行政院國家科學委員會專題研究計畫 成果報告

多邊形網格分割轉換
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 99-2221-E-009-118-
執 行 期 間 ：99 年 08 月 01 日至 101 年 03 月 31 日
執 行 單 位 ：國立交通大學資訊工程學系（所）

計 畫 主 持 人：莊榮宏

計畫參與人員：碩士班研究生-兼任助理人員：徐孟揚
碩士班研究生-兼任助理人員：趙修範
碩士班研究生-兼任助理人員：林育右
碩士班研究生-兼任助理人員：楊筱筑
碩士班研究生-兼任助理人員：廖仁傑
碩士班研究生-兼任助理人員：廖仁豪
碩士班研究生-兼任助理人員：黃程國
碩士班研究生-兼任助理人員：李紹禔
博士班研究生-兼任助理人員：劉文達
博士班研究生-兼任助理人員：李宜亭
博士班研究生-兼任助理人員：何丹期

報 告 附 件 ：國外研究心得報告

公 開 資 訊 ：本計畫可公開查詢

中 華 民 國 101 年 08 月 14 日

中 文 摘 要 ： 我們提出一個一致的網格分割轉換演算法，它可以將來源網格的分割結果對應到目標網格上。這個方法可以對語義上相似但幾何不同的網格做分割轉換。這個方法我們假設需要先提供兩個網格之間的骨架對應關係以及來源網格的分割結果。因此，我們也提出一個新的骨架對應方法，利用骨架節點和其周圍網格頂點的幾何資訊以及拓撲結構來做骨架對應。但是對於不同的部位，骨架也許也會有不同的連結方式，為了降低這種不確定性，我們建立了樹狀結構，產生好幾種不同的結果，然後從中選擇最好的一個。對於分割轉換，我們假設在執行之前需要來源和目標網格、兩個網格之間的骨架對應以及來源網格的分割資訊。有了骨架節點和網格頂點的關係我們先利用骨架的對應關係將來源網格分割的邊界儘量對應到目標網格的邊界上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。

中文關鍵詞： 骨架對應，網格分割，網格分割轉換

英 文 摘 要 ： We present a skeleton-based approach for consistently transferring a segmentation of a source mesh to a target mesh. The source and target meshes are assumed to be semantically similar but may have different geometry. For the given two meshes, the segmentation is transferred based on the skeleton correspondence. Hence we also develop a novel method for finding skeleton correspondence. The segmentation of the source mesh is transferred to the target by first using the skeleton correspondence and the skeleton-to-surface mapping and then the transferred segmentation is refined based on the geometric details of the source segmentation.

英文關鍵詞： Skeleton correspondence, Mesh segmentation, Segmentation transfer

行政院國家科學委員會補助專題研究計畫■**成果報告**
□**期中進度報告**

多邊形網格分割轉換

計畫類別：■個別型計畫　□整合型計畫
計畫編號：NSC 99-2221-E-009-118
執行期間：99 年 8 月 1 日至 101 年 3 月 31 日

執行機構及系所：

計畫主持人：莊榮宏　　國立交通大學資訊工程系教授
共同主持人：
計畫參與人員：李宜亭、劉文達、徐孟揚、趙修範、林育右、楊筱筑、何丹期、廖仁傑、廖仁豪、黃程國、李紹禔

成果報告類型(依經費核定清單規定繳交)：■精簡報告 □完整報告

本計畫除繳交成果報告外，另須繳交以下出國心得報告：
■赴國外出差或研習心得報告
□赴大陸地區出差或研習心得報告
□出席國際學術會議心得報告
□國際合作研究計畫國外研究報告

處理方式：除列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

中　華　民　國 101 年 6 月　4　日

中文摘要

我們提出一個一致的網格分割轉換演算法，它可以將來源網格的分割結果對應到目標網格上。這個方法可以對語義上相似但幾何不同的網格做分割轉換。這個方法我們假設需要先提供兩個網格之間的骨架對應關係以及來源網格的分割結果。因此，我們也提出一個新的骨架對應方法，利用骨架節點和其周圍網格頂點的幾何資訊以及拓撲結構來做骨架對應。但是對於不同的部位，骨架也許也會有不同的連結方式，為了降低這種不確定性，我們建立了樹狀結構，產生好幾種不同的結果，然後從中選擇最好的一個。對於分割轉換，我們假設在執行之前需要來源和目標網格、兩個網格之間的骨架對應以及來源網格的分割資訊。有了骨架節點和網格頂點的關係我們先利用骨架的對應關係將來源網格分割的邊界儘量對應到目標網格的邊界上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。

關鍵詞： 骨架對應，網格分割，網格分割轉換

ABSTRACT

We present a skeleton-based approach for consistently transferring a segmentation of a source mesh to a target mesh. The source and target meshes are assumed to be semantically similar but may have different geometry. For the given two meshes, the segmentation is transferred based on the skeleton correspondence. Hence we also develop a novel method for finding skeleton correspondence. The segmentation of the source mesh is transferred to the target by first using the skeleton correspondence and the skeleton-to-surface mapping and then the transferred segmentation is refined based on the geometric details of the source segmentation.

Keywords: Skeleton correspondence, Mesh segmentation, Segmentation transfer

1. 前言

特徵或幾何運算之轉換在幾何處理是一個重要的研究課題。此研究之動機是現今網格分割的諸多方法各有其優缺點，各有其適合的網格型態，通常一個方法無法對所有物體產出好的分割結果，而且分割的結果常與人為分割結果有所差距。我們的方法可以將一個好的網格分割結果轉換到一個類似的物體。目前我們的作法比較是以幾何出發，藉由骨架對應的方法先將來源網格分割的邊界儘量對應到目標網格上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。此方法對肢體動物之測試成果不錯，但對形體拓樸差異較大者之效果表現不太穩定，這次與馬教授討論的方向是希望可以應用機器學習的觀念到網格分割轉換。

2. 文獻探討

形體對應（shape correspondence）

形體對應在幾何處理是一個很重要的研究課題。形體對應通常是使用 Reeb graph 或物體的骨架來做對應[HSKK01, TS04, SSGD03, BMSF06]。在 Graph matching 的方法中，通常子圖（subgraph）會被根據幾何性質的最小誤差來進行一對一的對應。而這類方法通常會有兩個問題。一個是對網格的拓撲過於敏感，因為一般的子圖建立是依靠圖形的連結性。另一個則是對稱性交換[ZSCO*08]。例如左右手常會被對應錯誤，因為這樣的骨架會有相同的語義以及相似的幾何資訊。而我們的方法利用骨架節點和網格頂點之間的關聯、骨架的空間資訊來解決第一個問題。第二的問題則使用交叉節點連結的骨架進行局部的對應來避免。我們的方法會適當的合併骨架節點或是骨架分支，這會產生多對多的對應結果，並提高正確對應的可能性。而[ATCO*10]則是利用投票(voting)的方式去計算一對一個對應，根據票數高低來選出最好的結果。這個方法並不能合併任何的骨架節點，因為投票方法對於不同的合併設計有不同的意義，而目前並不清楚如何根據投票來調整最好的對應結果。而我們的方法則是採用計算最小成本的方式，因此可以得到最好的結果。

網格分割(mesh segmentation)

大多數網格分割的演算法大致分為兩種：一種是將網格分成一塊塊補丁(patch)，且每塊補丁都滿足一些屬性；另一種則是將網格分割成對人類來說具有語義的組成(如 part)。不同的分割演算法對於同一個網格通常會產生不同的分割結果，而一致的網格分割目的就在於能夠同時對一組網格作一致的分割。[GF09]考慮了個別網格的幾何特徵以及網格之間的對應資訊，利用對

齊的方式建立階層式的一致的分割結果。[KHS10]則提出了一個計算分割並對網格分配標籤的設計。他們把標籤的分配當作一個優化的問題，並設計一個目標函數去評估這些網格的元素組成(例：三角形) 的標籤的一致性。而我們的方法是利用骨架的對應去做網格分割的轉換，並且考慮了幾何的細節，因此會和來源網格的分割結果很相似。

3. 骨架對應

3.1 方法

在骨架對應的過程中我們會建立一個對應樹，每個樹節點表示到當時的骨架節點對應以及骨架分支。初始的對應樹只有一個根節點，就是還沒有任何節點配對的兩骨架。第二層我們會對骨架的接合節點(junction node)作對應，對應樹每產生一種新的骨架節點對應就會建立一個樹子節點在目前的檢查的節點上。接下來會使用遞迴的方法反覆去對剩下的骨架節點和骨架分支作對應直到沒有骨架節點可以作對應為止。對應樹的最低層，每一 leaf 代表一種可能的對應結果。我們會從中評估出成本最低的作為最好的對應結果。如圖 1 所示。

方法一開始我們會先對骨架做簡單的處理，去除多餘不必要的骨架節點以及不重要的骨架分支。接著我們會做接合節點的對應，我們利用接合節點和物體中心的距離以及以節點分段所形成的各種子骨架(見圖2)的相似程度來評估;若評估結果小於使用者設定的門檻值，就是對應成功，會建立一個樹的節點;若是大於使用者設定的門檻值,就可能會合併鄰近的接合節點或是合併骨架分支，這是因為也許骨架並不完美造成多餘的節點或分支卻在前處理無法清除的關係，此步驟可以校正減少對應錯誤的可能(見圖3和圖4)。之後我們會對目前有的對應節點，對它們的分支做對應，我們考慮了這些分支的相似程度以及空間的資訊。因為模型有些部位具有相同的語義，而它們通常會有相同的幾何特徵以及相似的形狀，因此我們會對骨架分支做分群，以減少錯誤對應以及降低搜尋的空間。方法流程如圖5所示。

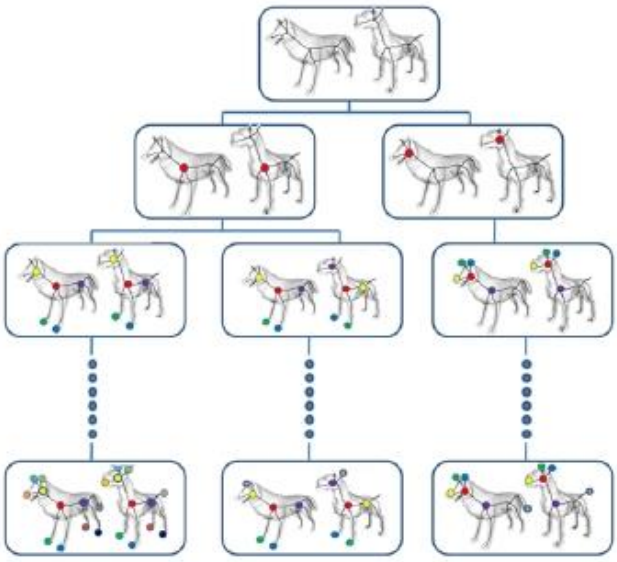

圖1 骨架對應的樹狀結構示意圖。

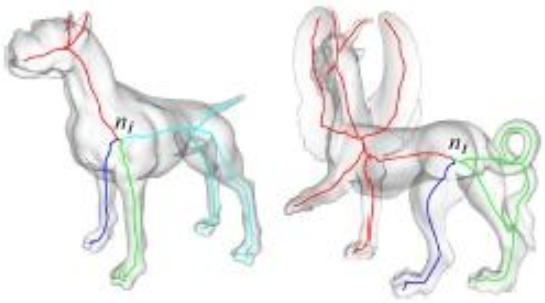

圖2 不同顏色代表不同的子骨架。

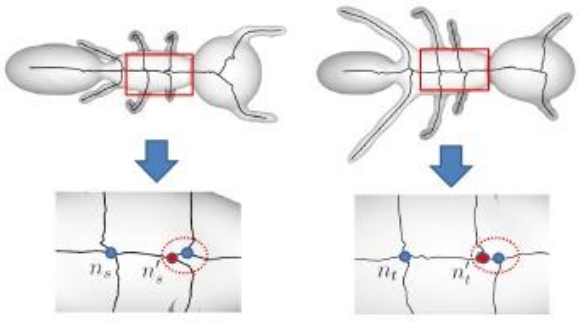

圖3 交叉節點的合併。

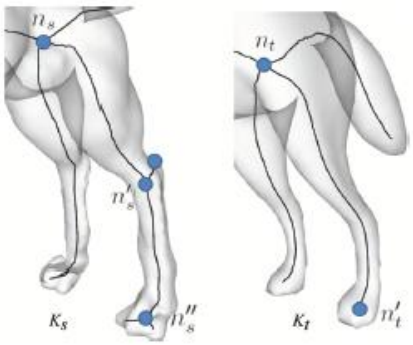

圖4 骨架分支的合併。

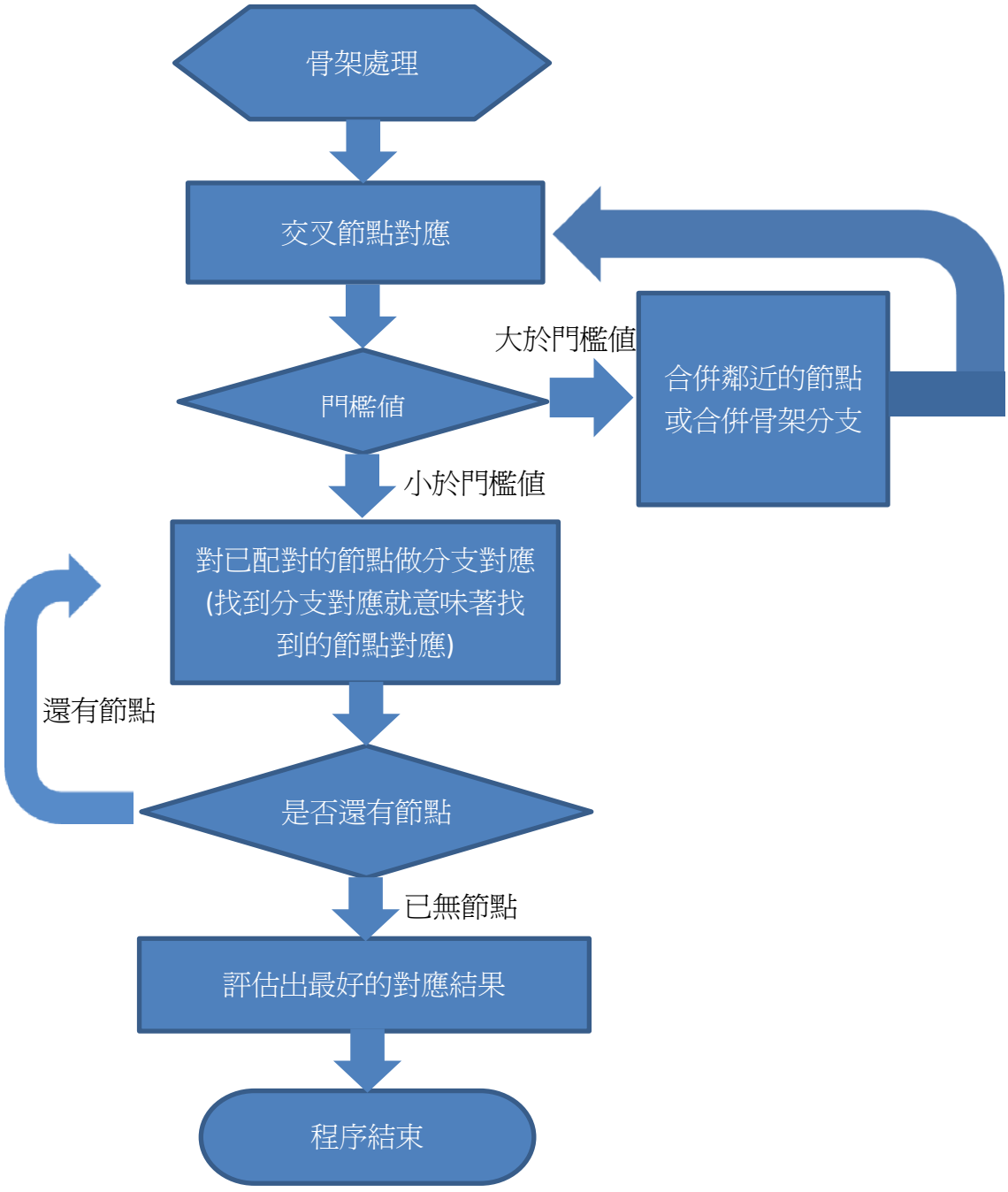

圖 5 方法流程圖。

3.2 測試結果

我們用好幾種模型證實我們的方法可以有效地找出正確的對應結果，如圖6所示。對應的節點會顯示出相同的顏色，沒有對應到的節點則不會顯示在圖上，語義相似的部位都可以有正確的對應。和[ATCO*10]做比較(見圖7)，可以很明顯地看到我們的結果比較好。[ATCO*10]在馬的耳朵和狗的嘴巴部分會對應錯誤，在豬的耳朵和龍的上顎也對應錯誤。

但是我們的方法在有些語義的對應仍有可能會出現錯誤，例如馬的耳朵和牛的牛角。還有在接合節點附近若是其形狀較為平緩，我們會無法用骨架推斷出前後，因此有可能會出現錯誤的對應。此外，若是人造模型我們無法取得較好的骨架或是骨架無法忠實地捕捉模型的形狀，那我們的結果就會受到很大的影響。

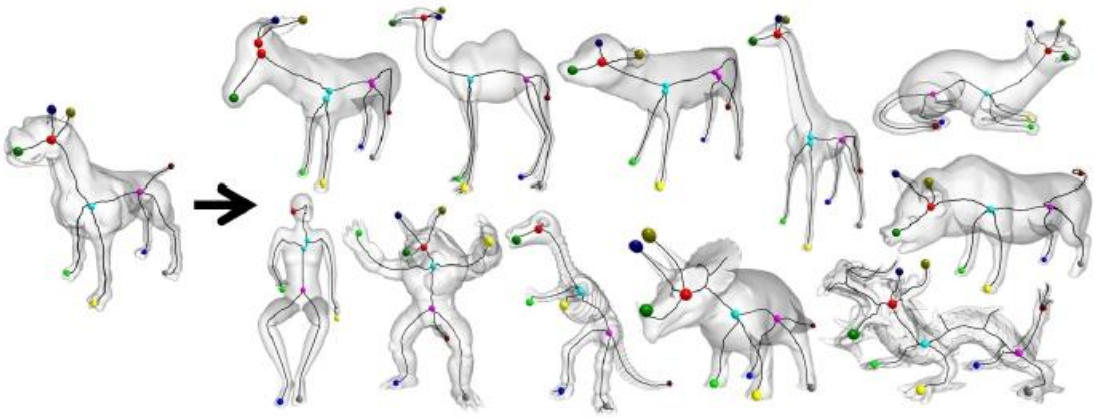

圖6 對不同模型的對應結果。

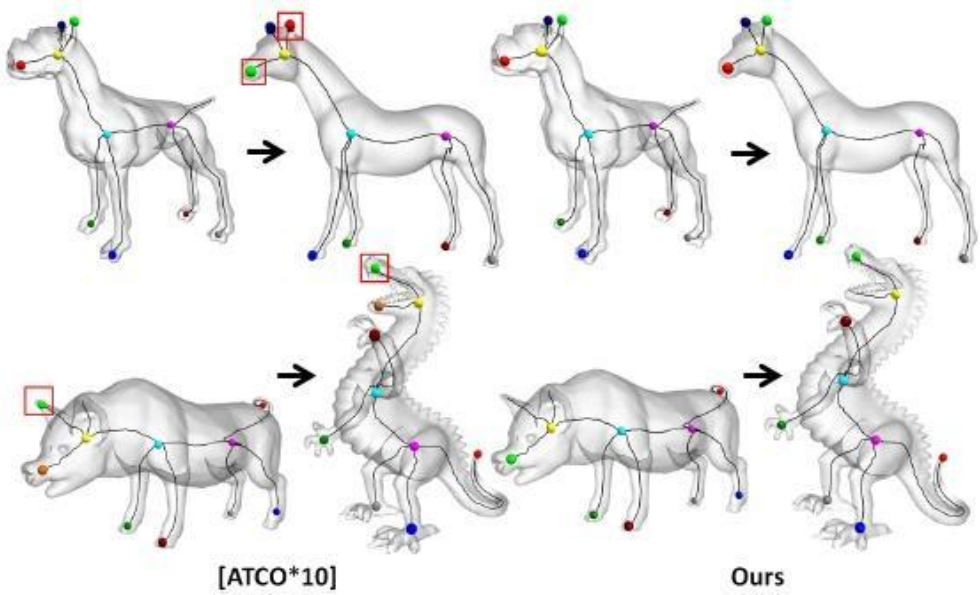

圖7 與[ATCO*10]比較結果。

4. 網格分割轉換

4.1 方法

要執行這個方法之前，我們需要有來源和目標網格及其骨架、兩個骨架之間的對應關係以及來源網格的分割資訊。我們會利用兩個骨架之間的對應關係，將來源網格的分割邊界轉換到目標網格上，再參考來源分割邊界形狀對目標分割邊界作幾何修正，最後就會得到目標網格的分割結果。

方法主要為三大步驟，首先我們會先做骨架邊界的對應，這是利用兩個骨架之間的對應關係來計算目標網格的分割邊界大約的位置。接著我們會做邊界轉換，由於我們的骨架萃取的方法可以取得骨架節點和一組網格頂點對應，因此骨架節點和網格頂點可以用來建立骨架節點和分割邊界的對應。邊界轉換可以將來源網格的分割資訊根據骨架對應的結果盡可能地和目標網格邊界一致。最後我們作邊界修正，它會調整目標網格的分割邊界，此步驟目的在於讓它的幾何細節能夠和對應的來源分割邊界一致。方法流程如圖8所示。

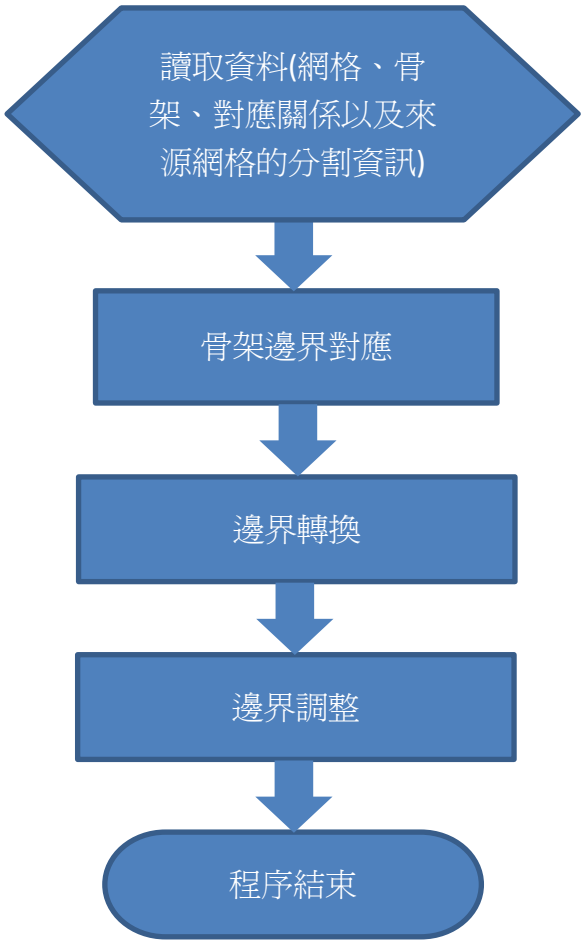

圖 8 方法流程圖。

4.2 測試結果

圖9可以看到我們的實驗結果，對應的部位會有相同的顏色。我們可以將來源網格的分割結果正確的轉換到目標網格上。在圖9，狗的耳朵和人的模型並沒有對應，因此分割的結果就不會轉換到人的模型上，但其他同類型的模型都可以有正確的分割轉換。我們將方法和一些近期的方法做比較(見圖10、圖11、圖12)。在圖10，可以很明顯的看到[GF09]在模型有較大不同形狀的時候，並不能很好的轉換，但我們的方法因為有做骨架的對應，因此可以有不錯的結果。圖11和圖12是和[KHS10]做比較的結果，我們的邊界切割結果更好。但是我們的結果好壞取決於骨架對應的結果，如果骨架無法很好的捕捉到網格的形狀，我們的分割結果會受到影響，這算是一個限制。

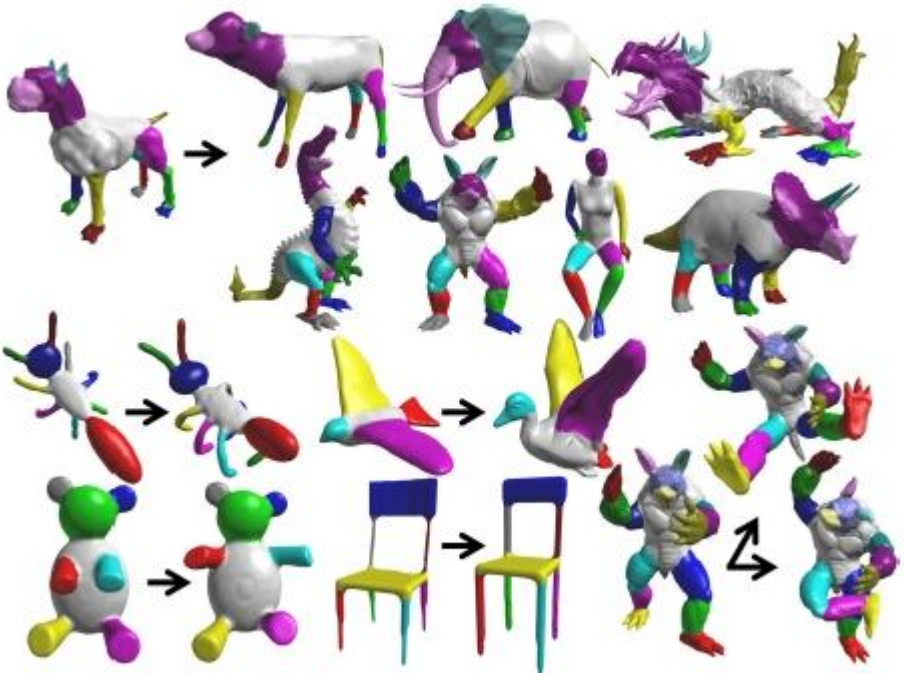

圖9 網格分割轉換結果。

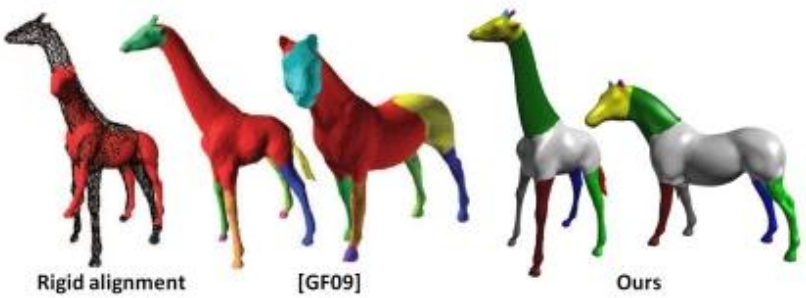

圖10 與[GF09]比較結果。

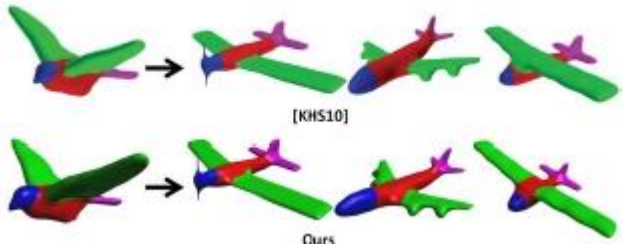

圖11 與[KHS10]比較結果。

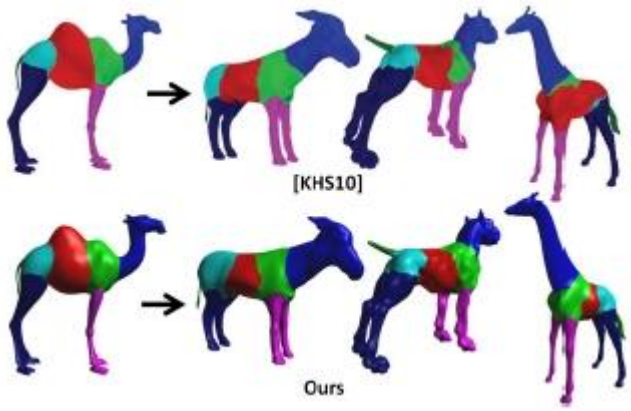

圖12 與[KHS10]比較結果。

5. 結論

載此計畫中，我們提出兩個方法。一個是骨架對應方法，另一個是一致的分割轉換方法。骨架對應的方法合併了骨架節點和骨架的分支使得對應的結果會是多對多的對應。我們測試許多相似的物體對應，結果顯示我們可以計算出適當的對應結果。分割轉換方法根據兩個網格的骨架對應以及骨架節點和網格頂點的對應關係能夠將來源網格的分割邊界轉換到目標網格上。我們使用很多相似但不同形狀和動作的網格作測試，結果都相當不錯。但是我們的結果也同樣會受到骨架對應的影響，如果骨架對應的結果不好，我們的分割轉換的結果同樣也會不好。但我們認為骨架對應可以提高分割轉換的結果，因此我們未來將進一步是研究出更好的骨架對應方法以及對於形狀不類似的網格都能做正確的分割轉換。

6. 自我評估

在此計劃中，我們提出一個一致的網格分割轉換的演算法，它可以將來源網

格的分割結果對應到目標網格上。來源和目標網格語義上需要相似，但幾何上可以有所不同。我們先發展出一個兩個網格間的新的骨架對應方法，利用骨架節點和其周圍網格頂點的幾何資訊以及拓撲結構來做骨架對應。但是對於不同的部位，骨架也許也會有不同的連結方式，為了降低這種不確定性，我們建立了樹狀結構，產生好幾種不同的結果，然後從中選擇最好的一個。而網格分割轉換就藉由骨架對應先將來源網格分割的邊界儘量對應到目標網格上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。這兩個網格處理的演算法都具相當的創新及產業應用價值。這兩項成果都已撰寫成論文，正投稿國際期刊中。

7. 參考資料

[ATCO_10] AU O.-C., TAI C., COHEN-OR D., ZHENG Y., FU. H.: Electors voting for fast automatic shape correspondence. *Comp. Graph. Forum 29*, 2 (2010), 645–654.

[BL08] BAI X., LATECKI L.: Path similarity skeleton graph matching. *IEEE Trans. on PAMI 30* (2008), 1282–1292.

[BMSF06] BIASOTTI S., MARINI S., SPAGNUOLO M., FALCIDIENO B.: Sub-part correspondence by structural descriptors of 3D shapes. *Comp. Aided Design 38*, 9 (2006), 1002–1019.

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Consistent segmentation of 3D models. *Comp. Graph. Forum 33,* 3 (2009), 262-269

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity

[KHS10] KALOGERAKIS E., HERTZMANN A., Singh K.: Learning 3D mesh segmentation and labeling. *ACM TOG Forum 29,* 4 (2010) 1-12

[SSGD03] SUNDAR H., SILVER D., GAGVANI N., DICKINSON S.: Skeleton based shape matching and retrieval. In *Shape Modeling Int'l* (2003), p. 130.

[TS04] TUNG T., SCHMITT F.: Augmented reeb graphs for content-based retrieval of 3D mesh models. In *Shape Modeling Int'l* (2004), pp. 157–166.

[ZSCO*08] ZHANG H., SHEFFER A., COHEN-OR D., ZHOU Q., VAN KAICK O., TAGLIASACCHI A.: Deformation-driven shape correspondence. *Symposium on Geometry Processing* (2008), 1431–1439.

EUROGRAPHICS Workshop on ... 2012
P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (*2012*), *Number 2*

A Skeleton-Based Approach for Shape Correspondence

Online submission ID:1077

Abstract

We present a novel skeleton-based approach for performing shape correspondence by exploiting the mesh geometric properties associated with the skeletons of the objects. The skeletal nodes which correspond to the similar parts of the two objects are matched. Current techniques can generate the skeleton of an object such that the skeletal nodes are associated with the local vertices of the object. The vertices scatter around the skeletal nodes and they are useful for computing the geometric properties of the object parts corresponding to the skeletal nodes. We compute a set of potential shape correspondences and then evaluate the best one. Our method may merge adjacent skeletal nodes and adjacent skeletal branches so that it can compute not only an 1-1 correspondence but also a many-many correspondence. We apply our method to various models with similar topological structures and compare with the latest techniques. Experimental results show that our method can compute acceptable shape correspondence.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Shape correspondence establishes meaningful relations between geometric elements of objects ([vKZHCO10]) and it has seen applications in a variety of domains, such as shape registration, information transfer, recognition and shape retrieval. A meaningful correspondence depends on the problem domain. Our focus is on matching elements with similar geometric properties and semantic meanings for objects of different shapes.

Skeleton is an important shape descriptor which provides the geometrical and structural information of the objects. A skeletal node often represents a part of an object. Using skeletons to compute shape correspondence is possible. The topological structure and geometry properties of the objects should be considered. Fig. 1 shows an example.

Recently, there are techniques which generate skeletons whose skeletal nodes and skeletal branches are associated with the local vertices of the objects. In [ATC*08], a method is proposed to adopt implicit Laplacian smoothing for shrinking an object and the refinement process is then adopted for generating the 1-D curved skeleton of the object. During the refinement process, some vertices are retained and merged into skeletal nodes. Hence, the skeletal nodes are associated with some object vertices. This association can be used for computing shape correspondence. We observe that the corresponding parts of two similar objects often share similar geometric information and the skeletal nodes around these parts have similar topological structures.

In this paper, we propose a novel skeleton-based approach for computing shape correspondence. The skeletal nodes which correspond to the similar parts of the two objects are matched. We exploit the association between skeletons and vertices of the two objects. The vertices scatter around their associated skeletal nodes and they are useful for computing the geometric properties of the object parts corresponding to the skeletal nodes. However, the skeletons may have different connectivity for different parts. To reduce the influence of the uncertainty, we compute a set of potential shape correspondences of the two objects and then evaluate the best one. Experimental results show that our method computes acceptable shape correspondence between objects with similar topological structures and semantics.

Employing skeletons for computing shape correspondence is not new. In [ATCO*10], the skeletons of two objects are used for computing an 1-1 correspondence between skeletal nodes. The skeletons produced by [ATC*08] are input to their method. Then their method computes a set of reliable skeletal nodes for voting and the correspondence with the highest vote count is selected as the best correspondence. The method does not merge any skeletal nodes. Indeed, if merging skeletal nodes is permitted, the total vote count on the skeletal nodes are different. It is not clear how the vote

counts are adjusted for computing the best shape correspondence. This is because the vote counts for different merging schemes may have different meanings. In our method, we may merge skeletal nodes and obtain different shape correspondences for different merging schemes. But we can still compute the best shape correspondence which has the minimum cost. Our method may compute not only 1-1 correspondences but also many-many correspondences between skeletal nodes of two objects.

A method based on path similarity is considered for computing correspondence between the skeleton graphs of two 2D images [BL08]. A skeleton is treated as a set of geodesic paths formed by the terminal skeletal nodes. The paths of the two skeletons are matched by using sequence matching. Their method can match the terminal skeletal nodes but it cannot match the internal skeletal nodes. Our method relies on the spatial information of skeletal branches and the association between the skeletons and vertices of objects. Our method can match both the external and internal skeletal nodes of the two skeletons.

An approximate skeleton which is a planar graph having at most degree three, is obtained by tracing the medial segments of the inner triangles in a constrained Delaunay triangulation of the polygon. Mortara and Spagnuolo [MS01] proposed a method for matching the approximate skeletons of 2D polygonal objects according to the similarity of the skeletal junction nodes and arcs. Our approach adopts the similar approach for matching skeletons based on the similarity of junction nodes and branches. However, our computation is involved with the association between the skeletal nodes and vertices of 3D objects. Furthermore, we cluster similar branches so as to reduce the searching space.

2. Related Work

Shape correspondence is one of the important research topics in geometry processing. A comprehensive survey can be found in [vKZHCO10]. We review the approaches for shape correspondence that are mostly related to our work.

Graph-matching has been adopted for matching the Reeb graphs or skeletons of objects [HSKK01, TS04, SSGD03, BMSF06]. Common subgraphs are computed so that the nodes of the common subgraphs form an one-to-one mapping with the minimum deviation of geometric properties. These approaches have two common problems: (1) sensitive to mesh topology and (2) symmetry switching [ZSCO*08]. The first problem is due to that the construction of the common subgraphs depends on the graph connectivity. These approaches may be affected by small unwanted branches and topological difference. The second problem is due to that some skeletal nodes representing the same semantic parts have similar geometric information, such as left/right hands in a human model. The left hand may be mapped to the right hand. We attempt to handle these two problems. The first problem is handled by using the association between skeletal nodes and vertices, and the spatial information of the skeletons. We may merge close-by junction nodes and group similar skeletal branches as clusters. Most of the bad correspondences can be eliminated. The second problem is tackled by using shape matching locally for the skeletal branches attached at the skeletal junction nodes. We observe that the orientation of the skeletal branches does not change much for most objects with different poses. We can match the skeletal branches attached at the junction nodes based on the local orientation of the skeletal branches.

A combinatorial search is adopted to compute electors and they are applied in the cascading pruning tests [ATCO*10]. Each elector votes on individual feature-to-feature matching for computing the final correspondence. The approach may mismatch the close-by nodes. This is because inconsistent results may be obtained by using MDS (multi-dimensional scaling) transformation [EK03]. In our approach, we may merge the adjacent skeletal junction nodes and the adjacent branches so as to avoid the mismatched problem. In the elector voting approach, the vote count is not easy to be justified for skeletons with various merged skeletal nodes and merged skeletal branches. Furthermore, the final correspondence of their approach is determined based on the vote count. The connectivity between the matched skeletal nodes may be messed up. On the other hand, we consider the spatial information of skeletal branches for matching and hence we do not have this problem.

Path similarity is considered for computing correspondence between two skeleton graphs of two images [BL08]. The method can match only the terminal nodes as they consider only the geodesic paths which connect terminal nodes. Mortara and Spagnuolo [MS01] proposed a method for evaluating the similarity between skeletal junctions and arcs so as to match the 2D skeletons. Our method shares the similar approach of these two techniques, but we also consider the association between the skeletal nodes and vertices that is important for analysing the geometric properties around the skeletal nodes.

3. Overview

We present the definitions, notations and a brief overview of our approach in this section.

3.1. Definitions and Notations

A skeletal node is associated with a set of vertices after the skeleton is built, such as the method in [ATC*08]. The set of vertices associated with a skeletal node often lay around it. A skeletal node having one adjacent skeletal node is a terminal node. A skeletal junction node is a skeletal node having three or more adjacent skeletal nodes. A feature skeletal node is either a terminal node or a junction node. A skeletal branch

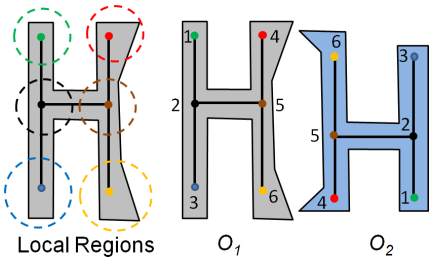

Figure 1: *Using the association between the skeletons and meshes of the objects for shape correspondence. (Left): Skeletal nodes of an object are associated with the local portions (marked by the dotted circles) of the mesh of the object; Middle and Right: the mapping between the skeletal nodes of two H-shaped objects O_1 and O_2. If the topological structures of the two skeletons are not considered, the four terminal nodes of O_1 can be mapped arbitrarily to the terminal nodes of O_2. If geometry properties are not considered, the skeletal node (1) of O_1 may be mapped to the skeletal node (6) of O_2.*

connects only two feature skeletal nodes. Given a branch $b = (n_1, n_2)$, we define $b/n_1 := n_2$ and $b/n_2 := n_1$.

A mesh Ω of an object is a pair (V^Ω, F^Ω), where V^Ω and F^Ω are the set of the vertices and the set of the faces, respectively. An augmented skeleton K is a triple (V, E, ϕ), where V is a set of skeletal nodes, E is the set of skeletal arcs connecting two adjacent skeletal nodes, and ϕ is a mapping between the vertices of the object and the skeletal nodes. Formally, $\phi : V \to \wp(V^\Omega)$, where $\wp(V^\Omega)$ is the power set of V^Ω. The mapping ϕ defines the association between the skeletal nodes and the vertices.

A measure at a vertex is a scalar function $\varphi : V^\Omega \to R$ that maps a vertex to a real number. The function φ is a kind of surface descriptor at a vertex. For example, the averaged geodesic distance can be measured at a vertex. We define $\varphi(n) := \frac{1}{|\phi(n)|}\sum_{v\in\phi(n)}\varphi(v)$, where $\varphi(n)$ is the value evaluated at a skeletal node n and $|\phi(n)|$ is the cardinality of the set $\phi(n)$.

A skeletal point p lies on a skeletal arc (n_1, n_2). Consider that the skeletal point is expressed as a linear combination of the two skeletal nodes, i.e. $p = \alpha n_1 + (1-\alpha)n_2$. To evaluate the value $\varphi(p)$ at a skeletal point p, we have $\varphi(p) = \alpha\varphi(n_1) + (1-\alpha)\varphi(n_2)$. Here, we focus on linear interpolation for computing the measure in a skeletal point. The other interpolation schemes are possible.

3.2. Algorithm Overview

We exploit the association between the skeletons and meshes of the objects to compute shape correspondence. We compute a set of potential correspondences and then select the best one. Each skeletal node is associated with a set of local vertices of the mesh. Notice that the vertices mapped to a skeletal node scatter locally on the mesh. The association roughly represents the mesh properties with the skeletal nodes. Fig. 2 shows a skeleton of a horse model.

Algorithm 1 Augmented Skeleton Correspondence

1: **Input:** two augmented skeletons K_s (source) and K_t (target)
2: **Output:** The correspondence between K_s and K_t
3: **Initialization**
4: The root node of the correspondence tree = R
5: $R.MatchingInfo = \emptyset$; *store the matching info. between K_s and K_t*
6: PerformShapeCorrespondence(R, K_s, K_t)
7: $S \leftarrow$ ComputeBestFinalCorrespondence($R.LeafNodes$)
8: **return** S

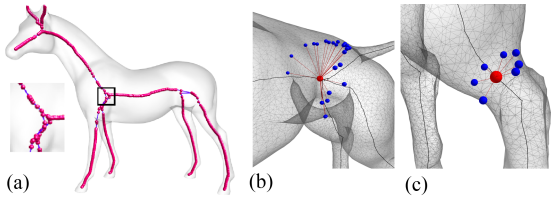

Figure 2: *A skeleton and the association between the object vertices and some skeletal nodes. (a): A skeleton of a horse model. There are skeletal nodes along each skeletal branch. (b): A skeletal junction node (red) and the vertices (blue) associated with it. (c): A skeletal node (red) with degree two and the vertices (blue) associated with it.*

The input of our method include two meshes, their skeletons and some geometric information of the two meshes. We start by matching the core junction nodes and then proceed to match the remaining skeletal branches and skeletal nodes in an interleaving manner. The entire method can be divided into four parts which are Alg. 1,2, 3 and 4. Notice that we may merge skeletal nodes during the process (see Alg.2). We describe the details in the following sections.

Algorithm 2 PerformShapeCorrespondence(P, K_s, K_t)

1: **Input:** the node P of correspondence tree; augmented skeletons K_s and K_t
2: matchOneCoreJunctionNodePair(P, K_s, K_t)
3: $K_s' \leftarrow$ mergeTwoAdjacentJunctionNodes(K_s)
4: **if** $K_s' \neq K_s$ **then**
5: add a child C_1 to P
6: PerformShapeCorrespondence(C_1, K_s', K_t)
7: **end if**
8: $K_t' \leftarrow$ mergeTwoAdjacentJunctionNodes(K_t)
9: **if** $K_t' \neq K_t$ **then**
10: add a child C_2 to P
11: PerformShapeCorrespondence(C_2, K_s, K_t')
12: **end if**
13: **if** $K_s' \neq K_s$ and $K_t' \neq K_t$ **then**
14: add a child C_3 to P
15: PerformShapeCorrespondence(C_3, K_s', K_t')
16: **end if**

Our method requires four parameters which are listed in

Algorithm 3 matchOneCoreJunctionNodePair(P, K_s, K_t)

Input: the node P of correspondence tree; augmented skeletons K_s and K_t
for each node $n_s \in K_s$ **do**
  for each node $n_t \in K_t$ **do**
    if n_s can be matched with n_t **then**
      add a child C to P
      $C.MatchingInfo \leftarrow P.MatchingInfo \cup (n_s, n_t)$ //set matching info. of C.
      matchRemaining(C, n_s, n_t, K_s, K_t)
    end if
  end for
end for

Algorithm 4 matchingRemaining(P, n_s, n_t, K_s, K_t)

Input: the node P of correspondence tree;matched junction node pairs (n_s, n_t); augmented skeletons K_s and K_t
$(B, K_1', K_2') \leftarrow$ matchBranches(n_s, n_t, K_s, K_t)
if B != nil **then**
  add a child C to P
  $C.MatchingInfo \leftarrow P.MatchingInfo \cup B$
  for each $(b_s, b_t) \in B$ **do**
    (n_s', n_t', K_s', K_t') <- matchJunctionNodePair($C, b_s/n_s, b_t/n_t, K_s', K_t'$)
    if (n_s', n_t') != nil **then**
      add a child C_1 to C
      $C_1.MatchingInfo \leftarrow C.MatchingInfo \cup (n_s', n_t')$
      matchRemaining($C_1, n_s', n_t', K_s', K_t'$)
    end if
  end for
end if

Table 1. The computation is fully automatic once the parameters are given. In practice, a user can usually obtain acceptable results after tuning the parameters two to three times.

Parameters	Purpose	Value
K	Number of pieces for constructing the branch descriptor (Sec. 4.2.2)	8
λ_B	Clustering of skeletal branches with similar branch descriptors (Sec. 4.2.2)	0.1
C_J	Cost for matching skeletal junction nodes (Sec. 4.2.3)	0.012 - 0.025
d_J	Distance for merging skeletal junction nodes (% of the bounding box diagonal of the mesh, Sec. 4.2.3)	5 - 10

Table 1: *The four parameters of our method.*

4. Augmented Skeleton-Based Shape Correspondence

In this section, we present our method for establishing the shape correspondence between two objects by matching their augmented skeletons K_s (source) and K_t (target). We use the subscript s and t to differentiate the elements of the two skeletons. A correspondence tree is built for matching the skeletal feature nodes and skeletal branches in a breadth-first-search manner. Each node of the correspondence tree stores the matched skeletal feature nodes and skeletal branches while the correspondence tree is built. A schematic diagram of constructing the correspondence tree is shown in Fig. 3.

Initially the correspondence tree has one node, i.e. root, which does not contain any matched pairs. Alg. 1 lists the initialization procedure. We match a skeletal junction node pair with the minimum cost and create a node of the correspondence tree. The newly created node of the correspondence tree becomes the child node of the current inspected node, i.e. the root node. The matching information is stored in the child node. We proceed to match the remaining skeletal branches and skeletal nodes in an interleaving manner. This process is invoked recursively until we cannot match any more. Each leaf node of the final correspondence tree represents a correspondence candidate. The leaf node with the minimum cost is the best skeleton correspondence.

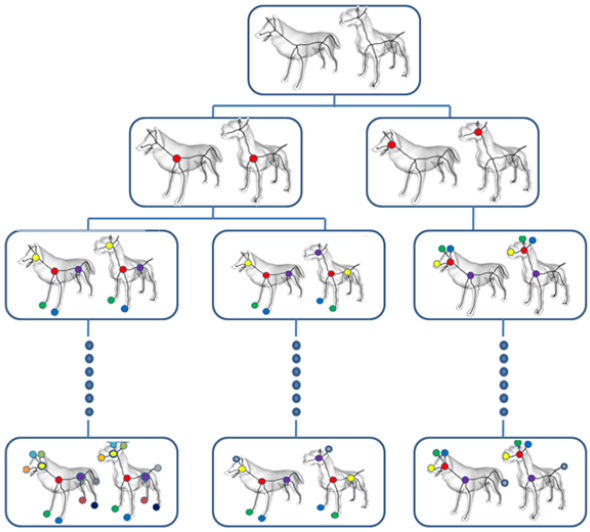

Figure 3: *A correspondence tree for matching two augmented skeletons. Core junction nodes are first matched. The remaining skeletal branches and skeletal junction nodes are matched in an interleaving manner.*

4.1. Core Junction Node Correspondence

The centricity cost $C_{center}(n_s, n_t)$ is defined by:

$$C_{center}(n_s, n_t) = |AGD(n_s) - AGD(n_t)|, \qquad (1)$$

where $AGD(\cdot)$ is the normalized averaged geodesic distance ($\in [0,1]$) of the junction node, which is computed by averaging the AGD values of the vertices associated with the junction node. In other words, $AGD(n_s) = \frac{1}{|\phi(n_s)|}\sum_{v\in\phi(n_s)} AGD(v)$ and $AGD(n_t) = \frac{1}{|\phi(n_t)|}\sum_{v\in\phi(n_t)} AGD(v)$. The centricity cost measures how far a skeletal node is away from the center of the object. The definition of the centricity cost is similar to [ATCO*10]. However, we consider the AGD of the vertices associated with the skeletal node instead of the skeletal node itself. In this way, the geometric properties of the vertices around the skeletal node can be taken into consideration.

The structure feature of a junction node is defined by its sub-skeletons. A sub-skeleton of a junction node starts from a skeletal branch attached at the junction node and contains the portion of the skeleton which can be reached by the skeletal branch without getting through the junction node. Fig. 4(a) shows the sub-skeletons of a junction node n_i. A sub-skeleton can be computed by traversing the skeleton in a breadth-first-search manner. Assume that n_s has N branches $B_s = \{s_1, s_2, ..., s_N\}$ and n_t has M branches $B_t = \{t_1, t_2, ..., t_M\}$. Denote $SG(B_s) = \{g_1, g_2, ..., g_N\}$ as a set of sub-skeletons corresponding to B_s. The normalized length $|g|$ ($\in [0,1]$) of g is computed as the total length of the skeletal branches of g divided by the total length of all the branches of the skeleton. If there are other branches attached at n_s forming the same sub-skeleton, then the sub-skeleton has a loop(s), as shown in Fig. 4(b). The value $|g|$ is then divided by the number of the skeletal branches sharing the same sub-skeleton.

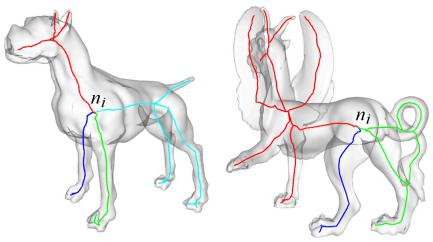

Figure 4: *The sub-skeleton of a junction node n_i:(a) without a loop, (b) with a loop.*

We define $SG(B_t) = \{h_1, h_2, ..., h_M\}$ similarly. The elements of $SG(B_s)$ and $SG(B_t)$ are sorted in decreasing order according to the normalized length of the elements. The structure feature cost $C_{struct}(B_s, B_t)$ is defined as

$$C_{struct}(B_s, B_t) = \sum_{i=1}^{M} (|g_i| - |h_i|)^2 + \sum_{i=M+1}^{N} (|g_i|)^2, N \geq M \quad (2)$$

C_{struct} measures the similarity of the sub-skeletons generated at n_s and $_t$, respectively. The roles of n_s and n_t are switched if $N < M$. The structure cost is used for measuring the similarity of the sub-skeletons generated at the junction node pair (n_s, n_t). The cost $C_{junc}(n_s, n_t)$ for matching (n_s, n_t) is defined as

$$C_{junc}(n_s, n_t) = (C_{center}(n_s, n_t) + \varepsilon) * (C_{struct}(B_s, B_t) + \varepsilon) \quad (3)$$

A small positive constant ε is added so that C_{junc} is not computed as zero. The value ε is set to 0.01 in our experiments.

We match the first junction node pair (n_s, n_t) with the minimum cost $C_{junc}(n_s, n_t)$ and we call this pair the *core junction pair*. A core junction node likely locates at the center part of the object. If there are two or more junction node pairs with similar minimum cost, they are also matched individually and the new nodes of the correspondence tree are created accordingly (see Fig. 3).

4.2. Matching Skeletal Branches

We have matched at least one junction node pair (n_s, n_t) and then we proceed to match the unmatched branches at (n_s, n_t). In this stage, some branches at (n_s, n_t) may be grouped as clusters. If this is the case, the new skeleton graphs K_s' and K_t' are computed and they are also used for the subsequent matching process.

4.2.1. Matching Skeletal Branches Using Shape Matching

The local orientation of skeletal branches at a junction node pair (n_s, n_t) are likely to be similar if the semantics of the parts around n_s and n_t are similar. We therefore exploit the spatial information for performing skeletal branch correspondence. Fig. 5 shows an example for matching the skeletal branches of (n_s, n_t) at the chests of a wolf and a dog. We compute an inscribed ball centered at the junction node and the radius of the inscribed ball is the average distance between the junction node and its associated vertices. The branch vector is computed based on the portion of the skeletal branches inside the inscribed ball. The branch vectors are normalized to unit vectors. In the following, we consider only the branch vectors whose corresponding branches are not matched yet.

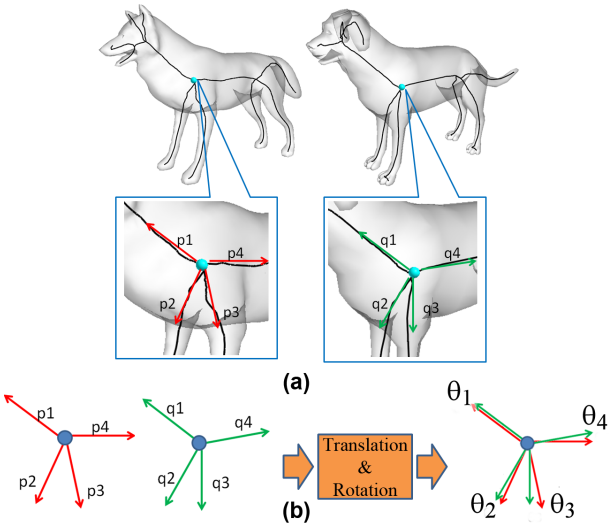

Figure 5: *Skeletal branch correspondence for a junction node pair. The local orientations of the skeletal branches are similar: (a) compute a branch vector for each skeletal branch; (b) compute the rotation transformation R which minimizes the sum of the distances between the branch vector pairs.*

The branch vectors of n_s are transformed to the local

coordinate system at n_t such that n_s locates at n_t (see Fig. 5(b)). Assume that there are N' and M' unmatched skeletal branches at n_s and n_t, respectively, $N' \geq M'$. Hence, there are M' branch pairs to match. In total, there are $M'!C_{M'}^{N'}$ combinations for matching the M' branch pairs.

We adopt a greedy approach to match M' branch pairs. Assume that $P = (p_1, p_2, ..., p_{M'})$ and $Q = (q_1, q_2, ..., q_{M'})$, where p_i and q_i are the unit branch vectors at the matched junction pair (n_s, n_t). We employ shape matching [MHTG05] to compute a rotation matrix R by minimizing the sum of the distances between the branch vector pairs. The angle difference between a branch vector pair (p_i, q_i) is:

$$ang(p_i, q_i) = acos(dot(p_i, Rq_i))/\pi, \tag{4}$$

where $dot(\cdot,\cdot)$ is the dot product of two vectors. The spatial cost for (P, Q) is given by

$$C_{spatial}(P,Q) = \sum_{i=1}^{M'} (ang(p_i, q_i))^2. \tag{5}$$

Now, the cost for matching P and Q is computed as

$$C_{branch}(P,Q) = (C_{struct}(B_s(n_s), B_t(n_t)) + \varepsilon) * (C_{spatial}(P,Q) + \varepsilon). \tag{6}$$

The cost C_{branch} not only depends on the spatial orientation of the skeletal branches but also the structures of the sub-skeletons generated at n_s and n_t. We sort all the possible skeletal branch correspondences with ascending cost and select the first 10% of the skeletal branch correspondences. For each possible branch correspondence (P, Q) at (n_s, n_t), a new node of the correspondence tree is created. After that we have matched the M' skeletal branch pairs. The roles of P and Q are swapped if $M' > N'$.

The rotation matrix R can be used for filtering the bad skeletal branch correspondences which are due to symmetry. If the determinant of the R is negative (i.e. -1), it represents that the rotation matrix involves a reflection. Such a skeletal branch correspondence should be ignored.

We do not adopt MDS [EK03] for computing the branch vectors. This is because the relative orientations of some branches may not be maintained, as shown in Fig. 6.

4.2.2. Clustering of Skeletal Branches

Some parts of a mesh have the same semantic meanings, such as front/back legs and ears. They have similar shapes and geometric features. We can group these similar parts as clusters in order to prune bad correspondences. The idea is as follows. Each part is usually associated with a skeletal branch. We divide each skeletal branch b into K pieces, namely b_1, b_2, ..., and b_K. After that we can compute a value for the geometric property of each piece. In our case, we compute MSP value of each piece (see MSP value in Sec. 5). MSP value captures roughly the local volume around a skeletal node. The advantage of using MSP value over the radius of the maximum sphere at a skeletal node is that using MSP value is more suitable if the objects deform. Consider a deformed circle in Fig. 8. The radius of the circle changes significantly but its perimeter does not change much. We have $MSP(b_k) = \frac{\int_{p \in b_k} MSP(p)dp}{|b_k|}$, where $|b_k|$ is the length of b_k. The value $MSP(b_k)$ is normalized by the maximum MSP value of individual objects so that $MSP(b_k)$ is scale-invariant. There are K values for each branch and they are assembled as a k-dimension vector, namely, branch descriptor. Assume that there are two branch descriptors ℓ_i and ℓ_j. The corresponding two skeletal branches are clustered if $\|\ell_i - \ell_j\|_2 \leq \lambda_B$. The branch clusters are illustrated in Fig. 7. If two branches are clustered, they cannot be clustered with other branches. To handle three or more branches with similar branch descriptors, we create a new node of the correspondence tree for each pair of the branches.

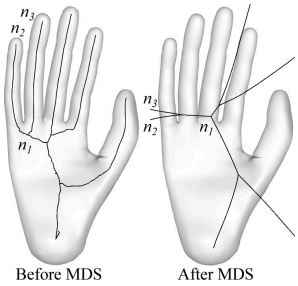

Figure 6: *The two skeletons are obtained before and after applying MDS, respectively. The two branches (n_1, n_2) and (n_1, n_3) are symmetric with respect to the junction node n_1 after MDS is performed.*

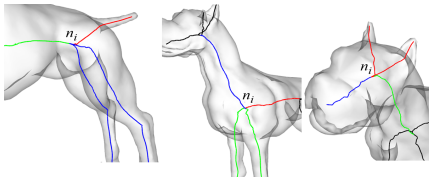

Figure 7: *Clusters of the skeletal branches at a junction node n_i. Each cluster is drawn in the same color. Left: The two back legs; Middle: The two front legs; Right: The two two ears.*

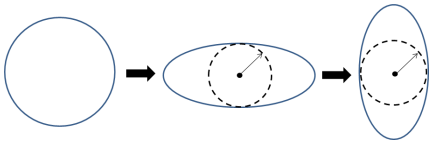

Figure 8: *A deformed circle. Its perimeter does not change much but the radius of the maximum incircle is changed significantly.*

The skeletal branch clusters can be used for eliminating most of the bad correspondences for skeletal branches. Consider the case in Fig. 5 in which the front legs of the wolf and the dog are clustered, respectively. These two clusters are matched and then the skeletal branches associated with the necks and trucks are matched. We do not match the clustered skeletal branches with the non-clustered skeletal branches. The number of branch pairs for checking is reduced from 4!=24 to 2!2! = 4, in this case.

4.2.3. Matching Junction Nodes

To determine whether or not a junction node n'_s should be matched with a junction node n'_t, we compute the cost for matching (n'_s, n'_t) using Equation 3. If the cost is smaller than or equal to a threshold C_J, they are matched. If not, we adopt junction node merging or branch merging to find the other closest junction nodes for matching.

Junction node merging can be employed for merging the adjacent junction nodes if the distance between the junction nodes is less than or equal to d_J. Consider the case for matching n'_s with n'_t, as shown in Fig. 9. But n'_s cannot be matched with n'_t due to the topological difference and $C_{junc}(n'_s, n'_t)$ is too high. After n'_s and n'_t are merged with their adjacent junction nodes, they can be matched. Notice that junction node merging is also applied when the core junction pairs are computed. Branch merging is performed if junction node merging does not work.

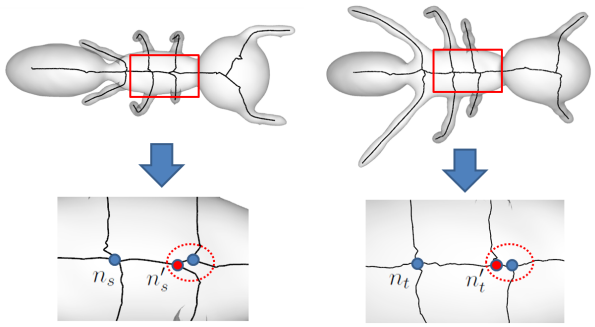

Figure 9: *Matching n'_s with n'_t using junction node merging. The two nodes n'_s and n'_t (colored red) are matched after merging them with the adjacent junction nodes (inside the two red dotted circles).*

Branch merging is adopted to match non-adjacent junction nodes. Consider the case that the junction node pair (n_s, n_t) has been matched. We want to match the next junction node pair from the two sub-skeletons of (n_s, n_t). We check the closest junction node pair (n'_s, n'_t) but find out that they cannot be matched. This is because an unwanted short branch is attached at n'_s and the cost is too high for matching. To solve this problem, we find the other junction node from the sub-skeletons generated by either n'_s or n'_t. The process is repeated until a junction node pair can be matched or no more can be matched. Fig. 10 shows an example in which the two branch paths (n_s, n''_s) and (n_t, n'_t) are checked for matching. In this case, the two source branches (n_s, n'_s) and (n'_s, n''_s) are merged into a new branch (n_s, n''_s). Hence, we successfully match n''_s with n'_t.

If branch merging or junction node merging is performed, the new skeleton graphs K'_s and K'_t are computed and they are used for the subsequent matching process. After we have matched a junction node pair, we proceed to match the remaining unmatched branches.

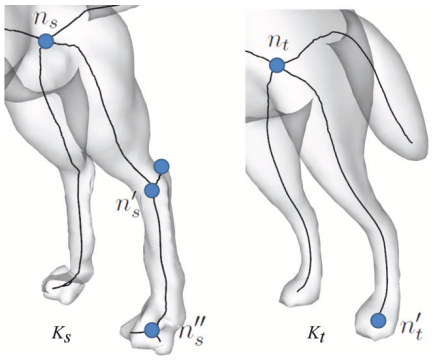

Figure 10: *Matching (n_s, n''_s) with (n_t, n'_t) using branch merging. The cost of matching n'_s with n'_t is too high due to the local topological difference. After performing branch merging, i.e. merging branches (n_s, n'_s) and (n'_s, n''_s) into (n_s, n''_s), n''_s can be matched with n'_t.*

4.3. Final Skeleton Correspondence

Each leaf node of the final correspondence tree represents a candidate correspondence. The maximum costs of C_{junc} and C_{branch} are computed and they are used for normalizing each cost to $(0, 1]$. The sets of costs C_{junc} and C_{branch} of a leaf node are denoted as $\{c^j_1, c^j_2, ..., c^j_{N_j}\}$ and $\{c^b_1, c^b_2, ..., c^b_{N_b}\}$. Then the cost of a leaf node is given by:

$$C_{leaf} = \Pi_{i=1}^{N_j} c^j_i * \Pi_{i=1}^{N_b} c^b_i \quad (7)$$

Usually, the cost of a leaf node is smaller if there are more matched skeletal node and branch pairs. The leaf node with the minimum cost is selected as the best correspondence.

5. Implementation

We adopt the minimum slice perimeter (MSP) [HJWC09] for computing our skeletons in the preprocessing stage. MSP of a vertex captures the local volume of the vertex. The method by [ATC*08] can also be adopted. Once the skeletons are constructed, the mapping ϕ between the vertices and the skeletal nodes is obtained intrinsically. We briefly review the MSP function and then the MSP skeleton as follows.

The MSP function is a scalar function defined on the surface of a mesh. Slices of a point p are defined as the intersection curves with the set of planes generated by the normal of the surface in p. The MSP value of p is the minimum perimeter of all slices of p. The MSP value of a point is usually approximated by uniformly sampling as it is not easy to generate all the slices of the point. In our experiments, we used 12 slices for each point. An example is shown in Fig. 11 for computing the MSP value of a point. The MSP value of a point can be used for approximating the volume around a slice by supplying a thickness value.

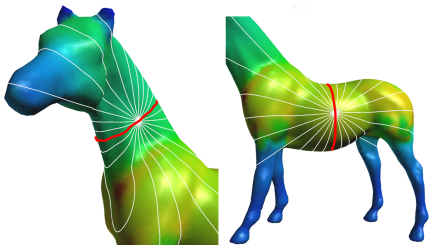

Figure 11: *The MSP value of a point p (red thick line). There are 12 slices for each model.*

An MSP skeleton is built as follows. The slice of each vertex which has the minimum perimeter is computed. Each vertex is then transformed to the center point of the intersected curve between the minimum slice and the object. The transformed vertices likely locate at the interior of the original mesh and they form a new mesh which is skinny. We call this skinny mesh a shrunk mesh. A greedy skeletonization process is then employed for degenerating the shrunk mesh by performing a sequence of edge swap operators in order to obtain a 1D curved skeleton. The greedy skeletonization process is similar to [ATC*08]. During the process, the transformed vertices are merged to obtain skeletal nodes. Hence, each skeletal node of the curved skeleton is mapped to a set of the vertices. Some of the vertices may not be associated with any skeletal nodes because unwanted short skeletal branches and their corresponding vertices are removed.

6. Results and experiments

We present the results of our method for computing shape correspondence and then discuss its limitations.

Results. In the preprocessing stage, we computed the skeletons of the models and AGD of the vertices. We then applied our method for a variety of models. The dog skeleton is the source. The skeleton correspondence results are shown in Fig. 12. The matched node pairs are drawn in the same colors but the unmatched nodes are not shown. All of the semantic parts (e.g. legs, heads, and ears) are matched correctly. Since our method performs junction node merging and branch merging, there may be one-to-many or many-to-many correspondences between junction nodes and also between branches. This is different from the previous approaches [HSKK01, BMSF06, ATCO*10] which establish only one-to-one mapping. For example, a junction node at the chest of the dog is matched with two junction nodes at the chest of the goat. Our method can handle the skeletons of chairs with loops, as shown in Fig. 13.

Comparisons. We compared our method with the method by Au et al. [ATCO*10]. Fig. 14 shows two sets of correspondence results. For the top example, their method mismatches an ear of the horse to the jaw of the dog. For the bottom example, their method mismatches the ears of the pig to the upper jaw of the dragon. Our method produces correct results due to branch clustering. The ears form clusters and they cannot be matched with the non-clustered jaws.

Compared to [BL08], our method can match the internal nodes of the skeletons. Our method matches the core junction nodes, and then match the remaining branches and junction nodes in an interleaving manner. So that our method can match the internal nodes.

Model	#Tri.	AGD Function	MSP Function	SE	MAS
Dog(source)	18976	157.71	25.86	33.23	-
Deer	7402	22.23	5.04	3.57	0.20
Human	11258	43.05	5.10	9.74	0.22
Armadillo	20000	193.32	30.10	23.63	0.24
Dragon	16000	199.02	27.79	28.16	0.19
Triceratops	15764	104.33	18.17	27.44	0.21
Elephant	30000	500.30	52.73	146.08	0.30
Asian dragon	28198	490.70	44.54	37.28	0.41

Table 2: *Model complexities, the timings (sec) of the preprocessing tasks and the timings of matching the augmented skeletons. SE: Skeleton extraction; MAS: Matching the augmented skeletons.*

Timing information. All the results were performed on Intel Core 2 Duo CPU E8400 3.0GHz with 4GB memory, using a single thread implementation. We precomputed AGD function, MSP function, and skeleton extraction. Table 2 shows the timing information of the preprocessing computation and the timings of performing the shape correspondence (i.e. matching the augmented skeletons of the objects). The computation time for shape correspondence is under 0.5 second in all our experiments. Table3 shows the information of the correspondence trees which were constructed for matching the augmented skeletons of the objects.

The searching space of our method is smaller than the combinatorial searching method [ZSCO*08, ATCO*10] as we perform branch clustering. The total number of the possible matchings between the two skeletons reduce dramatically. Fig. 15 shows three leaf nodes of the correspondence tree for matching the skeletons of the dog and giraffe models. The best result is shown in Fig. 15(a). The cost of the best node is smaller if more feature nodes and skeletal branches are matched. Similar to other existing techniques, our method requires the manual operations to change the parameters if the results are not good. It usually takes a few

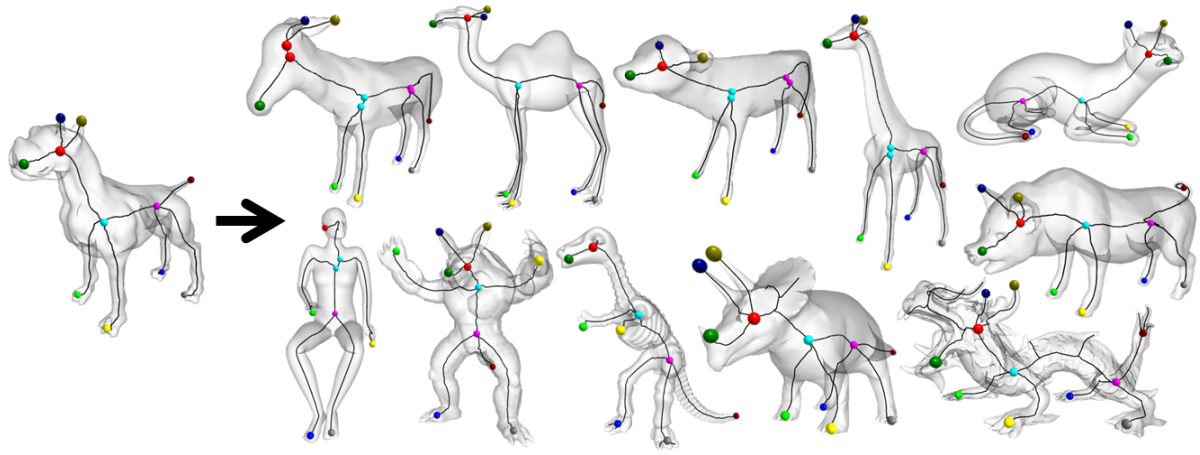

Figure 12: *Shape correspondence results between a dog and a variety of models: goat, camel, deer, giraffe, cat, pig, human, armadillo, dinosaur, triceratops and Asian dragon. The matched node pairs are drawn in the same colors.*

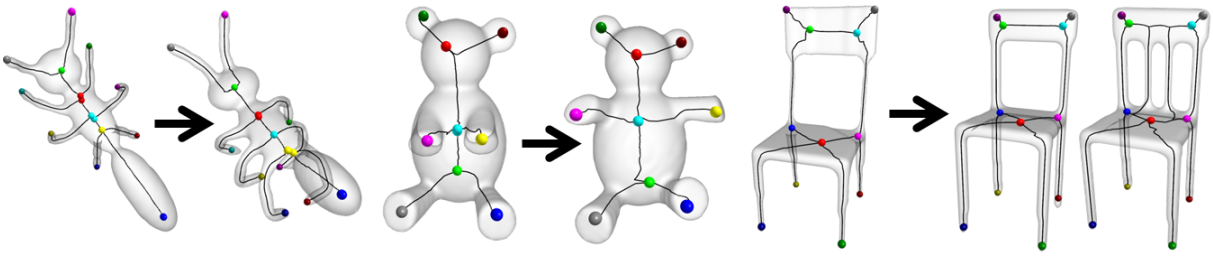

Figure 13: *Shape correspondence results for ant, teddy and chair models.*

minutes for making several attempts to obtain the desired results.

Model	#Tree Nodes	Tree Height	#Core Junction Nodes	#Branch and Node Merging	#Branch Clusters	#Leaf Nodes
Dog(source)	-	-	-	-	-	
Goat	18	4	3	12	20	4
Deer	18	4	3	15	26	5
Giraffe	23	4	3	8	36	7
Cat	18	4	3	8	22	5
Pig	11	4	3	0	16	3
Human	12	3	2	4	16	4
Armadillo	18	4	3	0	24	5
Dragon	14	4	3	0	18	4
Ant	42	5	4	50	24	10
Teddy	13	4	3	0	24	4
Chair	6	6	5	0	2	1

Table 3: *Information of the correspondence trees.*

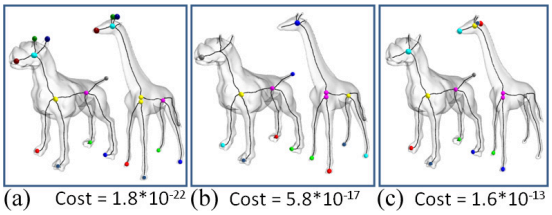

Figure 15: *Some leaf nodes of the correspondence tree of matching the skeletons of the dog and giraffe models. (a): Best result; (b): the chest of the dog is matched with the belly of the giraffe; (c): the head of the dog is matched with the mouth of the giraffe.*

Discussions and limitations. Our method may mismatch the branch clusters (e.g. mapping the ears of the horse to the horns of the cows) because of a lack of the semantic knowledge. In addition, some objects having flat shape around the junction nodes may not be inferred the front or back side by skeletons. A flipping correspondence may be obtained.

Our method requires manual operations to set the parameters and it may take several attempts to obtain acceptable results. Finally, we cannot construct good skeletons for some man-made models (e.g., cups, sphere ball, or engineering models) and the quality of shape correspondence results is affected. If the skeletons do not capture the shape of the meshes faithfully, our method may not work properly.

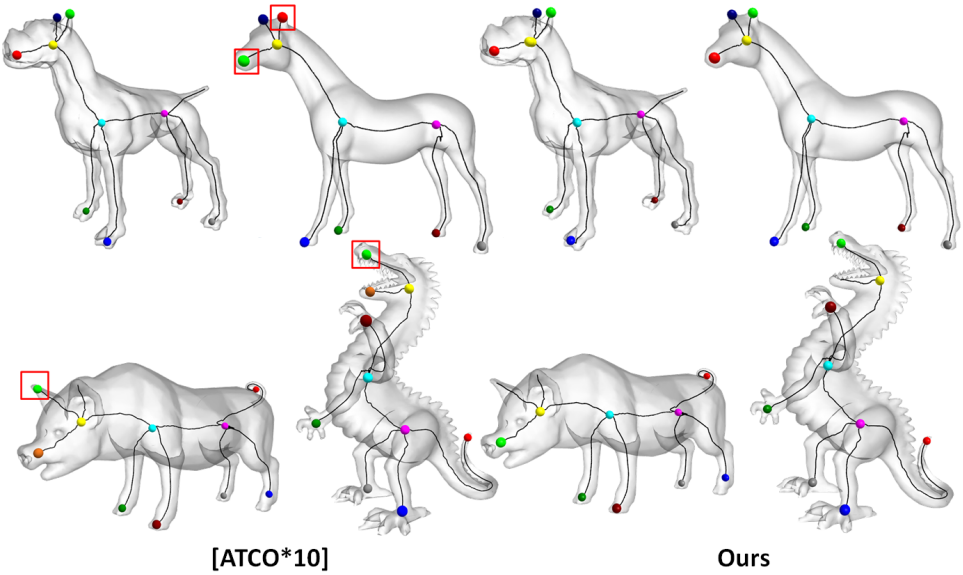

Figure 14: *Comparison with the method [ATCO*10]. Left: the results of [ATCO*10]; Right: our results. Their method mismatches the nodes at the heads (highlighted by the red squares).*

7. Conclusions

We have presented a novel skeleton-based approach for computing shape correspondence. Our method may merge skeletal nodes and branches so that it may compute 1-1 correspondences and many-many correspondences. Our method has been tested for a variety of similar objects and they may have different poses. The experimental results show that our method can compute acceptable shape correspondence.

Our method relies on the association between the skeletal nodes and object vertices, but we do not consider the actual shape of the parts for matching. Our method may fail to compute a reasonable shape correspondence if the objects are not similar. One possible extension of our work is to consider the actual shape of the parts so that partial correspondence can be computed. Another possible extension is that instead of selecting the core junction nodes, the parts of the objects are first extracted by employing skeletons. After that we can match the parts and build the entire shape correspondence. Furthermore, we would like to apply prior knowledge with our method for computing shape correspondence.

References

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. *ACM TOG 27*, 3 (2008).

[ATCO*10] AU O.-C., TAI C., COHEN-OR D., ZHENG Y., FU. H.: Electors voting for fast automatic shape correspondence. *Comp. Graph. Forum 29*, 2 (2010), 645–654.

[BL08] BAI X., LATECKI L.: Path similarity skeleton graph matching. *IEEE Trans. on PAMI 30* (2008), 1282–1292.

[BMSF06] BIASOTTI S., MARINI S., SPAGNUOLO M., FALCIDIENO B.: Sub-part correspondence by structural descriptors of 3D shapes. *Comp. Aided Design 38*, 9 (2006), 1002–1019.

[EK03] ELAD A., KIMMEL R.: On bending invariant signatures for surfaces. *IEEE Trans. on PAMI 25* (2003), 1285–1295.

[HJWC09] HO T. C., JI H. X., WONG S. K., CHUANG J. H.: Mesh skeletonization using minimum slice perimeter function. *Computer Science Technical Report CS-TR-2010-0001, The National Chiao Tung University, Taiwan, R.O.C.* (2009).

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH* (2001), pp. 203–212.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM TOG 24*, 3 (2005), 471–478.

[MS01] MORTARA M., SPAGNUOLO M.: Similarity measures for blending polygonal shapes. *Computers and Graphics 25*, 1 (2001), 13–27.

[SSGD03] SUNDAR H., SILVER D., GAGVANI N., DICKINSON S.: Skeleton based shape matching and retrieval. In *Shape Modeling Int'l* (2003), p. 130.

[TS04] TUNG T., SCHMITT F.: Augmented reeb graphs for content-based retrieval of 3D mesh models. In *Shape Modeling Int'l* (2004), pp. 157–166.

[vKZHCO10] VAN KAICK O., ZHANG H., HAMARNEH G., COHEN-OR D.: A survey on shape correspondence. *Eurographics State-of-the-art Report* (2010).

[ZSCO*08] ZHANG H., SHEFFER A., COHEN-OR D., ZHOU Q., VAN KAICK O., TAGLIASACCHI A.: Deformation-driven shape correspondence. *Symposium on Geometry Processing* (2008), 1431–1439.

A Skeleton-Based Approach for Consistent Segmentation Transfer

SAI-KEUNG WONG, JAU-AN YANG, TAN-CHI HO, JUNG-HONG CHUANG
Department of Computer Science
National Chiao Tung University
Hsin-chu, 300 Taiwan

We present a skeleton-based approach for mapping consistently a segmentation of a source mesh to a target mesh. The source and target meshes are semantically similar but their geometries may be different. We assume that the segmentation of the source mesh and the skeleton correspondence between the meshes are provided. Our method first establishes the skeleton-boundary correspondence and then performs the boundary transfer procedure. The skeleton-boundary correspondence is useful for computing approximately the target boundaries. The boundary transfer procedure aims to produce the segment boundaries of the target mesh such that their geometric details are consistent to the ones of the corresponding source segment boundaries. We have applied our method for various models and compared with the state-of-the-art techniques. Our method computes segment boundaries of the target mesh which are similar to the ones of the source mesh.

Keywords: segmentation transfer ·skeleton correspondence ·consistent segmentation·geometric algorithms · skeleton-boundary correspondence

1. INTRODUCTION

Mesh segmentation refers to partitioning a mesh into meaningful disjoint sub-patches or parts. It is a fundamental technique in computer graphics. Much effort has been devoted for segmenting individual objects into parts. Segmenting a set of similar objects consistently is required in applications, such as object retrieval and object classification [10].

Skeleton is an important object descriptor which encapsulates the structure of an object [4]. The skeleton correspondence between two objects establishes the matching for their similar semantic parts. Similar objects are likely to be segmented similarly for the matched parts as the similar semantic parts of the objects are likely to be segmented consistently. In this paper, we develop a skeleton-based approach for mesh segmentation transfer which can be applied to compute consistent segmentation for a set of similar objects. Given the segmentation of the source mesh and the skeleton correspondence between the source and target meshes, our method transfers the segment boundaries of the source mesh to the target mesh. Our method requires the skeleton-mesh correspondence which is useful for computing consistent segmentation.

In skeleton-mesh correspondence, each skeletal node corresponds to a set of mesh vertices and a skeletal arc consists of a set of skeletal nodes. Consider that we have a segmented mesh and its skeleton. Notice that a segment boundary consists of a set of vertices. Hence, the mapping between the skeletal nodes and the mesh vertices can be used for establishing the mapping between the skeletal nodes and the segment boundaries. Furthermore, the source segment boundaries can then be transferred approximately to the target mesh according to the skeleton correspondence. Finally, we refine the segment boundaries of the target mesh so that they are similar to the corresponding source segment boundaries. We have evaluated our method for various models and compared with the state-of-the-art techniques. Experimental results show that our method computes acceptable segmentation of the target meshes.

2. RELATED WORK

We review mesh segmentation techniques which are mostly related to our method. Mesh segmentation has been researched extensively [27,16,1,20,15,19,26,18,28,10,31]. Some mesh segmentation algorithms

partition a given mesh into patches which are satisfied with certain properties, such as volume, planarity and disc-like. On the other hand, there are algorithms which partition a mesh into semantic components based on the human perception. Readers are referred to [2,24,8] for details. Different segmentation algorithms may generate different segmentations for a mesh. Chen et al. [8] proposed a number of criteria to evaluate the similarity of two segmentations. Based on the criteria, they compared different segmentations to the ground truth segmentations defined by users. Consistent mesh segmentation aims to produce consistent segmentations for a set of meshes [28,25,31].

Golovinskiy and Funkhouser [10] employed rigid alignment [5] in a hierarchical clustering approach for consistent segmentation. Both the geometric features of individual meshes and the correspondence information between the set of meshes are considered. However, rigid alignment may not be able to correctly align the meshes in some cases, as reported in [14].

Kalogerakis et al. [14] proposed a scheme to compute segmentations and to assign labels for a set of meshes. The assignment of labels was formulated as an optimization problem and the objective function measured the consistency of primitives (i.e. triangles) with labels. The objective function was computed via a training process. Then the objective function was applied to the other meshes for computing consistent segmentations. Their approach handled various segmentations for a wide range of meshes. They reported that it took eight hours to train on a single Xeon E5355 2.66GHz processor for a database consisting of six training meshes of about 20K-30K faces and six labels. Our method computes consistent segmentations for similar objects but does not have a training process. Furthermore, our method refines the target boundaries so that their geometric details are as similar as possible to the ones of the corresponding source boundaries.

Parameterization methods [22,17,23] can be adopted for establishing a mapping between two objects. The segmentation of one object can then be transferred to another. This kind of approach may not produce similar boundaries of segments between the two objects as such techniques often rely on globally minimizing certain kind of energies. The segment boundaries may not locate at the desired positions.

3. PRELIMINARIES AND OVERVIEW

We present the preliminaries and overview of our method in this section. The inputs of our method are: (1) two meshes (source and target), (2) their skeletons, (3) skeleton mesh correspondence and (4) the segmentation of the source mesh. The outputs are: (1) the segmentation of the target mesh and (2) the boundary correspondence between the source and target segmentation. We not only build the correspondence between the segment boundaries but also the points of the segment boundaries. The encoding of the source segmentation includes: each segment boundary consists of a set of connected edges; and each face of the source mesh is assigned a segment index.

3.1 The Skeleton

We follow the similar notation presented in [34]. A skeleton has a set of skeletal nodes and skeletal branches. Two directly connected skeletal nodes form a skeletal branch. The shortest path between two skeletal nodes is called a skeletal path. There are two types of skeletal feature nodes: junction nodes and terminal nodes. A junction has at least three incident branches and a terminal node has one incident branch. A skeletal arc is a skeletal path and it has two feature nodes which are also its two end nodes. Two skeletons are shown in the top row of Fig. 1.

3.2 The Skeleton-Mesh Correspondence

Our method exploits the skeleton-mesh correspondence to perform boundary transfer from the source segmentation to the target. The skeleton-mesh correspondence can be obtained by applying the method presented in [4] to construct the skeleton of the mesh. The method shrinks the mesh and the vertices are grouped as skeletal nodes during the skeletonization process. After computing the skeleton, a skeletal node is associated with a subset of the mesh vertices. A non-feature node is mapped to a set of vertices forming a

ring around the node. Furthermore, a feature node is mapped to a set of vertices around the feature node and they scatter over a sphere-like pattern. The bottom row of Fig. 1 shows an example.

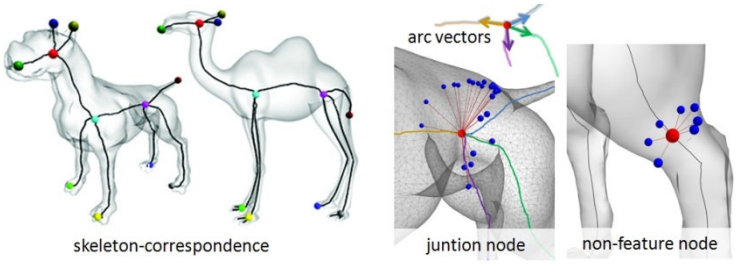

Fig. 1. Top row: two skeletons and their skeleton correspondence (color coded). Bottom row: skeleton-mesh correspondence. The arc vectors of the four skeletal arcs at a junction node are illustrated.

3.3 The Skeleton Correspondence

A variety of methods have been proposed for computing the skeleton correspondence between two objects [12,30,29,6,35]. Readers are referred to [32] for a comprehensive survey. In this paper, our method focuses on objects with similar geometries. Hence, we employ [3] to do so. The method in [3] computes a set of eligible skeletal nodes and then they are used to vote for the matchings between skeletal nodes. The matching with the highest vote count is selected as the best skeleton correspondence between the two objects. Hence, we also know the matched skeletal arc pairs if the end nodes of the skeletal arcs are matched pairwise. An example is shown in the top row of Fig. 1.

3.4 MSP function and MSP-gradient function

(a) (b)

Fig. 2. MSP function (a) and MSP gradient functions (b). The surface is colored from blue, green to red with increasing value.

The minimum slice perimeter (MSP) function is employed for locating the segment boundaries along the corresponding skeletal arcs. We apply the method in [13] to compute the MSP function which is a scalar function defined on the surface of a mesh. Slices of a point q are defined as the closed intersection curves between the mesh and the set of planes generated by the normal to the surface in q. The MSP value of q, denoted as $M(q)$, is the minimum perimeter of all slices of q. The $M(q)$ value is approximated by uniformly sampling. In our experiments, we used 12 slices for each point as they are adequate for our need (see an example in Fig.2). The MSP gradient function represents the change rate of the local volume in the neighboring region. The MSP gradient of a point q is given by:

$$\nabla M(q) = \frac{1}{|B|}\left(\sum_{edge\ e}\left(M(q_i^e) - M(q_j^e)\right)(e \cap B)\mathbf{e}\right),$$

where $|B|$ is the surface area of the neighboring region B of q, $e \cap B$ is the length of the part of an edge e inside B, q_i^e and q_j^e are the two vertices of e, and $\mathbf{e}$ is the unit direction along e. The region B is set as the 1-ring triangles at q. Fig. 2 shows the results of the MSP gradient for different meshes. The magnitude of the MSP gradient increases at the places (e.g. junctions) where MSP values change significantly. Notice that the parts with different volumes can be distinguished.

3.5 Overview

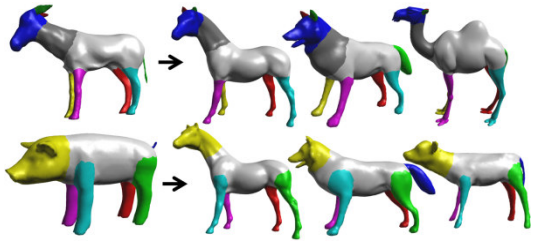

Fig. 3 The segmentation transfer results (top and bottom) for the four-legged animals. The characteristics of the source segmentation (goat and pig) are captured and reproduced to the target models.

There are two major stages in our method: skeleton-boundary correspondence and boundary transfer. The skeleton boundary correspondence is useful for computing approximately the target boundaries and it is built between segment boundaries and skeleton arcs of the source mesh. A boundary transfer procedure is adopted for mapping each source segment boundary enclosing a skeletal arc to the target mesh. In the procedure, a cylindrical parameterization is applied for mapping the regions around the source segment boundaries to the target mesh. Finally, we refine the segment boundaries on the target mesh such that their geometric details are consistent to the ones of the corresponding source segment boundaries. Fig. 3 shows some of our results. The interior (i.e. faces) of the segment regions can be computed after the segment boundaries are computed. Table 1 shows the major parameters of our method.

Table 1. The major parameters of our method.

Parameter	Purpose
θ	Determining local maximum of the skeletal-arc profile (for computing MSP-gradient-based boundaries)
K	Gaussian smoothing for denoising the skeletal-arc profile (Kernel Size)
λ	Searching region for target boundary refinement

4. CONSISTENT SEGMENTATION TRANSFER

Given the skeleton correspondence between the source and target skeletons, we transfer each source segment boundary enclosing a skeletal arc to the target mesh. Assume that the source arc $b^s(n_0^s, n_1^s)$ and the target arc $b^t(n_0^t, n_1^t)$ are matched, where n_0^s and n_1^s are the two end nodes of b^s, and n_0^t and n_1^t are the two end nodes of b^t. Besides, b^s and b^t are enclosed by the source segment boundary B^s and the target segment boundary B^t (to be computed), respectively. We perform four steps to transfer B^s to the target mesh so as to obtain B^t: skeleton-boundary correspondence, construction of covering regions, boundary transfer, and target segment boundary refinement.

4.1 Skeleton-Boundary Correspondence

We compute the fitting plane of B^s based on Principal Component Analysis [1]. If the fitting plane of B^s intersects with a skeletal arc b^s then B^s encloses b^s. Assume that $p(B^s)$ is the intersection point between b^s and the fitting plane. An arc ratio $r(B^s)$ of B^s with respect to b^s is given as: $r(B^s) = len(n_0^s, p(B^s))/len(n_0^s, n_1^s)$, where $len(\cdot, \cdot)$ is the distance between the two points on the skeletal arc b^s. Our goal is to compute $r(B^t)$ of the target segment boundary on bt and then compute $p(B^t)$ based on $r(B^t)$. A skeletal arc corresponds to a part of the object. Hence, the parts of the matched skeletal arcs are similar. Consider that the part associated with bs is divided into two sub-parts based on $r(B^s)$. Similarly, we can divide the part associated with b^t based on $r(B^t)$. Hence, we want to compute $r(B^t)$ such that the two sub-parts

associated with b^s are similar to the two sub-parts associated with b^t. To do so, we perform a skeletal-arc profile analysis on the skeletal arcs b^s and b^t to compute similar ghost boundaries. Then a one-to-one mapping is established between the matched ghost boundaries. Finally, $r(B^t)$ can be computed based on proportion.

In our case, the ghost boundaries are MSP-gradient based boundaries. We compute ghost boundaries based on MSP-gradient magnitude of the surface as boundaries may appear in the regions with the maximum of MSP-gradient magnitude. The MSP gradient magnitude of a skeletal node can be computed as the average MSP gradient magnitude of the associated mesh vertices, i.e. $\frac{1}{|S(n)|}\sum_{q \in S(n)}|M(q)|$, where $S(n)$ is the set of vertices associated with a skeletal node n and $|S(n)|$ is the cardinality of $S(n)$. An example is shown in Fig. 4. The MSP gradient magnitude on a skeletal arc is smoothened by using Gaussian smoothing [11] so as to eliminate unwanted details. A fuzzy region is built at each local maximum value using the vertices of a skeletal node [16]. The dual graph of the fuzzy region is then constructed and the MSP gradient-based boundary is extracted by graph cut [7].

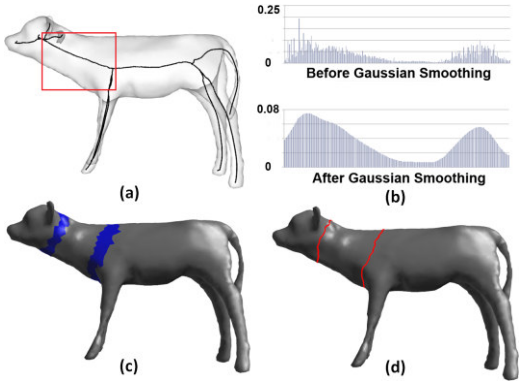

Fig. 4 The skeletal-arc profile analysis: (a) deer's neck; (b) the distribution of the MSP gradient magnitude before and after Gaussian smoothing; (c) a fuzzy region around each local maximum; and (d) the MSP-gradient-based boundaries

Let a_i be an arc in the dual graph and e_i be the common edge of the two dual faces connected by ai. The capacity of a_i is given by:

$$Cap(a_i) = \ell(e_i)\left(\frac{1}{|\nabla M(e_i)| + \varepsilon}\right),$$

where $\nabla M(e_i)$ is the MSP gradient of the edge e_i (the average MSP gradient of its two vertices), $\ell(e_i)$ is the length of e_i and ε is a small constant which is set to 0.001 in our experiments. The MSP-gradient-based boundary passes through the edges which have large magnitude of MSP gradient. Fig 5 shows the MSP gradient-based boundaries and the given segment boundaries of two meshes.

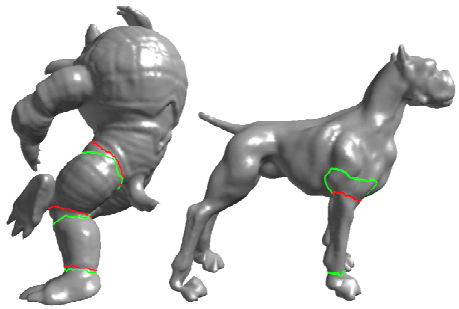

Fig. 5 The ghost boundaries (red) and the given segment boundaries (green) on two models.

Denote B_i^s as the i-th MSP-gradient-based boundary of the source and B_j^t as the j-th MSP-gradient based boundary of the target. The cost for matching the MSP-gradient-based boundary pair is given by:

$$C_b\left(B_i^s, B_j^t\right) = \left(r\left(B_i^s\right) - r\left(B_j^t\right)\right)^2 + \left(\frac{\Sigma_{q \in B_i^s} |\nabla M(q)|}{|B_i^s| \Delta_{MSP}} - \frac{\Sigma_{q \in B_j^t} |\nabla M(q)|}{|B_j^t| \Delta_{MSP}}\right)^2,$$

where ∇M(q) is the MSP gradient of a mesh vertex q, $|B_i^s|$ and $|B_j^t|$ are the numbers of the vertices of B_i^s and $|B_j^t|$, respectively; Δ_{MSP} is the maximum of the MSP gradient magnitude of the vertices. We find the MSP-gradient-based boundary pair with the lowest cost. Assume that $r\left(B_g^s\right)$ and $r\left(B_g^t\right)$ are the arc ratios of the best matched source and target MSP gradient-based boundaries, respectively. There are two cases to consider:

1. Case One: $p(B^s)$ lies between n_0^s and $p\left(B_g^s\right)$, then $r(B^t) = \frac{r(B^s) r(B_g^t)}{B_g^s}$;

2. Case Two: $p(B^s)$ lies between $p\left(B_g^s\right)$ and n_1^s, then $r(B^t) = r\left(B_g^t\right) + \frac{\left(r(B^s) - r\left(B_g^s\right)\right)\left(1 - r\left(B_g^t\right)\right)}{1 - B_g^s}$.

After $r(B^t)$ is computed, $p(B^t)$ can be computed based on proportion, i.e. the linear interpolation on the skeletal arc b^t, $\frac{\mathrm{len}(n_0^t, p(B^t))}{\mathrm{len}(n_0^t, n_1^t)} = r(B^t)$. Notice that in both cases, if $r(B^s) = r(B_g^s)$, we have $r(B^t) = r(B_g^t)$. In other words, if B^s lies around B_g^s, B^t lies around B_g^t, too.

If the MSP-gradient-based boundaries are not found, $r(B^t)$ is set as r(B^s). Then we compute p(B^t) based on proportion.

4.2 **Covering regions**

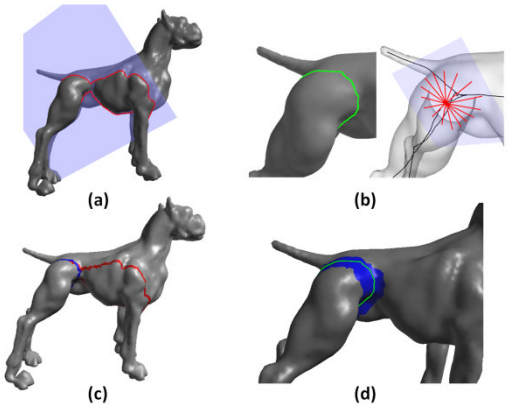

Fig. 6 Construct a covering region of a boundary: (a) construct a fitting plane; (b) project the points of the boundary to the fitting plane and construct the line segments; (c) remove the faces not overlapping with the boundary (red) ; and (d) refine faces (blue) around the boundary to obtain the final covering region.

A covering region of a boundary consists of a set of faces and the faces cover the entire boundary. The covering region is used for computing and refining the target segment boundaries. We rely on two characteristics to construct the covering region of a source segment boundary: (1) the relative orientation between the fitting plane of the source boundary and its associated arc; and (2) the distances between the points of the source segment boundary and its associated fitting plane. To construct a covering region of Bs, we consider the faces of the mesh intersecting with the corresponding fitting plane. If all the faces intersected by the fitting plane overlap with Bs, then we simply use these faces as the initial covering region. However, the fitting plane may intersect with some faces of the mesh and these faces may not overlap with B^s, as shown in Fig. 6(a). Assume that a set F_p contains the faces intersecting with the fitting

plane and overlapping with B^s. We project sample points of B^s to the fitting plane and construct a line segment at $p(B^s)$ for each projected sample point (Fig. 6(b)). In our experiments, we used 16 sample points. These line segments approximate the local shape of B^s. We compute a set F_c which contains the faces closest to the end points of the line segments. But the faces of F_c may not be connected to each other. The Dijkstra's algorithm is performed to collect the neighbouring faces to F_c and these neighbouring faces form a new set F_d. In this way, the faces of $F_c \cup F_d$ are connected. We use the faces of $F_c \cup F_d$ as the initial covering region, as shown in Fig. 6(c). The region growing method is then performed for the faces of $F_c \cup F_d$ until all the one-ring vertices of B^s are covered. An example of the final covering region is shown in Fig. 6(d).

An arc vector of a skeletal arc generated at a skeletal junction indicates the major direction of the skeletal arc. To compute an arc vector of a skeletal arc at a skeletal junction, we compute an inscribed ball centered at the skeletal junction and the radius of the inscribed ball is the average distance between the skeletal junction and its associated mesh vertices. The arc vector is computed based on the portion of the skeletal arc inside the inscribed ball. We perform five steps to construct the covering region of B^t, as illustrated in Fig. 7. First, the junction node n_0^s of b^s is picked as the origin of the local coordinate system. The x-axis is the arc vector of b^s. We pick a unit vector $\tilde{\mathbf{y}}$ which points to the direction of another skeletal arc connected at n_0^s. Second, the z-axis and the y-axis are computed as $\mathbf{z} = \mathbf{x} \times \tilde{\mathbf{y}}$ and $\mathbf{y} = \mathbf{z} \times \mathbf{x}$. Third, the source coordinate system is transferred to the target by using shape matching [21] such that the coordinate axes match with the arc vectors at the target joint. Fourth, the orientation of the target coordinate system is adjusted by aligning the x-axis to the arc vector of b^t. Finally, the local coordinate systems are translated to $p(B^s)$ and $p(B^t)$, respectively. Based on the coordinate mapping, the fitting plane of Bs is then transferred to the target so as to obtain the fitting plane of B^t.

We need to take into account the shape of the corresponding parts when the line segments are transferred from the source boundary to the target boundary. To do so, the line segments are scaled based on the MSP ratio which is the ratio of the MSP value of b^t to the one of b^s. They are clamped if they penetrate the mesh. Then we construct the covering region of B^t similarly as we have constructed the covering region of B^s.

4.3 Boundary transfer

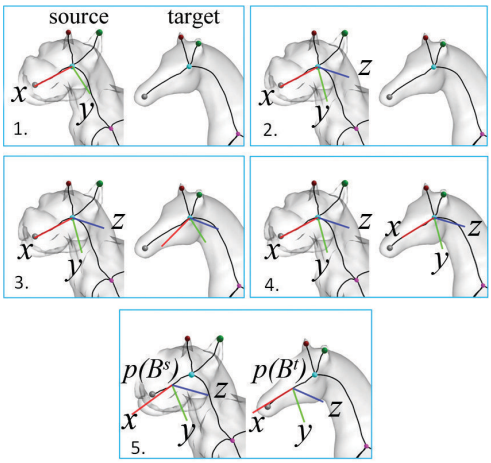

Fig. 7 Establishing the local coordinate systems on the source and target skeletal arcs. The local coordinate system at the source junction node is created and then it is transferred to the target junction node by using shape matching such that the coordinate axes are matched with the arc vectors at the target junction node. Finally, translate the origins of the local coordinate systems to $p(B^s)$ and $p(B^t)$, respectively.

At this stage, we describe the method to build B^t. Assume that the covering regions of B^s and B^t are C^s and C^t, respectively. We employ cylindrical parameterization ([33]) to parameterize C^s and C^t. A

covering region has two boundaries and they bound the internal region of the covering region. These two boundaries correspond to the upper and lower boundaries of a cylinder according to their arc ratios (e.g. upper boundary with smaller arc ratio). The internal region of the covering region corresponds to the lateral side of the cylinder. The cylindrical parameterization can be unfolded to a 2D plane domain. The v coordinates of the lower and upper boundaries are set to 0.0 and 1.0, respectively. The range of the parameterized coordinates (u, v) is [0, 1) × [0, 1].

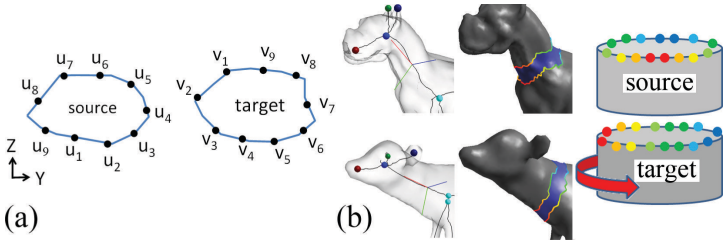

Fig. 8 The parameterized orientation adjustment. (a): The *y*-coordinate value is chosen as the alignment value as its range is larger than the range of the *z*- coordinate values of the samples. In this case, μ_o is around 6/9 ; (b): The vertices of the boundaries are colored according to their alignment values. The *u*-offsets are computed for minimizing the total difference between the coordinates of the boundary sample points.

The origin (i.e. (0, 0)) of the parameterization map is set as a point of the lower boundary of the covering region. The parameterized covering regions of the source and target may have a large deviation in the U-direction. A parameterized orientation adjustment is performed to address this issue. The parameterized orientation adjustment can be treated as rotating the boundaries of the target cylinder so as to align with the boundaries of the source cylinder. After the adjustment, the vertices of the boundaries have the similar distribution of the coordinates, as shown in Fig. 8. The details are given as follows. All the boundary vertices of C^s and C^t are transformed to the local coordinate systems, respectively.

Notice that the cylinder axis is pointing in the same direction with the *x*-axis of the coordinate system. We compute the two ranges for the *y*– and *z*– coordinates of the boundary vertices of C^s and C^t. The coordinate with the largest range is chosen as the alignment value of all the boundary vertices. We sample N points uniformly from the lower boundary L^s of C^s and their parameterized *u*-coordinates form a *N*-tuple $U_B{}^s = (u_1, u_2, ..., u_N)$. Similarly, we have $U_B{}^t = (v_1, v_2, ..., v_N)$. Denote that V (L, u) returns the alignment value at the sample position whose *u*-coordinate is *u* on the boundary *L*. The u-offset $\mu_o \in [0, 1)$ of L^t is computed by minimizing the following function:

$$argmin_{\mu_o} \sum_{i=1}^{N} \left(V(L^s, u_i) - V(L^t, v_i + \mu_o)\right)^2,$$

where $v_i + \mu_o$ is wrapped to [0, 1] if $v_i + \mu_o > 1$ (e.g. 1.2 is wrapped to 0.2). If the sample position is on an edge, the alignment value is computed by interpolating the values of the two vertices of the edge. After the process, we match the sample points of B^s to the sample points of B^t. Notice that the vertices of B^t may lay on the edges of the target mesh.

4.4 Boundary Refinement

We have matched the sample points of the source and target boundaries. The final step here is to refine the target boundaries. Most of the mesh segmentations compute boundaries close to the local geometric features, such as, curvature or dihedral angle. Some techniques may compute boundaries that have short and smooth shapes. In our case, we want to refine the target boundaries that may follow particular patterns such as jagged or wavy shapes according to the corresponding source boundaries. We use snake [19] to refine B^t based on the following energy function:

$$E_{snake}(B^t) = \int_{t \in B^t} \left(E_{feature}(t) + E_{shape}(t)\right) dt,$$

where $E_{shape}(B^t)$ and $E_{feature}(t)$ are the shape term and the feature term for a sequence of sample points t of B^t, respectively. $E_{feature}(t)$ controls the target boundary for getting close to the geometric features and $E_{shape}(t)$ maintains the shape of B^t as similar as the shape of B^s. The snake scheme [19] is adopted for searching the samples of the snake (snaxel) that locate on either the vertices or edges of the target mesh. We compute the corresponding point s of B^s for each point t of B^t based on equation 4. We then compute the offset of each point which is the distance between each boundary point and the fitting plane. The offset of a point is normalized by the length of the bounding box diagonal of the mesh. Assume $\mu(s)$ and $\mu(t)$ are the offsets of s and t, respectively. Then $E_{shape}(t) = |\mu(s) - \mu(t)|$.

To avoid the local minimum, a searching region centered at the point t is used for finding the closest vertex whose dihedral angle [18] is similar to s. The size of the region is defined as the longest geodesic distance of the mesh multiplied by a small constant (e.g. 0.02). If the searching region is too large, the target boundary may be drifted away. The feature energy is computed as $E_{feature}(t) = geo(t, c(t))$, where $geo(\cdot, \cdot)$ is the geodesic distance (on the mesh surface) between two points, c(t) is the closest vertex in the searching region of t such that $|dihedral(s) - dihedral(c(t))| < \lambda$. If there does not exist such c(t), $E_{feature}(t) = 0$. In our experiments, λ is set to 0.1.

5 Results and experiments

We present the results of segmentation transfer and the timing information in this section.

Results. Fig. 3 shows the two different types of segmentations for the four-legged animals and the corresponding segment pairs are drawn the same colors. For the top example of Fig. 3, the boundaries cross the legs and form the short shapes. For the bottom example, the boundaries cross the legs in the tilted manners and form longer shapes. We can see that the corresponding boundaries of target models have similar characteristics. Fig. 9 shows more segmentation transfer results. If the target model does not have parts corresponding to the source model (the ears of the dog and human in Fig. 9), the corresponding boundaries are not transferred.

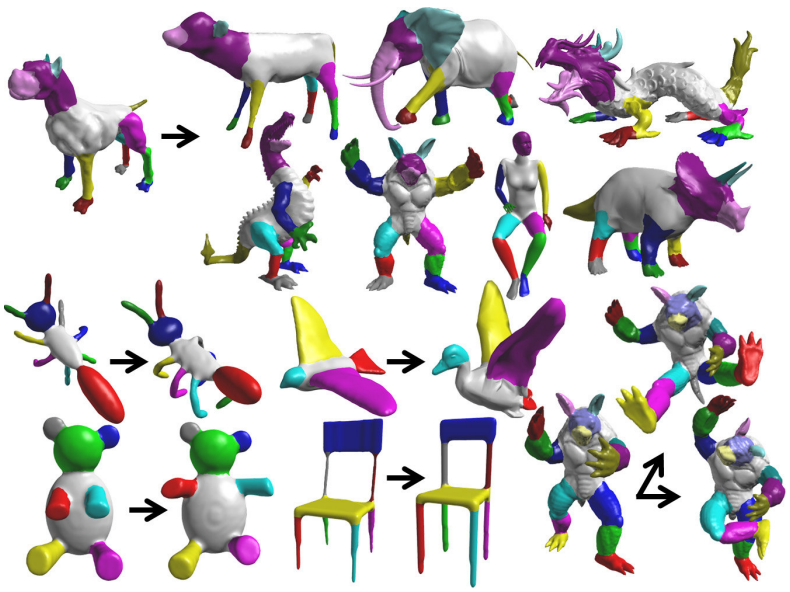

Fig. 9 The segmentation transfer results. The source segmentations (dog, ant, bear, bird, chair and armadillo) are on the left hand side of the arrows. The corresponding segment pairs are drawn in the same colors. The parameter values: θ = 1.5, K = 0.05, λ = 0.1. For sheep, λ = 1.0.

Comparisons. We compare with the method by [10] and Fig. 10 shows the results. The results produced by [10] are poor for the models with large difference in shapes or poses. Our method produces better results since our method adopts skeleton correspondence before segmentation transfer is performed.

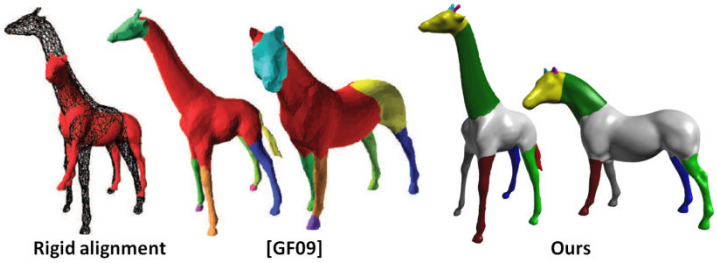

Fig. 10 Comparison with [10] for segmentation transfer.

In comparing with the method by Kalogerakis et al. [14], our method is more capable of controlling the positions of boundaries. Notice the differences to the rightmost airplanes in Fig. 11 and the giraffes in Fig. 12. The seat and back of the rightmost chair in Fig. 13 form a single segment since there is no boundary on some pillars. The method of [14] is region-based so that it can produce segments for all the pillars of the chairs. Finally, we compare with the other segmentation algorithms based on the Princeton Benchmark [8]. The results between ”Human” and our method ”Seg Trans.” are similar to each other. Our method can consistently transfer the source segmentation to the target mesh. The results also show that our method is comparable to the other methods, including Randomized Cuts [9], Shape Diameter Function [26], Normalized Cuts [9], Core Extraction [15], RandomWalks [18], Fitting Primitives [1] and K-Means [27], as indicated in Fig 14.

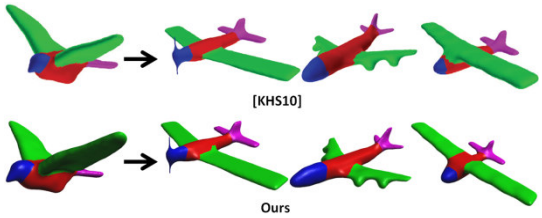

Fig. 11 The comparison with [14] on airplane models. The wings of the rightmost airplanes have difference segmentation types. A target segment boundary is not smooth in the second left airplane in our approach due to the poor alignment of the fitting plane.

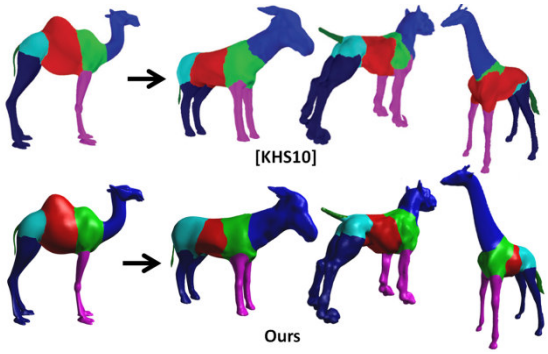

Fig. 12 The comparison with [14]. Our result is better for the giraffe.

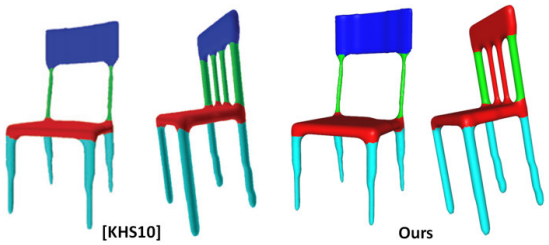

Fig. 13 The comparison with [14]. The seat and back chair form a single segment for the right chair.

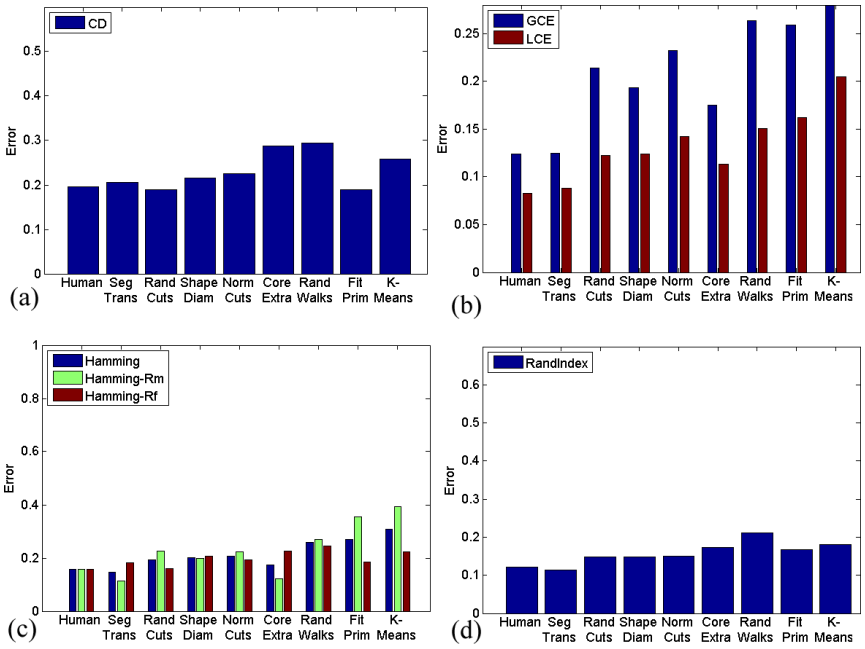

Fig. 14 Princeton Segmentation Benchmark: Comparison of segmentation algorithms with four evaluation metrics. Our method is denoted as "Seg Trans." and the given segmentation is denoted as "Human". The four metrics: CD: Cut Discrepancy; Consistency error (Global Consistency Error and Local Consistency Error); Hamming distance and RandIndex.

Timing information. All the results were performed on Intel Core 2 Duo CPU E8400 3.0GHz with 4GB memory, using a single thread implementation. We precomputed AGD function, MSP function, skeleton extraction and dihedral angle function. The total preprocessing time ranges from one minute to 30 minutes. The computation time depends on the complexity of the objects.

Table 2 The computation time of the segmentation transfer results in Fig. 10.

Model	Number of faces	Segment transfer time (seconds)	Number of boundaries	Avg. boundary transfer time (seconds)
Dog(source)	18976	-	15	-
Deer	7402	1.31	15	0.09
Human	11258	1.46	11	0.13
Armadillo	20000	2.15	15	0.14
Dragon	16000	1.51	13	0.12
Triceratops	15764	1.48	15	0.10
Elephant	30000	3.75	15	0.25
Asian dragon	28198	3.18	15	0.21
Armadillo(source)	50382	-	17	-
Armadillo(tg 1)	50212	6.21	17	0.37
Armadillo(tg 2)	49226	6.08	17	0.35

Table 2 lists the computation times of our segmentation transfer results presented in Fig. 10, including the total computation time, the number of segment boundaries, and the average time for performing a boundary transfer. On average it took around four seconds to perform segmentation transfer. Our method requires the manual operations to change the parameters if the segmentation results are not good. Usually, it took three to four attempts to obtain high quality results.

Discussions and limitations. The quality of the segmentation transfer results depends on the quality of the skeleton correspondence. If the skeletons do not capture the shape of the meshes faithfully, our method may not work properly. If the fitting planes do not fit well the source boundary, the arc ratios are not reliable for determining the target boundary positions. We cannot handle segment boundaries which enclose multiple arcs. The target boundaries may be distorted for non-cylindrical covering regions. Further studies are needed for building the local coordinate systems for computing the covering regions of the target segment boundaries.

6. CONCLUSION

We have presented a segmentation transfer approach for mapping consistently a segmentation of a source mesh to a target mesh. The segmentation transfer approach is based on the skeleton correspondence between the meshes. The characteristics of boundaries including the relative orientation between the boundaries and skeletal arcs, the shape of the corresponding parts and local surface features, are considered. We have shown that our method generates consistent segmentations for a variety of similar meshes of different shapes and poses. Our approach relies on the skeleton-mesh correspondence and skeleton correspondence to perform segmentation transfer. Our method does not handle the segment boundaries that do not enclose a skeletal arc.

In the future, we would like to develop techniques to transfer any kind of segment boundaries by combining consistent parameterization with skeletons. We believe that the skeleton-based approach can enhance the quality of the segment boundaries for the techniques based on consistent parameterization and other segmentation techniques.

7. REFERENCES

1. M. Attene, B. Falcidieno, and M. Spagnuolo, “Hierarchical mesh segmentation based on fitting primitives,” *The Visual Computer*, 22(3):181–193, 2006.
2. M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal, “Mesh segmentation - a comparative Study,” in *IEEE Int’l Conf. on Shape Modeling and Applications*, page 7, 2006.
3. O.-C. Au, C. Tai, D. Cohen-Or, Y. Zheng, and H. Fu, “Electors voting for fast automatic shape correspondence,” *Comp. Graph. Forum*, 29(2):645–654, 2010.
4. O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, “Skeleton extraction by mesh contraction,” *ACM TOG*, 27(3), 2008.
5. P. Besl and N. McKay, “A method for registration of 3-D shapes,” *IEEE Trans. on PAMI*, 14(2):239–256, 1992.
6. S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno, “Sub-part correspondence by structural descriptors of 3D shapes,” *Comp. Aided Design*, 38(9):1002–1019, 2006.
7. Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004.
8. X. Chen, A. Golovinskiy, and T. Funkhouser, “A benchmark for 3D mesh segmentation,” *ACM TOG*, 28(3):1–12, 2009.
9. A. Golovinskiy and T. Funkhouser, “Randomized cuts for 3d mesh analysis*,” ACM TOG*, 7(5), 2008.
10. A. Golovinskiy and T. Funkhouser, “Consistent segmentation of 3D models,” *Computers and Graphics*, 33(3):262–269, 2009.
11. R. Gonzalez and R. Woods, “Digital image processing”, *Addison-Wesley Publishing Company*, 1992.
12. M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, “Topology matching for fully automatic similarity estimation of 3D shapes,” in *SIGGRAPH*, pages 203–212, 2001.

13. T. C. Ho, “Slice-driven shape analysis and geometry processing of 3d models,” Ph.D. thesis, The National Chiao Tung University, Taiwan, R.O.C., 2011.
14. E. Kalogerakis, A. Hertzmann, and K. Singh, “Learning 3D mesh segmentation and labeling,” *ACM TOG*, 29(4):1–12, 2010.
15. S. Katz, G. Leifman, and A. Tal, “Mesh segmentation using feature point and core extraction,” *The Visual Computer*, 21(8-10):649–658, 2005.
16. S. Katz and A. Tal, “Hierarchical mesh decomposition using fuzzy clustering and cuts,” *ACM TOG*, 22(3):954–961, 2003.
17. V. Kraevoy and Sheffer, “Cross-parameterization and compatible remeshing of 3d models,” *ACM TOG*, 23(3):861–869, 2004.
18. Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, “Fast mesh segmentation using random walks,” in *ACM Symp. on Solid and Phys. Modeling*, pages 183–191, 2008.
19. 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(444–465), 2005.
20. R. Liu and H. Zhang, “Mesh segmentation via spectral embedding and contour analysis,” *Computer Graphics Forum*, 26(3):385–394, 2007.
21. M. M¨uller, B. Heidelberger, M. Teschner, and M. Gross, “Meshless deformations based on shape matching,” *ACM TOG*, 24(3):471–478, 2005.
22. E. Praun, W. Sweldens, and P. Schr¨oder, “Consistent mesh parameterizations,” in *SIGGRAPH*, pages 179–184, 2001.
23. J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, “Intersurface mapping,” *ACM TOG*, 23(3):870–877, 2004.
24. A. Shamir. A survey on mesh segmentation techniques,” *Computer Graphics Forum*, 27(6):1539–1556, 2008.
25. L. Shapira, S. Shalom, A. Shamir, R. H. Zhang, and D. Cohen-Or, “Contextual part analogies in 3d objects,” *Int’l Journal of Comp. Vision*, 89(2–3), 2010.
26. 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.
27. S. Shlafman, A. Tal, and S. Katz, “Metamorphosis of polyhedral surfaces using decomposition,” *Computer Graphics Forum*, 21(3):219–228, 2002.
28. P. Simari, D. Nowrouzezahrai, E. Kalogerakis, and K. Singh, “Multi-objective shape segmentation and labeling,” *Computer Graphics Forum*, 28(5), 2009.
29. H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, “Skeleton based shape matching and retrieval,” in *Shape Modeling International*, page 130, 2003.
30. T. Tung and F. Schmitt, “Augmented reeb graphs for content-based retrieval of 3D mesh models,” in *Shape Modeling International*, pages 157–166, 2004.
31. O. van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh, “Prior knowledge for part correspondence,” *Computer Graphics Forum*, 30(2), 2011.
32. O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, “A survey on shape correspondence,” in *Eurographics State-of-the-art Report*, 2010.
33. Y. Wang and J. Zheng, “Tubular triangular mesh parameterization and applications,” Computer *Animation and Virtual Worlds*, 21(2):91–102, 2010.
34. 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*, pages 456–465, 2009.
35. H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi, “Deformation-driven shape correspondence,” in *Symposium on Geometry Processing*, pages 1431–1439, 2008.

出國訪問研究成果報告

計畫編號	NSC 99-2221-E-009-118　　執行期間 2010/08/01 ~ 2012/7/31
計畫名稱	多邊形網格分割轉換(2/2)
出國人員姓名 服務機關及職稱	莊榮宏/交大資訊工程系教授
出國時間地點	3/25/2012~4/1/2012 UC Davis, CA, USA
訪問研究主題	（中文）網格分割轉換，人臉彩妝模型與即時顯像技術，與巨量視訊資料分析與應用之研究 （英文）Mesh Segmentation Transfer, Facial Cosmetic modeling and real-time rendering, analysis of large video data and its applications

一、出國訪問經過

計畫主持人安排 3/25/2012~4/1/2012 到 University of California at Davis 作訪問研究。UC Davis 電腦系在圖學(graphics)與視算(visualization)有堅強的研究團隊，有兩個相關研究中心： Institute for Data Analysis and Visualization (IDAV) 及 Institute for Ultra-Scale Visualization。我此次訪問是與 Institute for Ultra-Scale Visualization 的馬匡六教授合作，透過參與馬教授的實驗室研討與討論做了很豐富的交流。

二、訪問研究成果

此次訪問討論的方向是討論網格分割轉換，人臉彩妝模型與即時顯像技術，巨量視訊資料之分析與應用之技術交流與未來合作規劃。頭兩項是進行中的研究，最後一項則是新的合作方向。

特徵或幾何運算之轉換在幾何處理是一個重要的研究課題。此研究之動機是現今網格分割的諸多方法各有其優缺點，各有其適合的網格型態，通常一個方法無法對所有物體產出好的分割結果，而且分割的結果常與人為分割結果有所差距。我們的方法可以將一個好的網格分割結果轉換到一個類似的物體。目前我們的作法比較是以幾何出發，藉由骨架對應的方法先將來源網格分割的邊界儘量對應到目標網格上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。此方法對肢體動物之測試成果不錯，但對形體拓樸差異較大者之效果表現不太穩定，這次與馬教授討論的方向是希望可以應用機器學習的觀念到網格分割轉換，。

模擬逼真的彩妝效果在3D 臉部動畫以及化妝品產業都是一個相當重要的問題。目前的方法都是使用影像扭曲技術，把化妝效果從一個人臉的化妝影像轉移到另外一個人臉影像上。然而，這些方法有形狀變形失真的問題，且要求輸入的影像必須要有相似的拍攝角度、臉型以及打光參數才能夠使用，而且沒有考慮皮膚與彩妝整合之顯像效果。我們正發展一個整合人類皮膚顯像以及彩妝模擬的即時顯像方法。人類皮膚顯像將會使用螢幕空間的半透明材質顯像技術，而彩妝模擬將會使用Kubelka-Munk 模型來表示。此外，我們還設計如何量測化妝品，包含粉底、腮紅、唇膏等，的材質資訊，再從量測的材質光譜資料推導出符合Kubelka-Munk模型或其他多層次半透明散射模型的顯像相關參數。此成果能顯像蠻逼真的人臉彩妝效果，但我們希望能更精進的發展出更複雜更適合的人臉彩妝的多層次顯像模型，這包涵在具微幾何(Microfacet)的人臉皮膚擦上液態及粉狀粉底後對折、反射及半透明散射之影響為何，推導出精確的光反射顯像模型，及光線對多層化妝品後折、反射及半透明散射之影響。我們在這些部份也做了深入的討論。

最後的討論項目是巨量視訊資料之分析與應用。隨著資訊時代的進步，先進的計算、圖像與感測技術使科學家得以更精確的研究自然與物理現象，但同時也創造急速增加的資料，而網路與行動裝置所傳送或收集的資料則更為巨量。將這些巨量資料之分析應用於決策及知識探索已是目前資訊學界的一個重要研究問題。 我們討論的對象是目前在很多地方，

如在高速公路、大遊樂園、校園或是博物館，已廣設的監視攝影機所產生的巨量streaming資料。討論的技術主題包涵

1. Streaming data processing and management， 如streaming data extraction, summarization, redundant data removal, similarity matching等

2. Movement data analysis， 如路線分析

3. Visualization， 如人群行動或路線之視覺化，Multi-resolution viewing等

4. Simulation，如人群行動模擬、火災模擬等

我們將持續討論這些主題，期待能成為交大圖學同仁與馬教授合作的一個整合型計畫。

國科會補助計畫衍生研發成果推廣資料表

日期:2012/08/12

國科會補助計畫	計畫名稱: 多邊形網格分割轉換
	計畫主持人: 莊榮宏
	計畫編號: 99-2221-E-009-118- 學門領域: 計算機圖學

無研發成果推廣資料

99 年度專題研究計畫研究成果彙整表

計畫主持人：莊榮宏	計畫編號：99-2221-E-009-118-
計畫名稱：多邊形網格分割轉換	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	2	2	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	2	2	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 □未發表之文稿 ■撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：（以 100 字為限）

已撰寫成兩篇論文投稿至國際期刊

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

在此計劃中，我們提出一個一致的網格分割轉換的演算法，它可以將來源網格的分割結果對應到目標網格上。來源和目標網格語義上需要相似，但幾何上可以有所不同。我們先發展出一個兩個網格間的新的骨架對應方法，利用骨架節點和其周圍網格頂點的幾何資訊以及拓撲結構來做骨架對應。但是對於不同的部位，骨架也許也會有不同的連結方式，為了降低這種不確定性，我們建立了樹狀結構，產生好幾種不同的結果，然後從中選擇最好的一個。而網格分割轉換就藉由骨架對應先將來源網格分割的邊界儘量對應到目標網格上，再根據幾何的細節去調整目標網格的分割邊界，使目標網格的幾何細節能夠和對應到的來源分割邊界相似。這兩個網格處理的演算法都具

相當的創新及產業應用價值。 已撰寫成兩篇論文，正投稿國際期刊中。