. After that, the dual graph of the fuzzy region is constructed and then graph cut [BK04] is applied for extracting the boundary, as shown in figure 5.1(c-d). Let a_i be a arc in the dual graph, e_i be the common edge of the two dual faces connected by a_i. The capacity of the arc a_i is defined as follows:

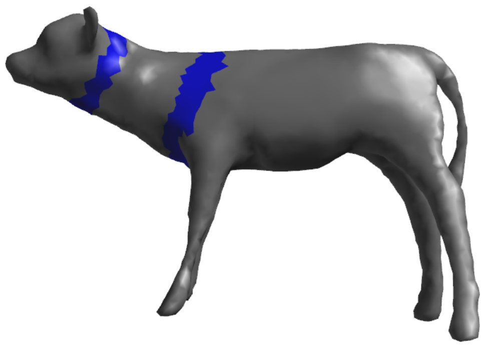
$$Cap(a_i) = (\frac{1.0}{\nabla MSP(e_i) + \epsilon}) * Len(e_i) \tag{5.1}$$

where $\nabla MSP(e_i)$ is the MSP gradient function of the edge e_i (the average of the two end vertices), $Len(e_i)$ is the length of e_i, ϵ is a small constant which prevent division by zero and is usually set to 0.001. Based on the capacity, the boundary will pass through the short edges which have large MSP gradient function. We denote this type of boundary as MSP gradient boundary. Figure 5.2 shows the MSP gradient boundaries on some objects, as well as the input source boundaries provided by user.


(a) Deer's neck.



(b) MSP gradient distribution.


(c) Fuzzy regions.

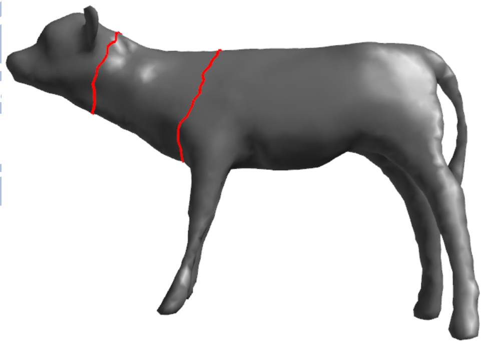

(d) Boundaries extracted by graph cut.

Figure 5.1: Volumetric analysis example.

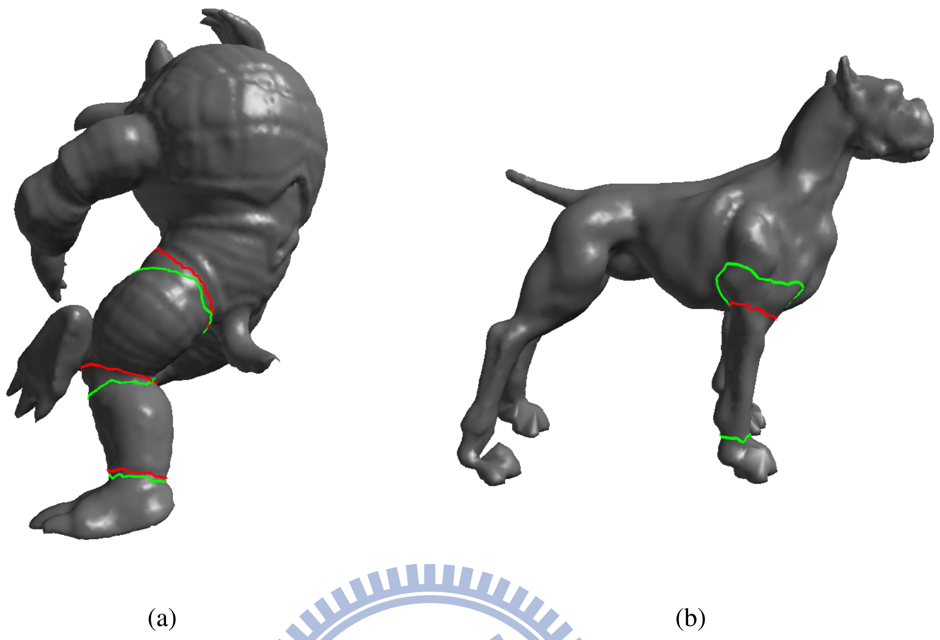


Figure 5.2: The MSP gradient boundaries and the input source boundaries. The MSP gradient boundaries are colored in red, and input source boundaries are colored in green.

After constructing the MSP gradient boundaries on the parts of source and target links, we establish the correspondence between these source and target boundaries. First, the LR value of each boundary is computed by using the fitting plane method (section 5.2). Then the boundary list is sorted according the LR value. Denote s_i is the i-th source boundary and t_j is the j-th target boundary. The boundary pair cost is defined as follows:

$$C_{boundary}(s_i, t_j) = \alpha * (LR(s_i) - LR(t_j))^2 + \beta * (\nabla MSP(s_i) - \nabla MSP(t_j))^2 \qquad (5.2)$$

where $\nabla MSP(b)$ is the MSP gradient function of the boundary b (the average of the boundary vertices). The mapping relations of the boundaries are one-to-one. And the order of the matched boundary pairs on both sides of the links must be consistent. We find the best correspondence by minimizing the sum of the boundary pair cost.

Now we are going to infer the boundary position on the target link. For each source boundary, we trace along the link to find the first matched MSP gradient boundaries in up and down direction. If no matched MSP gradient boundary is found, we use LR value of the start or end points (0.0 or 1.0) of the link. Then use the mapping relation to find the corresponding MSP gradient boundaries on the target link. As shown in figure 5.3, we compute the distance (LR value difference) between these boundaries. Then the LR value on the target link can be computed according to the same proportional relation.

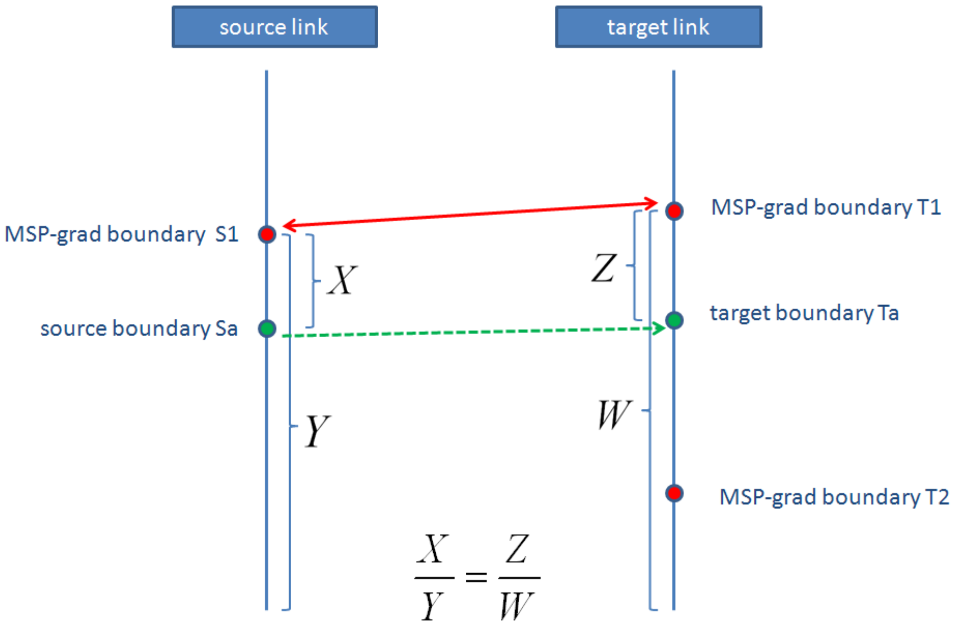


Figure 5.3: The position of the target boundary can be inferred from the adjacent matched MSP gradient boundaries.

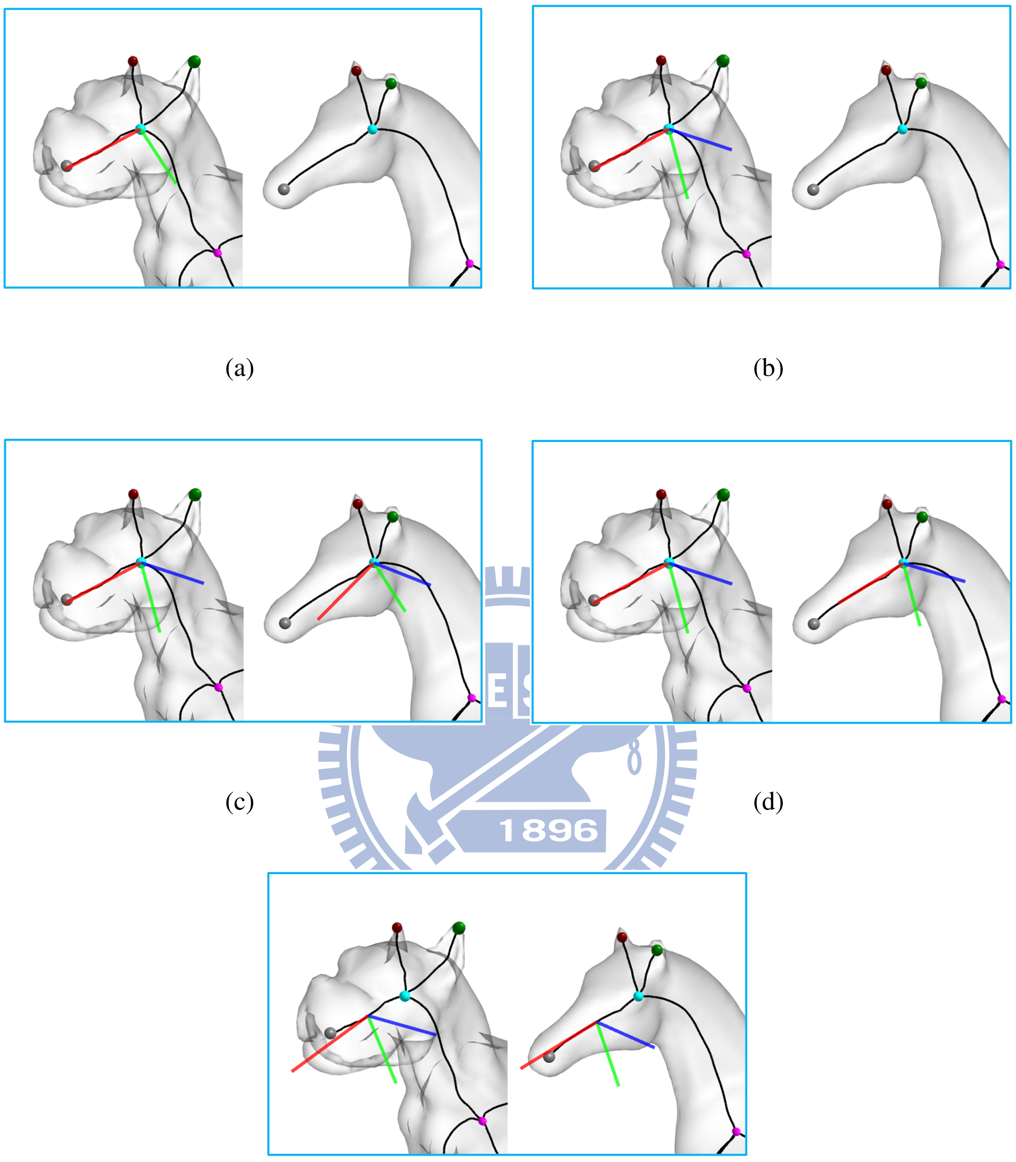
After the above process, for each source boundary we obtain the LR value on the target link. It represents its approximate location on the target mesh.

5.4 Cutting region construction

We have obtained the LR values which can infer the located points on the target link. However, it is not sufficient since the source boundary have some additional characteristics needed to be considered. We observe that the cross orientation of the boundary's fitting plane (section 5.2) and link is an important feature to globally describe the shape of the boundary. For example, some boundaries may cross the link vertically and form the short, straight shapes. And other boundaries may cross the link with non-vertical angles and produce more tilted, longer shapes. Another important feature is the surface extent covered by boundary. Some boundaries have small extents which mean the points of the boundaries are quite close to their fitting planes. And some boundaries may cover relatively wide extents so that they have large deviations to the fitting planes.

We take into account these two features to develop our algorithm which establishes the boundary's extent on the target mesh surface. We denote it as the cutting region of the boundary. The cutting region can be established in three steps. In step one we compute the normal of the source fitting plane then transfer it to the target link for building the corresponding target fitting plane. In step two the planar slice of the mesh and fitting plane is constructed and become the initial location of the cutting region. Finally in step three we consider the extent covered by source boundary, and expand the target cutting region to make it have similar extent as source.

The normal of the source fitting plane can be computed easily. But they can not be directly copied to the target space since the source and target meshes may have different orientations in the original 3D space. We solve the issue by establishing the consistent local coordinate systems on both sides of the source and target links, then transferring plane normals through the coordinate systems. The consistent coordinate system can be established in five steps as shown in figure 5.4.



(a) (b)

(c) (d)

(e)

Figure 5.4: The process of establishing consistent local coordinate systems on the source and target links. Where red, green, blue line represent x, y, and z axes, respectively.

First, for the current source link, we get the junction skeleton node connected the link and set it as the origin of the local coordinate system. The x axis is the local direction of the current link(section 4.3.2). The y axis is the local direction of another link connected to original junction node. Second, the z axis is the cross product of the x and y axes, and y axis is reset as the cross product of the z and x axes to form the orthogonal coordinate system. Third, the source coordinate system is transferred to the corresponding target node by using the shape matching technique(section 4.3.2). Fourth, the orientation of the target coordinate system is adjusted by aligning the x axis to the local direction of the current target link. Finally, the coordinate systems are moved from junction nodes to the points located by LR value, the orientations are adjusted according the local directions of the links.

We denote N_s as the normal of the source fitting plane, M_s and M_t as the transformation matrixes from original 3D space to the source and target local coordinate system, respectively. The normal of the target fitting plane N_t can be computed by using a linear transformation: $N_t = M_t^{-1} * M_S * N_s$.

We define a planar intersection as the slice caused by the fitting plane and mesh. It contains a series of line segments crossing the polygons of the mesh. For the source and target meshes, we respectively construct the planar intersections of their fitting planes, as shown in figure 5.5. For most of the positions on the link, the generated planar intersections are close to the boundaries and are suitable to be the initial cutting region, just like figure 5.5(a-b). Except the junction of the two parts, some proportions of the planar intersection will belong to areas which do not correspond to the current link. As shown in figure 5.5(c-d), the fitting planes cut the trucks and cause the large, inappropriate planar intersections. For this case we must perform intersection reduction, which removes the proportions of the planar intersections which belong to other parts and reconnects the gaps. Although we know that these proportions are always far from the current links, it is still hard to correctly process intersection reduction since using distances to identify the removed proportions is not a robust way.

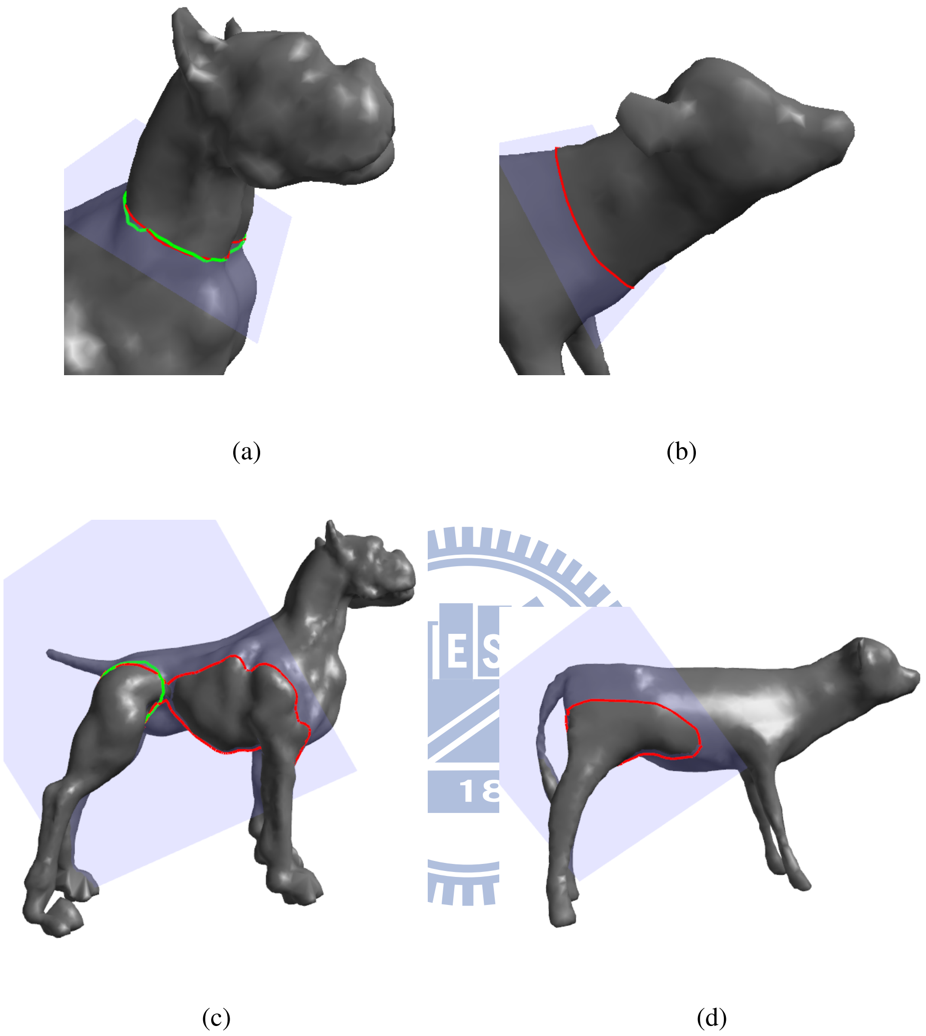
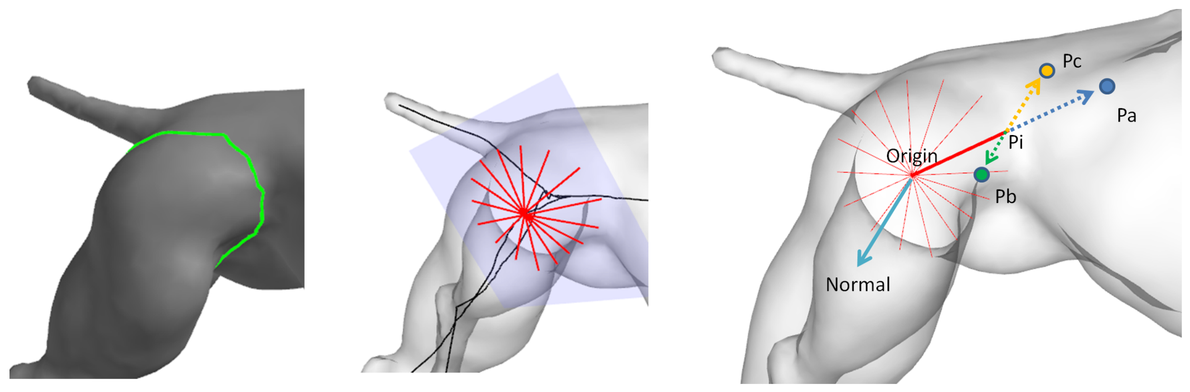


Figure 5.5: The planar intersections of the meshes and fitting planes. The intersections and boundaries are colored in red and green, respectively: (a-b) common position, (c-d) junction of the two parts

We thus take a different approach to deal with the intersection reduction. The distance information of the points of source boundary to the links is utilized to construct the planar intersections through the following processes. First, the source boundary are projected on its fitting plane. The origin and the x, y, z axes of the local coordinate system are also projected.

We use the longest projected axis as the start ray and rotate it around the normal of the fitting plane to generate several sample rays, as shown in figure 5.6(a-b). The number of the sample rays affects the accuracy of the boundary description. But in practice we found that even a few samples could achieve good results. The number is set to 16 in all of our experiments. Each sample ray is performed the ray-boundary intersection in the fitting plane domain to obtain the intersecting point P_i (note that it may be inside the mesh) and distance. And it casts a point P_a on the mesh which represents the sample of the original planar intersection. We additionally generate two rays which start at P_i and parallel to the normal and inward normal of the fitting plane. They cast two points on the mesh called P_b and P_c which represent the best fitting points by distance information of the boundary. For each sample ray a final sample point is chosen from one of the P_a, P_b, and P_c which has minimum distance to the current link (The origin in figure 5.6(c)). Finally the reduced planar intersection is constructed by using the geodesic paths to connect all the adjacent sample points. The idea of this method is that, when fitting plane is located at the junction of two parts, the mesh surfaces may be nearly parallel to the fitting plane. So it causes the intersections(P_a) which has large distance to the link(see figure 5.5(c-d)). In this case the points generated by distance information of the boundary (P_b or P_c) are substituted for these proportions. Actually, in most of the case these points are very close to the fitting plane. On the other hand, when fitting plane is located at the common positions, the normal of the fitting plane will be more parallel to the link (see figure 5.5(a-b)), so the points P_b and P_c will have larger distance then P_a.

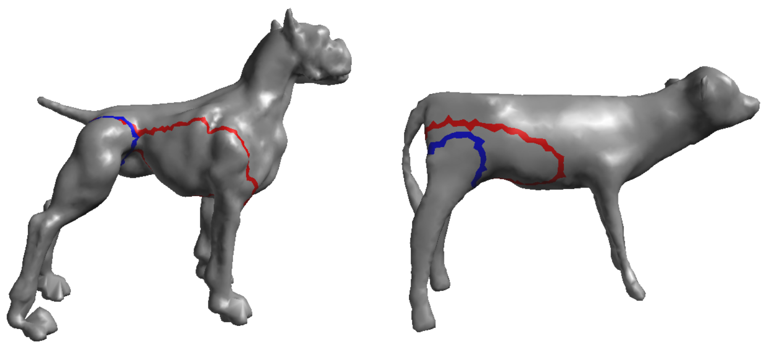


(a) (b) (c)

Figure 5.6: Intersections reduction by ray sampling: (a) boundary on the mesh, (b) sample rays on the fitting plane, (c) a diagram showing a sample ray generates three candidate points: P_a, P_b and P_c

The processes of the target intersection reduction are similar to source. Except the length of the sample rays which need to be provided from source (target mesh has no boundary). The ratio of the local MSP values of source to target links is computed and the lengths of the source sample rays are scaled and transferred to the target.

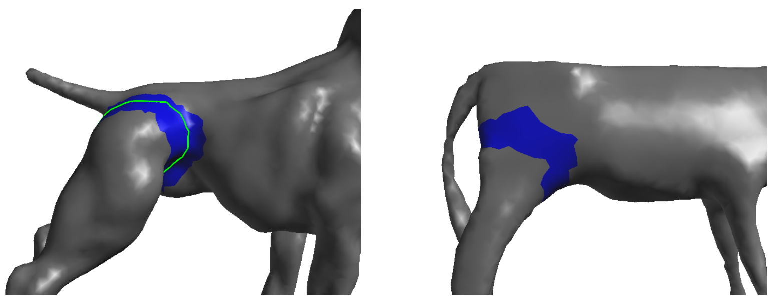
In fact, during the intersection reduction we do not really compute the geodesic paths. Instead we simply perform Dijkstra algorithm on the dual graph of the mesh to generate the face lists between the adjacent sample points. Figure 5.7 shows an example of the original and reduced planar intersection. We can see that both of the planar intersections are quite reduced.



(a) Source. (b) Target.

Figure 5.7: The original planar intersections colored in red, and the reduced planar intersectyions colored in blue.

Now the reduced planar intersections are expanded to become the cutting regions of the boundaries. For the source we set the distance of each face centroid on the planar intersections to zero, and run Dijkstra algorithm to gradually add adjacent faces to the cutting region until all the one-ring vertices of the source boundary are included in it. The maximum expanding distance of source is recorded, scaled by the ratio of the longest geodesic path of source to target meshes, and used to expand the target cutting regions. Figure 5.8 shows an example of the source and target cutting regions.

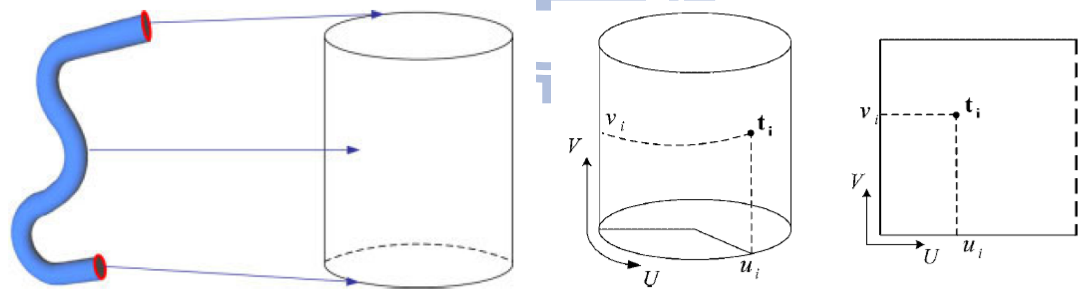


(a) Source. (b) Target.

Figure 5.8: An example of the source and target cutting regions.

5.5 Cylindrical parameterization

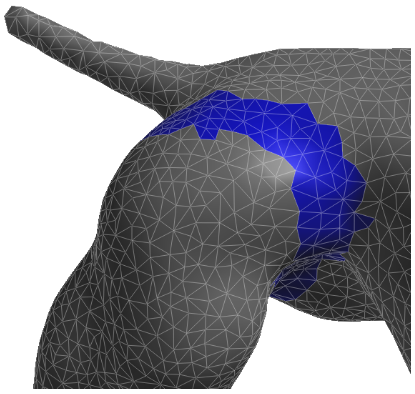
Next we parameterize the source and target cutting regions in order to constructing an intermediate domain which benefits the mapping the points of source boundary to target mesh. Since the cutting regions have two closed boundaries and form a ring shape, the common parameterization techniques which parameterize the meshes that have one boundary to the 2D plane domain are not suitable for this task due to the different topological types. So we use the special type of parameterize technique proposed by Wang and Zheng [WZ10] which parameterizes the tubular meshes to the cylindrical domains. Figure 5.9 shows an example of cylindrical parameterization. In figure 5.9(a) we know that the two boundaries of the tubular mesh correspond to the upper and lower boundaries of the cylinder, and the internal surface corresponds to the lateral side of the cylinder. Figure 5.9(b) shows the cylindrical parameterization result can be unfolded to a 2D plane domain. The v coordinates of the upper and lower boundaries are set to 1.0 and 0.0, respectively. And the ranges of the parameterized coordinates (u, v) are $[0, 1) \times [0, 1]$.

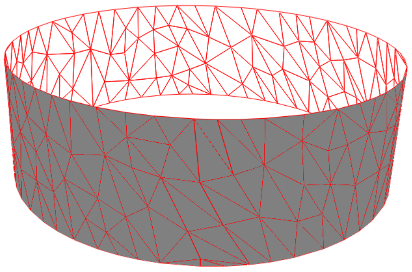


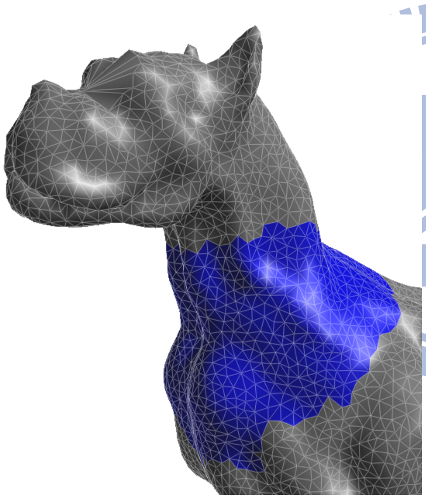
(a) Tubular mesh corresponds to the cylinder. (b) Cylindrical parameterization domain.

Figure 5.9: An example of cylindrical parameterization [WZ10].

Good cylindrical parameterization should avoid twisting in the U direction (See figure 5.11(a) for an example). For some area the coordinates of the vertices (u_i, v_i) need to be adjusted to $(v_i + 1, v_i)$ or $(v_i - 1, v_i)$ to keep the continuity of the parameters. These difficulties make it hard to construct the vertex-based equations to determine the parameterization. Therefore Wang and Zheng proposed an edge-based algorithm to overcome these difficulties. Each directed edge represents the coordinate offset on the parameterized domain. By randomly selecting a vertex on one of the boundary as the start vertex and set the coordinate to $(0, 0)$, the coordinate of all other vertices on the mesh can be inferred through propagating the parameters which are belonged to the connecting edges. To determining the v direction, we improve the original approach by constructing shortest path from the start vertex to another boundary in order to obtain the more robust direction that is almost parallel to vertical axis of the tubular mesh. Figure 5.10 shows some parameterization results. Figure 5.11 shows that we can avoid the twisted parameterization, and produce expected results as shown in figure 5.11(b).


(a) Cutting region on the leg.


(b) Paramterization result.


(c) Cutting region on the neck.

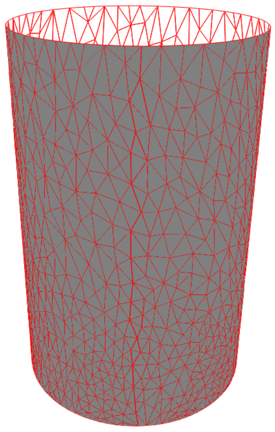
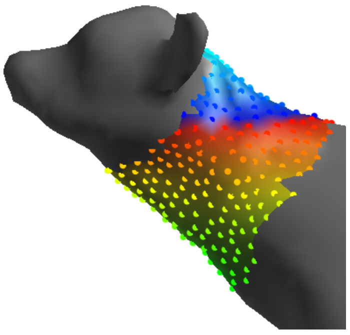
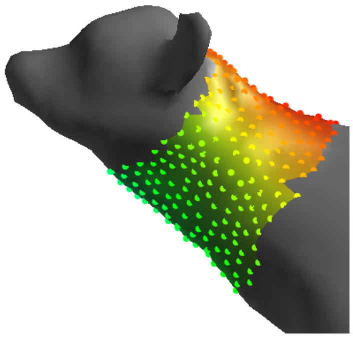
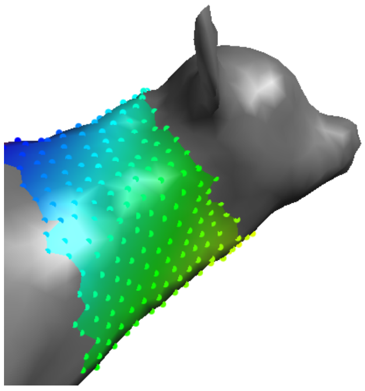

(d) Paramterization result.

Figure 5.10: Cylindrical paramterization result.


(a) Twisted.


(b) No-twisted.


(c) Twisted.

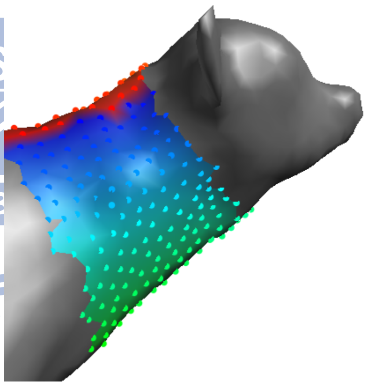
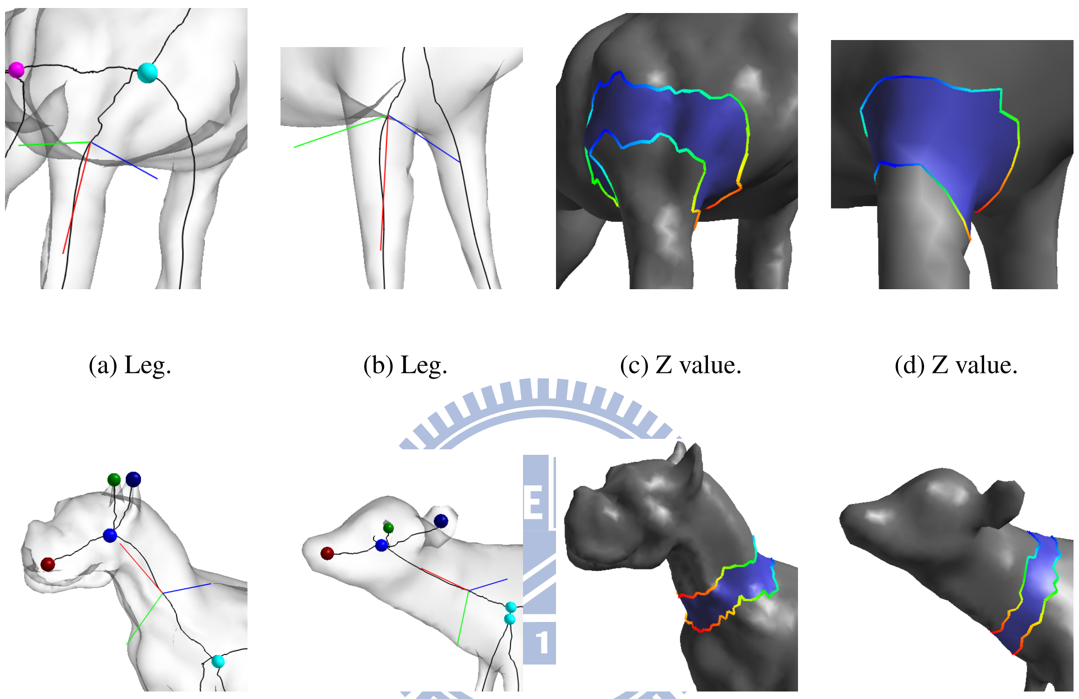

(d) No-twisted.

Figure 5.11: Twisted and no-twisted parameterizations. With the increasing of the u coordinate value, the vertices are colored from blue,green, to red.

Now we have constructed the parameterizations of the source and target cutting regions. A straightforward way for mapping the boundary is that we first transform the points of source boundary to the cylindrical domain, and then convert these points back to the 3D domain on the target mesh using barycentric coordinate. The method is effective but does not consider an important issue, is that the start vertices of the source and target parameterization are just randomly selected from the boundaries in the previous step. They may not be a good corresponding vertex pair and cause a large bias in the U direction on the cylindrical domain. We thus perform the parameterized orientation adjustment to address the issue.


(a) Leg. (b) Leg. (c) Z value. (d) Z value.

(e) Neck. (f) Neck. (g) Y value. (h) Y value.

Figure 5.12: Transform the boundaries to the local coordinate system. With the increasing of the coordinate value, the boundaries are colored from blue,green, to red.

We utilize the local coordinate system built in section 5.4 to transform all the boundary's vertices of the source and target cutting region to the local coordinates. Since the consistency of the coordinate systems, the both sides of boundaries have the similar distributions of the coordinate values, as shown in figure 5.12. We compute the range of the x, y, and z coordinate values on both sides of boundaries, choose the coordinate with largest range, and assign it to each vertex of the boundaries. We denote it as the alignment value of the vertex. Next we choose the boundary B_s of the source cutting region which has the start vertex, and the target boundary

B_t corresponding to B_s. We assume B_s has n vertices and define $U_{B_s} = \{u_1, u_2, ..., u_n\}$ as the set of parameterized u coordinate of the vertices. The orientation adjustment process aims to find the best u-offset $u_o \in [0, 1)$ of the B_s and B_t by minimizing the following equation:

$$\underset{u_o}{\operatorname{argmin}} \sum_{i=1}^{n} (AV(B_s, u_i) - AV(B_t, u_i + u_o))^2 \tag{5.3}$$

where the $AV(B, u_i)$ returns the alignment value at the position whose u coordinate is u_i on the boundary B. If it is on the edge, the alignment value is computed by interpolating the values of the two end points of the edge. The process can be treated as rotating the target boundary along the cylindrical axis so as to align the source boundary.

After computing the best offset u_o, we add u_o to all the points of the source boundary and transform them to the target mesh to form the initial target boundary.

5.6 Boundary refinement by snakes

In this section, the points of target boundary are refined locally, so that the characteristics of the source and target boundaries can be as similar as possible. We define the characteristics of the boundary as the following two terms:

1. **Surface detail:** According to the minima rule [HR84], most of the mesh segmentation approaches use curvature, dihedral angle, and normal deviation as the local geometric features, and guide the boundaries to close these features.

2. **Shape:** Most of the part-type boundaries are expected to have short and smooth shapes. However, in some scenarios, the user may depict the boundaries to the particular patterns such as jagged or wavy shapes.

The snakes were originally proposed by Kass et al. [KWT88] and were widely applied in the 2D image segmentations and feature detections. Some researches extended them to 3D mesh surfaces [LL02, JK04, BWK05, LMLR06].The general energy function of the snakes can be represented as:

$$E_{snake}(s) = \int (E_{int}(s) + E_{ext}(s))ds$$

where the internal energy term E_{int} makes the snakes shorten and smooth, and the external energy term E_{ext} drives the snakes toward the nearby features. The snakes evolve and converge on the final locations by minimizing the energy E_{snake}.

We use snake to refine target boundary and reformulate the energy function to the equation 5.4:

$$E_{snake}(s) = \int (E_{feature}(s) + E_{shape}(s))ds \tag{5.4}$$

where $E_{shape}(s)$ and $E_{feature}(s)$ are called the shape term and feature term. In the section 5.5, we have completed the initial boundary mapping. For each point of the target boundary we can find a corresponding point on the source boundary. The ability of the feature term is to evolve the target boundary to increasingly close the geometric features of each pair of corresponding points, and the shape term maintains the similar shape as source boundary during the refinement process.

We detail our snake scheme as follows. We adopt the snake scheme proposed by Bischoff et al. [BWK05]. The snake is composed by a sequence of discrete points called snaxels. The snaxel is stipulated to locate on the vertices or edges of the mesh. The points of the initial target boundary may need a local adjustment in order to satisfy this condition. In the following we call the points on the target boundary as snaxels. When the snake is evolved, for each snaxel several candidates are generated, and the snaxel is moved to the candidate which has smallest snake energy in the next round. If the snaxel locates on the edge of the mesh, two candidates are yielded on both sides of the snaxel on the same edge. If the snaxel locates on the vertex, the candidates are generated to each one-ring edge connected to the vertex. The original location of the snaxel is also as one of the candidates. The interval of the original snaxel and the candidate is set according to the application. In our implementation we set it as the average length of the edges of the mesh multiplied by a scalar(0.1). Our snake scheme requires the correspondence of the points between the source and target boundary. For each iteration, before the snake evolution we use equation 5.3 to find the corresponding point (may not on the vertex of the mesh) of the source boundary for each snaxel of the target boundary.

For the design of the shape term, we use the offsets of the points of the boundary and its fitting plane as the characteristics of the shape. For a straight and smooth boundary, all the points have small offsets. And for a jagged boundary, the offsets have larger deviations. To compute the offset, we calculate the projection of the point, measure the difference between them, and use the plane normal to judge the sign of the offset. The offset value is normalized by the diagonal of the bounding box of the model. We define the shape term as $E_{shape} = abs(offset(t) - offset(s))$, where s is the corresponding point of the source boundary for the snaxel t.

For the design of the feature term, we use the dihedral angle as the surface feature on the mesh. First the dihedral angles are calculated on the edges of the mesh, then the feature value is assigned to each vertex as the average dihedral angle of its one-ring edges and normalize it to $[0, 1]$. Hence for each candidate snaxel of the target boundary and the corresponding point of the source boundary, we can obtain their feature values. If the point is on the edge, the feature value is computed by linear interpolation. Our goal is to make the feature values of the both sides of points as close as possible. An intuitive idea is to design the feature term as $E_{feature} = abs(feaVal(t) - feaVal(s))$, where $feaVal(p)$ is the feature value of the point p. However, after performing experiments, we found that it has two drawbacks. First , when both the source and target boundary locate on the flat regions, the target boundary would be expected not to move. But actually it still moved due to the small differences of the float-point feature values. Second, the snaxel usually falls into the local minimum since the snake scheme only considers a small searching range for each iteration (the candidates are close to the origin location of the snaxel).

So we modified our strategy as follows. For each iteration before the snake movement, we define a searching range, and an adjacent vertex list which collects the vertices whose geodesic distances to the target boundary are under the searching range. The searching range should be large enough to cover the nearby feature regions, we set it as the longest geodesic distance on the model multiplied by a scalar(0.02). For each snaxel of the target boundary, we get the feature value of its corresponding point of the source boundary, and search the adjacent vertex list to find the closest vertex whose feature value is enough similar to the current snaxel (if the difference of the feature values is smaller than a threshold, they are enough similar. The threshold is set as 0.1). If no vertex is found, the feature energies of all the candidates are set to the same value(1.0). Otherwise, we simply define the feature energy as : $E_{feature} = geo(v, p)$, where v is the vertex searched in the adjacent vertex list, and p is the target candidate, $geo(v, p)$ returns the geodesic distance between v and p. Under this design, the candidate which is close to the searched vertex will have small energy. In order to address the small float-point difference problem, we compute the mean$(f_M)$and the standard deviation(f_{SD}) of the of the feature value of the mesh. If both the target snaxel and source point are on the flat region (the feature value is in the range of $f_M \pm f_{SD}$), we also set the energies of the candidates to 1.0 to disable the function of the feature term.

We dynamically adjust the weight of the shape and feature term during the snake movement. If both the source and target boundaries locate around the feature regions, the weight of the feature term should be large in order to make the boundary close to the feature. Otherwise, if both the source and target boundaries locate on the flat regions, then the weight of the shape term should be large to control the shapes of the boundary. For each iteration, we calculate the percentage of the case that a target snaxel successfully find a feature similar vertex in the adjacent vertex list, set the weight of the feature term (w_f) as the percentage value and set $(1.0 - w_f)$ as the weight of the shape term. The final snake energy for a target snaxel p (candidate) is defined as:

$$E_{snake}(p) = w_f * E_{feature}(p) + (1.0 - w_f) * E_{shape}(p)$$

Figure 5.13 and 5.14 demonstrate the function of the shape term when both source and target boundaries are on the flat regions. And figure 5.15 shows the function of the feature term which guides the target boundary toward the concave features as similar as the source boundary.

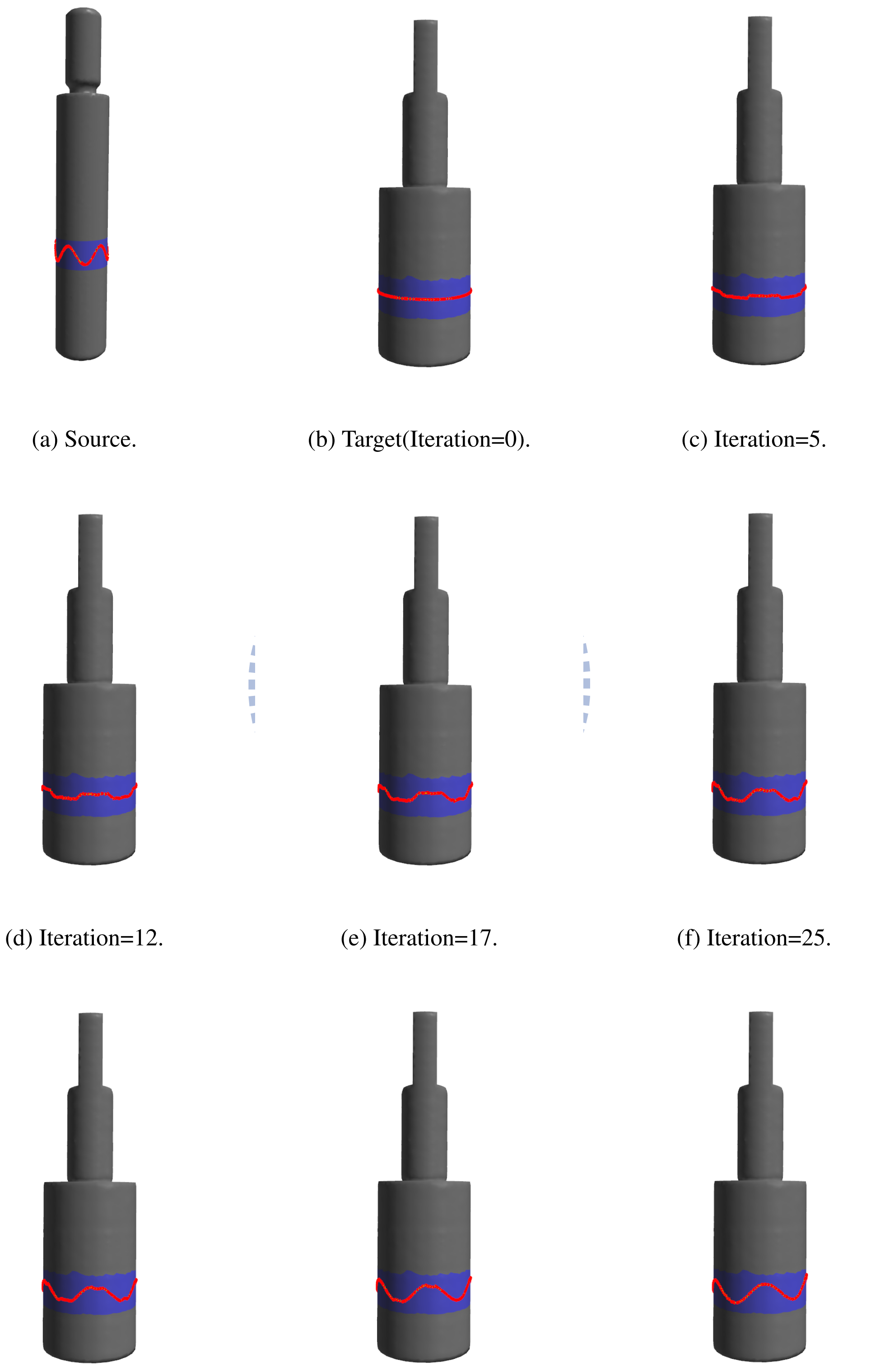
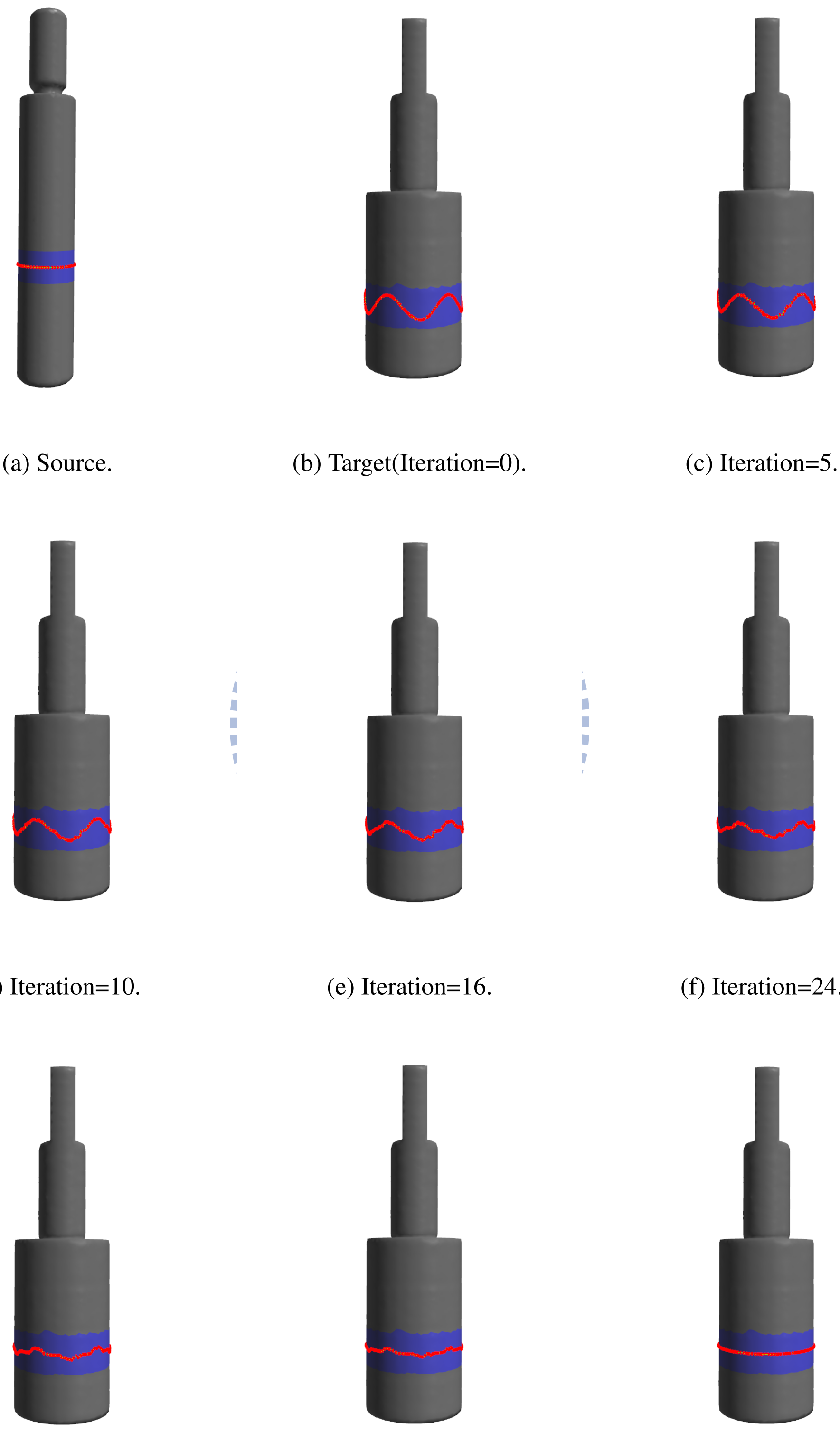

(a) Source.
(b) Target(Iteration=0).
(c) Iteration=5.
(d) Iteration=12.
(e) Iteration=17.
(f) Iteration=25.
(g) Iteration=30.
(h) Iteration=37.
(i) Result (Iteration=45).

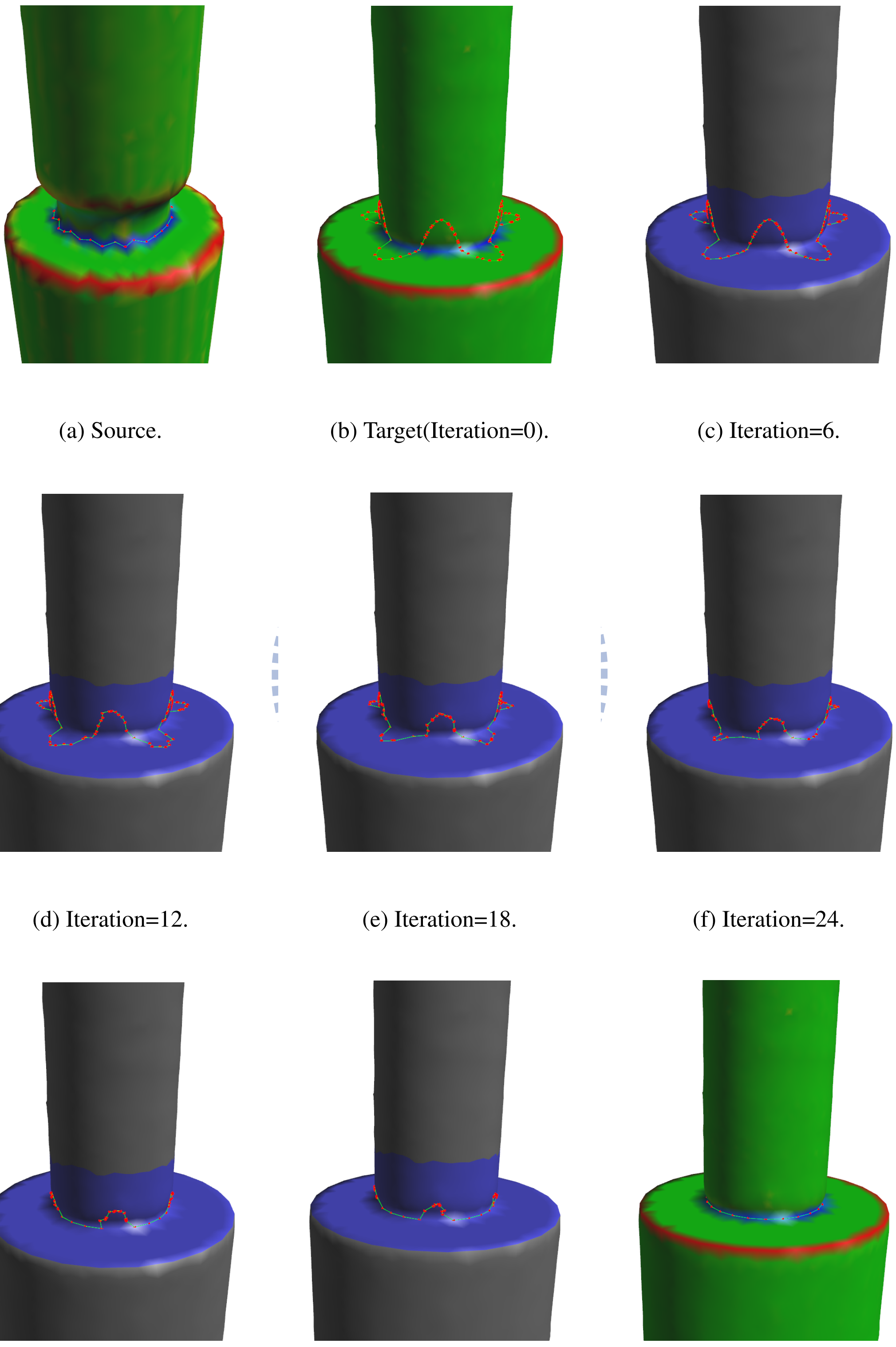
Figure 5.13: The evolution of the target boundary from line shape to wave shape.


(a) Source. (b) Target(Iteration=0). (c) Iteration=5.

(d) Iteration=10. (e) Iteration=16. (f) Iteration=24.

(g) Iteration=30. (h) Iteration=35. (i) Result (Iteration=41).

Figure 5.14: The evolution of the target boundary from wave shape to line shape.



(a) Source. (b) Target(Iteration=0). (c) Iteration=6.

(d) Iteration=12. (e) Iteration=18. (f) Iteration=24.

(g) Iteration=35. (h) Iteration=42. (i) Result(Iteration=50).

Figure 5.15: The evolution of the target boundary to the concave features. The color mapping on the model in (a)(b)(i) means the variations of the feature value. Green color represents flat region, red and blue color represent the convex and concave regions.

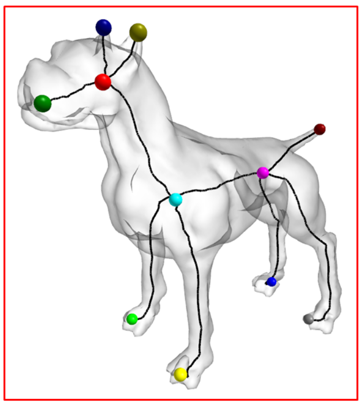
CHAPTER 6

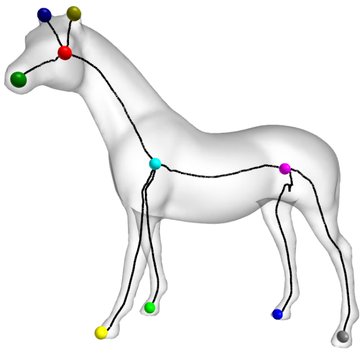
Result

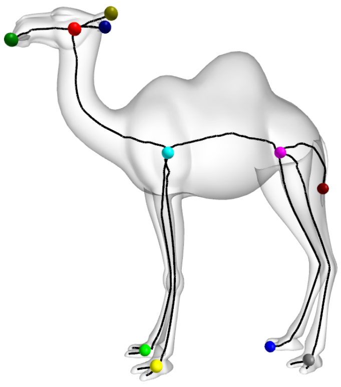
In this section, we present the results of the skeleton correspondence, segmentation transfer and the timing information, and then discuss the limitations of our system.

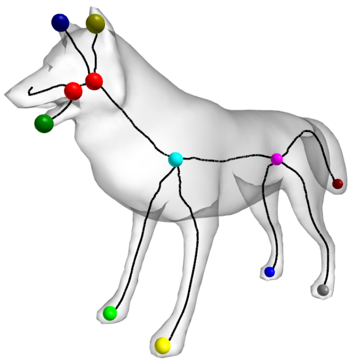
6.1 Skeleton correspondence

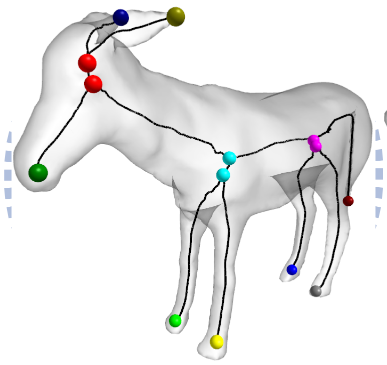
Figure 6.1 and 6.2 show the skeleton correspondence results for a variety of models including four-legged animals, human, and some dragons. The dog (figure 6.1(a) and figure 6.2(a)) is set as the source model, then perform the skeleton correspondence to other target models. The matched node pairs are drawn in the same colors, and the unmatched nodes are not drawn. Our results show good correspondences that most of the nodes belonged to the same semantic parts (e.g. bodies, legs, heads, and ears) are matched correctly without symmetry switching problem. Since the links are grouped with same semantic meanings, we can avoid the wrong correspondences for some parts of a model that do not exist in another models, such as mapping the tail of the dog to the leg of the horse in figure 6.1(b), or mapping the ears of the dog to the jaw of the dinosaur in figure 6.2(e). Our method can handle the models which are largely different in shapes (e.g. mapping the dog to the human, the armadillo, and the dinosaur in figure 6.1(c-e)), poses (e.g. the cat in figure 6.1(g)), the local different connectivities for the junction nodes (e.g. the junction nodes of the goat and deer in figure in 6.1(e-f)), and even can treat the skeletons with large topological differences caused by additional junction nodes and small branches (e.g. the triceratops, elephant, and asian dragon in figure 6.2(f-h)). Due to the topological differences in the skeletons, our method performs the node clustering and link extension. So there may be one-to-many or many-to-many correspondences between nodes and links. That are different from the previous skeleton matching algorithms [HSKK01, BMSF06, ATCO$^+$10] which just establish one-to-one node correspondence.

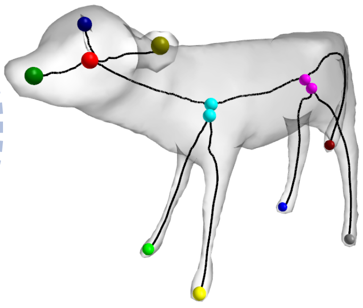

(a) Dog.

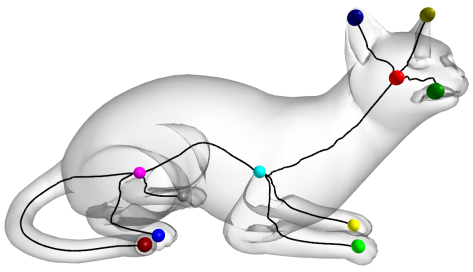

(b) Horse.


(c) Camel.


(d) Wolf.


(e) Goat.


(f) Deer.


(g) Cat.

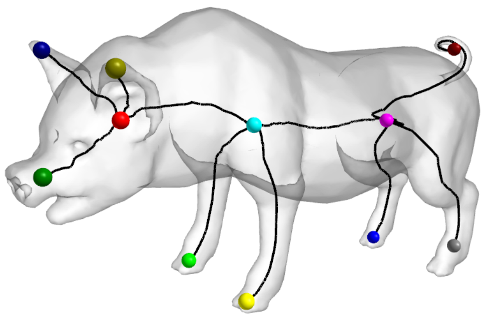
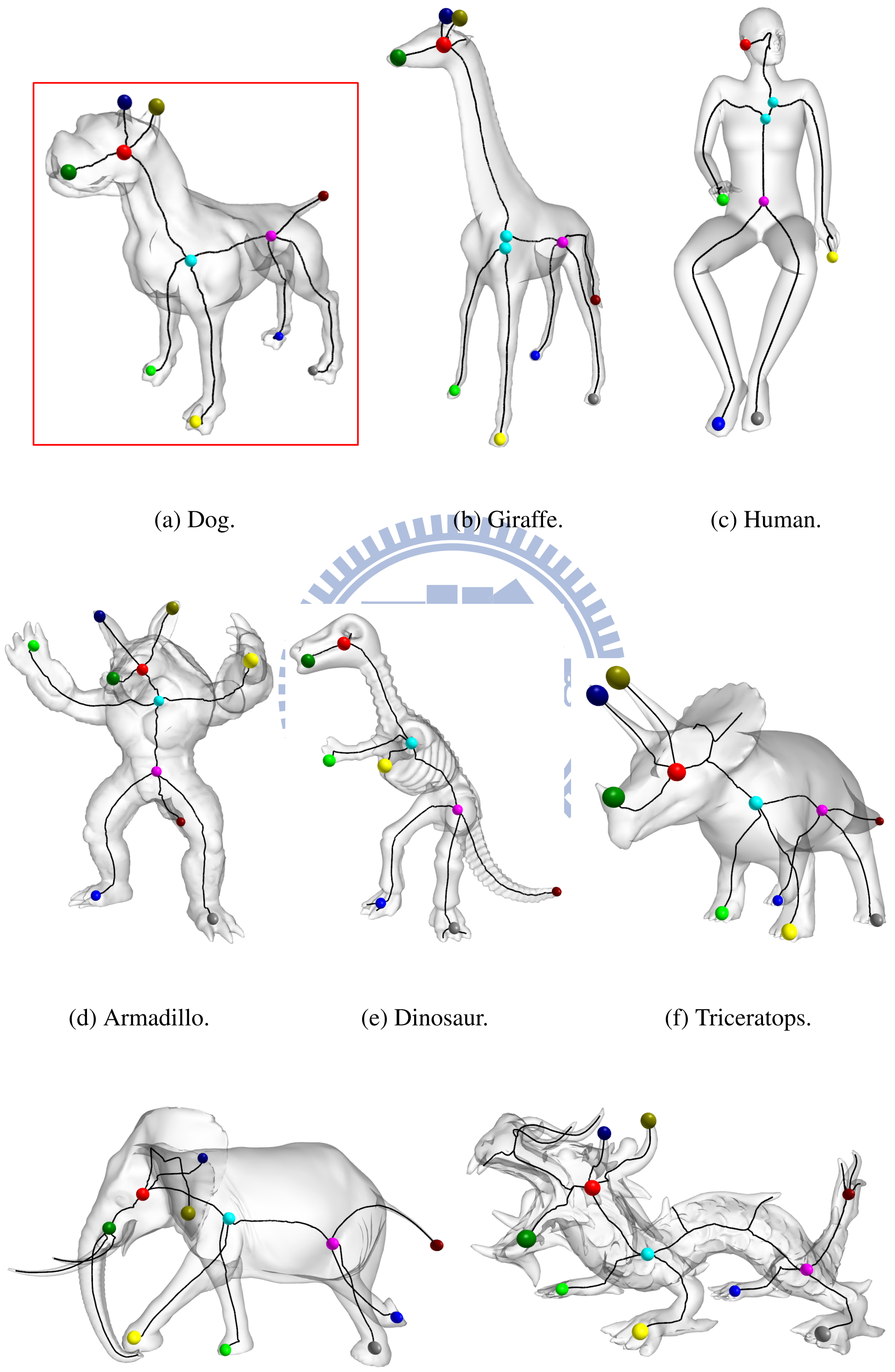

(h) Pig.

Figure 6.1: Skeleton correspondence results (sec. 1).



(a) Dog. (b) Giraffe. (c) Human.

(d) Armadillo. (e) Dinosaur. (f) Triceratops.

(g) Elephant. (h) Asian dragon.

Figure 6.2: Skeleton correspondence results (sec. 2).

We also apply our skeleton correspondence method to the various categories of models in the Princeton Segmentation Benchmark dataset [CGF09]. The results are shown as follows. Note that our method can also handle the skeletons with loops, such as the chairs in figure 6.6.

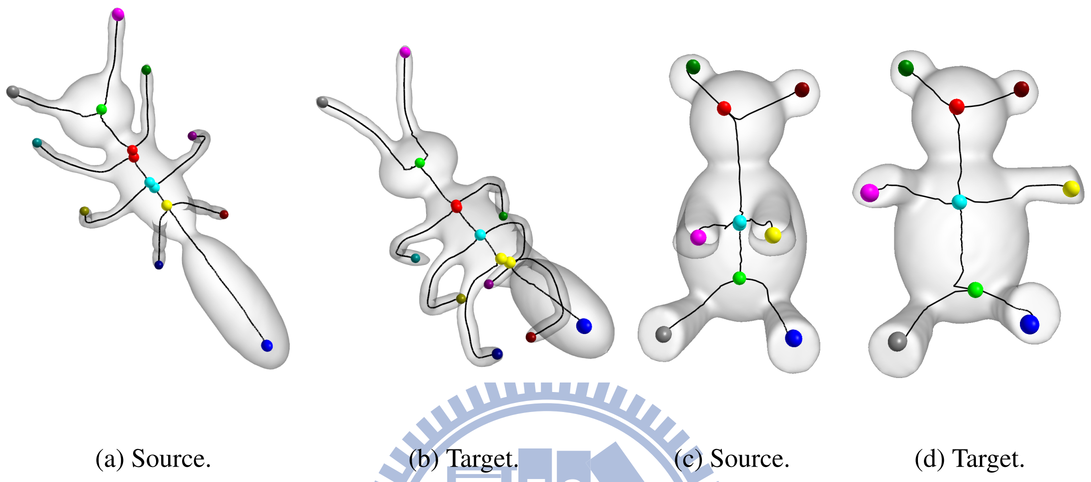

(a) Source. (b) Target. (c) Source. (d) Target.

Figure 6.3: Skeleton correspondence results of Ant and Teddy models.

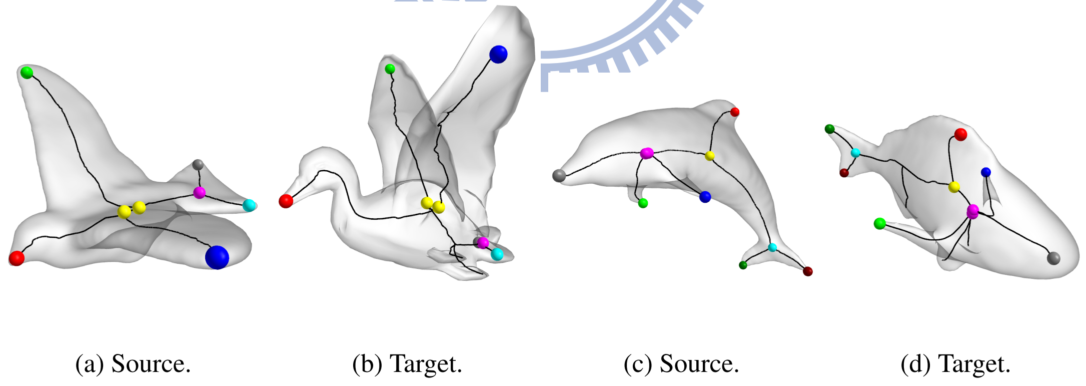

(a) Source. (b) Target. (c) Source. (d) Target.

Figure 6.4: Skeleton correspondence results of Bird and Fish models.

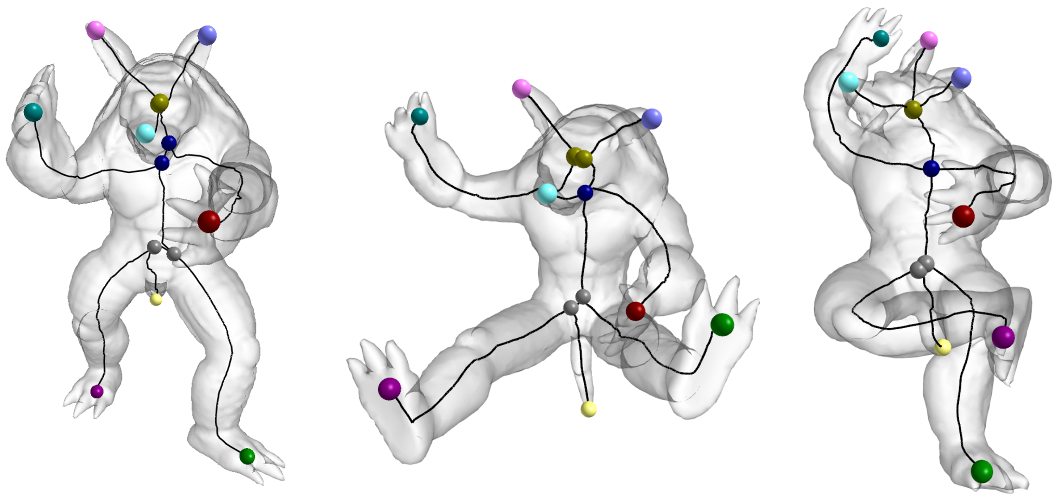

(a) Source. (b) Target 1. (c) Target 2.

Figure 6.5: Skeleton correspondence results of Armadillo models.

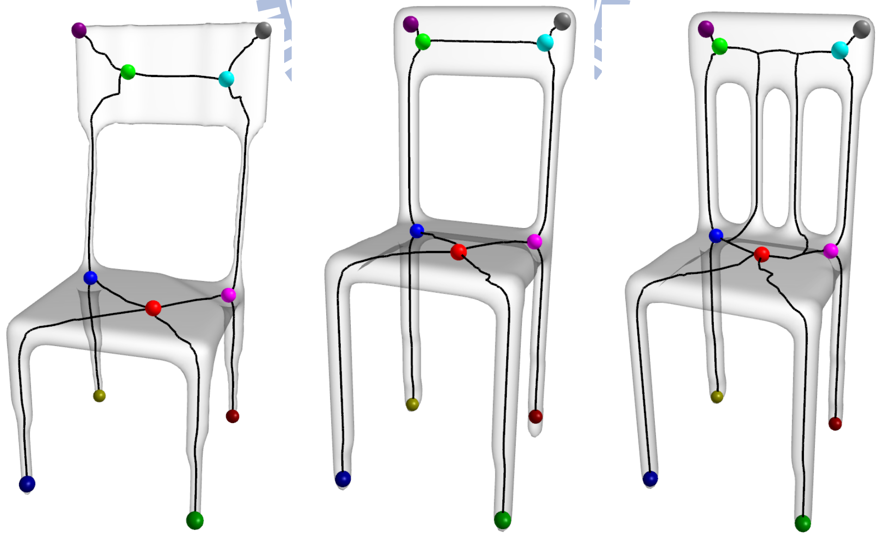

(a) Source. (b) Target 1. (c) Target 2.

Figure 6.6: Skeleton correspondence results of Chair models.

We compare our method with the method of elector voting [ATCO$^+$10]. Since in the paper they demonstrate results with the skeletons proposed in [ATC$^+$08]. For fairness of the comparison, we implement the method of elector voting, and use the same MSP skeletons as the input of our and their algorithms.

The matching processes of both our method and the elector voting can be described as the correspondence tree. The method of elector voting uses combinatorial search to establish all possible corresponding node pairs. Our method is based on graph traversing, which generates much less tree nodes of the correspondence tree. The meanings of the tree node are different: The tree node of the elector voting represents a one-to-one match of a node pair between the source and target skeletons. And the tree node of our method represents a partial correspondence between the source and target skeletons(may contains several matches of the node pairs and link pairs). The leaf node of our method represents a complete candidate of the skeleton correspondence. And a candidate correspondence(denote it as a elector) of the elector voting is comprised by all the node pair correspondences along the path from a leaf node to the root.

The pruning tests of our method and elector voting are also different: The method of elector voting use four cascading pruning tests involving geometric features(T1,T2), topology(T3), and spatial configuration(t4) to filter out the bad node correspondences. And the pruning tests of our approach exploit the spatial configuration to exclude the bad link correspondences. Our method does not require the check of topology since the graph traversing method essentially maintains the topological consistent.

After constructing the correspondence tree, their method uses voting scheme to synthesis the final correspondence. Since we have much less leaf nodes, so a traditional approach is used which presents an objective function in order to select the final result(equation 4.7). Currently we are unable to prove that our approach outperforms the voting scheme(on the other hand, the voting scheme also hard to guarantee that it is better than the traditional approach). Table 6.1 and 6.1 list the statistics of the skeleton correspondences for our method and the elector voting. Comparing to elector voting, our method has less requirements of parameter adjustment, the numbers of the tree node, and computation times.

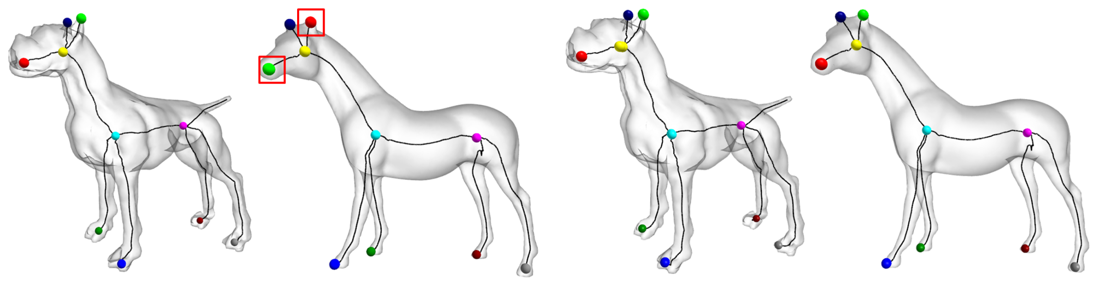
Model	Wolf	Goat	Deer	Giraffe	Cat	Pig	Human	Armadillo	Triceratops	Asian dragon
t_n	0.02	0.02	0.02	0.02	0.02	0.06	0.03	0.03	0.03	0.08
t_{in}	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Tree node	17	18	18	14	16	11	12	13	17	18
Time(sec.)	0.35	0.30	0.24	0.32	0.23	0.38	0.23	0.25	0.21	0.44

Table 6.1: The statistics of our skeleton correspondence results.

Model	Wolf	Goat	Deer	Giraffe	Cat	Pig	Human	Armadillo	Triceratops	Asian dragon
ϵ_C	0.8	0.8	0.8	0.8	1.0	1.2	1.0	1.0	1.0	0.8
ϵ_{RD}	1.0	1.0	1.0	1.0	1.2	2.5	1.6	1.5	10.0	0.7
Tree node	2880	2385	2937	2335	1817	8523	954	2874	3519	3495
Elector	103	633	324	279	676	196	229	372	902	1128
Time(sec.)	1.21	2.54	1.37	3.13	1.08	1.30	1.31	1.24	1.14	1.36

Table 6.2: The statistics of the results of elector voting.

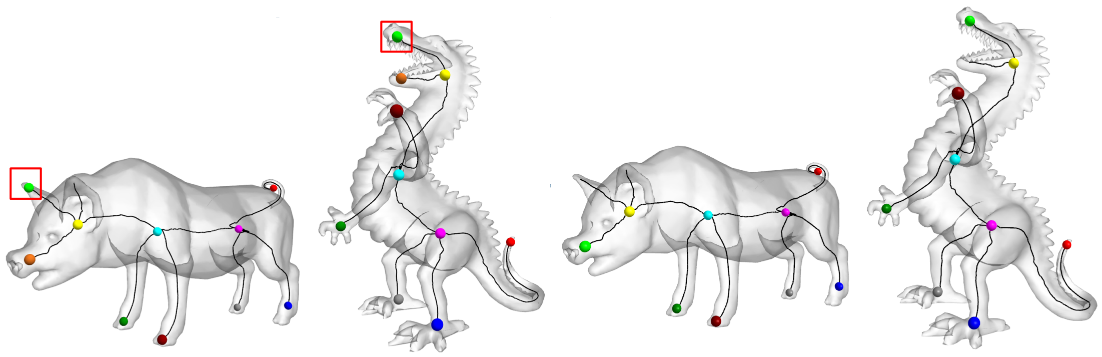
In figure 6.7(a), we can see that the method of elector voting mismatches the nodes at the head of the horse. The problem is that it applies multidimensional scaling (MDS) [EK03] to the skeletons in order to straighten the skeleton links. However, the directions of the straightened links are easily guided by the beginning directions of the original links. In some case the models with difference poses will cause the directions of the links are not consistent after the MDS process, and disturb their spatial pruning tests (T4). On the other hand, our method determines the link directions in the local range of the original models, which is less sensitive to the poses. Moreover, we use the volumetric characteristics which can distinguish these small links. The ears and the jaw will be classified into difference groups. The information helps our method to establish correct correspondence as shown in figure 6.7(b).



(a) Results of elector voting. (b) Our results.

Figure 6.7: The comparison with [ATCO$^+$10] on Dog and Horse models.

In figure 6.8(a), the method of elector voting mismatches the ear of the pig to the upper law of the dragon due to these nodes have similar features (e.g. centricity, path length and radius). On the other hand, our method produces correct result as shown in figure 6.8(b). After performing the link clustering, the two ears of the pig will form a group. And the upper and lower laws of the dragon will be classified into difference groups. Since these groups only have one link, we regards these links as ungrouped links and avoid mapping them to the links which are grouped (the ears of the pig). Using the link clustering, we have more information to establish correct correspondence.



(a) Results of elector voting. (b) Our results.

Figure 6.8: The comparison with [ATCO$^+$10] on Pig and Dragon models.

When the models have large structural differences, the mothod of elector voting may generate wrong correspondences such as in figure 6.9(a). It is a partial correspondence problem which attempts to find the human shape from the centaur models. Compare with the figure 6.9(a), our method produces more semantically correct result as shown in figure 6.9(b).

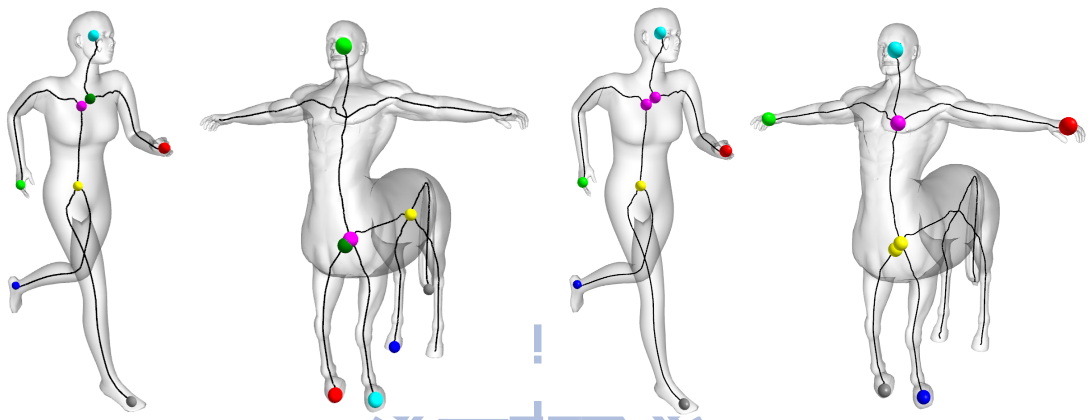
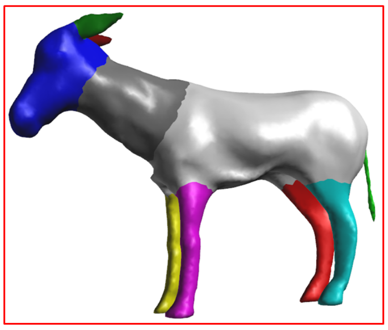

(a) Results of elector voting. (b) Our results.

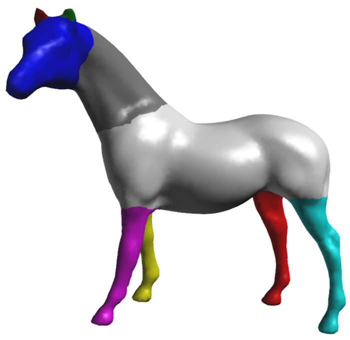
Figure 6.9: The comparison with [ATCO+10] on Human and Centaur models.

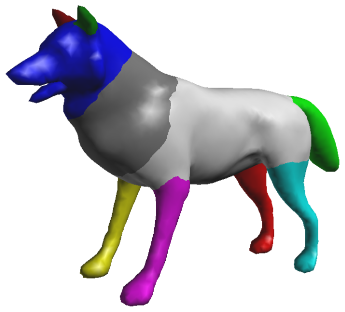
6.2 Segmentation transfer

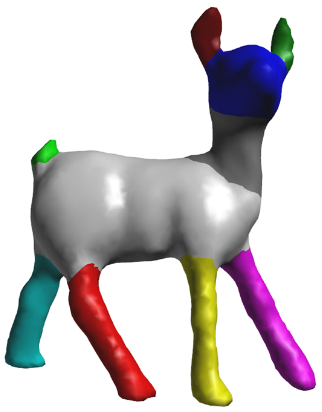
Figure 6.10, 6.11 and 6.12 show our segmentation transfer results for a variety of models including four-legged animals, human, and dragons. The source models in figure 6.10(a), 6.11(a) and 6.12(a) are framed by a red box. All the target models have consistent segmentations with source models such that the corresponding segment pairs are colored in the same colors. Figure 6.10 and 6.10 show the different types of segmentation for the four-legged animals. For the legs of the source animal in figure 6.10(a), the boundaries cross the legs vertically and form the short, straight shapes. In figure 6.11(b), the boundaries cross the legs in the tilted manners and form the more longer shapes. We can see that all the target models reproduce these characteristics since the fitting planes of the boundaries are transferred from the source models to the target models (section 5.4). Especially notice the differences between the horse in figure 6.10(b) and

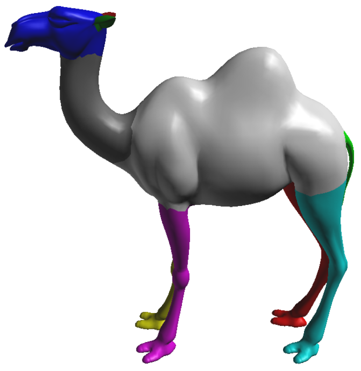
6.11(c), the wolf in figure 6.10(c) and 6.11(d), the deer in figure 6.10(g) and 6.11(b). Once the skeleton correspondences have been established, the segmentation transfer can be performed to the models with large shape and geometry differences, as shown in figure 6.12. According to the skeleton correspondences, the target model may not have some parts compared to the source model. For example, the dragon in figure 6.12(e) does not have the ears, and the human in figure 6.12(c) does not have the tail and ears. In such case the boundaries of these parts of the source model will not be transferred to the target model.

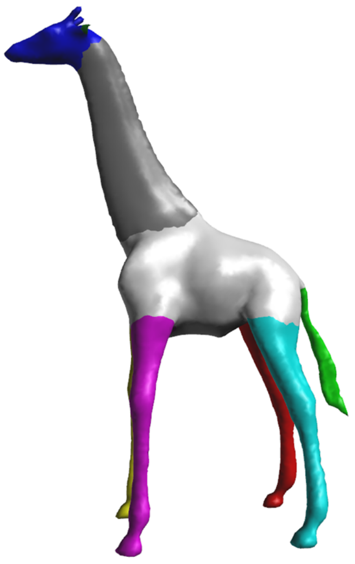

(a) Goat.

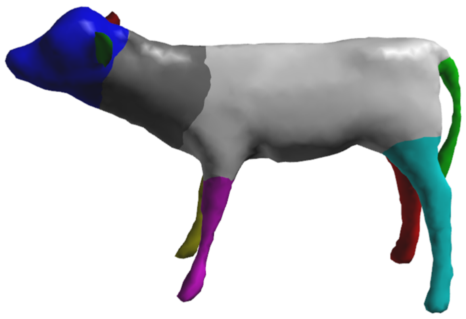

(b) Horse.


(c) Wolf.


(d) Reindeer.


(e) Camel.


(f) Giraffe.


(g) Deer.

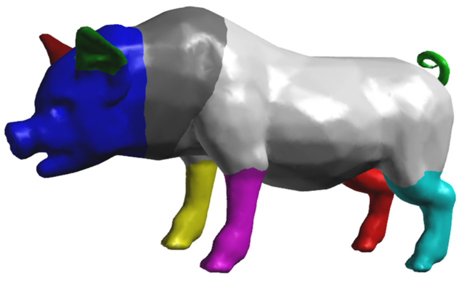
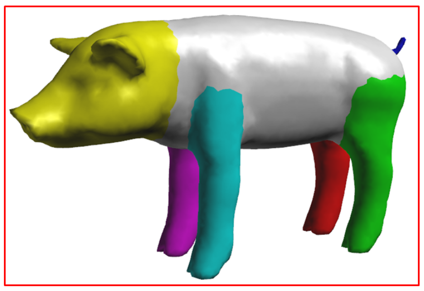
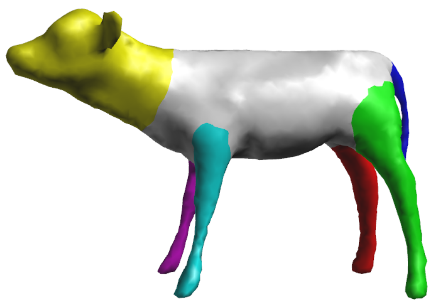
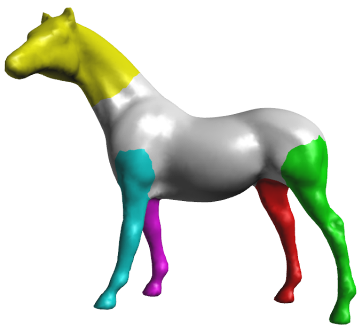
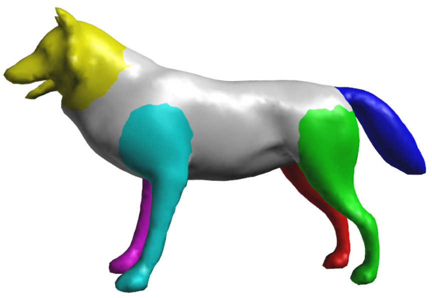

(h) Pig.

Figure 6.10: Segmentation transfer results (sec. 1).


(a) Pig.


(b) Deer.


(c) Horse.


(d) Wolf.

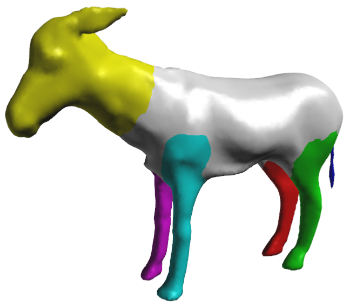
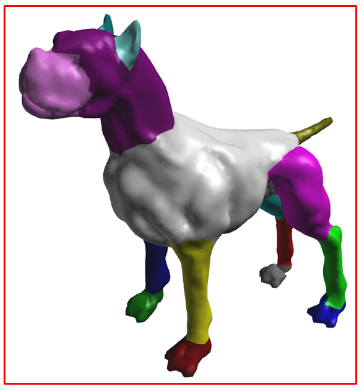
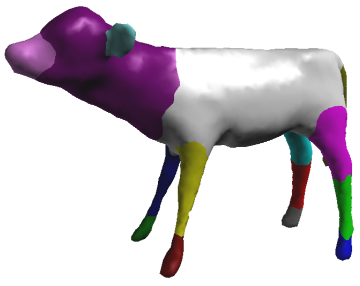
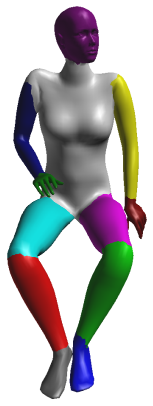
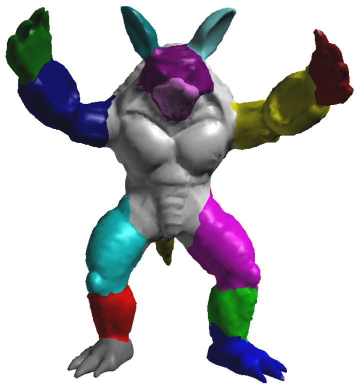

(e) Goat.

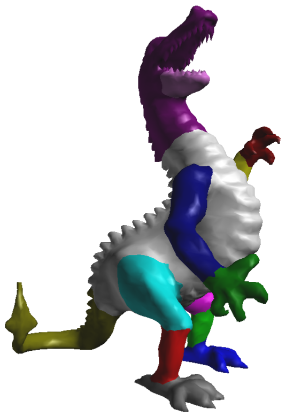
Figure 6.11: Segmentation transfer results (sec. 2).

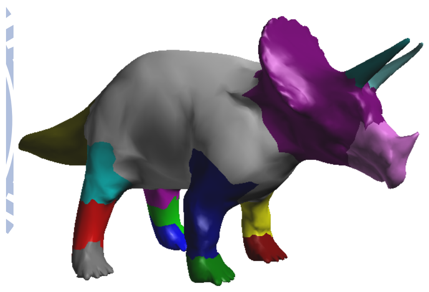

(a) Dog.

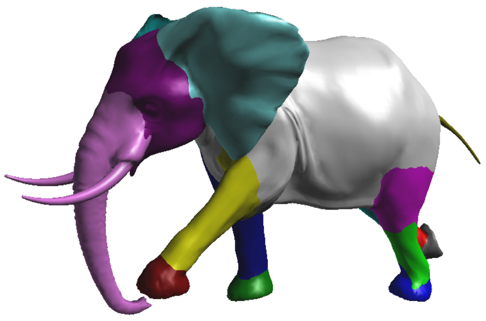

(b) Deer.


(c) Human.


(d) Armadillo.


(e) Dragon.


(f) Triceratops.


(g) Elephant.

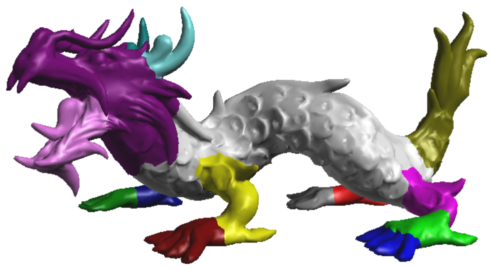
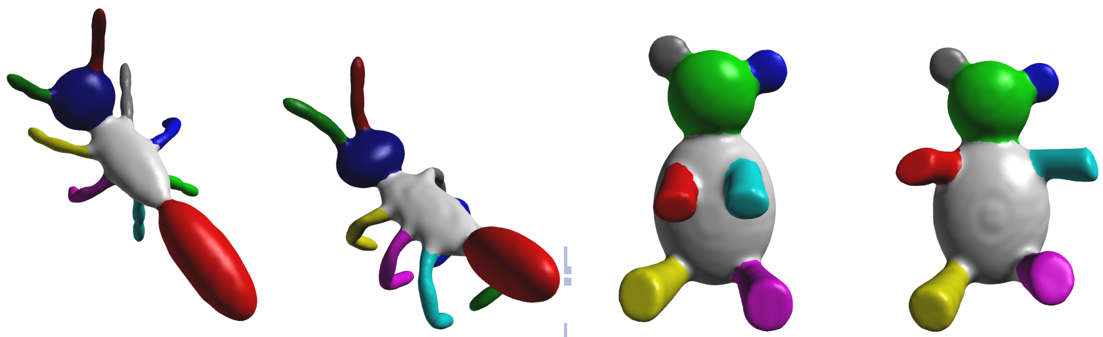

(h) Asian dragon.

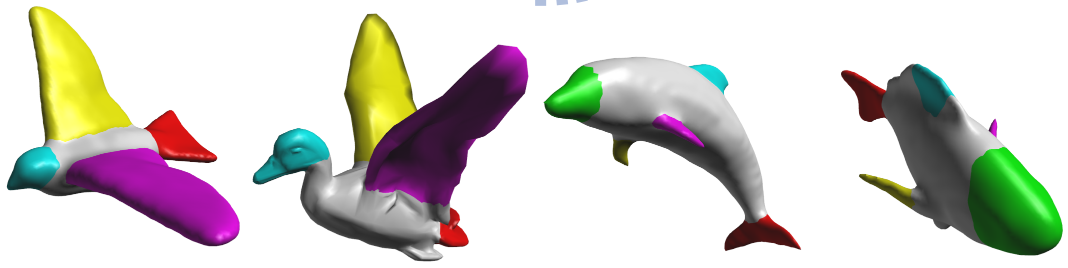
Figure 6.12: Segmentation transfer results (sec. 3).

We also demonstrate the segmentation transfer results for the other categories of models in the Princeton Segmentation Benchmark dataset. Note that the figure 6.16 illustrates a limitation of our method. When the models have different topological shapes, our method may produce inconsistent results. The seat and back of the chair in figure 6.16(c) can not be separated since there is no boundary on some pillars.



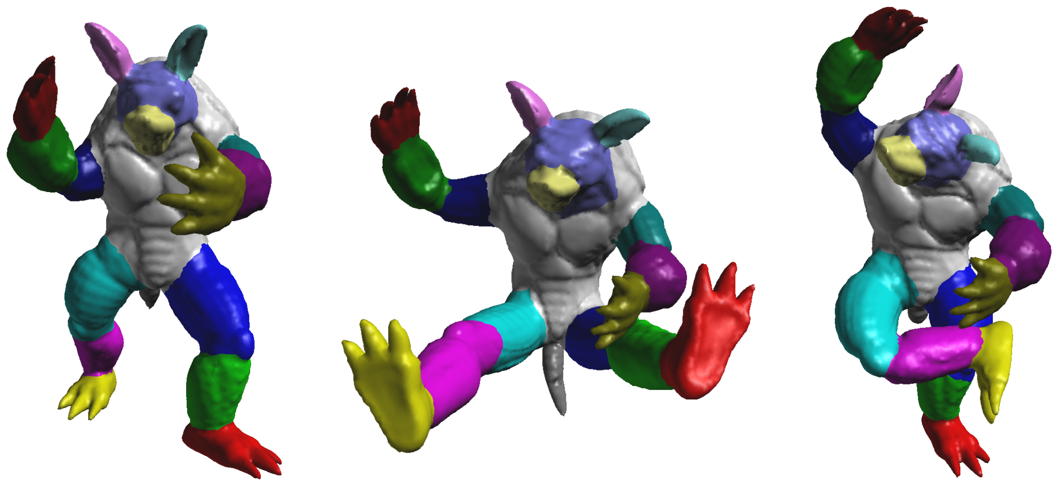
(a) Source. (b) Target. (c) Source. (d) Target.

Figure 6.13: Segmentation transfer results of Ant and Teddy models.



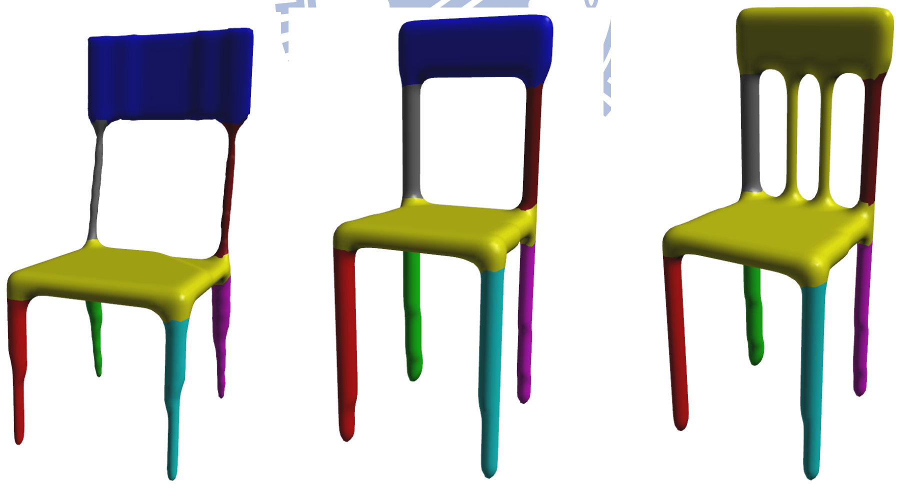
(a) Source. (b) Target. (c) Source. (d) Target.

Figure 6.14: Segmentation transfer results of Bird and Fish models.



(a) Source. (b) Target 1. (c) Target 2.

Figure 6.15: Segmentation transfer results results of Armadillo models.



(a) Source. (b) Target 1. (c) Target 2.

Figure 6.16: Segmentation transfer results of Chair models.

We first compare our apporach with the method proposed by Golovinskiy and Funkhouser [GF09]. Figure 6.17 shows the comparison. For the models which have large differences in shapes or poses, the rigid alignment used in [GF09] is Insufficient to correctly build the face correspondences between the models, and then disturbs the segmentation results. Our method achieves better results since the skeleton matching is more robust to construct the part-based correspondences between the models which have large differences.

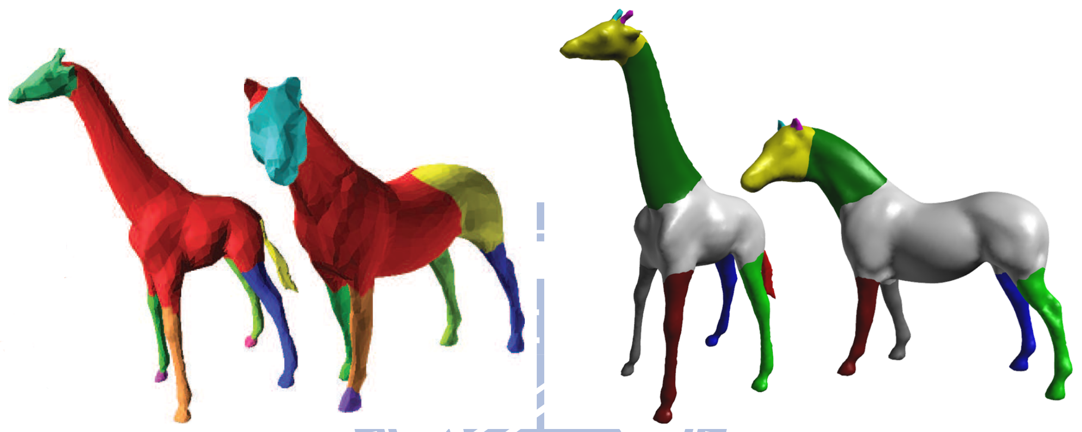
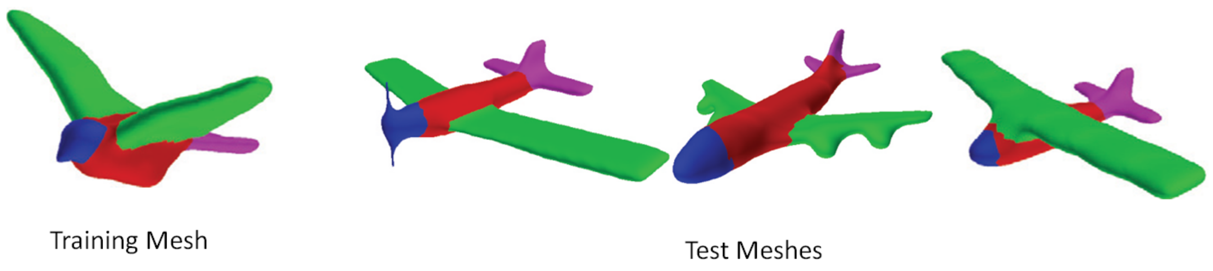
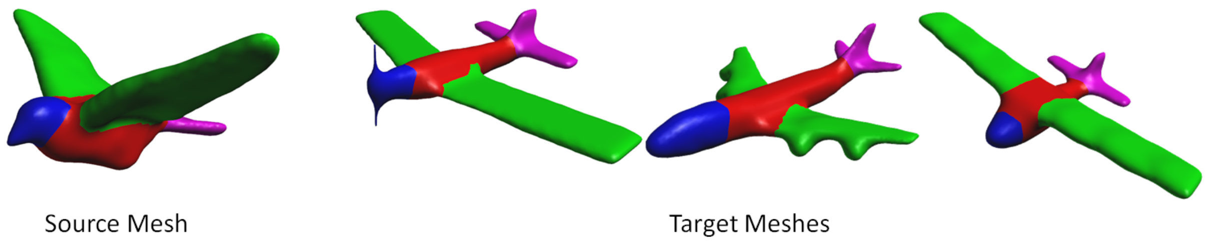

(a) Results of [GF09]. (b) Our results.

Figure 6.17: The comparison with [GF09] on Giraffe and Horse models

We also compare our apporach with the method proposed Kalogerakis et al. [KHS10] which simultaneously produces consistent segmentation and labeling on a set of models. Our method is more capable of controlling the number and location of the boundaries since our method is boundary based. We map the boundaries to the correspondence parts, and the segments are deduced from them. Compare the differences to rightmost airplanes in figure 6.18 and the giraffes in figure 6.19. However, for the models which have different shapes, some additional part will have no boundary and may cause inconsistent results, as shown in figure 6.20. Besides, our method only requires one source model, merely takes few seconds to complete segmentation transfer to a target model (see section 6.3). In contrast, [KHS10] needs multiple models which have attached labels, and costs several hours for training. But due to the constraint of the skeleton construction, our method is suitable to produce part-type segmentation on the articulated models, and is restricted on some man-mode models such as the cup, table and mech of the Segmentation Benchmark [CGF09]. On the other hand, [KHS10] can well handle all categories of the models of the Segmentation Benchmark with any type of segmentation including part-type and patch-type.

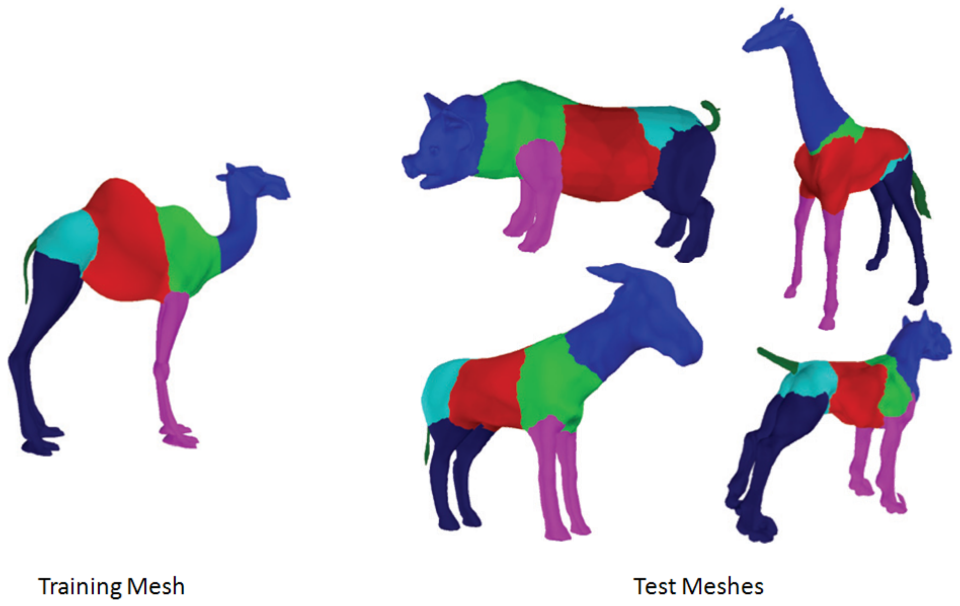


(a) Results of [KHS10].

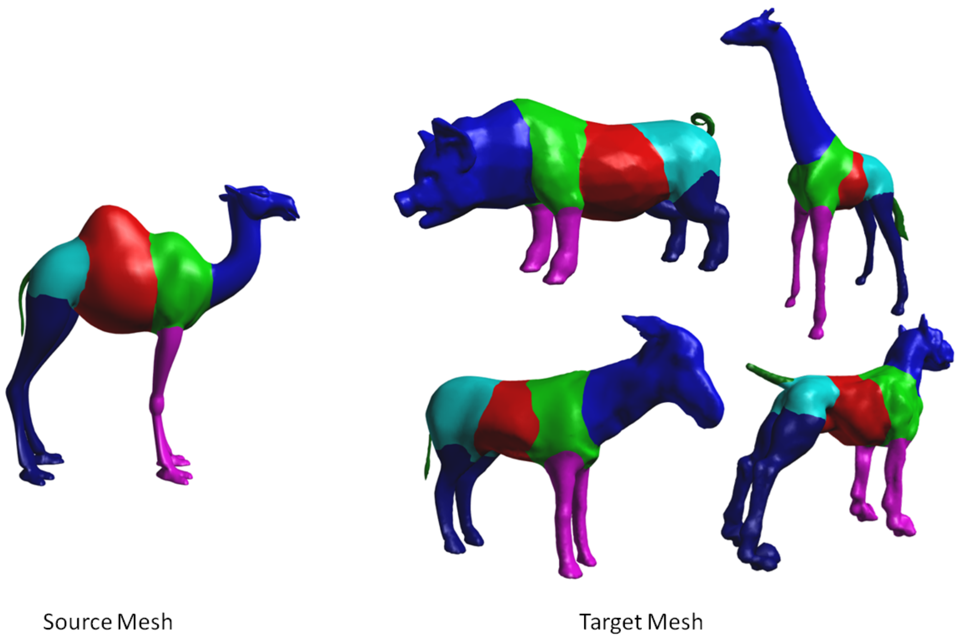


(b) Our results.

Figure 6.18: The comparison with [KHS10] on Airplane models

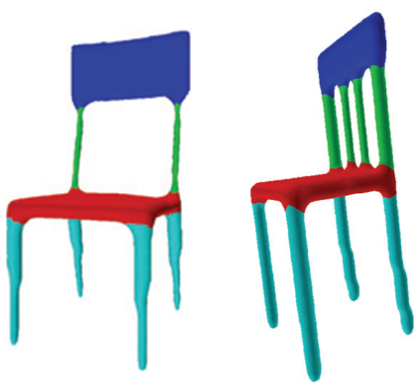


(a) Results of [KHS10].



(b) Our results.

Figure 6.19: The comparison with [KHS10] on Four-legged Animals models


(a) Results of [KHS10].

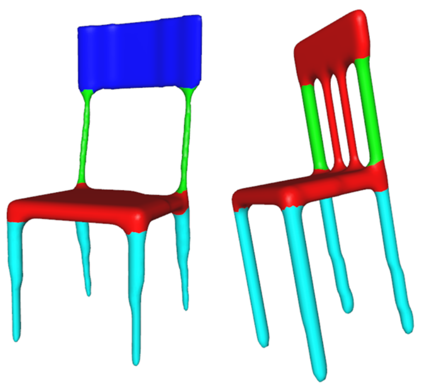
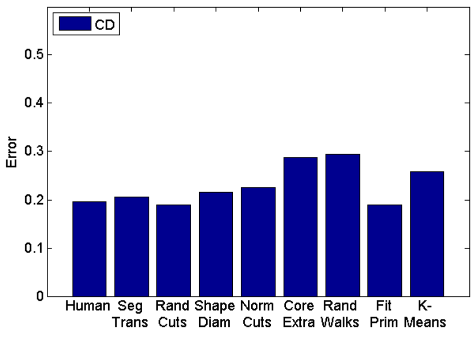
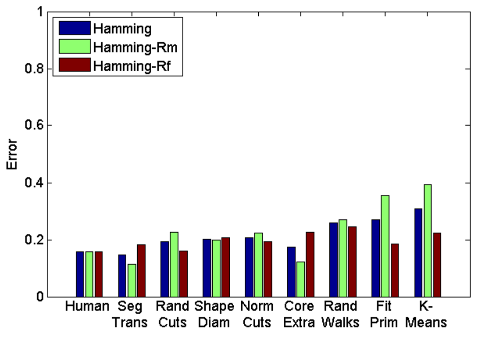

(b) Our results.

Figure 6.20: The comparison with [KHS10] on Chair models

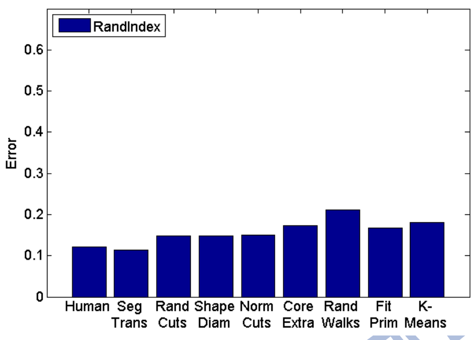
We employ the evaluation metrics proposed in [CGF09] to compare our method with other seven recent automatic segmentation algorithms. Where cut discrepancy measures the position deviation of the boundaries, and hamming distance, randindex, consistency error measure the dissimilarity of the segments. We use ground truth results provided in Segmentation Benchmark [CGF09] as our input source segmentations and apply our method to the same category of models. The categories of the testing models include human, ant, chair, teddy, bird, fish, armadillo and four-legged animals. To have a fair comparison, for each automatic segmentation algorithm we manually reserve the results which are most similar to our input source segmentations (consider the number of segments and the similarity of the segmentation style), then run the benchmark evaluation. Since our method performs the parameterization and snake refinement, the boundary of our results may not lie on the edges of the original mesh. To satisfy the input of the benchmark evaluation, we clamp each point of the boundary to the closest vertex of the original mesh. Figure 6.21 shows the evaluation results. Our segmentation results are closest to ground truth. It implies that if a user provides with a good segmentation reference, our algorithm could generate higher quality results than other algorithms do.



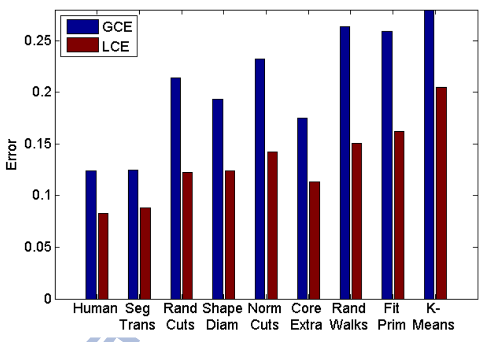
(a) Cut Discrepancy.



(b) Hamming Distance.



(c) RandIndex.



(d) Consistency Error.

Figure 6.21: Comparison with other segmentation algorithms using four evaluation metrics.

6.3 Timing information

Table 6.3, 6.3 and 6.3 list the computation times of our segmentation transfer results presented in figure 6.10, 6.12, 6.13 - 6.16. All the results were performed on Intel Core 2 Duo CPU E8400 3.0GHz with 4GB memory, using a single thread implementation. The source models are marked by a symbol (s) behind their names, and the symbol (ti) indicates that it is the i-th target models of a specific category of models.

For the skeleton correspondence, since our method is based on graph-traversing, the searching space is smaller than the previous methods which apply combinatorial search [ZSCO$^+$08, ATCO$^+$10]. On average the computation times took around 0.3s. A model that has higher resolution will form a more detailed skeleton, which has more internal nodes on each links. So it also leads to the increase in the computation time.

For the segmentation transfer, the computation times depend on the mesh resolutions and the number of boundary in the source segmentation. For the sake of clarity, we list the total computation time, the number of boundary to be transferred in the target model (which may be different to the source model since the target model lack some parts), and the average computation time which are the total time divided by the number of boundary. On average it took around 1.5s for a model with ten thousand faces.

Model	Num. of faces	Skeleton corres. time(sec.)	Segmentation trans. time(sec.)	Num. of boundary	Avg. segmentation trans. time(sec.)
Goat(s)	20868	-	-	9	-
Horse	16152	0.21	1.09	8	0.14
Wolf	9420	0.22	1.25	9	0.14
Reindeer	7818	0.19	0.86	9	0.09
Camel	19510	0.27	1.29	9	0.14
Giraffe	18474	0.29	1.21	9	0.13
Deer	7402	0.21	0.87	9	0.09
Pig	26658	0.28	2.02	9	0.25

Table 6.3: The computation time of the segmentation results in figure 6.10.

Model	Num. of faces	Skeleton corres. time(sec.)	Segmentation trans. time(sec.)	Num. of boundary	Avg. boundary trans. time(sec.)
Dog(source)	18976	-	-	15	-
Deer	7402	0.20	1.31	15	0.09
Human	11258	0.22	1.46	11	0.13
Armadillo	20000	0.24	2.15	15	0.14
Dragon	16000	0.19	1.51	13	0.12
Triceratops	15764	0.21	1.48	15	0.10
Elephant	30000	0.30	3.75	15	0.25
Asian dragon	28198	0.41	3.18	15	0.21
Armadillo(source)	50382	-	-	17	-
Armadillo(target 1)	50212	0.51	6.21	17	0.37
Armadillo(targte 2)	49226	0.60	6.08	17	0.35

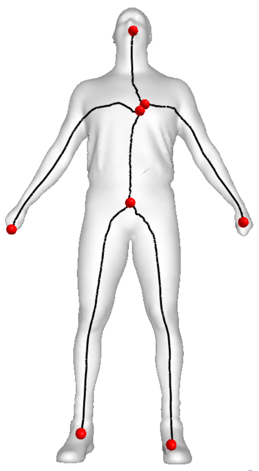
Table 6.4: The computation time of the segmentation results in figure 6.12.

Model	Num. of faces	Skeleton corres. time(sec.)	Segmentation trans. time(sec.)	Num. of boundary	Avg. segmentation trans. time(sec.)
Ant(s)	13696	-	-	10	-
Ant(t)	16124	0.27	1.41	10	0.14
Teddy(s)	27730	-	-	7	-
Teddy(t)	25082	0.50	1.83	7	0.26
Bird(s)	15694	-	-	4	-
Bird(t)	4990	0.14	0.71	4	0.18
Armadillo(s)	50382	-	-	17	-
Armadillo(t1)	50212	0.51	6.21	17	0.37
Armadillo(t2)	49226	0.60	6.08	17	0.35
Chair(s)	16998	-	-	8	-
Chair(t1)	17306	0.29	1.25	8	0.15
Chair(t2)	18530	0.38	1.23	8	0.15

Table 6.5: The computation time of the segmentation results in figure 6.13 - 6.16.

6.4 Discussions and limitations

For the skeleton correspondence, our link clustering process works well on grouping the pairs of parts for the individual skeleton. However, lacking the semantic knowledge easily causes confusions for the cases when there are multiple link clusters attached to a node pair, or the parts that do not exist on the other models(e.g., a wolf only has ears and a cow only has horns, then the ears will map to the horns). In addition, some objects which have flat shape around the junction nodes(the nodes marked by blue circles in figure 6.22), which are difficulty identified the front and back sides by using merely their skeletons and easily cause the wrong flipping correspondences. In some special case such as the hand in figure 6.22, we can utilize the thumb to judge the orientation of the remaining four figures, and for human we can decide the front side by performing the face detection. But it is still hard to find a general solution to solve the problem for various types of models.


(a)

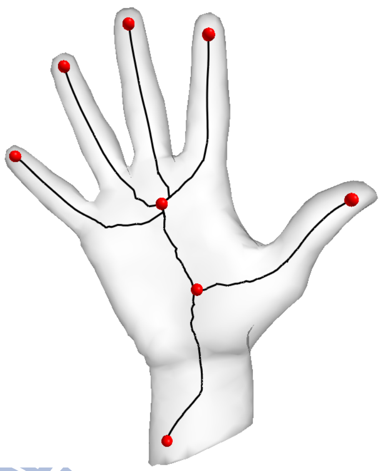
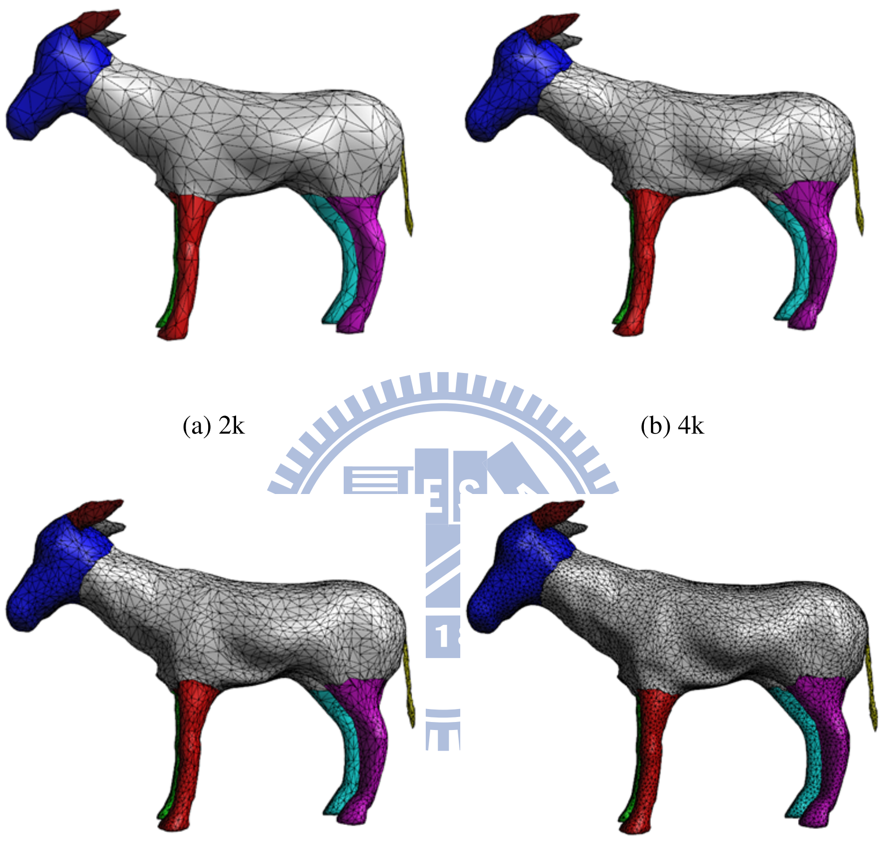

(b)

Figure 6.22: The models which are hard identified the front and back sides by using only skeletons.

For the segmentation transfer, since MSP volume information is not adequate for detecting the surface details such as concavity or local minimum of curvature, and our methods restricts that boundaries must cross to its corresponding links, currently we can only deal with part-type segmentation. Besides, we ask all the boundaries of the source model form a single loop. Consider the case in figure 6.24, the vertex marked by a red circle is passed by multiple boundaries. In our current implementation, we cannot divide them into several acyclic boundaries and transfer them. In addition, we also cannot deal with the boundaries which are large-scale and cross regions belonging to multiple links. Our method rely the local directions of the links to transfer the fitting planes of the boundaries. The approach may not produce good boundaries for the target models which have largely differences in shapes or local skeletal noises(see the wings of the airplanes in figure 6.18). Logically, the effect of constructing the cutting regions is affected by the resolutions of the meshes since the Dijkstra algorithm is used to expand the regions. But we found that it works well for most of our experiments. Figure 6.23 shows an example of the segmentation transfer results for the same model with different resolutions. Note that the consistencies of the boundarys locations between these results.



(a) 2k (b) 4k

(c) 8k (d) 16k

Figure 6.23: The consistent segmentation transfer results for the same model with different resolutions.

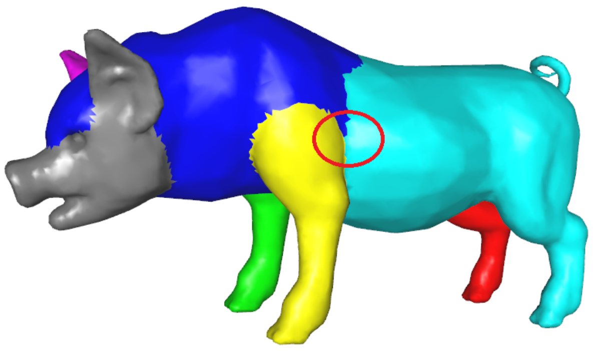


Figure 6.24: A vertex is passed by multiple boundaries.

Furthermore, for some man-made models (e.g., cups, sphere ball, or engineering models) we cannot construct the satisfactory MSP skeletons, and thus affect the processes of skeleton correspondence and segmentation transfer.

CHAPTER 7

Conclusions

In this section, we summarize our segmentation transfer algorithm, and propose several future works.

7.1 Summary

We have presented a novel algorithm that maps a segmentation of the source mesh to the target mesh such that two segmentations are consistent. For the shape correspondence, we use MSP skeleton as shape descriptor of the mesh, and propose a fast skeleton matching scheme to constructs the correspondence of two skeletons. By employing the clustering the similar links and spatial orientations, our scheme is more robust than previous graph-matching based methods. For the segmentation transfer, we apply the MSP gradient analysis along the links to determine the locations of the boundaries. Then construct the cutting regions and map the boundaries through parameterization. Finally the boundaries are refined according to the similarity of shape and surface feature. Our algorithm is fast and effectively produces consistent segmentation results to a variety of objects which have similar semantic structures and different to shape, pose, and geometric detail.

7.2 Future works

To further improve our skeleton correspondence algorithm, we could add other global properties, such as the symmetry, which help to classify the links in more high-level ways (e.g., for the symmetry, we can classify the links into three sets: the first is located at the symmetry line, the second is on the one side and the third is on the other side). We also need some more accurate shape descriptors, or semantic labels to describe the components of the objects. So the links associating with these components could be matched more meaningfully and correctly.

For the segmentation transfer, currently we can only deal with part-type segmentation, since the skeleton correspondence can only describes the parts correspondence between the objects. In the future we will extend our work to find the complete surface mapping of two objects by using the vertices correspondence information of MSP skeleton. At this time boundaries will map more easily and we can handle any kind of segmentation style.

Bibliography

[ACSD$^+$03] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003.

[AFS06] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[AKM$^+$06] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - a comparative study. *IEEE International Conference on Shape Modeling and Applications*, page 7, 2006.

[APP$^+$07] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3D mesh segmentation methodologies for cad applications. *Computer-Aided Design and Applications*, 4(6):827–841, 2007.

[ATC$^+$08] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3), 2008.

[ATCO$^+$10] O. K.-C. Au, C.-L. Tai, D. Cohen-Or, Y. Zheng, and H. Fu. Electors voting for fast automatic shape correspondence. *Computer Graphics Forum*, 29(2):645–654, 2010.

[BH03] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. *International Workshop On Artificial Intelligence and Statistics*, 2003.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[BM92] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[BMSF06] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3D shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[BWK05] S. Bischoff, T. Wey, and L. Kobbelt. Snakes on triangle meshes. *Bildverarbeitung fur die Medizin*, pages 208–212, 2005.

[CGF09] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):1–12, 2009.

[EK03] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1285–1295, 2003.

[FBCM04] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:214–225, 2004.

[GF08] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics*, 27(5):145, 2008.

[GF09] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3D models. *Computers and Graphics*, 33(3):262–269, 2009.

[GG04] N. Gelfand and L. J. Guibas. Shape segmentation using local slippage analysis. *Symposium on Geometry processing*, pages 214–223, 2004.

[HJWC] T. C. Ho, H. X. Ji, S. K. Wong, and J. H. Chuang. Mesh skeletonization using minimum slice perimeter function. *Computer Science Technical Report CS-TR-2010-0001*.

[HR84] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[HS97] D. D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63:29–78, 1997.

[HSKK01] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH 2001*, pages 203–212, 2001.

[JK04] M. Jung and H. Kim. Snaking across 3D meshes. In *12th Pacific Graphics*, pages 87–93, 2004.

[KHS10] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(4):1–12, 2010.

[KJS07] V. Kreavoy, D. Julius, and A. Sheffer. Model composition from interchangeable components. In *15th Pacific Graphics*, pages 129–138, 2007.

[KLT05] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[KT03] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[LF09] Y. Lipman and T. Funkhouser. Möbius voting for surface correspondence. *ACM Transactions on Graphics*, 28(3):1–12, 2009.

[LHMR08] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. *ACM symposium on Solid and Physical Modeling*, pages 183–191, 2008.

[LL02] Y. Lee and S. Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3), 2002.

[LLL07] H.-Y. S. Lin, H.-Y. M. Liao, and J.-C. Lin. Visual salience-guided mesh decomposition. *IEEE Transactions on Multimedia*, 9(1), 2007.

[Llo82] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[LLS$^+$04] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Intelligent mesh scissoring using 3D snakes. In *12th Pacific Graphics*, pages 279–287, 2004.

[LLS$^+$05] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, 2005.

[LMLR06] S. Liu, R.R. Martin, F.C. Langbein, and P.L. Rosin. Segmenting reliefs on triangle meshes. *ACM symposium on Solid and Physical Modeling*, pages 7–16, 2006.

[LWTH01] X. Li, T. W. Woon, T. S. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. *Symposium on Interactive 3D Graphics*, pages 35–42, 2001.

[LZ04] R. Liu and H. Zhang. Segmentation of 3D meshes through spectral clustering. In *12th Pacific Graphics*, pages 298–305, 2004.

[LZ07] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26(3):385–394, 2007.

[MHTG05] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 24(3):471–478, 2005.

[MPS$^+$04] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. *ACM symposium on Solid modeling and applications*, pages 339–344, 2004.

[PKA03] D. Page, A. Koschan, and M. Abidi. Perception-based 3D triangle mesh segmentation using fast marching watersheds. *Computer Vision and Pattern Recognition*, pages 27–32, 2003.

[RT07] D. Reniers and A. Telea. Skeleton-based hierarchical shape segmentation. *IEEE International Conference on Shape Modeling and Applications*, pages 179–188, 2007.

[RT08] D. Reniers and A. Telea. Patch-type segmentation of voxel shapes using simplified surface skeletons. *Computer Graphics Forum*, 27(7):1837–1844, 2008.

[Sha08] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[SSCO08] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008.

[SSGD03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International*, page 130, 2003.

[STK02] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum*, 21(3):219–228, 2002.

[TS04] T. Tung and F. Schmitt. Augmented reeb graphs for content-based retrieval of 3D mesh models. In *Shape Modeling International*, pages 157–166, 2004.

[TVD09] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Partial 3D shape retrieval by reeb pattern unfolding. *Computer Graphics Forum*, 28(1), March 2009.

[vKZHCO10] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. *Eurographics State-of-the-art Report*, 2010.

[WZ10] Y. Wang and J. Zheng. Tubular triangular mesh parameterization and applications. *Computer Animation and Virtual Worlds*, 21(2):91–102, 2010.

[XWLB09] Y. Xu, B. Wang, W. Liu, and X. Bai. Skeleton graph matching based on critical points using path similarity. In *Asian Conference on Computer Vision(3)*, pages 456–465, 2009.

[ZL05] H. Zhang and R. Liu. Mesh segmentation via recursive and visually salient spectral cuts. In *Vision, Modeling, and Visualization*, pages 429–436, 2005.

[ZSCO$^+$08] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. *Symposium on Geometry Processing*, pages 1431–1439, 2008.