Segmentation methods for deforming meshes and its application to similarity measurement

Guoliang Luo

UNIVERSITÉ DE STRASBOURG

THÈSE DE DOCTORAT présentée par :

Guoliang LUO

soutenue le : **04 novembre 2014**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Mention : Informatique

Segmentation de Maillages Dynamiques et Son Application pour le Calcul de Similarité

THÈSE dirigée par :

Mme. SEO Hyewon	HDR, université de Strasbourg

RAPPORTEURS :

Mme. HAHMANN Stefanie	Professeur, Institut polytechnique de Grenoble
M. DAOUDI Mohamed	Professeur, Institut Mines-Télcom/Télécom Lille1

AUTRES MEMBRES DU JURY :

M. CORDIER Frederic	HDR, Université de Haute Alsace
M. BASKURT Atilla	Professeur, INSA Lyon
M. XU Yong	Professeur, South China University of Technology

École doctorale N^0 269 : mathématiques, sciences de l'information et de l'ingénieur.
UMR 7357 CNRS / Unistra : laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie

Acknowledgement

First and foremost, I wish to deeply thank my supervisor HDR Hyewon Seo for her patience, encouragement, supports and guidance on my research works throughout these years that makes this thesis possible. I would also like to give my sincere gratitude to my co-supervisor HDR Frederic Cordier, who provided tremendous of valuable remarks and assistance on my works and papers. Moreover, I would like to thank my juries Prof. Stefanie Hahmann, Prof. Mohamed Daoudi, Prof. Atilla Baskurt and Prof. Yong Xu, for expressing their genuine appreciation on my work and their helpful comments on my manuscript.

I would like to extend my gratitude to Prof. Nadia Magnenat-Thalmann for inviting and supporting my visits to Miralab in University of Geneva, where I had benefited from an intellectually inspiring environment with colleagues Hon-Fai Choi, Yvain Tisserand, Andra Chincisan and Matthias Becker.

I also would like to take this opportunity to thank my previous supervisors Prof. Danica Kragic and Assistant Prof. Carl Henrik Ek in Royal Institute of Technology, Assistant Prof. Anders Brun in Uppsala University, who guided me to enter research world.

I am particularly grateful to my colleagues in Computer Graphics and Geometry group, who were always willing to help and give their best suggestions. They are Vasyl Mykhalchuk, Amir Hossein Jaberzadeh Ansari, Kenneth Vanhoey, Lionel Untereiner, Rémi Imbach, Olivier Génevaux, Sylvain Thery, Frédéric Larue. It would be a lonely journey of this thesis without them. Special gratitude goes to Noura Hamzé, Pierre Boutry, Alexandre Hurstel, Etienne Schmitt and Thomas Pitiot, for their great efforts for translating my abstract into French.

Many thanks to my friends Yihan Liu, Blazejewska Katarzyna, Zilong Zhao, who shared me countless pieces of precious memories and enjoyable moments in France. A special thanks also to Tinghui Chen, who could always cheer me up whenever I needed it.

Finally, I cannot thank my parents enough for their lasting love, supports and understanding.

Résumé

1. Contexte

Avec le développement important des techniques d'acquisition de la géométrie, les données 3D sont devenues un nouveau sujet de recherche permettant au calcul de maillages de devenir un important theme de recherches.

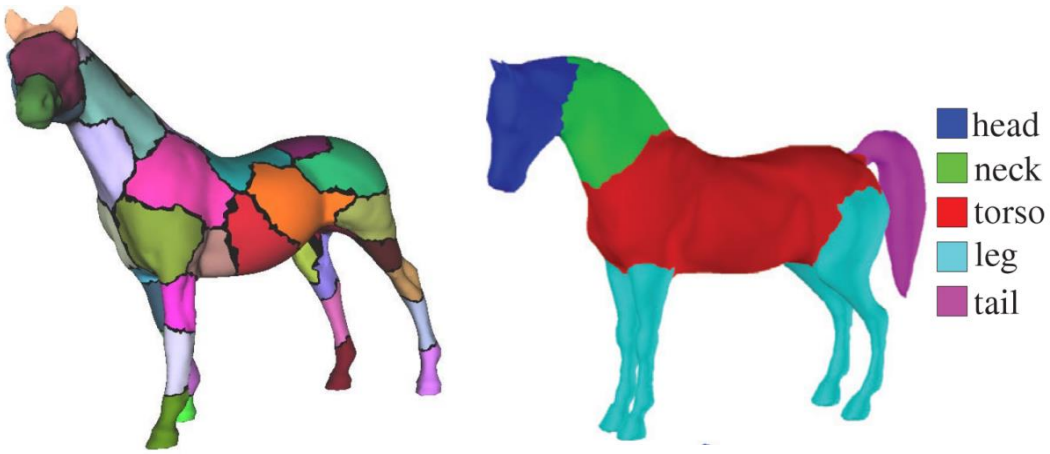
Durant les deux dernières décennies, la segmentation de maillages a été mise en avant en tant que première étape permettant d'extraire l'information sémantique vers le calcul et l'analyse de maillages pour de nombreuses applications. Par exemple, les algorithmes de "shape matching" peuvent être basés sur une décomposition de chaque état d'une forme, suivie d'une reconstruction à partir de sous-parties (Petitjean, 2002). La simplification de maillage ne pouvant être réalisée sans perte de certaines propriétés géométriques dûes à la segmentation du maillage en zones planaires et incurvées puis à la simplification des zones planaires (Sheffer, 2001). Une autre application classique de la segmentation de maillages est la paramétrisation (Julius et al., 2005). Elle permet à un utilisateur de décrire et contrôler une forme à partir d'un ensembles de parmètres de chaque sous-partie. Cette technique est utilisée pour les applications telles que le "mapping" de textures (Zhang et al., 2005) et le remaillage (Praun et Hoppe, 2003). D'autres applications basées sur la segmentation de maillages font de la compression (Karni et Gotsman, 2000), de la reconstruction (Funkhouser et al., 2004), de l'édition (Kovar et al., 2002), etc.

Etant donné un maillage 3D statique, l'objectif de la segmentation est de spatialement partitionner un maillage en plusieurs dans une des deux manières suivantes :

- Homogénéité de caractéristiques dans chaque partie, c.-à-d., les éléments dans le même segment partagent des propriétés géométriques similaires (voir Figure 1(a)). En raison de la similarité géométrique dans chaque segment, un nombre plus petit de coefficients spectrales va être demandé

pour reconstruire le segment en utilisant l'analyse spectrale, et par conséquent, attendre la compression de maillage.

- Sémantiquement significatif ou segmentation en parties fonctionnelles, par ex., une forme de cheval peut être segmenter dans un torse, une tête, un cou, une queue, et quarte jambes (voir Figure 1(b)) (Kalogerakis et al., 2010). Basé sur telles résultats de la segmentation de maillages, on peut soit extraire un squelette d'un maillage que peut être utile pour la création des animations (Katz and Tal, 2003), soitt associer les parties fonctionnelles entre des formes qui peuvent être étendues plus loin pour des applications tel que l'extraction de formes (Petitjean, 2002), ou les segmentations constantes (Kalogerakis et al., 2010) (Sidi et al., 2011), etc. Nous allons résumer les méthodes existantes qui génèrent telles résultats de segmentation de maillage dans Chapitre 2.



(a) (b)

Figure 1 Deux types de segmentation de maillages, (a) homogénéité dans chaque segment (Karni and Gotsman, 2000), (b) segments fonctionnellement significatifs (Kalogerakis et al., 2010).

1.1 Enjeux globaux de la segmentation de maillages

Comme cité par Attene et al. (Attene et al., 2006), il est très difficile de concevoir une méthode de segmentation pour des maillages statiques qui répondre parfaitement aux toutes les critères d'évaluation, y compris l'extraction de segments corrects, les frontières entre les segments, le type de la segmentation multi-échelle, la sensibilité de la forme et la complexité asymptotique. Il est difficile car les différentes méthodes de segmentation ont des différentes critères en fonction de leur

applications, et leurs critères de segmentation peuvent difficilement couvrir tous les types de maillages. Dans une recherche ultérieure Chen et al. (Chen et al., 2009), comparent la performance de plusieurs méthodes avancées de segmentation en utilisant des résultats de segmentation données par une groupe de presonnes, et ils ont arrivés à une conclusion similaire : Il reste toujours difficile à développer une méthode de segmentation qui peut fonctionne bien sur tout les types de maillages parce que les critères géométriques ne peuvent pas fournir tous les signales pour identifier toutes les parties sémantiques significatives. Par exemple, la méthode de segmentation par l'essayage de primitives, y compris un plan, une sphère, et un cylindre, peuvent être performantes avec des formes mécaniques mais pas avec des objectes complexe comme un maillage d'un « oiseau », car les ailes des oiseaux peuvent pas retrouver leurs formes par les primitives basiques.

1.2 Enjeux globaux de la segmentation de maillages dynamiques

Grâce au récents et rapides développements des technologies d'animation, les maillages dynamiques sont de plus en plus omniprésents. Bien qu'un nombre important de travaux ont été effectués sur les maillages statiques durant les deux dernières décennies, la recherche sur les maillages dynamique reste un sujet relativement récent.

Mis à part les enjeux principaux de la segmentation de maillages, présentés dans la sous-section précédente, la segmentation de maillages dynamiques reste particulièrement délicate pour les raisons suivantes :

- *Taille des données en entrée* : A l'inverse des maillages statiques qui ne contiennent que 3 dimensions (spatiales), les maillages dynamiques comportent une dimension supplémentaire, le temps, ce qui induit un problème relativement à la tailles des données. Un maillage dynamique standard sur, par exemple, une durée de 1 minute avec un taux de rafraîchissement de 30 images par seconde, regroupe sans peine plus de 1800 maillages, ce qui représente une augmentation très importante de la taille des données à traiter.
- *Comportement dynamique* : Contrairement aux méthodes existantes pour les maillages statiques, qui ont été développés en se reposant sur des propriétés figées de figures géométriques statiques, un algorithme de segmentation pour des maillages dynamiques doit tenir compte du comportement dynamique de ces maillages qui se caractérise par le mouvement dans le temps de leurs différentes sous-parties. Bien qu'il existe plusieurs méthodes de segmentation de maillages dynamiques, qui calculent une unique décomposition spatiale des maillages en utilisant les comportement dynamiques des primitives du

maillages traité (sommets, arêtes, ou triangles) sur la totalité d'une unique séquence (Sattler et al., 2005) (Wuhrer et al., 2010), ces méthodes, appliquées sur une longue séquence d'un maillage dynamique comportant plusieurs mouvement distincts, conduisent à un résultat découpé en *'sous-parties de mouvement'* souvent mal segmentées. Dès lors, se présente la nécessité de disposer d'un algorithme qui puisse à la fois découper un mouvement en plusieurs sous-séquences tout en segmentant spatialement la surface d'un maillage dynamique pour chacune des ces sous-séquences.

L'évaluation du résultat de la segmentation : Bien que l'on veuille décomposer un maillage animé en sous-parties pertinentes ou cohérentes, la formalisation mathématique de ce que la perception humaine considère comme étant 'pertinent' est très difficile, ce qui rend l'évaluation objective des résultats d'une segmentation compliquée. Dans le cas de segmentations de maillages statiques plusieurs études basées sur une appréciation humaine de telles segmentations ont été proposées ces dernières années (Chen and Funkhouser, 2009) (Bronstein et al., 2008). Cependant de telles données n'existent pas, à l'heure actuelle, pour des animations 3D, ce qui constitue donc un autre aspect du travail à effectuer.

2. Objectifs et contributions

Les maillages dynamiques peuvent être classifié en deux catégories: les maillages animés et les séquences variantes de maillages. Une animation 3D est un maillage animé si sa topologie reste constante durant la totalité de la séquence, c'est à dire si le nombre de sommet ainsi que la connectivité sont constants. Sinon, on parle de séquences variantes de maillages. Afin d'effectuer la segmentation d'une séquence variante de maillages. Il peut être nécessaire de calculer la correspondance de vertices entre les images successives où le problème de correspondance demeure une tâche difficile et coûteuse en calcul (Van et al., 2011) (Arcila et al., 2013). C'est pour cette raison que nous avons fait le choix de nous intéresser aux maillages animés. Nous allons nous intéresser plus particulièrement aux techniques de segmentation pour les maillages animés.

Cette thèse a pour but de développer des méthodes de segmentation qui calcule la segmentation temporelle et spatio-temporelle de maillages animés. À notre connaissance, aucune méthode n'avait encore été proposée pour la segmentation à la fois temporelle et spatio-temporelle de maillages animés jusqu'à aujourd'hui. De plus, nous étendrons les résultats de segmentation afin de mesurer la similarité entre maillages animeés. Ceci peut être significatif car cela permet de résoudre un

problème qui ne pouvait pas être traité jusqu'alors. Concernant le problème de la segmentation des maillages dynamiques présenté dans la section précédente, notre segmentation temporelle permet de diviser un mouvement en sous-mouvement ce qui permet de répondre à l'enjeu des comportements dynamiques d'un maillage animé. De plus, pour résoudre le problème d'évalutation, nous étendons notre segmentation spatio-temporelle afin de mesurer la similarité de mouvements. Ainsi nous validons indirectement la qualité de la segmentation spatio-temporelle de maillages animés en comparant les similariteés obtenues calculées à celle obtenues en questionnant des humains.

Dans le reste de cette section, nous définissons formellement la segmentation spatiale, temporelle et spatio-temporelle de maillages animés puis nous résumons les contributions de ce thèse.

2.1 Définitions formelles

Définition 1 : Segmentation spatiale de maillages dynamiques (Shamir, 2008) (Arcila et al., 2013). Soit $\mathcal{M}$=(V, E, T) la topologie d'un maillage dynamique, où V, E, T sont respectivement les ensembles de sommets, d'arêtes et de triangles. Une segmentation spatiale $\sum_s$ de $\mathcal{M}$ est un ensemble de sous-maillages $\sum_s$=$\{V_1, \dots, V_k\}$, $1 \leq i \leq kV_i = V$, où chaque $V_i, i = 1, \dots, k$ est un ensemble de sommets connetés. Notons que la segmentation spatiale peut aussi bien être définie comme une partition d'arêtes Eque de triangles$T$ en$k$ sous-ensembles disjoints.

Voir l'exemple de la Figure 2, une segmentation spatiale est calculée pour un maillage animé.

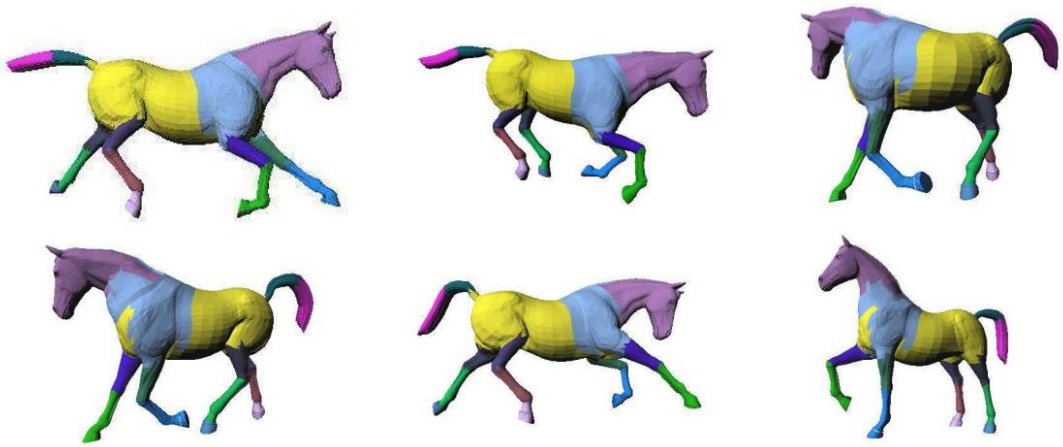


Figure 2 Un exemple d'une segmenation spatiale pour un maillage animé.

Definition 2 : Segmentation temporelle de maillages dynamiques (Arcila et al., 2013). Soit $\mathcal{M} = \{f^t, t = 1, \dots, M\}$ un maillage dynamique, oùM est le nombre d'images. Une segmentation temporelle $\sum_t$ de $\mathcal{M}$ est un ensemble de sous-séquences $\sum_t=\{F_1, \dots, F_k\}, 1 \leq i \leq kF_i = \mathcal{M}$, où chacun des $F_i, i = 1, \dots, k$ est un sous-séquence d'images successives.

Voir l'exemple de la Figure 3, un maillage animé est temporellement segmenté en plusieurs sous-séquences avec différents mouvements.

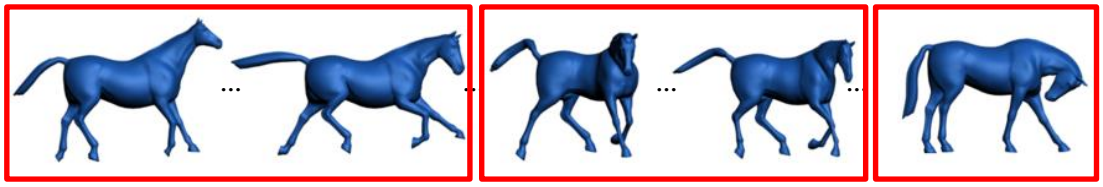


Figure 3 n exemple d'une segmentation temporelle de maillages animés.

Definition 3 : Segmentation spatio-temporelle de maillages dynalmiques. Soit $\mathcal{M} = \{f_i^t, t = 1, \dots, M, i = 1, \dots, N\}$ un maillage dynamique, avec *N* le nombre de sommets. Nous considérons $\mathcal{M}$ comme une donnée volumique, et définissons un segment spatio-temporel F_j^k comme un ensemble de sommets (ou de triangles) qui sont spatialement ou temporellement connectés les uns aux autres. Alors, le but de la segmentation spatio-temporelle est de partitionner un maillage dynamique M en segments spatio-temporels, i.e., $j, kF_j^k = \mathcal{M}$. Voir l'exemple de la Figure 4, une segmentation spatio-temporelle où les joints encerclés sont segmentés comme 'animé' ou 'rigid' selon leurs mouvements.

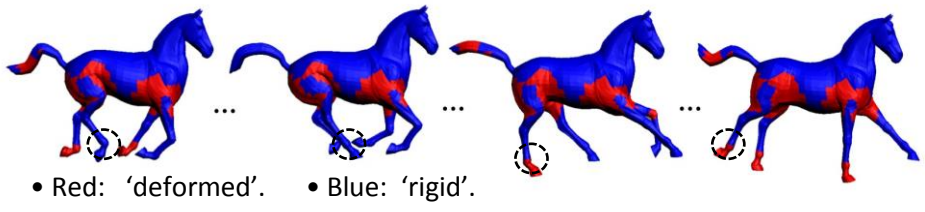


Figure 4 Un exemple d'une segmentation spatio-temporelle d'un maillage animé.

2.2 Contributions

Plusieurs travaux ont été publiés par le passé et qui permettent de segmenter un maillage animé en un ensemble de composants rigides. Dans cette thèse, nous présentons plusieurs techniques qui permettent de calculer une segmentation spatio-temporelle d'un maillage animé ; de tels travaux n'ont pas encore été publiés sur ce sujet. De plus, nous avons étendu cette méthode pour pouvoir comparer ces maillages animés entre eux à l'aide d'une métrique. À notre connaissance, aucune méthode existante ne permet de comparer des maillages animés entre eux. Dans mon travail de thèse, je présente les contributions suivantes :

Un nouveau descripteur pour les points caractéristiques dynamiques. Nous définissons un descripteur qui permet de mesurer les déformations au niveau des triangles et pour chaque image de l'animation. Ce descripteur est invariant par rapport à la rotation, translation et dilatation uniforme. De plus, il permet d'identifier la similarité entre des formes différentes, mais ayant les mêmes déformations.

Segmentation temporelle de maillages animés. La plupart des travaux existants sont basés sur le regroupement de sommets ou de triangles suivant des critères comme la distance géodésique ou l'affinité cinématique des sommets ou triangles regroupés. Dans ce cas, il apparait clairement que le résultat de la segmentation dépend de la déformation du maillage animé. Idéalement, cette segmentation doit correspondre aux mouvements du maillage animés. Mais lorsque le maillage animé correspond à une animation longue et complexe, le résultat est une segmentation trop morcelée qui ne représente pas correctement l'animation du maillage animé. C'est pour cette raison que nous pensons que la segmentation temporelle doit être précédée d'une segmentation spatiale. De cette façon, il est possible d'avoir une segmentation spatiale consistante à l'intérieur de chaque segment temporel.

À l'aide de nouveau descripteur, nous définissons ensuite une métrique pour mesurer la distance entre deux images-clefs du maillage animé. Ceci nous permet ensuite de définir une métrique pour les segments temporels ; cette métrique est calculée comme étant la moyenne des distances des images-clefs appartenant à un même segment. Enfin, nous proposons une méthode de segmentation temporelle qui minimise la métrique, c'est-à-dire, la distance entre les images-clefs appartenant au même segment. Ceci nous permet d'obtenir une segmentation où chaque segment est constitué des images-clefs correspondant à un même mouvement du maillage animé.

Les résultats de cette méthode de segmentation sur des données synthétiques et des données de capture de mouvement démontrent son efficacité. En particulier, ces essais montrent que les maillages animés de forme différente, mais représentant les mêmes mouvements sont segmentés de façon consistante entre eux.

Segmentation spatio-temporelle de maillages animés. À partir de la méthode de segmentation temporelle décrite précédemment, nous avons développé une méthode exploitant la cohérence spatio-temporelle des maillages animés. Notre méthode de segmentation spatio-temporelle pour les maillages animés est basée sur une nouvelle représentation qui permet de décrire de façon précise les mouvements du maillage animé.

En utilisant les valeurs de déformation de chaque triangle à chaque image-clef, les triangles sont classés en deux catégories : les triangles se déformant et ceux qui sont rigides. Ensuite, nous calculons une segmentation spatio-temporelle en regroupant les triangles se déformant et qui sont adjacents au niveau du maillage et au niveau des images-clefs. Puis, nous utilisons un graphe évoluant (*evolving graph*) pour représenter la segmentation spatio-temporelle, un graphe évoluant étant constitué d'une séquence de graphes correspondant aux images-clefs du maillage animé ; chaque nœud du graphe évoluant correspond à un segment spatio-temporel et chaque arête correspond à une relation de voisinage entre les segments.

À partir des graphes évoluant de deux maillages animés, nous calculons une métrique pour mesurer le niveau de similarité entre les deux maillages animés ; cette métrique est basée sur l'alignement des deux séquences de graphes. Cet alignement ne peut pas être calculé directement du fait que les graphes sont de dimension différente (nombre différent de nœuds et d'arêtes). Pour résoudre ce problème, nous classons les graphes suivant leur similarité pour créer des ensembles de graphes ; à chaque ensemble de graphes est attribuée une étiquette. En conséquence, chaque graphe évoluant est représenté sous la forme d'une séquence d'étiquettes. Ceci nous permet de calculer l'alignement des deux séquences avec une méthode inspirée de celle utilisée pour l'alignement des séquences d'ADN. Le résultat de l'alignement des deux graphes évoluant est finalement utilisé pour calculer le niveau de similarité entre eux.

Les avantages de cette méthode sont doubles. Premièrement, elle permet de calculer l'alignement temporel des deux maillages animés. Les essais que nous avons menés avec des maillages animés représentant les mêmes mouvements ont montré que l'alignement temporel est correctement calculé. Deuxièmement, cette mé-

thode permet de mesurer la similarité de mouvements entre les maillages animés. Pour cela, nous avons proposé une métrique qui utilise les résultats de la segmentation spatio-temporelle. Les essais que nous avons effectués avec un certain nombre de maillages animés montrent que notre méthode est capable de détecter correctement les similarités de mouvements. Nos résultats sont validés en les comparant avec des données produites par un groupe des personnes et en calculant le facteur de corrélation de Pearson. Le résultat obtenu montre une forte corrélation entre les mesures de similarité calculée par notre algorithme et celles données par les personnes.

3. Conclusions et Organisation

À notre connaissance, aucune méthode n'avait encore été proposée pour la segmentation à la fois spatiale et temporale de maillages animés jusqu'aujourd'hui. Pour ce travail, les contributions techniques de nos travaux sont les suivantes. Premièrement, nous avons proposé une méthode qui calcule une segmentation temporelle en minimisant la dissimilarité des images-clefs appartenant au même segment. Cette méthode permet de générer une segmentation en cohérence avec le mouvement du maillage animé. Les maillages animés présentant les mêmes animations sont segmentés de la même façon. Notre deuxième contribution est une méthode de segmentation spatio-temporelle qui utilise la cohérence des déformations des triangles du maillage animé. Grâce à l'utilisation des graphes évoluant (evolving graph), nous avons pu proposer une méthode qui calcule l'alignement temporel de deux maillages animés ainsi qu'une métrique pour mesure le degré de similarité de mouvements entre ces deux maillages animés. Les essais que nous avons effectués avec plusieurs types de maillages montrent que notre méthode est capable de détecter la similarité de la même façon que la perception humaine.

Le résumé de la thèse est organisé de la façon suivante : dans Chapter 2, nous faisons l'état de l'art des travaux sur la segmentation pour un unique maillage statique, un ensemble de modèles 3D similaires et la segmentation spatialle de maillages dynamiques. Ensuite, nous présentons un nouveau descripteur basé sur la déformation dans Chapter 3. En utilisant ce descripteur dans Chapter 4, nous présentons notre méthode de segmentation temporelle pour les maillages dynamiques. Après quoi, dans Chapter 5, nous présentons une nouvelle méthode de segmentation spatio-temporelle pour les maillages dynamiques, permettant de mesurer les similitudes de mouvement entre différents maillages dynamiques. En-

fin nous concluons par plusieurs remarques sur nos travaux et évoquons des piste de poursuites potentielles dans Chapter 6.

Mots clefs : maillage qui se déforme, descripteur de points caractéristiques dynamiques, segmentation temporelle, segmentation spatio-temporelle, mesure de similarité de mouvements.

Abstract

With an abundance of animation techniques available today, animated mesh has become a subject of various data processing techniques in Computer Graphics community, such as mesh segmentation and compression. Created from animation software or from motion capture data, a large portion of the animated meshes are deforming meshes, i.e. ordered sequences of static meshes whose topology is fixed (fixed number of vertices and fixed connectivity). Although a great deal of research on static meshes has been reported in the last two decades, the analysis, retrieval or compressions of deforming meshes remain as new research challenges. Such tasks require efficient representations of animated meshes, such as segmentation. Several spatial segmentation methods based on the movements of each vertex, or each triangle, have been presented in existing works that partition a given deforming mesh into rigid components.

In this thesis, we present segmentation techniques that compute the temporal and spatio-temporal segmentation for deforming meshes, which both have not been studied before. We further extend the segmentation results towards the application of motion similarity measurement between deforming meshes. This dissertation consists of the following contributions.

A new dynamic feature descriptor. We begin by devising an efficient per-triangle feature descriptor that measures the deformation of each triangle at each frame. This descriptor is invariant to global shape rotation, translation and uniform scale. In our experiments, we observe that the new descriptor is robust over shape differences when different shapes performing identical motions, which is desirable.

Temporal segmentation of deforming meshes. Most existing works on deforming mesh compute spatial clustering according to geodesic and kinematic affinities of vertices or triangles. In such cases, it is clear that the spatial segmentation results may significantly be different depending on the deformation exhibited on a deforming mesh. Ideally, they should represent well the motion exhibited on the mesh. However, when it comes a long and complex motion composed of several basic

motions, one may obtain overly segmented patches, which do not represent well each basic motion. To this end, we believe that temporal segmentation should be preceded prior to spatial segmentation, so as to compute consistent spatial segmentation within each temporal segment.

Based on our new descriptor, we define a distance metric for each frame pair, and further define within-segment frame dissimilarity as the average of all possible pairwise frame distance within a temporal segment. Then, the boundary frames for the temporal segmentation are determined by minimizing the sum of within-segment frame dissimilarities. This allows us to obtain the segmentation result that each temporal segment is a subsequence of similar frames with similar poses.

Our experiments on both synthesized and motion captured deforming meshes confirm the effectiveness of the presented approach. It also shows that we can obtain consistent temporal segmentation for different deforming meshes exhibiting similar motions, despite their shape differences.

Spatio-temporal segmentation of deforming meshes. Having the above temporal segmentation method, we step further to investigate both the spatial and temporal coherency simultaneously in deforming meshes. We devise a new spatio-temporal segmentation technique for deforming meshes, with an aim of developing a new representation that encodes well the motions exhibited in given deforming meshes.

Based on the degree of deformation of each triangle at each frame indicated by using strains, we binarily label the triangles with either ‘deformed’ or ‘rigid’. Then we compute a spatio-temporal segmentation by merging the ‘deformed’ triangles that are either spatially or temporally connected. We then use an *evolving graph* to represent the spatio-temporal segmentation, where each node represents a spatial segment, each edge the neighbourhood between two spatial segments, and each graph is a key frame representing a subsequence of frames with the same graph representation.

Having computed the evolving graphs of two deforming meshes, we proceed to compute the similarity of the evolving graphs by adopting a sequence alignment method. However, a sequence alignment method cannot be directly applied on two graph sequences because the graphs may have different dimensions, i.e. different node numbers. In order to avoid this problem, we classify the similar graphs and assign the graphs in the same cluster with the same label. As a result, each evolving graph is represented into a sequence of cluster labels. Finally, we compute

the alignment score between the two cluster label sequences by using a sequence alignment algorithm, which reflects the similarity between two deforming meshes.

The outcome of this method is two folds: (1) Temporal frame alignment. According to our experiments, the alignment results between two deforming meshes with similar motions show that the key frames performing similar actions are well matched to each other. (2) Motion similarity measurement. Based on the spatio-temporal segmentation results, we have devised a similarity measurement method for deforming meshes, which measures the similarity of motions that are performed by deforming meshes. Our experimental results on a number of deforming meshes show that the motion similarities can be captured correctly, despite shape differences. We validate our similarity results by computing Pearson's correlation with human-based ground truth motion similarities. The obtained high correlation indicates that our motion similarity measurement method successfully reflects human perception on the motion similarities of deforming meshes.

Keywords : deforming mesh, dynamic feature descriptor, temporal segmentation, spatio-temporal segmentation, motion similarity measurement.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1 Introduction

1.1 Background

With an abundance of computer hardware and geometry acquisition devices available today, 3D mesh data have become a new research subject and accordingly the mesh processing has become an important research topic. During the last two decades, mesh segmentation has drawn a great deal of attentions because it is a primary step that extracts semantic information towards mesh processing and analysis for numerous applications. For example, shape matching and retrieval can be done based on the decomposition of each shape, followed by the matching of the sub-parts (Petitjean, 2002). One can also achieve a mesh simplification without much loss of geometrical properties by segmenting a mesh into planary and curved regions and then simplifying the planary regions (Sheffer, 2001). Another common output of mesh segmentation is a parameterization (Julius et al., 2005), which enable a user to describe and control the shape with a set of parameters of each sub-part. This is useful for the applications such as texture mapping (Zhang et al., 2005) and remeshing (Praun and Hoppe, 2003). Other applications based on mesh segmentation include compression (Karni and Gotsman, 2000), reconstruction (Funkhouser et al., 2004), editing (Kovar et al., 2002), etc.

Given a 3D static mesh, the object of segmentation is to spatially partition the mesh into multiple parts in either of the two following manners:

- Homogeneity of features within each part, i.e., the elements within the same segment share similar geometrical properties (see Figure 1-1(a)). Due to the geometry similarity within each segment, a smaller number of spectral coefficients will be needed to reconstruct the segment by using spectral analysis, and therefore achieves mesh compression (Karni and Gotsman, 2000).

- Semantically meaningful or functional parts, e.g., a horse shape can be segmented into a torso, a head, a neck, a tail and four legs (see Figure

1-1(b)) (Kalogerakis et al., 2010). Based on such mesh segmentation results, one can either extract mesh skeleton that can be useful for creating animation (Katz and Tal, 2003), or match the functional parts between shapes that can be further extended to the applications such as shape retrieval (Petitjean, 2002) and consistent segmentations (Kalogerakis et al., 2010) (Sidi et al., 2011), etc. We will summarize the existing methods that generate such mesh segmentation results in Chapter 2.

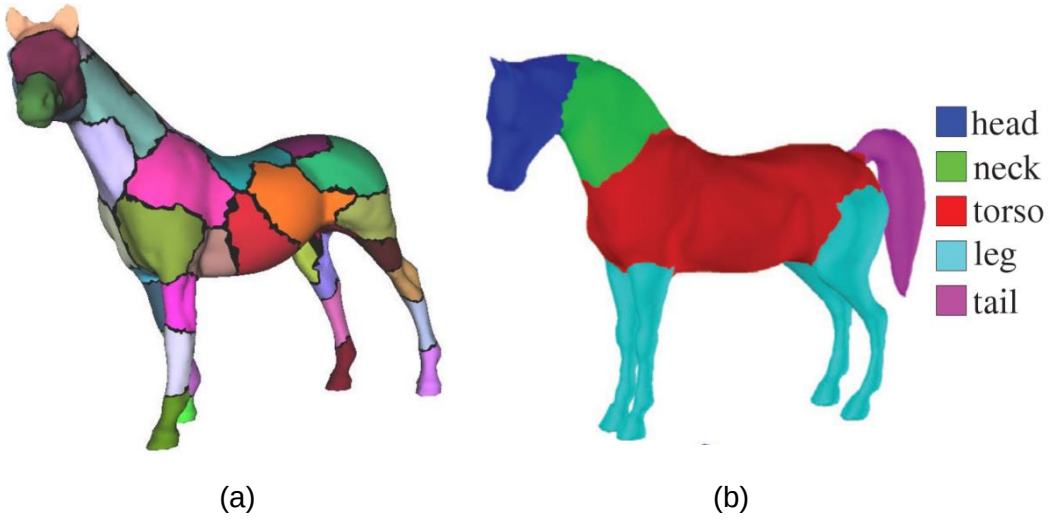


Figure 1-1 Two types of mesh segmentation, (a) homogeneity within each segment (Karni and Gotsman, 2000), (b) functional meaningful segments (Kalogerakis et al., 2010).

1.1.1 General challenges of mesh segmentation

As stated by Attene et al. (Attene et al., 2006), it is very difficult to devise a segmentation method for static meshes that perfectly meet all the evaluation criteria, including the extraction of correct segments, the boundaries between segments, the type of multi-scale segmentation, the sensitivity to pose and the asymptotic complexity. It is difficult because different segmentation methods have different segmentation criteria depending on their applications, and their segmentation criteria can hardly cover all mesh types. In a later research, Chen et al. (Chen et al., 2009) compare the performance of several advanced mesh segmentation methods by using human-based ground-truth segmentation results, and draw a similar conclusion: It remains a difficult problem to develop a segmentation method that can perform well over all mesh types because geometric criteria may not provide sufficient cues to identify all the semantically meaningful parts. For example, the segmentation method by fitting primitives, including a plane, a sphere, and a cylinder, can perform well with mechanical shapes but not with more complex objects such

as ‘bird’ meshes, because the bird’s wings could not be well fitted by the basic primitives.

1.1.2 General challenges of the segmentation of dynamic meshes

With the rapid advances of the animation technologies in recent years, dynamic mesh data are becoming ubiquitous. Although a great deal of research on 3D static mesh segmentation have been reported in the last two decades, the segmentation of dynamic meshes remains as a new research challenge.

Apart from the general challenges of mesh segmentation presented in the previous subsection the segmentation of dynamic meshes is particularly difficult due to the following reasons:

- *Input data size*. Unlike the static meshes that only contain 3D spatial dimensions, a dynamic mesh has an additional dimension of time, which poses a challenge because of larger problem size. Typically, a dynamic mesh of 1 minute with frame rate of 30 frames/second contains 1800 meshes, which is a significant increase of data size.

- *Dynamic behaviors*. In contrast to the existing segmentation methods for static meshes that are developed based on static geometrical features, a segmentation algorithm for dynamic meshes needs to take into account of the dynamic behaviors, which is the temporal movements of different mesh subparts. There exists several segmentation methods for dynamic meshes that compute a single spatial mesh decomposition based on the dynamic behaviors of the mesh primitives, i.e., vertices, edges, or triangles, in the entire mesh sequence (Sattler et al., 2005) (Wuhrer et al., 2010). By using these methods on a long dynamic mesh with different motions, we may obtain overly segmented patches that do not represent well of each sub-motion. In such cases, a segmentation algorithm that can both temporally divide a motion into sub-motions and spatially segment the mesh surface for each sub-motion would be preferable.

- *Evaluation of segmentation results*. One may aim to decompose an animated mesh into meaningful parts, whereas human perception of ‘*meaningful*’ is difficult to model mathematically, making the objective evaluation of segmentation results as another challenging task. To handle the evaluation problem of static mesh segmentation, several benchmarks of human-based ground-truth segmentation have been made available in recent

years (Chen and Funkhouser, 2009) (Bronstein et al., 2008). However, such ground-truth segmentation does not exist today for 3D animations, and constructing similar benchmark for 3D animations will be another challenging task.

1.2 Objectives and contributions

Dynamic meshes can be classified into two types: *deforming mesh* and *variant mesh sequence*. Given a 3D animation, it is a deforming mesh if the mesh has constant topology over the entire sequence, i.e., a fixed number of vertices and fixed connectivity. Otherwise, we consider it as a variant mesh sequence. In order to compute the segmentation of a variant mesh sequence, one may have to compute vertex correspondence among successive frames, where the correspondence problem however remains a challenging and a heavy computational task (Van et al., 2011) (Arcila et al., 2013). For this reason, we choose to work with deforming meshes so that we can focus on the discussion and investigation of the segmentation techniques for deforming meshes.

In this thesis, we aim to develop segmentation techniques that compute the temporal and spatio-temporal segmentation for deforming meshes, which both have not been studied before. Moreover, we further extend the segmentation results towards the application of motion similarity measurement between deforming meshes. This may be significant as it solves the problem that cannot be handled by current approaches. Concerning the challenges of the segmentation of dynamic meshes presented in the previous section, our temporal segmentation can divide a motion into submotions, which would address the challenge of dynamic behaviors performed in a deforming mesh. Additionally, in order to alleviate the challenge of evaluation problem, we extend the spatio-temporal segmentation for the application of motion similarity measurement. Thus, by evaluating the computed motion similarities with human-based ground-truth, we indirectly validate the quality of the spatio-temporal segmentation of deforming meshes.

In the remainder of this section, we formally define the *spatial segmentation*, the *temporal segmentation* and the *spatio-temporal segmentation* of deforming meshes, followed by a summary of the contributions of this thesis.

1.2.1 Formal definitions

Definition 1-1 : spatial segmentation of deforming meshes (Shamir, 2008) (Arcila et al., 2013). Let $\mathcal{M}$=(V, E, T) be the mesh topology of a deforming mesh, where V, E, T are vertex, edge and triangle sets, respectively. A spatial segmentation $\sum_s$ of $\mathcal{M}$ is a set of sub-meshes $\sum_s$=$\{V_1, \dots, V_k\}$, $\bigcup_{1\le i\le k} V_i = V$, where each of $V_i, i = 1, \dots, k$ is a set of connected vertices. Note that the spatial segmentation can also be expressed by a partition of either edges E or triangles T into k disjoint subsets.

Definition 1-2 : temporal segmentation of deforming meshes (Arcila et al., 2013). Let $\mathcal{M} = \{f^t, t = 1, \dots, M\}$ be a deforming mesh, where M is the total number of frames. A temporal segmentation $\sum_t$ of $\mathcal{M}$ is a set of subsequences $\sum_t$=$\{F_1, \dots, F_k\}$, $\bigcup_{1\le i\le k} F_i$ =$\mathcal{M}$, where each of $F_i, i = 1, \dots, k$ is a subsequence of successive frames.

Definition 1-3 : spatio-temporal segmentation of deforming meshes. Let $\mathcal{M} = \{f_i^t, t = 1, \dots, M, i = 1, \dots, N\}$ be a deforming mesh, where *N* is the total number of vertices. We consider $\mathcal{M}$ as a volumetric data, and define a spatio-temporal segment F_j^k as a set of vertices (or triangles) that are either spatially or temporally connecting to each other. Then, the object of spatio-temporal segmentation is to partition a deforming mesh $\mathcal{M}$ into spatio-temporal segments, i.e., $\bigcup_{j,k} F_j^k = \mathcal{M}$.

1.2.2 Contributions

To overcome the limitations of the existing spatial segmentation methods for deforming meshes, we propose new segmentation methods that investigate both temporal and spatial deformation coherency in deforming meshes. In this thesis, we first present a deformation-based descriptor to measure the degree of deformation of each triangle within each frame. Based on this feature descriptor, we devise a temporal segmentation algorithm that divides a deforming mesh into subsequences, each of which is a set of successive frames. In addition, we also propose a spatio-temporal segmentation algorithm for the efficient representation of deforming meshes. The major contributions of this thesis can be summarized as follows:

Deformation-based descriptor. To describe the dynamic motion information within a deforming mesh, we present a deformation-based feature descriptor to measure the degree of deformation for each triangle at each frame in a deforming mesh. This allows us to investigate the deformation coherency within a given deforming mesh by observing each triangle's feature descriptor. Based on this descriptor, we

devise the following two new segmentation methods for deforming meshes, which have not been studied before.

Temporal segmentation of deforming meshes. In recent years, several spatial segmentation methods have been proposed to segment a deforming mesh into near-rigid components (Sattler et al., 2005) (Lee et al., 2006) (Wuhrer et al., 2010). However, these methods overlook the temporal coherency, for which reason one spatial segmentation result may not be representative for each sub-motions. To address such problem, we devise a temporal segmentation algorithm by using the temporal coherency within deforming meshes. In specific, we cut a given mesh sequence so that each subsequence contains shapes with similar poses. Since we measure pose similarity based on our new deformation-based descriptor, we can obtain consistent temporal segments for deforming meshes that perform identical or similar motions, despite their shape differences.

Spatio-temporal segmentation of deforming meshes. Having the above temporal segmentation, we step further to explore both spatial and temporal deformation coherency at the same time in a deforming mesh. To achieve this, we devise a new spatio-temporal segmentation technique for deforming meshes, with an aim of developing a new efficient representation that encodes well the motions exhibited in the given deforming meshes. Knowing the advantage of graph that it is a convenient and compact representation for structured objects, we represent the spatio-temporal segmentation results of a deforming mesh into an evolving graph, a graph changes over time. Moreover, in order to validate the effectiveness of our spatio-temporal segmentation method and the graph representation of the segmentation results, we extend the segmentation results towards similarity measurement by comparing the corresponding evolving graphs.

1.3 Organization

The reminder of this thesis is organized as follows. In Chapter 2, we give a literature review of the existing segmentation works for a single static mesh, a set of similar 3D models and the spatial segmentation of dynamic meshes. Next, we present a new deformation-based feature descriptor in Chapter 3. By using this new descriptor, in Chapter 1, we present our temporal segmentation method for deforming meshes. After that, in Chapter 5, we present a new spatio-temporal segmentation for deforming meshes, which is further extended for measuring motion similarities among deforming meshes. Finally, we give concluding remarks of our works and discuss about the potential future works in Chapter 6.

Chapter 2 State of the Art

In this chapter, we explore the related literatures on 3-Dimension (3D) mesh segmentation and discuss about the background of this thesis. We first introduce the most popular standalone segmentation techniques for segmenting a single static mesh in Section 2.1. Then, in Section 2.2, we present a review of data-driven segmentation techniques that either learn feature patterns from human-based ground-truth data sets or extract common features from a set of similar shapes. After that, we summarize the existing spatial segmentation techniques for deforming meshes in Section 2.3. Both advantages and disadvantages of the reviewed methods and their possibilities of being applied on deforming meshes are discussed.

2.1 Standalone mesh segmentation

During the last two decades, a large number of standalone segmentation methods have been developed for one single static mesh. In this section, we propose to categorize the mesh segmentation methods based on the characteristics of the methods. Note that we study the most popular methods instead of covering all the related literatures. One may refer to details from the surveys of different segmentation methods and the other categorization criteria. Several surveys and comparative studies of the existing standalone segmentation methods can be found in (Attene et al., 2006b) (Shamir, 2008) (Chen et al., 2009) (Benhabiles et al., 2010) (Lavoué et al., 2012), where Shamir (Shamir, 2008) reviews the most popular standalone segmentation methods, and Attene et al. (Attene et al., 2006b), Chen et al. (Chen et al., 2009), Benhabiles et al. (Benhabiles et al., 2010), and Lavoué et al. (Lavoué et al., 2012) compare and evaluate the performance of different segmentation methods.

2.1.1 Hierarchical clustering based method

Hierarchical clustering methods can be divided into two classes, bottom-up and top-down (*k*-way). For a bottom-up type, the methods start by merging atomic

items, e.g., vertices or triangles, until stopping criteria are reached. On the other hand, for a top-down type, the method starts with a full 3D model and iteratively partition the model into *k* parts until predefined stopping criteria are reached. For both of the hierarchy clustering types, they require two key components, merging / partitioning rules and stopping criteria.

Bottom-up methods Attene et al. (Attene et al., 2006a) propose a bottom-up hierarchical clustering method for mesh segmentation. By taking each triangle as a tree leaf, this algorithm iteratively merges neighbouring leaves if they together fit well to a primitive, i.e., either a plane, a sphere, or a cylinder. The merging process is stopped until the fitting costs of all the possible merging exceed a predefined threshold, and the segmentation results are immediately obtained, which are the merged clusters. Similarly, Gelfand and Guibas (Gelfand and Guibas, 2004) present a bottom-up hierarchical clustering method to segment a given mesh into simple geometric parts that each part can be attained by a set of rigid transformations from primitive shapes. By starting with atomic items, this algorithm iteratively aggregate neighbouring regions so that the aggregated region fits well to one of the primitive shapes. One obvious limitation of these methods is that the results are biased to primitive shapes. For the same reason, the primitive-fitting based methods are most applicable to engineering models, which normally contain regular primitive shapes. However, oftenly meshes may contain irregular shapes.

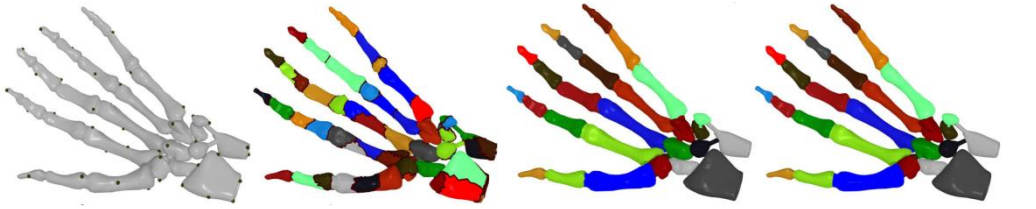


Figure 2-1 Hierachical clustering based mesh segmentation (Lai et al., 2008).

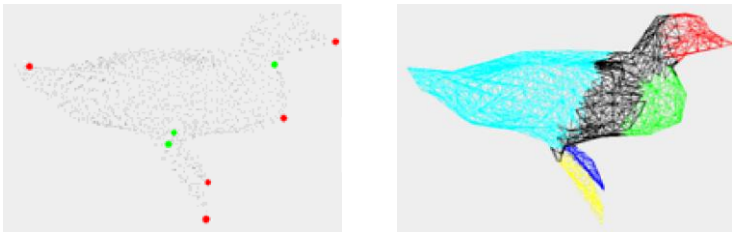
Lai et al. (Lai et al., 2008) present a hierarchical clustering method based on *K* automatically selected seed vertices that distribute on mesh. See the seed points in Figure 2-1(a). For each of the seed vertex, this method iteratively merges the neighbouring vertices with highest probability to reach, where the probability is derived based on the dihedral angles of the edges on the shortest path between the candidate vertex and the seed. After obtaining *K* segments until all the vertices are covered Figure 2-1(b), the merging process continues among the *K* segments until satisfactory number of segments is reached, Figure 2-1(c). Finally, the authors improve the segmentation result by smoothing the boundaries, Figure 2-1(d).

Top-down methods Contrarily to the above methods, Lai et al. (Lai et al., 2006) propose a *k*-way top-down approach that iteratively clustering a mesh region into *k* parts, by starting from a full 3D shape. For each clustering, K-means clustering method (Shlafman et al., 2002) is used and the pairwise vertex distance is measured by incorporating geodesic distance, curvature and texture difference. In an earlier work, Sheffer (Sheffer, 2001) propose another top-down hierarchical clustering method for handling the same problem. Sheffer formulates the mesh segmentation problem as a graph contraction problem: He first represents a mesh into a graph, where a node is a vertex/triangle or a surface region, and an edge denotes the connectivity. Then, in the partitioning process, the algorithm contracts the graph by taking into account of surface region size and region smoothness. Katz and Tal (Katz and Tal, 2003), Shapira et al. (Shapira et al., 2008) have also adopted the *k*-way top-down approach for addressing the mesh segmentation problem.

2.1.2 Region-growing based methods

Given the topology of a mesh, a region-growing based segmentation method starts with seed points, i.e., vertices or triangles, and then iteratively merges neighbouring points along the topology until pre-defined criteria are satisfied. Region-growing based methods have two critical characteristics: seed-point selection and the criteria for quitting the region-growing process.

Zhou and Huang (Zhou and Huang, 2004) propose a region-growing based segmentation method for polygon meshes. In this algorithm, a user supplies a root point, which is used to detect critical points : A critical point is a point whose geodesic distance to the root point is greater than those of its neighbouring points. See the red points in Figure 2-2(a). By taking the critical points as seed points, the segmentation is done by flooding from each seed point until reaching a point has geodesic distance greater than those of its neighbouring points, i.e., the same criteria as critical point detection. The stopping points are shown in green in Figure 2-2(a) and the final segmentation result is shown in Figure 2-2(b). One obvious limitation of this method is the dependency on human selected root point.



(a) Feature point detection (b) Segmentation result

Figure 2-2 Region-growing based mesh segmentation (Zhou and Huang, 2004).

More commonly, region-growing based methods randomly select an un-merged vertex / triangle to merge neighbouring vertices/triangles for a new segment. The following merging criteria have been used: the neighbouring vertices/triangles have small angular differences (Garland et al., 2001), similar curvatures (Lavoué et al., 2005), or the same number of intrinsic numbers (Lee et al., 2005).

The efficiency of region-growing based methods can benefit from the parallel computation of multiple regions growing independently, however this type of methods may result in over segmentation due to local noises.

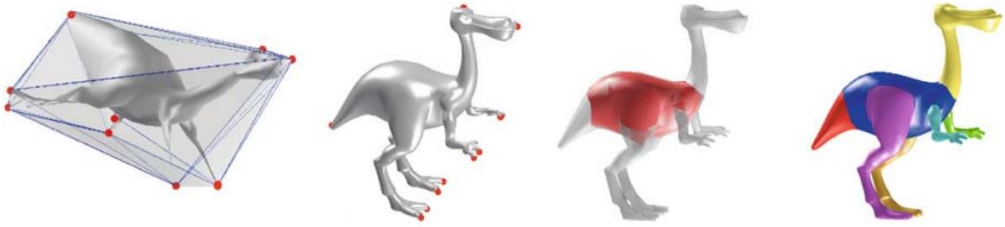
2.1.3 Spectral embedding based method

Due to the high dimensionality of mesh data, i.e., large number of vertices, a variety of previous works map mesh data into lower dimensionality space by using spectral embedding, so that classical clustering techniques such as K-means clustering and/or Mean-shift clustering techniques can be applied for segmentation. The following are two popular spectral embedding techniques:

- Principal Component Analysis (PCA) (Abdi and Lynne, 2010). The purpose of this method is to transform the high dimensionality data into a set of new basis (Principal Components) with eigenvectors where the coordinates are linearly uncorrelated. The number of Principal Components (PCs) is lower or equal to the original dimension, which is determined by the amount of information to be retained in the embedded space.

- Multi-Dimensional Scaling (MDS) (Borg and Patrick, 2005). Based on a pairwise distance matrix, the purpose of MDS is to map the points from high dimensional space into a lower dimensional space without too much loss of information of pairwise point distances.

In (Katz et al., 2005), based on a dual geodesic distance matrix among vertices, the authors apply MDS to map vertices into lower space, and extract prominent feature points that are farthest to the other vertices. These feature points are usually located around limb tips. Then the authors merge the feature points close to each other (see the feature points on leg tips and head in Figure 2-3(b)), and extract the mesh core that disconnects the feature points on limbs, Figure 2-3(c), which immediately results in a mesh segmentation, Figure 2-3(d). However, this method is limited to the mesh of star graph skeleton, i.e., a trunk with several limbs. Liu and Zhang (Liu and Zhang, 2004) propose a similar spectral embedding based segmentation method. In a later work by the same authors (Liu and Zhang, 2007), they improve the previous method by incorporating vertex geodesic distance with geometrical segmentability, which relates to the degree of concavity.



(a) MDS transform (b) Feature points (c) Core extraction (d) Segmentation result

Figure 2-3 Spectral embedding based mesh segmentation (Katz et al., 2005).

2.1.4 The other standalone mesh segmentation techniques

Watershed algorithm has been widely utilized in the Computer Vision domain, particularly for 2D image segmentation. The key idea of the algorithm is to transform an object, an image or a mesh, by defining a height function over points, and then segment the object into catchment basins. Several mesh segmentation techniques have been devised based on watershed algorithm (Mangan and Whitaker, 1999) (Chen and Georganas, 2006). As a classic example in an early work by Mangan and Whitaker (Mangan and Whitaker, 1999), the mesh segmentation method defines a height function of each vertex based on curvature, i.e., a vertex with higher curvature has larger height, and vice-versa. By applying watershed algorithm, the segmentation results tend to lay the segment boundaries along sharp surface area with high curvatures. The drawback of the watershed algorithm is over segmentation, due to noises on mesh surface.

Some segmentation methods involve human interactions (Fan and Liu, 2011) (Li et al., 2001) (Chen et al., 2009). In order to collect ground-truth segmentation from

human users, Chen et al. (Chen et al., 2009) develop an online system so that a user can pick vertices on mesh surface, which are automatically connected as boundary lines for mesh segmentation. As shown in Figure 2-4, human tend to consistently place segmentation boundaries in concave regions. Fan and Liu (Fan and Liu, 2011) propose a mesh segmentation method based on human painting. In this method, a user paint strokes on mesh to indicate the number and area of each mesh segment. Then, the segmentation is done by grouping vertices to the closest stroke, where the distance is measured by incorporating both Gaussian curvature (Yamauchi et al., 2005) and Shape Diameter Function (Gal et al., 2007). Although this type of methods benefit from human perception, they are not practical for processing massive data sets.

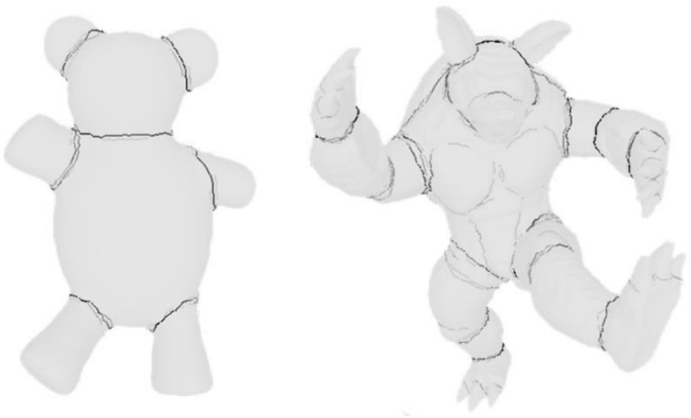


Figure 2-4 Human-based segmentation boundaries (Chen et al., 2009).

Golovinskiy and Funkhouser (Golovinskiy and Funkhouser, 2008) propose a segmentation method by integrating the segmentation results by using different methods. They first generate a set of over segmentation results by using existing segmentation methods, K-Means (Shlafman et al., 2002), Hierachical Clustering (Shi and Jitendra, 2000) and Min Cuts (Katz and Ayellet, 2003), each with randomly generated parameters to segment an input mesh. Along the boundaries of these segmentation results, the authors construct a statistical framework of edges in the boundaries. Finally, they determine the final boundaries by taking the edges appear most frequently in the boundary lines. However, this method suffers from heavy computation because of computing mesh segmentation several times of one model.

2.1.5 Discussions on the standalone mesh segmentation techniques

As shown in the above review, a large variety of techniques have been proposed for standalone mesh segmentation, either for segmenting a mesh into meaningful

parts or geometrical rigid segments. Thanks to the recent contributions by Giorgi et al. (Giorgi et al., 2007), Chen et al. (Chen et al., 2009), Benhabiles et al. (Benhabiles et al., 2009), Bronstein et al. (Bronstein et al., 2010), Kim et al. (Kim et al., 2011), and Lavoué et al. (Lavoué et al., 2012), they have not only made available of a considerable amount of static mesh sets, but also created ground-truth segmentations, which enable us to objectively evaluate segmentation results.

Object Category	[Golovinskiy and Funkhouser '08] Rand Cuts	[Shapira et al '08] Shape Diam	[Golovinskiy and Funkhouser '08] Norm Cuts	[Katz et al '05] Core Extra	[Lai et al '08] Rand Walks	[Attene et al '06] Fit Prim	[Shlafmanet al '02] K-Median
Human	1	5	2	7	6	3	4
Cup	1	5	2	3	4	6	7
Glasses	1	4	2	6	7	5	3
Airplane	2	1	4	7	6	3	5
Ant	2	1	3	4	5	6	7
Chair	4	2	1	5	3	6	7
Octopus	4	1	3	2	5	7	6
Table	7	4	1	5	2	3	6
Teddy	1	2	4	3	5	6	7
Hand	1	7	3	4	5	6	2
Plier	2	7	4	1	5	3	6
Fish	3	1	5	2	4	7	6
Bird	1	2	4	3	7	6	5
Armadillo	3	1	5	7	4	2	6
Bust	1	3	6	5	2	4	7
Mech	4	3	1	6	2	5	7
Bearing	2	1	3	7	5	4	6
Vase	1	4	3	2	5	6	7
FourLeg	2	1	6	4	7	3	5
Overall	1	3	2	5	6	4	7

Figure 2-5 Comparison between different segmentation methods, where the numbers are the rankings of the performance by using different methods on each dataset (Chen et al., 2009).

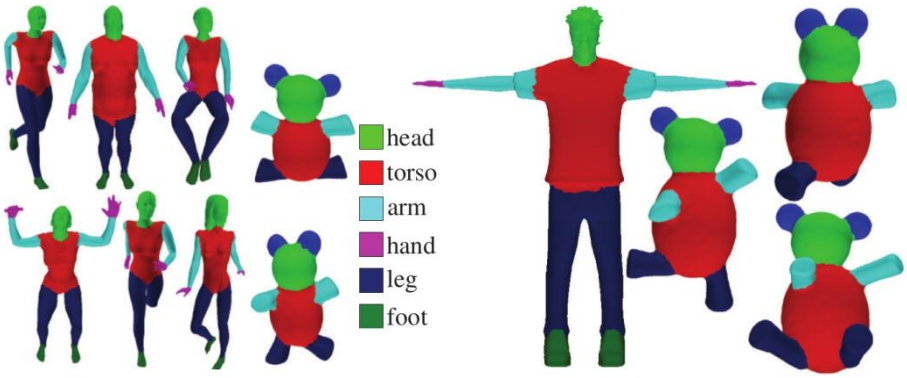
In order to compare a segmentation result against ground-truth, we can apply different distance metrics such as Cut Discrepancy (Huang and Dom, 1995), Hamming Distance (Huang and Byron, 1995), Rand Index (Rand, 1971), and Consistency Error (Martin et al., 2001). By using these distance metrics, Chen et al. (Chen et al., 2009) evaluate the performance of several most advanced segmentation techniques at the time. According to their experimental results, shown in Figure 2-5, none of the standalone segmentation methods performs well over all mesh types, because geometric criteria may not provide sufficient cues to identify all the semantically meaningful parts. This observation complies well with the conclusion of an earlier comparative study of standalone segmentation techniques by Attene et al. (Attene et al., 2006b) that each mesh segmentation method has its advantages and drawbacks. Due to this reason, the later works on mesh segmentation have been mostly devoted in data-driven approaches.

2.2 Data-driven mesh segmentation

More recently, researchers in the Computer Graphics community have proposed a variety of data-driven segmentation methods, aiming at the consistent segmentation of a set of 3D shapes. We categorize these data-driven methods into two classes: learning-based and co-segmentation.

2.2.1 Learning-based methods

A learning-based segmentation method learns the prominent features from a set of ground-truth segmentations at the training stage. Then the segmentation of input meshes can be achieved by detecting the learned features in the meshes.



(a) Training meshes (b) Test meshes

Figure 2-6 A learning-based mesh segmentation (Kalogerakis et al., 2010).

Kalogerakis et al. (Kalogerakis et al., 2010) propose a learning-based method for 3d mesh segmentation and labelling. The key idea of this method is a classifier that determines the label of a triangle based on the triangle's features. The classifier is a linear combination of two terms: a unary term that measures the probability for a triangle being assigned with each of the candidate label, and a binary term that measures the probability of two neighbouring triangles being assigned with different labels. Given this definition of the classifier, the algorithm trains the parameters for the classifier to adapt to human ground-truth segmentation. Finally, the segmentation of a new mesh is to feed with triangle features to the trained classifier, which generates the labelling of each triangle, i.e., the spatial segmentation of the input mesh. Several sample segmentation results are shown in Figure 2-6**(b)**.

In fact, a 3D mesh segmentation can be defined not only as segment patches of vertices or triangles, but also boundaries that cut a mesh. By following the latter definition, Benhabiles et al. (Benhabiles et al., 2011) propose a learning-based

method that addresses the mesh segmentation problem by computing boundaries. Similar to (Kalogerakis et al., 2010), Benhabiles et al. (Benhabiles et al., 2011) train a classifier based on ground-truth segmentation, to measure the possibility of an edge appearing in boundaries. By using a threshold, this method binary labels the edges with 'boundary edge' or not, which results in mesh regions with connected 'boundary edges'. For each of the 'boundary edge' region, the authors apply a thinning approach (Hubeli and Gross, 2001) to remove the border edges towards inside, which produces a boundary line. Finally, they apply snake movement alignment method (Jung and Kim, 2004) to connect head and tail of the boundary lines for the mesh segmentation.

In a recent work, Wang et al. (Wang et al., 2013) propose a method that tunes the segmentation of 2D images to 3D shapes. Depicted in Figure 2-7, this method first averagely projects a given upright 3D shape into 2D images from different rotation angles, Figure 2-7(b). In the meantime, they maintain a set of similar 2D images with ground-truth segmentations, Figure 2-7(c). The labelling of each projected 2D images can be inferred from closely matched labelled images from ground-truth set by using 2D image segmentation techniques, Figure 2-7(d). Finally, the segmentation is done by back projecting the image labelling to triangles on the original 3D shape, Figure 2-7(e,f). Although this method simplifies the challenge of mesh segmentation by working on lower dimensional data, the method has to match each 2D image projection with each ground-truth image segmentation from database and thus suffers from heavy computation.

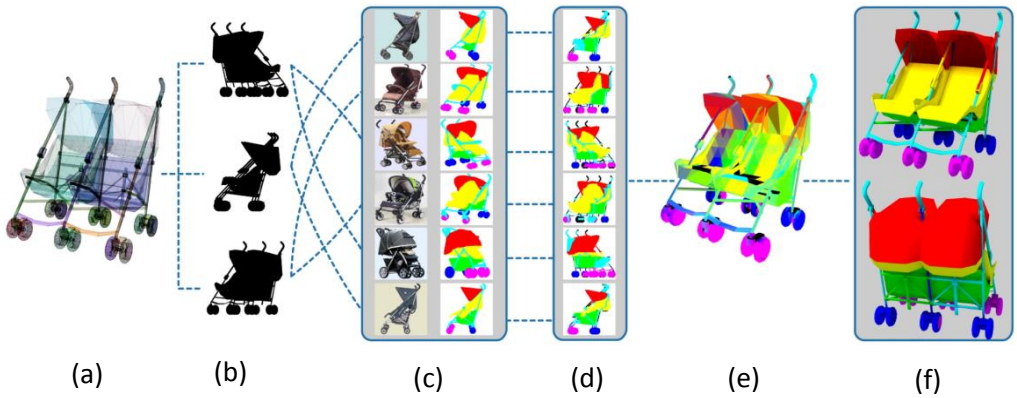


Figure 2-7 Mesh segmentation derived from 2D image segmentation (Wang et al., 2013): (a) original mesh, (b) 2D projections, (c) best matched ground-truth segmentation of each 2D projection, (d) segmentation of each projected image, (e,f) mesh segmentation.

Therefore, the learning-based methods can learn prominent features based on large sets of ground-truth segmentation that are available from (Chen et al., 2009)

(Benhabiles et al., 2009) (Bronstein et al., 2010). However, learning-based segmentation methods suffer from heavy computation, and not capable for treating meshes without ground-truth segmentation of similar models.

2.2.2 Co-segmentation methods

Another solution for the consistent segmentation problem is to extract commonly shared features from a set of similar shapes, and then segment meshes into consistent parts based on the same criteria.

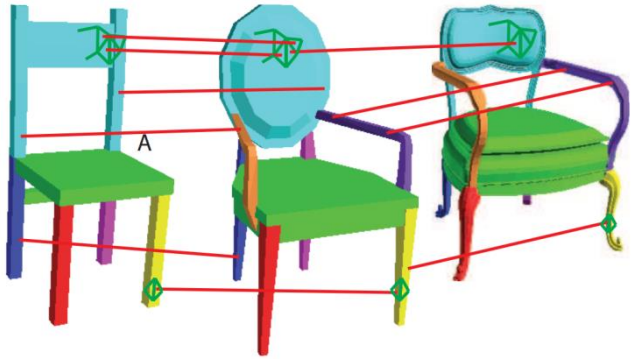


Figure 2-8 Consistent segmentation of a set of similar 3D models (Golovinskiy and Funkhouser, 2009).

Golovinskiy and Funkhouser (Golovinskiy and Funkhouser, 2009) propose an unsupervised segmentation method for segmenting a set of similar 3D models at the same time. They first define two edge types: adjacency edge (Green edges in Figure 2-8) and correspondence edge (Red edges in Figure 2-8). The adjacency edges connect neighbouring triangles within one mesh, where each edge is weighted by both the dihedral angle and the edge length. And the correspondence edges connect corresponding triangles between two models, where the correspondence is obtained from model alignment. The segmentation is then done by minimizing the sum of within segment adjacency edge weights and inter-model correspondence edge distance. This method allows to obtain consistent segmentation because the adjacency edges encourage smooth surface within segment and the correspondence edges encourage corresponding parts between two meshes to be aggregated into the same segment. Similarly, the segmentation methods by Sharma et al. (Sharma et al., 2010) and Kim et al. (Kim et al. 2013) both are also based on shape matching. In specific, they first generate the mesh segmentation of one shape, and then compute the shape correspondence with another shape for transferring the segmentation labelling to the shape.

Among the co-segmentation methods, a large portion make use of the over segmentation of each model independently (Hu et al., 2012) (Sidi et al., 2011) (Huang et al., 2011) (Wu et al., 2013) (Luo et al., 2013). As shown in Figure 2-9, Sidi et al. (Sidi et al., 2011) first apply mean-shift clustering (Chen, 1995) to segment each model into patches, Figure 2-9(a). Then they apply spectral analysis for each patch and classify the patches in their embedded space, in the object of grouping similar parts from different meshes into the same cluster, Figure 2-9(b,c). Finally, the segmentation is done by minimizing within cluster triangle energy and maximizing pairwise triangle energy along boundaries, which also optimizes the segmentation boundaries, Figure 2-9(d,e). With a similar segmentation framework, Huang et al. (Huang et al., 2011) first conduct an over segmentation of two similar shapes independently by using randomized cuts segmentation method (Golovinskiy and Funkhouser, 2008). Then they define a consistency term that both measures the geometric similarity between individual corresponding segments, and prioritizes adjacent segments in different shapes to have the same labelling. Therefore, the optimization of the consistency term allows to obtain consistent segmentation between similar shapes. Hu et al. (Hu et al., 2012), Luo et al. (Luo et al., 2013) and Wu et al. (Wu et al., 2013) also apply the same segmentation framework, but differently these approaches incorporate multiple features within each patch to improve the performance of the segmentation: Wu et al. automatically compute the optimized weight among multiple features through spectral analysis; Luo et al. iteratively project the affinity matrices of different features onto the same Laplacian eigenvectors (Chung, 1997), then they concatenate different features into one vector and apply k-means to classify patches.

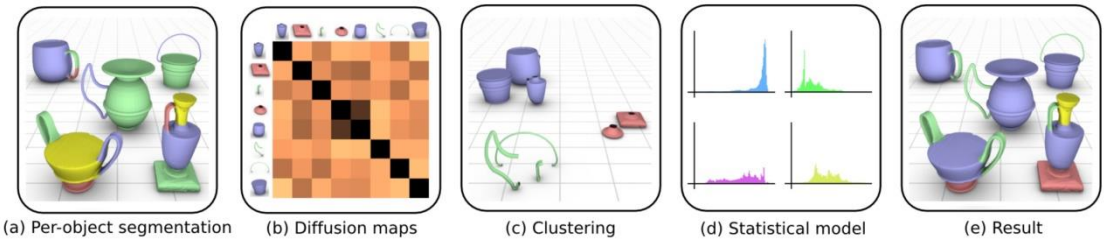


Figure 2-9 Co-segmentation of a set of similar meshes (Sidi et al., 2011).

The advantage of this type of method is that a set of similar shapes help to identify commonly shared prominent features. For this reason, not only a set of shapes can be segmented at the same time, but also the individual segmentation can be improved. However, the requirement of similar shapes poses a limitation on the input data. Moreover, most of the above methods also suffer from heavy computation,

for which reason these segmentation methods may not be applicable for dynamic meshes.

2.3 Segmentation of dynamic meshes

Since a dynamic mesh is normally a heavy data that contains an abundance of redundancy both in local mesh regions and between successive frames, segmentation provides a possibility to precede this heavy data for the applications such as compressions (Sattler et al., 2005) (Karni and Gotsman, 2004) (James and Twigg, 2005) (Mamou et al., 2006), skeleton extractions (James and Twigg, 2005) (De et al., 2008), etc. During the last decade, a variety of spatial segmentation methods have been proposed for dynamic meshes. In this section, we categorize these methods based on the criteria whether an approach uses vertex trajectories, or extracted high-level local feature descriptors.

2.3.1 Trajectory-based method

A trajectory-based mesh segmentation method treats a deforming mesh directly on coordination data, without pre-processing.

Figure 2-10 Trajectory-based spatial segmentation of a deforming mesh (Sattler et al., 2005).

Sattler et al. (Sattler et al., 2005) propose a segmentation method of deforming meshes by grouping vertices with similar trajectories. They first compute an initial clustering of vertex trajectories by using K-Means clustering, where pairwise vertex trajectory distance is computed by using Euclidean distance. Then, iteratively, they compute the number of Principle Components (PCs) of each cluster, and classify each vertex to the cluster with least reconstruction error by using the corresponding number of PCs. Now that the vertices with similar trajectories are classified into the same cluster, optimized number of PCs can be used to represent the vertices within each cluster, and therefore improves the compression rate. Figure 2-10 shows an example of spatial segmentation result of a chicken animation by using Sattler et al.'s approach, where the mesh is partitioned into rigid segments. Similar-

ly, De et al. (De et al., 2008) propose a segmentation method for automatic conversion of deforming meshes into skeleton animations by using spectral clustering. To this end, they define an affinity matrix among vertices where pairwise vertex affinity is derived from Euclidean distance of vertex trajectories. Having the spatial segmentation results, they can build a skeleton for each frame where each node represents the corresponding segment center and each edge represents neighbourhood between two segments.

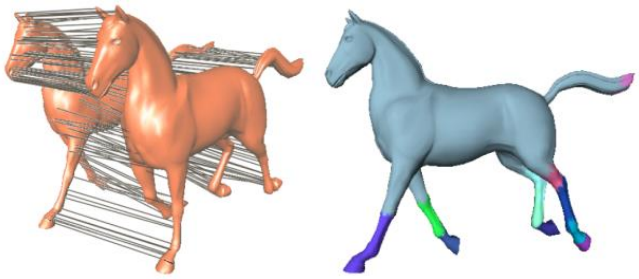
Lee et al. (Lee et al., 2005) propose a region-growing based segmentation method for animation sequences that groups triangles with the same intrinsic dimensionality. Started from a user supplied seed triangle, this method iteratively merges the neighbouring triangles that have the same number of PCs and the distance to the neighbouring triangle within cluster does not exceed a user threshold. The local greedy region-growing process continues until all the triangles have been clustered.

Since the above segmentation methods in (Sattler et al., 2005), (Lee et al., 2005) and (De et al., 2008) group vertices with similar trajectories, they allow to obtain the segmentation results that keep the motion rigidity within each segment.

2.3.2 Nontrajectory-based method

In fact, most of the segmentation methods have been developed based on local feature descriptors.

Arcilla et al. (Arcilla et al., 2010 and 2013) present a framework for the spatial segmentation of dynamic mesh with inconsistent topology. Their method first computes the vertex matching between two successive meshes, Figure 2-11(a), then groups the neighbouring vertices with similar motions, which is measured by the displacement vector for each corresponding vertex. In fact, both the 3D shape correspondence (Van et al., 2011) and segmentation are challenging problems for the research in Computer Graphics. We focus on the latter problem and therefore work with deforming meshes that the mesh topology is fixed.



(a) Vertex matching (b) Segmentation result

Figure 2-11 Spatial segmentation of a dynamic mesh based on shape correspondence. (Arcilla et al., 2010 and 2013)

Lee et al. (Lee et al., 2006) propose a region-growing based segmentation method for deforming meshes, as shown in Figure 2-12. They first binarily label each triangle with 'deformed' / 'rigid' based on the degree of deformation (Figure 2-12 (b)), which is measured by the maximal degree of transformation from the rest pose (Figure 2-12(a)). Then they extract the seed triangle in each rigid region that is farthest to the neighbouring 'deformed' regions (Figure 2-12 (c)). Finally, the segmentation is done by grouping triangles to seed triangles so that minimizes the total sum of within cluster triangle distances (Figure 2-12 (d,e)), where the pairwise triangle distance incorporates both the geodesic distance and the difference of deformation. Similarly, Kalafatlar and Yemez (Kalafatlar and Yemez, 2010) propose a segmentation method by minimizing the within-segment pairwise vertex distance, which incorporates the geodesic distance, vertex angular difference and the Euclidean distance in the rest pose.

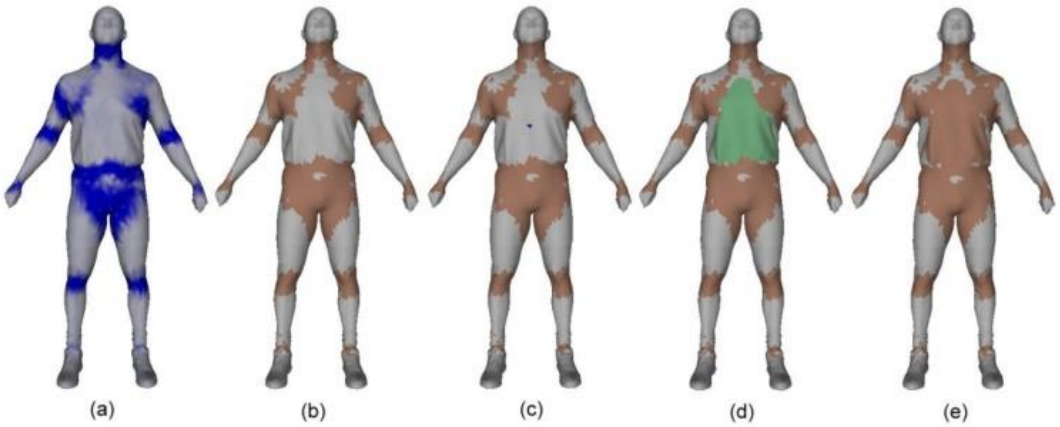


Figure 2-12 Region-growing based spatial segmentation of a deforming mesh (Lee et al., 2006). (a) degree of deformation of triangles, (b) binary labels, (c) a seed point in a rigid region, (d,e) region-growing.

Wuhrer and Brunton (Wuhrer and Brunton, 2010) achieve the same goal in a more efficient approach by using minimum spanning tree (MST) (Kleinberg and Tardos, 2006) in dual space. They first compute a dual graph wherein nodes are vertices and edges are vertex neighbourhood weighted by the maximum dihedral angle in the sequence. After computing the MST of the dual graph, the algorithm removes the edges with largest weight, which most likely indicate two dissimilar subgraphs. The graph cut process stops until a predefined number of segments is obtained.

The spatial segmentation of deforming meshes has also been addressed through clustering techniques. James and Twigg (James and Twigg, 2005) segments a deforming mesh by applying mean-shift clustering (Chen, 1995) to vertex rotation sequences, which contains the rotation matrix of each triangle in each frame with referring to the corresponding triangle in the rest pose. Similarly, Mamou et al. (Mamou et al., 2006) compute the triangle transformation sequence, which is the affine transformation of each triangle in each frame. Then they acquire the segmentation by applying K-means clustering to group the vertices with similar deformation behaviours. These two segmentation results have been applied for the skinning (James and Twigg, 2005) and the compression (Mamou et al., 2006) of deforming meshes, respectively.

Figure 2-13 Triangle rotation sequence (James and Twigg, 2005).

In one of the most recent literature, Vasilakis and Fudos (Vasilakis and Fudos, 2014) present a data-driven segmentation method for deforming meshes. This algorithm first applies existing static mesh segmentation methods to each pose in the mesh sequence, Figure 2-14(a), and then compute an over segmentation by intersecting all the spatial segments, Figure 2-14(b). Finally, for each frame, the method simplifies the over-segmentation by merging small neighbouring segments based on dynamic behaviours, which results in the final segmentation result, Figure 2-14(c).

(a) Per-frame segmentation.

(b) Over segmentation through intersection.

(c) Per-frame simplification.

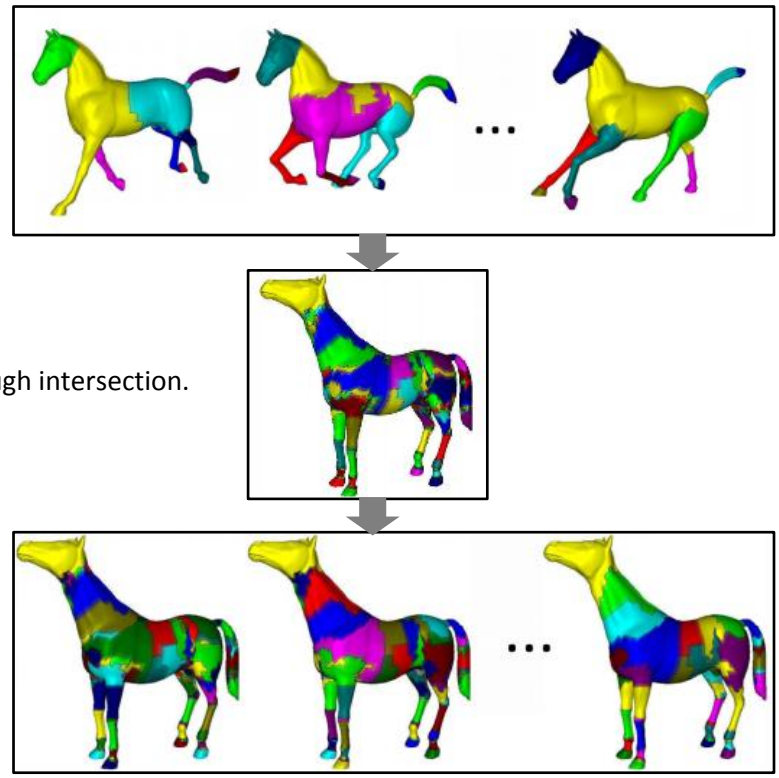


Figure 2-14 Spatial segmentation of deforming meshes based on pose partitioning (Vasilakis and Fudos, 2014).

Nonetheless, a long mesh sequence may result in different segmentation results because the existing methods do not consider the temporal coherency within a deforming mesh.

2.4 Summary

Mesh segmentation for both 3D static meshes and dynamic meshes has been an active research area during the last two decades. In this section, we have summarized the existing spatial segmentation techniques and categorized them based on their characteristics. See Table 2-1 in the next page. However, to our best knowledge, the existing segmentation methods for dynamic meshes only concerns about spatial deformation coherency without considering about temporal deformation coherency. In the remainder of this thesis, we present new segmentation methods that investigate both temporal and spatial coherency within deforming meshes, which have not studied before.

Table 2-1 Summary of 3D mesh segmentation techniques.

Mesh types	Segmentation types	Segmentation method type	Object	Drawback
Static mesh	Standalone segmentation	Hierarchical clustering, Region-growing, Spectral embedding, etc.	Segmenting a mesh by using the local geometrical cues in the mesh.	None of the methods can perform well over all mesh types.
	Data-driven segmentation	Learning-based	Segmenting a mesh by using the prominent features learned from ground-truth.	Requires human-based ground-truth, and time consuming.
		Co-segmentation	Segmenting a set of similar shapes by extracting commonly shared features.	Requires a set of similar shapes, and time consuming.
Dynamic mesh	Spatial segmentation	Trajectory-based	Grouping vertices with similar trajectories.	Overlooks the temporal deformation coherency in dynamic meshes.
		Non-trajectory based	Segmenting a dynamic mesh into near-rigid parts.	

Chapter 3 A Dynamic Feature Descriptor

In this chapter, we first review existing feature descriptors for 3D meshes in Section 3.1. Then, we present a deformation-based feature descriptor, named *strain*, to measure the degree of deformation of each triangle at each frame of deforming meshes. This descriptor is invariant to global shape rotation, translation and uniform scale, and its dynamic feature is robust over shape difference when different shapes undergoing identical or similar motions. Based on this new descriptor, we develop different segmentation methods for deforming meshes, as described in Chapter 1 and Chapter 5.

3.1 Existing 3D feature descriptors

As have been reviewed in Chapter 2, although there are several mesh segmentation methods directly make use of vertex coordinates, most of the methods use high-level feature descriptors. In this section, we give a review of related literatures about 3D feature descriptors. There are many ways to categorize the existing feature descriptors. In this thesis, we classify the descriptors for static meshes as *static descriptors* and the descriptors for dynamic meshes as *dynamic descriptors*.

3.1.1 Static descriptor

Many static descriptors have been proposed in the contexts of 3D shape correspondence and segmentation. We classify the existing static descriptors into *global descriptors* and *local descriptors*, based on the criterion whether a descriptor is representing a segment patch, or a mesh primitive, i.e., vertex or triangle.

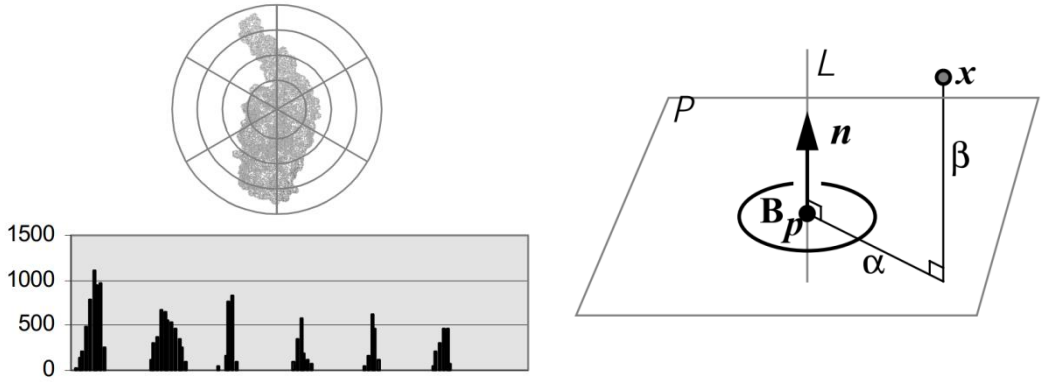
A *global descriptor* illustrates the overall properties of a segment patch, which can be an entire shape or a patch of mesh primitives. Several commonly used global descriptors are listed as follows.

- *Shape histogram.* Ankerst et al. (Ankerst et al., 1999) propose this global descriptor to represent a 3D object. This descriptor first decomposes an object into concentric shell bins and sector bins around the center point, then represent the object into a vector of number of points within each bin. Figure 3-1(a) shows an example of representing a 2D image with this descriptor, where the top row shows an image being divided into shell bins and sector bins, and the number of points in each bin is shown in the bottom row as a histogram. However, the *shape histogram* representations for one object may be different depending on the bin orders. In order to avoid such problem, based on the representation of 3D shapes with *shape histogram*, Ankerst et al. (Ankerst et al., 1999) measure shape similarities by using Quadratic Form Distance Function (Ankerst et al., 1998), which is invariant to the order of bins.

- *Spherical harmonics.* Kazhdan et al. (Kazhdan et al., 2003) present spherical harmonics based on a set of spherical basis functions. In specific, they first divide the object volume into a set of concentric shells with different radius, and then decompose each shell in frequency domain. Then, the norm for each frequency component at each radius becomes the bin size of a 2D histogram, which is indexed by radius and frequency.

- *Spin Images* (Johnson, 1997). To compute the spin images of a shape, we first map 3D vertices on surface into 2D space via the cylindrical coordinate system. That is, as shown in Figure 3-1(b), given a tangent plane *P* and an oriental point $\boldsymbol{p}$, each vertex $\boldsymbol{x}$ has a 2D basis representation (α, β), where α is the perpendicular distance to normal $\boldsymbol{n}$ and β is the perpendicular distance to *P*. Johnson and Hebert (Johnson and Hebert, 1997) have applied this descriptor to find matching points between two shapes by using iterative closest point algorithm (Zhang, 1994).

- Other global descriptors. After the over segmentation of a mesh, Huang et al. (Huang et al., 2011) represent each mesh segment with the percentage of surface area of the full mesh. More commonly, global descriptors are also devised as statistical histogram of local geometries of mesh primitives within a mesh segment (Sidi et al., 2011).



(a) (b)

Figure 3-1 Global descriptors, (a) shape histogram (Ankerst et al., 1999), (b) spin images (Johnson and Hebert, 1997).

A *local descriptor* describes the local geometric properties of a mesh primitive. Unlike global descriptors which are relatively coarse to characterize an entire shape, local descriptors carry more detail information and therefore can be more distinctively describing each vertex or triangle (Huang et al., 2007). Among different geometrical information, geodesic distance has been widely used for local descriptors (Wang et al., 2000) (Gal et al., 2007) (Lee et al., 2006). Below we introduce several other commonly used local descriptors for 3D shapes.

- *Average Geodesic Distance* (AGD) (Gal et al., 2007). AGD of a vertex is the average of the geodesic distance from this vertex to all the other vertices on mesh. This descriptor is useful for detecting extreme points that are farthest to the other points on the mesh.

- *Gaussian Curvature* (Yamauchi et al., 2005) (Lavoué et al., 2005). A curvature value measures the surface convexity, or concavity, which has been commonly used for mesh segmentation (Yamauchi et al., 2005) (Kalogerakis et al., 2010). Let κ_1 and κ_2 be the principal curvatures of a point on surface, the well-known Gaussian curvature is the product of the two principal curvatures $K = \kappa_1\kappa_2$. One can also derive the other curvature types by incorporating the two principle curvatures in different ways, including κ_1, $|\kappa_1|$, κ_2, $|\kappa_2|$, $|\kappa_1\kappa_2|$, $(\kappa_1+\kappa_2)/2$, $|(\kappa_1+\kappa_2)/2|$, $\kappa_1-\kappa_2$, etc. (Kalogerakis et al., 2010). Wang et al. (Wang et al., 2000) devise a vertex distance metric by incorporating curvature difference and geodesic distance between two vertices. Figure 3-2(a) shows the Gaussian curvature field of a 3D plane model, where the surface regions with higher convexity have higher curvature values, shown in red color, otherwise in blue.

- *Shape Diameter Function* (SDF) (Gal et al., 2007). Figure 3-2(b) shows an example of computing SDF of a vertex on head. For computing the SDF of a vertex, rays from the vertex are averagely sampled in a cone, which is opposite to the normal of the vertex. These rays intersect on the shape, and the local SDF is the average length of the ray segments. Gal et al. linearly combine AGD and SDF as a vertex signature, for computing vertex correspondence between two shapes by measuring the similarity of vertex signatures.

- *Shape Context* (Belongie et al., 2000). As depicted with a 2D image in Figure 3-2(c), for each point, this descriptor is a 2D histogram measuring the distribution of the other points in logarithmic distance bins and uniform angle bins, see the right matrix in Figure 3-2(c). Kalogerakis et al. (Kalogerakis et al., 2010) and Acosta et al. (Acosta et al., 2010) compute shape correspondence by comparing *shape context* of vertices.

- *Heat Kernel Signature* (HKS) (Sun et al., 2009). Sun et al. devise an intrinsic local descriptor HKS, which measures heat kernel values at multiple time scales for each vertex. HKS inherits isometric invariance and multi-scale property from the heat kernel (Sun et al., 2009) (Ovsjanikov et al., 2010). Tevs et al. (Tevs et al., 2011) compute vertex matching between two shapes based on the comparisons of the HKS of vertices. Figure 3-2(d) shows an example of HKS field of each point on a 3D model.

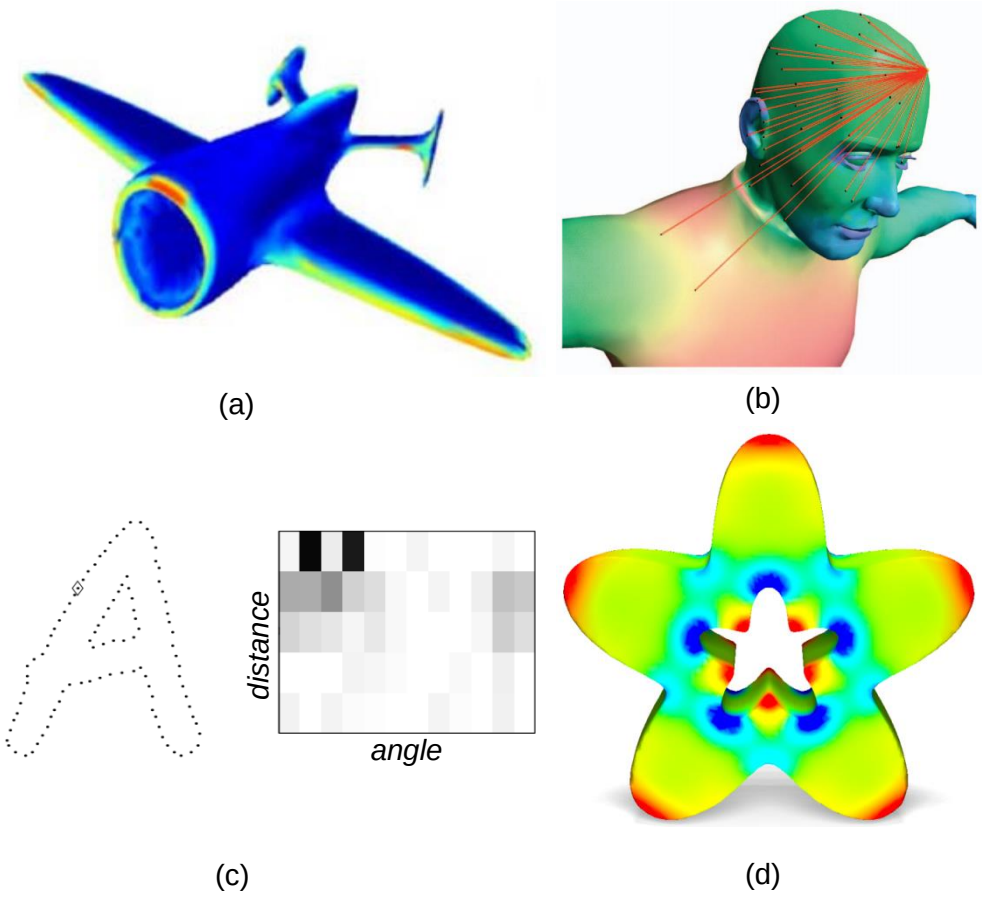


Figure 3-2 Local shape descriptors, (a) curvature (Lavoué et al., 2005), (b) shape diameter function (Gal et al., 2007), (c) shape context (Belongie et al., 2000), and (d) heat kernel signature (Sun et al., 2009).

3.1.2 Dynamic descriptor

A *dynamic descriptor* represents the deformation behaviours along time in an animation, such as the changing curvatures of a vertex due to deformation. Since a deforming mesh is a sequence of static meshes, every vertex/triangle on the shape can be represented with a vector of static shape descriptors, where the length of the vector is the number of frames in the sequence. Thus, given a deforming mesh, the dynamic descriptor of a triangle (or a vertex) can be represented as a vector of local descriptors at each frame. In this way, for example, the Gaussian curvature of a vertex varies at each frame and therefore represents the deformation behaviours of the vertex.

So far, most of the existing dynamic descriptors have been developed in the context of segmentation and/or compression of dynamic meshes. With an aim of clustering similarly moving vertices into the same cluster, several related literatures

have used vertex trajectories to measure similarity between vertices (Sattler et al., 2005) (Lee et al., 2005) (De et al., 2008). Apart from these, most of the existing works represent each mesh primitive with dynamic feature descriptors. James and Twigg (James and Twigg, 2005) represent each triangle of a deforming mesh with a rotation sequence, wherein each element is the rotation matrix of a triangle in a frame w.r.t. the corresponding triangle in the rest pose. Similarly, Lee et al. (Lee et al., 2006) compute the transformation of each triangle within each frame, which results in a transformation sequence for each triangle.

Note that a dynamic descriptor can also be derived from global static descriptor. That is, for a deforming mesh, we represent each frame with a global descriptor, and therefore the deforming mesh can be represented with a vector of global descriptors. However, in this thesis, we devise dynamic descriptors by using local static descriptors because local descriptors contain the detailed information of each triangle, with which we will classify into groups based on their deformation behaviors.

Having a variety of feature descriptors available, one may need to clarify the segmentation criteria before choosing descriptors, since different descriptors may lead to different segmentation results. Among the introduced static descriptors in Section 3.1.1, AGD, SDF, HKS and Shape Context are invariant to rigid transformations. Moreover, HKS holds the property of isometric invariance. However, these descriptors are not practical for deforming meshes because AGD is invariant for isometric meshes and therefore cannot represent dynamic behaviors in isometric deforming meshes; Gaussian Curvature represents well of bending but not stretching or shrinking; SDF, HKS and Shape Context may suffer from heavy computation due to their high algorithm complexities and the larger dimensionality of deforming meshes. In the following section, we present an efficient deformation-based dynamic descriptor for deforming meshes, which measures the degree of deformation for each triangle at each frame.

3.2 Our deformation-based feature descriptor

In this section, we present a new efficient dynamic descriptor for deforming meshes. For each triangle within each frame of a deforming mesh, this descriptor computes a scalar value that indicates the degree of deformation, by comparing to the corresponding triangle at a reference pose. In specific, we first introduce deformation gradient tensor, based on which we can compute principal stretches of

each deformed triangle. Then, we use the principal stretches to measure the degree of deformation.

3.2.1 Deformation gradient tensor

The Deformation Gradient Tensor $\mathbf{F}$ measures the degree of deformation in continuum mechanics, which is the derivative of each component of the deformed configuration with respect to each component of the reference configuration (Crandall et al., 1978). Depicted in Figure 3-3, we formulate the deformation gradient tensor between a deformed vector $d\boldsymbol{x}$ and a reference vector $d\boldsymbol{x}'$ as follows:

$$F_{ij} = \frac{\partial x_i}{\partial x'_j} = \begin{bmatrix} \frac{\partial x_1}{\partial x'_1} & \frac{\partial x_1}{\partial x'_2} & \frac{\partial x_1}{\partial x'_3} \\ \frac{\partial x_2}{\partial x'_1} & \frac{\partial x_2}{\partial x'_2} & \frac{\partial x_2}{\partial x'_3} \\ \frac{\partial x_3}{\partial x'_1} & \frac{\partial x_3}{\partial x'_2} & \frac{\partial x_3}{\partial x'_3} \end{bmatrix} \tag{3-1}$$

where $i, j = 1,2,3$.

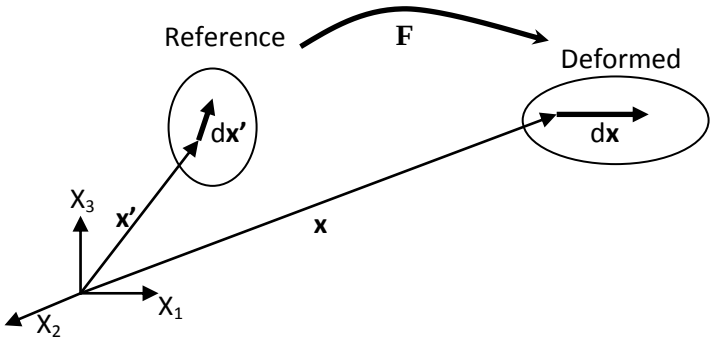


Figure 3-3 The reference and deformed configurations (Gullett et al., 2008).

Invariant to translation. The Deformation Gradient Tensor is invariant to body translation (Gullett et al., 2008). Consider an example of the translation of $\boldsymbol{x}$ to $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{s}$, based on Equation 3-1, we can obtain $\mathbf{F}$ as follows:

$$F_{ij} = \frac{\partial x_i}{\partial x'_j} = \frac{\partial x_i}{\partial (x_i + s)} = \mathbf{I}$$

where $\mathbf{I}$ is an identity matrix. It shows that the body translation $\boldsymbol{s}$ does not affect the deformation gradient.

Cauchy deformation tensors. As have seen above, $\mathbf{F}$ only contains rotation as rigid movement and stretch as deformation. Then, by using the polar decomposition

theorem (Higham, 1986), we can decompose $\mathbf{F}$ into the product of an *orthogonal* rotation tensor $\mathbf{R}$ and a *symmetric* stretch tensor $\mathbf{U}$ or $\mathbf{V}$ (Truesdell and Noll, 2004). Depending on whether we first apply rotation or stretch, we have the two decomposition types, $\mathbf{F} = \mathbf{RU}$ and $\mathbf{F} = \mathbf{VR}$, where $\mathbf{U}$ is right stretch tensor and $\mathbf{V}$ is left stretch tensor (see Figure 3-4). Because of the symmetric property of stretch tensors, we have $\mathbf{U}^{\mathrm{T}} = \mathbf{U}$ and $\mathbf{V}^{\mathrm{T}} = \mathbf{V}$.

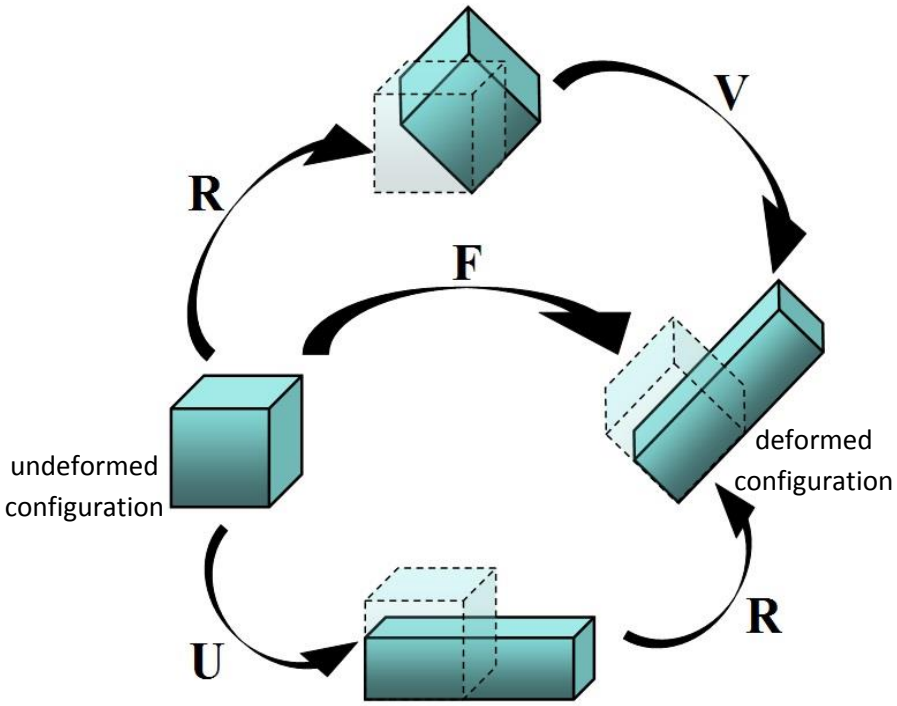


Figure 3-4 Representation of the polar decomposition of the deformation gradient. (Finite strain theory. In Wikipedia. Retrieved November 21st, 2014, from http://en.wikipedia.org /wiki/Finite_strain_theory).

Based on the above properties of deformation gradient tensor, we proceed to introduce the following two Cauchy deformation tensors (Note that the rotation tensor $\mathbf{R}$ is orthogonal, i.e., $\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$ and $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$):

- Right Cauchy deformation tensor

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F} = (\mathbf{RU})^{\mathrm{T}}(\mathbf{RU}) = \mathbf{U}^{\mathrm{T}}(\mathbf{R}^{\mathrm{T}}\mathbf{R})\mathbf{U} = \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{U}^{2}. \tag{3-2}$$

It shows that the right Cauchy deformation tensor is equal to the square of the right stretch tensor, which is invariant to body rotation.

- Left Cauchy deformation tensor

$$\mathbf{B} = \mathbf{F}\mathbf{F}^{\mathrm{T}} = (\mathbf{VR})(\mathbf{VR})^{\mathrm{T}} = \mathbf{V}(\mathbf{R}\mathbf{R}^{\mathrm{T}})\mathbf{V}^{\mathrm{T}} = \mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{V}^{2}.$$

It shows that the left Cauchy deformation tensor is equal to the square of the left stretch tensor, which is invariant to body rotation.

Moreover, since $\mathbf{F} = \mathbf{RU} = \mathbf{VR}$, we have

$$\mathbf{U} = \mathbf{R}^{\mathrm{T}}\mathbf{VR}. \tag{3-3}$$

Based on the definition of eigenvalues and eigenvectors, we assume

$$\mathbf{U}\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i, i = 1, \dots, n, \tag{3-4}$$

where λ_i and $\boldsymbol{x}_i$ are the corresponding eigenvalues and eigenvectors of $\mathbf{U}$, and n is the size of dimension. By applying Equation 3-3 into Equation 3-4, we have

$$\mathbf{U}\boldsymbol{x}_i = (\mathbf{R}^{\mathrm{T}}\mathbf{VR})\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i, i = 1, \dots, n. \tag{3-5}$$

By applying the same rotation $\mathbf{R}$ to both sides of Equation 3-5, we obtain

$$\mathbf{R}(\mathbf{R}^{\mathrm{T}}\mathbf{VR})\boldsymbol{x}_i = \mathbf{V}(\mathbf{R}\boldsymbol{x}_i) = \mathbf{R}\boldsymbol{\lambda}\boldsymbol{x}_i = \lambda_i(\mathbf{R}\boldsymbol{x}_i), i = 1, \dots, n.$$

Assuming $\boldsymbol{y}_i = \mathbf{R}\boldsymbol{x}_i$, we have

$$\mathbf{V}\boldsymbol{y}_i = \lambda_i \boldsymbol{y}_i, i = 1, \dots, n.$$

Therefore, the two stretch tensors $\mathbf{U}$ and $\mathbf{V}$ have the same eigenvalues, although the eigenvectors may be different. The obtained eigenvalues are called *principal stretches*. In the next section, we use the principal stretches to measure the degree of deformation of deforming triangles. Since $\mathbf{U}$ and $\mathbf{V}$ have the same principal stretches, either the right Cauchy deformation tensor $\mathbf{C}$ or the left Cauchy deformation tensor $\mathbf{B}$ can be used.

3.2.2 Strain

We begin by representing the deformation for each triangle as an affine transformation between the current frame and a reference frame. Non-translational component of the affine transformation encodes the change in orientation, scale, and skew induced by the deformation on the triangle.

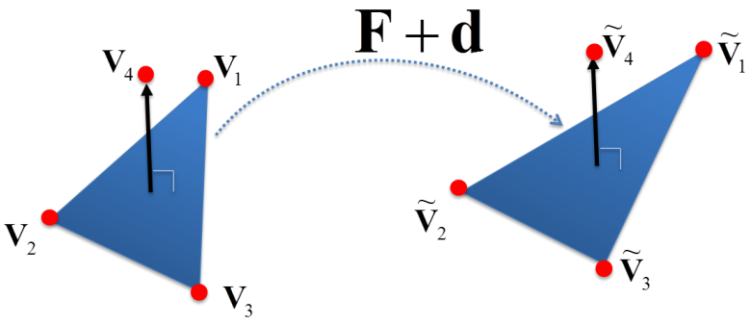


Figure 3-5 The deformation of a triangle in two frames.

As depicted in Figure 3-5, let $\mathbf{v}_i$ and $\tilde{\mathbf{v}}_i$ be the vertices of the triangle before and after the deformation, respectively. A 3 by 3 affine matrix $\mathbf{F}$ and displacement vector $\mathbf{d}$ transforms $\mathbf{v}_i$ into $\tilde{\mathbf{v}}_i$ as follows:

$$\mathbf{F}\cdot\mathbf{v}_i + \mathbf{d} = \tilde{\mathbf{v}}_i, i = 1, \dots 3$$

Similar to Sumner et al.'s approach (Sumner et al., 2005), we add a fourth vertex in the direction of the normal vector of the triangle and subtract the first equation from the others to eliminate $\mathbf{d}$. In specific, the fourth vertex $\mathbf{v}_4$ is an unit vector, obtained by the cross-product of two edge vectors of a triangle. In Figure 3-5, we use $\overrightarrow{v_1v_2}$ and $\overrightarrow{v_1v_3}$ before deformation (left-side triangle) and use $\overrightarrow{\tilde{v}_1\tilde{v}_2}$ and $\overrightarrow{\tilde{v}_1\tilde{v}_3}$ after deformation (right-side triangle). Then, we get $\mathbf{FV} = \tilde{\mathbf{V}}$ where

$$\mathbf{V} = [\mathbf{v}_2 - \mathbf{v}_1 \;\; \mathbf{v}_3 - \mathbf{v}_1 \;\; \mathbf{v}_4 - \mathbf{v}_1],$$

and

$$\tilde{\mathbf{V}} = [\tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1 \;\; \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1 \;\; \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1].$$

Note that this representation specifies the deformation in per-triangle basis, and therefore it is independent of the specific position and orientation of the mesh in world coordinates.

In continuum mechanics, the same matrix $\mathbf{F}$ is called deformation gradient tensor (Crandall et al., 1978) as it explains the relationship between a material vector in the reference object (before deformation) and the deformed one, i.e., $\mathbf{F}_{ij} = \partial\mathbf{v}_i / \partial\tilde{\mathbf{v}}_j$. Without loss of generality, we assume that the triangle is stretched first and then rotated. Thus, we have $\mathbf{F} = \mathbf{RU}$, where $\mathbf{R}$ denotes the rotation tensor and $\mathbf{U}$ the stretch tensor. Since we want to differentiate triangles according to their degree of stretch and shrinking, we eliminate the rotation component of $\mathbf{F}$ by computing the right Cauchy deformation tensor $\mathbf{C}$ as defined in Equation 3-2, where $\mathbf{C}$ is

equal to the square of the right stretch tensor. We obtain principal stretches through the eigen-analysis on $\mathbf{C}$, and compute the deformation of a triangle as $(\lambda_1 + 1/\lambda_3)$, where λ_1, λ_2 and λ_3 are the principal components of $\mathbf{C}$. Intuitively, λ_1 and $1/\lambda_3$ reflects the degree of stretch and shrinking, respectively. That is, the higher degree of stretch (or shrinking) occurs in the triangle, the larger values of λ_1 (or $1/\lambda_3$) we will obtain.

Therefore, the strain value we have obtained measures the degree of deformation of the triangle. In addition, since all the measurements are based on the adjacent edges and their relative length changes throughout deformation, the per-triangle feature descriptor is independent on rigid transformations.

Exceptional triangles Note that although the strain value of a deformed triangle can be computed efficiently by the spectral decomposition of a 3-by-3 deformation tensor, this computation may fail on some degenerated triangles. That is, when either of the reference or current triangle has edges with near-zero length, the gradient tensor analysis will fail while computing $\mathbf{F}_{ij} = \partial \mathbf{v}_i / \partial \widetilde{\mathbf{v}}_j$. In this thesis, we assume such exceptional triangles do not occur in the deforming meshes in our experiments.

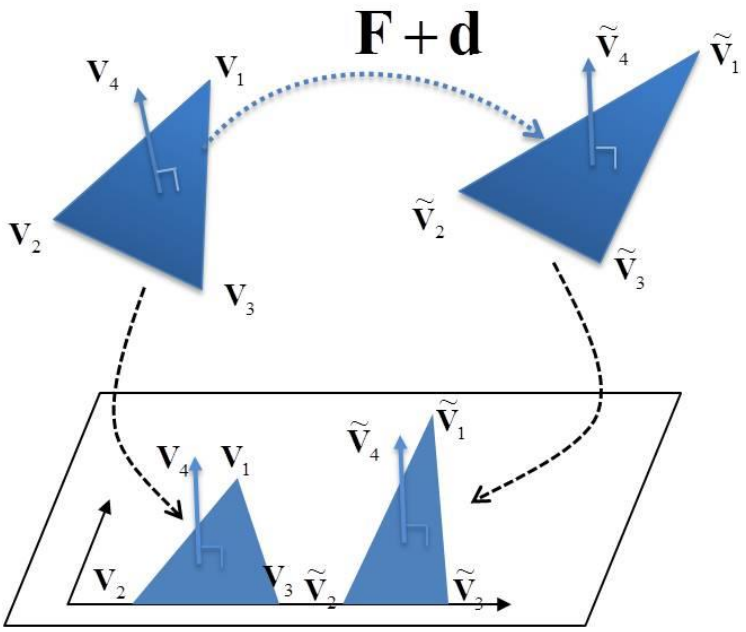


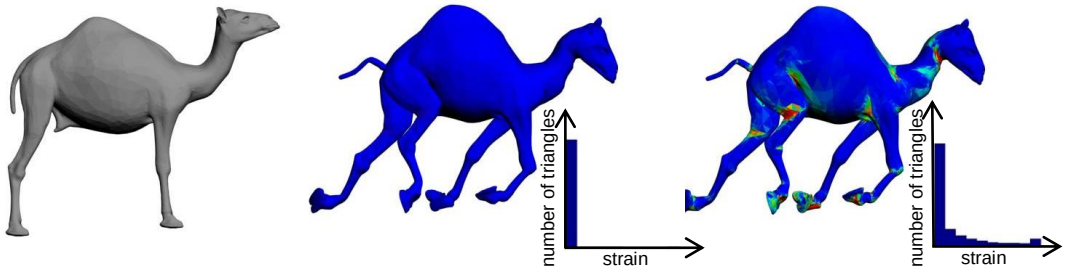
Figure 3-6 Mapping 3D triangles into the same 2D plane, to compute 2D strain.

3D strain vs. 2D strain In this section, we have introduced a 3D strain based on 3-by-3 deformation tensor. In fact, a 2D strain can be obtained by mapping the triangles into the same plane and compute the spectral decomposition of the 2-by-2 deformation tensor within the plane (see Figure 3-6). We compare the 3D strain and 2D strain as follows :

- *Computation*. 2D strain is more efficent on spectral decomposition because it analysis a 2D matrix. On the other hand, 2D strain requires extra computation to map triangles into the same plane.

- *Result*. We have obtained 3 eigenvalues for 3D strain, with one of them as a constant value 1. This is because we compute the fourth vertex $\mathbf{v}_4$ as an unit vector, which means there is no deformation in the dimension perpendicular to triangle. For the other 2 eigenvalues, they are identical to those of 2D strain which both measure the deformation independent on triangle rotation, translation and uniform scaling. Since we only use these 2 eigenvalues from 3D strain, the computed degree of deformation are the same as 2D strain.

3.2.3 Strain normalization



(a) Reference pose. (b) Original strains. (c) Normalized strains.

Figure 3-7 Strain normalizations. Mesh triangles are shown with colour varying from blue to red, indicating strain values from low to high. By comparing to (a) the reference pose, (b) shows the strain values computed by using the method in Section 2.1.1, and (c) shows the normalized strain values by using Gaussian Kernel Function. The histograms in (b) and (c) show the distributions of the strains before and after normalization, respectively.

Although we have excluded the occurrence of triangle with near-zero length edge, in a less extreme situation, we still may obtain exceptional high strain values for the case of long triangles, i.e., triangles with two long edges and a relatively short edge. In Figure 3-7, with a 'galloping-camel' mesh, we show the mesh triangles with colors varying from blue and red, to linearly indicate strain values from low to high. Figure 3-7(b) shows the original triangle strains on the mesh, where we see only blue triangles. This is because there are a few triangles with extraordinary large strain values (shown in red but not visible due to small quantity) due to noises, while the other triangle strain values are relatively small, see the histogram in Figure 3-7(b). Therefore, we observe the mesh triangles in blue without seeing the very small number of red triangles.

To alleviate such exceptional high strain values, we filter the extremely high strain values by using a Gaussian Kernel Function (GKF): Given a deforming mesh $\mathcal{M}$ with *M* frames and *N* triangles, we represent each frame $f^p, \mathrm{p} = 1, \ldots, \mathrm{M}$, as a vector of strain values $\mathrm{s^p} = (\mathrm{s_1^p}, \ldots, \mathrm{s_N^p})^{\mathrm{T}}$, which we obtain by the method described in Section 3.1. The strain values are then normalized into [0 1] by using a Gaussian Kernel Function (GKF):

$$\overline{s_i^p} = exp\left(-\frac{s_i^p}{2\sigma^2}\right), i = 1, \ldots, N,$$

where σ is a width parameter that is derived from the average of strain values: $\sigma = 2(\sum_{i,p} s_i^p)/(N \times M)$.The Gaussian filter normalizes the strain values into [0 1]. Figure 3-7(c) shows an example of normalized triangle strain values. By comparing

to the original strain values in Figure 3-7(b), we have significantly improved the granularity of the strain values, i.e., the most highly deformed regions are shown in red and the rigid regions are in blue.

Complexity. The computation of the *strain* for each triangle requires the deformation tensor analysis of a 3-by-3 affine matrix (see Section 3.2.2). Therefore, the computation of triangle strains of a deforming mesh $\mathcal{M}$ with *M* frames and *N* triangles consumes *O*(*M*·*N*) time.

Invariance. Since the measurement of the degree of deformation is based on the adjacent edges and their relative length changes along time, it is obvious that the per-triangle strain is independent of global rotation, translation and normal scale of the mesh.

Robustness over shape difference. Since *strain* is a deformation-based descriptor, we expect consistent behaviors for the deforming meshes undergoing identical or similar motions, despite their shape differences. To validate the motion consistency, we compute the strains of a 'Camel' and a 'Horse' with the same pose, by using the same reference poses. In Figure 3-8 and Figure 3-9, we show the behaviors of strains computed by using either rest-pose or previous-pose as reference poses, respectively. We show the mesh triangles in colors varying from blue to red, to indicate their low and high strain values.

In both Figure 3-8 and Figure 3-9, for the corresponding frames of 'horse' and 'Camel', we can observe similarly colored regions on both of the meshes, such as the 'neck', the knees on the right-front 'leg' and the top of the right-back 'leg'. Based on this observation, our per-triangle strain values are consistent between the two different meshes with similar poses.

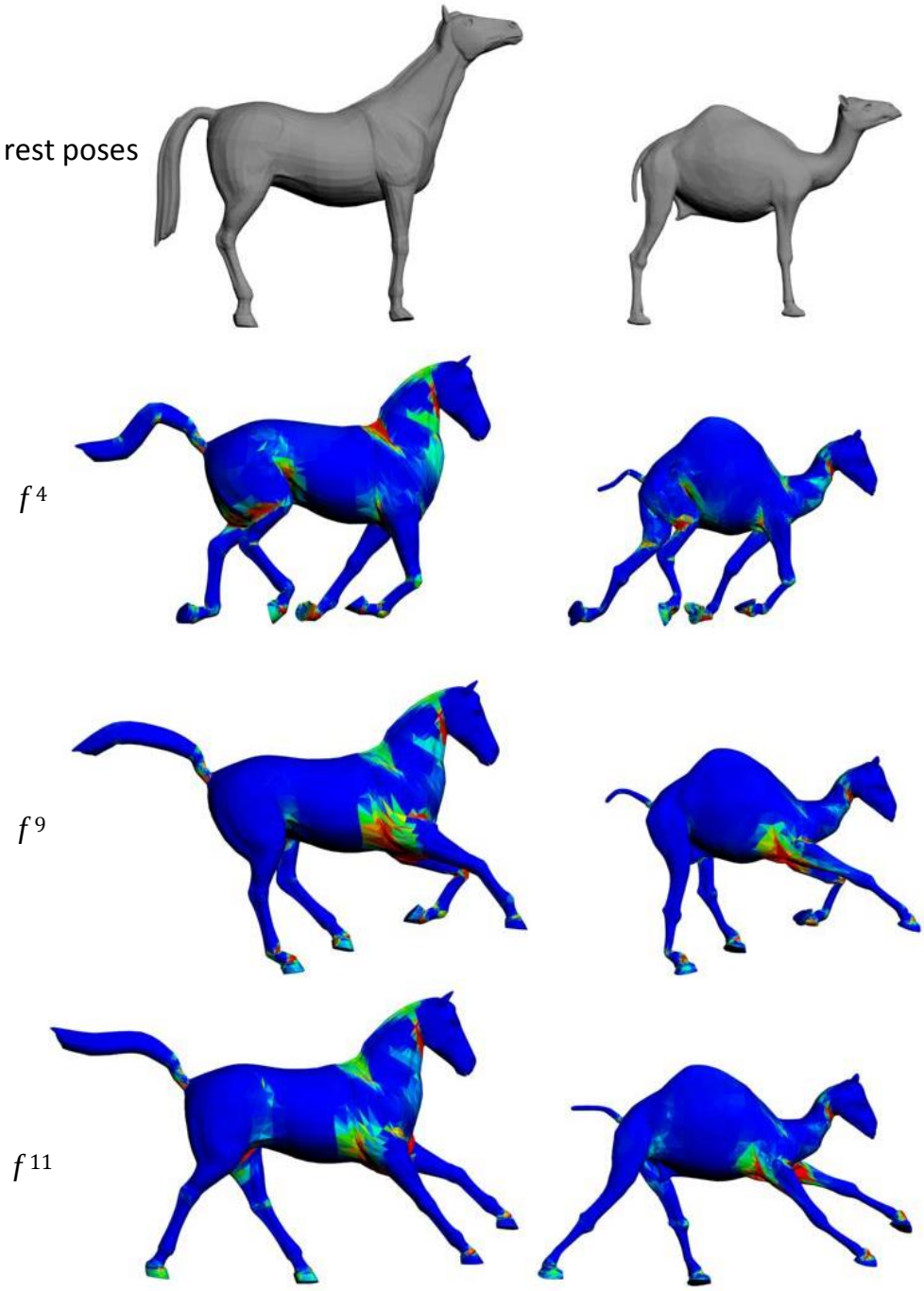


Figure 3-8 The same behaviors of rest-pose based strains for both 'Horse' and 'Camel', with respect to the same rest poses.

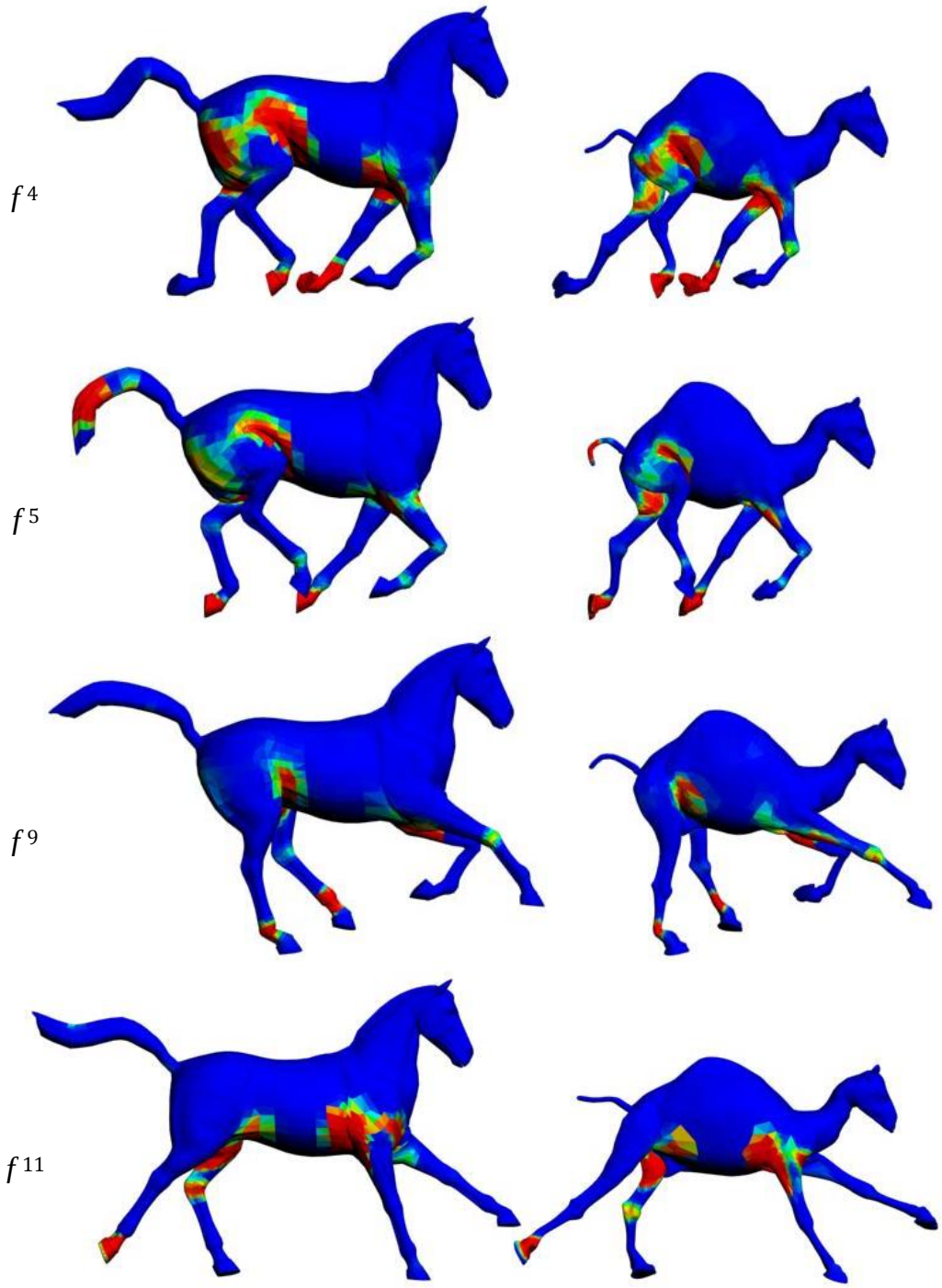
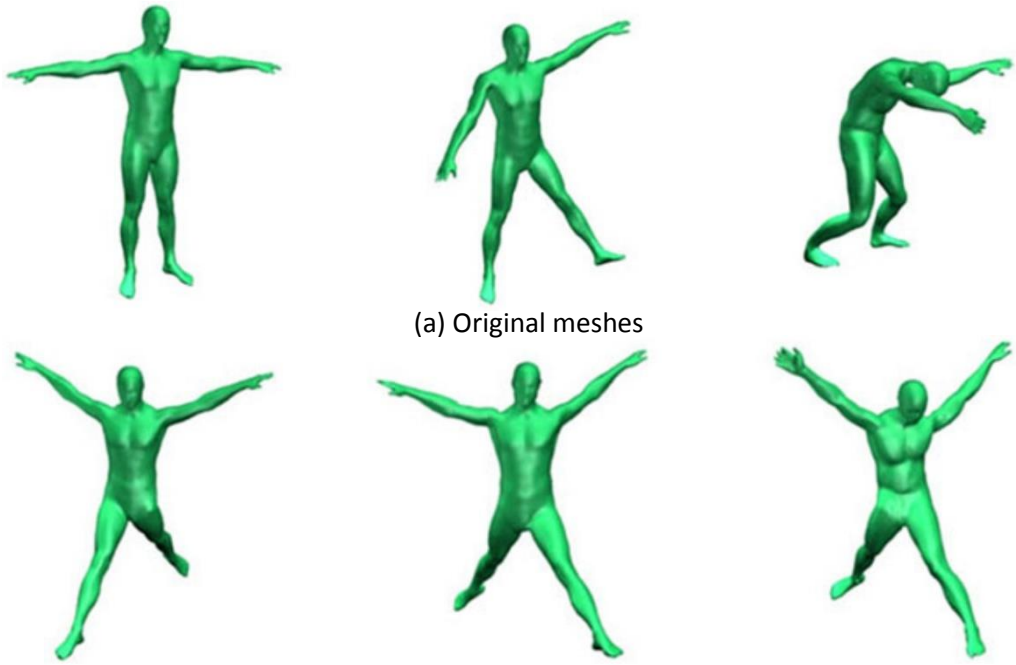


Figure 3-9 The same behaviors of previous-pose based strains for both 'Horse' and 'Camel'.

3.3 Strain with respect to rest-pose vs. previous-pose

In the previous section, we have presented a feature descriptor, which measures the degree of deformation of a triangle by comparing to a reference pose. In practice, the reference pose can be either the rest pose, which in many cases is the first

frame (see Figure 3-8), or the previous pose (the previous successive frame, see Figure 3-9). In this section, we present the different behaviors of the strain values computed based on both rest-pose and previous-pose, by using synthetic data and real animation data.



(a) Original meshes

(b) Feature-preserved Canonical Forms

Figure 3-10 (a) Non-rigid meshes and (b) their corresponding feature-preserved 3D canonical forms (Lian et al., 2013).

Note that for computing the rest-pose based strains, one can either use any pose in the deforming mesh as a rest-pose, or generate it by applying existing methods if none of the poses is appropriate as a reference. For example, Lian et al. (Lian et al., 2013) propose a method to transform different meshes into a *Canonical Form* through properly designed rotations and translations of each near-rigid subparts (see Figure 3-10). Then, a rest-pose can be obtained by taking the average of the Canonical Forms of all frames.

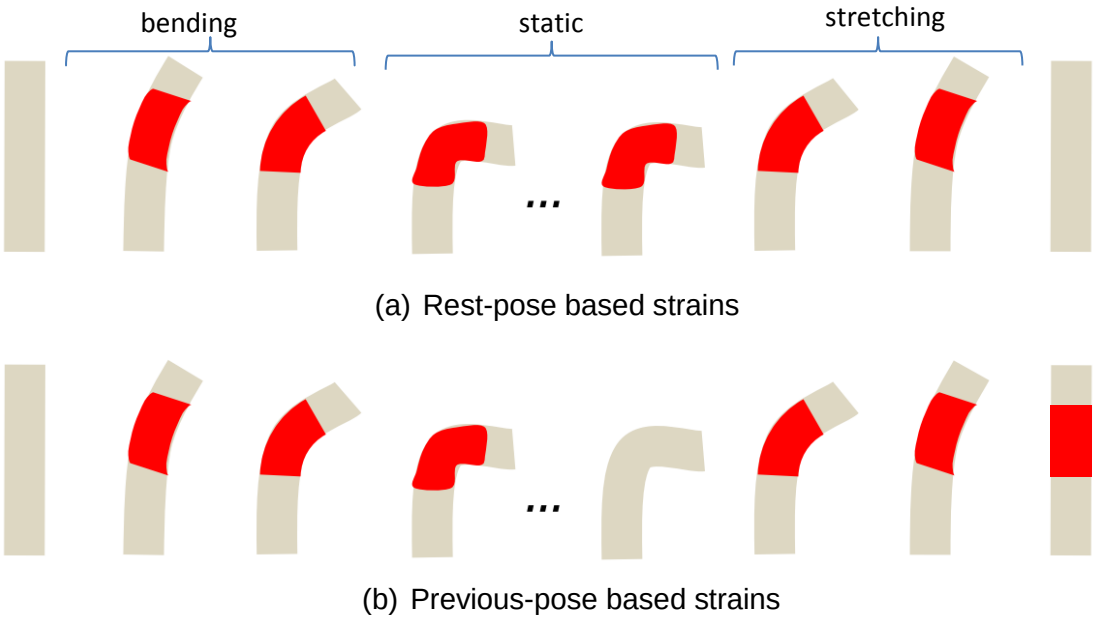


Figure 3-11 Comparisons between (a) rest-pose based and (b) previous-pose based strains for a synthetic deforming mesh 'bending-cylinder'.

Analysis of synthetic data. We first show the differences by using a synthetic 'bending-cylinder' mesh sequence in Figure 3-11. In this mesh sequence, the 'cylinder' mesh first bends a joint, then remains static, and finally stretches back to the starting pose. In each of the mesh, we use red color to indicate the deforming regions, where each is a patch of triangles with non-zero strains. We show both the rest-pose based and previous-pose based strains of the mesh sequence in Figure 3-11. We describe the different behaviours of each as follows:

- Rest-pose based strains. In the top row of Figure 3-11, we show the strains for each frame based on the rest pose (the first frame). We can observe that we obtain non-zero strains in the joint region in all the frames except the last frame. This is because the last frame has the same pose as the rest pose, while the other intermediate frames are all deformed in the bended-joint region, compared to the rest pose.

- Previous-pose based strains. In the bottom row of Figure 3-11, we show the strains for each frame based on the previous pose. We observe that we obtain zero strains while the 'cylinder' remains static with bended-pose. This is because each frame is not deformed by comparing to its previous frame in this 'static' period. On the other hand, we obtain non-zero strains in the other frames, including the last frame since it is deformed by comparing to its previous frame.

By observing the above behaviors of both rest-pose based and previous-pose based strains of the synthesized data in Figure 3-11, the main differences are from static frames and the last frame. The comparison shows that previous-pose based strain is more sensitive to the motions in the deforming mesh. That is, we obtain non-zero strains if a motion occurs, which deformes the mesh. On the other hand, the rest-pose based strains consider each frame independently by comparing to the rest pose.

Analysis of real-world data. To further analyse the differences between the two strain types, we compute both types of strains for three selected frames in a 'galloping-camel' mesh sequence, as shown in Figure 3-12. We use a neutral pose as the reference pose for computing rest-pose based strains. In each of the mesh in Figure 3-12, the triangles are colored from blue to red to indicate strain values from low to high. We describe the behaviours of both strains as follows:

- *Rest-pose based strains*. In the top-right row in Figure 3-12, we show the strain values for each frame based on the rest pose shown in the left-most in Figure 3-12. We can observe that we obtain non-zero strains in the 'neck' region and the front-right 'knee' in all the frames. This is because these regions in these frames are all deformed by comparing to the corresponding regions in the reference pose.

- *Previous-pose based strains*. In the bottom-right row in Figure 3-12, we show the strains for each frame based on each frame's previous pose. We observe that the 'neck' regions are mostly shown in blue. This is because the Camel rarely moves the 'neck' when galloping. Moreover, we obtain mostly zero-strains in the front-right 'knee' in f^6. This is because the front-right 'knee' remains nearly the same pose from f^5 to f^6.

Based on the above comparisons on the 'galloping-camel' data, we observe similar strain behaviors computed from the synthesized 'bending-cylinder' data. These differences can be seen clearly from two regions: (1) 'neck' regions. Because the 'neck' barely moves during galloping, we obtain mostly zero-value of the previous-pose based strains. On the contrary we obtain nonzero-value of the rest-pose based strains because the 'neck' poses are different from its pose in the rest pose. (2) Front-right 'knee'. We obtain zero-values of the previous-pose based strains in f^6, since the 'knee' pose is almost the same in f^5. On the contrary, we obtain nonzero-value of the rest-pose based strains in f^6, because the 'knee' poses in both frames are different from its pose in the rest pose.

Based on the above comparisons, we summarize the differences between the two strain types and their potential applications as follows:

1) Rest-pose based strain.

This type of strain independently measures the deformation of each frame by comparing to a rest pose. Since the rest-pose based strains are computed based on the same reference pose, we can apply these strains to compare the pose similarities by representing each frame into triangle strain vector. By using this mesh representation, in Chapter 1, we devise a temporal segmentation method of deforming meshes that each temporal segment contains similar frames, where the frame similarities are computed based on rest-pose based strains.

2) Previous-pose based strain.

Comparing to the rest-pose based strains, the previous-pose based strain are more sensitive to motions. More specifically, each time a motion occurs, a frame is deformed from its previous frame, and therefore we obtain non-zero strains. Based on this fact, in Chapter 5, we use the previous-pose based strains for computing spatio-temporal segmentation of deforming meshes, whose results is extended towards motion similarity measurement between deforming meshes.

3.4 Summary

In this chapter, we first give a review of the most popular existing feature descriptors for 3D shapes in Section 3.1. After that, we present a new per-triangle deformation-based feature descriptor that is invariant to rigid deformations, including rotation, translation and normal scale. We also further process the strains so that they are robust under presence of small noises. After that, we have discussed several properties of the proposed descriptor: (1) It is invariant to global mesh rotation or translation, and (2) it shows the similar behaviors for two different shapes performing identical or similar motions. Finally, we compare and summarize the behaviours of two types of strains, i.e., rest-pose based and previous-pose based strains, by using both synthesized animations and real-world 3D animations.

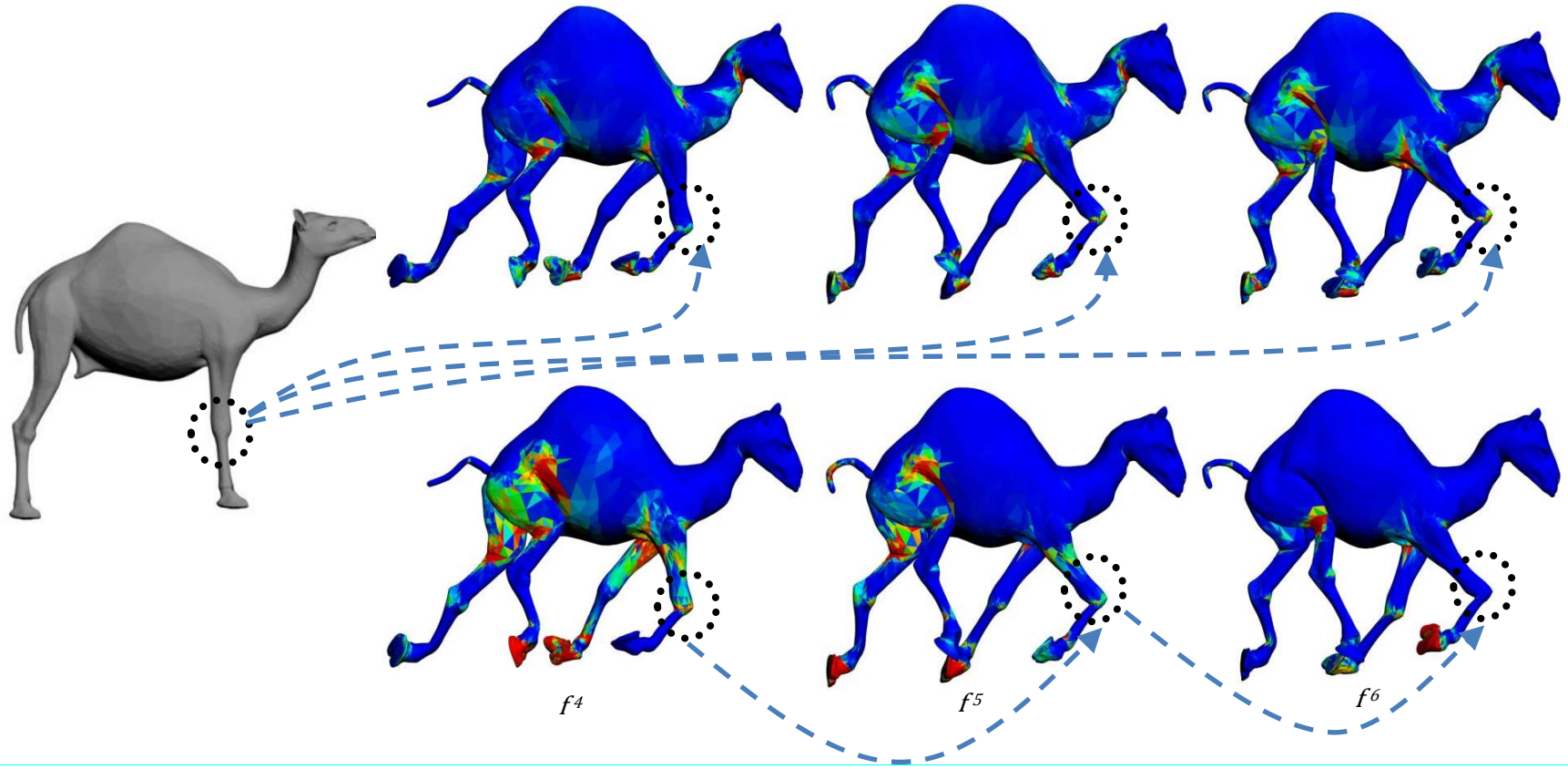


Figure 3-12 Strains for a 'galloping-camel' data. From left to right and top to bottom, left-most is the rest pose, top-row the rest-pose based strains and bottom-row the previous-pose based strains. The dashed arrows indicate the corresponding reference poses for computing strains.

Chapter 4 Temporal Segmentation of Deforming Meshes

Most of the existing works on deforming meshes compute one single spatial segmentation according to geodesic and kinematic affinities of mesh primitives, i.e., vertices or triangles. In such cases, it is clear that the spatial segmentation results may significantly be different depending on the motions performed by a deforming mesh. Ideally, the mesh decomposition results should represent well the deformation exhibited on the mesh. However, when it comes to a long and complex motion composed of several basic motions, one may obtain overly segmented patches, which do not represent well each sub-motion. To avoid such problem, we perform temporal segmentation prior to spatial segmentation, aiming to compute the representative spatial segmentation for the sub-motion within each temporal segment.

In this chapter, based on the dynamic feature descriptor described in Chapter 3, we present a method for the temporal segmentation of deforming meshes that allows to obtain consistent temporal segmentation for different deforming meshes exhibiting identical or similar motions, despite their shape differences, which is desirable. In Section 4.3, we demonstrate our segmentation results over both synthesized and motion captured deforming meshes, and compare with Barbič et al.'s method (Barbič et al., 2004). Finally, we conclude the descriptions of our new temporal segmentation method in Section 4.4.

4.1 Background

Given a long sequence of animation, the objective of the temporal segmentation is to cut the sequence into meaningful motion clips, or gestures in the case of human motions. A large variety of applications have been developed based on temporal segmentation, such as motion classification / recognition (Müller and Röder, 2006) (Spriggs et al., 2009), motion data retrieval (Müller et al., 2005), compression (Liu and McMillan, 2006), animation editing (Kovar et al., 2002), etc. In this section,

we give a review of several existing methods for character animations, i.e., motion capture data, and for the video analysis in Computer Vision field.

4.1.1 Temporal segmentation of motion capture data

As noted by Kahol et al. (Kahol et al., 2004), the temporal segmentation of human motion capture data is a challenging problem because :

- Motion boundaries are often subjective.
- A motion sequence can be segmented into different levels of sub-motions. For example, a human 'walking' motion can be decomposed into sub-motions of legs : 'left-stop', 'left-forward', 'right-forward' and 'right-stop' (Zhou et al., 2013).
- It may not be possible to enumerate all sub-motions for the methods by using sub-motion templates, such as HMM-based or learning-based methods.

Several supervised (Kahol et al., 2004) and unsupervised (Barbič et al., 2004) (Janus and Nakamura, 2005) methods have been proposed to address these problems. Kahol et al. (Kahol et al., 2004) propose a temporal segmentation method for dance sequences by learning from empirical data, which contains the ground-truth boundary frames of training data. Their algorithm first recognizes the potential boundaries as the local minima of feature cues, then determines the correct boundaries by feeding the cues' binary values (whether a feature cue within a frame is locally minimal or not) in the potential boundaries to a naïve Bayesian classifier, which has been trained by using empirical data. One obvious limitation of such supervised method is that the segmentation results are biased to human empirical data, which may not cover all the possible boundaries. Another method proposed by Barbič et al. (Barbič et al., 2004) avoids such limitation. They conduct PCA analysis of the motion data within a sliding window with a fixed number of intrinsic dimensionalities, which returns an error for the reconstruction of the motion data. By following this procedure, one can obtain and observe an error curve as the sliding window moves forward, see Figure 4-1. Finally, the method automatically detects boundary frames along the error curve if significant changes occur, i.e., the change of the error exceeds a predefined threshold. Figure 4-1 (Barbič et al., 2004) shows an example of segmenting 'walk' and 'run' motions in a sequence by applying this algorithm. Similarly, Janus and Nakamura (Janus and Nakamura, 2005) model motion data in a sliding window with Hidden Markov Model (HMM) (Baum, 1972), and their method

automatically determines the boundary frames by observing the significant changes of the probability density of the HMM.

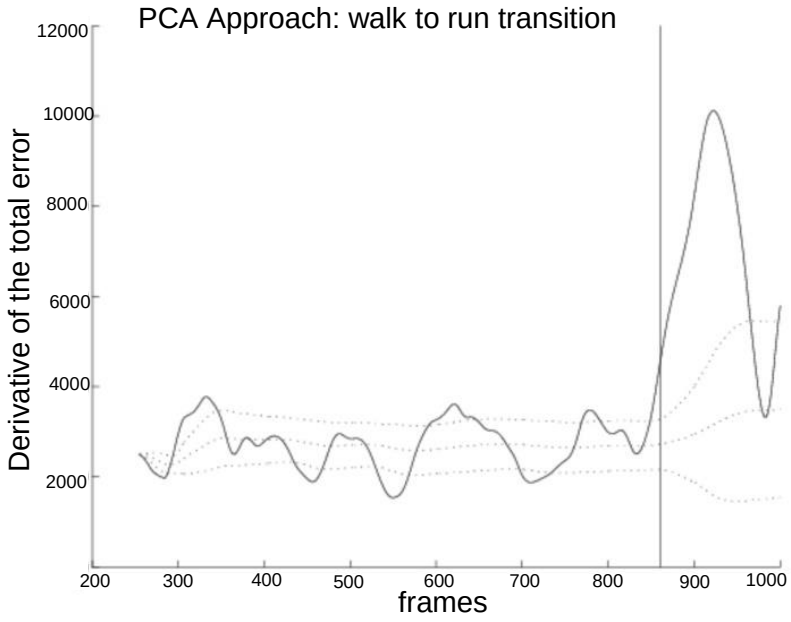


Figure 4-1 Temporal segmentation of motion capture data (Barbič et al., 2004). The solid curve is a reconstruction error curve by applying spectral analysis on a sliding window. The dashed line shows the standard deviation of the previous values. The vertical solid line indicates a boundary for the temporal segmentation.

4.1.2 Temporal segmentation of videos

In Computer Vision field, the objective of the temporal segmentation of videos is to determine boundary frames that divide a video into meaningful clips. The temporal segmentation of videos is considered as a primary step for automatic annotation of video sequences, which can be used as basic elements for applications such as video browsing and retrieval (Koprinska and Carrato, 2001). A large variety of techniques for video segmentation have been developed, which we categorize into two classes as follows:

- Threshold-based method . A considerable portion of the temporal video segmentation methods determine boundary frames by using a threshold based on frame distance (Boreczky and Rowe, 1996) (Pass and Zabih, 1999) (Shahraray, 1995) (Krishna et al., 2014). Most of these methods are based on the similarity measurement between successive images, and determine a cut if the distance between two neighbouring images exceeds a predefined threshold. Since these approaches are based on inter-image distance that limits their applicability to scenarios requiring semantic modelling.

Krishna et al. (Krishna et al., 2014) recently propose an approach that first trains a One Class Classifier based on Support Vector Machine (Maji et al., 2008) with the frames for 1 to 3 seconds. Then, they apply the learned model on each of the successive image to compute a *novelty score*. When the novelty score of a frame exceeds a prefined threshold, the frame is chosen as a boundary frame.

- Over-segmentation based method. Another class of the temporal video segmentation methods detects the turning points of the motion acceleration and/or deceleration. Since neither inter-image distance nor image modelling are required, such methods are fast and threshold-free, and thus can be used for online segmentation (Liu et al., 2003). Several frequently used criteria for determining primitive movements are the local minima / maxima of velocity (Wang et al., 2001), angular velocity (Fod et al., 2002) of the trajectories of observed points, or objects. Having the primitive movements, higher level actions can then be defined as the combinations of primitive movements. By taking musical conducting gestures as an example, five conducting prototype gestures can be defined by using the primitive movements (Wang et al., 2001) (see Figure 4-2).

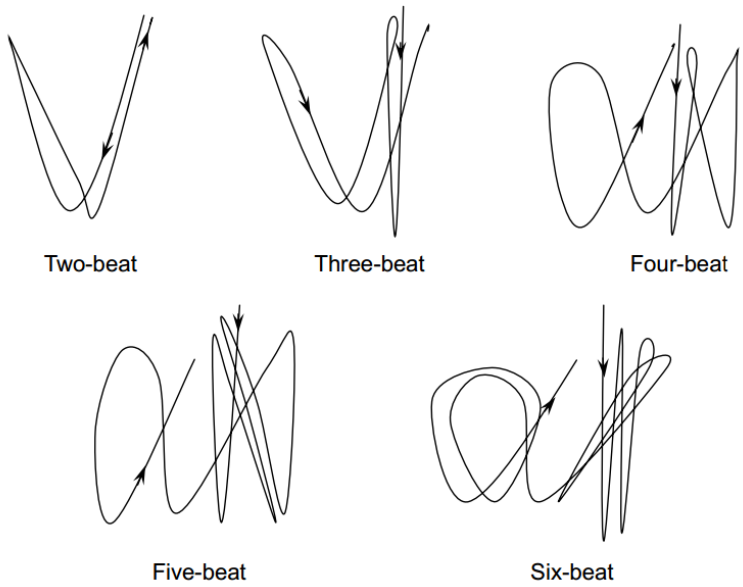


Figure 4-2 Five conducting prototype gestures defined based on primitive movements, which are obtained by over-segmentation (Wang et al., 2001).

4.1.3 Motivation of our approach

As have been reviewed in the previous section, there has been a variety of techniques for temporally segmenting videos and motion capture data. However, to the

best of our knowledge, no previous work has been done for the temporal segmentation of deforming meshes. Moreover, the existing temporal segmentation techniques for video processing or motion captured animations may not be directly applicable for deforming mesh due to the following reasons :

- For the temporal segmentation techniques for video processing in Computer Vision, there exists a large heap of local information available from image data, such as colour, texture and geometry, etc., yet mesh data in most cases contain only geometry information.
- Comparing to the motion captured animations, deforming mesh data consists of much higher dimension : a motion capture data can be obtained as the movements of up to hundreds of markers, while a deforming mesh may have tens of thousands of vertices. For this reason, the temporal segmentation methods for motion capture data may not be applicable to handle deforming meshes. For example, the temporal segmentation method proposed by Barbič et al. (Barbič et al., 2004) may fail with a deforming mesh if the mesh has a large vertex number, because the computational complexity of PCA is $O(n^3)$, which increases sharply as the number of vertex n increases.

Therefore, in the next section, we present a new temporal segmentation method for deforming meshes.

4.2 Temporal segmentation of deforming meshes

Having computed the per-triangle strain values at each frame by using the descriptor presented in Chapter 3, we proceed with temporal segmentation as follows: we first compute the deformation distance between every frame pair, and measure the average frame dissimilarities within all possible sub-sequences. Then, the boundary frames for the temporal segmentation are determined by minimizing the sum of within-segment frame dissimilarities. This optimization-based method allows to obtain the segmentation results where each temporal segment is a subsequence of similar frames, i.e., similar poses.

Note that we use the rest-pose based strains (see Section 3.3) in this work. This is because our temporal segmentation method is based on the measurement of pairwise frame distances, which indirectly requires that the strain of each frame should be computed with respect to the same reference pose.

4.2.1 Within-segment frame dissimilarity

Given a deforming mesh $\mathcal{M}$ with M frames and N triangles, we represent each frame $f^p, p = 1, \dots, M$, with a vector of triangle strain values $\boldsymbol{s^p} = (s_1^p, \dots, s_N^p)^{\boldsymbol{T}}$, where the per-triangle strains can be computed by using the method described in Chapter 3. We then can compute the distance $A_f(p, q)$ between frames f^p and f^q by taking the Euclidean distance between their corresponding strain vectors $\boldsymbol{s^p}$ and $\boldsymbol{s^q}$, i.e.,

$$A_f(p, q) = \left\| \boldsymbol{s^p} - \boldsymbol{s^q} \right\|_{L_2}.$$

Based on this distance metric between frame pairs, we consider every possible subsequence $[p, q]$ (from f^p to f^q) as a candidate temporal segment, with the corresponding within-segment dissimilarity $D_s(p, q)$ defined as follows:

$$D_s(p, q) = \begin{cases} \frac{\sum_{m=n+1}^{q} \sum_{n=p}^{q-1} A_f(n,m)}{(q-p+1)*(q-p)/2}, & p < q, \\ A_s(q, p), & p > q, \\ 1, & p = q, \end{cases}$$

where $p, q = 1, \dots, M$.

Intuitively, $D_s(p, q)$ is the average dissimilarity of all possible frame pairs within $[p, q]$. Figure 4-3 shows the within-segment dissimilarity matrix computed for a facial motion capture data. We show the dissimilarity matrix in color varying from blue to red, indicating dissimilarity values from low to high.

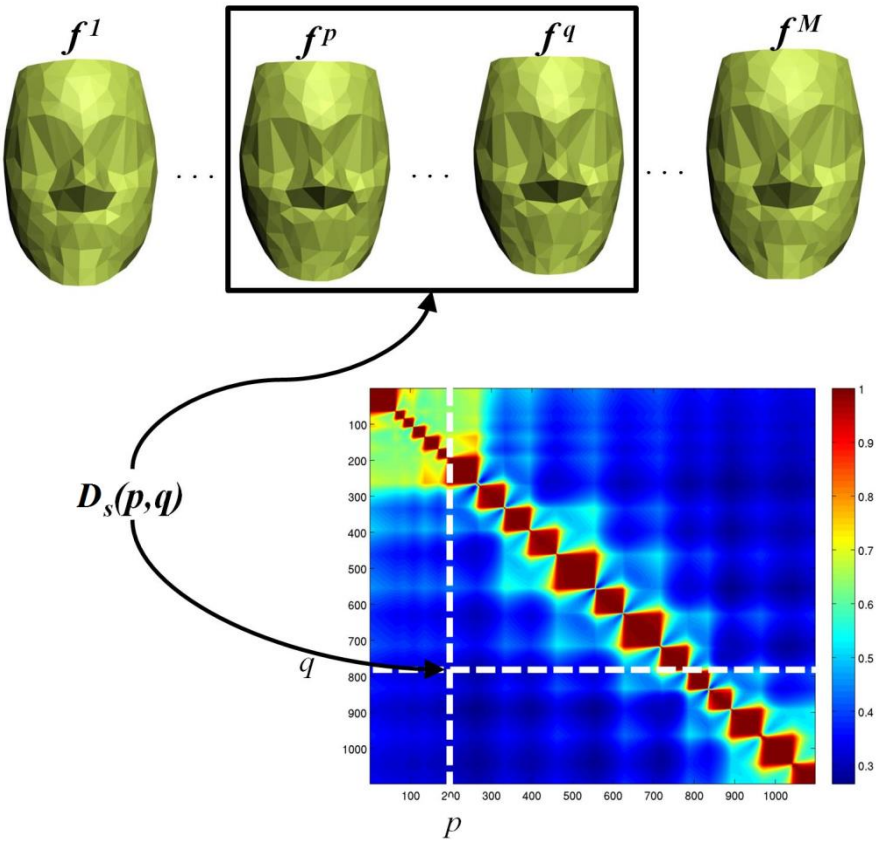


Figure 4-3 Within-segment dissimilarity matrix.

4.2.2 Temporal segmentation

The goal of our temporal segmentation is to cut the given mesh sequence $\mathcal{M}$ into distinct segments $\mathcal{M}_1$, ..., $\mathcal{M}_K$, where K is the number of segments. K as well as the boundary frames $\boldsymbol{I}_{\boldsymbol{B}_k}$, $k=1,\ldots, K$, (indices of the first frame in each segment) are to be determined. To achieve this, we consider every possible subsequence represented by the average of pairwise frame distance. The main idea is to determine a temporal segmentation where each subsequence represents a low dissimilarity value. This indicates that the subsequence contains a set of meshes with similar poses.

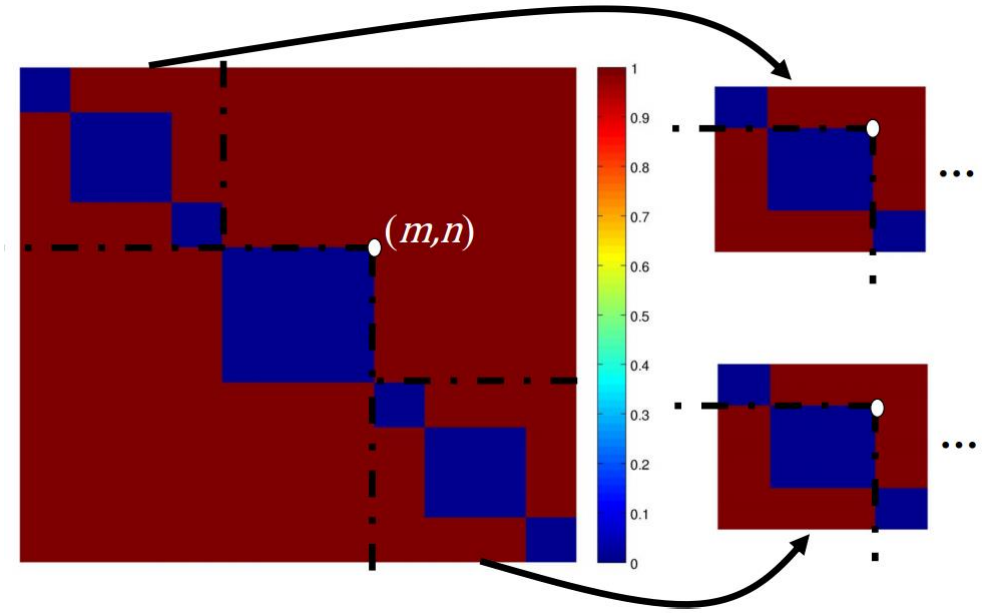


Figure 4-4 A synthetic example of temporal segmentation. (left) The colored background is a within-segment average dissimilarity matrix, and (m,n) is the first detected temporal segment. (right) The remaining of the sequences are iteratively segmented by using the same approach.

We use a threshold ρ_t to determine whether the dissimilarity is low enough to form a temporal segment, i.e., if $D_s(p,q)<\rho_t$, $[p, q]$ is the considered temporal segment. In our experiments, ρ_t is driven from $\theta_1 \cdot \max(D_s)$, where $\theta_1 \in [0,1]$ is a user specified parameter. The corresponding settings of θ_1 for different experimental data are shown in Table 4-1. To avoid over-segmentation, we scan the dissimilarity matrix in descending order of subsequence length in favor of longer sub-sequences. For example, given two sub-sequences $[p_1, q_1]$ and $[p_2, q_2]$ in $[p_0, q_0]$ with $p_1<p_2<q_2<q_1$, $D_s(p_1,q_1)<\rho_t$ and $D_s(p_2,q_2)<\rho_t$, we take $[p_1, q_1]$ as a temporal segment because $q_1 - p_1 > q_2 - p_2$. In practice, this procedure is to scan the within-segment dissimilarity matrix from top-right corner towards the diagonal of the matrix until we encounter a dissimilarity value that is lower than θ_1. Next, we repeat the segmentation over the remaining sub-sequences $[p_0, p_1-1]$ and $[q_1+1, q_0]$. In Figure 4-4, sub-sequence $[m, n]$ is first found as a temporal segment, i.e., $D_s(m, n)<\rho_t$. Then, we recursively apply the temporal segmentation over the remaining sub-sequences $[1, m-1]$ and $[n+1, M]$. Sometimes, we may obtain short temporal segments due to noises. To alleviate this problem, for each of the temporal segment whose number of frames is less than $\theta_2 \cdot M$, where θ_2 is fixed as 0.02 in our experiments, we merge it to its neighboring segment with less dissimilarity.

The complete algorithm is shown in Algorithm 4-1. In this algorithm, ***max*()** is a function that returns the maximum value and the corresponding index in a vector.

In the most extreme case where each frame becomes a temporal segment, the algorithm has to scan all the items in the dissimilarity matrix D_s. Therefore, the algorithm runs in $O(M^2)$ time in the worst case.

Algorithm 1: *TempSeg*(I_B,D_s,I_h,I_t)

Init: I_B=[], I_h=1, I_t=M, A_s

$L=I_t-I_h+1$

for $l=L$ to 1 **do**

  for p=1 to $L-l+1$ **do**

    $D_{s\text{-}sub}(p)=D_s(p,p+l-1)$

  end for

  $[D_{s\text{-}max},p]=\boldsymbol{max}(D_{s\text{-}sub})$

  if $D_{s\text{-}max}>\rho_0$ **then**

    $I_B=[I_B\ p]$

    $q=p+l-1$;

    TempSeg(I_B,D_s,I_h,p-1)

    TempSeg(I_B,D_s,q+1,I_t)

    break

  end if

end for

Return: I_B

Algorithm 4-1 Temporal segmentation algorithm for deforming meshes. For the inputs, I_B is a vector of boundary frames' indices, D_s is the within-segment average dissimilarity matrix, I_h and I_t are the indices of the first frame and the last frame in the subsequence, respectively.

4.3 Experimental results and discussions

In this section, we first introduce our experimental environment and data, then show the temporal segmentation results. We also present a comparison between our method and the temporal segmentation method for character animations proposed by Barbič et al. (Barbič et al., 2004).

4.3.1 Experimental environment and data

We have tested our temporal segmentation method on both synthesized and motion captured animation data. Table 4-1 shows the dimensions of the datasets used, the thresholds, and the timings for computing the temporal segmentation of each data. The algorithm has been implemented in Matlab on an Intel Core 3.40GHz PC with 16GB of RAM. The computation for our temporal segmentation starts to be heavy as the mesh sequence becomes long, or the data dimension becomes large. For example, the runtime of 'Face1' with 1472 frames is 1128.9s, which is almost doubled comparing to 565.8s for 'Face2' with 1097 frames.

Table 4-1 Used data and the timings for temporal segmentation. The timings are the runtime for the temporal segmentation of each data (the computation of the within-segment dissimilarity matrix is excluded).

Name	**Number of faces**	**Number of frames**	**Motions**	$\boldsymbol{\theta_1}$	**Timings (second)**
Michael	29999	55	Walking	0.8	2.1
Gorilla	29999	55	Walking	0.8	1.9
Camel	43778	51	Galloping	0.8	3.3
Face1	286	1097	Facial expressions	0.6	565.8
Face2	269	1472	Facial expressions	0.7	1128.9

4.3.2 Temporal segmentation results

The segmentation results of the above deforming meshes are shown in Figure 4-5, Figure 4-7, Figure 4-10, Figure 4-11 and Figure 4-12, respectively. In each of these figures, we show a within-segment dissimilarity matrix with colours ranging from blue to red, to indicate within-segment dissimilarity values from low to high. And we use vertical dashed lines to indicate boundary frames. We discuss on the temporal segmentation results of each data below.

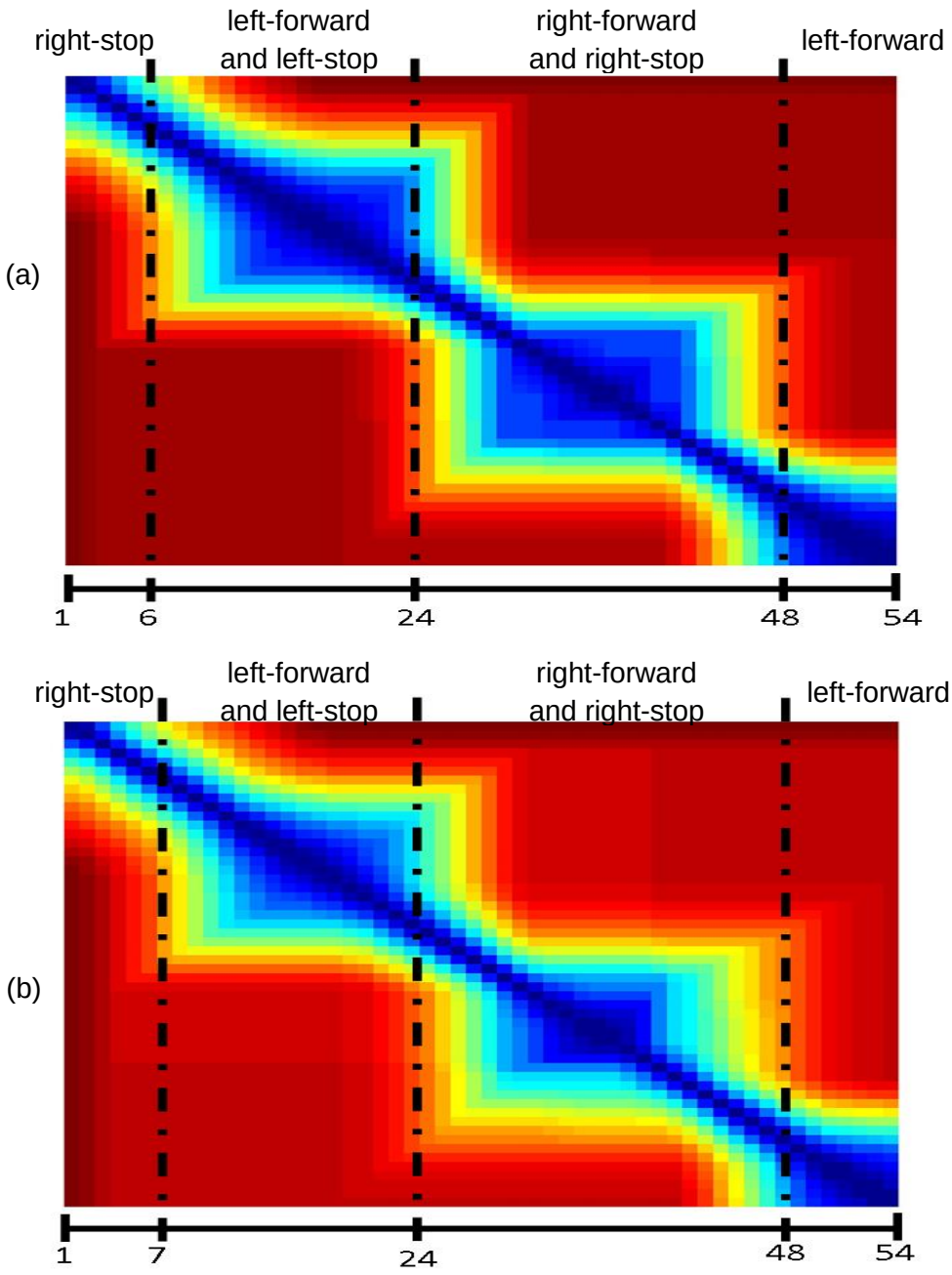


Figure 4-5 Temporal segmentation results of (a) 'Michael' and (b) 'Gorilla', which are both segmented into submotions: 'right-stop', 'left-forward and left-stop', 'right-forward and right-stop' and 'left-forward'.

'Michael' and 'Gorilla'. Both 'Michael' and 'Gorilla' data have been generated by rigging TOSCA high-resolution meshes (Bronstein et al., 2008) to the same walking skeleton provided by 3ds Max Studio (3dS MAX L&T CD., 2006). In this way, the two deforming meshes perform identical 'walking' motion. The temporal segmentation results of 'Michael' and 'Gorilla' are shown in Figure 4-5. As noted by Zhou et al. (Zhou et al., 2013) in their recent temporal segmentation work on motion capture

data, a ‘walking’ motion can be decomposed into 4 types of sub-motions: ‘left-stop’, ‘left-forward’, ‘right-forward’ and ‘right-stop’. By following this convention, we name each of the obtained temporal segments as follows : ‘right-stop’, ‘left-forward and left-stop’, ‘right-forward and right-stop’ and ‘left-forward’. Sample frames of each temporal segment are shown in Figure 4-12. As can be seen, our method decomposes the motions performed by both ‘Michael’ and ‘Gorilla’ into the same sub-motions, despite their shape differences.

‘Face1’ and ‘Face2’. Both of ‘Face1’ and ‘Face2’ data have been created by using a commercial motion capture system Vicon (http::Vicon). For a subject attached with reflective markers, Vicon system uses a set of cameras to track the trajectory of each marker and reconstruct these markers in 3D space. To collect human facial expressions, we have attached about 160 reflective markers on each subject’s face (see Figure 4-6), and use the Vicon system to capture facial movements. Sample meshes of the captured facial animations ‘Face1’ and ‘Face2’ are shown in Figure 4-8 and Figure 4-9, respectively. To validate the consistency of our approach, we ask each subject to perform the following facial expressions in order: three times of ‘eyebrow-raise’, ‘anger’, ‘disgust’, ‘fear’, ‘happy’, ‘surprise’, ‘sad’, with a ‘neutral’ expression as an interval between each. Note that the facial expressions in the two data are not temporally synchronized, and the markers on the two faces are not spatially coordinated either.

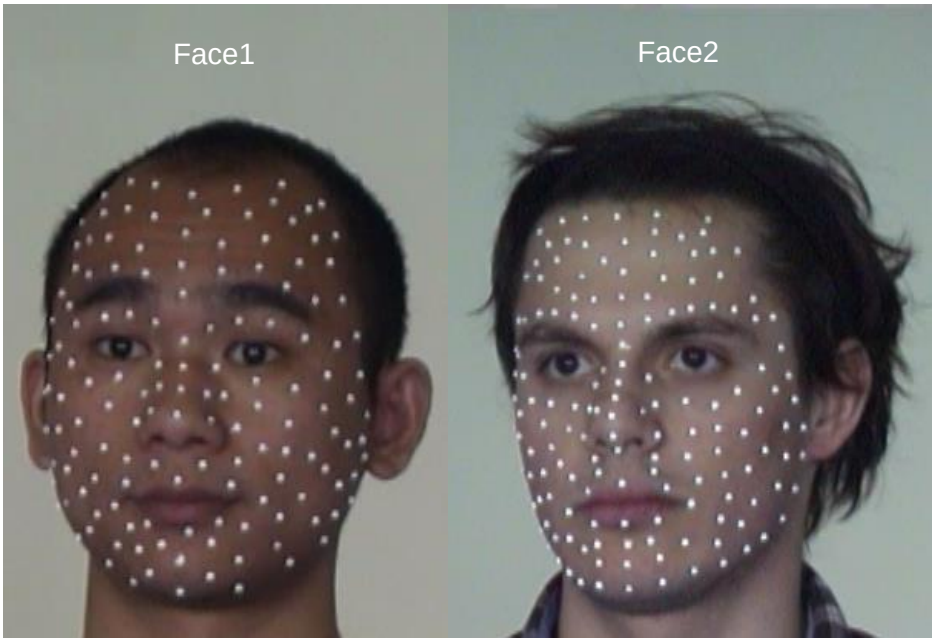


Figure 4-6 Capturing facial expressions with markers for 'Face1' and 'Face2'.

The temporal segmentation results of the two facial animations are both shown in Figure 4-7. Although 'Face1' and 'Face2' have different number of frames and different durations, we obtain 13 segments for both meshes which correspond to the following facial expression sequences: (1) 'eyebrow-raise', (2) 'anger', (3) 'neutral', (4) 'disgust', (5) 'neutral', (6) 'fear', (7) 'neutral', (8) 'happy', (9) 'neutral', (10) 'surprise', (11) 'neutral', (12) 'sad', (13) 'neutral'. Sample frames of the facial expression in each temporal segment for 'Face1' and 'Face2' are shown in Figure 4-8 and Figure 4-9, respectively. Quite encouragingly that we have obtained consistent temporal segmentation results between the two facial expressions.

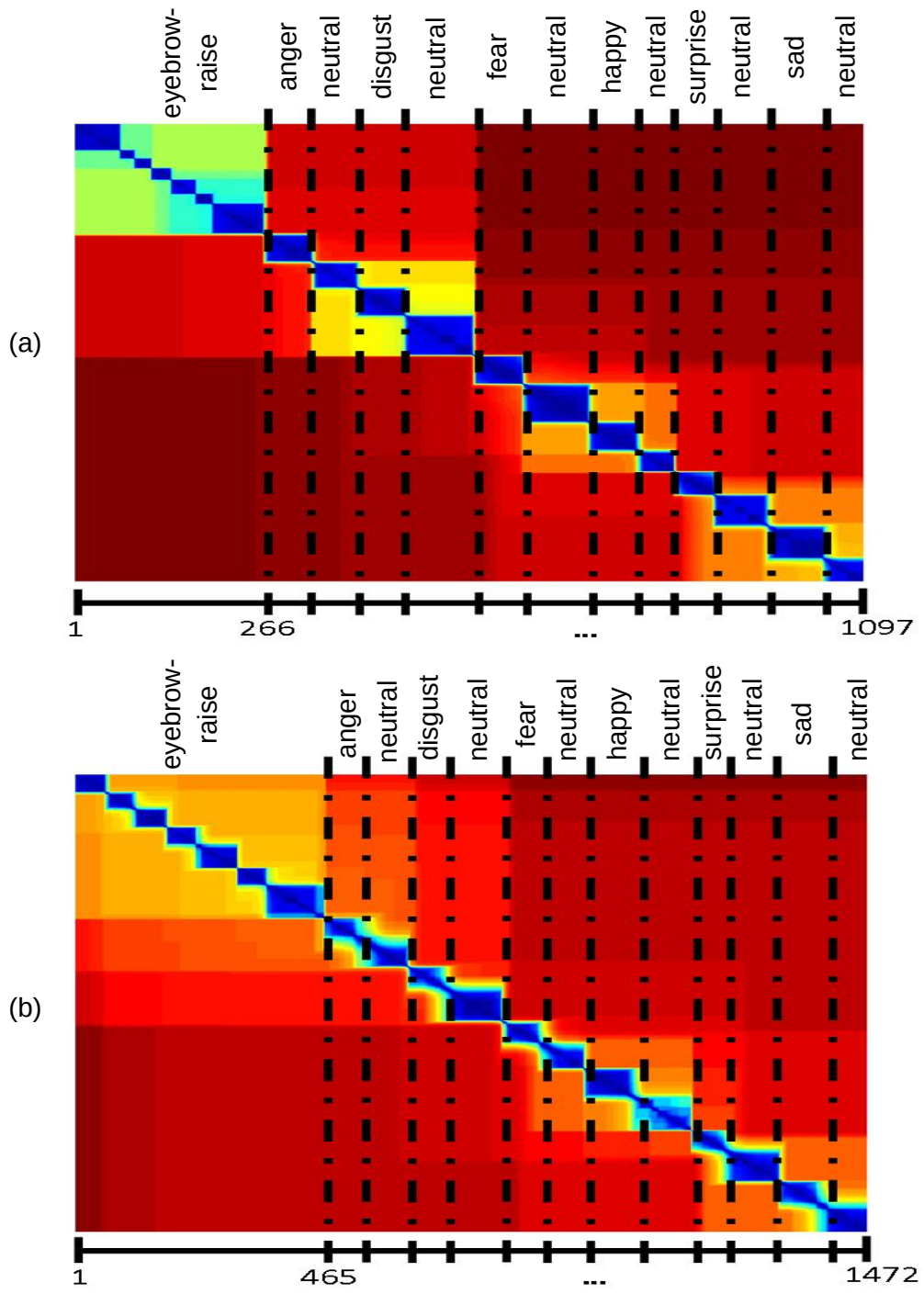


Figure 4-7 Temporal segmentation results of (a) 'Face1' and (b) 'Face2', which are both segmented into facial expressions 'eyebrow-raise', 'anger', 'neutral', 'disgust', 'neutral', 'fear', 'neutral', 'happy', 'neutral', 'surprise', 'neutral', 'sad', and 'neutral'.

Note that in the first temporal segment of 'eyeborw-raise' motion, it contains three times of 'eyeborw-raise' motions with 'neutral' expressions in between. Such atomic motions can be further segmented by using a different threshold θ_1 (see Sec-

tion 4.2). We will discuss about the segmentation for the atomic motions of facial expressions in the next section.

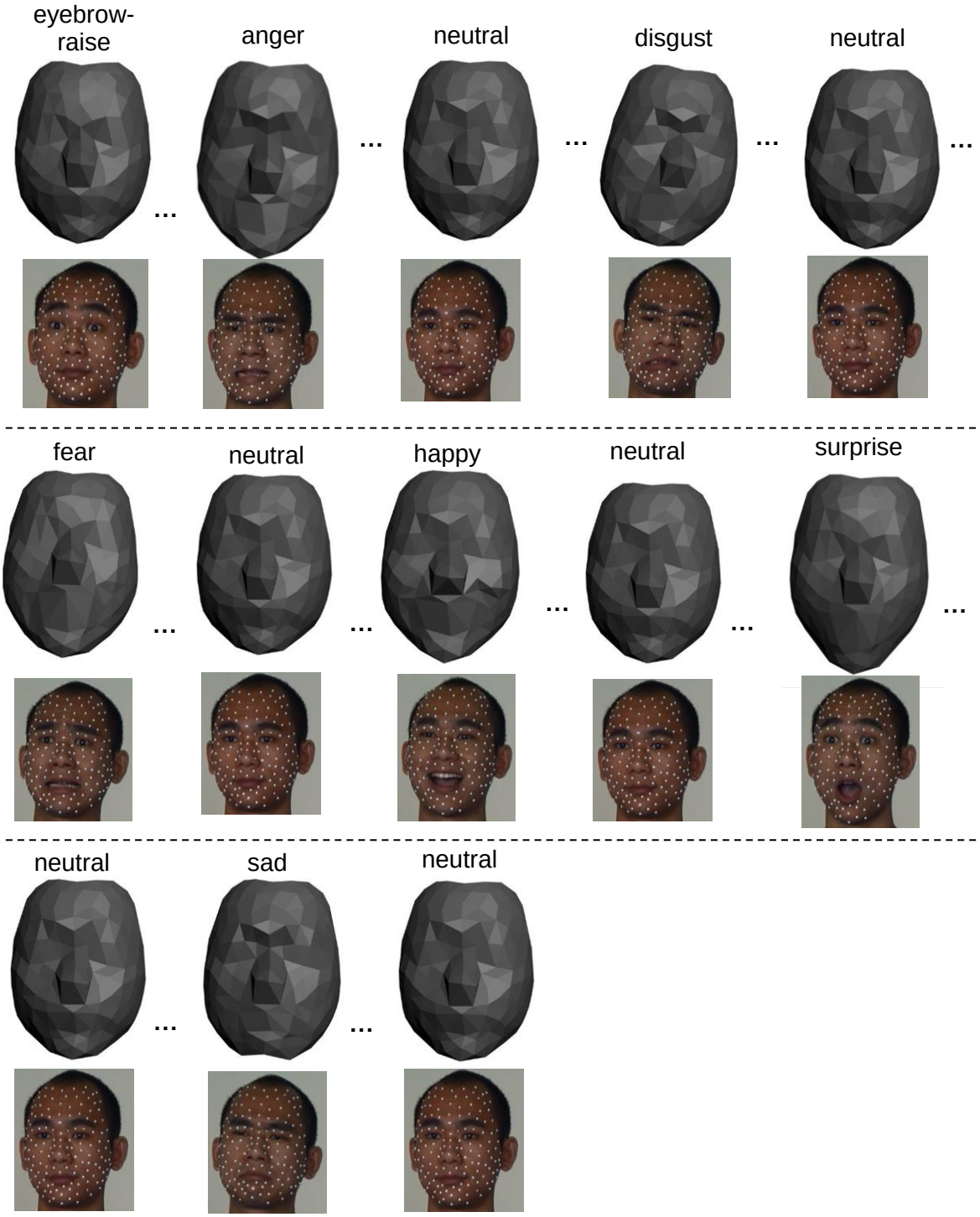


Figure 4-8 Sample meshes within each temporal segment of 'Face1'.

Figure 4-9 Sample meshes within each temporal segment of 'Face2'.

4.3.3 Discussion on the threshold

As have been presented in Section 4.2, the number of temporal segments obtained by using our method is determined by a user specified threshold θ_1. Intuitively, this user parameter determines the maximally allowed average dissimilarity within each temporal segment.

Figure 4-10 shows the temporal segmentation results of 'Face1' by using two different user thresholds, $\theta_1 = 0.6$ (top) and $\theta_1 = 0.45$ (bottom). The comparisons between these two results can be summarized as follows:

- Although the two temporal segmentation results have different boundary frames, they both successfully segment the motion sequence of 'anger', 'disgust', 'fear', 'happy', 'surprise' and 'sad', with 'neutral' expressions between them.

- The temporal segmentation result by using lower threshold, Figure 4-10(b), further returns the atomic motions in the first temporal segment (Figure 4-10(a)), which are (1) 'neutral', (2) 'eyebrow-raise', (3) 'neutral', (4) 'eyebrow-raise', (5) 'neutral', (6) 'eyebrow-raise', (7) 'neutral'.

To compute the temporal segmentation with atomic motions, instead of using a global threshold over the entire sequence, we can also recursively compute the temporal segments with different thresholds. That is, for example, after obtaining the temporal segmentation result shown in Figure 4-10(a), we repeat our temporal segmentation method on each temporal segment with a lower threshold to acquire the atomic motions. The advantage of such approach is that it can generate temporal segments of different level of motions (motions like 'walking' is a high level motion, while 'left-forward' and 'right-forward' are atomic motions of 'walking'). However, it requires automatic determining, or pre-settings, of the thresholds for different level of motions, which will be challenging tasks. Another interesting improvement would be to learn the threshold from a set of human-based ground-truth temporal segmentation.

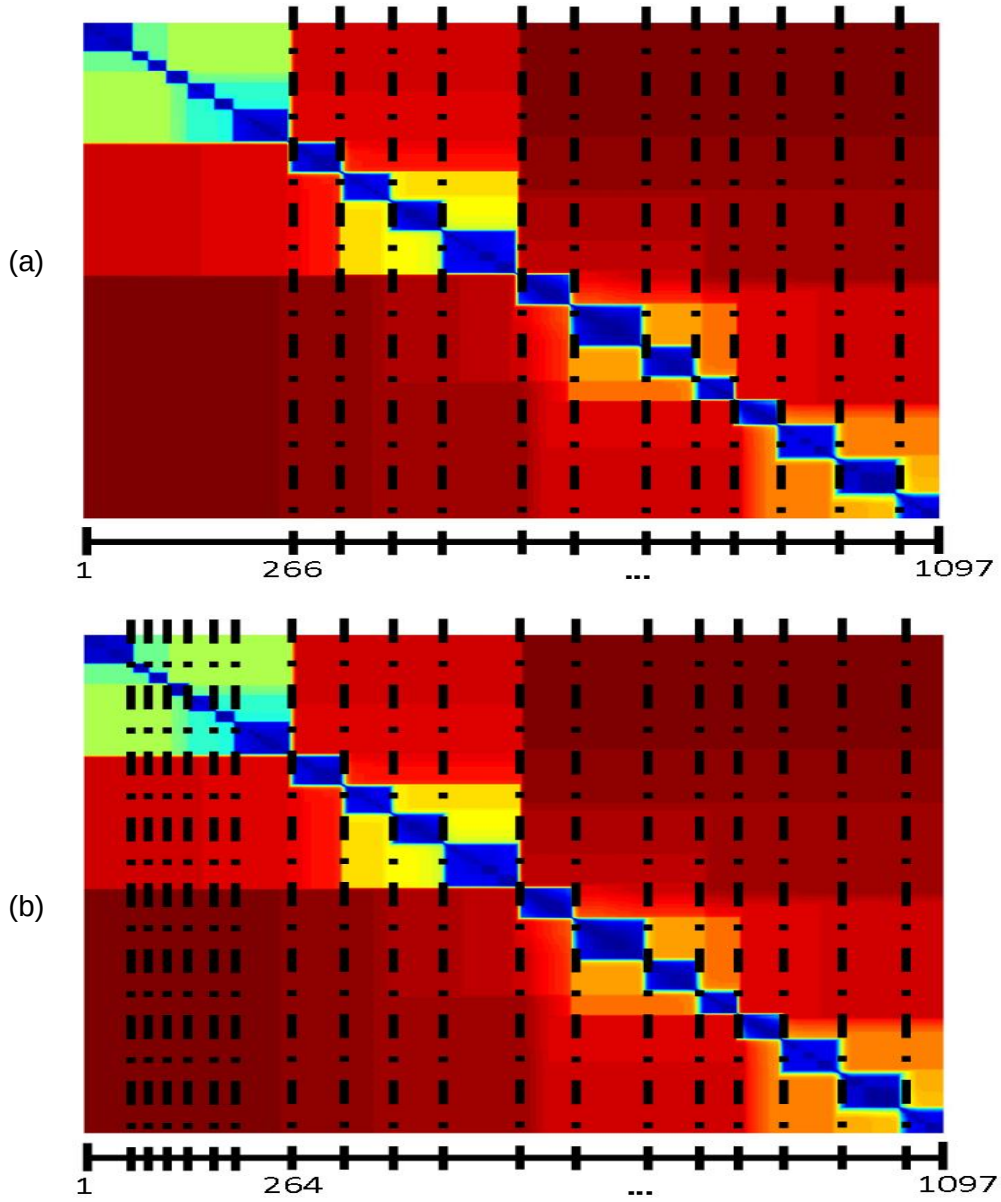


Figure 4-10 Temporal segmentation results of 'Face1' by using different thresholds (θ_1, Section 4.2), (a) $\theta_1 = 0.6$ (the same results in Figure 4-7(a)) and (b) $\theta_1 = 0.45$. Comparing to (a), the obtained results in (b) further divide the 'eyebrow-raise' into atomic facial expressions: 'neutral', 'eyebrow-raise', 'neutral', 'eyebrow-raise', 'neutral', 'eyebrow-raise', 'neutral'.

4.3.4 A comparison with Barbič et al.'s method

To validate the performance of our new temporal segmentation method, we have compared our method with a PCA-based motion segmentation method proposed for motion capture data by Barbič et al. (Barbič et al., 2004). In order to adapt their method for deforming meshes, one can do either of the followings: (a) Replace the joints in motion capture data with triangles as the primitives of each

deforming mesh, which means the dimensionality of each frame data is significantly increased since the number of triangles within each frame of a deforming mesh may be utterly larger than the number of joints of a motion capture data; (b) Extract the skeleton of a deforming mesh and apply their method on the skeleton sequence. For the sake of algorithm simplicity, we have chosen the approach (a). Barbič et al.'s method starts with a short initial motion segment to estimate the number of Principal Components (PCs). Then, for the successive frames that can be precisely reconstructed by using the estimated number of PCs, we merge them to the initial segment because they share similar intrinsic basis. Otherwise, a boundary frame is determined. The remaining motion sequence is segmented by repeating this procedure.

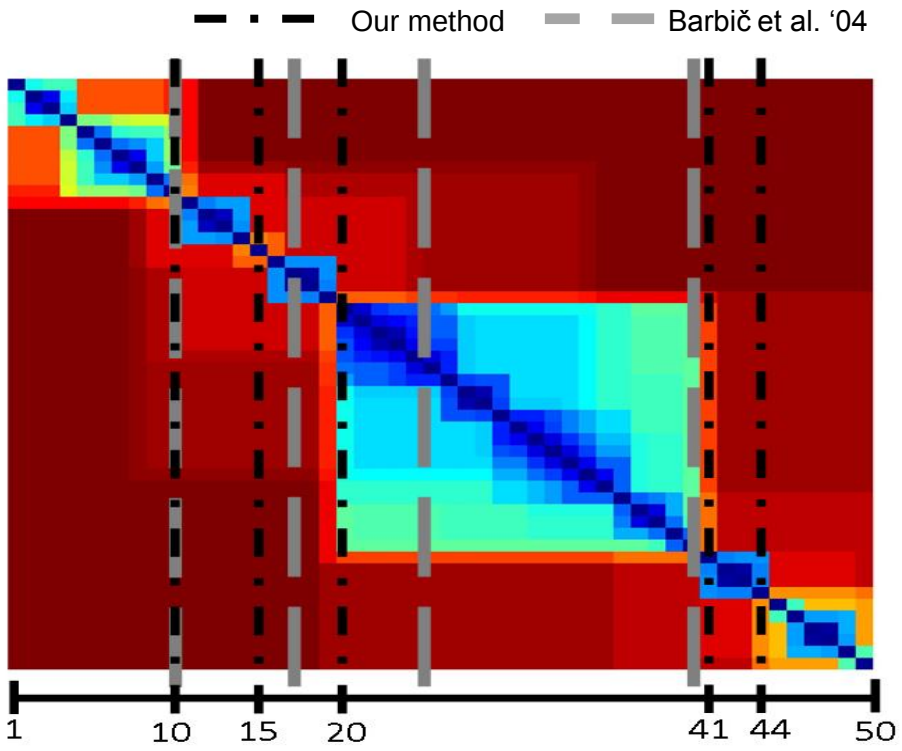


Figure 4-11 Temporal segmentation results of 'Camel' by using both our method and Barbič et al.'s method (Barbič et al., 2004).

In Figure 4-11, we have shown the temporal segmentation results by using both methods on 'Camel' data. Figure 4-12(b) shows selected meshes of the temporal segments 'run', 'head-right', 'head-left', 'run', 'head-down', 'head-up', obtained by using our method. To apply Barbič et al.'s method on 'Camel', we set the length of the initial segment as 7, so that it is sufficient for PCA method to capture the features of 'run' motion of 10 frames. However, because this length is longer than the durations of 'head-right' and 'head-left' (both have 5 frames), for which reason we

obtain boundary frames (f^{18} and f^{25}) shifted from the ground-truth (f^{15} and f^{20}). Worse, since the 'head-down' motion is too short (from f^{41} to f^{44}), the initial segment (from f^{41} and f^{47}) contains both 'head-down' and 'head-up' motions. This method cannot separate the two short motions because of the requirement of the minimum length of the initial segment. That is, this method over-fits to the initial segment. In order to avoid such drawback, Lin et al. (Lin et al., 2011) apply Barbič et al.'s method in both forward and backward directions, and unify the two boundary sets, which tends to result in over segmentation in many cases. In comparison, our method our method neither requires an initial segment nor has the limitation of the minimal length of segments.

Furthermore, Barbič et al.'s method takes inputs of joint angular data and the segmentation is based on the PCA analysis of the number principal dimensions of frames. Due to the significant increase of the computation of PCA on large dimensional data (Kambhatla and Leen, 1997), the application of their method would suffers from heavy computation. Comparing to 3.3 seconds by using our temporal segmentation method (see Table 4-1), Barbič et al.'s method consumes nearly 10 minutes.

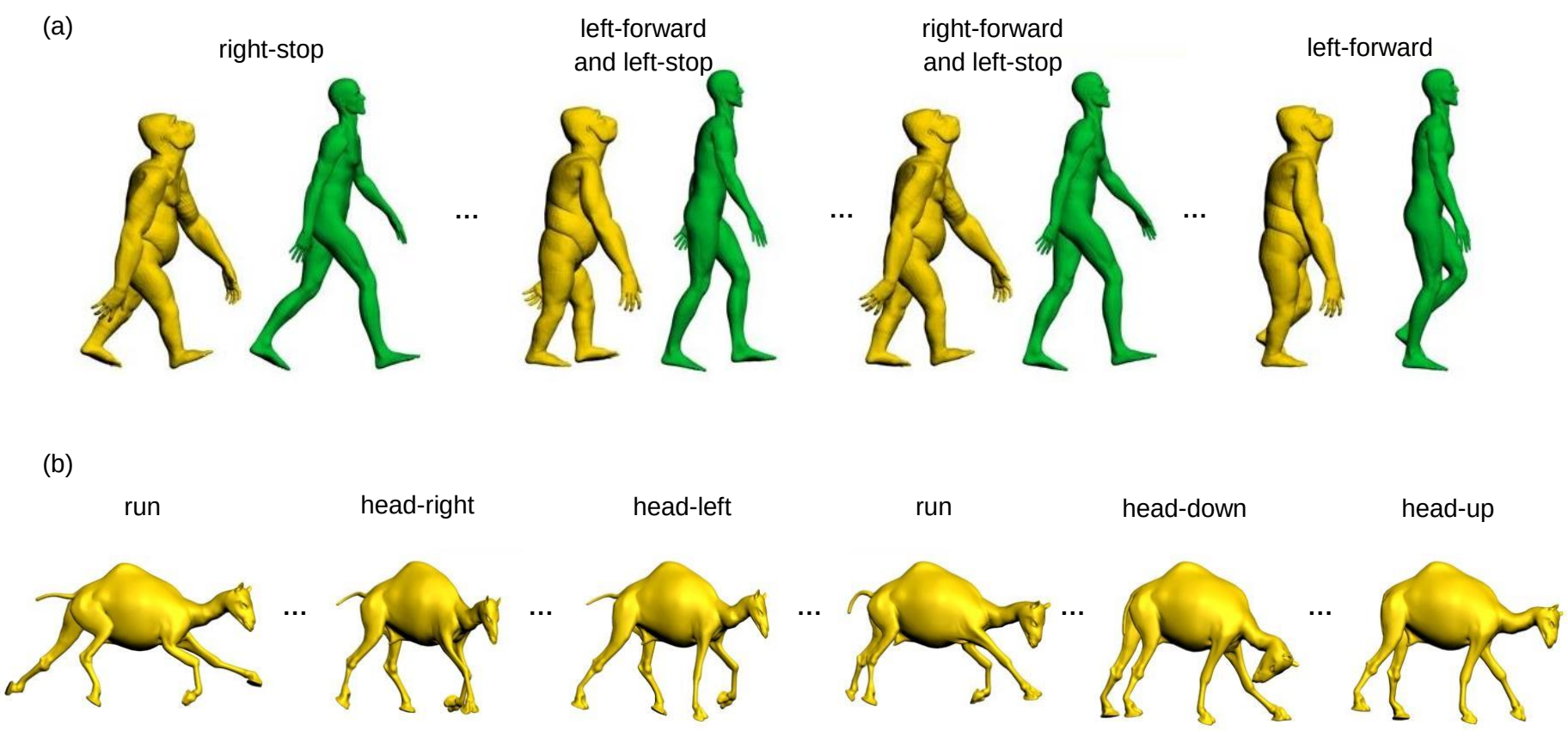


Figure 4-12 Sample meshes within each temporal segment of (a) 'Gorilla', 'Michael' and (b) 'Camel'.

4.4 Conclusion

4.4.1 Contributions

In this chapter, we present a method for the temporal segmentation of deforming meshes. We first compute the deformation of each triangle at each frame, i.e., rest-pose based *strain* values, by using the per-triangle dynamic feature descriptor described in Chapter 3. We then define a distance metric for each frame pair, and further define within-segment frame dissimilarity as the average of all possible pairwise frame distance within a candidate temporal segment. Finally, the boundary frames for the temporal segmentation are determined by minimizing the sum of within-segment frame dissimilarities. This optimization-based method allows to obtain the segmentation results where each temporal segment is a subsequence of similar frames, i.e., similar poses.

The contributions of this research can be summarized as follows:

- We devise a frame distance metric based on deformation behaviour by using our dynamic descriptor. The distance between two frames is small if the two frames undergo similar deformation with respect to the same rest-pose.
- We formulate the temporal segmentation into an optimization problem that minimizes the sum of within-segment dissimilarities. This allows to automatically determine the number of segments.
- Since the frame distance metric is defined based on the similarity of deformation behaviours, our temporal segmentation method allows to obtain consistent temporal segments with similar sub-motions among two deforming meshes showing identical or similar motions,.

In order to validate the effectiveness of our method, we have conducted a set of experiments with both synthesized and motion captured deforming meshes. The experimental results show that we can obtain consistent temporal segmentation for different deforming meshes exhibiting similar motions, despite their shape differences. Moreover, the comparisons with Barbič et al.'s segmentation method (Barbič et al., 2004) show that our method avoids over-fitting problem by taking into account of both forward and backward neighbouring frames, while Barbič et al.'s method only considers forward frames. A threshold θ_1 (see Section 4.2) is needed though, which can be properly provided by an experienced user.

4.4.2 Summary

In this chapter, we have proposed a new method for the temporal segmentation of deforming meshes based on the new dynamic descriptor presented in Chapter 3 We show the performance of our method with experiment results on both synthetic and motion captured deforming meshes, from which we obtain the same sub-motion sequences for the deforming meshes exhibiting identical or similar motions, despite their shape differences. Moreover, we observe more efficiency and effectiveness of our temporal segmentation method, by comparing with an existing temporal segmentation method.

Chapter 5 Spatio-temporal Segmentation of Deforming Meshes

We have presented a temporal segmentation method which divides a deforming mesh into temporal segments of similar poses in the previous chapter. However, that method has not taken into account of the spatial deformation coherency on the deforming mesh. To the best of our knowledge, the spatio-temporal segmentation of deforming meshes has not been studied by existing works. In Section 5.3 of this chapter, we present a spatio-temporal segmentation method, which investigates both the spatial and temporal deformation coherency simultaneously by using the dynamic feature descriptor described in Chapter 3. Furthermore, we extend the segmentation results towards the application of motion similarity measurement between deforming meshes (see Section 5.4). This application is particularly interesting, because it partly alleviates the challenge of the evaluation of segmentation results. By evaluating the computed motion similarities among deforming meshes with respect to human-based ground-truth, we indirectly validate the quality of the segmentation results (see Section 5.6).

5.1 Background

In this section, we start by reviewing several spatio-temporal segmentation techniques that have been developed in Commputer Vision field. Next, we review several studies in Computer Network field about *Evolving Graph*, which can be used as a compact representation of spatio-temporal segmentation results. Then, we introduce two existing methods for measuring the similarities among spatio-temporal sements.

5.1.1 Spatio-temporal segmetnation techniques in Computer Vision field

Spatio-temporal segmentation is a primary step of video analysis for the applications such as scene interpretation and video understanding. Megret and Demen-

thon (Megret and Dementhon, 2002) give a survey of the spatio-temporal segmentation techniques on video data, among which graph-based segmentation (Grundmann et al., 2010) (Tian et al., 2011) is the most popular method type.

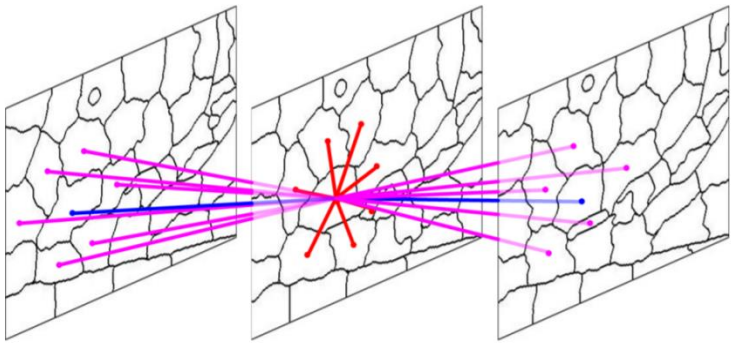


Figure 5-1 The spatial and temporal neighborhoods in the 3D graph cut model (Tian et al., 2011). The red lines show spatial neighborhoods, the purple and the blue lines show temporal neighborhoods.

Grundmann et al. (Grundmann et al., 2010) propose a hierarchical graph-based algorithm for segmenting long video sequences. This method first initializes a video as a graph, whose nodes are pixels and edges are the spatial and temporal neighborhoods. Then, the segmentation is done by using a region-growing method that merges both the spatial and temporal neighboring nodes with similar colors. However, this method suffers from the poor efficiency because of the high dimensionality of the initial graph, and over-segmentation due to local noises. To address these problems, Tian et al. (Tian et al., 2011) propose an improved graph cut segmentation method: They first compute the initial over-segmentation of each frame independently by using an existing efficient segmentation method proposed by Comaniciu and Meer (Comaniciu and Meer, 2002). Then , they construct an initial graph, whose nodes represent spatial segments within each frame and whose edges the immediate spatio-temporal neighborhoods (see Figure 5-1). Finally, the segmentation can be done by merging similar neighboring nodes both spatially and temporally, where each node is represented with the statistical model of pixel colors.

5.1.2 The studies of Evolving Graphs in Computer Network field

A computer network can be represented as a graph, where each node represents a computer, a server or a web browser, and each edge represents the communication between two nodes. An *Evolving Graph* can then be defined as a graph changes over time, either graph nodes or edges. Based on the graph representation, a large quantity of applications has been developed such as the detection of the

most correlated sub-networks (Chan et al., 2008) (Bilgin and Yener, 2006), querying or mining (Desikan and Srivastava, 2004) (Kan et al., 2009) (Robardet, 2009) (Berlingerio et al., 2009) (Yang et al., 2014) (Sun et al, 2007), and dynamic graph compression (Liu et al, 2012), etc.

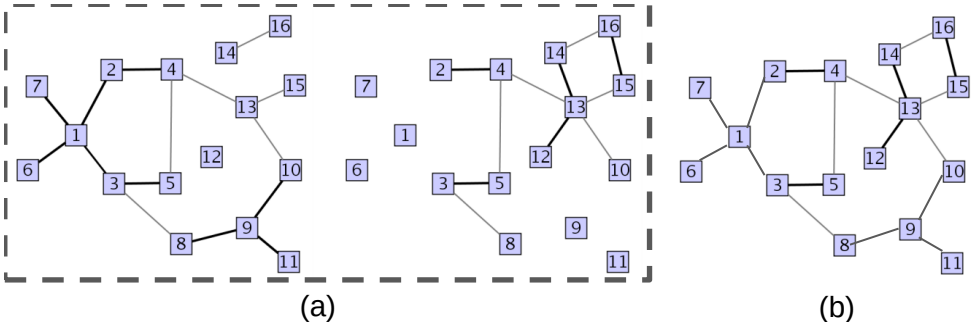


Figure 5-2 (a) Sample graphs in an evolving graph and (b) their union graph (Chan et al., 2008).

Chan et al. (Chan et al., 2008) proposed a method to discover regions in a computer network that have correlated spatio-temporal changes, i.e., the edges evolve synchronously. To achieve this goal, the authors proposed two distance metrics between graph edges: spatial distance and temporal distance. To compute the spatial distance between two edges, they compute the union graph over the evolving graph and then the shortest path distance between the two edges are taken as the spatial distance (see Figure 5-2). To compute the temporal distance between two edges, the authors first define a transition sequence that represents the changes of an edge in a binary waveform as a sequence of transitions (see Figure 5-3). Based on this representation, they can compute the temporal distance between two edges by using Euclidean distance. Finally, Chan et al. compute the spatio-temporal segmentation by merging the neighbouring edges that have either small spatial distance or temporal distance.

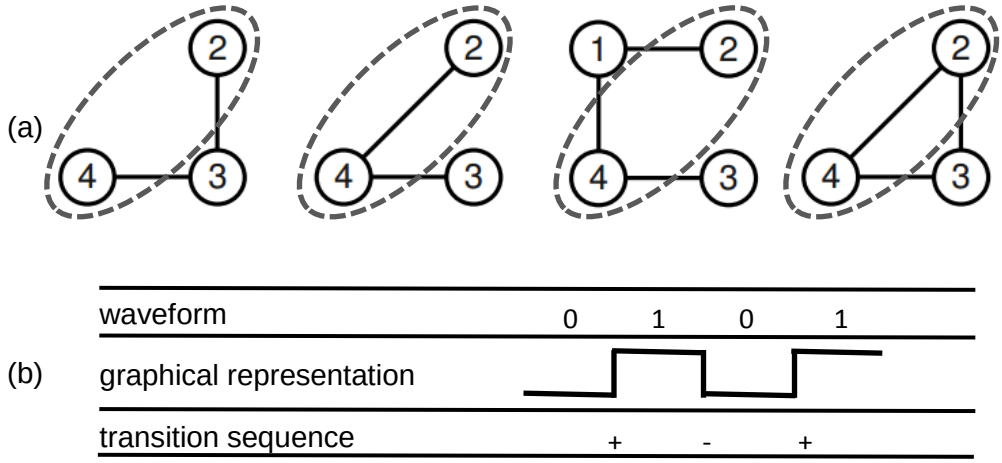


Figure 5-3 Sample waveform and the corresponding transition sequence of an evolving edge (Kan et al., 2009).(a) An evolving graph, (b) the corresponding waveform, graphical representation and transition sequence representation of the evolving edge in the dashed circle in (a).

Similarly, Kan et al. (Kan et al., 2009) also model the changes of a graph edge into transition sequence. Based on this representation, a user can query by a transition sequence for the sub-graphs that contain edge changes indicated by the input transition sequence. See an example in Figure 5-4.

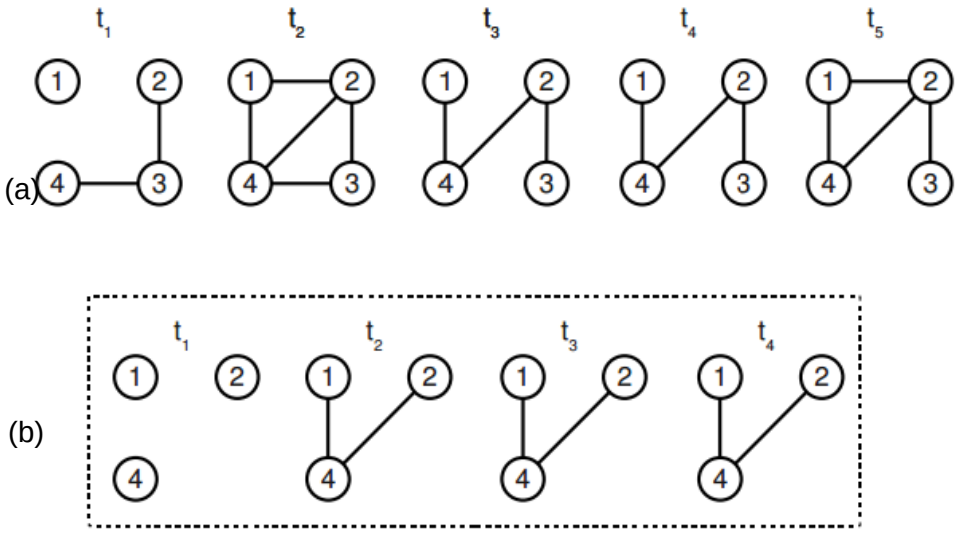


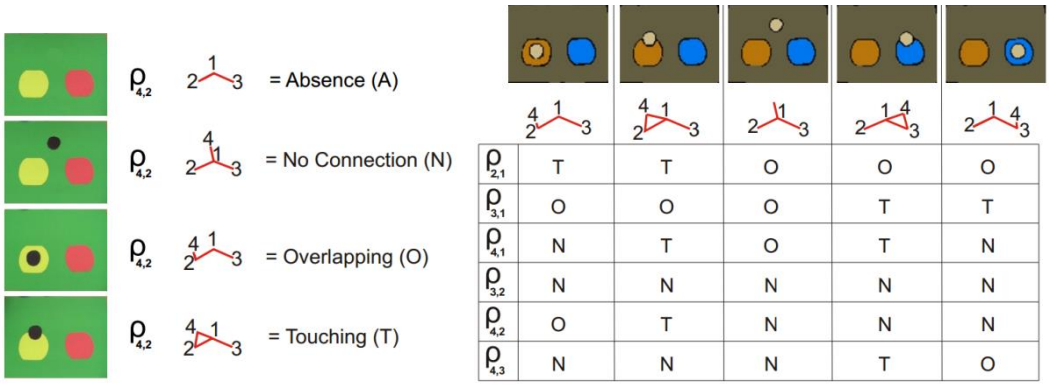
Figure 5-4 An example of graph query (Kan et al., 2009). By querying the maximal subgraph with waveform '0111' in (a), we obtain an evolving subgraph shown in (b), in which the edges '1-4' and '2-4' can be represented with the waveform '0111'.

5.1.3 Similarity measurement based on spatio-temporal segmentation

In our work, we aim to use our spatio-temporal segmentation results of deforming meshes to compare their differences. In Computer Network field, however,

although the spatio-temporal segmentation techniques can be used to discover correlated regions in an evolving graph (Chan et al, 2008), they are not applicable to compare two evolving graphs. In Computer Vision field, on the other hand, several distance measurement methods have been proposed for human action recognition.

Ryoo and Aggarwal (Ryoo and Aggarwal, 2009) compare two spatio-temporal segmentation results that have been obtained by using Dollár et al.'s method (Dollár et al., 2005) by computing the spatial and temporal relationships between the spatio-temporal segments : Assuming S_1 and S_2 are the spatio-temporal segments of two videos, respectively, the authors define the temporal relationships such as 'S_1 is *before* S_2' if the last frame of S_1 occurs before the first frame of S_2, and the spatial relationships such as 'S_1 is *near* S_2' if the pixel colors in S_1 is similar to the pixel colors in S_2. Based on these definitions, the method detects a similar human action if all the spatio-temporal segments of a sample video matches to those of a ground-truth video. However, this method is sensitive to noises such as light conditions, which affect the occurrences and duration of spatio-temporal segments and therefore influence the temporal relationship.



$\rho_{2,1}$	T	T	O	O	O
$\rho_{3,1}$	O	O	O	T	T
$\rho_{4,1}$	N	T	O	T	N
$\rho_{3,2}$	N	N	N	N	N
$\rho_{4,2}$	O	T	N	N	N
$\rho_{4,3}$	N	N	N	T	O

Figure 5-5 Representation of spatio-temporal segmentation results (Aksoy et al., 2010). The left column (a) shows 4 types of interactions between node 2 and 4. In the right column (b), from top to bottom, each row shows the spatio-temporal segmentation results, graph representation, and the interaction sequences of node pairs.

Aksoy et al. (Aksoy et al., 2010) and Luo et al. (Luo et al., 2011) propose another human action similarity metric by comparing the interactions between spatial-temporal segments. They first define four types of interactions (see Figure 5-5), then represent the interactions of each pair of segments as character labels (see Figure 5-5(b)). Finally, Luo et al. concatenate each row of the matric in Figure 5-5(b) for a character string, and achieve action recognition by comparing two strings by

using string kernels. However, this metric may be not effective for complex human actions because of higher complexity of pairwise interactions, which increases the dimensionality of the matrix in Figure 5-5(b) significantly, and therefore reduce the efficiency of this similarity metric.

5.2 Outline of our approach

Having introduced the temporal segmentation method in the previous chapter, we step further to investigate the spatial and temporal coherency simultaneously in deforming meshes. First, based on the degree of deformation of each triangle in each frame, we binarily label the triangles with either 'deformed' or 'rigid'. Then, we compute a spatio-temporal segmentation by merging the 'deformed' triangles that are either spatially or temporally connected. We then represent the spatio-temporal segmentation with *Evolving Graph*, where each node represents a spatial segment, each edge the neighborhood between two spatial segments, and each graph a subsequence of frames with the same graph representation.

Having computed the Evolving Graphs of two deforming meshes, we proceed to compute the similarity of Evolving Graphs: We first classify the similar graphs and assign the graphs in the same cluster with the same cluster labels. As a result, each evolving graph is represented with a graph cluster label sequence, which allows us to apply existing sequence alignments to find the optimized matching between the two sequences. Finally, we obtain an alignment score between the two cluster label sequences by using a local sequence alignment algorithm, which we use as the similarity between two deforming meshes.

The reminder of this chapter is organized as follows. In Section 5.3, we first present a spatio-temporal segmentation method for deforming meshes, and represent the segmentation results into Evolving Graphs. Next, in Section 5.4, we convert the graph sequences into graph cluster label sequences by using graph clustering, and then adopt a sequence alignment algorithm to measure the motion similarity between the two deforming meshes. Then, we validate the obtained similarity values with human-based ground-truth in our experiments, in Section 5.6. Finally, in Section 5.7, we discuss about (1) the differences of the segmentation results by using either the previous-pose based or the rest-pose based strains, and (2) the differences between our temporal segmentation and spatio-temporal segmentation results.

5.3 Spatio-temporal segmentation of deforming meshes

5.3.1 Spatio-temporal segmentation algorithm

We now describe our spatio-temporal segmentation algorithm that makes use of the dynamic feature descriptor based on the deformation behaviour of each triangle at each frame of the deforming mesh. Our goal is to obtain a compact representation of the segmentation results, which is done by adopting evolving graphs.

Having obtained the *strains* of each triangle at each frame by using the method described in Chapter 3, we start by labeling each triangle of each frame as either 'deformed' or 'rigid'. We chose binary labeling for the sake of simplicity although multi-way labeling could also work but with higher cost.

Note that we use previous-pose based strains, since this strain type reflects motion information. In Section 5.7, we provide a discussion about the influences of spatio-temporal segmentation results by using either previous-pose based or rest-post-pose based strains.

Given a deforming mesh $\mathcal{M}$ with M frames and N triangles, we represent each frame f^p $(p = 1, \ldots, M)$ with a vector of strain values $\boldsymbol{s}^p = (s_1^p, \ldots, s_N^p)^T$, which we obtain by the method described in Chapter 3. Next, all the triangles t_i^p $(i = 1, \ldots, N)$ in each frame f^p are binary labeled as 1 ('deformed') or 0 ('rigid'), by comparing their strain values to a threshold τ_s:

$$L_i^p = \begin{cases} 0, & if\ s_i^p < \tau_s. \\ 1, & \text{otherwise.} \end{cases}$$

The threshold τ_s has been fixed as 0.5 in our experiments.

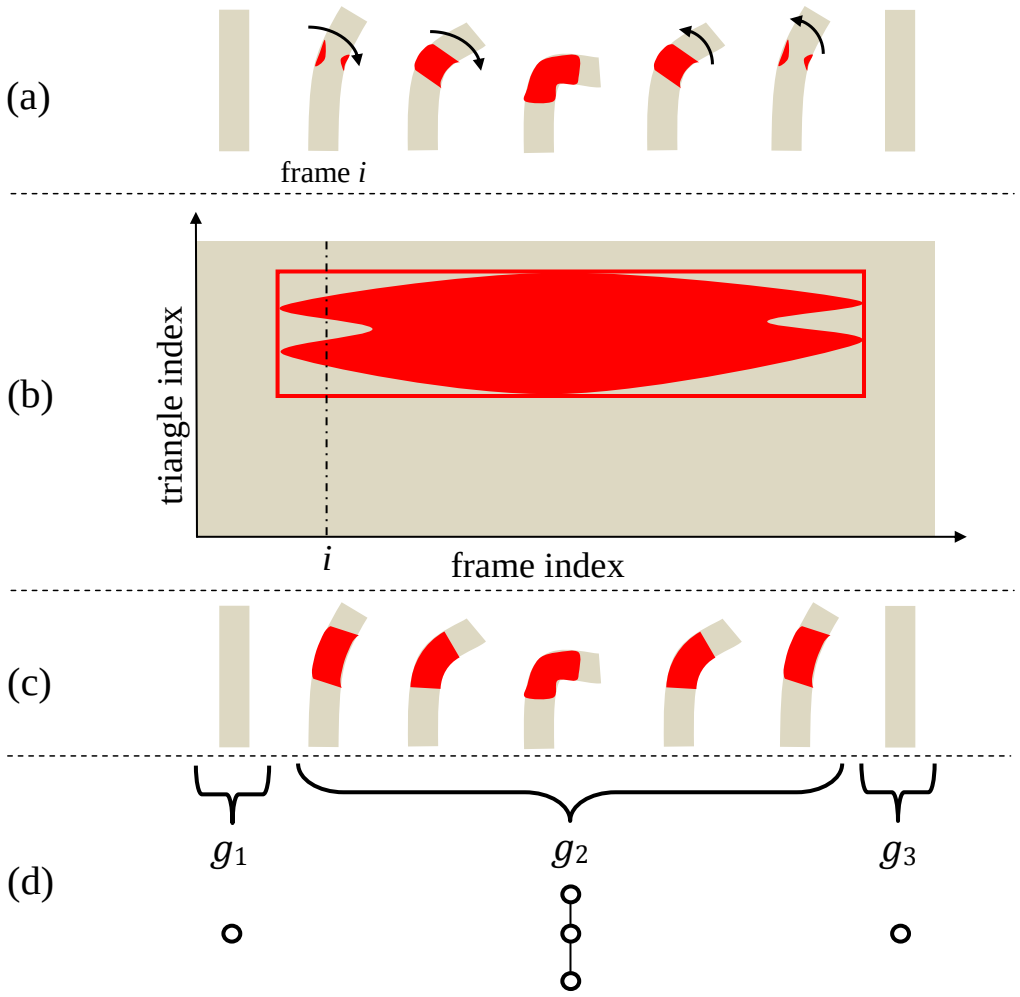


Figure 5-6: An example of spatio-temporal segmentation of 'bending-cylinder'. (a) Binary labeling. (b) Spatio-temporal segmentation. (c) Spatio-temporal segment. (d) Evolving graph representation.

Once we have the per-frame and per-triangle labeling, we carry out our spatio-temporal segmentation by finding triangles with identical labels that are adjacent along space or time. Figure 5-6(a) illustrates this idea with a 'bending-cylinder' example. The red regions represent the 'deformed' regions with a set of 'deformed' triangles that are connected. Figure 5-6(b) shows the 2D representation of the labeling, with the horizontal axis denoting the time (frame index) and the vertical axis denoting the space (triangle index). The two small disconnected 'deformed' regions at frame *i* (the dashed vertical line) are merged into one region in the later frames. We consider these two regions as belonging to the same spatio-temporal segment because they are resulted from the same *deforming action*. A spatio-temporal segment is computed as follows: We start with any 'deformed' triangle, and apply a region-growing algorithm to merge the 'deformed' triangles that are adjacent ei-

ther along space or time. See the red area in Figure 5-6(b). Then, we compute the space interval and the time interval of the merged area (rectangle in Figure 5-6(b)), and take all the triangles in that interval as a spatio-temporal segment. The obtained spatio-temporal segment can be seen as a solid rectangle in 2D, see Figure 5-6(b), and is also shown on the mesh in Figure 5-6(c).

We continue the above procedure until all the 'deformed' triangles have been merged into spatio-temporal segments. The complete spatio-temporal segmentation algorithm is shown in Algorithm 5-1, which runs in $O(M \cdot N)$ time. Note that the function **Neighbors**(**S**) returns the 'deformed' triangles that are adjacent in either space or time of every triangle in the triangle patch **S**.

Algorithm 2: *Spatio-temporal segmentation*

Input: $L_i^p, i = 1, \dots, N, p = 1, \dots, M$.

Init: S=T=I=P=∅;$\mathrm{ST}_i^p = \mathbf{0}, i = 1, \dots, N, p = 1, \dots, M$.

while $\exists i, p, L_i^p = 1$ **do**

S=L_i^p, T=**Neighbors**(S)-S.

while $\exists t, t \in \mathrm{T},\ L(t) = 1$ **do**

S=[S t], T= **Neighbors**(S)-S.

end while

while $\exists i, p, t_i^p \in \mathrm{S}$ **do**

$L_i^p = 0, \mathrm{S} = \mathrm{S} - t_i^p$.

$\mathrm{I} = [\mathrm{I}\ i], \mathrm{P} = [\mathrm{P}\ p]$.

end while

$\forall i \in \mathrm{I}, p \in \mathrm{P},\ \mathrm{ST}_i^p = 1$.

S=T=I=P=∅.

end while

Return: S, T

Algorithm 5-1 Spatio-temporal segmentation algorithm for deforming meshes.

Below we show the segmentation results of several deforming meshes in Figure 5-7, Figure 5-8 and Figure 5-9.

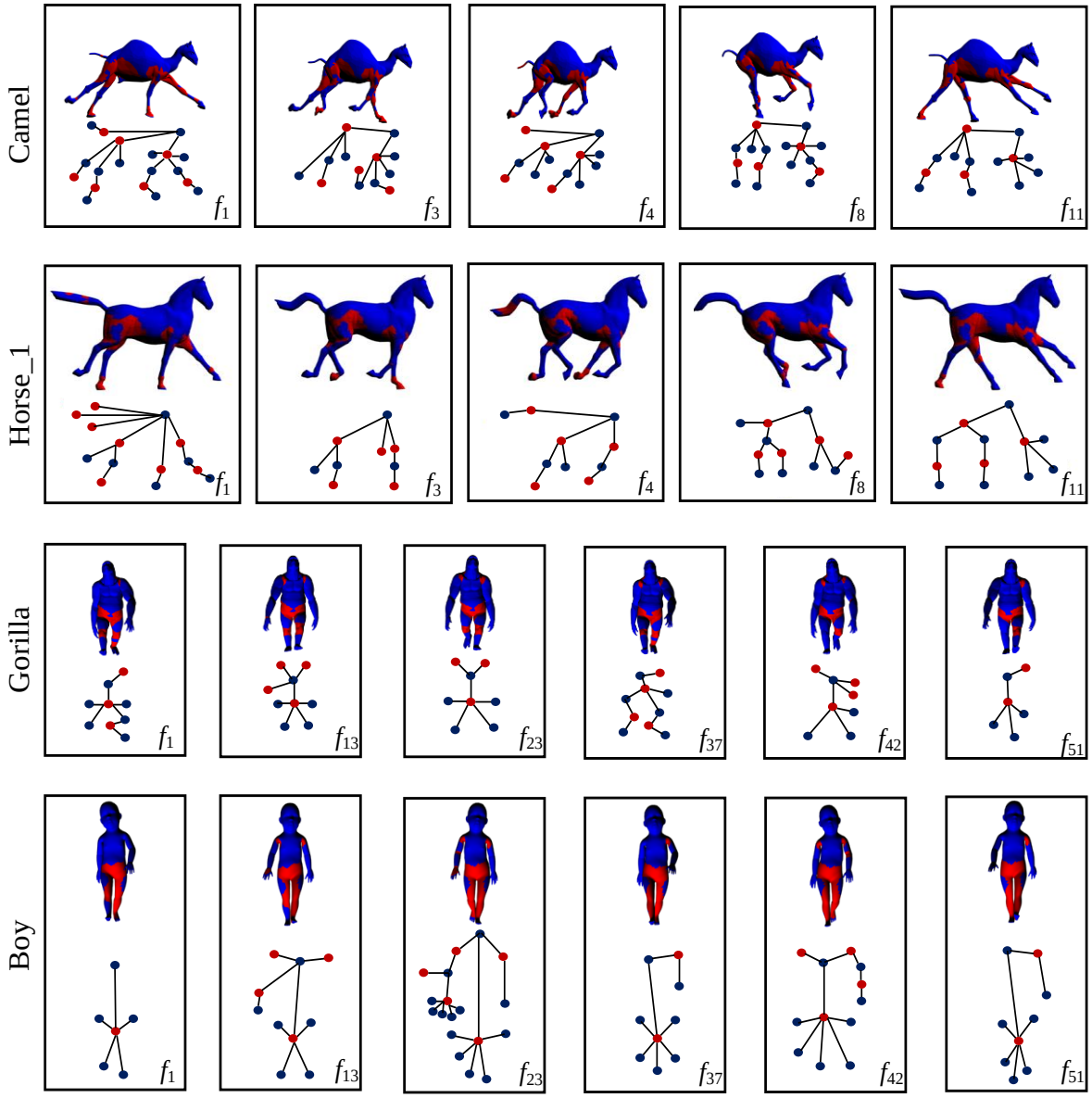


Figure 5-7 The spatio-temporal segmentation and the graph representation of 'Camel', 'Horse_1', 'Gorilla' and 'Boy'.

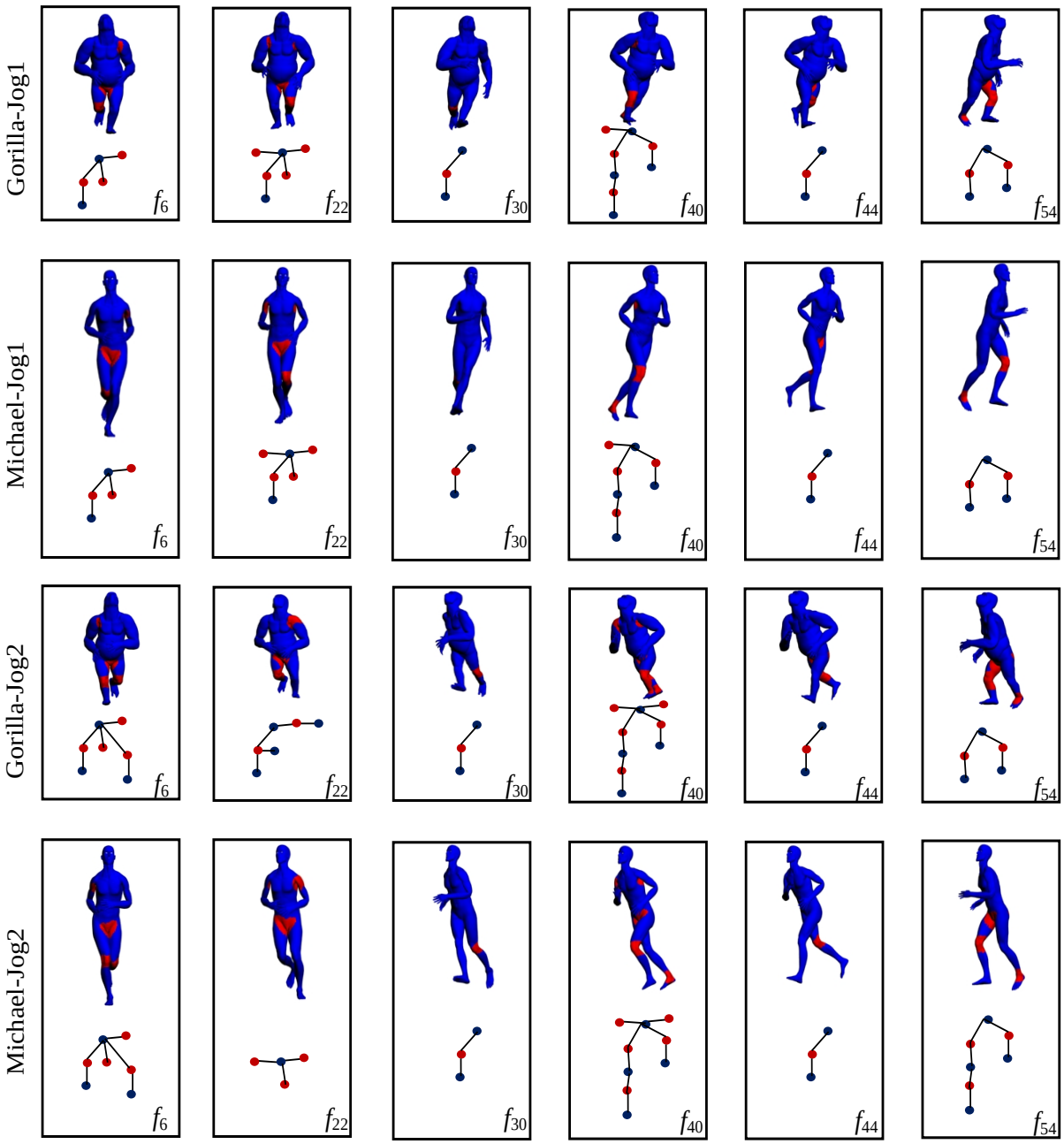


Figure 5-8 The spatio-temporal segmentation and the graph representation of 'Gorilla-Jog1', 'Michael-Jog1', 'Gorilla-Jog2' and 'Michael-Jog2'.

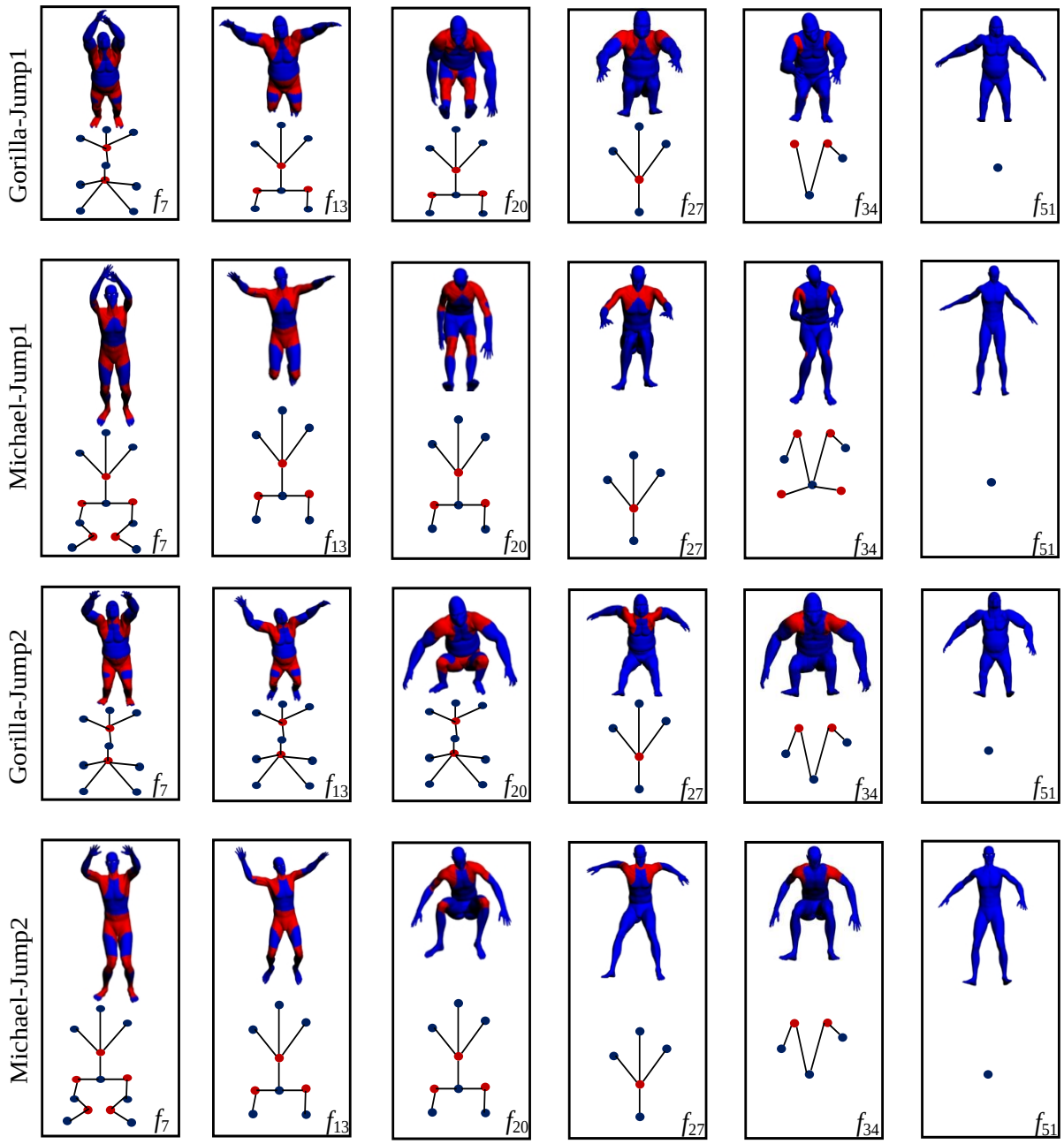


Figure 5-9 The spatio-temporal segmentation and the graph representation of 'Gorilla-Jump1', 'Michael-Jump1', 'Gorilla-Jump2' and 'Michael-Jump2'.

5.3.2 Evolving graph representation

Graph is a convenient and compact representation that has been broadly used for representing structured objects. To this end, we now describe our graph representation of the spatio-temporal segmentation results.

Given the spatio-temporal segmentation of a deforming mesh, we first extract key frames from the mesh sequence, where each key frame contains either the occurrence of a new spatio-temporal segment or the disappearance of a segment. Note that the first frame is always considered as a key frame. Then, for each key frame, we represent its spatial segmentation with a graph, where a node represents a spatial segment and an edge connects two spatially adjacent segments. A series of graphs of key frames that we obtain for a deforming mesh is an *evolving graph*, i.e., a graph that evolves over time. The evolving graph of the 'bending-cylinder' is shown in Figure 5-6(d).

5.4 Similarity measurement of deforming meshes

In this section, we present a method for measuring the similarity between two deforming meshes, which are represented with evolving graphs. We first cluster similar graphs of the two deforming meshes, thereafter graphs belonging to the same cluster are assigned with the same label. As a result, each deforming mesh is represented with a sequence of cluster labels. Then, we apply a local sequence alignment algorithm to compute the locally optimal alignment between the two cluster label sequences. Finally, we measure the similarity between the two deforming meshes by normalizing the sequence alignment score.

5.4.1 Graph clustering

Let $\mathcal{M}^A$ and $\mathcal{M}^B$ be two deforming meshes. By using the segmentation algorithm described in Section 5.3, we can generate the evolving graphs $G^A = \{g_1^A, g_2^A, \dots, g_{n^A}^A\}$ and $G^B = \{g_1^B, g_2^B, \dots, g_{n^B}^B\}$ for $\mathcal{M}^A$ and $\mathcal{M}^B$, respectively. Note that n^A and n^B are the number of graphs for the two evolving graphs, respectively.

To adopt sequence alignment algorithm for comparing two evolving graphs, we have to cluster the graphs and assign the graphs within the same cluster with the same label, to transform the evolving graphs into graph cluster label sequences. As having been discussed in Section 5.1.3, this operation not only filters out noises of segmentation results by labeling similar graphs as the same, but also increases the efficiency of the sequence alignment algorithm by reducing the number of element types (or alphabet size). Unfortunately, the graph clustering cannot be done by using existing clustering methods, such as K-mean clustering; this is because the graphs have different number of vertices and edges and these clustering methods work only for vectors of the same dimension. To avoid of such problem, we

adopted the *graph embedding* method proposed by Riesen et al. (Riesen et al., 2009a, 2009b). The purpose of graph embedding is to compute a mapping between the graphs and a vector space. The graph embedding method works as follows:

Given a set of graphs $G = \{g_1, g_2, \dots, g_n\}$ whose cardinality is n, the vector associated to a graph g_i is defined as $V_i = (d(g_i, g_1), d(g_i, g_2), \dots, d(g_i, g_n))^T$; $d(g_i, g_j)$ is a graph dissimilarity metric between the graphs g_i and g_j. In our case, the set G is the union of the sets of evolving graphs of $\mathcal{M}^{\mathrm{A}}$ and $\mathcal{M}^{\mathrm{B}}$, i.e., $G = G^A \cup G^B$, with its cardinality $n = n^A + n^{\mathrm{B}}$. The graph dissimilarity metric $d(g_i, g_j)$ can be calculated by using either the Maximum Common Subgraph (Bunke and Shearer, 1998) or the Graph Edit Distance (Gao et al., 2010). Below we introduce the definitions of both graph distance metrics.

Definition 5-1: Maximum Common Subgraph (MCS) (Gao et al., 2010). Let G_1 and G_2 be two graphs and $G_1' \subseteq G_1$, $G_2' \subseteq G_2$. If there exists a graph isomorphism between G_1' and G_2', then both G_1' and G_2' are called a common subgraph of G_1 and G_2. If there exists no other common subgraph of G_1 and G_2that has more nodes than G_1' and G_2', G_1' and G_2' are called a MCS of G_1 and G_2.

Definition 5-2: Graph distance metric based on maximum common subgraph (Bunke and Shearer, 1998). Let $mcs(G_1, G_2)$ be the maximum common subgraph of G_1 and G_2, and $max(G_1, G_2)$ be the graph with more nodes between G_1 and G_2. The distance of G_1 and G_2 is defined as

$$d(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)}.$$

Definition 5-3: Graph Edit Distance (GED) (Gao et al., 2010). The graph edit distance between g_i and g_j is defined as the minimum number of graph edit operations to transform g_i to g_j; these operations include additions and deletions of nodes and edges.

In this work, we use the latter graph distance metric GED (Definition 5-3). The computation of GED is an optimization problem that is usually addressed by using a tree search algorithm, whose objective is to find an edit path with minimum costs. The idea is to dynamically construct a search tree, where each node of the tree is a candidate graph edit operation and each leaf node represents the complete solution that a graph is transformed to the other. At each step during tree traversal, all the successor nodes are evaluated to find the route with lowest cost. Iteratively,

this method can compute the best approximation of the optimization objective. However, the running time and memory consumption increase exponentially with the increase of problem size, i.e., the number of vertices of graphs. To speedup the computation, Neuhaus et al. (Neuhaus et al., 2006) have proposed a simple modification : Instead of expanding all the successor nodes in the search tree, only a fixed number of the best of them are kept. Therefore, in each step during tree traversal, the number of successor nodes is limited, and the fast computation of GED is realized. We use this efficient algorithm for computing GED in our method.

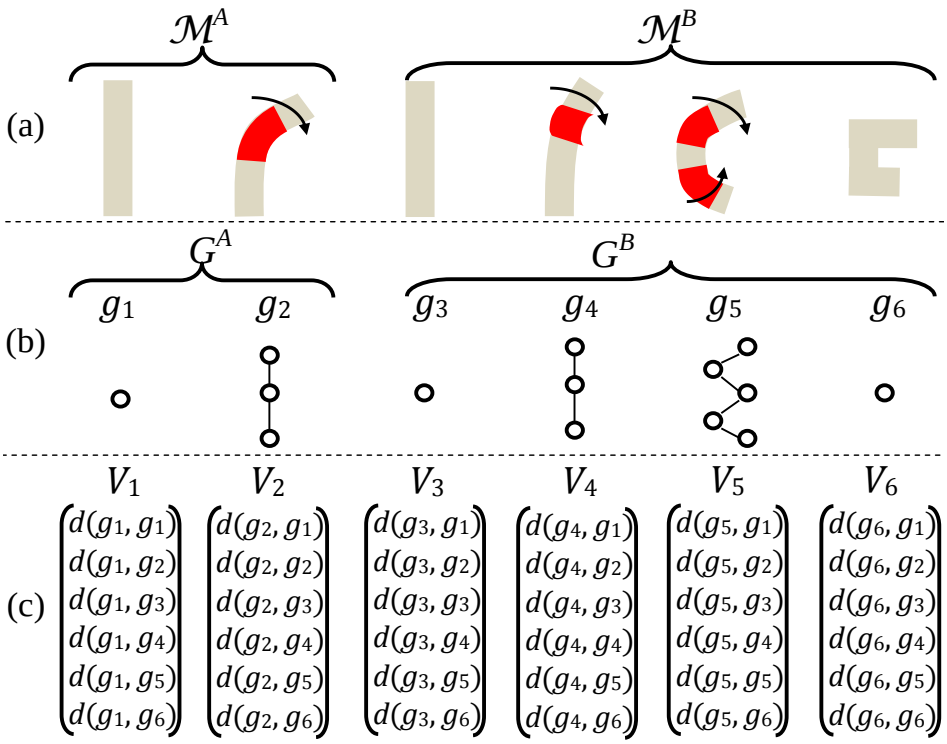


Figure 5-10: Graph embedding. (a) The input deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$. (b) The sequences of evolving graphs G^A and G^B. (c) The graph embedding. Each graph gi is represented with a vector V_i , where $d(g_i, g_j)$ denotes the graph edit distance between graphs g_i and g_j.

The overview of graph embedding algorithm is shown in Figure 5-10. After the graph embedding, each graph g_i is represented with a vector $V_i = \left(d(g_i, g_1), d(g_i, g_2), \ \dots, \ d(g_i, g_n)\right)^T, i \in [1,2, \dots, n]$.

Note that the size of the data produced by the graph embedding may be very large depending on the size of G. If G is composed of n graphs, the dimension of the output data is n^2 (n vectors V_i whose dimension is n). We apply PCA (Abdi and Williams, 2010) to reduce the dimension of the data. The PCA method uses orthogonal transformation to convert the set of vectors V_i into a set of values of linearly uncor-

related variables called principal components. Redundant information is removed by representing the vectors V_i with the top r principal components. Hence, each graph g_i is represented with a vector V'_i whose dimension is r.

Finally, we apply K-means clustering method on the vectors V'_i to cluster all the graphs g_i into K cluster; K is a user-specified parameter, which is chosen depending on the range of the deformation in $\mathcal{M}^{\mathrm{A}}$ and $\mathcal{M}^{\mathrm{B}}$. This value is set from 5 to 8 in our experiments. All the graphs g_i belonging to the k-th cluster ($k \in [1, \dots, K]$) are given the same cluster label ∂_k. Therefore, the deforming mesh $\mathcal{M}^{\mathrm{A}}$, which is represented with a sequence of evolving graphs $G^A = \{g_1^A, g_2^A, \dots, g_{n^A}^A\}$, is now be represented with a sequence of cluster labels $\partial^A = \{\partial_1^A, \partial_2^A, \dots, \partial_{n^A}^A\}$. A cluster label sequence $\partial^B = \{\partial_1^B, \partial_2^B, \dots, \partial_{n^B}^B\}$ is also computed for G^B. Although G^A and G^B contain different graphs, the same cluster label may appear in ∂^A and ∂^B. This is because the K-mean clustering has been computed on the union set $G^A \cup G^B$. Therefore, two graphs of G^A and G^B may belong to the same cluster, and thus be assigned with the same label.

In addition to the cluster labels ∂_k, we also compute the center of each cluster c_k, which is the mean vector of all the vectors V'_i whose corresponding graph g_i belongs to the cluster k. These cluster centers are required later to compute the sequence alignment, see Section 5.4.2.

Figure 5-11 shows an example of graph clustering for two deforming cylinder meshes. The deformation of $\mathcal{M}^A$ is composed of the bending of the center part of the cylinder. The deformation of $\mathcal{M}^B$ includes the bending of the upper and lower parts of the cylinder with the bending of upper part starting first. After graph clustering, both of $\mathcal{M}^A$ and $\mathcal{M}^B$ are represented with the cluster label sequences, ∂^A and ∂^B respectively (Figure 5-11(c)).

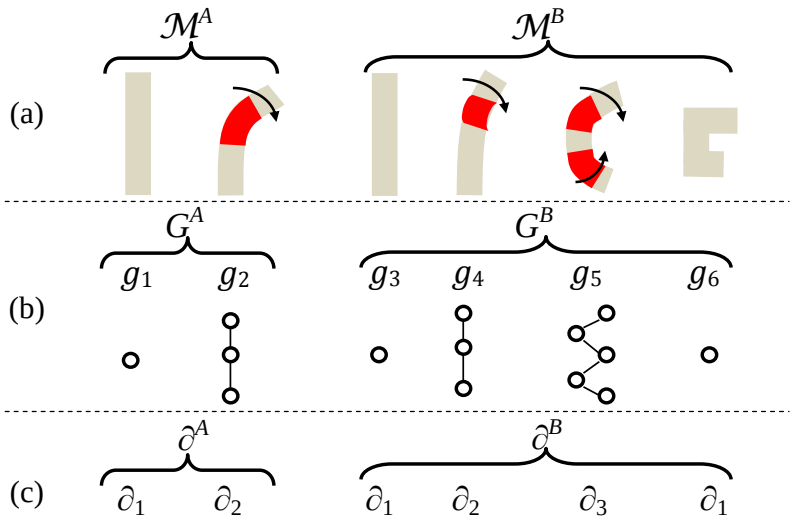


Figure 5-11: Graph clustering. (a) The input deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$. (b) The sequences of evolving graphs G^A and G^B. (c) The sequences of graph cluster labels ∂^A and ∂^B.

5.4.2 Local sequence alignment

Now that we have computed the cluster label sequences ∂^A and ∂^B of the deforming mesh $\mathcal{M}^A$ and $\mathcal{M}^B$, the next step is to compute the alignment between the two sequences ∂^A and ∂^B by matching identical subsequences between them.

Sequence alignment algorithm is commonly used in bioinformatics to identify similar regions among DNA sequences. The purpose of the alignment method is to locate and align the most similar subsequences between two DNA sequences, which allow gaps within the alignment. One of the most known methods is the Smith-Waterman algorithm (Smith and Waterman, 1981), which finds the optimal local alignment based on dynamic programming approach. It requires inputs of an affinity matrix between sequence items (or alphabets) and a gap penalty value. See an example in Figure 5-12.

In order to use the Smith-Waterman algorithm to compute the alignment between two cluster label sequences ∂^A and ∂^B, we first need to compute the affinity matrix of the clusters. As explained in Section 5.4.1, each of these cluster labels ∂_k corresponds to a cluster whose center is c_k. The cluster distance matrix D is a matrix whose size is K by K; each of its elements $D_{k_1k_2}$ is the distance between the cluster k_1 and k_2; it is calculated as the Euclidean distance between the cluster centers c_{k_1} and c_{k_2}, that is, $D_{k_1k_2} = \sqrt{c_{k_1} - c_{k_2}}$. The affinity matrix ϑ is a matrix whose dimension is K by K; each of its elements $\vartheta_{k_1k_2}$ is the affinity value between the clusters k_1 and k_2 and is computed as follows:

$$\vartheta_{k_1k_2} = \overline{D} - D_{k_1k_2} \; with \; k_1, k_2 \in [1, \dots, K] \tag{5-1}$$

where $\overline{D}$ is the average value of all the elements of the distance matrix *D*. Unlike the distance matrix, the affinity matrix has negative and positive values, where positive values indicate high level of affinities between the clusters and negative values indicate low affinities.

Once the similarity matrix has been computed, we use the improved Smith-Waterman algorithm proposed by Barton et al. (Barton et al., 1993). An implementation is available by using Matlab (http::Matlab). This algorithm takes as input the two cluster label sequences ∂^A and ∂^B with their corresponding similarity matrix ϑ; it generates a set of pairs of matching cluster labels $Q = \{\partial_i^A \leftrightarrow \partial_{\mathrm{Q}(i)}^B | i = 1, \dots, n_Q\}$, where $Q(i)$ indicates the label in ∂^B that is aligned to the i-th label in ∂^A, and T is the total number of non-matching cluster labels that are located among the matched ones. The set of matching pairs $\boldsymbol{Q}$ is computed such that the following matching score is maximized:

$$\delta_{\mathrm{AB}} = \sum_{\mathrm{i=1}}^{\mathrm{n_Q}} (\vartheta_{\partial_{\mathrm{i}}^{\mathrm{A}} \partial_{\mathrm{Q(i)}}^{\mathrm{B}}}) - \mathrm{T} \cdot \varepsilon, \tag{5-2}$$

where $\varepsilon = \beta \cdot \overline{D}$ is the penalty coefficient for the gaps occurring in the alignment; The coefficient β, which has been set to $1/6$ in our experiments, can be adjusted depending on how large gaps we want to allow (smaller β value will allow larger gaps and vice versa) in the alignment.

∂^A: $[\partial_1 \quad \partial_2]$

∂^B $[\partial_1 \quad \partial_2 \quad \partial_3 \quad \partial_1]$

D:

	∂_1	∂_2	∂_3
∂_1	0	0.5	1
∂_2	0.5	0	0.5
∂_3	1	0.5	0

Figure 5-12: The sequence alignment between ∂^A and ∂^B. Matching cluster labels are shown with dashed lines.

The matching score δ_{AB} is simply the summation of the similarity values $\vartheta_{\partial_i^A \partial_{\mathrm{Q(i)}}^{\mathrm{B}}}$ of each of the matching pairs of cluster labels subtracted by $T \cdot \varepsilon$, which is the penalty score of the gaps. Figure 5-12 shows an example of computing alignment score between ∂^A and ∂^B without a gap. Here the alignment score is $\delta_{AB} = \vartheta_{\partial_1^A \partial_2^{\mathrm{B}}} + \vartheta_{\partial_2^A \partial_2^{\mathrm{B}}} - 0 = 2$.

Although δ_{AB} computed in Equation 5-2 can be negative in theory, the algorithm that computes the matching score must return a non-negative result. This is be-

cause the empty set $Q = \emptyset$ is always taken into account when computing the most optimal alignment. In case of mismatching between ∂^A and ∂^B such that δ_{AB} is negative, the algorithm returns the empty set Q whose matching score is 0.

5.4.3 Time complexity

Let $\mathcal{M}^A$ and $\mathcal{M}^B$ be two deforming meshes whose evolving graph sequences are G^A and G^B. Let n^A , n^B and *n* be the numbers of graphs of G^A and G^B and the total number of graphs (i.e., $n = n^A + n^B$), respectively. We show the computation complexity of our method in Table 5-1. Our method involves computing the PCA whose time complexity is $O(n^3)$ (Abdi and Williams, 2010), followed by the K-means clustering whose time complexity is $O(n^{rK+1} \log n)$ (Inaba and Imai, 1994), with *r* being the number of principal components used for the PCA and *K* the number of clusters (see Section 5.4.1). Our algorithm also requires computing the sequence alignment whose time complexity is $O(n^A \cdot n^B)$ (Smith and Waterman, 1981) and the graph embedding whose time complexity is $O(n^2 \cdot T_{GED})$, with T_{GED} being the polynomial time for computing the graph edit distance (Neuhaus et al., 2006).

Table 5-1 Computation complexities of the used techniques.

Algorithms	*Complexity*
PCA	$O(n^3)$
K-means	$O(n^{rK+1} \log n)$
Sequence alignment	$O(n^2 \cdot T_{GED})$

5.5 Similarity measurement

The alignment score we have defined in Equation 5-2 relies on the lengths of sequences. For example, given two similar sequences, we obtain higher scores for longer sequences. In order to alleviate such problem, we normalize the alignment score as follows:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}}. \tag{5-3}$$

This normalized alignment score by using our similarity measurement method holds the following properties:

P.1 Non-negativity $\rho_{AB} \geq 0$.

As explained in Section 5.4.2, the matching score δ_{AB} is non-negative, and so is the value of ρ_{AB}. In the extreme case, a value of ρ_{AB} equals to 0 implies that no alignment has been found between the two sequences.

P.2 Symmetry $\rho_{AB} = \rho_{BA}$.

The alignment algorithm score in Equation 5-2 does not depend on the order in which the sequences are aligned. That is, the same pairs of matching cluster labels are found whether ∂^A is aligned to ∂^B, or ∂^B to ∂^A. It follows that δ_{AB} is equal to δ_{BA}, and therefore ρ_{BA} is equal to ρ_{AB}.

P.3 Boundness $\rho_{AB} \leq \sqrt{\frac{\delta_{AA}}{\delta_{BB}}} \leq 1$, assuming $\delta_{AA} \leq \delta_{BB}$.

According to Equation 5-2, the matching score increases as the number of matching pairs gets larger. It follows that $\delta_{AB} \leq \delta_{AA}$. This is because the number of matching pairs for ∂^A being matched to itself is always larger than or equal to those matched to ∂^B. It follows that:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} = \sqrt{\frac{\delta_{AB} \cdot \delta_{AB}}{\delta_{AA} \cdot \delta_{BB}}} \leq \sqrt{\frac{\delta_{AA} \cdot \delta_{BB}}{\delta_{AA} \cdot \delta_{BB}}} = 1.$$

If two input sequences are the same, i.e., $\partial^A = \partial^B$, we have $\rho_{AB} = 1$.

More strictly, the upper bound of ρ_{AB} is $\sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, assuming $\delta_{AA} \leq \delta_{BB}$. As explained above, $\delta_{AB} \leq \delta_{AA}$, thus it follows that

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} \leq \frac{\delta_{AA}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} = \sqrt{\frac{\delta_{AA} \cdot \delta_{AA}}{\delta_{AA} \cdot \delta_{BB}}} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}.$$

In a special case where ∂^A is a subsequence of ∂^B, based on the definition of the alignment score in Equation 5-2, we have $\delta_{AB} = \delta_{AA}$. Hence, $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$.

P.4 Subsequence Assuming $\delta_{AA} \leq \delta_{BB}$, if $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, we have $A \subseteq B$, i.e., ∂^A is a subsequence of ∂^B.

Given $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, it follows that:

$$\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}} = \sqrt{\frac{\delta_{AA} \cdot \delta_{AA}}{\delta_{BB} \cdot \delta_{AA}}} = \frac{\delta_{AA}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}}.$$

By comparing to the definition of ρ_{AB} in Equation 5-3, we have

$$\delta_{AA} = \delta_{AB}.$$

As explained in **P.3**, the alignment score between ∂^A and ∂^B being equal to the alignment score between ∂^A and ∂^A indicates that ∂^A is a subsequence of ∂^B, i.e., $\partial^A \subseteq \partial^B$.

To sum up, the properties **P.1** and **P.3** show that $\rho_{AB} \in [0\ \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}]$. A value close to 1 indicates that the two sequences ∂^A and ∂^B are similar, i.e., the two deforming meshes $\mathcal{M}^{\mathrm{A}}$ and $\mathcal{M}^{\mathrm{B}}$ perform similar motions. In addition, the properties **P.3** and **P.4** show that $\rho_{\mathrm{AB}} = \sqrt{\frac{\delta_{\mathrm{AA}}}{\delta_{\mathrm{BB}}}}$ is the necessary and sufficient condition for $\partial^A \subseteq \partial^B$, i.e., ∂^A is a subsequence of ∂^B.

5.6 Experimental results

In this section, we first introduce our experimental environment and data, and then present the pairwise sequence alignment results and the similarity measurement results between deforming meshes. For the sequence alignment results, we compare the performance by using our method and the other classical sequence alignment methods (see Section 5.6.2). For the experiments on similarity measurement, we evaluate our results by collecting and comparing with human-based ground-truth motion similarities (see Section 5.6.3.2). In Section 5.7, we continue with the analytical discussions of our method including: (1) the possibilities of extending GED with node/edge attributes, (2) the differences of the spatio-temporal segmentation results by using either the previous-pose based or the rest-pose based strains, and (3) the comparative discussions on the results between the spatio-temporal segmentation in this chapter and the temporal segmentation presented in previous chapter.

5.6.1 Experimental environment and data

The deforming meshes used in our experiments include both synthetic animations and motion capture sequences, which are summarized in Table 5-2. The three models ‘Michael’, ‘Gorilla’ and ‘Boy’ are generated by rigging TOSCA high-

resolution meshes (Bronstein et al., 2008) with the same walking skeleton provided by 3D Max Studio (3dS MAX L&T CD., 2006). The two other models, 'Head' and 'Face_1' are obtained by linear interpolation of 8 key poses (anger, fury, grin, laugh, rage, sad, smile and surprise) (Sumner and Popović, 2004). The two models 'Camel' and 'Horse_1' are from Sumner et al.'s work on Deformation Transfer (Sumner and Popović, 2004). The model 'Horse_2' is the same model as 'Horse_1' except that the frame rate and the starting pose are different. The two models 'Face_2' and 'Face_3' have been obtained with the motion capture of two person's facial expressions using the Vicon system (http:Vicon); these motion data have been used to animate the scanned faces of the two persons. These two models 'Face_2' and 'Face_3' contain the facial expressions in the following order: 'eyebrow-raise' for three times, 'anger', 'disgust', 'fear', 'happy', 'surprise', 'sad', and with a 'neutral' facial expression in between. Selected frames of several deforming meshes are shown in Figure 5-7.

Table 5-2 Used deforming meshes for spatio-temporal segmentation and timings.

Name	Nb. Of Triangles	Nb. Of Frames	Timings (Second)	Description
Camel	43778	48	6.2	Gallop animation
Horse_1	16858	48	2.8	Gallop animation
Horse_2	29984	80	4.1	Gallop animation
Michael	29999	54	2.0	Walk animation
Gorilla	29999	54	1.9	Walk animation
Boy	10146	54	0.9	Walk animation
Head	31620	80	2.3	Facial expression animation
Face_1	57836	80	1.0	Facial expression animation
Face_2	1171	1473	20.1	Facial expression motion capture
Face_3	1272	1064	8.4	Facial expression motion capture

All our algorithms have been implemented in Matlab, and the results are computed on a Windows PC with 3.4 GHz Intel Core i7-2600 processor, 4GB of RAM.

We first process each deforming mesh with our segmentation method to generate the sequence of evolving graphs for each of them. The computation time of the

segmentation of each data in a Matlab implementation has been shown in Table 5-2. Figure 5-7 shows several segmentation results we have obtained by using our algorithm. In each figure, 'deformed' segments are shown in red and 'rigid' segments in blue.

5.6.2 Frame alignment

One important by-product of computing sequence alignment is the frame alignment, i.e., optimal alignment of graphs (or the corresponding key frames). Having the evolving graph representation of deforming meshes, we apply Smith-Waterman algorithm for computing the sequence alignment between two sequences. In fact, there exist a variety of methods for computing sequence alignment, such as Dynamic Time Wrapping (DTW) and its variations. DTW is a well-known technique for computing optimal global temporal alignment between two sequences, which computes the sequential alignment of each element (a frame of a deforming mesh in our case) from a sequence to an element in the other sequence by minimizing total aligned element distance (Kruskall and Liberman, 1983). DTW has been commonly used for speech recognition (Turetsky and Ellis, 2003). One variation of the classical DTW algorithm is to include another constraint on the alignment rule that the first (and the last) elements of the two sequences are forced to be aligned to each other. This rule is especially useful when the input sequences are well synchronized. We name this modified classical DTW as mDTW in this work.

In order to apply DTW and mDTW for optimally aligning two deforming meshes, one needs to compute a frame distance matrix. To do this, for each deforming mesh, we first represent each frame with the key frame that is used to represent the subsequence containing this frame. That is, if a key frame is representing a subsequence with n' frames, the key frame repeats n' times in the sequence. We then compute the frame distance as graph edit distance since each key frame is associated with a graph representation.

We show the distance matrices between several deforming meshes in Figure 5-13, Figure 5-14, Figure 5-15, Figure 5-16, with color varying from blue to red indicating distance values from low to high. In these figures, we also show the comparisons of sequence alignment results by using Smith-Waterman, DTW and mDTW, where each of the alignment result can be seen as an alignment path on the similarity matrix. We describe the comparisons of the alignment methods between several deforming meshes as below.

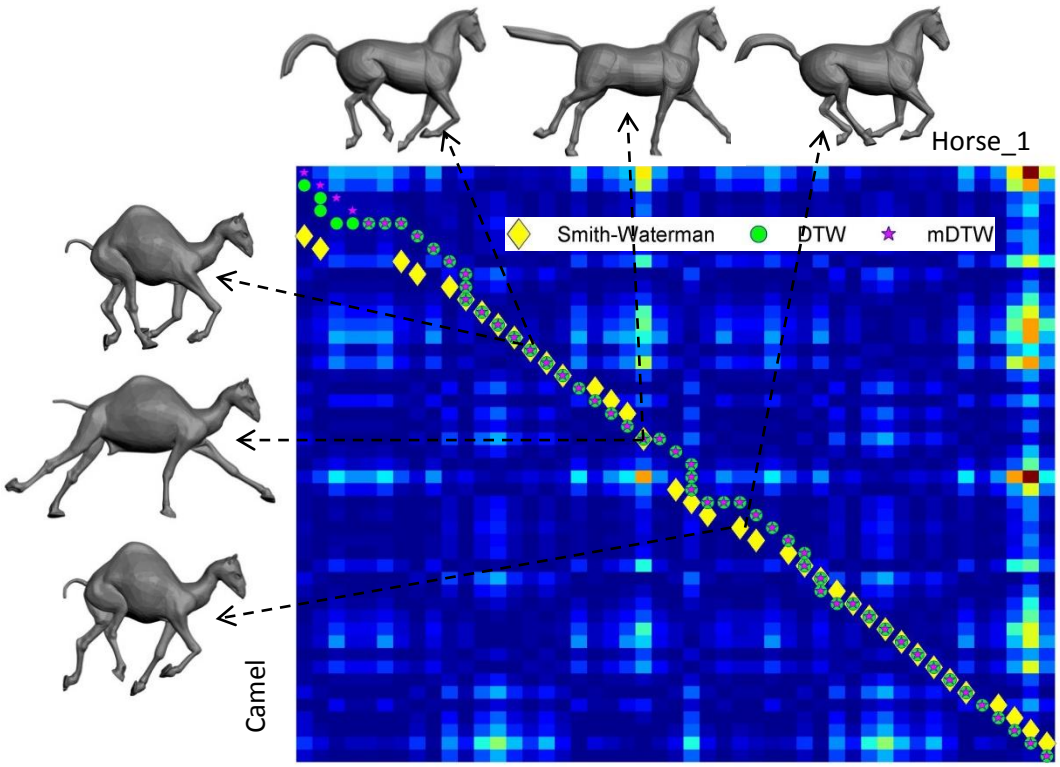


Figure 5-13 Comparisons of the sequence alignment results by using Smith-Waterman, DTW and mDTW for 'Camel' (vertical-wise) and 'Horse_1' (horizontal-wise).

Created by the deformation transfer (Sumner and Popović, 2004), the two mesh sequences 'Camel' and 'Horse_1' are time-synchronized and have the same number of frames and their corresponding frames have the same poses. In Figure 5-13, the alignment paths computed by using three alignment methods are all around the diagonal of the distance matrix, which correctly reflects the synchronized motions between the two deforming meshes. Note that because we force to match the first frames by using mDTW, we obtain slightly improved alignment path in the first 5 frames than the path computed by using DTW.

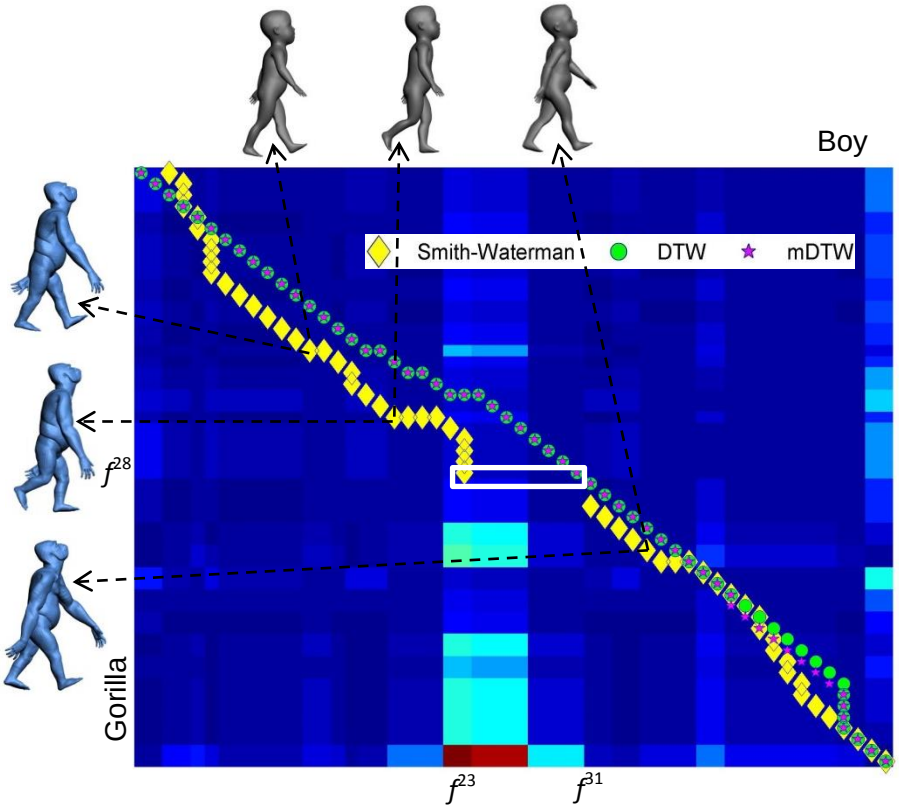


Figure 5-14 Comparisons of the sequence alignment results by using Smith-Waterman, DTW and mDTW for 'Gorilla' (vertical-wise) and 'Boy' (horizontal-wise).

Similarly, 'Gorilla', 'Boy' and 'Michael' are time-synchronized deforming meshes performing 'Walking' motions. Both in Figure 5-13 and Figure 5-14 we observe gaps in the alignment paths computed by using Smith-Waterman algorithm. This could have been resulted from the following two reasons:

- Smith-Waterman algorithm is a local sequence alignment method that allows gaps among the aligned pairs. This is particularly interesting when we want to skip noisy dissimilary subsequences while matching two sequences.

- The other reason could be because we apply Smith-Waterman algorithm over the key-frames, each of which represents a subsequence of frames (Section 5.3). For example, in Figure 5-14, the key-frame representing the subsequence from f^{23} to f^{31} (see the rectangle in Figure 5-14) of 'Boy' is aligned to the key-frame representing one frame f^{28} of 'Gorilla', which is the reason why we observe a gap in the alignment path in the rectangle in Figure 5-14.

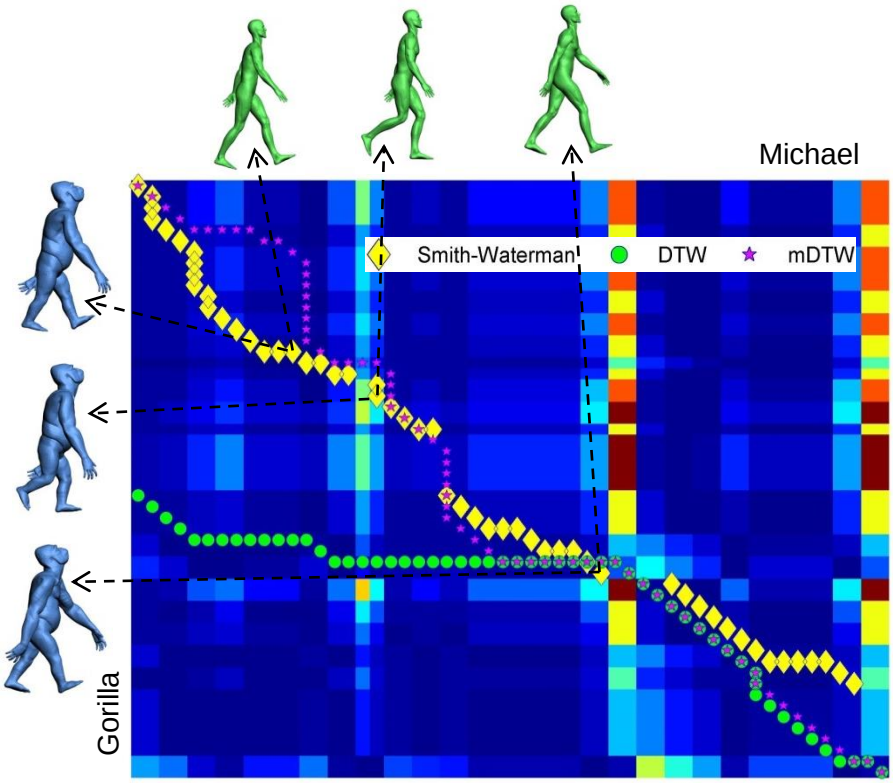


Figure 5-15 Comparisons of the sequence alignment results by using Smith-Waterman, DTW and mDTW for 'Gorilla' (vertical-wise) and 'Michael' (horizontal-wise).

Although the matchings computed by using DTW and mDTW have shown similar results so far, they are significantly different on 'Gorilla' and 'Michael' data as shown in Figure 5-15: the alignment path computed by using DTW only aligns about a half of 'Gorilla' to 'Michael', which is the globally minimized sum of the alignment distance. In comparison, the alignment path computed by using mDTW stays near the diagonal of the matrix, resulted from the constraint that the first and the last frames are forced to be aligned. The latter alignment path is more preferable knowing that 'Gorilla' and 'Michael' are time-synchronized deforming meshes. Note that the path computed by Smith-Waterman algorithm stays around the diagonal of the matrix, which is similar to that of mDTW.

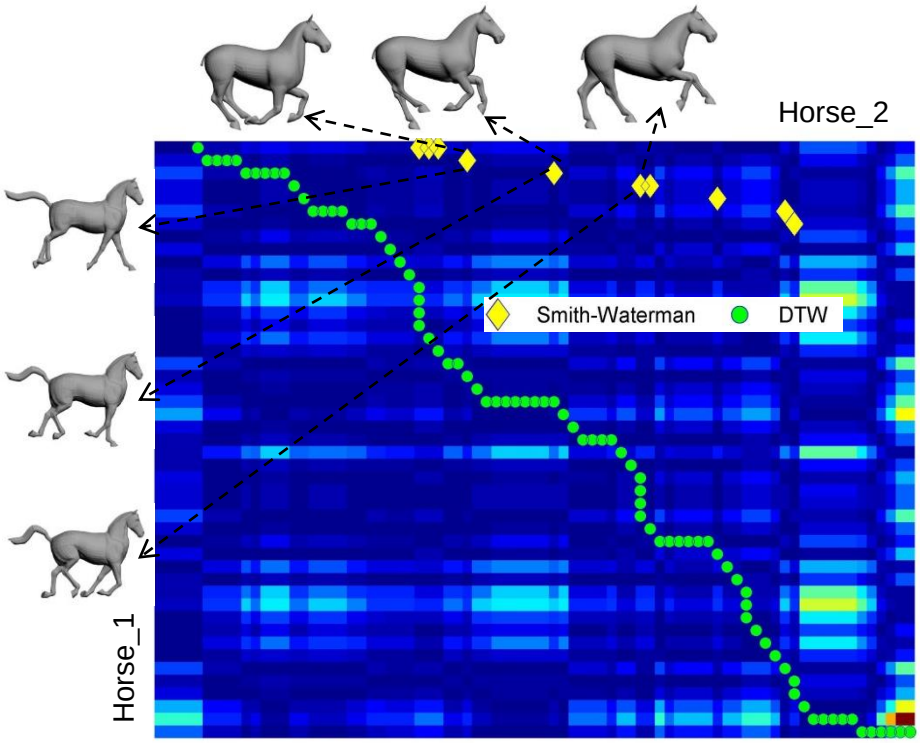


Figure 5-16 Comparisons of the sequence alignment results by using Smith-Waterman, DTW and mDTW for 'Horse_1' (vertical-wise) and 'Horse_2' (horizontal-wise).Arrows are directed to several samples of the corresponding matched frames between the two sequences.

Although 'Horse_1' and 'Horse_2' are created in different manners, 'Horse_1' is created by the deformation transfer and 'Horse_2' is an animated mesh in Maya, they both perform 'Galloping' motions. In Figure 5-16, we do not apply mDTW between 'Horse_1' and 'Horse_2'. Since these two deforming meshes are not synchronized, their first/last frames may have different poses and therefore it is not reasonable to assume respective matchings between them. In the alignment obtained by using Smith-Waterman algorithm, 'Horse_2' is aligned to 7 key frames of 'Horse_1', which correctly reflects the fact that 'Horse_2' contains 1-cycle of 'Galloping' motion, while 'Horse_1' contains 4-cycles. Similarly, the path computed by using DTW fails in this comparison.

Therefore, by comparing the three sequence alignment algorithms, if the input deforming meshes contain time-synchronized motions, mDTW has similar performance with Smith-Waterman algorithm, and is more robust than DTW by forcing the first/last frames being aligned, see Figure 5-16. However, mDTW is not capable of handling motion non-synchronized data, while Smith-Waterman algorithm is by computing local alignments.

5.6.3 Similarity measurement

5.6.3.1 Similarity of deforming meshes

Figure 5-17 shows the similarity scores we have obtained for the example deforming meshes. As expected, deforming meshes with similar motion shows high similarity scores. Note that 'Horse_2' has different motion speed and starting pose compared to 'Camel' and 'Horse_1', but the similarities among these three models are higher than the others because they all show 'Gallop' motions. On the other hand, although the shape of 'Face_1' is similar to those of 'Face_2' and 'Face_3', similarities of 'Face_1' to the other two facial models are low because they perform different facial expressions. Additionally, the average similarity between 'Gallop' and 'Walk' motions is higher than either 'Gallop'-'Facial expression' or 'Walk'-'Facial expression', which complies with human judgement.

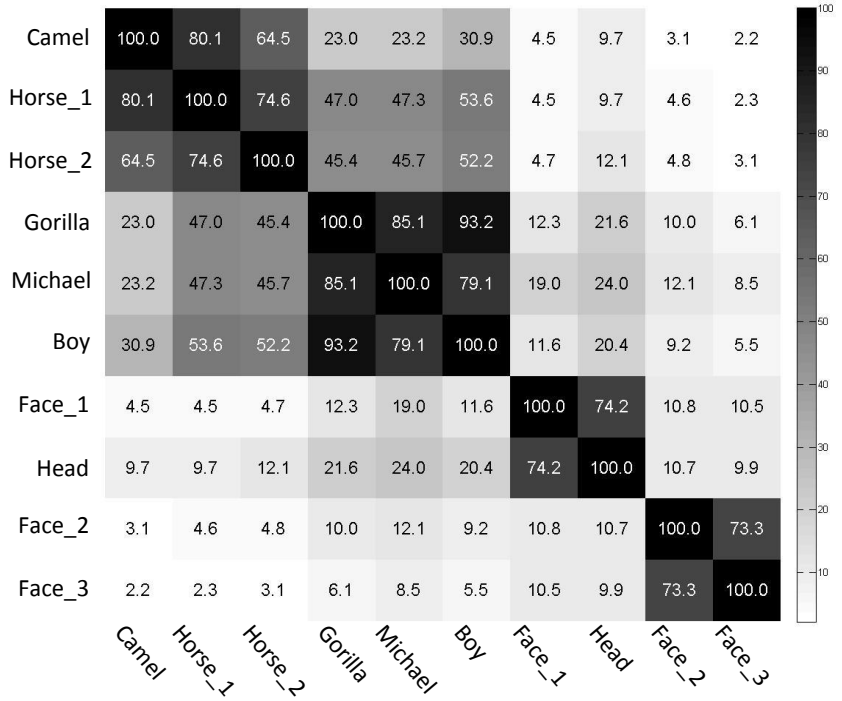


Figure 5-17 Similarity matrix among deforming meshes. The values are shown in percentage (%).

5.6.3.2 Evaluation of the motion similarities

In order to evaluate our similarity measurement method, we first study how human perceive the motion similarity between deforming meshes. To this end, we invite 11 participants who are not aware of our segmentation method and show them with the 10 animated meshes used in our experiments. Based on subjective observations, each participant gives a score on motion similarity (with a larger

number between [0 100] indicating higher similarity) between each pair of the deforming meshes. Therefore, we obtain $11 \times C_{10,2} = 495$ pairwise motion similarities of deforming meshes based on human perception.

Having created the human-based ground truth similarity between deforming meshes, we evaluate our similarity results by applying Pearson's correlation (Bartko, 1976). A Pearson's correlation ranges from -1 to +1, with +/- indicating positive/negative relationship between two variables, and the values reflecting the degree of linear relationships. In order to compute Pearson's correlation between ground-truth and our results, we save the 495 human rated similarities into a vector $\mathbf{V}_{\mathbf{gt}}$, and create another vector $\mathbf{V}_{\mathbf{ours}}$ where each value $\mathbf{V}_{\mathbf{ours}}(i)$, i=1,…,495, is the corresponding similarity value of $\mathbf{V}_{\mathbf{gt}}$ (i) but computed by using our method. That is, $\mathbf{V}_{\mathbf{ours}}$ actually contains 11 times repetition of the similarity results shown in Figure 5-18.

Figure 5-18 shows the scatter plot between $\mathbf{V}_{\mathbf{gt}}$ and $\mathbf{V}_{\mathbf{ours}}$, and the linear regression between the two vectors. Among the human score, there are 4 participants out of 11 give scores of the similarities between 'Horse_2' and deforming meshes with 'Walk' motion ('Groilla', 'Boy' and 'Michael') less than 10%, shown in the dashed circle. On the other hand, there are 4 participants give scores of the similarities between 'Camel' and the deforming meshes with 'Walk' motion more than 50%, shown in the dotted circle. Finally, although human perception on the similarity between 'Gallop' and 'Walk' does not show clear consistency, we still obtain 0.9008 as the Pearson's correlation between $\mathbf{V}_{\mathbf{gt}}$ and $\mathbf{V}_{\mathbf{ours}}$. The correlation value indicates that our similarity measurement method has a high degree of correlation with human perception on motion similarity.

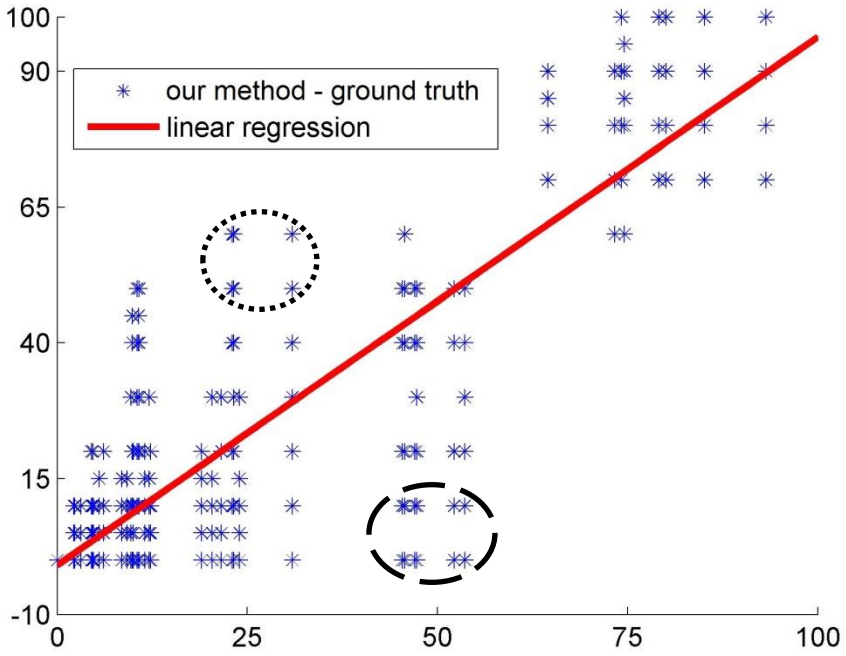


Figure 5-18 Scatterplot between the similarities of deforming meshes computed by using our method (horizontal) and the human scores of similarities (vertical). The red line is the linear regression of the 2D point distribution.

5.6.3.3 Granularity of the motion similarities

In this section, we further evaluate the granularity of our similarity measurement method with similar motions. To this end, we use 6 biped animations (including 'Jog1', 'Jog2', 'Jump1', 'Jump2', 'Walk1', and 'Walk2') from 3Ds Max motion library, and attach them to 3 meshes, i.e. 'Michael', 'Gorilla', and 'Boy', which results in 18 deforming meshes. The spatio-temporal segmentation and the graph representation of selected frames among the new deforming meshes are shown in Figure 5-8 and Figure 5-9. By applying our similarity measurement method, we obtain a motion similarity matrix among these deforming meshes (see Figure 5-19). We describe this result and its evaluation as follows:

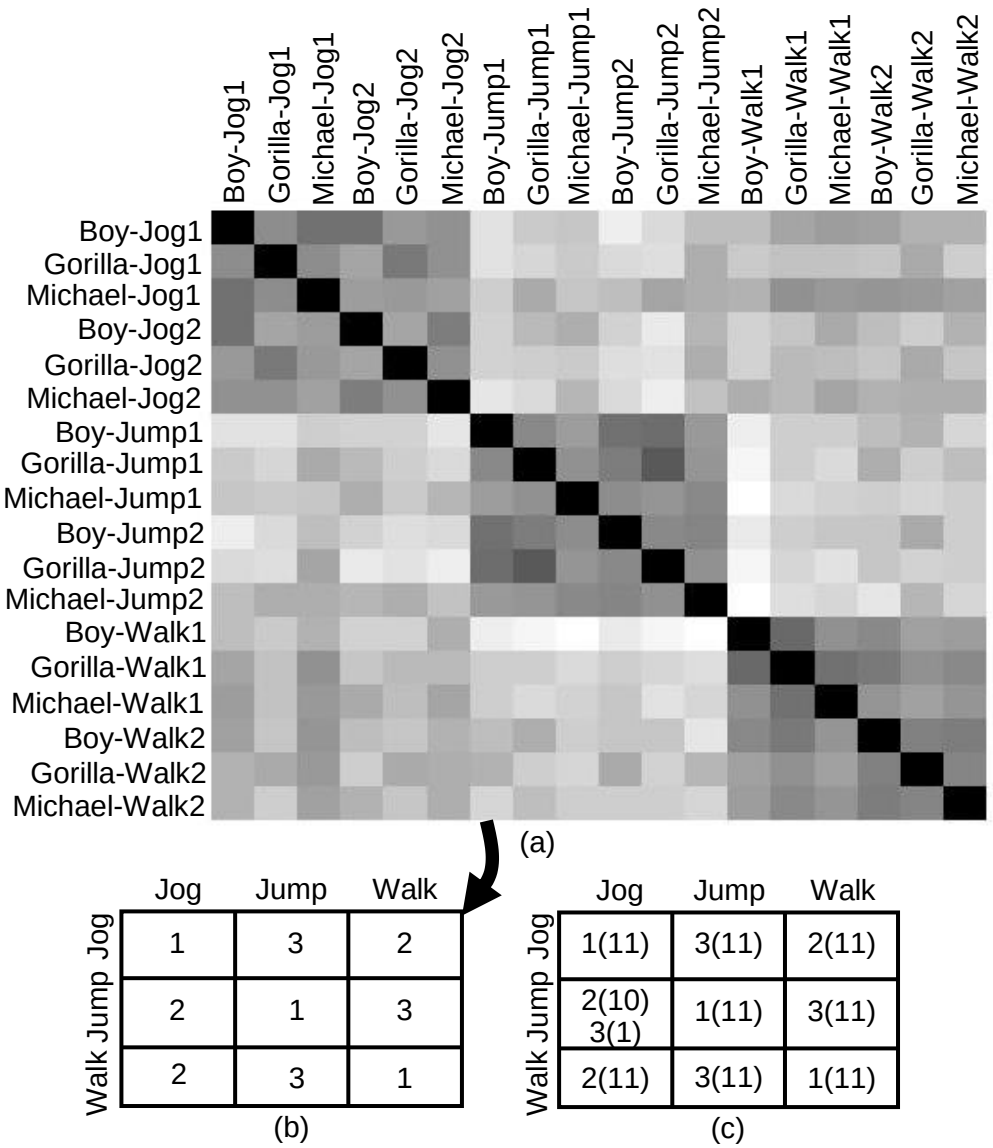


Figure 5-19 Similarities among 3 similar motions, 'Jog' , 'Jump' and 'Walk'. (a) Similarity matrix among 18 deforming meshes. (b) Each row shows the rankings of all the motions to a motion based on the average motion similarities in (a). (c) Human rated motion similarity rankings for each motion, where the numbers within each parentheses is the number of participants who give the ranking before the corresponding parentheses.

In Figure 5-19 (a), we can represent each deforming mesh as a vector of motion similarities, i.e., the corresponding row of the similarity matrix. Then, by applying K-means clustering, we successfully classify the 18 deforming meshes into 3 clusters of different motion types, i.e., 'Jog', 'Jump' and 'Walk'.

Based on the above motion classification, we convert the motion similarity matrix of deforming meshes from Figure 5-19 (a) to motion similarity ranking matrix in Figure 5-19 (b), where each row shows the rankings of all the motions to a motion

based on the average motion similarities in Figure 5-19 (a). In this motion similarity ranking matrix, we use '1/2/3' to indicate the rankings of the similarity to all the motions, where '1'/'3' is the highest/lowest ranking.

In order to validate our motion similarity rankings, we invite 11 participants to give the rankings for the 3 motions by observing the 18 animations. In Figure 5-19 (c), the number within each parentheses is the number of participants who give the ranking number before the corresponding parentheses. Note that 1 participant out of the 11 considered 'Jog' and 'Jump' being equally different to 'Walk' and gave ranking '3' for both, see the second row in Figure 5-19 (c). Apart from this, our computed ranking results are met with most of the human rankings of the 3 motions by comparing Figure 5-19 (b) and Figure 5-19 (c).

Therefore, based on the above experiments on deforming meshes with similar motions, i.e., 'Jog', 'Jump' and 'Walk', our similarity measurement method can successfully distinguish these 3 similar motion types. Moreover, our similarity measurement method reflects human perceptions on motion similarity because our motion ranking results comply well with human rankings among the 3 similar motions.

5.7 Discussions

5.7.1 Previous-pose based strains vs. rest-pose based strains

As having been discussed in Chapter 3, previous-pose based strains of a deforming mesh contain motion information, for this reason we have chosen previous-pose based strains for computing the spatio-temporal segmentation of deforming meshes. In Figure 5-20, we simulate the spatio-temporal segmentation results of the 'bending-cylinder' animation by using both previous-pose based strains and rest-pose based strains. By representing the spatio-temporal segmentation into evolving graphs, in the top row of Figure 5-20, we clearly observe the bending motion and the bended pose remains static for a period before stretching. However, the segmentation results based on rest-pose strains cannot distinguish the bending/stretching motions, see the bottom row in Figure 5-20.

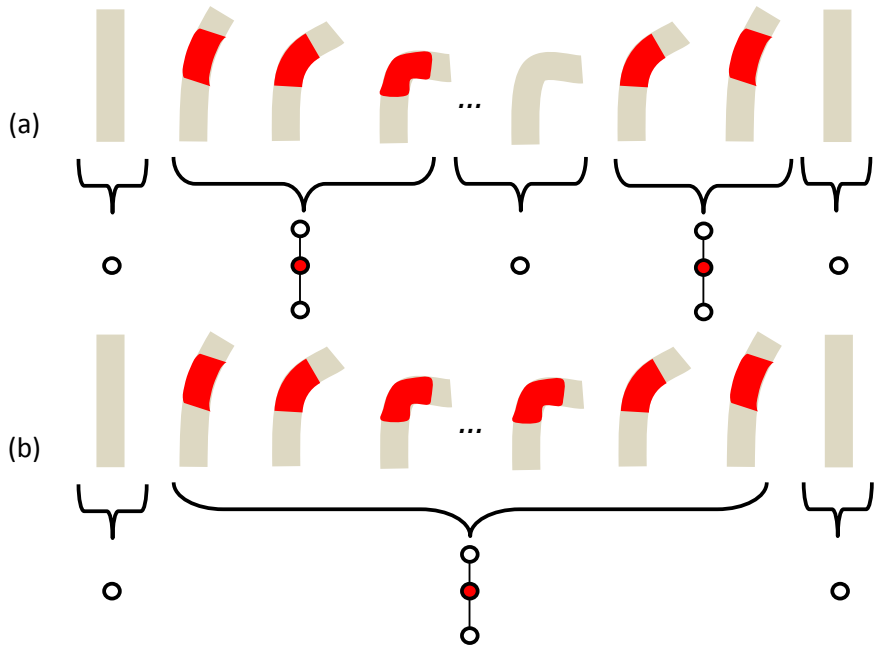


Figure 5-20 The spatio-temporal segmentation results and the evolving graph representation of 'bending-cylinder', by using (a) previous-pose based strains and (b) rest-pose based strains.

5.7.2 Graph edit distance (GED)

While computing the sequence alignment between two evolving graphs in Section 5.4, we use GED as the graph dissimilarity metric. One natural extension of computing GED is to take into account of node attribute such as surface area of spatial segments, and edge attribute such as distance between spatial segments. However, such node and edge attributes could vary due to shape differences. For example, in the frame f^4 of both 'Camel' and 'Horse_1' in Figure 5-7, although there is a 'deformed' segment on the tail of both 'Camel' and 'Horse_1', the one of 'Camel' is much smaller than the one of 'Horse_1'. Due to this reason, by taking into account of node attribute of surface area of the corresponding spatial segments, the GED between these two graphs becomes larger, which is not desirable. Moreover, if we use these GED between graph pairs for computing sequence alignment, we will obtain lower alignment score between 'Camel' and 'Horse_1', which is contrary to our object for devising a similarity measurement method independent on shape difference. Therefore, in our similarity measurement method, surface area of spatial segment and distance between spatial segments are not considered as graph attributes for computing GED.

In addition, we have investigated with binary node attributes, i.e., 'deformed' or 'rigid', for computing GED. In order to compute GED with graph attributes, Neu-

haus et al. (Neuhaus et al., 2006) compute the dissimilarity of both graph structures and attributes, and linearly combine them. However, in our case, determining the weight would be challenging because the graph structure distance and the graph attribute distance varies significantly between each frame pair (Note that we have more than a thousand of graphs for our 10 experimental models.). An intelligent self-adapting weight and a learning-based weight may be potential solutions.

5.7.3 Comparison with temporal segmentation

As have been described in Section 5.3, we represent the spatio-temporal segmentation of a deforming mesh into an evolving graph, wherein each graph represents a subsequence of frames that have the same spatial mesh segmentation. Note that by interpreting each graph as a temporal segment, we obtain a temporal segment as a partial result of our spatio-temporal segmentation. Differently from the temporal segmentation method presented in Chapter 1, whose objective is to maximize the pose similarity within temporal segments, an evolving graph detects the local deformation in a deforming mesh, i.e., the occurance and disappearance of 'deformed' segments.

We elaborate the above differences between our segmentation methods by using a 'bending-cylinder' animation, shown in Figure 5-21. In this data, the 'cylinder' remains static in the beginning, then starts bending, and keeps the bended pose in the end (Figure 5-21(a)). Figure 5-21(b) and Figure 5-21(c) shows the results by using our temporal segmentation method with different threshold θ_1 (see Section 4.2.2). In Figure 5-21(c), the value of θ_1 is lower, and therefore the 'bending' mesh sequences is divided into shorter subsequences so that the dissimilarity within each becomes smaller. On the other hand, for the spatio-temporal segmentation results, although the boundary frames may vary if we input different thresholds, we obtain three temporal segments consistently which can be represented into three graphs as shown in Figure 5-21(d) : This method detects the occurance of the 'bending' action from g_1 to g_2, and the disappearance of the 'bending' action from g_2 to g_3, see the red node in Figure 5-21(d).

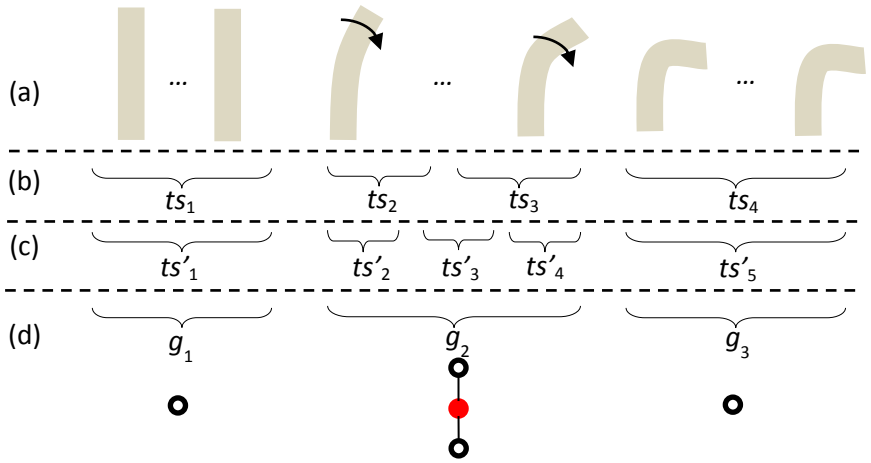


Figure 5-21 A comparison between our temporal segmentation and spatio-temporal segmentation. (a) shows a 'bending-cylinder' animation. (b) and (c) are two temporal segmentation results with different user-threshold, where the threshold in (c) is lower than that in (b). (d) shows the evolving graph representation of the spatio-temporal segmentation result.

To further illustrate the differences between our segmentation methods, we use another 'bending-cylinder' data that it first bends the upper joint and then bends the lower joint together, until both joints reach 90 degree at the same time. In Figure 5-22 (b) and Figure 5-22 (c), we show the temporal segmentation results with different threshold θ_1 by using the method presented in Chapter 4. In Figure 5-22 (c), the value of θ_1 is lower. Similar to the results in Figure 5-21(c), the 'bending' mesh sequences is divided into shorter subsequences so that the dissimilarity within each becomes smaller. Again, for the spatio-temporal segmentation results, we obtain four temporal segments consistently which can be represented into the graph sequence shown in Figure 5-22 (d) , the boundary frames may vary by using different thresholds though. In this result, g_2 can represent the action that the only the upper joint is 'bending', and g_3 can represent that both upper and lower joints are 'bending'.

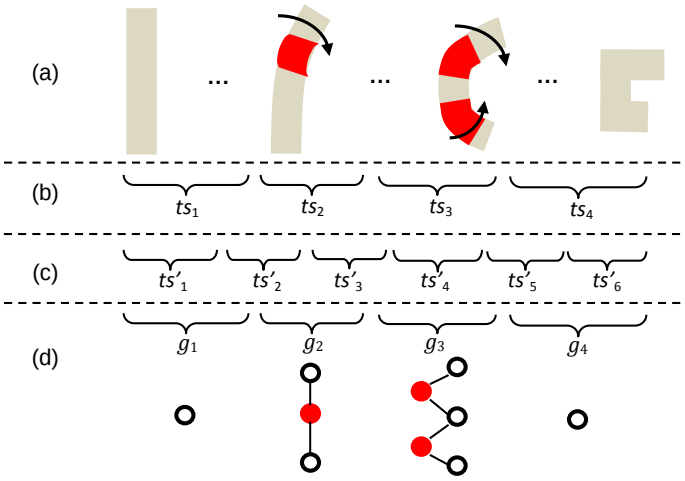


Figure 5-22 Another comparison between our temporal segmentation and spatio-temporal segmentation. (a) shows another 'bending-cylinder' animation with two 'bending' joints. (b) and (c) are two temporal segmentation results with different user-threshold, where the threshold in (c) is lower than that in (b). (d) shows the evolving graph representation of the spatio-temporal segmentation result.

5.8 Conclusion

5.8.1 Contributions

In this chapter, we have presented a method for the spatio-temporal segmentation of defoming meshes, whose results are represented with *Evolving Graphs*. We then use a sequence alignment algorithm to match *Evolving Graphs*, with an aim of computing the motion similarities between the corresponding deforming meshes. The contributions of this research are shown as follows:

- Spatio-temporal segmentation. Based on the deformation-based feature descriptor presented in Chapter 3, we have developed an efficient and effective spatio-temporal segmentation method, which incorporates both spatial and temporal deformation coherency in the deforming mesh.

- Compact representation. We represent the obtained spatio-temporal segmentation results of a deforming mesh with an evolving graph, where each graph represent the spatial segmentation within a temporal segment, with nodes denoting spatial segments and edges denoting spatial neighborhood.

- Graph clustering. By using the *graph embedding* method (Riesen and Bunke, 2009a, 2009b), we can embed graphs into vectors with the same dimensionality. Moreover, we can further convert the evolving graphs into graph cluster labels by applying K-means clustering and representing each graph

with its cluster label. This allows us to apply sequence alignment algorithms to measure similarities among different graph sequences.

- Temporal frame alignment. One important by-product of the sequence alignment between two deforming meshes is that the key frames performing similar motions are matched to each other.

- Motion similarity. Since an evolving graph represent the dynamic motions of a deforming mesh, by applying the sequence alignment algorithm between two sequences, we obtain an alignment score, which indicates their motion similarity.

- Evaluation with ground-truth. In order to validate the obtained motion similarities, we compare with human-rated motion similarities that are collected from a number of volunteers. In specific, we compute Pearson's correlation (Bartko, 1976) between the obtained similarities by using our method and the human-based ground-truth motion similarities. The obtained high correlation indicates that our motion similarity measurement method successfully reflects human perception on the motion similarities of deforming meshes.

5.8.2 Summary

In this chapter, after a review of the existing spatio-temporal techniques in both Computer Network and Computer Vision fields, we have presented a new method for the spatio-temporal segmentation of deforming meshes, which to the best of our knowledge has not been studied before. Moreover, we represent the spatio-temporal segmentation results into *Evolving Graphs*, and therefore compare deforming meshes by using existing sequence alignment algorithm to match the corresponding *Evolving Graphs*. In this work, we obtain two interesting results, the frame alignments and the similarity between deforming meshes, both have been shown with our experimental results. Additionally, we evaluate our similarity results with human-based ground-truth motion similarities among the experimental deforming meshes, which show that our similarity measurement method complies well with human perception.

Chapter 6 Conclusions

6.1 Contributions

In this thesis, we have developed segmentation techniques that compute the temporal and spatio-temporal segmentation for deforming meshes based on the deformation coherency in the data. Although there have been several works on deforming meshes, the best of our knowledge, the temporal and spatio-temporal segmentation have not been studied before. We further extend the segmentation results towards the application of motion similarity measurement between deforming meshes.

Knowing the fact that the deformations in a deforming mesh are normally both spatially and temporally correlated. The existing works on deforming mesh however compute one single spatial segmentation for an entire deforming mesh, which overlooks the temporal deformation coherency. By taking the subsequences with similar poses as temporal segments, our temporal segmentation method can divide deforming meshes undergoing identical motions into temporal segments with similar sub-motions, despite of their shape differences. Furthermore, our spatio-temporal segmentation method enables us to develop a compact representation for deforming meshes, which allows us to measure motion similarities among them.

To summarize, this dissertation contains the following contributions :

Deformation-based feature descriptor: Strain. Given a deforming mesh, we begin by devising a per-triangle feature descriptor that measures the deformation of a triangle within each frame. As having been introduced in Chapter 3, this feature descriptor is independent to global shape translation, rotation and uniform scale. Moreover, the per-triangle strain value is robust over shape difference when different shapes performing identical motions.

Temporal segmentation of deforming meshes. We have presented a temporal segmentation method for deforming meshes. In our temporal segmentation algorithm, based on our deformation-based feature descriptor, we first define a distance me-

tric for each frame pair based on the difference of their triangle deformation, then we further define within-segment frame dissimilarity as the average of all possible pairwise frame distance within each candidate temporal segment. Finally, the boundary frames for the temporal segmentation are determined by minimizing the sum of within-segment frame dissimilarities. This allows us to obtain the segmentation result that each temporal segment is a subsequence of similar frames, i.e., frames with similar poses.

Our experiments on both synthesized and motion captured deforming meshes validate the effectiveness of the presented approach. It is also encouraging that we can obtain consistent temporal segmentation for different deforming meshes exhibiting similar motions, despite their shape differences.

Spatio-temporal segmentation of deforming meshes. Next, we have presented a spatio-temporal segmentation method for deforming meshes. First of all, based on the degree of deformation of each triangle in each frame, we binarily label the triangles with either 'deformed' or 'rigid'. Then, we compute a spatio-temporal segmentation by merging the 'deformed' triangles that are either spatially or temporally connected. We then use an *evolving graph* to represent the spatio-temporal segmentation, where each node represents a spatial segment, each edge the neighbourhood between two spatial segments, and each graph is a key frame representing a subsequence of frames with the same graph representation.

Having computed the *evolving graphs* of two deforming meshes, we proceed to compute the similarity of the *evolving graphs* by adopting a sequence alignment method. However, a sequence alignment method cannot be directly applied on two graph sequences because the graphs may have different dimensions, i.e., different node numbers. In order to avoid this problem, we classify the similar graphs and assign the graphs in the same cluster with the same label. As a result, each *evolving graph* is represented into a sequence of cluster labels. Finally, we compute the alignment score between the two cluster label sequences by using a sequence alignment algorithm, which reflects the similarity between two deforming meshes.

The outcome of this method is two folds: (1) Temporal frame alignment. According to our experiments, the alignment results between two deforming meshes with similar motions show that the key frames performing similar actions are well matched to each other. (2) Motion similarity measurement. Based on the spatio-temporal segmentation results, we have devised a similarity measurement method for deforming meshes, which measures the similarity of motions that are performed by deforming meshes.

Our experimental results on a number of deforming meshes show that the motion similarities can be captured correctly, despite shape differences. We validate our similarity results by computing Pearson's correlation with human-based ground truth motion similarities. The obtained high correlation indicates that our motion similarity measurement method successfully reflects human perception on the motion similarities of deforming meshes.

6.2 Perspectives

6.2.1 Temporal segmentation of deforming meshes

By applying our temporal segmentation method, we have successfully handled mesh sequences with over a thousand frames, as well as meshes with thousands of triangles. In our experiments, the values of the threshold θ_1 (see Section 4.2) for processing different deforming meshes are provided by user depending on which level of motion details are desired. However, we have been aware that the requirement of user-parameter limits the application of this algorithm to experienced users. To alleviate this limitation, an interesting improvement would be to learn the user parameter θ_1 from human-based ground-truth segmentations. A sufficient variety of mesh types with different motions would be needed for building the ground-truth dataset.

Additionally, assuming we have the automatic computation of temporal segmentations, a further future scenario is to assist applications such as shape retrieval from a long mesh sequence based on motion similarities: We first apply our method to divide the long mesh sequence into temporal segments of submotions, and then search among temporal segments. We save the computation time in such application because we compute the matching between a query sequence with each temporal segment independently, instead of with the entire sequence.

6.2.2 Spatio-temporal segmentation and similarity measurement of deforming meshes

One limitation of our spatio-temporal segmentation method is that we assume a deforming mesh can be segmented into either 'deformed' or 'rigid' parts. Due to this reason, our segmentation method will not be applicable to highly dynamic animations such as the surface simulation of flowing water, which would result in one single 'deformed' segment by using our method.

Note that for computing the spatio-temporal segmentation, we binarily label each triangle at each frame with ‘deformed’ and ‘rigid’, and group the ‘deformed’ triangles that are either spatially or temporally connected as ‘deformed’ spatio-temporal segments. In the mean time, we also obtain ‘rigid’ spatio-temporal segments. By mapping the segmentation results onto one mesh, we will obtain ‘deformed’ regions wherein each triangle is from ‘deformed’ segment and ‘rigid’ regions wherein each triangle is from ‘rigid’ segments. Thefore, our segmentation results could be used for mesh simplification by simplifying the ‘rigid’ regions with larger triangles while keeping dense sampling of the ‘deformed’ regions. In this way, we can simply a deforming mesh while keeping the information of motions as much as possible.

Another limitation of the proposed similarity measurement method is the expensive computation cost, mainly due to the heavy computation of the pairwise graph edit distance. To compute the similarities among all the evolving graphs of the 10 deforming meshes used in our experiment, consisting of 1135 graphs, it takes about two hours to compute the graph distance matrix, followed by computing graph clustering. However, once the clusters have been computed for a dataset with sufficient variety, computing the labels for a new deforming mesh will only be a matter of computing the graph embedding of each graph in its evolving graph, and clustering each of the graphs to the closest graph cluster center.

One obvious potential of our segmentation-based similarity measurement method is its extension towards shape query applications, which will enable to search from a database for deforming meshes performing identical or similar motions.

References

3dS MAX L&T CD., (2006). MAX User Reference, Auto Desk Co., Version 9.0, 2006.

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 433-459.

Acosta, O., Fripp, J., Rueda, A., Xiao, D., Bonner, E., Bourgeat, P., & Salvado, O. (2010, April). 3D shape context surface registration for cortical mapping. In*Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on* (pp. 1021-1024). IEEE.

Aksoy, E. E., Abramov, A., Worgotter, F., & Dellen, B. (2010, May). Categorizing object-action relations from semantic scene graphs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (pp. 398-405). IEEE.

Alexa, M., & Müller, W. (2000, September). Representing animations by principal components. In *Computer Graphics Forum* (Vol. 19, No. 3, pp. 411-418). Blackwell Publishers Ltd.

Amjoun, R. (2007). Efficient compression of 3d dynamic mesh sequences.

Anagnostopoulos, A., Kumar, R., Mahdian, M., Upfal, E., & Vandin, F. (2012, January). Algorithms on evolving graphs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (pp. 149-160). ACM.

Ankerst, M., Kastenmüller, G., Kriegel, H. P., & Seidl, T. (1999, January). 3D shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases* (pp. 207-226). Springer Berlin Heidelberg.

Ankerst, M., Kriegel, H. P., & Seidl, T. (1998). A multistep approach for shape similarity search in image databases. *Knowledge and Data Engineering, IEEE Transactions on*, *10*(6), 996-1004.

Arcila, R., Buddha, S. K., Hétroy, F., Denis, F., & Dupont, F. (2010). A framework for motion-based mesh sequence segmentation.

Arcila, R., Cagniart, C., Hétroy, F., Boyer, E., & Dupont, F. (2013). Segmentation of temporal mesh sequences into rigidly moving components.*Graphical Models*, *75*(1), 10-22.

Attene, M., Falcidieno, B., & Spagnuolo, M. (2006a). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, *22*(3), 181-193.

Attene, M., Katz, S., Mortara, M., Patané, G., Spagnuolo, M., & Tal, A. (2006b). Mesh segmentation-a comparative study. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on* (pp. 7-7). IEEE.

Barbič, J., Safonova, A., Pan, J. Y., Faloutsos, C., Hodgins, J. K., & Pollard, N. S. (2004, May). Segmenting motion capture data into distinct behaviors. In*Proceedings of the 2004 Graphics Interface Conference* (pp. 185-194). Canadian Human-Computer Communications Society.

Bartko, J. J. (1976). On various intraclass correlation reliability coefficients.*Psychological bulletin*, *83*(5), 762.

Barton, G. J. (1993). An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. *Computer applications in the biosciences: CABIOS*, *9*(6), 729-734.

Baum, L. E. (1972). An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes.*Inequalities*, *3*, 1-8.

Belongie, S., Malik, J., & Puzicha, J. (2000, November). Shape context: A new descriptor for shape matching and object recognition. In *NIPS* (Vol. 2, p. 3).

Belongie, S., Mori, G., & Malik, J. (2006). Matching with shape contexts. In*Statistics and Analysis of Shapes* (pp. 81-105). Birkhäuser Boston.

Benhabiles, H., Lavoué, G., Vandeborre, J. P., & Daoudi, M. (2011, December). Learning Boundary Edges for 3D‐Mesh Segmentation. In *Computer Graphics Forum* (Vol. 30, No. 8, pp. 2170-2182). Blackwell Publishing Ltd.

Benhabiles, H., Vandeborre, J. P., Lavoué, G., & Daoudi, M. (2009, June). A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. In *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on* (pp. 36-43). IEEE.

Benhabiles, H., Vandeborre, J. P., Lavoué, G., & Daoudi, M. (2010). A comparative study of existing metrics for 3D-mesh segmentation evaluation.*The Visual Computer*, *26*(12), 1451-1466.

Berlingerio, M., Bonchi, F., Bringmann, B., & Gionis, A. (2009). Mining graph evolution rules. In *Machine learning and knowledge discovery in databases* (pp. 115-130). Springer Berlin Heidelberg.

Boreczky, J. S., & Rowe, L. A. (1996). Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, *5*(2), 122-128.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer.

Bronstein, A. M., Bronstein, M. M., Bustos, B., Castellani, U., Crisani, M., Falcidieno, B., ... & Sun, J. (2010). SHREC 2010: robust feature detection and description benchmark. *Proc. 3DOR*, *2*(5), 6.

Bronstein, A. M., Bronstein, M. M., & Kimmel, R. (2008). *Numerical geometry of non-rigid shapes*. Springer. ISBN: 978-0-387-73300-5.

Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, *19*(3), 255-259.

Chan, J., Bailey, J., & Leckie, C. (2008). Discovering correlated spatio-temporal changes in evolving graphs. *Knowledge and Information Systems*, *16*(1), 53-96.

Chen, L., & Georganas, N. D. (2006). An efficient and robust algorithm for 3D mesh segmentation. *Multimedia Tools and Applications*, *29*(2), 109-125.

Chen, X., Golovinskiy, A., & Funkhouser, T. (2009, July). A benchmark for 3D mesh segmentation. In *ACM Transactions on Graphics (TOG)* (Vol. 28, No. 3, p. 73). ACM.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *17*(8), 790-799.

Chung, F. R. (1997). *Spectral graph theory* (Vol. 92). American Mathematical Society. Page 21.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *24*(5), 603-619.

Crandall, S. H., Lardner, T. J., Archer, R. R., Cook, N. H., & Dahl, N. C. (1978). An introduction to the mechanics of solids.

De Aguiar, E., Theobalt, C., Thrun, S., & Seidel, H. P. (2008, April). Automatic Conversion of Mesh Animations into Skeleton‐based Animations. In *Computer Graphics Forum* (Vol. 27, No. 2, pp. 389-397). Blackwell Publishing Ltd.

Desikan, P., & Srivastava, J. (2004, August). Mining temporally evolving graphs. In *Proceedings of the the Sixth WEBKDD Workshop in conjunction with the 10th ACM SIGKDD conference* (Vol. 22).

Dollár, P., Rabaud, V., Cottrell, G., & Belongie, S. (2005, October). Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on* (pp. 65-72). IEEE.

Fan, L., & Liu, K. (2011, April). Paint mesh cutting. In *Computer Graphics Forum* (Vol. 30, No. 2, pp. 603-612). Blackwell Publishing Ltd.

Fod, A., Matarić, M. J., & Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous robots*, *12*(1), 39-54.

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., & Dobkin, D. (2004, August). Modeling by example. In *ACM Transactions on Graphics (TOG)* (Vol. 23, No. 3, pp. 652-663). ACM.

Gal, R., Shamir, A., & Cohen-Or, D. (2007). Pose-oblivious shape signature. *Visualization and Computer Graphics, IEEE Transactions on*, *13*(2), 261-271.

Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and applications*, *13*(1), 113-129.

Garland, M., Willmott, A., & Heckbert, P. S. (2001, March). Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (pp. 49-58). ACM.

Gelfand, N., & Guibas, L. J. (2004, July). Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 214-223). ACM.

Giorgi, D., Biasotti, S., & Paraboschi, L. (2007). Shrec: shape retrieval contest: Watertight models track. *Online]: http://watertight. ge. imati. cnr. it*.

Golovinskiy, A., & Funkhouser, T. (2008, December). Randomized cuts for 3D mesh analysis. In *ACM Transactions on Graphics (TOG)* (Vol. 27, No. 5, p. 145). ACM.

Golovinskiy, A., & Funkhouser, T. (2009). Consistent segmentation of 3D models. *Computers & Graphics*, *33*(3), 262-269.

Grundmann, M., Kwatra, V., Han, M., & Essa, I. (2010, June). Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 2141-2148). IEEE.

Gullett, P. M., Horstemeyer, M. F., Baskes, M. I., & Fang, H. (2008). A deformation gradient tensor and strain tensors for atomistic simulations.*Modelling and Simulation in Materials Science and Engineering*, *16*(1), 015001.

(http::Matlab) MathWorks - MATLAB and Simulink for Technical Computing, www.mathworks.com/.

(http::Vicon) Vicon Motion Systems, http://www.vicon.com.

Higham, N. J. (1986). Computing the polar decomposition-with applications.*SIAM Journal on Scientific and Statistical Computing*, *7*(4), 1160-1174.

Huang, P., Starck, J., & Hilton, A. (2007, November). Temporal 3d shape matching. In *Visual Media Production, 2007. IETCVMP. 4th European Conference on* (pp. 1-10). IET.

Huang, Q., & Dom, B. (1995, October). Quantitative methods of evaluating image segmentation. In *Image Processing, 1995. Proceedings., International Conference on* (Vol. 3, pp. 53-56). IEEE.

Huang, Q., Koltun, V., & Guibas, L. (2011, December). Joint shape segmentation with linear programming. In *ACM Transactions on Graphics (TOG)* (Vol. 30, No. 6, p. 125). ACM.

Hu, R., Fan, L., & Liu, L. (2012, August). Co ‐ Segmentation of 3D Shapes via Subspace Clustering. In *Computer graphics forum* (Vol. 31, No. 5, pp. 1703-1713). Blackwell Publishing Ltd.

Hubeli, A., & Gross, M. (2001, October). Multiresolution feature extraction for unstructured meshes. In *Proceedings of the Conference on Visualization'01* (pp. 287-294). IEEE Computer Society.

Inaba, M., Katoh, N., & Imai, H. (1994, June). Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the tenth annual symposium on Computational geometry* (pp. 332-339). ACM.

James, D. L., & Twigg, C. D. (2005, July). Skinning mesh animations. In *ACM Transactions on Graphics (TOG)* (Vol. 24, No. 3, pp. 399-407). ACM.

Janus, B., & Nakamura, Y. (2005, July). Unsupervised probabilistic segmentation of motion data for mimesis modeling. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on* (pp. 411-417). IEEE.

Julius, D., Kraevoy, V., & Sheffer, A. (2005, September). D-Charts: Quasi-Developable Mesh Segmentation. In *Computer Graphics Forum* (Vol. 24, No. 3, pp. 581-590). Blackwell Publishing, Inc.

Johnson, A. E. (1997). *Spin-images: a representation for 3-D surface matching*(Doctoral dissertation, Microsoft Research).

Johnson, A. E., & Hebert, M. (1997, May). Surface registration by matching oriented points. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in* (pp. 121-128). IEEE.

Jung, M., & Kim, H. (2004, October). Snaking across 3d meshes. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (pp. 87-93). IEEE.

Kahol, K., Tripathi, P., & Panchanathan, S. (2004, May). Automated gesture segmentation from dance sequences. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on* (pp. 883-888). IEEE.

Kalafatlar, E., & Yemez, Y. (2010, August). 3d articulated shape segmentation using motion information. In *Pattern Recognition (ICPR), 2010 20th International Conference on* (pp. 3595-3598). IEEE.

Kalogerakis, E., Hertzmann, A., & Singh, K. (2010). Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, *29*(4), 102.

Kambhatla, N., & Leen, T. K. (1997). Dimension reduction by local principal component analysis. *Neural Computation*, *9*(7), 1493-1516.

Kan, A., Chan, J., Bailey, J., & Leckie, C. (2009, December). A query based approach for mining evolving graphs. In *Proceedings of the Eighth Australasian Data Mining Conference-Volume 101* (pp. 139-150). Australian Computer Society, Inc..

Karni, Z., & Gotsman, C. (2000, July). Spectral compression of mesh geometry. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (pp. 279-286). ACM Press/Addison-Wesley Publishing Co..

Karni, Z., & Gotsman, C. (2004). Compression of soft-body animation sequences. *Computers & Graphics*, *28*(1), 25-34.

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

Katz, S., & Tal, A. (2003). *Hierarchical mesh decomposition using fuzzy clustering and cuts* (Vol. 22, No. 3, pp. 954-961). ACM.

Kazhdan, M., Funkhouser, T., & Rusinkiewicz, S. (2003, June). Rotation invariant spherical harmonic representation of 3 D shape descriptors. In*Symposium on geometry processing* (Vol. 6).

Kim, V. G., Li, W., Mitra, N. J., Chaudhuri, S., DiVerdi, S., & Funkhouser, T. (2013). Learning part-based templates from large collections of 3D shapes.*ACM Transactions on Graphics (TOG)*, *32*(4), 70.

Kim, V. G., Lipman, Y., & Funkhouser, T. (2011, August). Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)* (Vol. 30, No. 4, p. 79). ACM.

Kleinberg, J., & Tardos, É. (2006). *Algorithm design*. Pearson Education India. page. 160.

Koprinska, I., & Carrato, S. (2001). Temporal video segmentation: A survey.*Signal processing: Image communication*, *16*(5), 477-500.

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM transactions on graphics (TOG)*, *21*(3), 473-482.

Krishna, M. V., Bodesheim, P., Körner, M., & Denzler, J. (2014). Temporal video segmentation by event detection: A novelty detection approach. *Pattern Recognition and Image Analysis*, *24*(2), 243-255.

Kruskall, J. B. & Liberman, M. (1983). The symmetric time warping algorithm: From continuous to discrete. In *Time Warps, String Edits and Macromolecules*. Addison-Wesley.

Lai, Y. K., Hu, S. M., Martin, R. R., & Rosin, P. L. (2008, June). Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling* (pp. 183-191). ACM.

Lai, Y. K., Zhou, Q. Y., Hu, S. M., & Martin, R. R. (2006, June). Feature sensitive mesh segmentation. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling* (pp. 17-25). ACM.

Lavoué, G., Dupont, F., & Baskurt, A. (2005). A new CAD mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design*, *37*(10), 975-987.

Lavoué, G., Vandeborre, J. P., Benhabiles, H., Daoudi, M., Huebner, K., Mortara, M., & Spagnuolo, M. (2012, May). SHREC'12 Track: 3D mesh segmentation. In *Proceedings of the 5th Eurographics conference on 3D Object Retrieval* (pp. 93-99). Eurographics Association.

Lee, N. S., Yamasaki, T., & Aizawa, K. (2008, June). Hierarchical mesh decomposition and motion tracking for time-varying-meshes. In *Multimedia and Expo, 2008 IEEE International Conference on* (pp. 1565-1568). IEEE.

Lee, T. Y., Lin, P. H., Yan, S. U., & Lin, C. H. (2005). Mesh decomposition using motion information from animation sequences. *Computer Animation and Virtual Worlds*, *16*(3‐4), 519-529.

Lee, T. Y., Wang, Y. S., & Chen, T. G. (2006). Segmenting a deforming mesh into near-rigid components. *The Visual Computer*, *22*(9-11), 729-739.

Lezama, J., Alahari, K., Sivic, J., & Laptev, I. (2011, June). Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE.

Li, X., Woon, T. W., Tan, T. S., & Huang, Z. (2001, March). Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (pp. 35-42). ACM.

Lian, Z., Godil, A., & Xiao, J. (2013). Feature-preserved 3D canonical form.*International journal of computer vision*, *102*(1-3), 221-238.

Lin, I. C., Peng, J. Y., Lin, C. C., & Tsai, M. H. (2011). Adaptive motion data representation with repeated motion analysis. *Visualization and Computer Graphics, IEEE Transactions on*, *17*(4), 527-538.

Liu, G., & McMillan, L. (2006, September). Segment-based human motion compression. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 127-135). Eurographics Association.

Liu, R., & Zhang, H. (2004, October). Segmentation of 3D meshes through spectral clustering. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (pp. 298-305). IEEE.

Liu, R., & Zhang, H. (2007, September). Mesh segmentation via spectral embedding and contour analysis. In *Computer Graphics Forum* (Vol. 26, No. 3, pp. 385-394). Blackwell Publishing Ltd.

Liu, T., Zhang, H. J., & Qi, F. (2003). A novel video key-frame-extraction algorithm based on perceived motion energy model. *Circuits and Systems for Video Technology, IEEE Transactions on*, *13*(10), 1006-1013.

Liu, W., Kan, A., Chan, J., Bailey, J., Leckie, C., Pei, J., & Kotagiri, R. (2012, October). On compressing weighted time-evolving graphs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*(pp. 2319-2322). ACM.

Luo, G., Bergstrom, N., Ek, C. H., & Kragic, D. (2011, September). Representing actions with kernels. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (pp. 2028-2035). IEEE.

Luo, P., Wu, Z., Xia, C., Feng, L., & Ma, T. (2013). Co-segmentation of 3D shapes via multi-view spectral clustering. *The Visual Computer*, *29*(6-8), 587-597.

Maji, S., Berg, A. C., & Malik, J. (2008, June). Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

Mamou, K., Zaharia, T., & Prêteux, F. (2006). A skinning approach for dynamic 3D mesh compression. *Computer Animation and Virtual Worlds*, *17*(3 - 4), 337-346.

Mangan, A. P., & Whitaker, R. T. (1999). Partitioning 3D surface meshes using watershed segmentation. *Visualization and Computer Graphics, IEEE Transactions on*, *5*(4), 308-321.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (Vol. 2, pp. 416-423). IEEE.

Megret, R., & DeMenthon, D. (2002). *A survey of spatio-temporal grouping techniques* (No. LAMP-094). MARYLAND UNIV COLLEGE PARK LANGUAGE AND MEDIA PROCESSING LAB.

Müller, M., & Röder, T. (2006, September). Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 137-146). Eurographics Association.

Müller, M., Röder, T., & Clausen, M. (2005). Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics (TOG)*, *24*(3), 677-685.

Neuhaus, M., Riesen, K., & Bunke, H. (2006). Fast suboptimal algorithms for the computation of graph edit distance. In *Structural, Syntactic, and Statistical Pattern Recognition* (pp. 163-172). Springer Berlin Heidelberg.

Ovsjanikov, M., Mérigot, Q., Mémoli, F., & Guibas, L. (2010, July). One point isometric matching with the heat kernel. In *Computer Graphics Forum* (Vol. 29, No. 5, pp. 1555-1564). Blackwell Publishing Ltd.

Pass, G., & Zabih, R. (1999). Comparing images using joint histograms.*Multimedia systems*, *7*(3), 234-240.

Petitjean, S. (2002). A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys (CSUR)*, *34*(2), 211-262.

Praun, E., & Hoppe, H. (2003). Spherical parametrization and remeshing. *ACM Transactions on Graphics (TOG)*, *22*(3), 340-349.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods.*Journal of the American Statistical association*, *66*(336), 846-850.

Riesen, K., & Bunke, H. (2009a). Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(06), 1053-1081.

Riesen, K., & Bunke, H. (2009b). Reducing the dimensionality of dissimilarity space embedding graph kernels. *Engineering Applications of Artificial Intelligence*, *22*(1), 48-56.

Robardet, C. (2009, December). Constraint-based pattern mining in dynamic graphs. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*(pp. 950-955). IEEE.

Rossi, R. A., Gallagher, B., Neville, J., & Henderson, K. (2013, February). Modeling dynamic behavior in large evolving graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining* (pp. 667-676). ACM.

Ryoo, M. S., & Aggarwal, J. K. (2009, September). Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *Computer Vision, 2009 IEEE 12th International Conference on* (pp. 1593-1600). IEEE.

Sattler, M., Sarlette, R., & Klein, R. (2005, July). Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 209-217). ACM.

Shahraray, B. (1995, April). Scene change detection and content-based sampling of video sequences. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology* (pp. 2-13). International Society for Optics and Photonics.

Shamir, A. (2008, September). A survey on mesh segmentation techniques. In*Computer graphics forum* (Vol. 27, No. 6, pp. 1539-1556). Blackwell Publishing Ltd.

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

Sharma, A., Von Lavante, E., & Horaud, R. (2010). Learning shape segmentation using constrained spectral clustering and probabilistic label transfer. In *Computer Vision–ECCV 2010* (pp. 743-756). Springer Berlin Heidelberg.

Sheffer, A. (2001). Model simplification for meshing using face clustering.*Computer-Aided Design*, *33*(13), 925-934.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *22*(8), 888-905.

Shlafman, S., Tal, A., & Katz, S. (2002, September). Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum* (Vol. 21, No. 3, pp. 219-228). Blackwell Publishing, Inc.

Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., & Cohen-Or, D. (2011).*Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering* (Vol. 30, No. 6, p. 126). ACM.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, *147*(1), 195-197.

Spriggs, E. H., De La Torre, F., & Hebert, M. (2009, June). Temporal segmentation and activity classification from first-person sensing. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference On* (pp. 17-24). IEEE.

Sumner, R. W., & Popović, J. (2004, August). Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)* (Vol. 23, No. 3, pp. 399-405). ACM.

Sumner, R. W., Zwicker, M., Gotsman, C., & Popović, J. (2005, July). Mesh-based inverse kinematics. In *ACM Transactions on Graphics (TOG)* (Vol. 24, No. 3, pp. 488-495). ACM.

Sun, J., Faloutsos, C., Papadimitriou, S., & Yu, P. S. (2007, August). Graphscope: parameter-free mining of large time-evolving graphs. In*Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 687-696). ACM.

Sun, J., Ovsjanikov, M., & Guibas, L. (2009, July). A Concise and Provably Informative Multi‐Scale Signature Based on Heat Diffusion. In *Computer Graphics Forum* (Vol. 28, No. 5, pp. 1383-1392). Blackwell Publishing Ltd.

Tevs, A., Berner, A., Wand, M., Ihrke, I., & Seidel, H. P. (2011, April). Intrinsic shape matching by planned landmark sampling. In *Computer Graphics Forum*(Vol. 30, No. 2, pp. 543-552). Blackwell Publishing Ltd.

Tian, Z., Xue, J., Zheng, N., Lan, X., & Li, C. (2011, September). 3d spatio-temporal graph cuts for video objects segmentation. In *Image Processing (ICIP), 2011 18th IEEE International Conference on* (pp. 2393-2396). IEEE.

Truesdell, C., & Noll, W. (2004). *The non-linear field theories of mechanics* (pp. 1-579). Springer Berlin Heidelberg.

Turetsky, R. J., & Ellis, D. P. (2003). Ground-truth transcriptions of real music from force-aligned midi syntheses. *ISMIR 2003*, 135-141.

Van Kaick, O., Zhang, H., Hamarneh, G., & Cohen‐Or, D. (2011, September). A survey on shape correspondence. In *Computer Graphics Forum* (Vol. 30, No. 6, pp. 1681-1707). Blackwell Publishing Ltd.

Vasilakis, A. A., & Fudos, I. (2014, May). Pose partitioning for multi‐resolution segmentation of arbitrary mesh animations. In *Computer Graphics Forum* (Vol. 33, No. 2, pp. 293-302).

Wang, T. S., Shum, H. Y., Xu, Y. Q., & Zheng, N. N. (2001). Unsupervised analysis of human gestures. In *Advances in Multimedia Information Processing—PCM 2001* (pp. 174-181). Springer Berlin Heidelberg.

Wang, Y., Asafi, S., van Kaick, O., Zhang, H., Cohen-Or, D., & Chen, B. (2012). Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, *31*(6), 165.

Wang, Y., Gong, M., Wang, T., Cohen-Or, D., Zhang, H., & Chen, B. (2013). Projective analysis for 3D shape segmentation. *ACM Transactions on Graphics (TOG)*, *32*(6), 192.

Wang, Y., Peterson, B. S., & Staib, L. H. (2000). Shape-based 3D surface correspondence using geodesics and local geometry. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (Vol. 2, pp. 644-651). IEEE.

Wu, Z., Wang, Y., Shou, R., Chen, B., & Liu, X. (2013). Unsupervised co-segmentation of 3D shapes via affinity aggregation spectral clustering.*Computers & Graphics*, *37*(6), 628-637.

Wuhrer, S., & Brunton, A. (2010). Segmenting animated objects into near-rigid components. *The Visual Computer*, *26*(2), 147-155.

Yamauchi, H., Gumhold, S., Zayer, R., & Seidel, H. P. (2005). Mesh segmentation driven by Gaussian curvature. *The Visual Computer*, *21*(8-10), 659-668.

Yang, Y., Yu, J. X., Gao, H., Pei, J., & Li, J. (2014). Mining most frequently changing component in evolving graphs. *World Wide Web*, *17*(3), 351-376.

Zhou, F., De la Torre, F., & Hodgins, J. K. (2013). Hierarchical aligned cluster analysis for temporal clustering of human motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *35*(3), 582-596.

Zhou, K., Synder, J., Guo, B., & Shum, H. Y. (2004, July). Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 45-54). ACM.

Zhou, Y., & Huang, Z. (2004, January). Decomposing polygon meshes by means of critical points. In *Multimedia Modelling Conference, 2004. Proceedings. 10th International* (pp. 187-195). IEEE.

Zhang, E., Mischaikow, K., & Turk, G. (2005). Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics (TOG)*,*24*(1), 1-27.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2), 119-152.

Publications

International Journal:

- **Guoliang Luo**, Frederic Cordier, and Hyewon Seo. Compression of 3D mesh sequences by temporal segmentation. In *Computer Animation and Virtual Worlds* (2013) 24, pp. 365–375. Special issue of the *26th International conference on Computer Animation and Social Agents 2013*, Istanbul, Turkey.
- **Guoliang Luo**, Frederic Cordier, and Hyewon Seo. Spatio-temporal segmentation for the similarity measurement of deforming meshes. *The Visual Computer*. (In preparation)

International Conference:

- Guoliang Luo, Frederic Cordier, and Hyewon Seo. Similarity of deforming meshes based on spatio-temporal segmentation. E*urographics 2014 Workshop on 3D Object Retrieval (2014)* pp. 77-84. Strasbourg, France.
- Guoliang Luo, Hyewon Seo, and Frederic Cordier. Temporal segmentation of deforming meshes. Short paper, the *31st Computer Graphics International (June 2014)*, Sydney, Australia.