Clustering-based shape segmentation for surface unfolding, model retrieval, and 3D printing applied in bolus shaping

By

Rui Li

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

University of Manitoba

Winnipeg

Abstract

Bolus is used to cover treatment areas of the patient skin to improve the dose distribution. The existing method of bolus shaping is a manual process to cut the material into pieces, then wrap and cover them on the target area, which is time-consuming and inaccurate. This research proposes clustering-based surface segmentation methods to improve accuracy and efficiency of the bolus shaping. Segmentation methods are developed for accurate shape unfolding, efficient model retrieval, and optimal 3D printing direction.

For the accurate shape unfolding, surface flattenability is improved using a spectral clustering method based on features combined topology and segmentation saliency. A 3D mesh surface is optimized using a particle swarm optimizer with four requirements and one constraint to further improve surface unfolding accuracy. The 3D mesh surface is unfolded into 2D planes in a coordinate transformation process to improve the iterative efficiency using a mass-spring model with crossed springs. For the efficient model retrieval, an image-based method is proposed using visual entropies and feature skeletons of images and models. For the optimal 3D printing direction, an improved spectral clustering method is developed by combining the surface topology and printing features to improve surface quality and reduce support material. High-level features of the topology and printing are abstracted by Stacked Auto-encoders.

Case studies verify the proposed segmentation and optimization methods. The shape unfolding method significantly reduces processing errors. The image-based model retrieval method shows its accuracy and efficiency in different applications. The surface segmentation increases surface quality and reduces support material of the 3D printed product.

Acknowledgments

I would first like to express my sincere gratitude to my supervisor Dr. Qingjin Peng. My research and thesis were completed under the ardent care and patient guidance of Dr. Peng. He provided me much support in both academics and my life. I am not able to complete my Ph.D. study without his constant support and great patience.

I wish to thank Drs. Carson Leung and Malcolm Xing. They provided insightful comments and encouragement for my research and thesis. The valuable questions incented me to widen my research from various perspectives. I would also like to thank my colleagues in the Virtual Manufacturing Lab who helped me with my research: Rajiv Kumar Vashisht, Hamid Fazeli, Marwan Baloch and Bocheng Xu.

I wish to acknowledge that this research has been supported by the Discovery Grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Graduate Enhancement of Tri-Council Stipends (GETS) program from the University of Manitoba and the MITACS Accelerate program.

Finally, I would like to thank my parents and my husband. They are the ones who love me the most. Their love and understanding are the support and motivation for me over four years of the Ph.D. study.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background

1.1.1 Research motivation

High-energy radiotherapy (HER) is a common method used in cancer treatment (Vyas et al, 2013). When treating superficial tumors, a phenomenon of "skin-sparing effect" occurs to affect the dose distribution (Zhou et al, 2015). A piece of material called bolus is required to cover the treatment area of patients' skin to minimize this effect for the sufficient dose distribution (Menzel, 2014). The bolus is usually made from synthetic materials such as RW3 phantom (Sroka et al, 2010) and Elasto-Gel (Vyas et al, 2013) in a thickness of 5 to 20 mm (Canters et al, 2016). Ideally, the bolus should conform closely to the underlying patients' skin without air gaps.

The existing method of bolus shaping used in CancerCare Manitoba is a manual process to cut the material into pieces, then wrap and cover them onto the target area. This process is time-consuming and inaccurate. Especially, for irregular shapes of body parts such as knee, nose, and elbow, the manual-made bolus can cause large air gaps between the treatment area and bolus, which reduces the therapeutic effect.

Different methods have been proposed to improve the bolus shaping accuracy and reduce air gaps in bolus applications. Paraffin wax bolus was used to fit irregular surfaces by making a plaster mold and pouring molten wax (Driscoll et al, 1992). Super Stuff bolus was proposed to improve the accuracy and efficiency of wax bolus shaping (Humphries et al, 1996). However, making the wax bolus requires design and manufacture of tools before building a wax model.

3D printing methods were also proposed for bolus shaping. Different materials such as PLA (Polylactic Acid) (Robar, et al, 2018) and ABS (Acrylonitrile Butadiene Styrene) (Burleson et al, 2015) have been used for printing bolus. There are limited bolus materials available for 3D printing. The bolus printing is also time-consuming. In addition, if the structure is required to support the surface during the printing process, the material utilization is reduced and manufacturing time is increased.

Removing support structures after printing can damage the surface of the printed bolus. Thus inefficiency and inaccuracy are two main problems in bolus shaping.

A 3D-2D-3D process originated in garment manufactures can improve accuracy of the bolus shaping. In the process, a 3D shape of the human body surface is unfolded into 2D patches as patterns to cut clothing materials and sew them into the garment (Huang et al, 2012). Another example is the process of sofa production (Zhang et al, 2018). To apply the 3D-2D-3D process in bolus shaping, three problems should be solved: the minimum 2D patches to meet required accuracy, the minimum self-overlaps after unfolding, and the minimum difference between the original and refolded models. A bolus retrieval method can improve the bolus shaping efficiency to quickly find the bolus in a model base. 3D printing bolus is a new trend with potential to meet difference requirements if the printing efficiency and material utilization can be improved.

Figure 1-1 shows the motivation of this research. For improving the bolus shaping accuracy, surface segmentation and unfolding methods are proposed based on a 3D-2D-3D process to reduce air gaps between the bolus and patient skins in the existing manual process. For improving the bolus shaping efficiency, a model retrieval method is proposed to search the suitable bolus in a bolus base formed by the unfolded boluses. For improving automation and accuracy of the bolus shaping process, the 3D printing method is introduced. Shape segmentation methods are proposed to reduce support structures and improve surface accuracy for the 3D printed bolus.

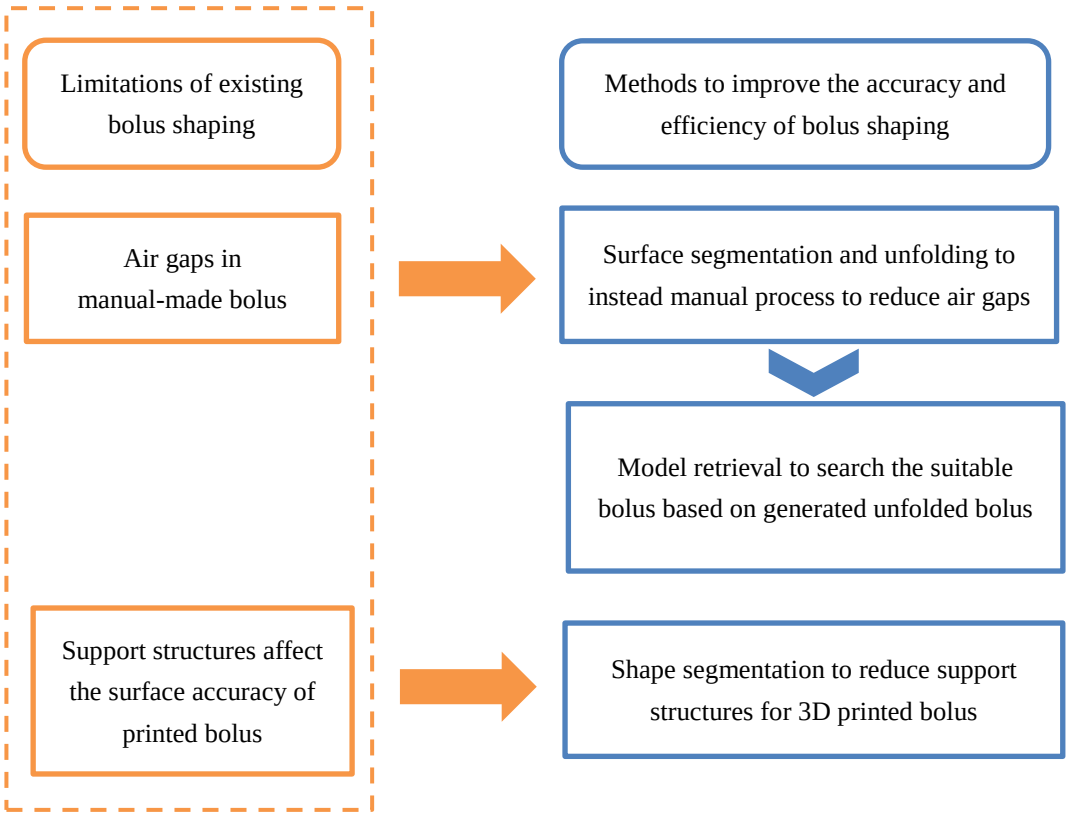

Figure 1-1 Motivation of the research

1.1.2 Research problems

Research problems of bolus shaping are identified as follows.

(1) Shape deformation and distortion in the unfolding process of 3D surfaces.

(2) Inefficient model retrieval methods.

(3) Inappropriate 3D printing direction and support material that reduce the surface quality.

1.2 Problems analysis

1.2.1 Surface unfolding

Surface unfolding is a process to transfer a 3D shape into 2D planes. Based on the computational geometry, only developable surfaces such as cylindrical and cone can be unfolded without deformation (Wang, 2008). As most 3D free-form surfaces are non-developable, it is a challenge to unfold a non-developable 3D surface into 2D

planes without deformation. Surface flattenabiliy measures the ability to unfold a 3D surface into 2D planes (Wang, 2008). Shape segmentation can add some cutting lines to divide a non-developable 3D surface into several parts to increase surface flattenability for reducing unfolding deformation. Thus, the problem of non-developable 3D surface unfolding becomes the surface segmentation to select a suitable position to add cutting lines.

The surface unfolding process can also be considered as a coordinate transformation process that needs to fix a center triangle of the 3D surface on a 2D plane and transfer other triangles of the 3D surface onto the 2D plane. After the coordinating transformation, an optimization search can be applied to reduce the difference between the original 3D surface and 2D unfolded surfaces. A major problem of existing unfolding methods is the low efficiency in the solution search as the large number of iterations required in the coordinate transformation process. In addition, if unfolded patches have the large deformation in shapes and contours, it will not be able to meet the requirement.

1.2.2 3D model retrieval

Model retrieval can improve shaping efficiency. Three-dimensional (3D) model retrieval is commonly used in searching and processing digital models for product modeling and visualization. Efficient model retrieval can improve the product design, process and management.

Based on the input to search a 3D model, 3D model retrieval methods include the shape-based and image-based model retrieval. The shape-based model retrieval uses model descriptions such as 3D point cloud data, 3D solid model, or 3D surface model as input to search a 3D model in a model base. The image-based model retrieval uses the image of a model as input to search the related 3D model. A key of image-based model retrieval methods is to build relations of the image and 3D model.

Most existing 3D model retrieval algorithms use 3D models as queries (Li et al, 2014). Retrieving 3D models using 2D images as input is increasing in different fields such as the 3D geometry-based product retrieval (Kumar and Suguna, 2016). Because

of the proliferation of 2D images and 3D models, effective and efficient image-based model retrieval techniques have become an essential research topic in applications of Augmented Reality (AR) and Virtual Reality (VR), 3D geometry-based video retrieval and highly capable autonomous vehicles. It is efficient to search a suitable bolus using a 2D picture rather than the 3D surface model.

1.2.3 3D printing direction

The printing direction optimization can improve the surface quality of 3D printed products. For the layered additive manufacturing process, the product surface quality depends on the printing direction and layer thickness. For most 3D printers, the best surface quality is obtained if the surface normal is perpendicular to the printing direction. If the surface normal is toward the negative printing direction, a support structure is required. Removing these support structures can damage surfaces of the printed product. Support structures also use the extra material and increase printing time.

In addition, a complex model should not be printed into a whole piece for the high surface quality because of support structures. A large sized model cannot be printed in one piece because of the limited printer volume. Researchers have proposed methods to segment a 3D model into pieces for 3D printing. The existing methods of surface segmentation have limitations in the feature extraction and clustering number decision. Although features including the curvature, surface quality and support materials are considered in the model segmentation, there is not a method combing all the features for model segmentation in the 3D printing process.

1.3 Research objectives and methods

(1) Development of a surface segmentation method based on an improved spectral clustering method and a mesh shape optimization method based on Particle swarm optimization (PSO) to increase surface flattenability and reduce deformation and distortion in surface unfolding.

(2) Development of a surface unfolding process based on a mass-spring model with crossed springs to improve the unfolding efficiency from a 3D surface to 2D

patches.

(3) Development of an effective image-based 3D model retrieval method based on visual entropy and skeleton of features.

(4) Development of a surface segmentation method based on spectral clustering and Deep learning for the direction optimization of 3D printing products to improve surface quality and reduce support structures.

1.4 Contents of the thesis

The organization of this thesis is shown in *Figure 1-2*. Chapter 1 introduces the research background, problem analysis, research objectives and methods, and contents of the thesis. A detailed literature review is presented in Chapter 2 to discuss related research including surface unfolding, model retrieval, direction optimization of 3D printing, and shape segmentation methods. Chapter 3 introduces methods used in this research including the spectral clustering, Affinity Propagation (AP) clustering, Stacked Auto-encoder (SAE), mass-spring model with crossed springs, and Particle swarm optimization (PSO). In Chapter 4, a surface segmentation method based on spectral clustering and a mesh shape optimization method based on PSO are proposed to improve surface flattenability and to reduce shape deformation and distortion in surface unfolding process. A surface unfolding method is proposed based on the coordinate transformation and mass-spring model to improve unfolding efficiency in the surface unfolding process. Chapter 5 develops an image-based 3D model retrieval method based on feature skeletons of the model and model image. In Chapter 6, an optimization method of the 3D printing direction is developed based on the surface segmentation method with spectral clustering and deep-learning to improve the surface quality and reduce support materials of 3D printed products, followed by Chapter 7 to summarize the research and future work.

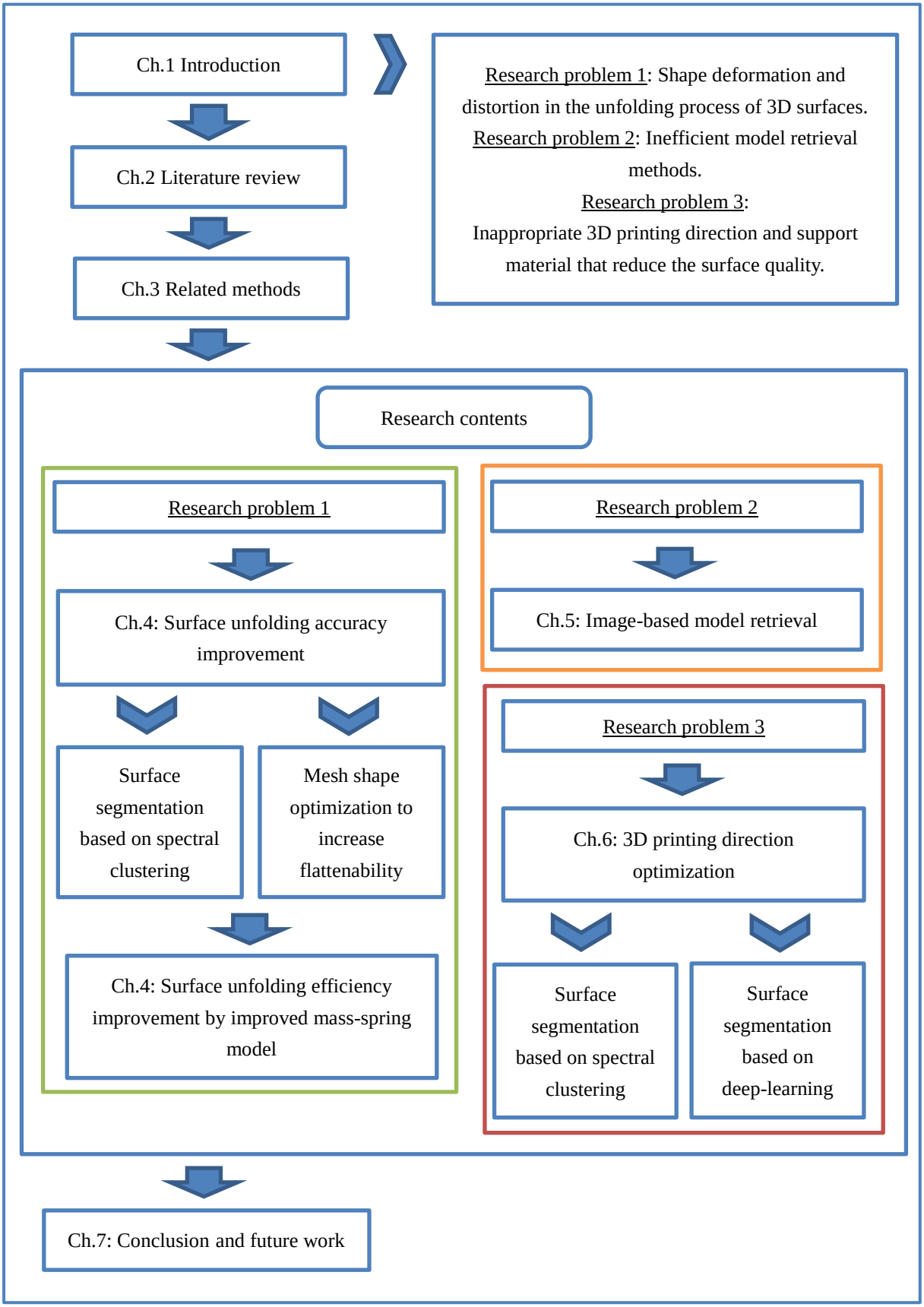

Figure 1-2 Structure of the thesis

Chapter 2 Literature review

Literature review covers topics related to research problems of this thesis as follows. (1) Surface unfolding. (2) Model retrieval. (3) Direction optimization of 3D printing products. (4) Shape segmentation. Applications of surface unfolding, model retrieval, and 3D printing are also reviewed.

2.1 Surface unfolding

Surface unfolding is mainly used in two areas, one is in texture mapping and the other is in the 3D-2D-3D process. In the texture mapping, surface parameterizations are generated from meshed 3D surfaces to perform annotations including color and lighting information (Smith and Schaefer, 2015). Research on unfolding a 3D surface into 2D patches using the 3D-2D-3D process mainly focuses on the improvement of efficiency and accuracy of unfolding algorithms. *Table 2-1* lists major methods and processes of surface unfolding.

Table 2-1 Surface unfolding research

Method	**Author**	**Process**
Mass-spring model	Wang et al, 2002	A triangle mesh surface was unfolded by selecting a central triangle and unfolding its neighbor triangles layer by layer. A mass-spring model was applied to reduce length and area errors.
	Li et al, 2005	Triangle strips were used to reduce the number of iterations in the surface unfolding. The mass-spring model was improved by adding a crossed spring to increase the optimization efficiency. They applied the unfolding process in the garment and shoe design.
	Liu et al, 2013	A simplified energy model for surface unfolding to unfold sheet metal components.
	Liu et al, 2016	Using triangle strips and improved mass-spring model with a crossed spring to build a flattening system for unfolding sheet metal components.
Origami	Xi et al, 2016	A strategy to combine unfolding and segmentation for designing paper craft. Before unfolding, the mesh model was simplified for semantics to avoid overlapping.
	Takahashi et al, 2011	A heuristic approach to unfolding simplified 3D triangular meshes into a single connected patch without shape distortions. A genetic-based algorithm was presented for optimizing usage of the paper sheet.
	Mitani and Suzuki, 2004	A strip-based approximation unfolding method for producing unfolded paper craft patterns of the 3D triangular mesh model.

Although the existing surface unfolding methods are widely used in different fields such as garment manufacturing and sheet metal shaping processes, these methods have following weaknesses. The number of patches is difficult to measure in the process of unfolding a 3D model into 2D single patch or multi-patches. In bolus shaping, air gaps are the noncontact distance between the human surface and bolus. Air gaps are difficult to evaluate during the bolus shaping. The number of iterations in the coordinate transformation process is very large. In addition, when using the mass-spring model to optimize unfolded patches, the efficiency is based on the original unfolded patches. If the original unfolded patches have a large deformation in shapes and contours, it is difficult to obtain an optimal result because virtual forces of the mass-spring model in the boundary are unbalanced. The mass-spring model always tries to reduce errors by moving positions of inner vertexes. However, shapes and contours of 2D patches have significant impacts on the accuracy when they are folded back to the 3D shape.

2.2 3D Model retrieval

Model retrieval searches similar models in a model base using properties of the model (Qin et al, 2016). Based on input of the model retrieval to search a 3D model, the model retrieval includes shape-based and image-based methods. The shape-based methods use model descriptions such as point cloud data, solid model or surface model as input to search a 3D model in a model base. The image-based model retrieval uses the image of a 3D model to search its similar 3D models in a model base. Based on matching contents of the model, the model retrieval includes feature-based retrieval and skeleton-based retrieval methods. For feature-based model retrieval methods, features denote geometric and topological properties of 3D models (Tangelder and Veltkamp, 2008). For skeleton-based model retrieval methods, the skeleton represents a geometric meaningful shape of the model for connection relations of components (Tangelder and Veltkamp, 2008). *Table 2-2* shows major literature in the model retrieval research.

Table 2-2 Model retrieval research

Category	Method	Author	Process
Input	Shape-based	Laga et al, 2006	The model retrieval used spherical wavelet-based shape descriptors.
		Akgül et al, 2009	The model retrieval used a probabilistic generative description of local shape properties and sampled multivariate probability density functions.
	Image-based	Aono and Iwabuchi, 2012	Zernike moments and Histogram of Oriented Gradient features were used.
		Shao et al, 2011	Contour-based matching was used
		Li and Johan, 2013	A context matching viewed the context of 2D sketches of a 3D model.
		Bell and Bala, 2015	A visual search for the interior design to train a joint embedding for two types of image objects using a convolutional neural network (CNN).
		Li et al, 2015	A modified CNN model to map the 2D image and 3D model into a same feature space for their similarity.
		Nie et al, 2019	A holistic generative adversarial network model for relations of the image and its model.
		Nie et al, 2020	A multi-branch graph CNN by searching similarity between 2D image and 3D model with visual information for a cross-modalities graph model.
Retrieval method	Features	Osada et al, 2002	Shape distribution-based similarity search between 3D shapes.
		Ip et al, 2002	Shape distribution-based method based on the comparison of solid models.
		Shih et al, 2007	The elevation descriptor describes the altitude information of 3D models from six different views.
		Furuya and Ohbuchi, 2009	To render 3D models into a set of depth images and locate visual features from images into a feature vector of the model.
		Li and Johan, 2013	A hybrid 3D shape descriptor and a class-based retrieval approach using the existing class information of the database to depict a 3D model from others using four features.
	Skeleton	Sundar et al, 2003	Skeletal points by a distance transform-based thinning algorithm to connect them into a skeleton using the Minimum Spanning Tree.
		Iyer et al, 2003, January and Lou et al, 2003	Skeleton formed by a thinning method to search models using a decision tree.

For the model retrieval, feature-based methods extract geometric and topological details of 3D models for all kinds of 3D models without restrictions. Skeleton-based methods build connection relations between 3D model components, but minor changes of the model topology may cause significant differences of the skeleton. Thus, skeleton-based methods are less robust than feature-based methods but they are good for partial matching of models. A common process of the image-based model retrieval builds relations of the image and 3D model using feature-based or skeleton-based methods. The skeletons built from features of the model and image can avoid information redundancy, which has not been used in existing model retrieval methods.

2.3 Direction optimization of 3D printing products

Printing direction plays an important role in the printing time, surface quality and weights of support materials for 3D printed products. As most 3D printers use a fixed printing plane to grow material in a single direction, the surface quality of a 3D printing product relies on its directionally layered printing process. It is therefore ideal that the surface normal of a 3D printing model is perpendicular to the printing direction for a high quality of the product surface. In addition, when the surface normal vector is toward the negative printing direction, or there is an angle larger than a standard between the surface and printing direction, it causes face overhang and needs adding support structure (Gardan and Schneider, 2015). The support structure will deteriorate the product surface finish when the support material is removed in post-processing. The support structure also increases the material use. Therefore, a 3D model should be printed following the surface direction for the high surface quality and using less support structure. *Table 2-3* shows major literature of the research on the 3D printing direction and support material reduction.

Table 2-3 3D printing

Method	**Author**	**Process**
3D printing direction optimization	Wang et al, 2016	A Ncut algorithm based on features of the surface quality
	Hao et al, 2011	Curvature analysis of the model surface to improve printing volume
	Zhang et al, 2005	A training-and-learning methodology to avoid placing supports in perceptually significant regions

	Yao et al, 2015	Dividing a model into multiple printable pieces to print a large model by small printers
Support material reduction	Hu et al, 2014	Using approximate pyramidal shapes to reduce support structures
	Liu and To, 2017	Using a novel level set-based topology optimization method for 3D printing model self-support to reduce support material.
	Guo et al, 2017	Using moving Morphable Components and Moving Morphable Voids frameworks to optimize a 3D printing model into a self-supportive model.
	Zhang and Zhou, 2018	Using a topology optimization method to design self-supporting structures where polygon-featured holes are basic design primitives and overhang constraints are introduced for controlling overhang angle.

Complex models with many branches cannot be printed in a whole piece with high quality because of requiring too many support structures. In addition, a large surface model cannot be printed as one piece because of the limited printer volume of existing 3D printers.

Traditional methods optimize a single global printing direction for the whole model to improve limitations of the surface quality and support materials or use an optimization method to achieve 3D printing model self-support for reducing support material. Model segmentation methods have been proposed in the 3D printing process to further improve the surface quality and reduce support materials. The existing methods of surface segmentation for 3D printing models have limitations in the feature extraction and clustering number decision. Features including the curvature, surface quality and support materials can be considered respectively in the model segmentation for 3D printing, but there is a lack of the method combing all the features.

2.4 Shape segmentation

Shape segmentation is commonly applied in computer-aided design to divide a geometric shape into several parts. A key of the segmentation is to find characteristics of different parts in the shape (Theologou et al, 2015). Two types of the segmentation are geometric (surface-type) and semantic (part-type) segmentations (Jiao et al, 2018).

The geometric methods segment a mesh shape into several patches based on the geometric characteristic. Geometric segmentation considers the position and structure of each part on a shape such as geodesic distances of the mesh, curvatures and types of the shape to separate the shape. For semantic methods, a mesh shape is segmented using the human perception theory (Shamir, 2008, September). Semantic segmentation aims at the study of human perception to separate 3D shapes into meaningful parts based on image understanding in computer vision.

In general, a shape segmentation process consists of two steps, feature extraction and shape segmentation based on features including special points, geodesic distance, shape curvature, convex region, concave region and shape skeleton. The shape can be segmented based on the extracted feature using methods of shape fitting, region growing, clustering, spectral analysis and neural networks. Feature points and skeleton of a 3D shape can be extracted based on analysis of the geodesic distance, shape curvature, convex or concave regions (Attene et al, 2006). Clustering and neural networks are two machine learning methods. A spectral analysis can be combined with clustering methods. Neural networks include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Stacked Auto-encoders (SAEs). CNN and RNN are supervised learning methods. Clustering and Stacked Auto-encoders are unsupervised methods. *Table 2-4* shows major literature of the shape segmentation.

Table 2-4 Surface segmentation research

Category	**Method**	**Author**	**Process**
Geometric	Curvatures	Lavoué et al, 2005	A curvature tensor method with similar curvatures using absolute values of the maximum and minimum principal curvatures.
	Curvatures and region growing	Wang and Yu, 2011	An algorithm based on curvature labeling. Morse theory and region growing technologies to segment molecular surfaces.
	Clustering	Golovinskiy and Thomas, 2008	Random cuts of the 3D mesh surface by clustering for stable cuts using statistical knowledge.
		Attene et al, 2006	An automatic hierarchical face algorithm by fitting primitives belonging to an arbitrary set.
	Minimizes a lost	Araújo et al,	Called Surface2Volume to generate assembly

	function	2019	parts affiliated with a single attribute. The outer surface of whose assembly conformed to the input surface geometry.
Semantic	Human-computer interaction	Zheng and Tai, 2010	An interactive process with part-brush and patch-brush by applying a cut along with an isoline of a harmonic field to form a small patch using a facet-based surface metric.
		Zheng et al, 2011	A simple interface called Dot Scissor for a directional search strategy supported by a concavity-aware harmonic field. The best isoline was selected for cutting using a robust voting.
		Miandarhoie et al, 2017	Data points are approximated by an extruded surface using a variational algorithm.
	Fuzzy C-means	Lerma and Biosca, 2005	A mode-seek algorithm to separate data points belonging to planar surfaces using a fuzzy C-means (FCM) algorithm to classify the point into various categories based on six point-based parameters.
	K-means	Chen et al, 2016	Improved K-means clustering to cluster surface point cloud into groups, discontinuity plane fitting and coordinate transforming to extract discontinuity orientation.
	Spectral analysis	Liu and Zhang, 2004	Spectral clustering using Graph Laplacian to select eigenvectors for data representation.
		Au et al, 2011	A score-based greedy algorithm to select best cuts in concavity-sensitive scalar fields using a Laplacian matrix.
		Wang et al, 2014	The hierarchical spectral analysis and isoline-based boundary detection. A concavity-aware Laplacian to obtain sequence of eigenvectors through eigen-decomposition.
		Zhang et al, 2018	A spectral clustering procedure based on a Laplacian matrix and a variational refining procedure.
		Jiao et al, 2018	A visual method to combine the mesh saliency and model characteristics using spectral clustering. A concave region identifying method to build a Laplacian matrix.
	Neural networks	Wang et al, 2018	A novel fully convolutional network architecture for shapes applied the convolution and pooling operations on images.
		Sun et al, 2016	A novel unsupervised method by using stacked auto-encoders to extract high-level features.

Clustering methods are most commonly used in the shape segmentation to not only cluster vertexes but also triangles. A distance combines different features to achieve the segmentation. Comparing with K-means methods, fuzzy clustering does not need to assign the number of clusters. Spectral clustering combines spectral analysis and clustering to transfer 3D models into a feature space for the part-type shape segmentation. Clustering methods can also be combined with other methods such as region growing, surface fitting and stacked auto-encoders for shape segmentation. Comparing with surface fitting, clustering and spectral analysis, neural networks methods have a high accuracy in the shape segmentation. Neural networks methods need more time to generate segments compared to other methods.

In applications of surface unfolding, there is a lack of the method to consider relationships between flattenability and surface segmentation to divide a free-form surface into several parts that can be unfolded with less distortion and deformation. In addition, the model segmentation decomposes a 3D model into a set of parts to enable each part to be printed with an optimal printing direction for the high surface quality and less support structure. Printing features are combined for dividing a 3D printing model into several parts to increase surface quality and reduce support structure. Clustering methods and neural networks can be improved to combine several features considering relationships between flattenability and surface segmentation.

2.5 Summary

This Chapter reviews research and methods of surface unfolding, model retrieval, direction optimization of 3D printing, and shape segmentation. The surface unfolding, model retrieval and 3D printing direction optimization all need the shape segmentation. Methods are proposed in following chapters to improve identified problems of surface unfolding, model retrieval and 3D printing applied in bolus shaping by developing methods of the shape segmentation.

Chapter 3 Related methods

To improve surface unfolding, model retrieval and 3D printing direction, the topological data structure of mesh shape, spectral clustering, Affinity Propagation (AP) clustering, and Stacked Auto-encoder (SAE) methods are introduced to assist the shape segmentation. A mass-spring model with crossed springs is built to optimize the surface unfolding process to reduce the number of iterations. The particle swarm optimization (PSO) is proposed to improve the surface mesh to increase surface flattenability for the surface unfolding accuracy.

3.1 Topological structure of mesh shape

Topological structure plays an important role in 3D surface analysis and needs to be built before the feature extraction (Zhang et al, 2019). The topological structure of a mesh surface *M* is defined as a set $\{V, E, F\}$ including vertices, edges and triangles, where $V = \left\{ v_i \middle| v_i \in R^3, 1 \le i \le m \right\}$ is a vertex set with *m* vertices. $E = \left\{ e_{ij} = (v_i, v_j) \middle| v_i, v_j \in V, i \ne j \right\}$ is an edge set, $F = \left\{ f_{ijk} = (v_i, v_j, v_k) \middle| v_i, v_j, v_k \in V, i \ne j,\ j \ne k, i \ne k \right\}$ is a triangle mesh set.

One-ring neighborhood vertices of vertex are vertices that have common edges with v_i. A set of one-ring neighborhood vertices of vertex v_i is defined as N^i. One-ring neighborhood triangles of vertex v_i are triangles composed by v_i and its one-ring neighborhood vertices. The set of one-ring neighborhood triangles of vertex v_i is defined as F^i. The normal vector of triangle *f* is defined as follows and shown in *Figure 3-1*.

$$N(f) = \frac{e_{ij+1} \times e_{ij}}{\left\| e_{ij+1} \times e_{ij} \right\|} = \frac{(v_i - v_{j+1}) \times (v_i - v_j)}{\left\| (v_i - v_{j+1}) \times (v_i - v_j) \right\|} \tag{3-1}$$

where, e_{ij} and e_{ij+1} are edge vectors from v_j to v_i and v_{j+1} to v_i, respectively.

The normal vector of v_i can be decided using Taubin's method as follows (Taubin, 1995).

$$N(v_i) = \frac{\sum_{f \in F^i} \text{area}(f) N(f)}{\left\| \sum_{f \in F^i} \text{area}(f) N(f) \right\|} \tag{3-2}$$

where F^i is a set of one-ring neighbor triangles for v_i.

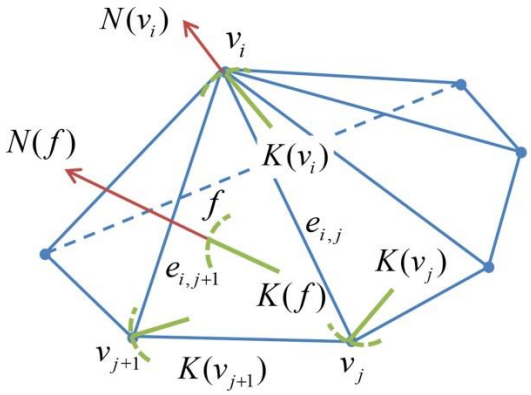

Figure 3-1 Triangular mesh

Based on mesh surface *M*, one-ring neighborhood vertices and one-ring neighborhood triangles of each vertex are searched. In addition, the area of each triangle, centroid of each triangle, three inner angles of each triangle, normal vector of each triangle, length of each edge and normal vector of each vertex are calculated. The process of the data structure preparation of mesh surface *M* is shown in Algorithm 3-1 as follows.

Algorithm 3-1: Topological data structure preparation

Input: Mesh surface *M*, number of vertices *m*, number of edge *n*, number of triangle *h*.

Output: Data structure of mesh shape *M*

1: **for** $i \leftarrow 1$ **to** *m*

2: Search one-ring neighborhood vertices of v_i

3: Search one-ring neighborhood triangles of v_i

4: **end for**

5: **for** $i \leftarrow 1$ **to** *n*

6: Calculate the length of e_i

7: Search boundary edges of mesh shape *M*

8: Search triangles with common edge e_i

9: **end for**

10: **for** *i* $\leftarrow$ 1 **to** *h*

11: Calculate the area of f_i

12: Calculate centroid of f_i

13: Calculate three inner angles of f_i

14: Calculate normal vector of f_i

15: **end for**

16: **for** *i* $\leftarrow$ 1 **to** *m*

17: Calculate normal vector of v_i

18: **end for**

3.2 Spectral clustering

Spectral clustering is evolved from the graph theory. The main idea of spectral clustering is to treat all data as points in space, and these points are connected by edges. The weight value of the edge between two apart points is lower than that between two close points. The purpose of spectral clustering is to make the sum weight of edges between sub-graphs as low as possible and the sum weight of edges in sub-graphs as high as possible by cutting the graph composed of all data points (Jiao et al, 2018).

For a triangular mesh model *M* with *m* vertices, the Laplacian matrix is defined in Eq. (3-3).

$$L = D - W \tag{3-3}$$

Laplacian matrix (*L*) is a semi-definite positive symmetric matrix. Matrix *W* is an adjacency matrix as follows.

$$W_{ij} = \begin{cases} \exp(-\dfrac{d_{ij}^2}{2\sigma^2}) & \text{if } (P_i, P_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3-4}$$

where d_{ij} measures the similarity between vertices P_i and P_j, σ is the average of d_{ij}. Matrix *D* is a diagonal matrix as follows.

$$D_{ii} = \sum_{j=1}^{m} W_{ij} \tag{3-5}$$

RatioCut is defined as follows to cut the graph *G* (*V*, *E*) into *k* subgraphs that are not connected to each other.

$$RatioCut(A_1, A_2, ..., A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} h_i^{\mathrm{T}} L h_i = Tr(H^{\mathrm{T}} L H) \tag{3-6}$$

where *H* is a matrix combined with indication vector $h_j = (h_{1j}, h_{2j}, ..., h_{nj})^{\mathrm{T}}$.

$$h_{ij} = \begin{cases} \dfrac{1}{\sqrt{|A_j|}} & v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \tag{3-7}$$

RatioCut can be transferred to an optimization problem as follows.

$$\arg \min \ Tr(H^T L H) \quad s.t.\ H^T H = I \tag{3-8}$$

The optimization searches each sub-object $h_i^{\mathrm{T}} L h_i$ in $Tr(H^{\mathrm{T}} L H)$, h is a unit orthogonal basis. Laplacian matrix *L* is a symmetric matrix. Base on the Rayleigh quotient, the maximum value of $h_i^{\mathrm{T}} L h_i$ is the maximum eigenvalue of *L*. The minimum value of $h_i^{\mathrm{T}} L h_i$ is the maximum eigenvalue of *L*. For sub-object $h_i^{\mathrm{T}} L h_i$, the goal is to find the smallest eigenvalue of *L*. For $Tr(H^{\mathrm{T}} L H)$, the goal is to find first *k* minimum eigenvalues. Corresponding *k* eigenvectors of the first *k* minimum eigenvalues are used to define a matrix $U^{(k)} = (u_1, u_2, \cdots, u_k) \in R^{n \times k}$, where the *i*-row of matrix *U* is a *k*-dimensional feature space of the mesh vertex v_i. After building a feature space, data in the feature space are clustered into several clusters by a K-means method or Gaussian mixture model (GMM).

3.3 Affinity Propagation (AP) clustering

AP clustering is an exemplar-based clustering algorithm to treat all sample points as potential cluster representative points and optimize the maximum clustering criterion function using the factor graph and belief propagation theory to generate clustering results. AP clustering does not need to assign the number of clustering and can directly use the existing cluster center (Frey and Dueck, 2007).

In the AP clustering algorithm, all sample points {1, 2, …, N} are nodes in the network. Based on similarity matrix $S=\{s(i,k)\}_{N\times N}(i,k=1,2,...,N)$, an optimal exemplar K is generated in a two-way message transfer process between nodes $(a(i,k),r(i,k))$ to search the global function of the network (clustering criterion function) for a maximum value until convergence as follows.

$$\mathrm{MAX}: S(C;s)=\sum_{i=1}^{N}s(i,c_i)+\sum_{k=1}^{N}\log f_k(C) \tag{3-9}$$

where, $s(i,k)$ is the similarity between point i and point k. $C=\{c_i\}$, c_i is exemplar of the point i. $s(i,c_i)$ is the similarity between point i and its exemplar c_i. $f_k(C)$ is constraint to limit the generation of clusters without exemplars.

$$f_k(C)=\begin{cases}0 & \text{if } c_k\neq k \text{ but } \exists i: c_i=k\\ 1 & \text{otherwise}\end{cases} \tag{3-10}$$

Values of $a(i,k)$ and $r(i,k)$ in AP clustering are updated in the iteration process. Availability $a(i,k)$ indicates an appropriate degree of point i to select point k as its exemplar from candidate exemplar k to data point i. Responsibility $r(i,k)$ indicates whether point k is suitable as the exemplar of point i from point i to candidate exemplar k.

AP clustering uses a max-product rule for data transformation to reduce complexity. $r(i,k)$ and $a(i,k)$ are updated using Eq. (3-11) and Eq. (3-12), respectively.

$$r(i,k)=s(i,k)-\max_{k'\neq k}(s(i,k')+a(i,k')) \tag{3-11}$$

$$a(i,k)=\begin{cases}\sum_{i'\neq k}\max\{0,r(i',k)\} & k=i\\ \min\{0,r(k,k)+\sum_{i'\neq i,k}\max\{0,r(i',k)\}\} & k\neq i\end{cases} \quad (3\text{-}12)$$

When the iteration is convergence, the corresponding class representative point of point i (c_i) is as follows.

$$c_i=\arg\ \max_k\{a(i,k)+r(i,k)\} \quad (3\text{-}13)$$

Comparing with other clustering algorithms such as *K*-means, Fuzzy C means and Spectral clustering, AP clustering does not need to specify the initial clustering and assign the number of clusters. The input similarity matrix can be a square matrix or a sparse matrix, and may be symmetrical or asymmetric.

To combine hidden features of a 3D printing model and the geometric distance between sub-graphs, a similarity matrix is formed as follows.

$$s(i,k)=\exp(-\frac{\|g_i-g_k\|^2}{2\sigma^2})+\|pf_i-pf_k\|^2 \quad (3\text{-}14)$$

where, g_i and g_k are coordinates of the central triangles for sub-graphs i and k in a 3D triangular model. σ is the average of $\|g_i-g_k\|^2$. pf_i and pf_k are hidden features extracted from the sub-graph data structure for sub-graphs i and k in a 3D triangular model by the Stacked Auto-encoder.

3.4 Stacked Auto-encoder

Auto-encoder is an unsupervised neural network model to learn hidden features of input data (Bengio et al, 2007). Auto-encoder is a system to generate the output as same as the original input. Two stages of Auto-encoder are encoding and decoding (Mao et al, 2020). Auto-encoder first encodes input data and then reconstructs the input data as output data that are same as input data in the decoding process. *Figure 3-2* shows a structure of Auto-encoder with an input layer, a hidden layer, and an output layer.

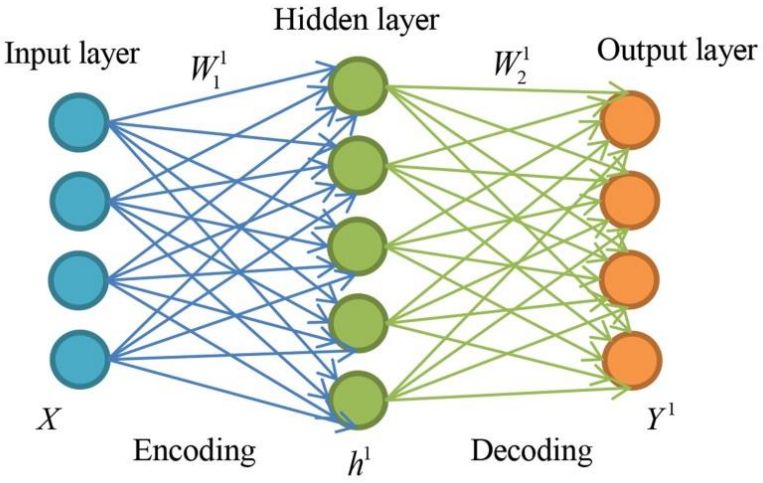

Figure 3-2 Structure of an Auto-encoder

The Auto-encoder generates a function to satisfy Eq. (3-15) as follows.

$$G_{W_1^1, W_2^1, b_1^1, b_2^1}(X) \approx X \tag{3-15}$$

where, $W_1{}^1$ and $W_2{}^1$ are the encoding matrix and decoding matrix, respectively. $b_1{}^1$and $b_2{}^1$ are bias vectors.

$G(\cdot)$ is divided into two stages in the encoding stage from the input layer to hidden layer as shown in Eq. (3-16), and decoding stage from the hidden layer to output layer as shown in Eq. (3-17).

$$h^1 = g_1(W_1^1 X + b_1^1) \tag{3-16}$$

$$Y^1 = g_2(W_2^1 \cdot h^1 + b_2^1) \tag{3-17}$$

Thus, the output is as follows.

$$Y^1 = g_2(W_2^1 \cdot g_1(W_1^1 \cdot X + b_1^1) + b_2^1) \tag{3-18}$$

A cost function is built to find the reconstruction error between the output and input in Eq. (3-19). The learning process is to minimize the cost function.

$$C(X, Y, \theta) = \frac{1}{2}\|Y - X\|^2 + \lambda D(\theta) \tag{3-19}$$

where λ is the weight of decay parameter, $\theta = \{W, b\}$ and $D(\theta) = \sum_{i=1}^{|\theta|} \theta_i^2$.

Auto-encoder can learn useful information when the number of units in the hidden layer is less than the number of input nodes. However, if the number of units in the hidden layer is larger than or equal to the input layer, Auto-encoder will learn some useless information. To avoid this problem, a sparsity constraint is added into Auto-encoder where only a part of units will be activated in the learning process.

A cost function with the sparsity constraint based on the Kullback–Leibler (KL) divergence (Pérez-Cruz, 2008) is defined in Eq. (3-20).

$$C_s(X,Y,\theta) = C(X,Y,\theta) + \beta \sum_{i=1}^{N_H} KL(\rho \,||\, \hat{\rho}_i) \tag{3-20}$$

$$KL(\rho \,||\, \hat{\rho}_i) = \rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i} \tag{3-21}$$

where, β is the weight of the sparsity penalty term, N_H is the number of hidden units and ρ is the sparsity parameter. $\hat{\rho}_i$ is the average activation of the *i*th hidden unit. $KL(\rho \,||\, \hat{\rho}_i)$ measures the difference between ρ and $\hat{\rho}_i$. If $\hat{\rho}_i = \rho$, the sparsity penalty term is zero.

Stacked Auto-encoders (SAEs) are based on the simple Auto-encoder by increasing the depth of its hidden layer to obtain better feature extraction capabilities and training effect. Stacked Auto-encoder is a neural network as shown in *Figure 3-3*. Unsupervised greedy layer-wised pre-training is used to train the stacked Auto-encoder to generate parameters of deep neural networks for desirable results.

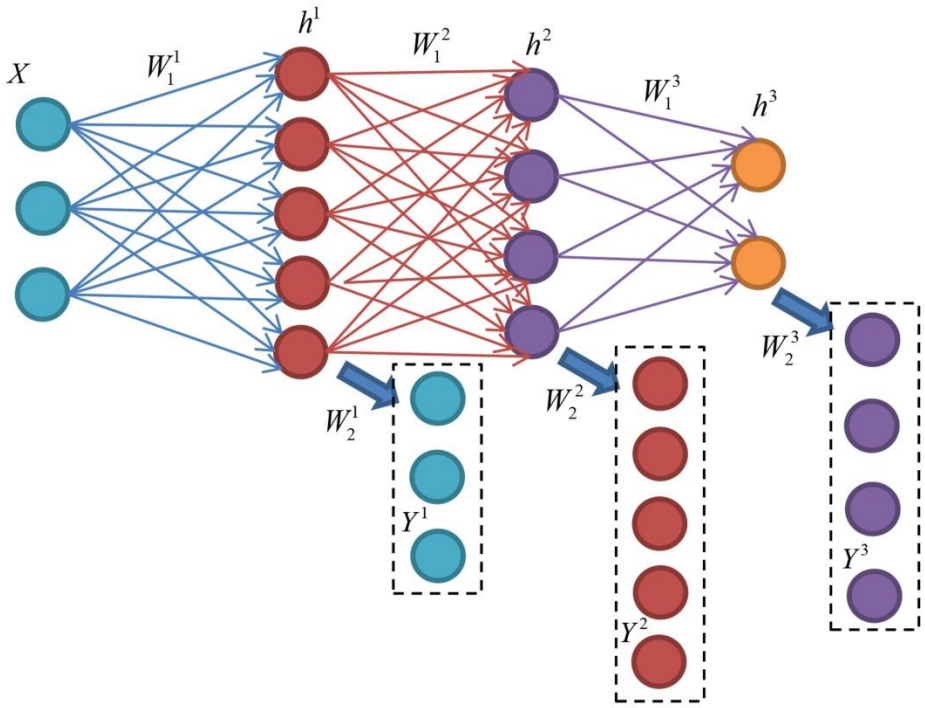

Figure 3-3 Structure of Stacked Auto-encoder

3.5 Mass-spring model with crossed springs

The mass-spring model with crossed springs can be used to control the shape of triangular meshes (Li et al, 2005). The crossed spring based mass-spring model has the high efficiency in optimizing 2D patches compared to the traditional mass-spring model because the mass-spring model with crossed springs consists of not only virtual masses and tension springs, but also crossed springs. *Figure 3-4* shows the structure of a crossed spring-based mass-spring model. A tension spring connects with two masses along with an edge to correct the length deformation of the edge. In addition, a crossed spring connects to two masses by crossing an edge where the two masses are on the edge of neighboring triangles (Li et al, 2005).

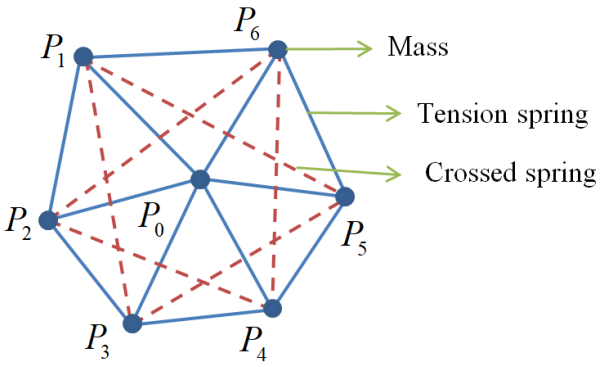

Figure 3-4 Mass-spring model with crossed springs

A pair (*K*, *P*) is used to determine the topological type of a triangular mesh surface $P=\{P_1,P_2,\cdots,P_m\}, P\in R^3$. Unfolded triangular mesh Φ is a pair (*K*, *Q*), $Q=\{Q_1,Q_2,\cdots,Q_m\}, Q\in R^2$, where *K* expresses the connection of vertices.

Node v_0 is linked to its adjacent node v_i by edge (v_0,v_i). The movement of node v_0 is restricted by spring force $f(Q_0,Q_i)$ from springs. For each node v_i, the total spring force $\vec{f}_i$ is calculated as follows.

$$\vec{f}_i=\sum_{j=1}^{N} c(\left|Q_iQ_j\right|-D_{P_iP_j})\vec{n}_{Q_iQ_j} \tag{3-22}$$

where *c* is the stiffness coefficient, $\left|Q_iQ_j\right|$ is the length between unfolded nodes Q_i and Q_j in 2D patches, $D_{P_iP_j}$ is the distance between original nodes P_i and P_j in a 3D surface, $\vec{n}_{Q_iQ_j}$ is the force direction. *N* is the number of one-ring neighboring nodes for Q_i.

The Lagrange equation is used to release energy in the mass-spring model as follows.

$$M\ddot{x}+Kx=0 \tag{3-23}$$

where *M* is mass, Kx is equal to $-\vec{f}_i$.

Euler's method is used to solve Eq. (3-23). It is assumed that the acceleration of

mass Q_i is constant when Δt is very small. For each node Q_i, mass m_i can be written as follows.

$$m_i = \frac{1}{3}\rho \sum \text{area}_k \tag{3-24}$$

where ρ is the density of area_k in a one-ring neighboring triangle of node Q_i. According to Newton's law, acceleration $\ddot{x}_i$ at node Q_i can be decided by Eq. (3-25).

$$\ddot{x}_i(t) = \frac{f_i(t)}{m_i} \tag{3-25}$$

Position $x_i(t_{n+1})$ and velocity $\dot{x}_i(t_{n+1})$ at step n+1 can be calculated based on position $x_i(t_n)$, velocity $\dot{x}_i(t_n)$ and acceleration $\ddot{x}_i(t_n)$ at step n as shown in Eqs. (3-26) and (3-27). The new spring force is updated using Eq. (3-22) based on results in Eq. (3-27).

$$\dot{x}_i(t_{n+1}) = \dot{x}_i(t_n) + \Delta t \cdot \ddot{x}_i(t_n) \tag{3-26}$$

$$x_i(t_{n+1}) = x_i(t_n) + \Delta t \cdot \dot{x}_i(t_n) + \frac{1}{2}\Delta t^2 \cdot \ddot{x}_i(t_n) \tag{3-27}$$

Deformations of edge lengths are shown by the elastic deformation energy $E(Q_0, Q_i)$. The total elastic energy $E(Q_i)$ is calculated by Eq. (3-28).

$$E(Q_i) = \sum_{j=1}^{N} \frac{1}{2} c(\left|Q_i Q_j\right| - D_{P_i P_j})^2 \tag{3-28}$$

The optimization objective is to minimize the total energy of the whole triangular mesh surface. The total energy E_{total} is calculated by Eq. (3-29).

$$E_{\text{total}} = \sum_{i=1}^{m} E(Q_i) \tag{3-29}$$

where, m is the total number of nodes in the whole mesh surface.

To evaluate the accuracy of unfolded 2D patches, two types of errors are proposed including the length error EL and area error EA as follows.

$$\text{EL} = \frac{\sum \left|L - L_0\right|}{\sum L_0} \tag{3-30}$$

$$\mathrm{EA} = \frac{\sum |A - A_0|}{\sum A_0} \tag{3-31}$$

where L_0 and A_0 are the edge length and triangle area on the triangular mesh 3D surface. *L* and *A* are the corresponding edge length and triangle area in unfolded 2D patches.

3.6 Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic evolution method based on the swarm intelligence. An individual in the population is considered as a particle without mass and volume, but velocity and position with dimension *D* in a searching space.

The position of each particle represents a feasible solution in the searching space. The particle moves at a constant speed, and the particle flight is the searching process for the feasible solution. The particle position is usually measured by a given objective function. The speed of a particle can be adjusted based on its best local and global positions compared to other particles. After multiple iterations, the best particle position can be found as the solution of an optimization problem. The evolution mechanism of a single particle in the canonical PSO is described as follows (Kennedy and Eberhart, 1995).

In a *D*-dimensional searching space, a particle population with NP particles has a vector $\boldsymbol{X}$ to represent position and velocity vector ς for its moving direction and speed. The *i*th particle in the population can be expressed by its position and velocity in the *t*th iteration, where the position vector is $X_i(t) = [x_i^1(t), x_i^2(t), \cdots x_i^D(t)]^{\mathrm{T}}$ and velocity vector is $\varsigma_i(t) = [\varpi_i^1(t), \varpi_i^2(t), \cdots \varpi_i^D(t)]^{\mathrm{T}}$. After *t* iterations, the best position of the *i*th particle (the best local position) can be found as $\mathrm{pbest}_i(t) = p_i(t) = (p_i^1(t), p_i^2(t), \cdots p_i^D(t))$. The best global position recorded as $gbest(t)$ is decided as follows.

$$\mathrm{gbest}(t) = \min\left\{\mathrm{pbest}_1(t), \mathrm{pbest}_2(t), \cdots \mathrm{pbest}_{\mathrm{NP}}(t)\right\} \tag{3-32}$$

At the *t*+1th step in the iteration, positional parameter $x_i^j(t)$ and velocity

parameter $\varpi_i^j(t)$ of the *i*th particle in the *j*th dimension are updated based on the information of the previous generation and current information as shown in Eqs. (3-33) and (3-34), respectively.

$$x_i^j(t+1) = x_i^j(t) + \varpi_i^j(t+1) \tag{3-33}$$

$$\varpi_i^j(t+1) = w\varpi_i^j(t) + c_1 r_1(\text{pbest}_i^j(t) - x_i^j(t)) + c_2 r_2(\text{gbest}^j(t) - x_i^j(t)) \tag{3-34}$$

where c_1 and c_2 are acceleration factors, r_1 and r_2 are random numbers with a uniform distribution, w is an inertia coefficient. Processing steps of PSO are as follows.

1) Defining population size NP and the maximum number of iterations T.

2) Setting an initial position of the particle: $x_i^j = x_{\min} + \text{rand}(0,1) \times (x_{\max} - x_{\text{mix}})$ and the initial velocity of the particle: $\varpi_i^j = \varpi_{\min} + \text{rand}(0,1) \times (\varpi_{\max} - \varpi_{\text{mix}})$.

3) If $t < \text{T}$, go to Step 4, else, the search stops.

4) Calculating objective function $f(x_i(t))$ for each particle in population, updating $\text{pbest}_i(t)$ for each particle and $\text{gbest}(t)$.

5) Calculating the updated position vector and velocity vector for particles using Eqs. (3-30) and (3-31), respectively. Go to Step 3).

The process of PSO iteration is shown in Algorithm 3-2.

Algorithm 3-2: PSO

1: $t \leftarrow 1$
2: $j \leftarrow 1$
3: $i \leftarrow 1$
4: **input** Initial population NP, Initial position ***X***, Initial velocity ***V***, Iteration number T, The dimension of searching space *D*
5: **while** *t* ≤ T (iteration number) **do**
6: **while** *i* ≤ NP (initial population) **do**
7: **for** *j*=1 **to** D (The dimension of searching space)
8: $\varpi_i^j(t) \leftarrow w\varpi_i^j(t-1) + c_1 r_1(\text{pbest}_i^j(t-1) - x_i^j(t-1)) + c_2 r_2(\text{gbest}^j(t-1) - x_i^j(t-1))$
9: $x_i^j(t) \leftarrow x_i^j(t-1) + \varpi_i^j(t)$
10: **end for**
12: $i \leftarrow i$+1

13: **end while**

14: Update the best local position $\text{pbest}(t) \leftarrow \{\text{pbest}_1(t), \text{pbest}_2(t), \cdots \text{pbest}_{\text{NP}}(t)\}$

15: Update the best global position $\text{gbest}(t) \leftarrow \min\{\text{pbest}_1(t), \text{pbest}_2(t), \cdots \text{pbest}_{\text{NP}}(t)\}$

16: $t \leftarrow t+1$

17: **end while**

18: **output** gbest(t)

3.7 Summary

Existing methods related to this research are introduced to improve the shape segmentation for surface unfolding, model retrieval and 3D printing direction optimization. The methods justifications are as follows. For the spectral clustering method, the Laplacian matrix can extract features of the model based on the purpose of segmentation and then cluster the model into several parts based on the features. AP clustering can divide a 3D model into several clusters to improve automatic decision-making of the number of clusters. SAEs are unsupervised learning that can be used to extract hidden features from models for segmentation. PSO has the good performance to optimize data without calculating derivatives of the objective function. The initial input of PSO does not have much influence on the final result and convergence rate.

Chapter 4 Surface unfolding improvement by surface segmentation

As a non-developable 3D surface cannot be unfolded into 2D planes without deformation, surface segmentation is required to divide a non-developable 3D surface into a few of pieces before unfolding them separately to meet the accuracy requirement.

A surface segmentation method is therefore proposed based on spectral clustering. Segmentation saliency is used to search potential positions in a 3D mesh model to avoid the large deformation in unfolding. A Laplacian matrix is designed to improve the segmentation accuracy by combing the segmentation saliency and geometric information. The spectral clustering method is used to divide the 3D mesh model into several parts. Cutting lines between two adjacent parts are smoothed to reduce difficulties of refolding 2D patches into the 3D model by connecting the segmented parts. A mesh shape optimization method is proposed to improve surface flattenability and measure air gaps. A strategy of the dimension reduction is proposed for the selection of local nodes on a meshed surface to increase the searching efficiency of surface flattenability. A local node selection based PSO (L-PSO) method is developed to search the optimal solution.

To improve the unfolding efficiency from a 3D surface to 2D planes, triangle crossings are used to reduce the number of iterations. To improve efficiency of contours optimization of unfolded 2D planes, a disturbing spring is added into the mass-spring model with crossed springs to change shape of the unfolded 2D plane. Surface unfolding is a coordinate transformation process from a 3D surface to its corresponding 2D planes where a triangle mesh in the 3D surface model is fixed as the central triangle. After unfolding the central triangle, a central triangle crossing is chosen to unfold other triangles of the model without deformation.

4.1 Surface segmentation by integrated spectral analysis and clustering

4.1.1 Flattenability and a new Laplacian matrix

Flattenability ϖ measures a surface to be unfolded into 2D planes based on the

summed inner angle in each vertex (Wang, 2008). $\varpi(v_i)$ at v_i is defined in Eq. (4-1). A lower value of $\varpi(v_i)$ means that when unfolding one-ring neighborhood triangles of v_i into a 2D plane, the unfolded 2D plane has less deformation and distortion.

$$\varpi(v_i) = |\theta(v_i) - 2\pi| \tag{4-1}$$

where $\theta(v_i) = \sum_j \theta_j$ is the summed inner angle in vertex v_i as shown in *Figure 4-1*. If $\theta(v_i) = 2\pi$, one-ring neighborhood triangles of v_i can be unfolded into a 2D plane without distortion and deformation. If $\theta(v_i) - 2\pi > 0$, unfolding one-ring neighborhood triangles of v_i will cause overlaps. If $\theta(v_i) - 2\pi < 0$, unfolding one-ring neighborhood of triangles of v_i will cause gaps.

$\sum \varpi$ is the summarized value of ϖ in a model as follows.

$$\sum \varpi = \sum_{i=1}^{m} \varpi(v_i) \tag{4-2}$$

Flattenablility value (FLV) is applied to measure the surface flattenablility as follows.

$$\mathrm{FLV} = 10 \times 0.5^{\sum_{i=1}^{m} \varpi(v_i)} \tag{4-3}$$

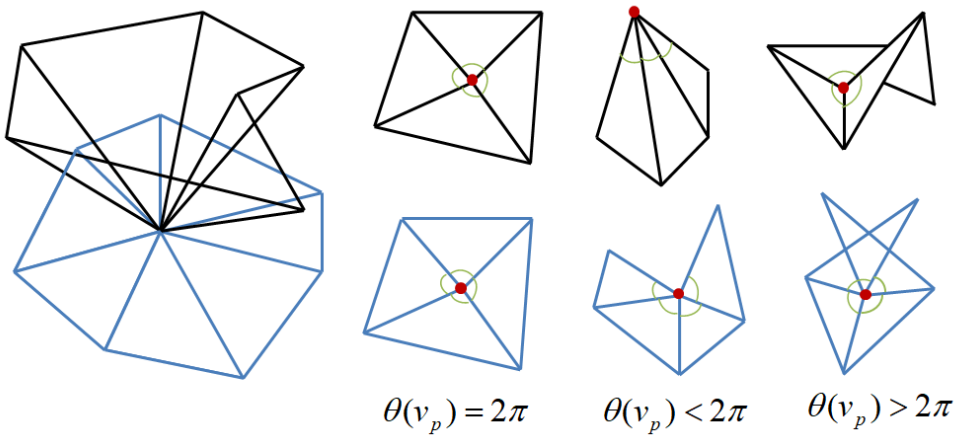

Figure 4-1 Deformation of different inner triangular meshes (Wang, 2008)

The segmentation saliency is defined in Eq. (4-4).

$$s(v_p) = \begin{cases} \theta(v_p) - 2\pi & \text{if } \theta(v_p) - 2\pi > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4\text{-}4)$$

Spectral analysis and clustering methods are then integrated to divide the 3D mesh model into several parts. A Laplacian matrix is formed to improve the segmentation accuracy by combing the surface segmentation saliency and geometric information.

d_{ij} in adjacency matrix is then decided using Eq. (4-5) to combine the segmentation saliency and topology of the 3D shape.

$$d_{ij} = \eta \cdot |s'(v_i) - s'(v_j)| + e'_{ij} \quad (4\text{-}5)$$

where $s'(v_i)$ and $s'(v_j)$ are the segmentation saliency of vertices v_i and v_j after being transferred into interval between 0 and 1, respectively. e'_{ij} is the length of an edge to connect vertices v_i and v_j being transferred to interval between 0 and 1. If both vertices v_i and v_j of edge e_{ij} have a large value of the segmentation saliency, the area will be unfolded with a large deformation. Its flattenability should be increased. $\eta = 5$ is assigned as a threshold if both v_i and v_j have the high segmentation saliency, otherwise $\eta = 0$.

4.1.2 Boundary optimization and topology construction

Clustering results of the segmentation cannot be used directly as the clustering method divides nodes into several groups that do not have common nodes in a 3D mesh model. Gaps will be formed when refolding the segmented parts because of missing a set of edges.

A set of triangle strips are added to segmented parts as common boundaries to avoid gaps. Flattenability is applied to measure the error between the 2D and 3D shapes. However, new boundaries in segmented parts are not smooth, which increases difficulty of unfolding and refolding processes. A process of designing smooth cutting

lines and building new topologies is therefore proposed to reduce difficulties of refolding 2D patches into the 3D model.

After clustering, the common boundary is a set of edges in the triangle strip, which may increase difficulty and complexity of refolding. To generate a smooth curve in a 3D mesh surface without changing shape of the original surface, a surface is built to use its intersection with the original 3D shape as the smooth cutting line.

A plane Γ (blue plane in *Figure 4-2* (a)) is the tangent plane estimated by nodes in the common boundary represented by $p_1x + p_2y + p_3z + 1 = 0$ and its normal vector is $\vec{d}_\Gamma = (p_1, p_2, p_3)$.

Nodes in the common boundary are projected onto plane Γ and fitted to a curve l in plane Γ by the least square method. The 3D coordinate of node v_1 is $v_1 = (x_1, y_1, z_1)$. To project v_1 onto plane Γ, the new coordinate of v_1 is $v_{1\Gamma} = (x_{1\Gamma}, y_{1\Gamma}, z_{1\Gamma})$ as follows.

$$\frac{x_{1\Gamma} - x_1}{p_1} = \frac{y_{1\Gamma} - y_1}{p_2} = \frac{z_{1\Gamma} - z_1}{p_3} \tag{4-6}$$

$$x_{1\Gamma} = \frac{p_1}{p_1^2 + p_2^2 + p_3^2} \cdot \left(\frac{p_2^2 + p_3^2}{p_1} \cdot x_1 - p_2 \cdot y_1 - p_3 \cdot z_1 \right) \tag{4-7}$$

$$y_{1\Gamma} = \frac{p_2}{p_1}(x_{1\Gamma} - x_1) + y_1 \tag{4-8}$$

$$z_{1\Gamma} = \frac{p_3}{p_1}(x_{1\Gamma} - x_1) + z_1 \tag{4-9}$$

To reduce difficulty of unfolding and refolding, linear and quadratic functions are applied in the curve fitting process to generate curve *l*. Surfaces S (orange surface in *Figure 4-2* (a)) is built to be vertical with the tangent plane and intersect with the initial 3D shape to generate a smooth cutting line to divide the original 3D shape into two parts.

After generating a new smooth common boundary, new vertices, edges and triangles are formed in segmented parts to change the original topology. A new edge

generated from a smoothed boundary can divide a triangle of the original 3D shape into two parts, one is a triangle (the orange shape in *Figure 4-2* (b)) and the other is a quadrilateral (the green shape in *Figure 4-2* (b)). New vertices are intersection points between surface S and smoothed common boundary. Thus, for the new triangle (the orange shape in *Figure 4-2* (b)), two new vertices (*Figure 4-2* (b), vertices D and E), three new edges (edges BD, BE and DE in *Figure 4-2* (b)), and a new triangle (BDE in *Figure 4-2* (b)) are added in the new topology. For the new quadrilateral (the green shape in *Figure 4-2* (b)), an additional new edge (AE in *Figure 4-2* (b) is added to segment the quadrilateral into two triangles (ACE and ADE in *Figure 4-2* (b)). Thus, two new vertices (D and E in *Figure 4-2* (b)), three new edges (AD, AE and CE in *Figure 4-2* (b)), and two new triangles (ACE and ADE in *Figure 4-2* (b)) are added in the new topology.

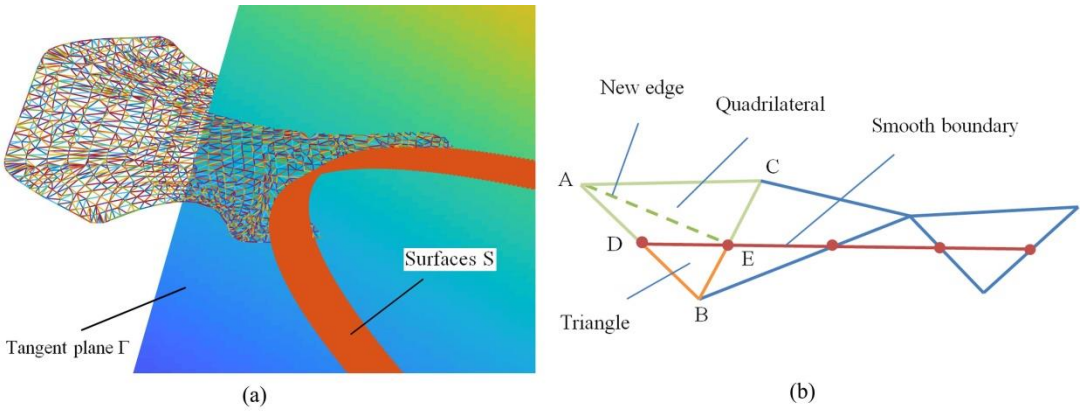

Figure 4-2 Fitting surface and generating cutting line

4.2 Results of surface segmentation

A nose model is used to test the developed surface segmentation method. The segmentation saliency of the nose model is decided as shown in *Figure 4-3* (a), where red areas have the high segmentation saliency, blue areas have the low segmentation saliency. Therefore, when unfolding the nose surface model, the red areas may cause the large deformation.

A Laplacian matrix is built based on the proposed method in Chapter 4.1.1. Eigenvectors of the Laplacian matrix is then used to build a feature space. *Figure 4-3* shows the spectral embedding of the nose model, where (b) shows the original 3D

model shape, (c) displays a 2D feature space of the nose model, and (d) illustrates a 3D feature space of the nose model. Red points in each diagram are vertices with the high segmentation saliency. The 3D nose model is extracted into a feature space based on the topology and segmentation saliency. Some of red points are concentrated and some of them are scattered in both of 2D and 3D feature spaces. The 2D feature space can show this kind of features clearly. Therefore, points in the 2D feature space can replace vertices in the 3D mesh shape to be clustered to generate segmentations.

The spectral clustering method requires an assigned clustering number. To compare the segmentation effect, this example assigns clustering numbers as 2, 3, 4 and 5. *Figure 4-4* shows clustering results of the nose model with clustering numbers 2, 3, 4 and 5, respectively, where (a) is clustering results of the nose model with clustering number 2 (the yellow part for part 1, blue for part 2), (b) shows clustering results of the nose model with clustering number 3 (the orange part for part 1, green part for part 2, blue part with two patches for part 3,). *Figure 4-4* (c) shows results of the nose model with clustering number 4 (the green part for part 1, orange for part 2, yellow for part 3, and blue for part 4). *Figure 4-4* (d) illustrates clustering results with clustering number 5 (the purple part for part 1, orange for part 2, blue for part 3, green for part 4, and yellow for part 5).

Table 4-1 lists results of $\sum \varpi$ for pieces with different clustering numbers after adding a common boundary. $\sum \varpi$ decreases with segmentation. The more parts are divided; the less $\sum \varpi$ each part will have. However, many parts will bring a complicated refolding process for the clinic application. To balance the accuracy of unfolding the original 3D model into 2D patches and difficulties of refolding unfolded 2D patches into the 3D model, a maximal clustering number is limited to 3 for the surface segmentation in total 4 pieces.

After generating clustering results, a smooth cutting line is designed based on the proposed method. $\sum \varpi$ is shown in *Table 4-2* based on numbers of nodes, edges and triangles for segmented 3D pieces with a smooth boundary. Results show that

segmented 3D pieces with a smooth boundary have more nodes, edges and triangles compared to segmented 3D pieces without a smooth boundary. Values of $\sum\varpi$ are reduced in parts 1 and 3 (with 2 pieces), while the value is increased a bit in part 2.

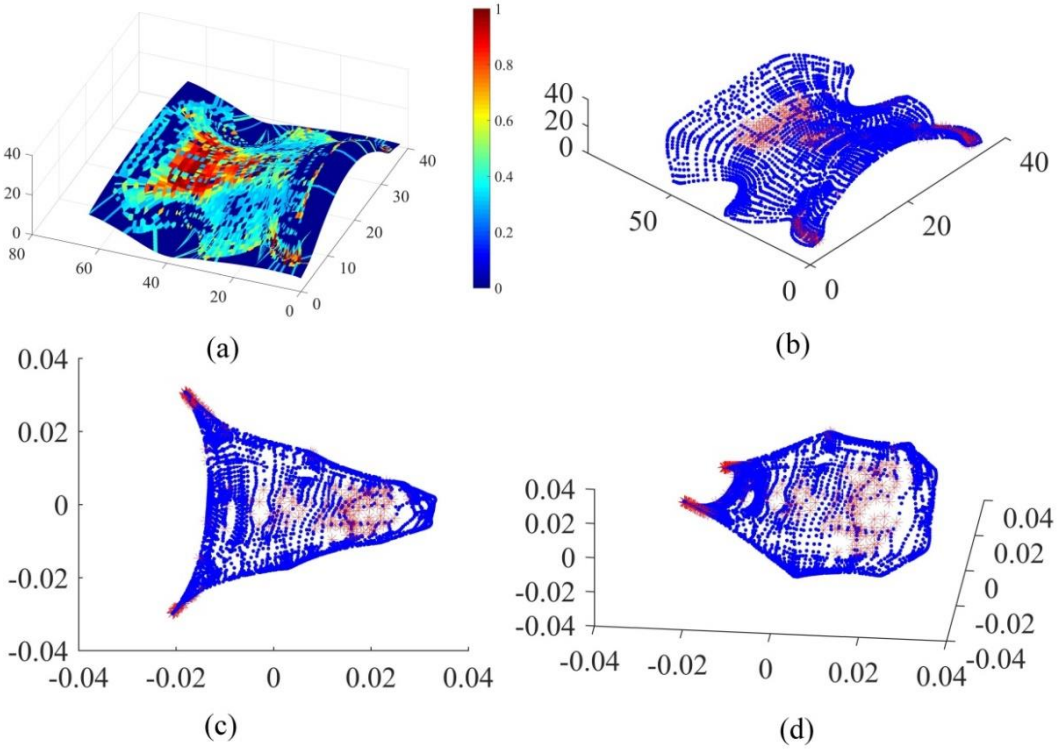

Figure 4-3 Segmentation saliency and spectral embedding of a nose model

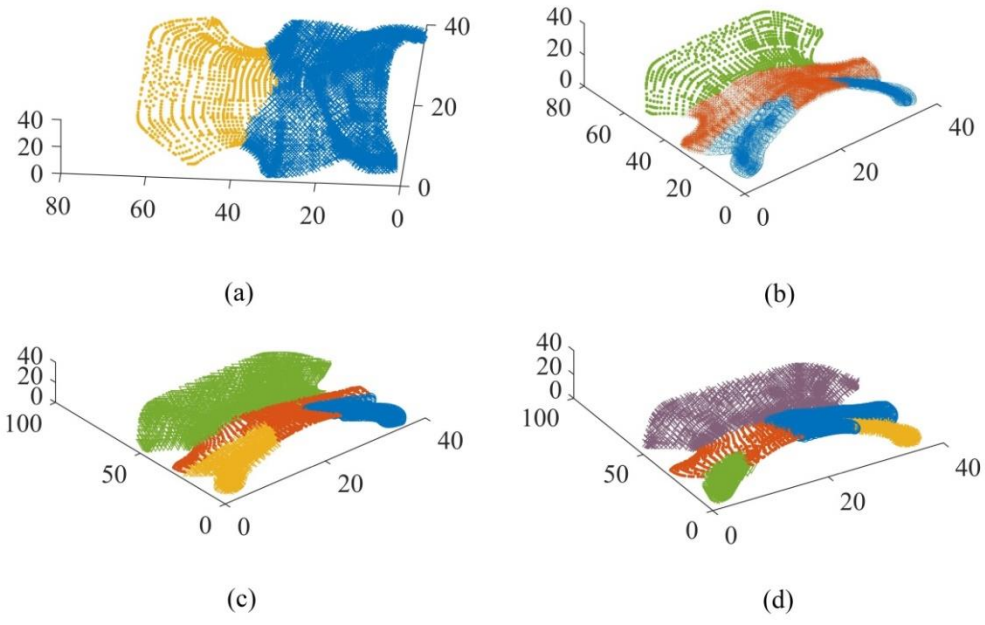

Figure 4-4 Clustering results of the nose model with clustering numbers 2, 3, 4 and 5

Table 4-1 $\sum \varpi$ and FLV of the nose model by segmentation without smooth boundary

3D nose model		Vertices	Edges	Triangles	$\sum \varpi$	FLV
Without segmentation		4019	11762	7744	7.9636	0.0401
Clustering number=2	Part1	1094	3140	2047	2.4006	1.8939
	Part2	2970	8667	5698	5.3835	0.2396
Clustering number=3	Part1	886	2522	1637	1.6460	3.1952
	Part2	1598	4578	2990	2.7243	1.5132
	Part3(1)	811	2344	1534	1.7255	3.0239
	Part3(2)	843	2431	1589	1.6917	3.0956
Clustering number=4	Part1	1002	2869	1868	2.1304	2.2839
	Part2	1357	3872	2516	2.2584	2.0900
	Part3	897	2587	1691	1.7054	3.0664
	Part4	880	2548	1669	1.7215	3.0323
Clustering number=5	Part1	1040	2982	1943	2.2783	2.0614
	Part2	929	2663	1735	1.1474	4.5144
	Part3	917	2613	1697	1.2036	4.3419
	Part4	530	1515	986	1.4231	3.7291
	Part5	732	2115	1384	1.6262	3.2394

Table 4-2 $\sum \varpi$ and FLV of the nose model by segmentation with smooth boundary

	Vertices	Edges	Triangles	$\sum \varpi$	FLV
Part1	958	2780	1728	1.5537	3.4064
Part2	1877	5280	3404	2.8392	1.3974
Part3(1)	832	2367	1536	1.7112	3.0541
Part3(2)	870	2457	1588	1.6688	3.1451

4.3 Mesh Surface optimization

4.3.1 Objective function to improve surface

Searching for improving surface M_{Inp} from M is considered as a constrained optimization problem. The improved surface M_{Inp} should approximate the shape of original triangular mesh surface M. Improved position $\boldsymbol{v}_p$ of a vertex with original position $\boldsymbol{v}_p{}^0$ is an ideal position to satisfy the optimization constraint. The relationship between $\boldsymbol{v}_p$ and $\boldsymbol{v}_p{}^0$ can be presented as: $\boldsymbol{v}_p = \boldsymbol{v}_p^0 + \boldsymbol{d}_p$. $\boldsymbol{d}_p$ is the vector of a node v_p

optimized from the position of original mesh surface to the position of improved mesh surface. Relationships between the original mesh surface and improved mesh surface are shown in *Figure 4-5* using a triangle mesh.

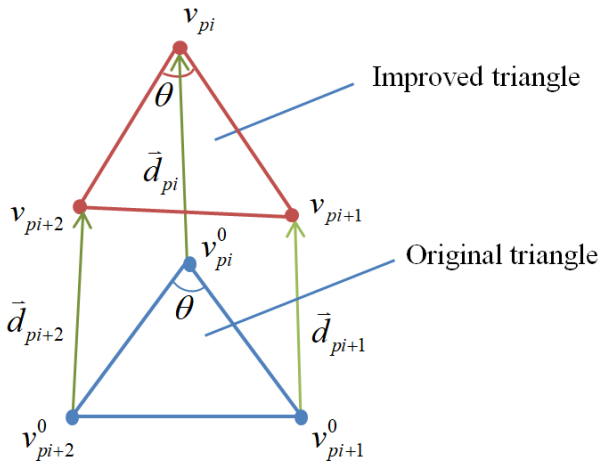

Figure 4-5 Relationship between an original mesh surface and improved mesh surface

Air gaps between the refolded model and original surface are used to measure the unfolding accuracy. To reduce air gaps between the refolded model and original surface, the shape of the improved surface should be similar to the original surface. Thus, all the vertices in the original mesh surface should be moved in the positive direction of normal vectors of an outer surface. The moving distance can be calculated as air gaps between the original surface and refolded models. As both original surface and refolded models are continuous surface, the original surface should be converted into a discrete mesh surface as smooth as a continuous surface.

In short, the process uses four requirements and one constraint to search the surface improvement with a high flattenability as follows. 1) Minimum deformation and distortion, 2) Maximum smoothness of the surface, 3) Maintaining the shape of original mesh model, 4) Minimum air gaps, and one constraint of moving nodes on the original surface in the positive direction of normal vectors of an outer surface.

Based on definition of the flattenable mesh surface in Chapter 4.1.1, $\omega(v_p)$ at v_p is defined to measure the flattenablility in v_p. A smaller $\omega(v_p)$ at vertex v_p in a triangular mesh surface means that the surface can be unfolded into 2D planes in less

distortion and deformation. For an improved 3D surface M_{Inp}, if its summed inner angles are closer to 2π, surface M_{Inp} can be unfolded into 2D planes in less deformation and distortion. To generate an improved surface for flattenability, the surface should satisfy Eq. (4-10).

$$\min \sum_{v_p \in V_{in}} \left| \theta(v_p) - 2\pi \right| \tag{4-10}$$

The Laplacian mesh is widely used to remove rough features from irregularly triangular mesh surfaces for maintaining desirable geometric features (Desbrun et al, 1999). For a mesh surface expressing with triples $\{V, E, F\}$, $\boldsymbol{v}_p$ denotes the position of vertex v_p in a 3D coordinate as follows.

$$\boldsymbol{v}_p - \frac{1}{\left|N^p\right|} \sum_{j \in N^p} \boldsymbol{v}_p = 0 \tag{4-11}$$

where N^p is a set of 1-ring neighboring vertices of v_p, v_p is an interior vertex ($v_p \in V_{in}$), and $\left|N^p\right|$ denotes the number of elements in the set of 1-ring neighboring vertices of v_p. The linear system can be rewritten in the matrix as Eq. (4-11)

$$\begin{aligned} L\boldsymbol{x} &= 0 \\ L\boldsymbol{y} &= 0 \\ L\boldsymbol{z} &= 0 \end{aligned} \tag{4-11}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are $n \times 1$ vectors that contain x, y and z coordinates of vertices. Matrix L is a Laplacian operator as shown in Eq. (4-12).

$$L_{i,j} = \begin{cases} 1 & i = j \\ -\dfrac{1}{\left|N^p\right|} & j \in N^p \\ 0 & \text{otherwise} \end{cases} \tag{4-12}$$

When fixing boundary vertices, coordinates of inner vertices on a Laplacian mesh surface are decided by the membrane functional minimization $\int_\Omega \left\| v_\mu \right\|^2 + \left\| v_\nu \right\|^2 \mathrm{d}\mu \mathrm{d}\nu$ (Desbrun et al, 1999). The optimized mesh surface from Eq. (4-11) is a smooth mesh surface which is uniquely defined by the original surface and position of boundary vertices. However, a Laplacian mesh surface is not a flattenable

mesh surface. To generate a smooth mesh surface, the optimization function is derived from the Laplacian mesh as shown in Eq. (4-13).

$$\min \sum_{v_p \in V_{in}} \left\| L\boldsymbol{v}_p \right\|^2 \tag{4-13}$$

The area of the unfolded surface or improved surface should be as same as the area of the original surface. Area accuracy is applied to evaluate the area difference between a 3D shape and the improved 3D shape (Wang et al, 2002). To evaluate difference between the original surface and improved surface, area accuracy ε_A is defined in Eq. (4-14).

$$\varepsilon_A = \frac{\sum \left| \text{area}(f) - \text{area}(f)^0 \right|}{\sum \text{area}(f)^0} \tag{4-14}$$

where A_f^0 is the area of triangle f in original surface M, A_f is the area of triangle f in improved surface M_{inp}. To generate the improved surface similar to the original surface area, areas of triangles in two surfaces should satisfy Eq. (4-15).

$$\min \frac{\sum \left| \text{area}(f) - \text{area}(f)^0 \right|}{\sum \text{area}(f)^0} \tag{4-15}$$

Air gaps between the improved surface and original surface can be calculated as the distance between new positions and original positions of vertices. To generate the improved surface with smaller air gaps, the distance between new positions and original positions of vertices should satisfy Eq. (4-16).

$$\min \sum_{v_p \in V_{in}} \left\| \boldsymbol{v}_p - \boldsymbol{v}_p^0 \right\|^2 \tag{4-16}$$

When improving a surface, nodes in the surface should be moved in the positive direction of normal vectors of an outer surface. Thus, the original coordinate of each node is transferred to a new coordinate where the positive direction of the *z*-axis is the normal vector of the node. A constraint is therefore added in the optimization model to ensure that all the nodes are moved in the positive direction of the *z*-axis.

The computational scheme for optimizing the surface with four defined requirements and one constraint is formulated into a constrained multi-objective optimization problem as shown in Eq. (4-17).

$$
\begin{aligned}
&\min \sum_{v_p \in V_{in}} \left\| \boldsymbol{v}_p - \boldsymbol{v}_p^0 \right\|^2 \\
&\min \sum_{v_p \in V_{in}} \left| \theta(v_p) - 2\pi \right| \\
&\min \sum_{v_p \in V_{in}} \left\| L\boldsymbol{v}_p \right\|^2 \\
&\min \frac{\sum \left| \text{area}(f) - \text{area}(f)^0 \right|}{\sum \text{area}(f)^0} \\
&s.t. \ \left\| \boldsymbol{v}_p - \boldsymbol{v}_p^0 \right\|^2 = 0 \ \ v_p \notin V_{in} \\
&\quad v'_{pz} - v'^0_{pz} > 0
\end{aligned}
\tag{4-17}
$$

where V_{in} is a set of inner vertices that are allowed to move during the optimization search, and boundary vertices are fixed. v'_{pz} and v'^0_{pz} are directions of normal vectors for v_p and v_p^0, respectively. The normal vector of vertex v_p is estimated by Eq. (3-2) in Chapter 3.1.

A spatial coordinate transformation is applied to transfer the original coordinate of each node to a new coordinate where the positive direction of the z-axis is the normal vector of the node. The transformation process is shown in *Figure 4-6* and Eq. (4-18).

$$
\begin{bmatrix} v'_{px} \\ v'_{py} \\ v'_{pz} \end{bmatrix} = R(\alpha)R(\gamma) \begin{bmatrix} v_{px} \\ v_{py} \\ v_{pz} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{px} \\ v_{py} \\ v_{pz} \end{bmatrix} \tag{4-18}
$$

where α and γ are calculated by Eqs. (4-19) and (4-20), respectively.

$$
\alpha = -\arccos\left(\frac{N_{vpx}}{\sqrt{N_{vpx}^2 + N_{vpy}^2 + N_{vpz}^2}} \right) \tag{4-19}
$$

$$
\gamma = -\arccos\left(\frac{N_{vpy}}{\sqrt{N_{vpx}^2 + N_{vpy}^2}} \right) \tag{4-20}
$$

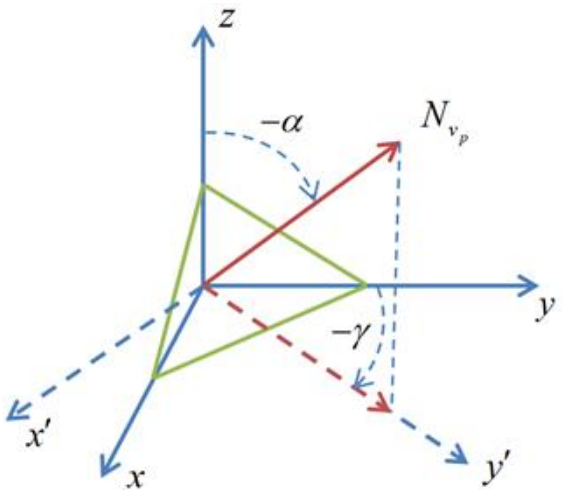

Figure 4-6 Candidate transformation process

Eq. (4-17) can be transferred into a constrained optimization problem as Eq. (4-21).

$$J(X)=\mu_1\sum_{v_p\in V_{in}}\left|\theta(v_p)-2\pi\right|+\mu_2\sum_{v_p\in V_{in}}\left\|L\boldsymbol{v}_i\right\|^2+\mu_3\sum_{v_p\in V_{in}}\left\|\boldsymbol{v}_p-\boldsymbol{v}_p^0\right\|^2+\mu_4\frac{\sum\left|\text{area}(f)-\text{area}(f)^0\right|}{\sum\text{area}(f)^0}$$

$$s.t.\ \ \left\|\boldsymbol{v}_p-\boldsymbol{v}_p^0\right\|^2=0\ \ v_p\notin V_{in}$$

$$v'_{pz}-v'^0_{pz}>0$$

(4-21)

Four coefficients μ_1, μ_2, μ_3 and μ_4 are weights of each part in the optimization model and assigned by users. μ_1=10, μ_2=1.0, μ_3=0.1 and μ_4=10 are used based on the magnitude of each part for all examples. The value of μ_3 must be much smaller than μ_2 for the system stability (Huang et al, 2012). Additionally, for increasing the surface flattenability, μ_1 should be larger than μ_2 and μ_3. The order of magnitudes for area accuracy is less than other requirements, because the area accuracy is measured in percentage. Thus, μ_4 should be larger than μ_2 and μ_3.

4.3.2 Optimization of search processing

As the large number of nodes in a triangular meshed surface, optimization of the

search process is normally a high-dimensional computational problem. To improve the efficiency of the optimization process, a local node selection strategy is proposed in the optimization process to decide initial pre-optimized nodes. The local vertices selection strategy is the evaluation of nodes that contribute large $\omega(v_p)$ for the surface.

$\omega(v_p)$ is used to search contribution of each node in the surface improvement process. These nodes with the large value of $\omega(v_p)$ are selected as high contribution nodes.

All the nodes in the original surface should be moved only to the positive direction of the z-axis. If a node meets $\theta(v)-2\pi>0$, the node will be moved to the positive direction of the z-axis to improve the flattenability as shown in *Figure 4-7* (b). While if a node $\theta(v)-2\pi<0$, the node will be moved to the negative direction of the z-axis. One-ring neighbor nodes for a node with $\theta(v)-2\pi<0$ can be moved to the positive direction of the z-axis as shown in *Figure 4-7* (a).

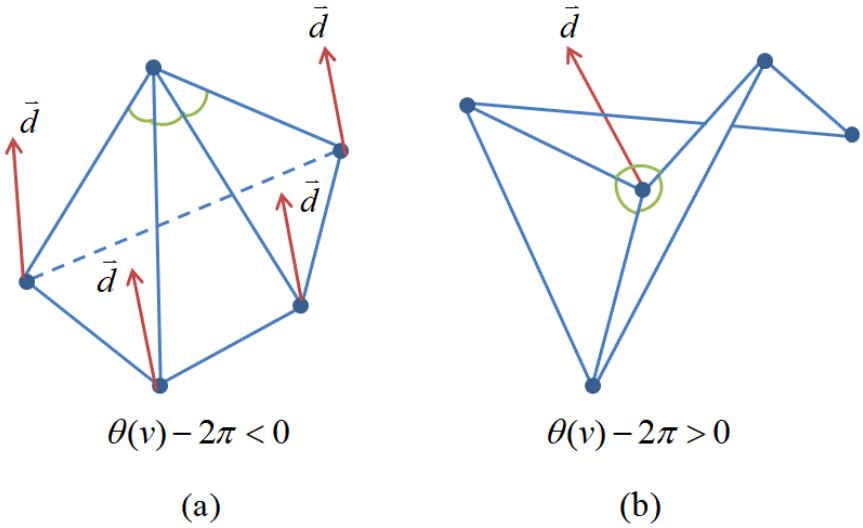

Figure 4-7 Node movements in different conditions

Based on the local vertices selection strategy, nodes with the high contribution of improving flattenability are moved in each step. The process of the local vertices

selection is shown in Algorithm 4-1 as follows.

1) $\omega(v_p)$ is calculated for each node in the original surface. Nodes with the top 30% large values of $\omega(v_p)$ are selected as reference nodes.

2) Reference nodes with $\theta(v)-2\pi>0$ are selected as pre-optimized nodes.

3) Reference nodes with $\theta(v)-2\pi<0$ are left in the reference nodes set, and one-ring neighbor nodes of them are selected as candidate nodes. If nodes in the candidate nodes set are not in the reference nodes set, they are selected as pre-optimized nodes to move in each step of the iteration.

4) After several iterations, some nodes have been moved to the positive direction of the z-axis. When updating initial input nodes by the local node selection strategy, reference nodes with $\theta(v)-2\pi<0$ are left in the reference nodes set. If they have been moved to the positive direction of the z-axis in the iteration process, they are selected as pre-optimized nodes and can be moved to the negative direction of the z-axis with a distance that shorter than those that have been moved to the positive direction of the z-axis in the previous iteration process.

Algorithm 4-1: local node selection strategy

for i =1 **to** number of vertices
    $\omega(v_i)=|\theta(v_i)-2\pi|$
end for
Top $\leftarrow$ Top 30% large values of $\omega(v_i)$
for i =1 **to** number of vertices
    if $\omega(v_i)>$ Top **then**
        if $\theta(v_i)-2\pi>0$ **then**
            v_i selected as pre-optimized nodes
        else
            v_i is left in reference nodes set
            v_i is selected as pre-optimized nodes and moved to the negative direction of the z-axis
            one-ring neighbor nodes of v_i are selected as candidate nodes
        end if

14: **end if**
14: **end for**
15: **for** i =1 **to** number of vertices in candidate nodes set
16: **if** candidate nodes are not in the reference nodes set **then**
17: v_i selected as pre-optimized nodes
18: **end if**
19: **end for**
20: **return** pre-optimized nodes

The L-PSO process is shown in Algorithm 4-2 to improve flattenability of a surface. Main processes of the proposed optimization are as follows.

Initial pre-optimized nodes are selected by the local node selection strategy as inputs of the PSO method. Initial inputs also include the initial population NP, initial position $\boldsymbol{X}$, initial velocity $\boldsymbol{V}$, and initial iteration number T for the PSO process. After completing iteration number T, gbest is the best position in the current iteration process. If f(gbest) is smaller than f(Gbest$^{k\text{-}1}$), where Gbest$^{k\text{-}1}$ is the best global position of the k-1 time of choosing pre-optimized nodes, Gbest^{k}=gbest; else, the iteration number will be increased to T+T. If f(Gbest^{k}) – f(Gbest$^{k\text{-}1}$) <ε, the optimization process stops and Gbest^{k} is the finally improved surface, else, initial pre-optimized nodes are updated based on the current surface using the local nodes selection strategy.

Algorithm 4-2: L-PSO

1: $k \leftarrow 1$
2: $t \leftarrow 1$
3: $i \leftarrow 1$
4: **do**
5: **input** Pre-optimized nodes, Initial population NP, Initial position $\boldsymbol{X}$, Initial velocity $\boldsymbol{V}$, and Initial iteration number T
6: **do**
7: **while** $t \leq$ T (Initial iteration number) **do**
8: **while** $i \leq$ NP (initial population) **do**
9: $\varpi_i^j(t) \leftarrow w\varpi_i^j(t-1) + c_1 r_1(\text{pbest}_i^j(t-1) - x_i^j(t-1)) + c_2 r_2(\text{gbest}^j(t-1) - x_i^j(t-1))$
10: $x_i^j(t) \leftarrow x_i^j(t-1) + \varpi_i^j(t)$
11: $i \leftarrow i$+1
12: **end while**
13: Update the best local position $\text{pbest}(t) \leftarrow \{\text{pbest}_1(t), \text{pbest}_2(t), \cdots \text{pbest}_{\text{NP}}(t)\}$

14: Update the best global position $\text{gbest}(t) \leftarrow \min\{\text{pbest}_1(t), \text{pbest}_2(t), \cdots \text{pbest}_{\text{NP}}(t)\}$
15: $t \leftarrow t+1$
16: **end while**
17: **if** f(gbest)< f(Gbest^{k-1}) **then**
18: Gbest$^k \leftarrow$ gbest
19: **else**
20: T=T+T (increase the iteration number)
21: **end if**
22: **until** f(gbest)< f(Gbest^{k-1})
23: **end do**
24: **until** f(Gbest^k) – f(Gbest^{k-1}) <ε
25: **end do**
26: **output** Gbest^k

4.4 Optimization results

4.4.1 A simple free-form surface example

The free-form surface contains 89 nodes, 247 edges and 159 faces. Its initial 3D model and 3D mesh model are shown in *Figure 4-8* (a) and (b). FLV of the original surface calculated by Eq. (4-3) is 0.6548. The largest value of $\omega(v_p)$ among all the vertices is 0.2899. *Figure 4-8* (c) shows a color graph of flattenability of the surface where blue and yellow areas show the vertex having the large value of $\omega(v_p)$.

The initial population NP and initial iteration number T are set as 100 and 10,000, respectively. According to Algorithm 4-2, the iteration number can be increased based on the solution search in the iteration process. Thus, the value of initial iteration number T will not affect the accuracy and efficiency of the optimization process. The total iteration number is 30,000 to reach the improved surface. Data of the original surface, intermediate surface generated with iteration number 10,000, and improved surface with iteration number 30,000 are listed in *Table 4-3*.

The surface flattenability is improved to 5.0261 by the optimization. The improved 3D mesh model is shown in *Figure 4-8* (g) (h), and (i). Values of $\omega(v_p)$ for most triangles are near to zero. The maximum air gap is 3.6055 mm. After the optimization, the surface flattenability is improved from 0.6548 to 5.0261. The

surface smoothness is decreased from 26.880 to 26.822. The area of the surface is increased from 2871.8 mm^2 to 2889.2 mm^2. The optimization improves the flattenability significantly, but it only changes the surface smooth and surface area a bit. *Figure 4-8* (j) shows the frequency distribution histogram of air gaps in the improved surface where most of the nodes have air gaps between 0 and 0.4 mm. *Figure 4-8* (k) shows the position of air gaps where red areas show vertexes with large air gaps.

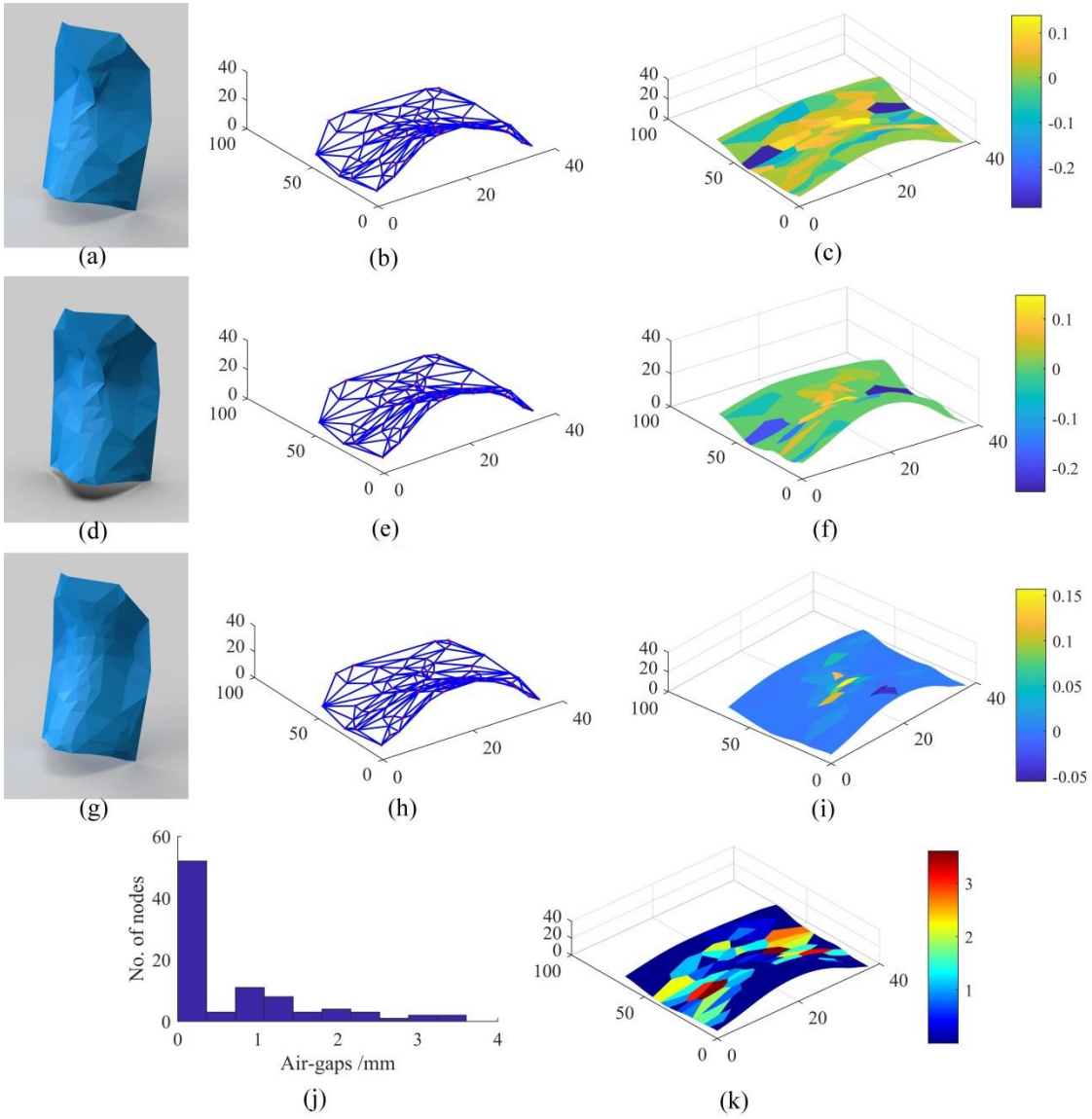

Figure 4-8 Original surface and improved surface of the simple free-form surface

Table 4-3 Results of original free-form surface and improved free-form surface

	Number of nodes	Number of edges	Number of faces	Number of iterations	FLV	Maximum value of $\omega(v_p)$	Smooth	Surface area (mm^2)	Air gaps (mm)
Original surface	89	247	159	0	0.6548	0.2899	26.880	2871.8	0
Intermediate surface	89	247	159	10000	1.8422	0.2478	26.944	2942.8	3.7450
Improved surface	89	247	159	30000	5.0261	0.1571	26.822	2889.2	3.6055

4.4.2 A nose model example

The nose model contains 548 nodes, 1575 edges and 1028 faces. The original 3D model and triangular mesh model are shown in *Figure 4-9* (a) and (b) with FLV 0.1212. The largest value of $\omega(v_p)$ among all the vertices is 0.3151. *Figure 4-9* (c) shows the color graph of $\omega(v_p)$ in each vertex, blue and yellow areas show vertexes with the large value of $\omega(v_p)$. The tip and bridge of the nose model have large values of $\omega(v_p)$.

The initial population NP and initial iteration number T are set as 100 and 10,000, respectively. The total iteration number is 70,000 to reach the improved surface. Data of the original surface, improved surface and two intermediate surfaces generated by iteration numbers 20,000 and 40,000 are listed in *Table 4-4*. After the optimization, FLV of the nose surface is improved to 0.6487 as shown in *Figure 4-9* (j), (k) and (i). The maximum air gap is 0.3295 mm. *Figure 4-9* (m) shows the frequency distribution histogram of air gaps in the nose model, where most of the nodes have air gaps between 0 and 0.1 mm. *Figure 4-9* (m) shows the position of air gaps, red areas show the vertex with the large air gap.

Although the air gaps are smaller than the acceptable maximum value (1mm), the surface flattenability has not met the requirement. Based on the solution of previous research, the segmentation can increase the surface flattenability. Thus, the segmenting surface is applied to divide the nose model into 3 pieces. The cutting line

is formed by finding positions with large values of $\omega(v_p)$ as shown in *Figure 4-10* (a). The segmented results are shown in *Figure 4-10* (b), (c), and (d). The number of nodes, edges and triangles, and flattenabilities of three parts are listed in *Table 4-4*.

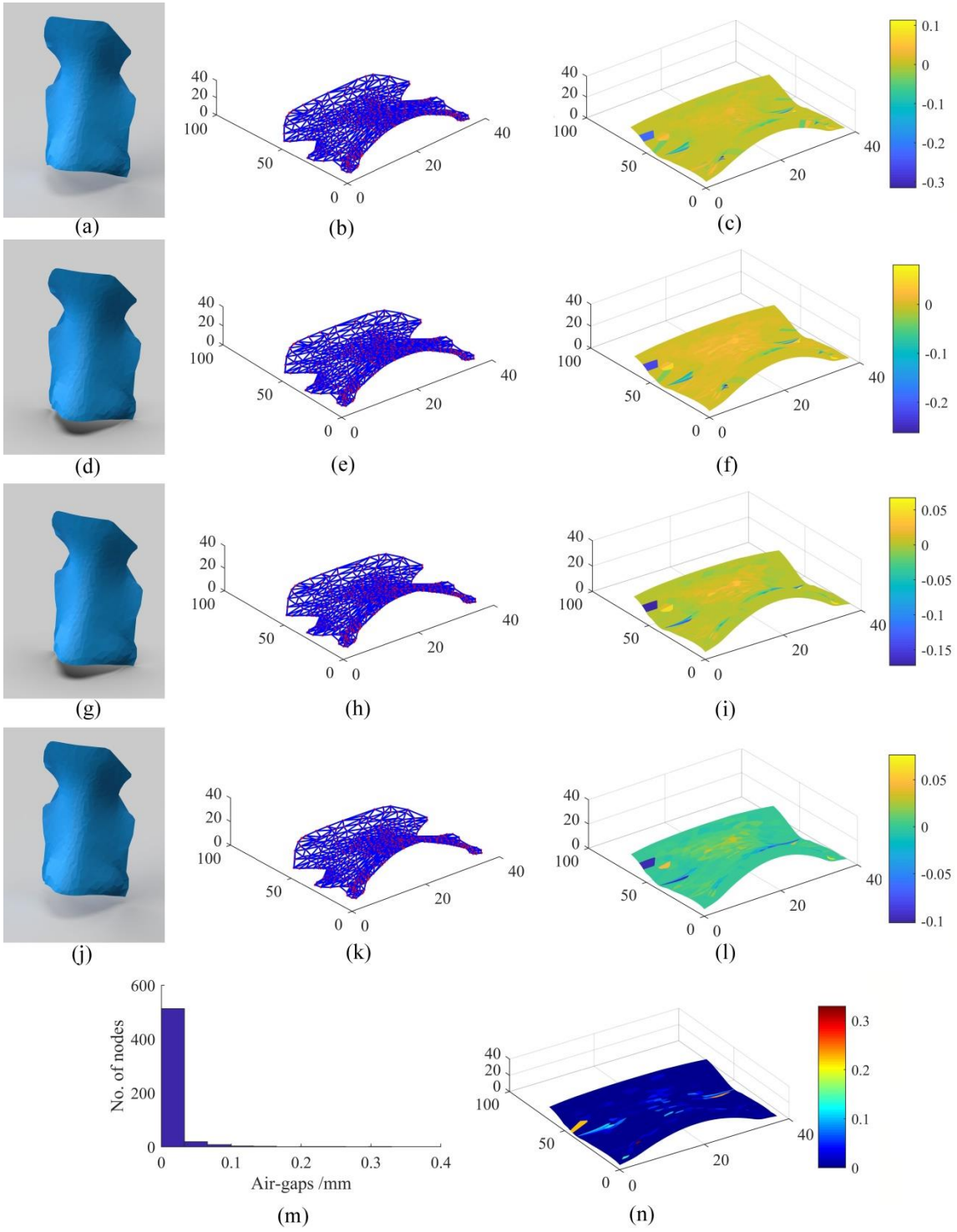

Figure 4-9 Original surface and improved surface of the nose model

An improved surface is formed using the same initial population and iteration number. The total iteration number is 50,000. Data of the original surface, two intermediate surfaces generated with iteration number 10,000 and 30,000, and improved surface with iteration number 50,000 are listed in *Table 4-4*. After the optimization, FLV of three pieces are improved to 5.2935, 5.1577 and 3.4570, respectively.

Table 4-4 Results of the original nose model and improved nose model

		Number of nodes	Number of edges	Number of faces	Number of iterations	FLV	Maximum value of $\omega(V_p)$	Smooth	Surface area (mm^2)	Air-gaps (mm)
Original surface	Whole surface	548	1575	1028	0	0.1212	0.3151	15.368	2692.8	0
	Patch1	198	525	328	0	4.0481	0.2234	11.013	807.41	0
	Patch2	274	704	431	0	2.6449	0.1526	8.829	889.27	0
	Patch3	264	716	453	0	1.1318	0.3151	8.056	996.13	0
Intermediate surface 1	Whole surface	548	1575	1028	20000	0.2512	0.2624	15.404	2694.4	0.3251
	Patch1	198	525	328	10000	4.7159	0.2156	11.085	808.57	0.6809
	Patch2	274	704	431	10000	3.4073	0.1388	11.141	889.88	0.5058
	Patch3	264	716	453	10000	2.0001	0.2418	9.326	997.27	0.4973
Intermediate surface 2	Whole surface	548	1575	1028	40000	0.5030	0.1725	15.413	2692.9	0.3299
	Patch1	198	525	328	30000	5.0362	0.1914	11.203	809.35	0.7036
	Patch2	274	704	431	30000	4.4934	0.0910	11.162	889.07	0.5749
	Patch3	264	716	453	30000	3.3017	0.0772	9.346	996.89	0.4957
Improved surface	Whole surface	548	1575	1028	70000	0.6487	0.1022	15.451	2690.6	0.3295
	Patch1	198	525	328	50000	5.2935	0.1400	11.110	808.71	0.7036
	Patch2	274	704	431	50000	5.1577	0.0464	11.150	888.50	0.5638
	Patch3	264	716	453	50000	3.4570	0.0757	9.359	996.74	0.4954

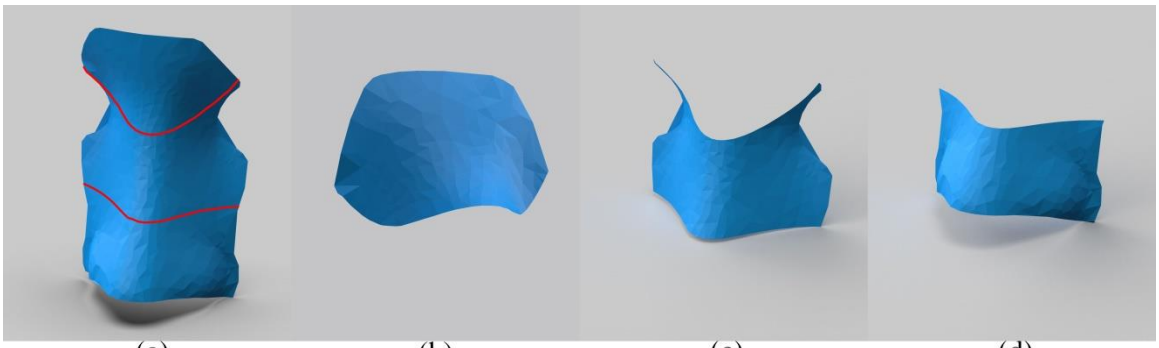

Figure 4-10 Surface segmentation process

The improved surfaces are shown in *Figure 4-11* (j), *Figure 4-12* (j) and *Figure 4-13* (j), respectively. The maximum air gaps of three parts are 0.7036 mm, 0.5638 mm, and 0.4954 mm, respectively. *Figure 4-11* (m), *Figure 4-12* (m) and *Figure 4-13* (m) show the frequency distribution histogram of air gaps in each part, respectively. Most of the nodes have air gaps between 0 and 0.1 mm in three parts. In addition, *Figure 4-11* (n), *Figure 4-12* (n) and *Figure 4-13* (n) show the position of air gaps, red areas show vertices with large air gaps.

The flattenability is closely related to the number of vertexes. If a surface has many vertexes, it will be difficult to improve its flattenability. In addition, the constraint of nodes in the original surface moved in the positive direction of normal vectors of an outer surface also restricts the further improvement of the flattenability.

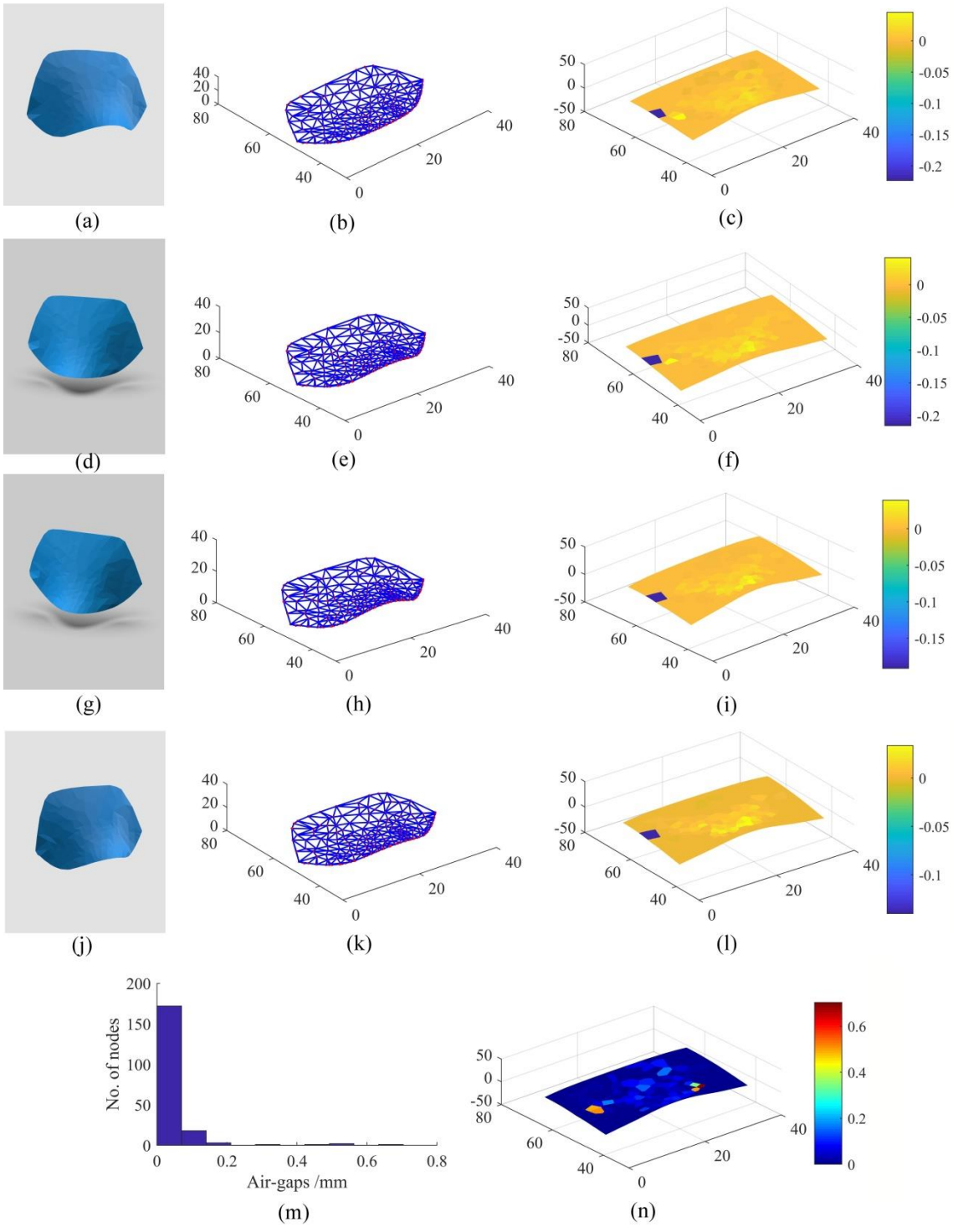

Figure 4-11 Original surface and improved surface of segmentation part 1 of the nose model

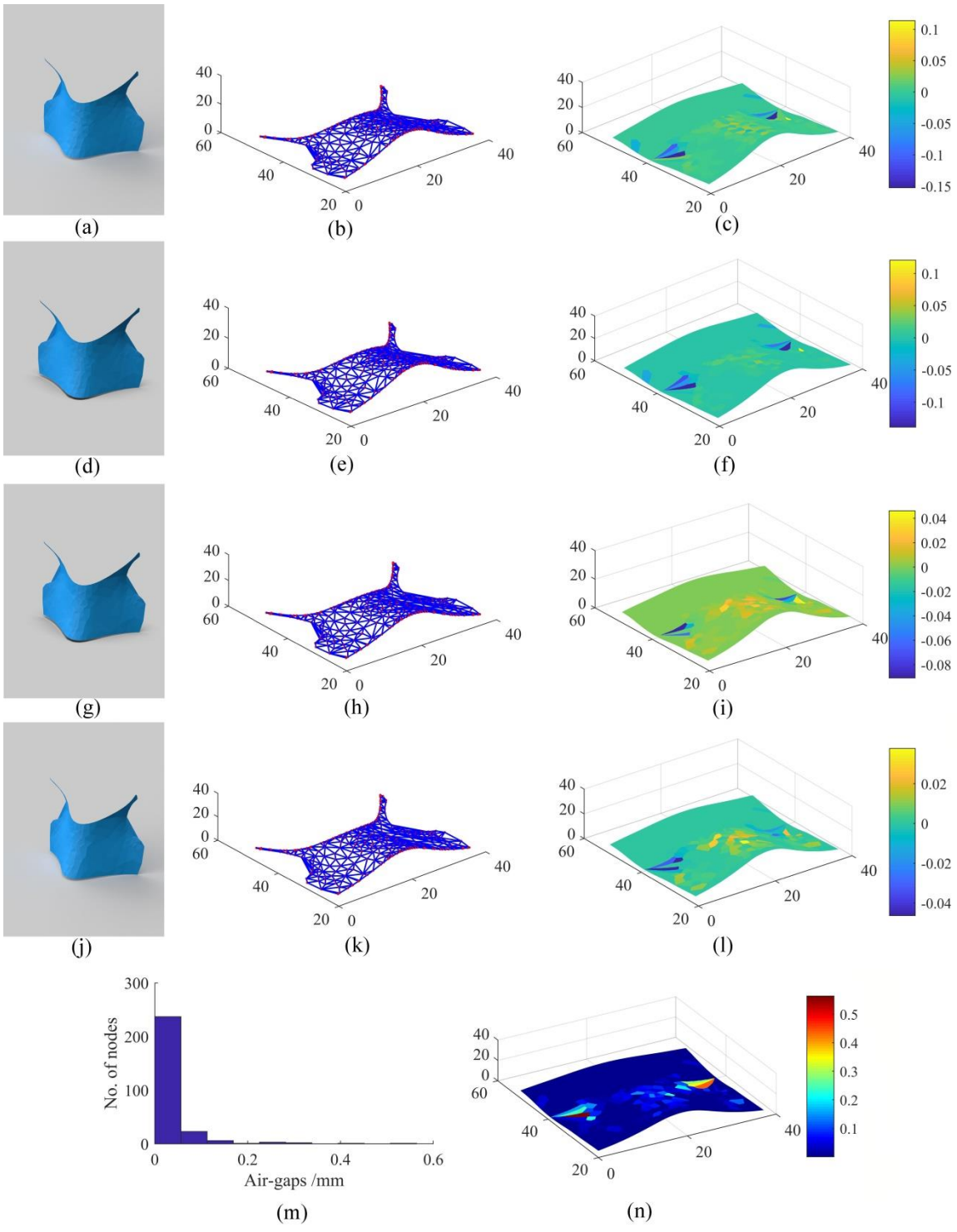

Figure 4-12 Original surface and improved surface of segmentation part 2 of the nose model

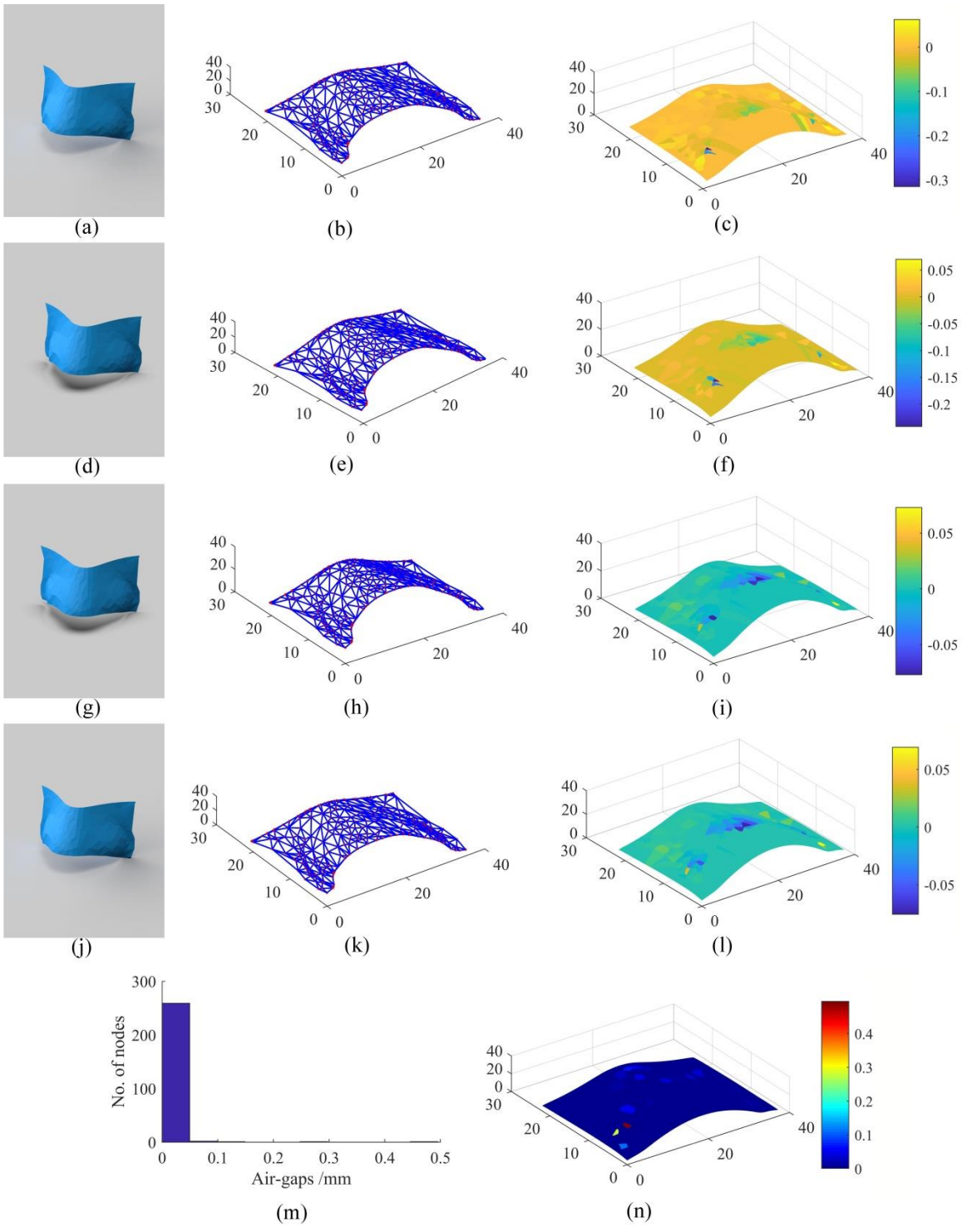

Figure 4-13 Original and improved surfaces of segmented part 3 of the nose model

4.5 Surface unfolding

4.5.1 Coordinate transformation

The coordinate transformation is required in the process of surface unfolding from a 3D model to its 2D surfaces. A triangle mesh in the 3D model is first chosen as the

central triangle. The plane located the central triangle is chosen as the fixed 2D plane and other triangles in the 3D shape are transformed to the 2D plane layer by layer from triangles near the central triangle to triangles far away from the central triangle. The central triangle is located in the middle of the 3D shape to reduce the number of iterations.

In the unfolding process, a central triangle is first selected to unfold it without deformation. Labels of boundary vertexes are defined as 0, and labels of other vertexes are defined as Inf at first. Vertexes with label Inf that directly connect to boundary vertexes are searched and changed their labels from Inf to 1. Vertexes with label Inf directly connected to vertexes with label 1 are searched and changed their labels to 2, until there are no vertexes with label Inf. Labels of each triangle are defined as the summation of labels for vertexes. The triangle with the largest total label value is chosen as a central triangle. The central triangle is unfolded into a 2D surface without deformation. *Figure 4-14* (a) shows the triangle labeling process.

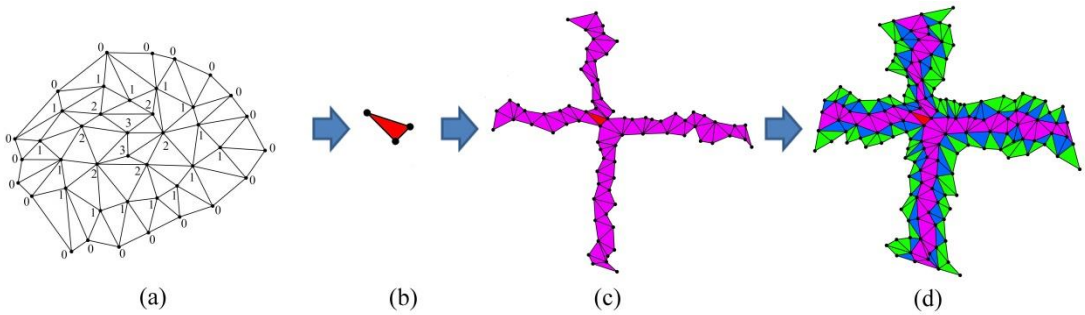

Figure 4-14 Triangle labeling and schematic diagram of triangle crossings

To reduce the number of iterations in the coordinate transformation, a triangle crossing is proposed to increase the number of triangles unfolded in each unfolding step. *Figure 4-14* (b)-(d) show the sequence of triangle crossings, where the pink crossing is the central triangle crossing and other triangles are adjacent triangle crossings.

Thus, after unfolding the central triangle without deformation, a central triangle crossing is chosen and unfolded. The weight center of the central triangle is calculated

and several unfolding directions are defined through the weight center of the central triangle. This process starts at the unfolded central triangle to search whether each edge of unfolded triangle intersects with each unfolding direction. If an edge of an unfolded triangle intersects with the unfolding direction, the neighbor face of the edge will be unfolded. After unfolding triangles pass through each unfolding direction, the total number of triangles in each unfolding direction is calculated. The unfolding direction with most triangles is chosen as a main stripe. The main stripe and its vertical direction are composed as the central triangle crossing as shown in *Figure 4-14* (c). The central triangle crossing is unfolded with no deformation.

If two or more triangles appear as the same value of the label, all the triangles will be unfolded without deformation. All the triangles will be tested to find central triangle crossing. The triangle with the most triangles in the main stripe will be selected as the central triangle.

To unfold adjacent triangle crossings layer by layer, targeted triangles unfolded in one iterative process should be obtained first. The targeted triangles are 3D triangles having common vertexes with unfolded triangles. Before unfolding targeted triangles, inner angles of targeted triangles should be calculated first. To unfold 3D surface without deformation, the inner angle of a 2D plane should be equal to the corresponding inner angle of the 3D triangle mesh. However, the summed inner angle of an internal vertex in the 3D model may not equal to 2π (Eq. (4-22) and *Figure 4-15* (a)), while the summed inner angle of an internal vertex in the 2D unfolded surface should equal to 2π (Eq. (4-23) and *Figure 4-15* (b)). Thus, the inner angle of an internal vertex is calculated as follows.

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 \neq 2\pi \tag{4-22}$$

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 = 2\pi \tag{4-23}$$

If $\beta_1, \beta_2, \beta_3$ have been unfolded, $\beta_4, \beta_5, \beta_6$ are calculated as follows.

$$\beta_i = \frac{\alpha_i}{\alpha_4+\alpha_5+\alpha_6} \cdot [2\pi - (\beta_1 + \beta_2 + \beta_3)] \quad i = 4,5,6 \tag{4-24}$$

There are two kinds of targeted triangles in adjacent triangle crossings. One is triangles with one unfolded edge that belongs to boundaries of unfolded triangles (The

blue triangles in *Figure 4-14* (d)). The other is triangles only with one unfolded vertex that belongs to boundaries of unfolded triangles (The green triangles in *Figure 4-14* (d)).

To decide the position of unfolded vertexes in targeted triangles, for triangles with one unfolded edge that belongs to boundaries of unfolded triangles, targeted triangles are blue colored triangles in *Figure 4-16* (a), where *P* is the vertex to be unfolded, β_1, β_2 are calculated using Eq. (4-24). l_1 is the length of an unfolded edge of a 2D plane. Using two angles and one edge of a triangle can uniquely determine a shape of the triangle. After unfolding, vertex *P* has two possible positions as shown in *Figure 4-16* (a), *P* and *P*'. To uniquely determine the position of vertex *P*, the triangle with the center far away from the center of its adjacent unfolded triangle through the unfolded edge (the orange colored triangle in *Figure 4-16* (a)) is chosen as the right triangle position. In *Figure 4-16* (a), *P* is in the right position. In *Figure 4-16* (b), *Q* is in the right position.

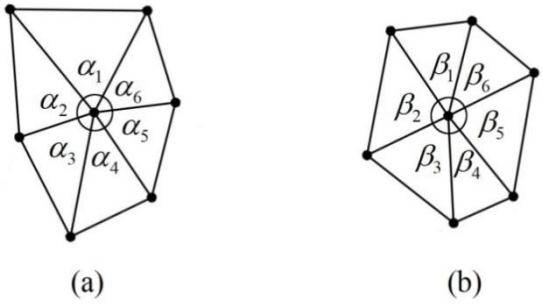

Figure 4-15 Inner angle of 3D model and 2D patches

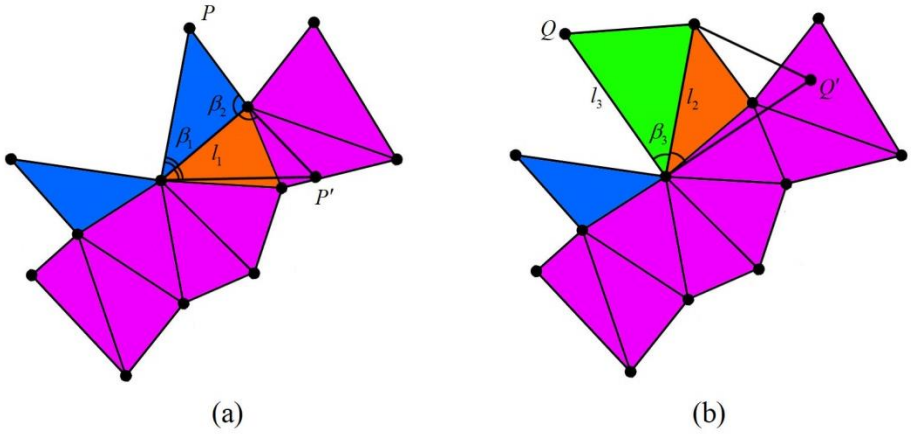

Figure 4-16 Two categories of adjacent triangles

4.5.2 Disturbing spring

Shapes and contours of 2D surfaces have a significant impact on the accuracy of their folding back to the 3D shape. In *Figure 4-17*, v_1 is a boundary node and v_2 is an inner node. It shows that the virtual force of the mass-spring model with a crossed spring in the boundary node is unbalanced compared to inner nodes. Thus, if an unfolded surface has large deformation in the shape and contour, it is difficult to get an optimal result. A disturbing spring is added in some important parts of the 2D surface such as the length, width and contour to improve optimization results.

In *Figure 4-17*, the red line shows the disturbing spring for controlling shape and blue line shows the disturbing spring for controlling contours. Two boundary vertexes v_{b1} and v_{b2} are linked by a disturbing spring. If v_{b1} is fixed, the movement of node v_{b2} is restricted by spring force $f(v_{b1}, v_{b2})$ from springs and calculated by Eq. (4-25).

$$\vec{f}(v_{b2}) = c(\sum |v_{bi}v_{bj}| - \sum D_{bibj})\vec{n}_{v_{b1}v_{b2}} \tag{4-25}$$

where *c* is the stiffness coefficient. $\sum |v_{bi}v_{bj}|$ is the sum length between unfolded nodes v_{b1} and v_{b2} in 2D patches. $\sum D_{bibj}$ is the sum distance between original nodes v_{b1} and v_{b2} in a 3D surface. $\vec{n}_{v_{b1}v_{b2}}$ is the force direction. Similar to Chapter 3.5, Newton's second law is used to calculate the new position of node v_{b2}. With the disturbing spring, the deformation in 2D patches can be reduced to improve efficiency of the optimization.

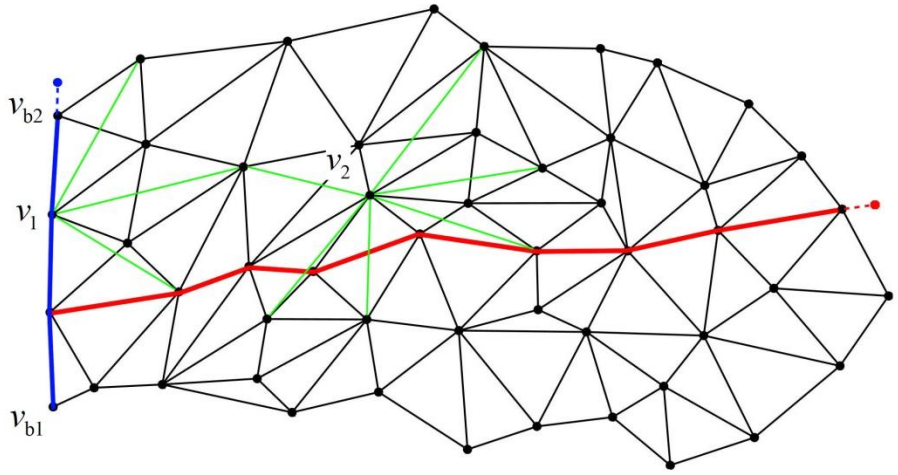

Figure 4-17 Disturbing spring

4.5.3 Overlapping correction

In the unfolding process, overlaps of 2D planes may be generated as the accumulation of deformations when more triangles are unfolded. Two methods are used to avoid overlapping. First one is a local algorithm for searching and optimizing overlaps. Second applies the energy release by the mass-spring model to reduce deformation of each triangle crossing in the unfolding process.

The first method uses a local algorithm to optimize the unfolding process to avoid overlaps. *Figure 4-18* shows examples of unfolded 2D planes with overlaps.

With the occurrence of overlaps, wrong positioned vertexes have two features. One is that the one-ring-neighbor edges of a wrong positioned node have a large error in the length accuracy, and the one-ring-neighbor faces of a wrong positioned node have a large error in the area accuracy. The other is that the one-ring-neighbor edges of a wrong positioned node intersect with other edges (Blue edges in *Figure 4-18*). Thus, nodes with overlaps are searched based on those two features.

To correct the wrong positioned node to the right position, one-ring-neighbor edges of a wrong positioned node intersected with other edges (green edges in *Figure 4-18*) are selected and assumed as springs. The direction of the spring force is along the edge with the direction of moving with wrong positioned vertexes closer to the edge (the arrow directions in *Figure 4-18*). After iteration, the vertex in the wrong

position will be corrected to the right position.

The second method prevents overlaps using the mass-spring model to optimize unfolded triangles after unfolding each adjacent triangle crossing to reduce deformation in the unfolding process, which can reduce the accumulation of deformations and possibility of overlaps.

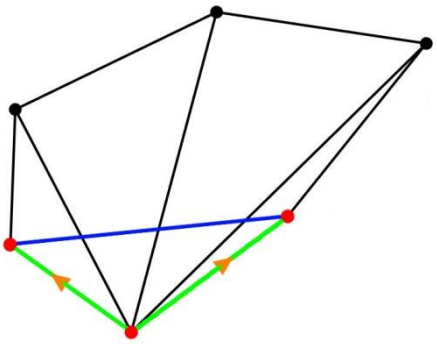

Figure 4-18 Unfolded 2D patches with overlaps

4.6 Results of surface unfolding

A nose model is segmented into 4 parts to reduce the value of $\sum \varpi$ in Chapter 4.4. 3D shapes and 2D patches of the nose model without segmentation are shown in *Figure 4-19* (a) and (b). 3D shapes and 2D patches of the model without a smooth boundary are shown in *Figure 4-19* (c) and (d). 3D shapes and 2D patches of the model with a smooth boundary are shown in *Figure 4-19* (e) and (f). Errors of the length and area of the models without segmentation, with and without the smooth boundary are shown in *Table 4-5*. Results show that surface segmentation can reduce errors of the length and area significantly. In addition, both length and area errors between two models for the model with a smooth boundary are small in an accepted level.

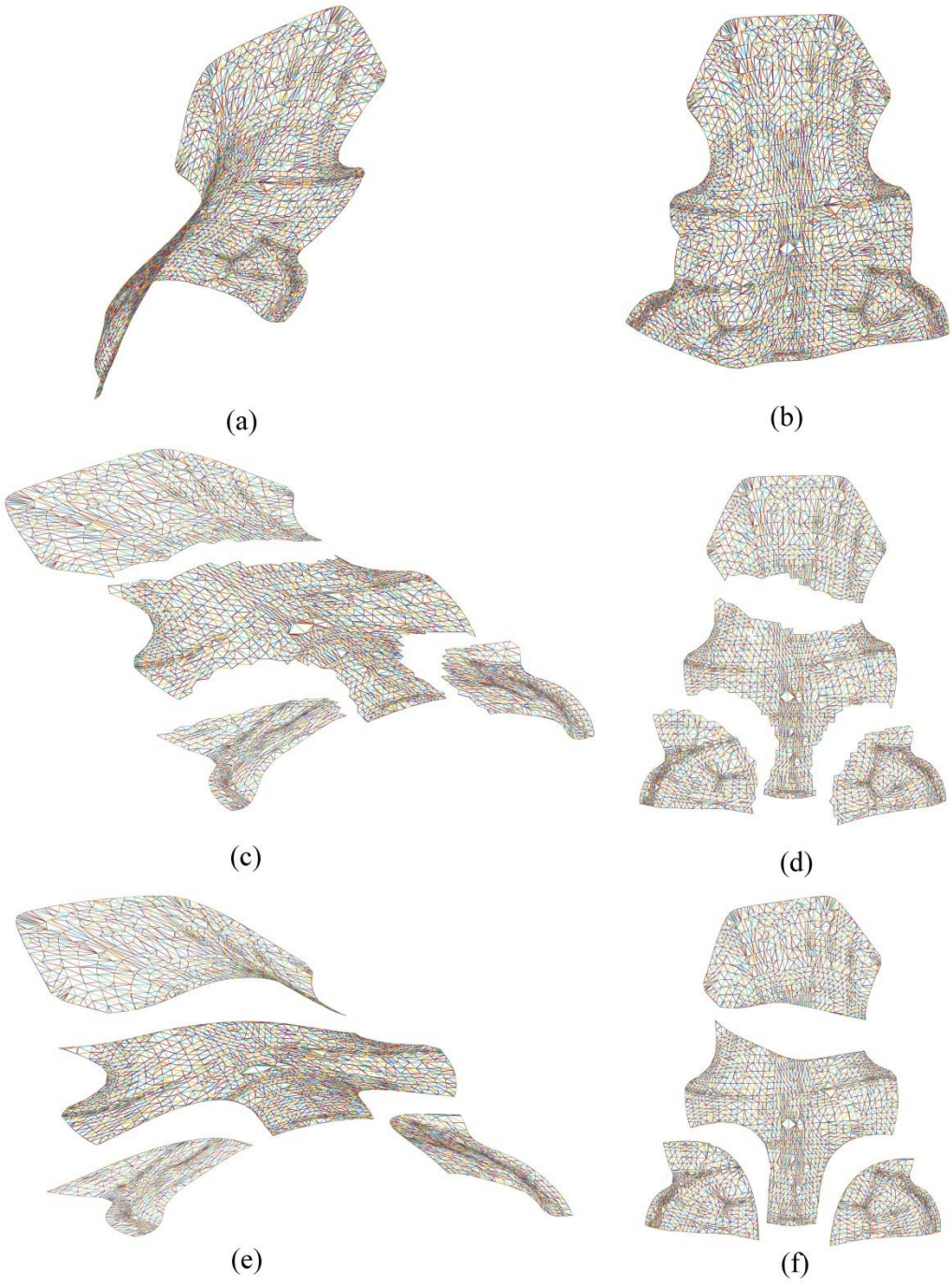

Figure 4-19 3D shapes and 2D patches of the nose model with and without segmentation and smooth boundary

Table 4-5 Errors of the nose model after segmentation with and without the smooth boundary

	Length error without smooth boundary	Area error without smooth boundary	Length error with smooth boundary	Area error with smooth boundary
Without segmentation	-	-	0.0758	0.1125
Part1	0.0093	0.0194	0.0092	0.0187
Part2	0.0227	0.0479	0.0232	0.0495
Part3(1)	0.0073	0.0164	0.0070	0.0162
Part3(2)	0.0075	0.0186	0.0071	0.0179

4.7 Summary

In this Chapter, a surface segmentation and unfolding process is developed based on the spectral clustering and mass-spring model. The result shows that surface segmentation can increase the surface flattenability significantly. A surface optimization method is proposed to improve flattenability and surface unfolding accuracy. Results verify the proposed method in accuracy and efficiency of surface unfolding. Errors of the length and area between original 3D model and 2D unfolded patterns are reduced significantly.

Chapter 5 Image-based model retrieval

A 3D model retrieval method is proposed based on feature skeletons of the model and model image. Mesh nodes in a 3D model are embedded into a feature space by the spectral analysis. Skeletons are formed from features of 3D models to build a skeleton base. Visual entropies are applied to detect edges of the model image. The edges are then segmented into the object and background pixels for contours of the object using a spectral clustering method. The skeleton of the image is compared with skeletons in the model skeleton base to find the best-matched 3D model using an iterative closest point method.

5.1 Contour detection of image

5.1.1 Edge detection

Visual entropies are human visual system (HVS) characteristics to simulate the human eye by comparing the intensity of an image with its background (Lai, 2011). The edge detection using HVS shows a good performance. Edges in an image are detected based on visual entropies for a set of pixels. A 2D image can be divided into small blocks for the edge detection as follows.

An image is partitioned into $k \times k$ blocks. For an image with 5×3 pixels shown in *Figure 5-1*, a window of the yellow color block with 2×2 pixels is used in the process. There are $(m-k+1) \times (n-k+1)$ blocks for a $m \times n$ sized image. Based on the block selection method shown in *Figure 5-1*, two neighbor blocks have some overlapped pixels in different blocks. Because of the overlapped pixels, one pixel is tested for several times in different blocks to decide the boundary pixel. This block selection method can reduce errors when using different sizes of blocks in the edge selection.

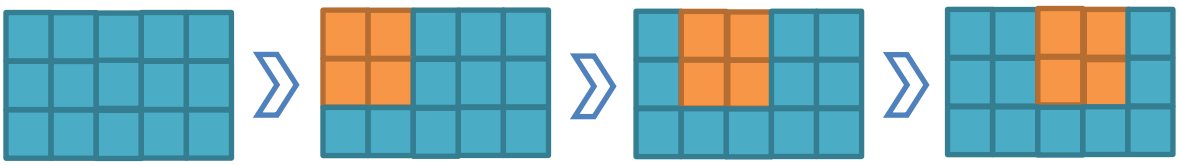

Figure 5-1 Block selection

Visual texture entropy and visual edge entropy are used to decide regions of edges in an image. The visual texture entropy is a common measure to describe textures of an image, which can be used to decide the spatial correlation of neighboring pixels (Lai, 2011) as follows.

$$E_{\text{texture entropy}} = -\sum_{i=1}^{n} p_i \log(p_i) \tag{5-1}$$

where, n is the number of image intensity and p_i represents the occurrence probability of an image intensity i with $0 \le p_i \le 1$ and $\sum_{i=1}^{n} p_i = 1$.

The image intensity of RGB color images is combined by three image intensities of R, G and B channels as follows.

$$\text{Intensity}_{RGB} = \text{Intensity}_R \times 0.299 + \text{Intensity}_G \times 0.587 + \text{Intensity}_B \times 0.114 \tag{5-2}$$

The visual texture entropy is based on the probability distribution of pixel intensities without the co-occurrence of pixel values. In an image, pixel intensities are not independent each other. The edge can be detected by comparing intensities of neighbor pixels. The visual edge entropy is defined by Eq. (5-3).

$$E_{\text{edge entropy}} = \sum_{i=1}^{n} p_i \exp^{1-p_i} \tag{5-3}$$

where, $1-p_i$ indicates the ignorance or uncertainty of the pixel value.

An edge detection process based on visual entropies is as follows:

1) The image is divided into $k \times k$ blocks.

2) The image intensity for each pixel of the original image is decided using Eq. (5-2).

3) The visual texture entropy and visual edge entropy of each block are decided using Eqs. (5-1) and (5-3), respectively.

4) The total entropy of each block is summed by Eq. (5-4).

$$E_{\text{block}} = \eta_1 \cdot E_{\text{texture entropy}} + \eta_2 \cdot E_{\text{edge entropy}} \tag{5-4}$$

where η_1 and η_2 are two weights.

5) The entropies of blocks are sorted into an ascending order based on their magnitude. The block with the large value of entropy is selected as the edge block.

5.1.2 Contour detection by spectral clustering

Contour detection extracts contour pixels of objects in a 2D image. A spectral clustering method is used to divide edge pixels for the object and background of the image.

Adjacency matrix *W* for building the Laplacian matrix defined in Eq. (3-3) is decided as follows.

$$W_{ij} = \begin{cases} \exp(-\frac{d_{ij}^2}{2\sigma^2}) & i \neq j \\ 0 & i = j \end{cases} \tag{5-5}$$

where d_{ij} measures the similarity between two edge pixels *i* and *j*, σ is the average of d_{ij}. d_{ij} in the adjacency matrix *W* is decided by Eq. (5-6) to combine the difference of RGB colors between two pixels and the distance between two pixels.

$$d_{ij} = p_{ij} + w \cdot c_{ij} \tag{5-6}$$

where p_{ij} is the distance between two edge pixels *i* and *j*. The distance between two adjacent pixels is 1. c_{ij} is the difference of RGB colors between two edge pixels *i* and *j* as follows.

$$c_{ij} = \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2} \tag{5-7}$$

where R_i and R_j are values of the R channel for pixels *i* and *j*, respectively. G_i and G_j are values of the G channel for pixels *i* and *j*, respectively. B_i and B_j are values of the B channel for pixels *i* and *j*, respectively.

After detecting edges of the object, the object boundary in the edges is selected as the object contour. For example, three images in *Figure 5-2* (a), (e) and (i) are images of an airplane from BSDS 500 database (Arbelaez et al, 2010), a glasses and a teddy, respectively. Edges are detected based on methods in Chapter 5.1.1 as shown in *Figure 5-2* (b), (f) and (j), respectively. Edge pixels are then clustered into edge pixels of the object and background as shown in *Figure 5-2* (c), (g) and (k), respectively. Finally, object contours of the images are decided as shown in *Figure 5-2* (d), (h) and

(l), respectively.

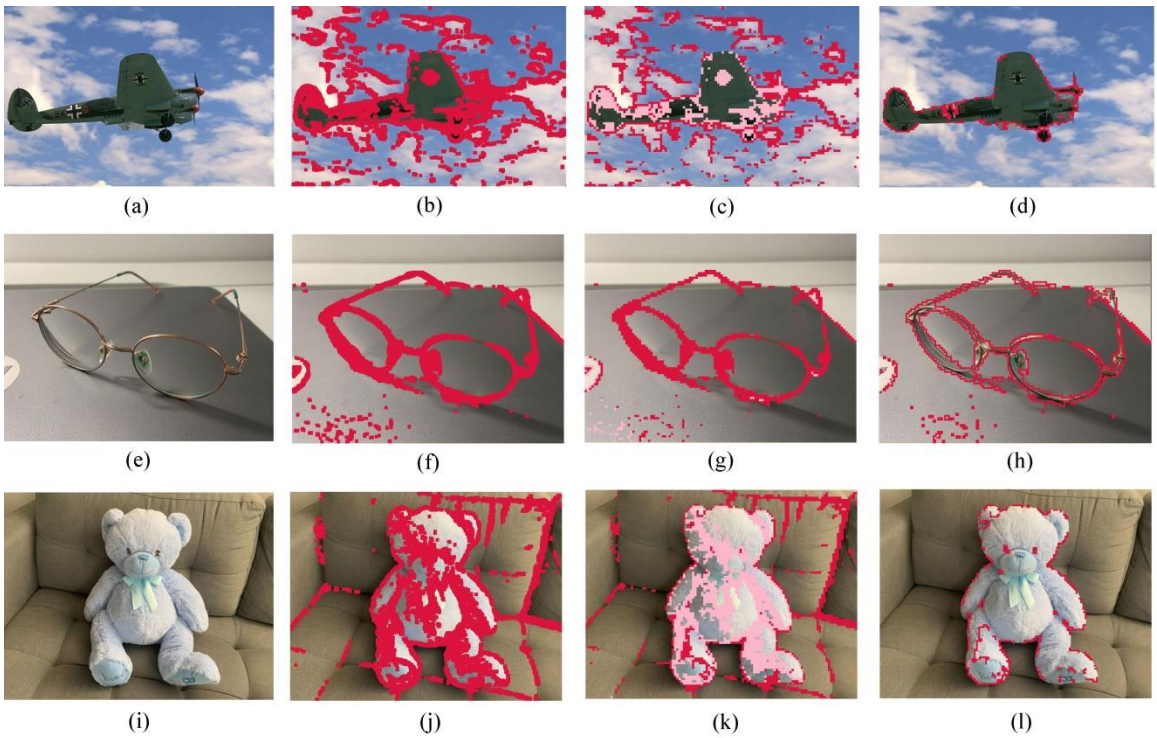

Figure 5-2 Contour detections of 2D images

5.2 Feature extraction and skeleton extraction from images and 3D models

5.2.1 Feature extraction from images and 3D models

The object contour of an image consists of a set of pixels. Each pixel in an image can be considered as a node in the 2D space. Contour pixels in the image can be embedded into a 3D feature space based on the spectral analysis. A 3D model consists of a set of nodes in the 3D space. Nodes of the 3D model can also be embedded into a feature space based on the spectral analysis. The feature space of contour pixels in an image and feature space of nodes in a 3D model are considered as features of the 2D image and features of 3D models, respectively. Thus, features of the image and 3D model are in the same dimension for the comparison each other.

In the spectral analysis, Laplacian matrix L is defined in Chapter 3.2 by Eq. (3-3). Adjacency matrix *W* for the feature extraction of an image is as follows.

$$W_{2Dij} = \begin{cases} \exp(-\frac{d_{2Dij}^2}{2\sigma_{2D}^2}) & i \neq j \\ 0 & i = j \end{cases} \quad (5\text{-}8)$$

where d_{2Dij} measures the distance of two contour pixels *i* and *j*, σ_{2D} is the average of d_{2Dij}.

Distance d_{2Dij} of contour pixels in adjacency matrix *W* of an image is decided using Eq. (5-9) as follows.

$$d_{2Dij} = \sqrt{(m_{ix} - m_{jx})^2 + (m_{iy} - m_{jy})^2} \quad (5\text{-}9)$$

where m_{ix} is coordinate *x* of pixel *i*, m_{jx} is coordinate *x* of pixel *j*. m_{iy} is coordinate *y* of pixel *i*, and m_{jy} is coordinate y of pixel *j*.

Adjacency matrix *W* for the feature extraction of 3D models is as follows.

$$W_{3Dij} = \begin{cases} \exp(-\frac{d_{3Dij}^2}{2\sigma_{3D}^2}) & \text{if } j \in U_{3D}(i) \\ 0 & \text{otherwise} \end{cases} \quad (5\text{-}10)$$

where d_{3Dij} measures the distance between node *i* and node *j* in a 3D model, σ_{3D} is the average of d_{3Dij}. $U_{3D}(i)$ is the set of one-ring neighbor nodes of node *i*.

Distance d_{3Dij} of nodes in adjacency matrix *W* of a 3D model is calculated using Eq. (5-11).

$$d_{3Dij} = \sqrt{(n_{ix} - n_{jx})^2 + (n_{iy} - n_{jy})^2 + (n_{iz} - n_{jz})^2} \quad (5\text{-}11)$$

where n_{ix} is coordinate *x* of node *i*, n_{jx} is coordinate *x* of node *j*. n_{iy} is coordinate *y* of node *i* and n_{jy} is coordinate *y* of node *j*. n_{iz} is coordinate *z* of node *i* and n_{jz} is coordinate *z* of node *j*.

A feature space is represented by matrix $U^{(k)} = (u_1, u_2, \cdots, u_k) \in R^{n \times k}$, composed of corresponding *k* eigenvectors of the first *k* minimum eigenvalues. Normally, *k*=3 for features of both 2D image and 3D model in a feature space.

5.2.2 Skeleton extraction from features of images and 3D models

Skeletons extracting from features of an image and 3D model can represent characteristics of the image and model. The extraction process is as follows.

1) Extraction of the center vertex and feature points by the clustering method

A feature point is the vertex located at the endpoint of the protruding area of a model, it can be used to compose skeletons of features. Feature points can fully represent characteristics of the most significant shape of a 3D model. To extract these feature points, Gaussian mixture model (GMM) is used to cluster features of the image and 3D model into several sets. Center vertex v_0 of the feature model is the vertex that is closest to the model centroid. After defining center vertex v_0 of the feature model, vertices are grouped into several clusters by the GMM clustering method. GMM clustering requires users to assign the number of clustering. The clustering process is a human-computer interactive process. The vertex on each cluster with the largest distance to center vertex v_0 is the feature point. For example, *Figure 5-3* (a) shows a teddy model and *Figure 5-3* (b) shows its feature model with 5 feature points.

2) Topological analysis and skeleton point extraction

Three topological structures are defined including branch, rendezvous, and end. All the points in the same branch of a model are merged into one vertex to form a skeleton point. Normally, the feature point selected in the last step can be used as the skeleton point in a branch. Most clusters only contain one branch. If a cluster contains more than one branch, the cluster will be divided into sets by GMM to further search skeleton points.

3) Skeleton connection

According to the topology information carried by skeleton points, the skeleton is formed by connecting adjacent skeleton points to replace a branch with a skeleton, such as the example shown in *Figure 5-3* (c).

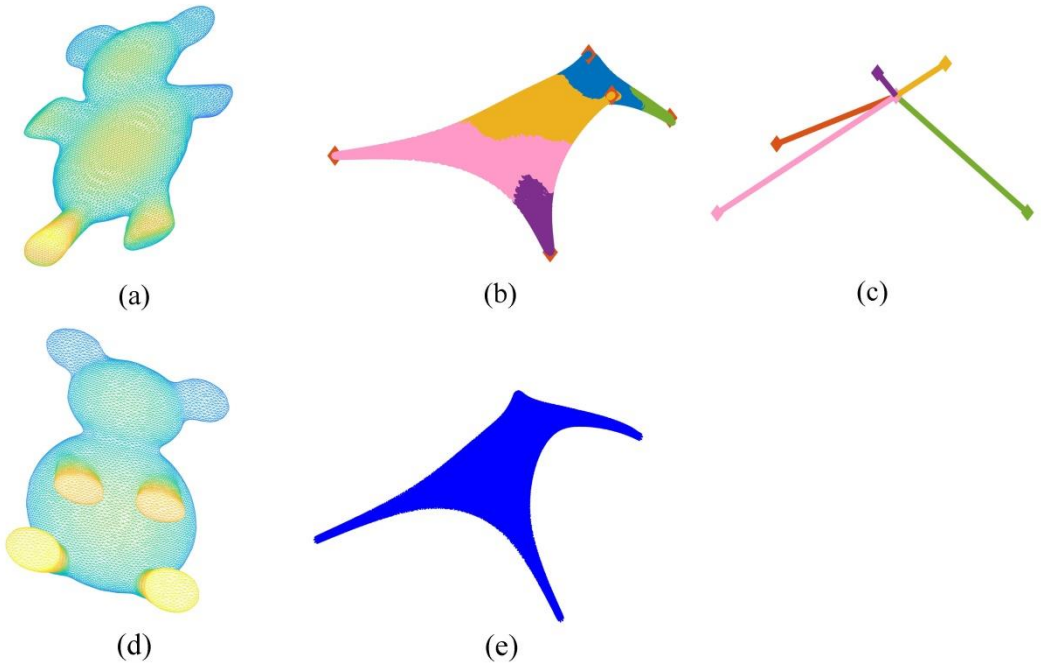

Figure 5-3 Feature space and skeletons of features for the bear model

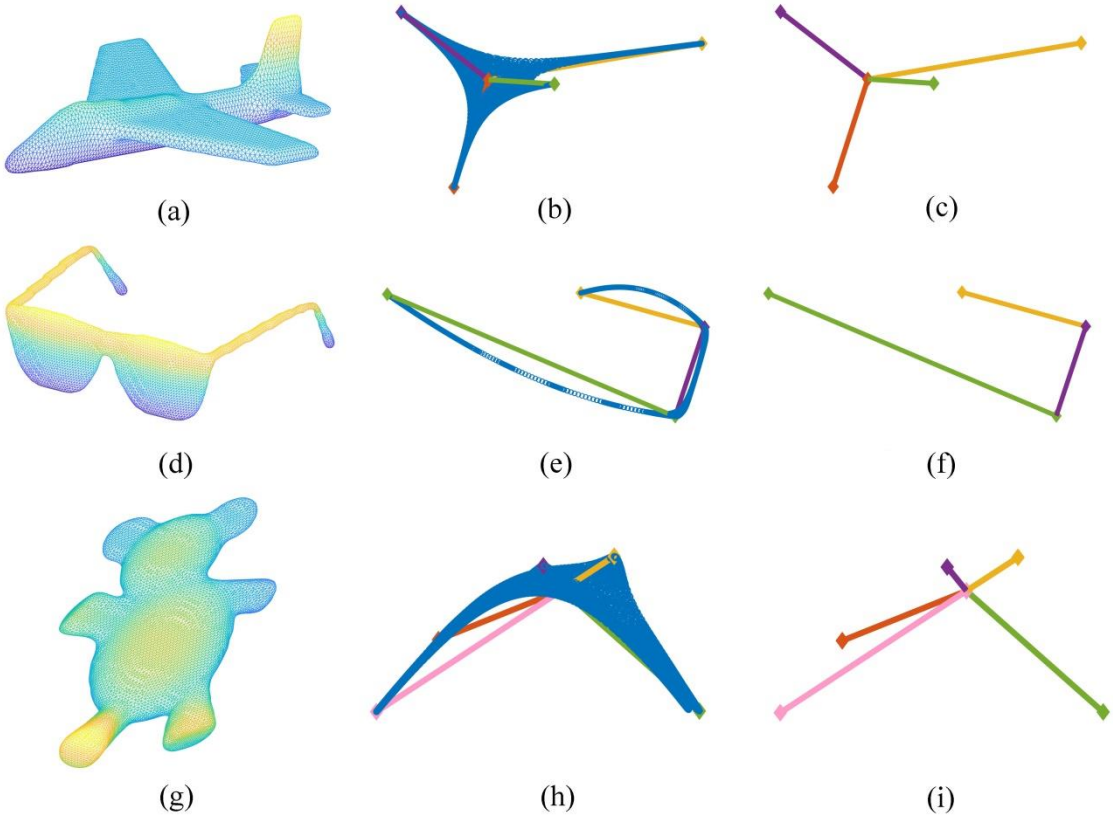

Figure 5-4 3D models, feature space and skeletons of features

Skeleton-based methods usually are sensitive to variations of model postures. But when embedding models with different postures into a feature space before the skeleton extraction, they have the similar feature space and skeleton. Robustness of the method is improved. For example, although teddy bears in *Figure 5-3* (a) and (d) have different postures, their features are similar as shown in *Figure 5-3* (b) and (e). Therefore, models are retrieved based on categories rather than different postures.

For example, skeletons of features of 3D models are extracted from 3D models of the airplane, glasses and teddy from Shape Retrieval Contest SHREC'07 (Giorgi et al, 2007) as shown in *Figure 5-4* (a) (d) and (g), respectively. The spectral analysis extracts features of all 3D models as shown in *Figure 5-4* (b) (e) and (h). Skeletons of features of 3D models are shown in *Figure 5-4* (c) (f) and (i).

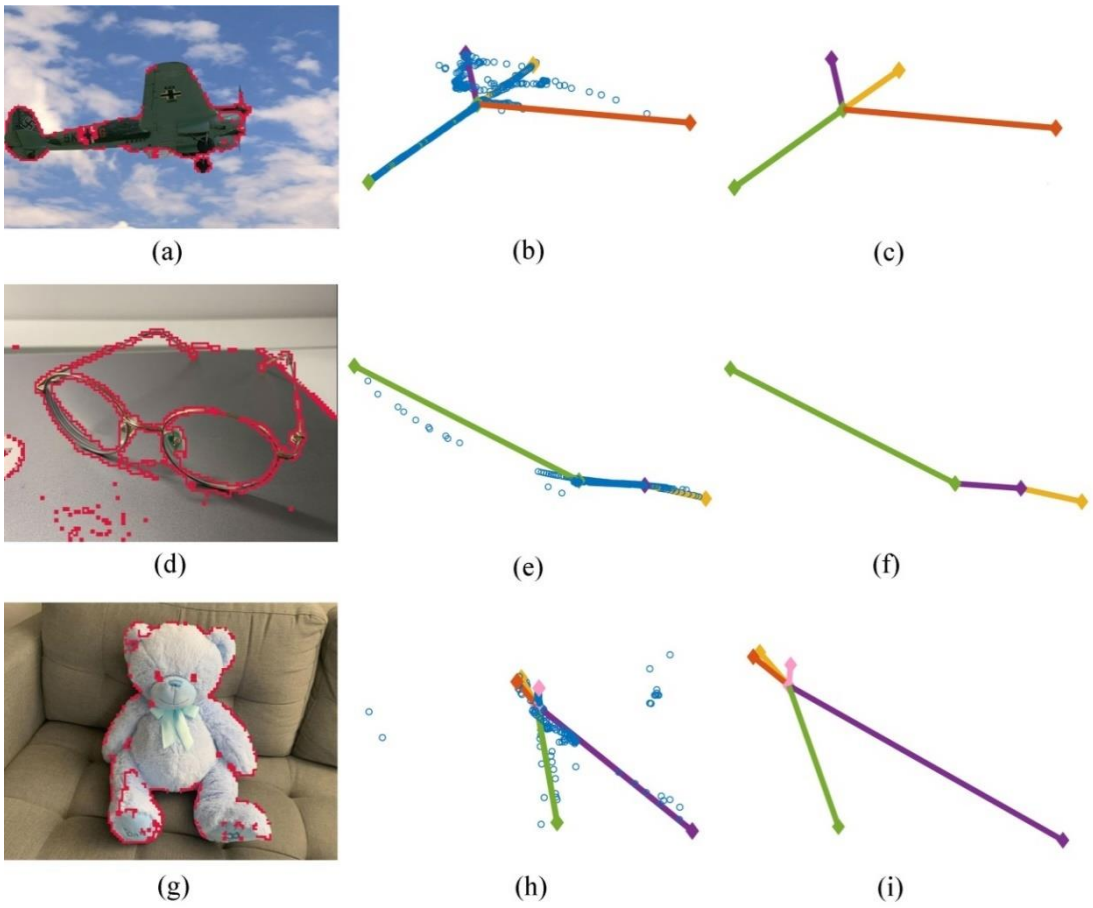

Figure 5-5 Contours of 2D image, feature space and skeletons of features

Skeletons are also extracted from their 2D images. Features of image contours

are extracted by the spectral analysis as shown in *Figure 5-5* (b), (e) and (h). Skeletons of features of images are formed in *Figure 5-5* (c) (f) and (i).

5.3 Model retrieval

An iterative closest point (ICP) method is proposed to retrieve a 3D model based on the comparison of skeletons of the image and 3D model. The ICP method searches a transformation F to transfer standard data set Y into the best alignment set $Z = RY + T$ with compared data set X (Ying et al, 2009). The iteration process pairs each point in X to the closest point in Y after the transformation. The next transformation is decided by minimizing the distance between paired points.

For example, there are two data sets $X = \{x_i\}_{i=1}^{m}$ (red color) and $Y = \{y_i\}_{i=1}^{n}$ (blue color) in *Figure 5-6*. Using rotation R^k and translation T^k, a subset $Z^k = \{z_i^k\}_{i=1}^{n}$ of X is searched for the same size Y to minimize the objective function in Eq. (5-12).

$$\varepsilon^k(Z^k) = \sum_{i=1}^{n} \left\| R^k y_i + T^k - z_i^k \right\|^2 \tag{5-12}$$

By fixing obtained Z^k, next rotation R^{k+1} and translation T^{k+1} are searched to minimize the objective function in Eq. (5-13) (Ying et al, 2009).

$$e^k(R^{k+1}, T^{k+1}) = \sum_{i=1}^{n} \left\| R^{k+1} y_i + T^{k+1} - z_i^k \right\|^2 \tag{5-13}$$

The ICP method only shows the relation of different points rather than coordinates of the points. The difference between data set $X = \{x_i\}_{i=1}^{m}$ and its closest data set Y after the transformation can be obtained as follows.

$$E = R_F Y + T_F - X \tag{5-14}$$

where R_F and T_F are final results of the rotation and translation, respectively.

In a retrieving process, the feature skeleton of a 2D image is compared with feature skeletons of 3D models in the skeleton base using the ICP method. The 3D model with the closest skeleton to the image is considered as the 3D model of the image. As the ICP method requires a similar size of the two data sets, some data points may be added on skeletons to meet the requirement. The error between

skeletons of the image and 3D model is decided by Eq. (5-14).

For example, skeletons of images in *Figure 5-5* are compared with skeletons of 3D models in *Figure 5-4* by the ICP method. The comparison results are shown in *Figure 5-6*. The model with the best matched skeleton to the image skeleton is considered as the retrieved model. The comparison data are listed in *Table 5-1*. Distances or dissimilarities between the skeleton of airplane image and skeletons of the three 3D models are 0.1499, 0.1529 and 0.1807, respectively. The best match is the airplane. For the glasses image, dissimilarities of its skeletons to skeletons of the 3D models are 0.9468, 0.7146 and 1.0210, respectively. The best match is the glasses model. For the teddy image, dissimilarities of its skeleton to skeletons of the 3D models are 0.0079, 0.0081 and 0.0071, respectively. The best match is the teddy model.

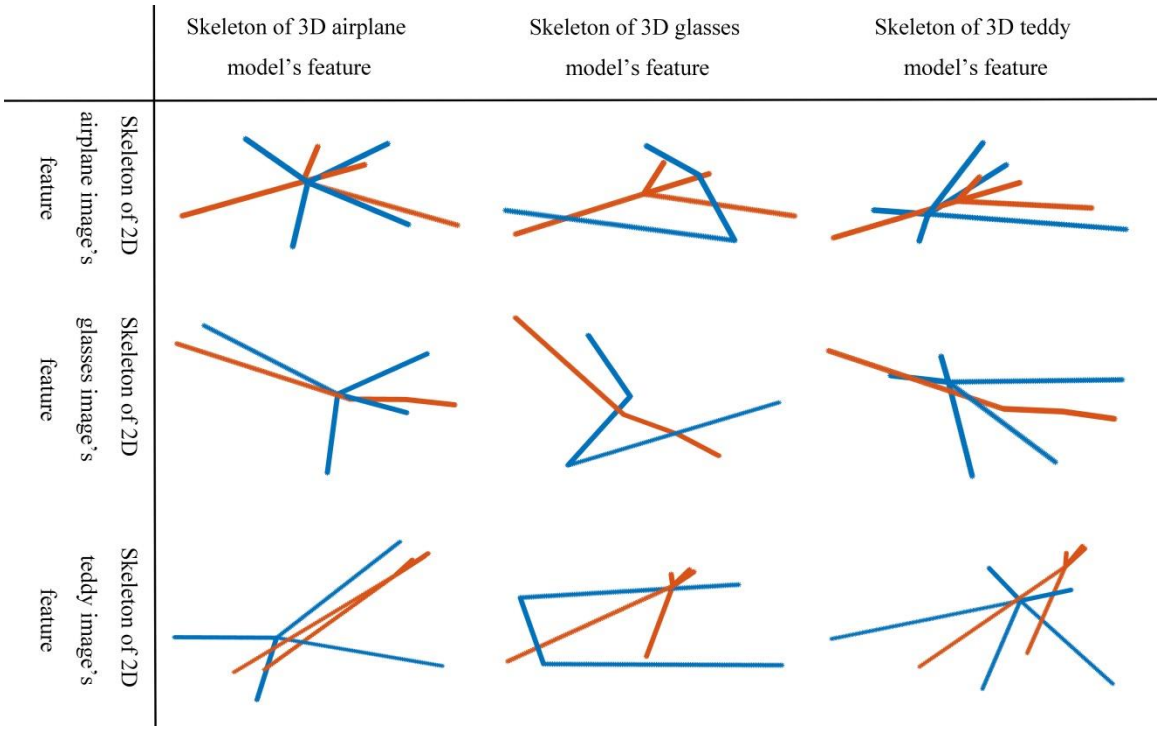

Figure 5-6 Comparison models

Table 5-1 Model retrieving based on the skeleton comparison

Input \ Skeleton base	Skeleton of feature of 3D airplane model	Skeleton of feature of 3D glasses model	Skeleton of feature of 3D teddy model
Skeleton of feature of 2D airplane image	**0.1499**	0.1529	0.1807
Skeleton of feature of 2D glasses image	0.9468	**0.7146**	1.0210
Skeleton of feature of 2D teddy image	0.0079	0.0081	**0.0071**

5.4 Retrieval results

Feature skeletons of 3D models are extracted to build a skeleton base. 2D images are used as input for model retrieving. The proposed method is implemented using MATLAB 2017 in a computer with Intel i7 CPU and 8 GB RAM.

To test the efficiency, CPU times of the skeleton extraction of a 3D model and image are recorded as shown in *Figure 5-7* (a) and (b), respectively. For 3D models, the CPU time closely depends on the number of vertices in the model. If the number of vertices is less than 3000, the CPU time is less than 50 seconds. In addition, the CPU time of the skeleton extraction from 2D image depends on the number of pixels in an image. For an image with 400×300 pixels, the CPU time is about 5 seconds. The CPU time of the comparison for two skeletons is about 0.33 second. Thus, for the high efficiency and acceptable accuracy, the size of the image should be less than 800×600 pixels.

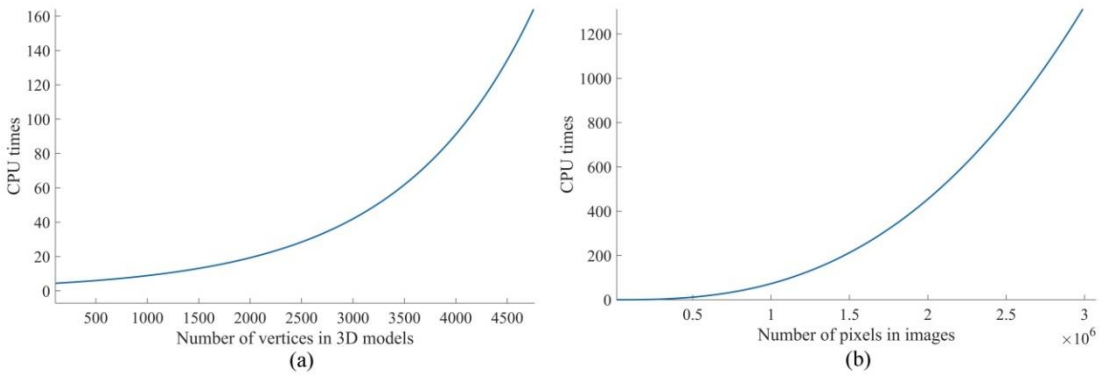

Figure 5-7 CPU times of the skeleton extraction of 3D models and images

To test the accuracy, several standard measures including average dynamic recall (ADR), mean average (MR) and last ranking (LR) indexes are used to measure the accuracy of the proposed method. The results are listed in *Table 5-2*.

Average Dynamic Recall (ADR) (Typke et al, 2006, July) is an average recall value defined as follows.

$$\mathrm{ADR} = \frac{1}{q}\sum_{i=1}^{q}\frac{\mathrm{RI}(i)}{i} \tag{5-15}$$

where q is the number of matches in the ground truth and $\mathrm{RI}(i)$ is the number of retrieved relevant items within the first *i* retrieved item.

The mean average ranking (MR) is computed by averaging the retrieval ranking of all relevant items when considering a particular query as follows.

$$\mathrm{MR} = \frac{1}{q}\sum_{i=1}^{q} c \tag{5-16}$$

where *c* is the position in the return vector of ordered items.

The last place ranking (LR) is defined as follows.

$$\mathrm{LR} = 1 - \frac{\rho - n}{N - n} \tag{5-17}$$

where ρ is the rank at which the last relevant object is found, *n* is the number of relevant items and *N* is the database size.

Table 5-2 Accuracy assessment

	ADR	MR	LR
Ideal	1	10.5	1
aMRG based method (Tung and Schmitt, 2008)	0.8577	30.31	0.74
Multivariate density-based method (Akgul et al, 2007)	0.7931	45.90	0.6
Kernels and extended Reeb graphs-based method (Barra and Biasotti, 2013).	0.7931	20.49	0.892
The proposed method	0.8589	14.1	0.9744

The proposed method is also applied in the human model surface retrieval to match the 3D model to 2D image. 3D models of the human body parts are divided from 3D human models. All the models in the model base are extracted into skeletons. To search the most similar 3D model in the model base, the skeleton of a nose image

is compared with all the skeletons in the model base to find the best matched skeleton. Images of two noses and one arm are shown in *Figure 5-8* (a), (e) and (i), respectively. Object boundaries of the images are shown in *Figure 5-8* (b), (f) and (j), respectively. Their contour features are extracted by the spectral analysis as shown in *Figure 5-8* (c), (g) and (k), respectively. Retrieved 3D models are shown in *Figure 5-8* (d), (h) and (l), respectively. Results of the retrieving process are listed in *Table 5-3*. Results show, for the nose 1 image, the best matched 3D model is a 3D nose model with a difference of 0.0106. For the nose 2 image, the best matched 3D model is also a 3D nose model with a difference of 0.0139. For the arm image, the best matched 3D model is a 3D arm model with a difference of 0.1374.

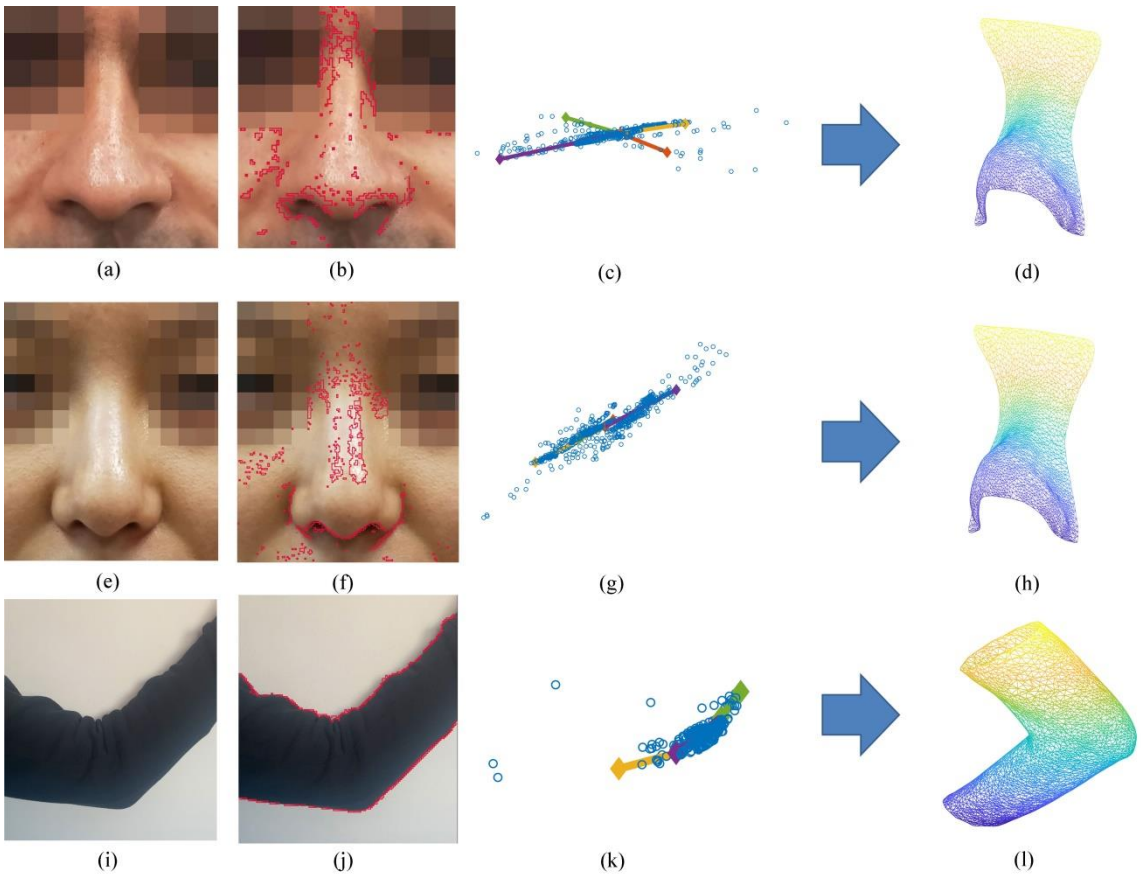

Figure 5-8 Model retrieving for human body surfaces

Table 5-3 Results of model retrieving for human body surfaces

	Best 3D model	Relation matrix	Translation matrix	Difference	CPU time (s)
Skeleton of feature of nose 1 image	Nose model	-0.7887, -0.4604, 0.4074 -0.5902, 0.3816, -0.7114 0.1720, -0.8015, -0.5727	0.0014 0.0013 -0.0003	0.0106	70.61863
Skeleton of feature of nose 2 image	Nose model	0.1240, -0.5919, 0.7965 -0.8653, 0.3284, 0.3787 -0.4857, -0.7361, -0.4714	0.0008 0.0014 0.0009	0.0139	65.0665
Skeleton of feature of the arm image	Arm model	-0.6507, -0.5128 0.5600 -0.6049, -0.0958, -0.7905 0.4590, -0.8532, -0.2479	-0.0023 -0.0114 0.0246	0.1374	70.05843

5.5 Summary

This Chapter proposes an image-based model retrieval method based on feature skeletons of the model and model image. The proposed method shows its accuracy and efficiency in different applications of the model retrieval.

Chapter 6 Optimization of 3D printing direction

Model segmentation divides a 3D model into several parts to be printed in different printing directions to increase the surface quality (Wang et al, 2016). Spectral clustering and deep-learning methods are proposed for the model segmentation to improve surface quality and reduce support materials of 3D printed products.

6.1 Feature extraction

6.1.1 Printing direction and surface quality of 3D printed products

3D printing is a layered manufacturing process. Surface quality of printed products is affected by the printing direction, called the staircase effect (Oropallo and Piegl, 2016). *Figure 6-1* shows the relation between outer surfaces of a 3D printing model and printing directions. An outer surface of the 3D printing model should be parallel to the printing direction for a high surface quality. The layered 3D printing causes the cusp height as follows.

$$\gamma_\theta = |\cos(\theta)| \cdot h = |n^{\mathrm{T}} d| \cdot h \tag{6-1}$$

where, h is the thickness of each layer, and θ is an angle between surface normal vector n and printing direction d.

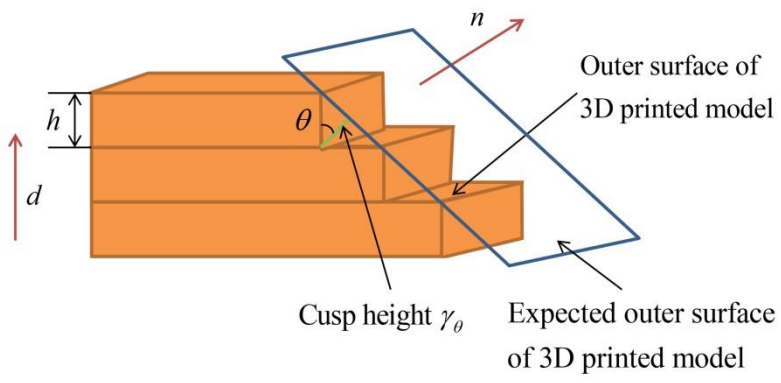

Figure 6-1 Cusp height of the surface in a 3D printed model

Cusp height γ_θ is affected by *h* and $\cos(\theta)$. The value of *h* is decided based on setup of the 3D printer. When *h* is a constant, cusp height γ_θ is influenced by $\cos(\theta)$. A smaller cusp height γ_θ results in a better surface quality. Thus, when a model surface is parallel to the printing direction, γ_θ is the lowest and the surface quality is the best. When the surface is vertical to the printing direction, γ_θ is the highest and the surface has the poor surface quality. If θ=0°, $\gamma_\theta = h$; if θ=45°, $\gamma_\theta = 0.707h$; and if θ=90°, $\gamma_\theta = 0$. To improve the surface quality, θ should meet $\theta^- < \theta < \theta^+$, when $\theta^- < 90°$ and $\theta^+ > 90°$, cusp height *h* remains small.

For each triangle *f* in mesh surface *M*, the surface quality is as follows.

$$E(f) = \gamma_\theta \cdot \text{area}(f) \tag{6-2}$$

where, $\text{area}(f)$ is the area of triangle *f*.

To decide the surface quality of whole triangular mesh model *M*, $E(M)$ is defined by integrating the surface quality of a triangle *f* as follows. Reducing $E(M)$ can increase surface quality of a 3D printed model.

$$E(M) = \sum_{f \in M} E(f) \tag{6-3}$$

$q(i, j)$ is a measure between a triangle and printing direction as follows.

$$q(i, j) = \left| N(f_j)^{\mathrm{T}} N(f_i) \right| \cdot h \cdot \text{area}(f_j) \tag{6-4}$$

where *i* is the triangle with a normal vector selected for the printing direction and *j* is another triangle in the mesh model. $N(f_j)$ is the normal vector of triangle f_j. $N(f_i)$ is the normal vector of triangle f_i for the printing direction.

After searching for all printing directions, the surface quality of triangle *j* is defined in Eq. (6-5).

$$Q_f(j) = \min[q(j)] \tag{6-5}$$

where $q(j)$ is the column of matrix $q(i, j)$ for the surface quality of triangle *j* with

different printing directions. Q_f in each triangle is then normalized into interval $[0, 1]$ by the Min-max normalization as follows.

$$Q'_f = \frac{Q_f - \min(Q_f)}{\max(Q_f) - \min(Q_f)} \tag{6-6}$$

After segmentation, the surface quality of an entire model is defined by Eq. (6-7) when the normal vector of triangle *i* is selected as the printing direction.

$$WQ_M = \min\left[\sum_{j=1}^{n} q(i, j)\right] \tag{6-7}$$

6.1.2 Support structure of 3D printing product

When printing overhangs of a 3D model, the support structure is required to keep the structure stable. There are three types of overhangs including point overhang, edge overhang and face overhang (Vanek et al, 2014). Point overhang is that the position of a point is lower than its neighboring points. For a triangular face, if angle α between a plane defined by the face and printing direction vector *d* is larger than critical angle $\hat{\alpha}$, a face overhang will form. *Figure 6-2* shows an example of the face overhang. When $\alpha_1 < \hat{\alpha}$, f_1 can be printed without the support structure. Face f_2 requires the support structure because of $\alpha_2 > \hat{\alpha}$ (Vanek et al, 2014). Similarly to the face overhang, edge overhang is measured by the edge normal which is the average of normal vectors from two incident faces.

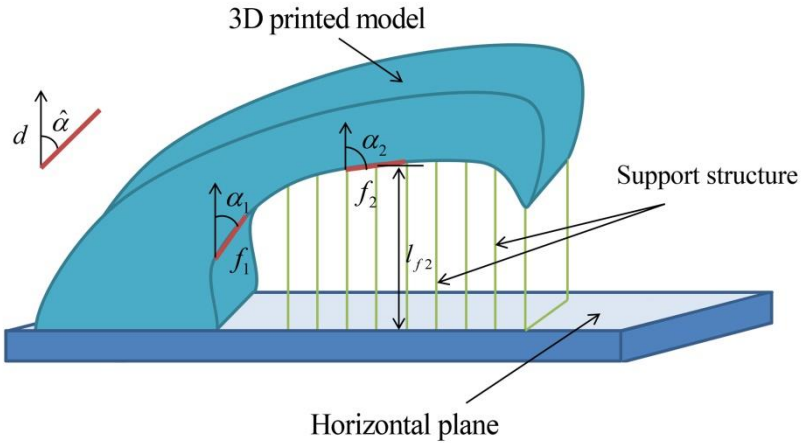

Figure 6-2 The structure of support material

Critical angle $\hat{\alpha}$ is 45° for most of 3D printers (Vanek et al, 2014, August). Support structures are assumed in the vertical direction. The support structures can be decided by the total support structures based on face overhangs.

$$S = \lambda \int_{\alpha \geq \hat{\alpha}} l_v \mathrm{d}\Gamma \tag{6-8}$$

where, S is the support structure; α is a subtended angle; l_v is the length of support structure at point v; λ is a fill ratio (relative material density) of the support structure; Γ is the area of surface.

To simplify the integration process in Eq. (6-8), the minimum unit of integration is used as the area of each triangular. Thus, Eq. (6-8) can be rewritten as follows.

$$S = \lambda \sum_{\alpha \geq \hat{\alpha}} \mathrm{area}(f) \cdot l_f \tag{6-9}$$

where $\mathrm{area}(f)$ is the area of triangle f, l_f is the distance between the center of gravity of triangle f and the horizontal place.

$u(i, j)$ is the support structure between each triangle and printing direction d as shown in Eq. (6-10), where i is the triangle to decide printing direction; and j is another triangle on M.

$$u(i,j) = \begin{cases} \lambda \cdot \mathrm{Area}(f_j) \cdot l_{ij} & N(f_j)^{\mathrm{T}} N(f_i) > \cos(45°) \\ 0 & N(f_j)^{\mathrm{T}} N(f_i) \leq \cos(45°) \end{cases} \tag{6-10}$$

If the normal vector of triangle i is selected as printing direction d, and a place which is vertical to direction d and through the centroid of triangle i is selected as the datum plane, l_{ij} is the distance between the centroid of triangle j and the datum plane as follows.

$$l_{ij} = \|c_j - c_i\| \cdot (N(f_j)^{\mathrm{T}} N(f_i)) \tag{6-11}$$

where, c_i and c_j are the centroid of triangle i and triangle j, respectively.

After searching for all printing directions, the support structure of triangle j is decided as follows.

$$U_f(j) = \min[u(j)] \tag{6-12}$$

U_f of each triangle is normalized into interval $[0, 1]$ by the min-max

normalization as follows.

$$U_f' = \frac{U_f - \min(U_f)}{\max(U_f) - \min(U_f)} \tag{6-13}$$

After segmentation, the support structure of a whole model is decided using Eq. (6-14) when the normal vector of triangle *i* is selected as the printing direction.

$$WU_M = \min\left[\sum_{j=1}^{n} u(i, j)\right] \tag{6-14}$$

6.1.3 Normal curvature

Normal curvature of a mesh model is applied as a feature in the model segmentation to instruct segmenting 3D model in a concave area.

The normal curvature of vertex v_i is defined by Eq. (6-15).

$$K(v_i) = \frac{2\left\langle N(v_i), \overline{v_j - v_i}\right\rangle}{\left\|v_j - v_i\right\|^2} \tag{6-15}$$

where v_j is one-ring neighbor vertices of v_i.

The normal curvature of v_i cannot be calculated by Eq. (6-15) directly, Thus the Moreton-Sequin method is used to estimate the normal curvature of v_i. N_{v_i} is the normal vector of vertex v_i. t_i is the unit projection of vector v_iv_j in the tangent plane of v_i. The normal curvature in v_i is the reciprocal of radius from the circle, which can be estimated using vertexes v_i. v_j and tangential vector t_i in v_i are shown in *Figure 6-3*.

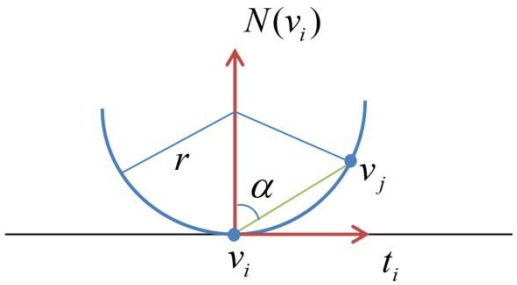

Figure 6-3 Sketch map of estimations for normal curvature

$$\frac{N(v_i)\cdot(v_j - v_i)}{\left\|v_j - v_i\right\|} = \cos\alpha = \frac{\frac{1}{2}\left\|v_j - v_i\right\|}{r} \tag{6-16}$$

$$\tilde{K}(v_i) = 2\frac{N(v_i)\cdot(v_j - v_i)}{(v_j - v_i)\cdot(v_j - v_i)} \tag{6-17}$$

Triangle f is connected with vertices v_i, v_j, and v_{j+1}. The normal curvature of triangle f is estimated as follows.

$$\tilde{K}(f) = \frac{\tilde{K}(v_i) + \tilde{K}(v_j) + \tilde{K}(v_{j+1})}{3} \tag{6-18}$$

Triangle f_i is a concave area if its normal curvature satisfies the follow equation.

$$\tilde{K}(f) > \varepsilon \tag{6-19}$$

$\tilde{K}(f)$ of each triangle is normalized into interval $[0, 1]$ by the min-max normalization as follows.

$$\tilde{K}'(f) = \frac{\tilde{K}(f) - \min(\tilde{K}(f))}{\max(\tilde{K}(f)) - \min(\tilde{K}(f))} \tag{6-20}$$

6.2 Model segmentation by spectral clustering

For the spectral clustering-based shape segmentation methods, triangles on a 3D mesh surface are considered as nodes in a graph, distances between centers of gravity for two adjacent triangles are considered as edges. Printing features of surface quality and support structure are extracted from each triangle of a mesh model to form a similarity matrix. A Laplacian matrix is then structured based on the similarity matrix. A spectral clustering method is applied to cluster the surface into several parts. To reduce the connection difficulty of pieced parts, the cutting line is optimized to generate smooth boundaries.

6.2.1 Nodes transformation for graph-cut

To segment a 3D shape into several pieces, a spectral clustering method is proposed to combine the surface feature and support structure to guide segmentation. Spectral clustering uses an algorithm evolved from the graph theory. The main idea of the proposed spectral clustering-based shape segmentation method is that all the vertices

of a mesh surface are considered as nodes in a space and connected by edges. The weight value of an edge is lower if two nodes are far away from each other compared to the higher weight value of an edge with two closed nodes. The cutting graph divides a whole graph into several sub-graphs by optimizing weighting values between sub-graphs as low as possible and weighting values in sub-graphs as high as possible.

Surface quality and support structure in a 3D printing product are extracted based on triangles of a 3D surface mesh model. Thus, triangles of the model are considered as nodes in the graph. Common edges between two triangles are considered as edges in the graph. The length of edges in the graph is calculated for the distance between centers of gravity for two adjacent triangles. The transformation is shown in *Figure 6-4*.

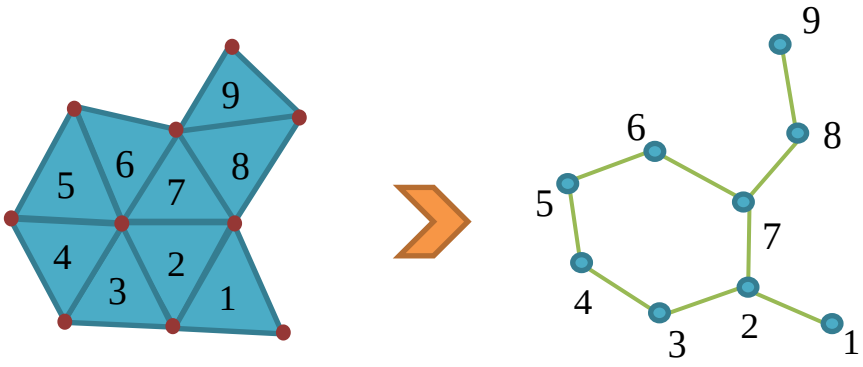

Figure 6-4 Transformation from a 3D mesh surface to graph

$e(i, j)$ is a matrix to show the length of edges in a graph as follows.

$$e(i,j) = \begin{cases} \|c_i - c_j\|^2 & i, j \text{ are adjacent triangles} \\ 0 & \text{otherwise} \end{cases} \quad (6\text{-}21)$$

6.2.2 Laplacian matrix

Building a Laplacian matrix is the key step in the spectral clustering. Matrix W is an adjacency matrix in Laplacian matrix L to measure similarity between two data points. For segmenting a surface by combining features including the surface quality and

support structure, each triangle is considered as a node in the graph, and two connected triangles form an edge. Adjacency matrix W_{ij} is formed by Eq. (6-22) to combine features including the surface quality and support structure with topology of the 3D model.

$$W_{ij}=\begin{cases}\exp(-\dfrac{e'(i,j)^2}{2\sigma^2})+\eta_1\cdot q'(i,j)+\eta_2\cdot u'(i,j) & i,j \text{ are adjacent triangles}\\ 0 & \text{otherwise}\end{cases} \tag{6-22}$$

where, $q'(i,j)$ and $u'(i,j)$ are features of the surface quality and support structure, respectively, they are transferred to interval [0, 1] as shown in Eqs. (6-23) and (6-24). η_1 and η_2 are weights.

$$q'(i,j)=q'(j,i)=\begin{cases}1 & q(i,j)<0.5,\ q(j,i)<0.5\\ 0.5 & q(i,j)<0.5,\ q(j,i)>0.5 \text{ or } q(i,j)>0.5,\ q(j,i)<0.5\\ 0 & \text{otherwise}\end{cases} \tag{6-23}$$

$$u'(i,j)=u'(j,i)=\begin{cases}1 & u(i,j)<\text{threshold},\ u(j,i)<\text{threshold}\\ 0.5 & u(i,j)<2\cdot\text{threshold},\ u(j,i)<2\cdot\text{threshold}\\ 0 & \text{otherwise}\end{cases} \tag{6-24}$$

Normalization of $e(i,j)$ in Eq. (6-21) forms Min-max normalization $e'(i,j)$ as follows.

$$e'(i,j)=\frac{e(i,j)-\min(e)}{\max(e)-\min(e)} \tag{6-25}$$

6.2.3 Area balance cut

To generate a 3D printed product with several parts, if a small-sized part is generated by the shape segmentation, it may increase the difficulty of connecting the small size part with other parts to form the entire product. To avoid generating small-sized parts by considering the minimum loss function and maximum area in each sub-graph, Area balance cut (ABcut) is proposed as follows to avoid generating the very small-sized piece.

$$\mathrm{ABcut}(A_1, A_2, \cdots A_k) = \frac{1}{2}\sum_{p=1}^{k}\frac{W(A_p, \bar{A}_p)}{\mathrm{area}(A_p)} \tag{6-26}$$

where, $W(A_p, \bar{A}_p) = \sum_{i\in A_p, j\in \bar{A}_p} W_{ij}$, $\mathrm{area}(A_p) = \sum_{i\in A_p}\mathrm{area}(f_i)$, $\mathrm{area}(f_i)$ is the area of triangle *i*.

For a $k\times m$ dimensional indicator matrix $H = \{h_1, h_2, \cdots h_k\}$, h_p is an *m*-dimensional vector as follows.

$$h_{pi} = \begin{cases} 0 & i \notin A_p \\ \dfrac{1}{\sqrt{\mathrm{area}(A_p)}} & i \in A_p \end{cases} \tag{6-27}$$

$$\begin{aligned} h_p^{\mathrm{T}} L h_p &= \frac{1}{2}\sum_i\sum_j W_{ij}(h_{pi} - h_{pj})^2 \\ &= \frac{1}{2}\left(\sum_{i\in A_p, j\notin A_p} W_{ij}\left(\frac{1}{\sqrt{\mathrm{area}(A_p)}} - 0\right)^2 + \sum_{i\notin A_p, j\in A_p} W_{ij}\left(0 - \frac{1}{\sqrt{\mathrm{area}(A_p)}}\right)^2 \right) \\ &= \frac{1}{2}\frac{W(A_p, \bar{A}_p)}{\mathrm{area}(A_p)} \\ &= \mathrm{ABcut}(A_p, \bar{A}_p) \end{aligned} \tag{6-28}$$

$$\mathrm{ABcut}(A_1, A_2, \cdots A_k) = \sum_{p=1}^{k} h_p^{\mathrm{T}} L h_p = tr(H^{\mathrm{T}} L H) \tag{6-29}$$

Matrix *F* is built as shown in Eq. (6-30) which is a diagonal matrix.

$$F_{ii} = \mathrm{area}(f_i) \tag{6-30}$$

$$h_p^T F h_p = \sum_{i=1}^{m} h_{pi}^2 \cdot \mathrm{area}(f_i) = \frac{1}{\mathrm{area}(A_p)}\sum_{i\in A_p}\mathrm{area}(f_i) = 1 \tag{6-31}$$

The cut is transferred into an optimization problem as follows.

$$\begin{aligned} &\min\left\{H^T L H\right\} \\ &s.t. \quad H^T F H = I \end{aligned} \tag{6-32}$$

If $H = F^{-\frac{1}{2}} H'$,

$$H^T L H = H'^T F^{-\frac{1}{2}} L F^{-\frac{1}{2}} H' \tag{6-33}$$

$$H^T F H = H'^T F^{-\frac{1}{2}} F F^{-\frac{1}{2}} H' = H'^T H' = I \tag{6-34}$$

Based on the Rayleigh quotient model,

$$\lambda_{\min} \le h'^{T} F^{-\frac{1}{2}} L F^{-\frac{1}{2}} h' \le \lambda_{\max} \tag{6-35}$$

where, $\lambda_{\min}$ is the minimum eigenvalue of $F^{-\frac{1}{2}} L F^{-\frac{1}{2}}$, $\lambda_{\max}$ is the maximum eigenvalue of $F^{-\frac{1}{2}} L F^{-\frac{1}{2}}$.

Thus, for the area balance cut, Laplacian matrix can be normalized as follows.

$$L' = F^{-\frac{1}{2}} L F^{-\frac{1}{2}} \tag{6-36}$$

Normalized Laplacian matrix L' has *m* non-negative eigenvalues $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_m$. Corresponding eigenvectors $U = (u_1, u_2, \cdots, u_m)$ form an orthogonal basis. First *k* largest non-null eigenvectors are used to define a matrix $U^{(m \times k)} = (u_1, u_2, \cdots, u_k)$. After normalized by $U' = F^{-\frac{1}{2}} U^{(m \times k)}$, matrix U' is a *k*-dimensional feature space of the mesh triangular surface.

To segment a model into several parts, data in the feature space are clustered into several clusters by a Gaussian mixture model (GMM).

6.2.4 Boundary optimization

After segmenting a model into several parts, the common boundary between two parts is not smooth because of combining edges of triangles on a mesh model. A surface with such common boundaries cannot be used to build a model for 3D printing directly, because it is difficult to put pieces together. Therefore, smooth boundaries are formed for easy to put pieces together, thus, liner or quadratic functions are used as smooth boundaries between two parts.

A tangent plane in the surface is estimated based on positions of all vertices. After projecting all the points to the tangent plane, a cutting line $Q_{ij} : v_y = f(v_x) = a_1 {v_x}^2 + a_2 v_x + a_3$ between two parts (parts *i* and *j*) is formed to divide two neighboring patches into two parts with a smooth boundary. The method of generating a smooth boundary is to minimize the objective function as follows.

$$\min \sum_{ij} \left\| VT_{ij} - VQ_{ij} \right\|^2 \tag{6-37}$$

where VT_{ij} is positions of vertices transferred to the tangent plane for parts *i* and *j*. VQ_{ij} is the corresponding position in cutting line Q_{ij} of vertices transferred to the tangent plane for parts *i* and *j*.

6.3 Results of model segmentation by spectral clustering

6.3.1 Model 1

Model 1 is a free-form surface with 4019 vertices, 11762 edges and 7744 triangles for 3D printing. Triangles on the triangular mesh surface are transferred into nodes of a graph in graph-cut, and the number of nodes is 7744.

This study tests performance of the spectral clustering method with ABcut to segment a surface model into several parts in order to increase the surface quality and reduce the support material. The solution is compared to the result generated by a traditional spectral clustering method that builds the similarity matrix by nodes and edges in the mesh surface directly and cuts the graph using the RatioCut method.

The traditional and proposed methods are applied to segment Model 1 with clustering number 3. *Figure 6-5* shows Model 1 segmented by the traditional method, where *Figure 6-5* (a) is the feature space extracted from the original triangle surface, *Figure 6-5* (b) shows segmentation results. The model surface is segmented into 3 parts (green-part 1, yellow-part 2 and blue-part 3). The area of part 1 is larger than areas of parts 2 and 3. *Figure 6-5* (c) shows segmented parts after generating a smooth boundary.

Figure 6-6 shows segmented results of Model 1 by the proposed method, where *Figure 6-6* (a) is the feature space extracted from the original triangle surface. *Figure 6-6* (b) shows segmentation results. The surface is segmented into 3 parts (green-part 1, yellow-part 2 and blue-part 3). *Figure 6-6* (c) shows segmented parts after generating a smooth boundary.

In *Figure 6-5* (b), the area of part 1 is larger than areas of parts 2 and 3, and the size of part 3 is very small. In *Figure 6-6* (b), parts 1, 2 and 3 have a similar size.

Comparing with the traditional method, the proposed method can divide a surface into several parts with similar areas to avoid generating very small-sized parts.

Table 6-1 lists numbers of vertices, edges, and triangles, values of surface areas, surface quality, surface support structure, and balanced surface quality and surface support structure for the whole surface. Each part is generated from the two clustering methods. Comparing with the traditional spectral clustering method, the proposed method can segment the surface into several parts with similar areas and there is not a small-sized part.

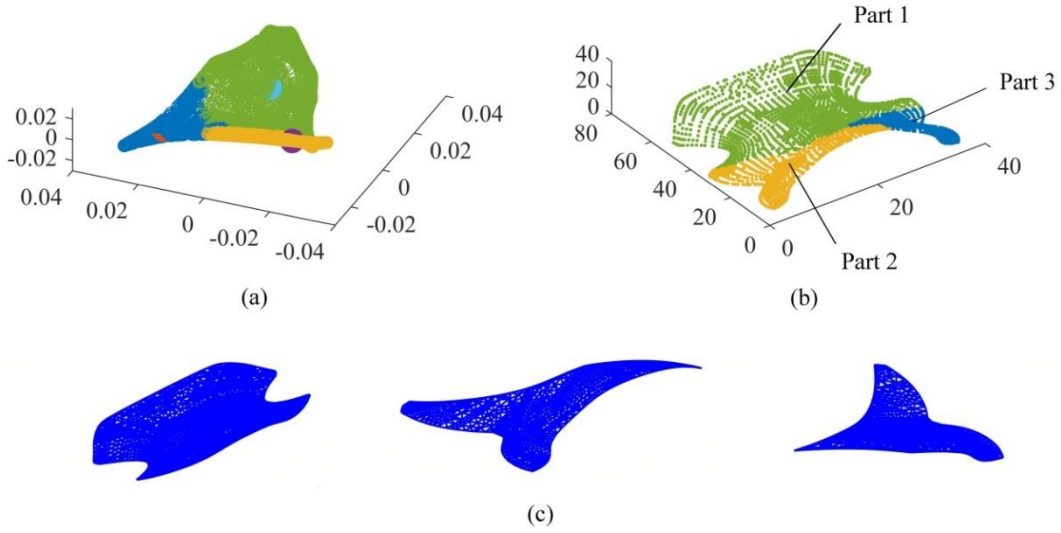

Figure 6-5 Results of Model 1 segmented by the traditional spectral clustering method

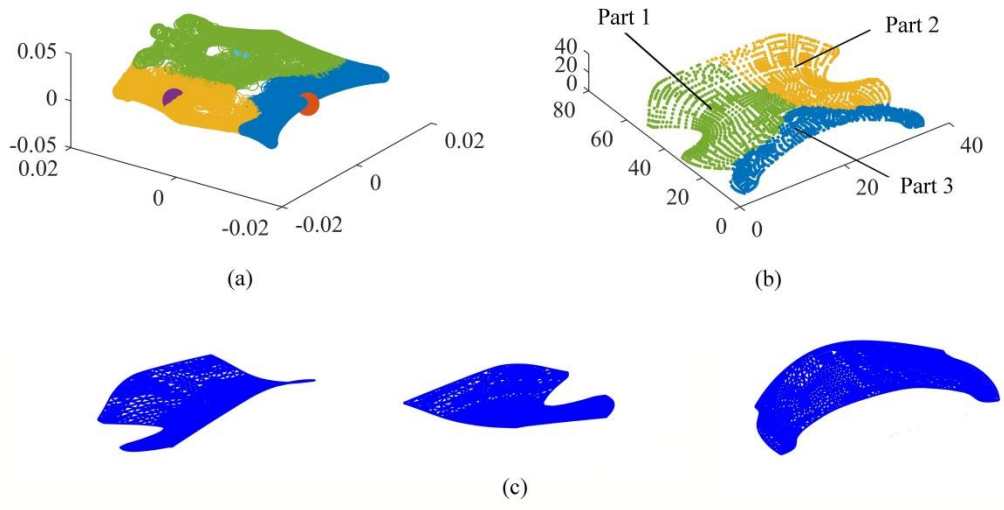

Figure 6-6 Results of Model 1 segmented by the proposed method

Table 6-1 Results of Model 1

Model 1		No. vertex	No. edge	No. triangle	Area	WQ	WU	WQ+WU
Whole model		4019	11762	7744	2648.9	690.7	185.6	1202.1
Traditional method	Part1	2237	6401	4165	1667.4	764.03	147.91	949.5
	Part2	1204	3360	2157	527.1	71.9	1.1	73.4
	Part3	1062	2963	1902	454.4	73.4	0.5	73.9
Proposed method	Part1	1310	3361	2352	937.0	213.2	46.8	262.7
	Part2	1228	3461	2234	872.3	335.5	41.8	638.9
	Part3	1943	5558	3616	839.6	140.2	2.1	146.1

Figure 6-7 shows clustering results using the traditional spectral method and proposed method for Model 1 with cluster numbers 2, 4, 5 and 6, respectively. *Figure 6-7* (a)-(d) are formed using the traditional method. *Figure 6-7* (e)-(h) are generated using the proposed method. Comparing with the traditional method, the proposed method can segment the surface into several patches with a similar size. In *Figure 6-7* (a), the blue part is larger than red part, while the area of the blue part is similar to the area of red part in *Figure 6-7* (e). In *Figure 6-7* (b), the area of purple part is very small, and in *Figure 6-7* (c), sizes of yellow and blue parts are also very small. There is no small area formed in *Figure 6-7* (f)-(h). Thus, the proposed method can segment the surface into several parts without very small-sized pieces.

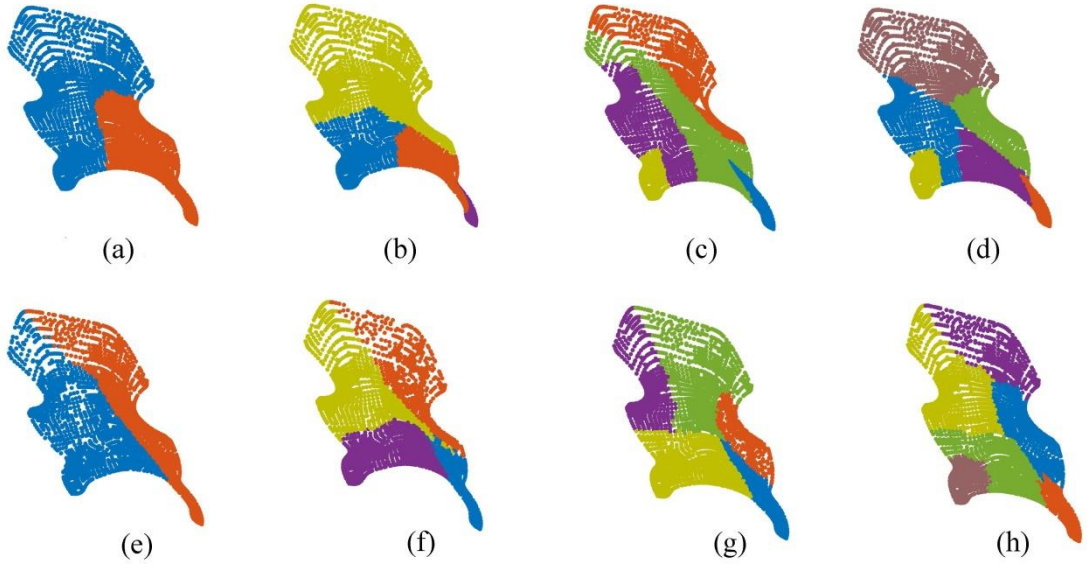

Figure 6-7 Clustering results of Model 1 using the traditional and proposed methods with clustering number 2, 3, 4, 5 and 6, respectively

6.3.2 Model 2

Model 2 is applied to further test the proposed method. It contains 2509 vertices, 7451 edges and 4943 triangles. Triangles on the triangular mesh surface are transferred into nodes of a graph in graph-cut, and the number of nodes is 4943.

Figure 6-8 and *Figure 6-9* show results of Model 2 segmented by the traditional and proposed methods respectively with clustering number 3. *Figure 6-8* (a) and *Figure 6-9* (a) show the feature space extracted from the original triangle surface. *Figure 6-8* (b) shows segmentation results of Model 2 with 3 parts (blue-part 1, yellow-part 2 and green-part 3). Sizes of three parts are similar. *Figure 6-9* (b) shows results generated by the proposed method where Model 2 is segmented into 3 parts (blue-part 1, green-part 2 and yellow-part 3). Sizes of three parts are also similar. *Figure 6-8* (c) and *Figure 6-9* (c) are segmented results after generating a smooth boundary.

Table 6-2 lists numbers of vertices, edges, and triangles, values of area, surface quality, support structure, balanced surface quality and surface support structure for the whole surface and segmented results by two clustering methods. For this model, the surface segmented by the proposed methods can generate several parts with similar areas. The segmented surface increases the surface quality and reduces the surface support material based on the value of WQ+WU.

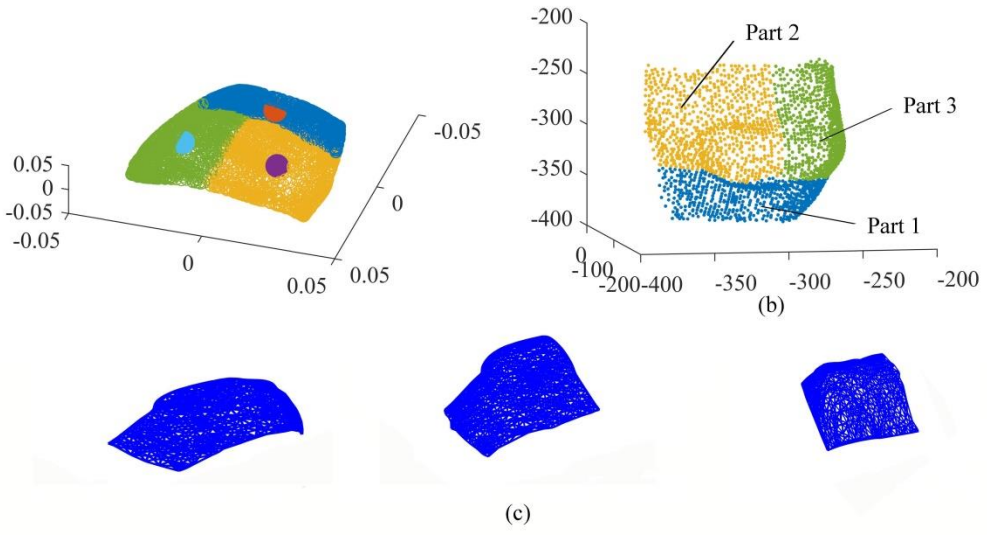

Figure 6-8 Model 2 segmented by the traditional spectral clustering method

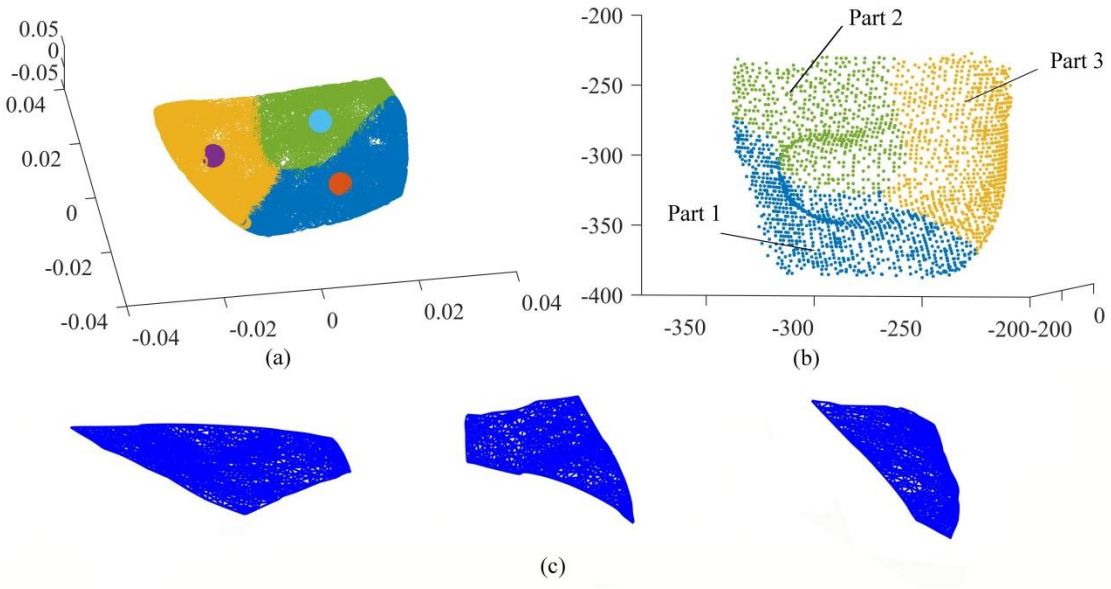

Figure 6-9 Model 2 segmented by the proposed method

Table 6-2 Results of Model 2

Model 2		No. vertex	No. edge	No. triangle	Area	WQ	WU	WQ+WU
Whole model		2509	7451	4943	34491	12690	0	13384
Traditional method	Part1	853	2411	1559	9302.4	1222.6	0	1222.6
	Part2	1011	2872	1862	12802	3796.5	1399.3	5955.6
	Part3	1029	2930	1902	12387	2657.3	1337.1	5005.0
Proposed method	Part1	1177	3346	2170	13711	2119.8	0	2119.8
	Part2	842	2288	1145	8782.4	1939.9	2739.5	4985.0
	Part3	998	2824	1826	11986	2269.2	1010.5	4251.7

Figure 6-10 shows clustering results using the traditional and proposed methods with cluster numbers 2, 4, 5 and 6, respectively for Model 2. *Figure 6-10* (a)-(d) are formed using the traditional method. *Figure 6-10* (e)-(h) are generated using the proposed method. For cluster number 2, *Figure 6-10* (a) and (e) show the similar performance. For cluster number 4 in *Figure 6-10* (b), the area of red part is larger than sizes of other three parts. In *Figure 6-10* (f), areas of red and yellow parts are larger than sizes of green and blue parts. The proposed method can segment Model 2 into five and six clusters with similar sizes. Areas of clusters segmented by the

traditional method have large differences. The proposed method shows good performance compared to the traditional method.

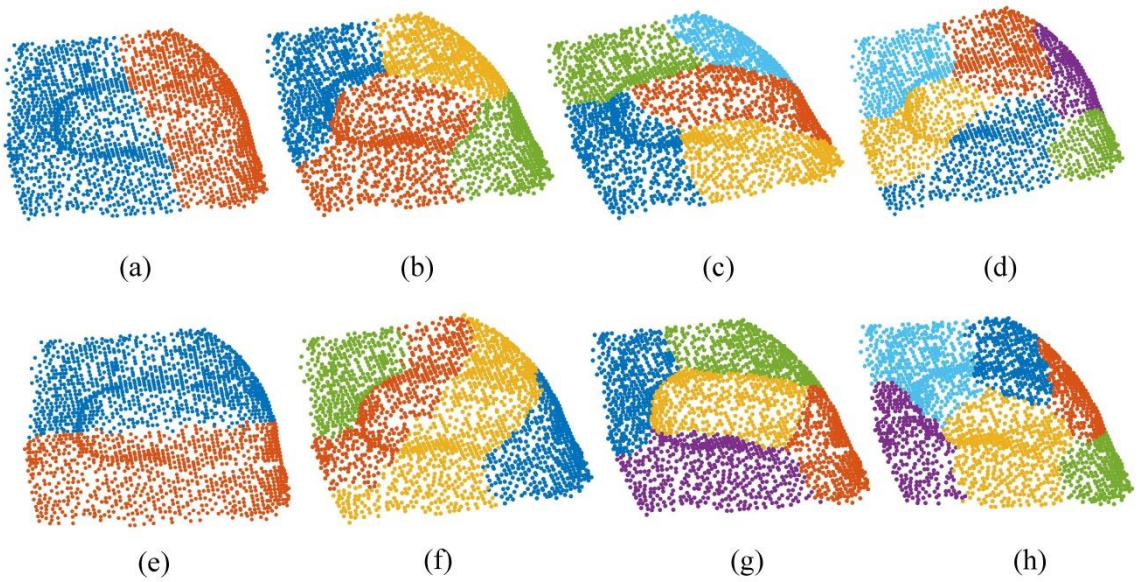

Figure 6-10 Clustering results using the traditional and proposed methods with clustering number 2, 3, 4, 5 and 6, respectively, for Model 2

6.3.3 Experiment

To further validate the performance of the proposed method, the whole model and segmented parts of Model 1 by the proposed method are printed using a 3D printer. To compare the surface quality and support material between the whole model and pieced models, the models are printed using the same 3D printer and same material. Printing directions of each part are shown in *Figure 6-11*, where the red arrow shows the printing direction.

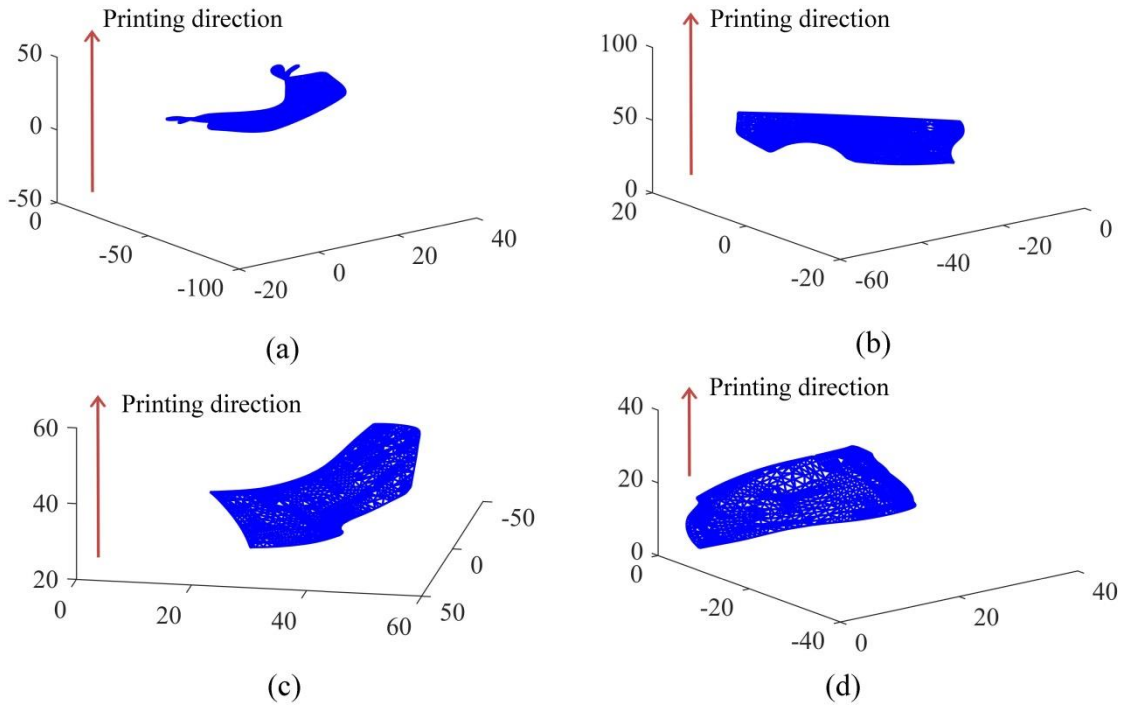

Figure 6-11 Printing directions of whole model and segmented results of Model 1

Based on segmented results, 3D models are built using a 3D modeling tool as shown in *Figure 6-12* (a) for the whole model, (d) is part 1 of the segmented result, (g) is part 2 of the segmented result, and (j) is part 3 of the segmented model.

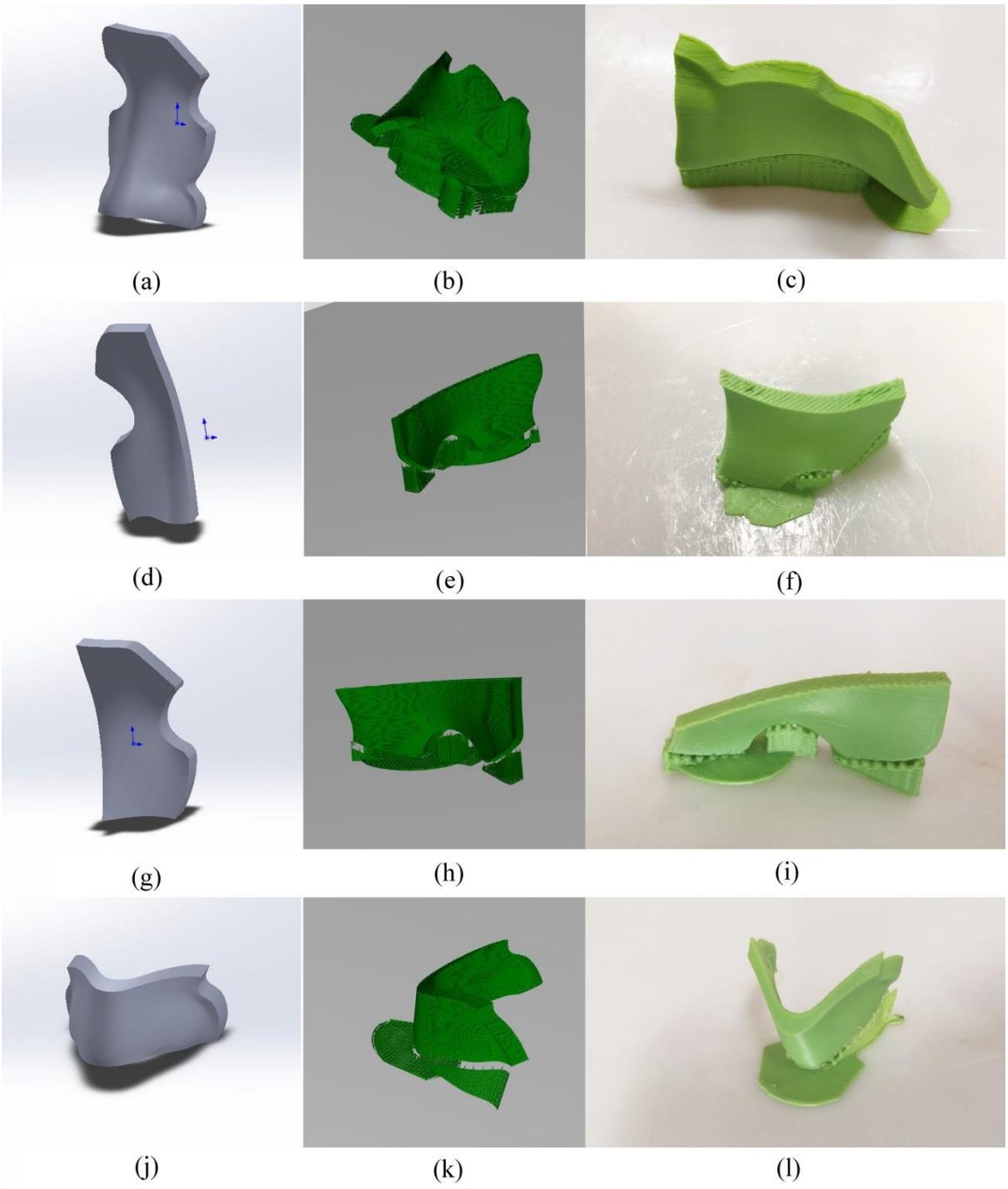

Figure 6-12 The 3D model, layered model and printed model of Model 1

For the layered 3D model with the support structure, *Figure 6-12* (b) is the whole Model 1, (e) shows part 1 of the segmented model, (h) is part 2 of the segmented model, and (k) is part 3 of the segmented model.

A CubePro Duo 3D printer is applied to print the models. The material used is green ABS and the layer resolution is 200um. The printed models are shown in *Figure*

6-12, where (c) is whole Model 1, (f) is part 1 of the segmented model; (i) is part 2 of the segmented model; and (l) is part 3 of the segmented model.

After printing the models, the support structures are removed from printed models and weighted by a weight scale. *Figure 6-13* shows weights of the support material for each model, where (a) is for whole Model 1, (b) is part 1 of the segmented model, (c) is part 2 of the segmented model, and (d) is part 3 of the segmented model. The total weight of the support structure for three segmented models is 1.32g which is lighter than the weight of the support material for the whole model (2.09g).

After removing the support structure from the printed model, the outer surface finish of the model is affected as shown in *Figure 6-14* (a). Thus, the support structure should have the small contact area with the printing model and located at the unimportant position.

Based on the printing direction, there are some stripes that occur in the inner surface of the whole model as shown in *Figure 6-14* (b). The scatter plot shows that the distance between two peaks or valleys is 1.83 mm and the distance between peaks and valleys is 0.4mm.

Three segmented parts are printed respectively, and then pieced together as shown in *Figure 6-14* (c) and (d). There is a gap between each part, which should be reduced by increasing the printing accuracy. The surface quality of inner and outer surfaces of the pieced model depends on the printing layer resolution. The scatter plot shows that the distance between two peaks or valleys is 0.2 mm and the distance between peaks and valleys is 0.2 mm. Thus, the pieced model has better surface quality compared to the whole model.

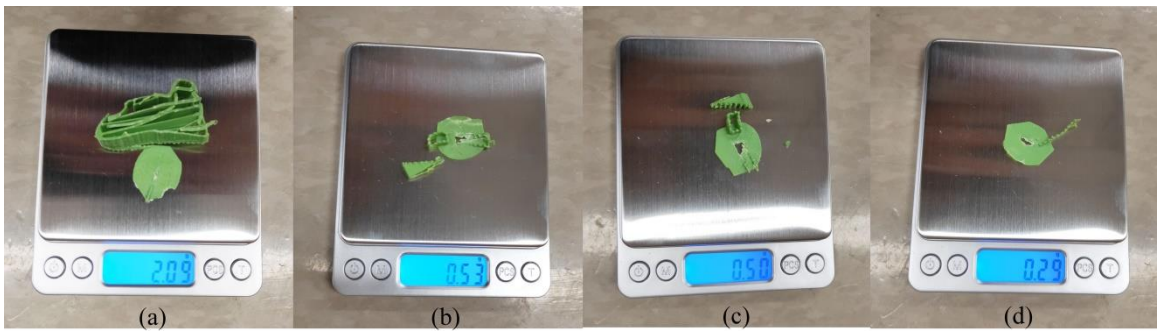

Figure 6-13 Weight of support material

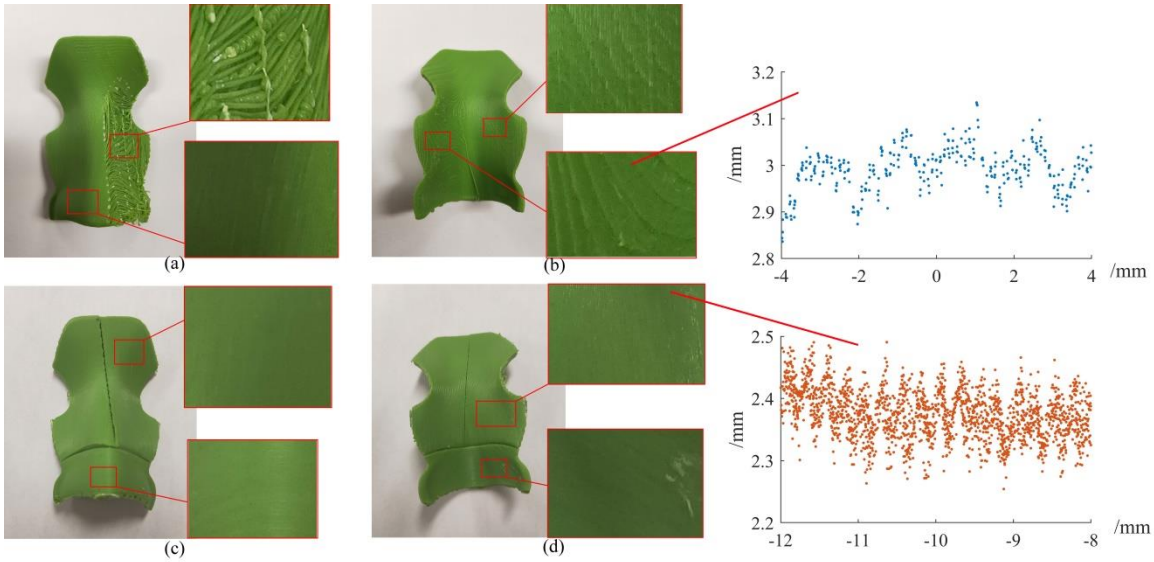

Figure 6-14 Surface quality analysis of 3D printed models

6.4 Model segmentation by Auto-encoder

The above proposed method only combines features of surface quality and support structure, which has limitations in the clustering number decision. Features including the curvature, surface quality and support materials can be considered respectively in the model segmentation for 3D printing, but there is not a method combing all the features. Therefore, a deep-learning based shape segmentation method is developed to combine all the printing features of the 3D model.

For the deep-learning based shape segmentation method, sub-graphs are generated by pre-segmenting 3D triangular mesh models to extract printing features. A data structure is proposed to form training data sets based on the sub-graphs with printing features of the original 3D model including surface quality, support structure

and normal curvature. After training a Stacked Auto-encoder (SAE) using the training set, a 3D model is pre-segmented to build an application set by the sub-graph data structure. The set is applied by the trained deep-learning system to generate hidden features. An Affinity Propagation (AP) clustering method is introduced to combine hidden features and geometric information of the set to segment a product model into several parts.

6.4.1 Sub-graph data structure

A sub-graph data structure is proposed to generate a data set for training the Stacked Auto-encoder to form an application set of hidden features for the model segmentation. For example, a sub-graph with 9 triangles is shown in *Figure 6-15* (a).

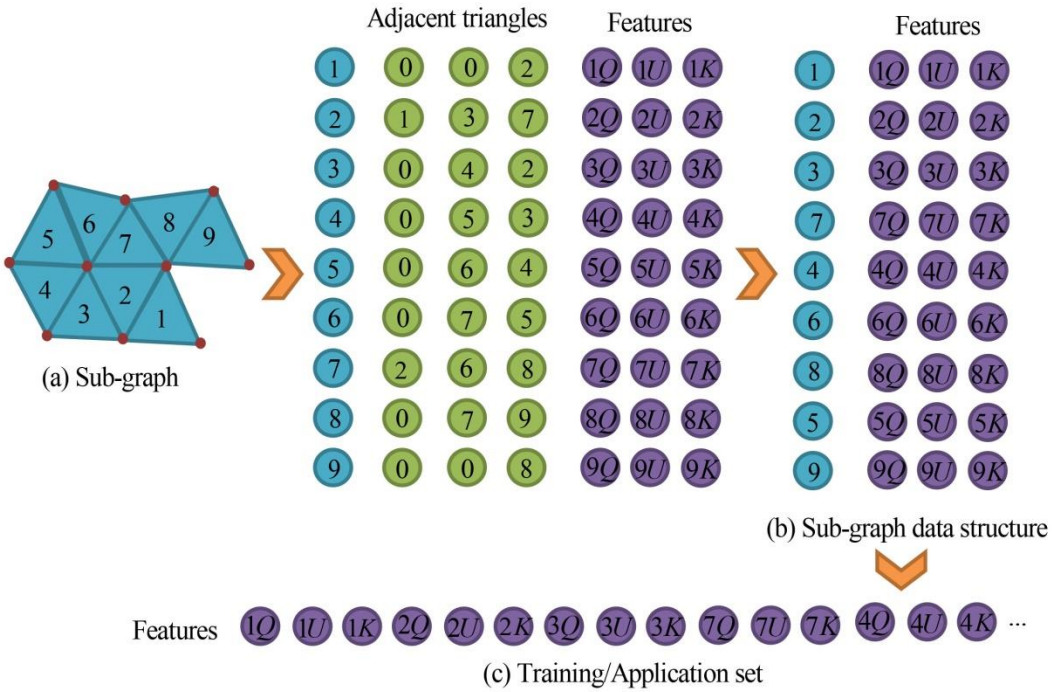

Figure 6-15 Sub-graph data structure

The sub-graph data structure is a matrix formed by triangles, adjacent triangles and features. In a sub-graph, an inner triangle has three adjacent triangles, and a boundary triangle has one or two adjacent triangles. If a triangle has three adjacent triangles, the position of adjacent triangles is filled with the label of the adjacent triangles. If the triangle has one or two adjacent triangles, the position without an

adjacent triangle is filled with 0. After that, the order of the triangle is changed to build a new structure by searching the triangle and its adjacent triangles. For example, in *Figure 6-15* (b), triangle 1 and its adjacent triangles are searched. Triangle 2 is the adjacent triangle of triangle 1. Adjacent triangles of triangle 2 are searched with results of triangles 1, 3, and 7. After adjacent triangles of triangle 1 are searched, adjacent triangles of triangle 3 are searched. The searching process continues until all the triangles are searched. Three printing features, including surface quality, support structure and normal curvature, are selected to characterize geometric properties in different perspectives. Features for increasing surface quality and reducing support structure are normalized into interval [0, 1] using Eqs. (6-6), (6-13) and (6-20), respectively. Features 1Q, 1U and 1K represent the normalized surface quality, support structure and normal curvature for triangle 1, respectively. To form the training set and application set, printing features of triangles in a sub-graph are reformatted into a row in *Figure 6-15* (c).

6.4.2 Training and application processes

Stacked Auto-encoders (SAEs) extract hidden features from printing features of an original 3D mesh model for unsupervised learning. The training process of generating hidden features is as follows.

Original input X, output $Y^1=X$ and features h^1 are used to train an Auto-encoder. Features h^1 are used as input to train the second Auto-encoder with output $Y^2= h^1$ and features h^2. Features h^2 are also used as input to train the third Auto-encoder. After training several times and stacking all Auto-encoders, features h^k are used as hidden features to guide the model segmentation.

Figure 6-16 shows an overall workflow of the process. It contains a training process and an application process.

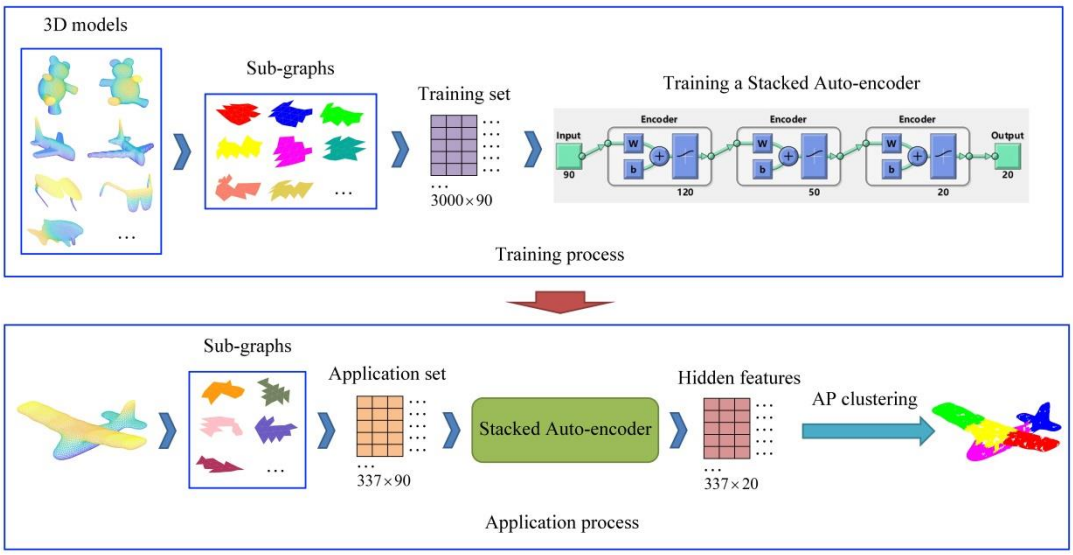

Figure 6-16 Workflow of training and application processes

In the training process, a sub-graph contains 30 triangles and 3000 sub-graphs is generated from 30 3D triangular mesh models in the PSB dataset (Kalogerakis et al, 2010) to build the training data set. Features are automatically extracted using Eqs. (6-6), (6-13) and (6-20) coded in MATLAB. Stacked Auto-encoders (SAE) was suggested to use 2-6 layers (Chen et al, 2014). It is not necessary to use more layers or hidden units for improving the system performance (Zabalza et al, 2016). Thus, three Auto-encoders are stacked to generate a SAE. Thus, the SAE has four layers including an input layer, two hidden layers, and one output layer. The number of vector dimensions in the input layer is 90. For two hidden layers, numbers of vector dimensions are 120 and 50, respectively. 20-dimensional hidden feature vectors are collected in the output layer. The training process is shown in Algorithm 6-1. For training the three Auto-encoders, iteration times are 1319, 413, and 237, respectively. The training time of the three Auto-encoders is 114s, 25s, and 5s, respectively, using MATLAB 2019a in a computer with an Intel Core i7-5500U, 8.00 GB RAM.

Algorithm 6-1 Training SAE

Input:

Sub-graph data structure, A matrix $PT_{3000\times90}$

Number of nodes for hidden layers, K_1=120, K_2=50, K_3=20

Weight of decay parameter, $\lambda = 0.004$

Sparsity parameter, $\rho = 0.05$

Weight of the sparsity penalty term, $\beta = 5$

Output:

SAE, W_1^1, $b_1^1, W_1^2, b_1^2, W_1^3, b_1^3$

1 Initialization: generating the initial encoder W_1^1, b_1^1 and decoder W_2^1, b_2^1 randomly

2 Calculation of objective function of the first Auto-encoder.

$$C_{1s}(X,Y,\theta) = C(X,Y,\theta) + \beta\sum_{i=1}^{N_H} KL(\rho \,\|\, \hat{\rho}_i)$$

3 Generating W_1^1, b_1^1, W_2^1, b_2^1

4 Calculating feature h^1

5 Inputting h^1 to train the second Auto-encoder

6 Initialization: generating the initial encoder W_1^2, b_1^2 and decoder W_2^2, b_2^2 randomly

7 Calculating objective function of the second Auto-encoder. $C_2(X,Y,\theta) = \frac{1}{2}\|Y - X\|^2 + \lambda D(\theta)$

8 Generating W_1^2, b_1^2, W_2^2, b_2^2

9 Calculating feature h^2

10 Inputting h^2 to train the third Auto-encoder

11 Initialization: generating the initial encoder W_1^3, b_1^3 and decoder W_2^3, b_2^3 randomly

12 Calculating objective function of the third Auto-encoder. $C_3(X,Y,\theta) = \frac{1}{2}\|Y - X\|^2 + \lambda D(\theta)$

13 Generating W_1^3, b_1^3, W_2^3, b_2^3

14 Calculating feature h^3

15 Stacking these three Auto-encoders to form SAE

In the application process, a 3D mesh model is pre-segmented into a set of sub-graphs. One sub-graph contains 30 triangles. Coordinates of central triangles for sub-graphs are combined with hidden features extracted from the sub-graph data structure by the Stacked Auto-encoder to build a similarity matrix. The AP clustering

method is used to cluster the sub-graphs into clusters to segment the model. The clustering process is shown in Algorithm 6-2.

Algorithm 6-2 Clustering process

Input:

Sub-graph data structure, A matrix $PU_{N\times 90}$

SAE, W_1^1, b_1^1, W_1^2, b_1^2, W_1^3, b_1^3

Attenuation coefficient, $\lambda = 0.9$

The maximum number of iterations, T=10000

The number of convergence iterations, IT=15

Output:

Clustering results

1 Generating hidden features using the SAE

2 Calculating similarity matrix based on the coordinate of central triangle for each sub-graph data structure and the hidden features

3 Initialization: generating the initial availability matrix and responsibility matrix as zero matrices

4 for i =1 to T

5 Updating responsibility matrix

6 Updating availability matrix

7 Decaying the responsibility matrix using λ, $r_{t+1}(i,k) = \lambda \times r_t(i,k) + (1-\lambda) \times r_{t+1}(i,k)$

8 Decaying the availability matrix using λ, $a_{t+1}(i,k) = \lambda \times a_t(i,k) + (1-\lambda) \times a_{t+1}(i,k)$

9 if IT<15 do

10 i=i+1

11 else

12 end for

13 end if

14 end for

15 Generating the clustering center

16 Generating clustering results

6.4.3 Common surface optimization and connector design

As a common surface between two adjacent parts is non-planar after clustering, the difficulty of assembling all the parts together increases. To reduce the difficulty of shaping and assembly of parts, Support Vector Machines (SVM) (Suykens et al, 2002) is introduced to fit a planar between two adjacent parts as this method is more robust than using the least squares method (Wang et al, 2016). The model is finally segmented based on the fitted planes.

A common plane P: $\boldsymbol{x}^T W + b = 0$ is generated for each pair of two adjacent patches. Each SVM plane P_{ij} generates an optimal common surface between two adjacent parts.

Generating the best position of common plane P can be defined as follows.

$$\begin{aligned} &\min_{W,b} J(W) = \min_{W,b} \frac{1}{2} \| W \|^2 \\ &s.t. \quad y_i = (\boldsymbol{x}_i^{\mathrm{T}} W + b) \geq 1, i = 1, 2, 3, ..., n \end{aligned} \tag{6-38}$$

where, $\boldsymbol{x}_i$ is a column vector of position for node v_i in a 3D triangular mesh shape, y_i is the label of clusters.

For the optimization problem of solving W and b to form plane P, Eq. (6-38) can be rewritten as follows.

$$\begin{aligned} &\min_{\alpha} \ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j - \sum_{i=1}^{n} \alpha_i \\ &s.t. \ \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\qquad \alpha_i \geq 0, i = 1, 2, 3, ..., n \end{aligned} \tag{6-39}$$

Plane P can be represented as follows.

$$\boldsymbol{x}^{\mathrm{T}} \hat{W} + \hat{b} = 0 \tag{6-40}$$

where, $\hat{W} = \sum_{i \in SV} \hat{\alpha}_i y_i \boldsymbol{x}_i$ and $\hat{b} = y_j - \sum_{i \in SV} \hat{a}_i y_i \boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{x}_i$

After the segmentation based on the fitted plane, connectors are added between adjacent segments to ensure that parts do not shift in the assembly. The shape of connectors is a simple cylinder. For two adjacent sub-objects, two connectors are added into one segment, one is convex and the other is concave to avoid shifting and increase firmness in contact. The position of the connectors is based on the shape of connected surfaces. Normally, connectors are placed on two sides closed to the edge of connected surfaces as shown in *Figure 6-17*. After deciding an assembly order, a connector is added to each segment with appropriate dimensions. Connectors are placed on the connected surface without affecting the assembly process.

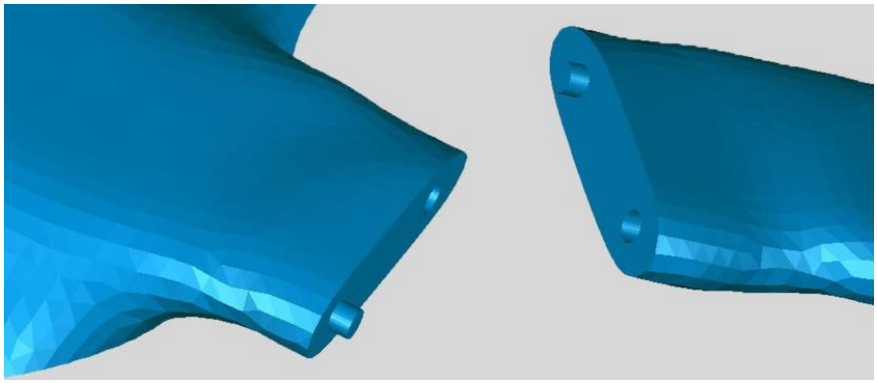

Figure 6-17 Connector design

6.5 Results of model segmentation by Auto-encoder

6.5.1 Simulation of the model segmentation

3D triangular mesh models are used to validate the proposed SAEs for the model segmentation. Eight models are applied including models of airplane, ant, bird, chair, fish, fourleg, glass and teddy as listed in *Table 6-3*. For segmenting models with hidden features, all the eight models are transferred into a data set based on the proposed method. As some triangles are not included in the data structure, the triangle usage rate is defined for the percentage of triangles used in the data structure. For these eight models, all of them have a triangle usage rate higher than 93%. Thus, the data structure can represent the original model for clustering.

To evaluate solutions of the increased surface quality and reduced support by the proposed method, structure values of WQ calculated by Eq. (6-7), WU calculated by Eq. (6-14) and WQ+WU in the segmented models based on features of surface quality and support structure are compared with values of the original model, and values of segmented models based on the geometric distance.

The AP clustering does not need to assign the number of clusters. It generates the number of clusters based on input data. Thus, numbers of clusters are different for models segmented with hidden features and the geometric distance. Numbers of clusters for eight models are listed in *Table 6-3*. *Figure 6-18* shows clustering results of the eight models. The first row is the original 3D models, the second row is

clustering results with the geometric distance, and the third row is clustering results by the proposed method.

Table 6-4 lists segmentation results of the airplane model with the geometric distance and hidden features. *Table 6-5* shows segmentation results of a fish model with the geometric distance and hidden features. The value of WQ + WU of the whole model is larger than the two segmented results. The value of WQ + WU for clustering results with hidden features is smaller than clustering results with the geometric distance. Thus, the proposed method using hidden features increases the surface quality and reduces the support structure.

Table 6-3 Data of 3D triangular mesh models

	Airplane	Ant	Bird	Chair	Fish	Fourleg	Glass	Teddy
No. of nodes	5400	6370	7849	84499	7121	8078	7016	10233
No. of edges	16194	19104	23541	25497	21357	24228	21042	30693
No. of triangles	10796	12736	15694	16998	14238	16152	14028	20462
No. of data structures	337	412	512	531	463	521	454	669
Triangles usage rate %	93.65	97.05	97.87	93.72	97.56	96.77	97.09	98.08
No. of cluster with geometric distance	5	13	5	4	11	16	5	15
No. of cluster with hidden features	5	14	5	4	12	15	6	17

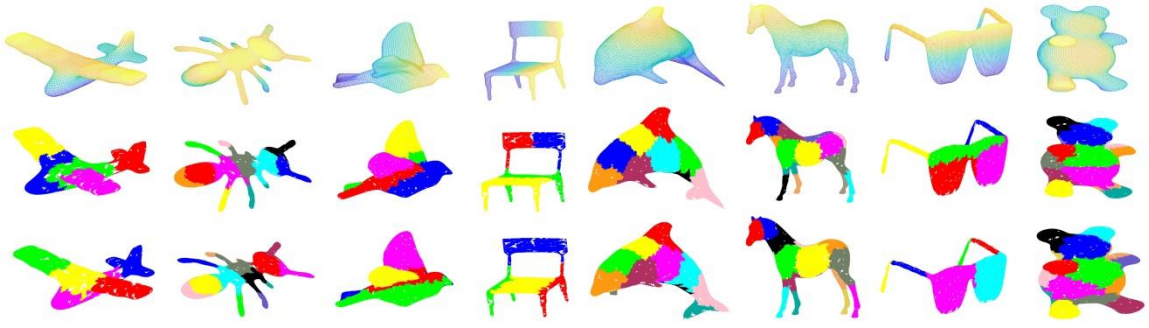

Figure 6-18 Clustering results of 3D triangular mesh models

Table 6-4 Segmentation results of the airplane model with the geometric distance and hidden features

Air plane		Clustering with geometric distance					Clustering with hidden features				
		No. of triangles	No. of sub-graphs	WQ	WU $\times 10^{-4}$	WQ + WU	No. of triangles	No. of sub-graphs	WQ	WU $\times 10^{-4}$	WQ + WU
Whole model		-	-	0.4948	468	0.5696	-	-	-	-	-
Segmented model	Part 1 red	1976	61	0.0988	16.9	0.1031	2305	71	0.0464	12.9	0.0486
	Part 2 yellow	2459	79	0.0540	8.56	0.0522	2326	72	0.0862	8.95	0.0908
	Part 3 green	2015	62	0.0820	4.72	0.0828	2238	72	0.0510	5.86	0.0522
	Part 4 blue	1879	60	0.0929	4.07	0.0933	1939	60	0.0988	16.9	0.1020
	Part 5 pink	2467	75	0.0484	12.9	0.0506	1988	62	0.0800	3.86	0.0804
	Total	10796	337	0.3761	47.15	0.382	10796	337	0.3624	48.47	0.374

Table 6-5 Segmentation results of the fish model with the geometric distance and hidden features

Fish		Clustering with geometric distance					Clustering with hidden features				
		No. of triangles	No. of sub-graphs	WQ	WU $\times 10^{-4}$	WQ+WU	No. of triangles	No. of sub-graphs	WQ	WU $\times 10^{-4}$	WQ+WU
Whole model		-	-	1.6851	402	1.7384	-	-	-	-	-
Segmented model	Part 1	1609	53	0.0637	1.38	0.0639	1729	57	0.0697	1.39	0.0699
	Part 2	1192	39	0.1243	193	0.1558	1222	40	0.1252	193	0.1568
	Part 3	1212	40	0.1189	80.3	0.1310	1090	36	0.1231	66.2	0.1352
	Part 4	1090	36	0.1231	66.1	0.1352	1053	34	0.0724	67.4	0.0809
	Part 5	1152	37	0.0858	102	0.0979	1037	34	0.0743	68.7	0.0817
	Part 6	1190	39	0.0906	118	0.1030	1097	36	0.0985	32.1	0.1021
	Part 7	1134	37	0.0946	1.43	0.0948	1161	38	0.1079	8.38	0.1119
	Part 8	1187	39	0.0417	1.59	0.0419	1259	41	0.0428	1.59	0.0430
	Part 9	1372	44	0.0444	1.22	0.0449	1308	42	0.0431	1.22	0.0436
	Part 10	1145	37	0.0687	3.06	0.0720	1206	39	0.0742	3.06	0.0776

	Part 11	1955	62	0.0888	7.09	0.0903	1325	43	0.0615	2.50	0.0618
	Part 12	-	-	-	-	-	751	23	0.0247	5.17	0.0254
	Total	14238	463	0.9446	575.2	1.0307	14238	463	0.9174	450.7	0.9899

6.5.2 Experiment

For further validation of the proposed method, the whole airplane model and segmented airplane model with hidden features are printed using the polylactic acid (PLA) material. The models are simulated to generate 3D layered results after using the common surface to segment the airplane model and design connectors of different parts. The printing direction is generated based on Eqs. (6-7) and (6-14) with the lowest value of WQ+WU. To show the surface quality clearly, the used printing layer thickness is 200μm. When reducing the thickness of each layer, the surface quality can be improved.

The removed support material of each part from the printed model is weighted by a scale. The support material of the whole model is 16.58g. The support materials of segmented parts are 0.5g, 2.35g, 0.48g, 1.92g, and 1.57g, respectively. Comparing with the whole model, the total weight of the support material for the segmented airplane model is 6.82g, lighter than 16.58g.

The outer surface of the printed whole model is deteriorated because of removing the support material. Thus, the support structure should be located at the unimportant position of the printed model. The support structure of the segmented model is on the inner surface without affecting the surface quality. Some stripes occur on the outer surface of the whole model because the surface is vertical to the printing direction. The distance between two stripes is 2 mm and the depth of a stripe is 0.2 mm as shown in *Figure 6-19* (a).

In the assembled printed model, the junction between two parts can cause a gap. The gap can be decreased by increasing the printing accuracy. For each part, most of the area of the outer surface is parallel to the printing direction for surface quality improvement. The surface quality of outer surfaces is increased compared to the whole printed model. The distance between two stripes is 0.2 mm and the depth of a

stripe is 0.1 mm, which is lower than the value of the whole model as shown in *Figure 6-19* (b).

Figure 6-19 Surface quality of printed models

6.6 Summary

In this Chapter, a spectral clustering-based shape segmentation method is proposed to improve surface quality and reduce support materials. In addition, a deep-learning based shape segmentation method is also proposed. The surface segmentation methods have been applied in case studies. Results show that the surface segmentation increases the surface quality and reduces the support structure.

Chapter 7 Conclusion and future work

7.1 Research summary

This research is motivated by bolus shaping. Problems of the existing bolus shaping in inefficiency and inaccuracy are transferred into three research problems as follows.

(1) Improvement of surface unfolding methods to reduce deformation and distortion of bolus shaping.

(2) Introduction of a model retrieval method to improve efficiency of bolus shaping.

(3) Optimization of the printing direction to increase surface quality and reduce support material of 3D printed models.

For the first research problem, surface segmentation methods are developed to increase the surface flattenability and unfolding accuracy by spectral clustering methods, where a Laplacian matrix is designed to combine the surface segmentation saliency and geometric information. Particle swarm optimizer (PSO) is applied to optimize surface flattenability and estimate air gap. The objective function and constraint are identified to search an optimal solution for the 3D surface model with a high flattenability. In addition, triangle crossings are introduced to reduce the number of iterations in the coordinate transformation process from 3D surfaces to 2D planes. A disturbing spring is added into the mass-spring model with crossed springs to improve the efficiency of contours optimization for unfolded 2D planes.

For the second research problem, an image-based model retrieval method is developed by comparing feature skeletons of the model and model image. Nodes in 3D mesh models are embedded into a feature space by the spectral analysis, and then formed into skeletons to build a skeleton base. Edges of the model images are detected using visual entropies, and segmented into the object and background pixels for contours of the object using spectral clustering. The image skeleton is compared with skeletons in the model skeleton base to find the best-matched 3D model using an iterative closest point method.

For the third research problem, a surface segmentation method is proposed based

on spectral clustering to improve surface quality and reduce support structures. A Laplacian matrix is designed to combine features of surface quality and support structure. The spectral clustering method clusters the surface model into several parts to improve surface quality and reduce support materials. To further improve the spectral clustering-based surface segmentation method, a deep learning-based method is proposed by combining printing features of surface quality, support structure and normal curvature. A 3D product triangular mesh model is pre-segmented into sub-graphs. A Stacked Auto-encoder (SAE) is trained to extract hidden features based on printing features. An Affinity Propagation (AP) clustering method is introduced to combine the hidden features and geometric information to segment a 3D shape model into several clusters for 3D printing.

7.2 Research contributions

For the surface unfolding improvement

- The developed spectral clustering method can efficiently improve accuracy of the surface segmentation by using a Laplacian matrix to combine the surface segmentation saliency and geometric information.
- Proposed four objectives and one constraint for a particle swarm optimizer (PSO) can effectively support the optimization of surface flattenability and air gaps estimation.
- A disturbing spring added into the mass-spring model with crossed springs can effectively meet the efficiency of surface unfolding by changing the shape of 2D planes.

For the model retrieval

- Feature skeletons of the image and model are proposed to improve the image-based 3D model retrieval method.
- Visual entropies are proposed to extract edges of images.
- Gaussian Mixture Model (GMM) is built to extract skeletons from features of both 3D model and 2D image.

For 3D printing

- The proposed similarity matrix can effectively include features of the surface quality and support material of a 3D printed model.
- The proposed area balance cut can efficiently split a graph based on the minimum loss function and maximum area in each sub-graph.
- The developed deep learning-based shape segmentation method can effectively extract hidden features and combine the geometric distance to increase the surface quality and reduce the support structure of 3D printed parts.

7.3 Future work

The image-based model retrieval method will be tested in different applications of bolus retrieving to further validate the method. Shapes and positions of the support structure will be optimized based on the 3D printing feature extraction and model segmentation to further improve the surface quality and reduce the support structure. The model segmentation method will be further evaluated using different models and materials.

Papers published during this research

Following papers relate to contents of Chapter 4.

1. Li, Rui., Peng, Qingjin., Ingleby, Harry., Sasaki, David. (2020). Improved bolus shaping accuracy using the surface segmentation and spectral clustering. International Journal of Modelling and Simulation, 1-12.
DOI: https://doi.org/10.1080/02286203.2020.1756036

2. Li, Rui., Peng, Qingjin., Ingleby, Harry., Sasaki, David. (2020). Improvement of flattenability using particle swarm optimizer for surface unfolding in bolus shaping. SN Applied Sciences, 2(9), 1-19. DOI: https://doi.org/10.1007/s42452-020-03330-9

3. Li, Rui., Peng, Qingjin., Ingleby, Harry., Sasaki, David. (2020). A Surface unfolding method for bolus shaping using the mass-spring model. Computer-aided design and application, 17(5), 979-992.
DOI: https://doi.org/10.14733/cadaps.2020.979-992

4. Li, Rui., Peng, Qingjin. (2022). Surface Segmentation based on Concave Region and Flattenability. Computer-aided design and application, 19(3), 561-574.
DOI: https://doi.org/ 10.14733/cadaps.2022.561-574

5. Li, Rui., Peng, Qingjin. (2021). 3D Shape Segmentation: A Review. Recent Patents on Engineering.15:1, Bentham Science Publisher.

Following paper relates to contents of Chapter 5.

6. 3D Model Retrieval based on Feature Skeletons of the Image and Model, Draft.

Following papers relate to contents of Chapter 6.

7. Li, Rui., Peng, Qingjin. (2020). Surface quality improvement and support material reduction in 3D printed shell products based on efficient spectral clustering. The International Journal of Advanced Manufacturing Technology, 107(9), 4273-4286.
DOI: https://doi.org/10.1007/s00170-020-05299-6

8. Li, Rui., Peng, Qingjin. (2021). Deep learning-based optimal segmentation of 3D printed product for surface quality improvement and support structure reduction.

Journal of Manufacturing Systems, 60, 252-264.
DOI: https://doi.org/10.1016/j.jmsy.2021.06.007

References

Akgul, C. B., Sankur, B., Schmitt, F., & Yemez, Y. (2007). Multivariate density-based 3D shape descriptors. In IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07) (pp. 3-12). IEEE. https://doi.org/10.1109/SMI.2007.27

Akgül, C. B., Sankur, B., Yemez, Y., & Schmitt, F. (2009). 3D model retrieval using probability density-based shape descriptors. IEEE transactions on pattern analysis and machine intelligence, 31(6), 1117-1133.

Aono, M., & Iwabuchi, H. (2012). 3D shape retrieval from a 2D image as query. In Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (pp. 1-10). IEEE.

Araújo, C., Cabiddu, D., Attene, M., Livesu, M., Vining, N., & Sheffer, A. (2019). Surface2Volume: Surface Segmentation Conforming Assemblable Volumetric Partition. arXiv preprint arXiv:1904.10213.

Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2010). Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5), 898-916.

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

Au, O. K. C., Zheng, Y., Chen, M., Xu, P., & Tai, C. L. (2011). Mesh segmentation with concavity-aware fields. IEEE Transactions on Visualization and Computer Graphics, 18(7), 1125-1134. https://doi.org/10.1109/TVCG.2011.131

Barra, V., & Biasotti, S. (2013). 3d shape retrieval using kernels on extended reeb graphs. Pattern Recognition, 46(11), 2985-2999. https://doi.org/10.1016/j.patcog.2013.03.019

Bell, S., & Bala, K. (2015). Learning visual similarity for product design with convolutional neural networks. ACM transactions on graphics (TOG), 34(4), 1-10. https://doi.org/10.1145/2766959

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. Advances in neural information processing systems, 19.

Burleson, S., Baker, J., Hsia, A. T., & Xu, Z. (2015). Use of 3D printers to create a patient‐specific 3D bolus for external beam therapy. Journal of applied clinical medical physics, 16(3), 166-178.

Canters, R. A., Lips, I. M., Wendling, M., Kusters, M., van Zeeland, M., Gerritsen, R. M., et al. (2016). Clinical implementation of 3D printing in the construction of patient specific bolus for electron beam radiotherapy for non-melanoma skin cancer. Radiotherapy and Oncology, 121(1), 148-153.

Chen, J., Zhu, H., & Li, X. (2016). Automatic extraction of discontinuity orientation from rock mass surface 3D point cloud. Computers & geosciences, 95, 18-31. https://doi.org/10.1016/j.cageo.2016.06.015

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep learning-based classification of hyperspectral data. IEEE Journal of Selected topics in applied earth observations and remote sensing, 7(6), 2094-2107.

Desbrun, M., Meyer, M., Schröder, P., & Barr, A. H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (pp. 317-324).

Driscoll, C. F., Taylor, M. A., & Ostrowski, J. S. (1992). Fabrication of bolus compensators used in the treatment of irregular tissue surfaces in radiation therapy. The Journal of prosthetic dentistry, 67(3), 370-374.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. science, 315(5814), 972-976.

Furuya, T., & Ohbuchi, R. (2009). Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features. In Proceedings of the ACM international conference on image and video retrieval (pp. 1-8). https://doi.org/10.1145/1646396.1646430

Gardan, N., & Schneider, A. (2015). Topological optimization of internal patterns and support in additive manufacturing. Journal of Manufacturing Systems, 37, 417-425. https://doi.org/10.1016/j.jmsy.2014.07.003

Giorgi, D., Biasotti, S., & Paraboschi, L. (2007). Shape retrieval contest 2007: Watertight models track. SHREC competition, 8(7)

Golovinskiy, A., & Funkhouser, T. (2008). Randomized cuts for 3D mesh analysis. In ACM transactions on graphics (TOG) (Vol. 27, No. 5, p. 145). ACM. https://doi.org/10.1145/1409060.1409098

Guo, X., Zhou, J., Zhang, W., Du, Z., Liu, C., & Liu, Y. (2017). Self-supporting structure design in additive manufacturing through explicit topology optimization. Computer Methods in Applied Mechanics and Engineering, 323, 27-63. https://doi.org/10.1016/j.cma.2017.05.003

Hao, J., Fang, L., & Williams, R. E. (2011). An efficient curvature‐based partitioning of large‐scale STL models. Rapid Prototyping Journal. https://doi.org/10.1108/13552541111113862

Hu, R., Li, H., Zhang, H., & Cohen-Or, D. (2014). Approximate pyramidal shape decomposition. ACM Trans. Graph., 33(6), 213-1.

Huang, H. Q., Mok, P. Y., Kwok, Y. L., & Au, J. S. (2012). Block pattern generation: From parameterizing human bodies to fit feature-aligned and flattenable 3D garments. Computers in Industry, 63(7), 680-691.

Humphries, S. M., Boyd, K., Cornish, P., & Newman, F. D. (1996). Comparison of Super Stuff and paraffin wax bolus in radiation therapy of irregular surfaces. Medical Dosimetry, 21(3), 155-157.

Ip, C. Y., Lapadat, D., Sieger, L., & Regli, W. C. (2002). Using shape distributions to compare solid models. In Proceedings of the seventh ACM symposium on Solid modeling and applications (pp. 273-280). https://doi.org/10.1145/566282.566322

Iyer, N., Kalyanaraman, Y., Lou, K., Jayanti, S., & Ramani, K. (2003). A

reconfigurable 3D engineering shape search system: Part I—Shape representation. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 36991, pp. 89-98).

Jiao, X., Wu, T., & Qin, X. (2018). Mesh segmentation by combining mesh saliency with spectral clustering. Journal of Computational and Applied Mathematics, 329, 134-146.

Kalogerakis, E., Hertzmann, A., & Singh, K. (2010). Learning 3D mesh segmentation and labeling. In ACM SIGGRAPH 2010 papers (pp. 1-12).

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

Kumar, C. R., & Suguna, S. (2016). Visual semantic based 3D video retrieval system using HDFS. Data mining and knowledge discovery, 10(8), 3806.

Laga, H., Takahashi, H., & Nakajima, M. (2006). Spherical wavelet descriptors for content-based 3D model retrieval. In IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06) (pp. 15-15). IEEE.

Lai, C. C. (2011). An improved SVD-based watermarking scheme using human visual characteristics. Optics Communications, 284(4), 938-944. https://doi.org/10.1016/j.optcom.2010.10.047

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. https://doi.org/10.1016/j.cad.2004.09.001

Lerma, J. L., & Biosca, J. M. (2005). Segmentation and filtering of laser scanner data for cultural heritage. In CIPA 2005 XX International Symposium (Vol. 26).

Li, B., & Johan, H. (2013). Sketch-based 3D model retrieval by incorporating 2D-3D alignment. Multimedia Tools and Applications, 65(3), 363-385.

Li, B., & Johan, H. (2013). 3D model retrieval using hybrid features and class information. Multimedia tools and applications, 62(3), 821-846.

https://doi.org/10.1007/s11042-011-0873-3

Li, B., Lu, Y., Godil, A., Schreck, T., Bustos, B., Ferreira, A., et al. (2014). A comparison of methods for sketch-based 3D shape retrieval. Computer Vision and Image Understanding, 119, 57-80.

Li, J., Zhang, D., Lu, G., Peng, Y., Wen, X., & Sakaguti, Y. (2005). Flattening triangulated surfaces using a mass-spring model. The International Journal of Advanced Manufacturing Technology, 25(1-2), 108-117. https://doi.org/10.1007/s00170-003-1818-4

Li, Y., Su, H., Qi, C. R., Fish, N., Cohen-Or, D., & Guibas, L. J. (2015). Joint embeddings of shapes and images via cnn image purification. ACM transactions on graphics (TOG), 34(6), 1-12. https://doi.org/10.1145/2816795.2818071

Liu, J., & To, A. C. (2017). Deposition path planning-integrated structural topology optimization for 3D additive manufacturing subject to self-support constraint. Computer-Aided Design, 91, 27-45. https://doi.org/10.1016/j.cad.2017.05.003

Liu, Q., Xi, J., & Wu, Z. (2013). An energy-based surface flattening method for flat pattern development of sheet metal components. The International Journal of Advanced Manufacturing Technology, 68(5-8), 1155-1166. https://doi.org/10.1007/s00170-013-4908-y

Liu, R., & Zhang, H. (2004). Segmentation of 3D meshes through spectral clustering. In 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings. (pp. 298-305). IEEE. https://doi.org/10.1109/PCCGA.2004.1348360

Liu, X., Li, S., Zheng, X., & Lin, M. (2016). Development of a flattening system for sheet metal with free-form surface. Advances in Mechanical Engineering, 8(2), 1687814016630517. https://doi.org/10.1177/1687814016630517

Lou, K., Jayanti, S., Iyer, N., Kalyanaraman, Y., Prabhakar, S., & Ramani, K. (2003). A reconfigurable 3d engineering shape search system: Part ii—database indexing, retrieval, and clustering. In International Design Engineering Technical Conferences

and Computers and Information in Engineering Conference (Vol. 36991, pp. 169-178).

Menzel, H. G. (2014). International commission on radiation units and measurements. Journal of the ICRU, 14(2), 1-2.

Miandarhoie, A., Khalili, K., & Mohammadinejad, H. (2017). CAD mesh models segmentation into swept surfaces. The International Journal of Advanced Manufacturing Technology, 92(9-12), 3659-3671. https://doi.org/10.1007/s00170-017-0437-4

Mitani, J., & Suzuki, H. (2004). Making papercraft toys from meshes using strip-based approximate unfolding. In ACM transactions on graphics (TOG) (Vol. 23, No. 3, pp. 259-263). ACM. https://doi.org/10.1145/1015706.1015711

Mao, W., Tian, S., Fan, J., Liang, X., & Safian, A. (2020). Online detection of bearing incipient fault with semi-supervised architecture and deep feature representation. Journal of Manufacturing Systems, 55, 179-198. https://doi.org/10.1016/j.jmsy.2020.03.005

Nie, W., Ren, M., Liu, A., Mao, Z., & Nie, J. (2020). M-GCN: Multi-branch graph convolution network for 2D image-based on 3D model retrieval. IEEE Transactions on Multimedia. https://doi.org/10.1109/TMM.2020.3006371

Nie, W., Wang, W., Liu, A., Nie, J., & Su, Y. (2019). HGAN: Holistic Generative Adversarial Networks for Two-dimensional Image-based Three-dimensional Object Retrieval. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 15(4), 1-24. https://doi.org/10.1145/3344684

Oropallo, W., & Piegl, L. A. (2016). Ten challenges in 3D printing. Engineering with Computers, 32(1), 135-148. https://doi.org/10.1007/s00366-015-0407-0

Osada, R., Funkhouser, T., Chazelle, B., & Dobkin, D. (2002). Shape distributions. ACM Transactions on Graphics (TOG), 21(4), 807-832. https://doi.org/10.1145/571647.571648

Pérez-Cruz, F. (2008). Kullback-Leibler divergence estimation of continuous distributions. In 2008 IEEE international symposium on information theory (pp. 1666-1670). IEEE.

Qin, F., Gao, S., Yang, X., Li, M., & Bai, J. (2016). An ontology-based semantic retrieval approach for heterogeneous 3D CAD models. Advanced Engineering Informatics, 30(4), 751-768.

Robar, J. L., Moran, K., Allan, J., Clancey, J., Joseph, T., Chytyk-Praznik, K., et al. (2018). Intrapatient study comparing 3D printed bolus versus standard vinyl gel sheet bolus for postmastectomy chest wall radiation therapy. Practical radiation oncology, 8(4), 221-229.

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

Shao, T., Xu, W., Yin, K., Wang, J., Zhou, K., & Guo, B. (2011). Discriminative sketch‐based 3d model retrieval via robust shape matching. In Computer Graphics Forum (Vol. 30, No. 7, pp. 2011-2020). Oxford, UK: Blackwell Publishing Ltd.

Shih, J. L., Lee, C. H., & Wang, J. T. (2007). A new 3D model retrieval approach based on the elevation descriptor. Pattern Recognition, 40(1), 283-295. https://doi.org/10.1016/j.patcog.2006.04.034

Shu, Z., Qi, C., Xin, S., Hu, C., Wang, L., Zhang, Y., & Liu, L. (2016). Unsupervised 3D shape segmentation and co-segmentation via deep learning. Computer Aided Geometric Design, 43, 39-52.

Smith, J., & Schaefer, S. (2015). Bijective parameterization with free boundaries. ACM Transactions on Graphics (TOG), 34(4), 70. https://doi.org/10.1145/2766947

Sroka, M., Reguła, J., & Łobodziec, W. (2010). The influence of the bolus-surface distance on the dose distribution in the build-up region. Reports of Practical Oncology and Radiotherapy, 15(6), 161-164.

Sundar, H., Silver, D., Gagvani, N., & Dickinson, S. (2003). Skeleton based shape

matching and retrieval. In 2003 Shape Modeling International. (pp. 130-139). IEEE.

Suykens, J. A., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. P. (2002). Least squares support vector machines. World scientific.

Takahashi, S., Wu, H. Y., Saw, S. H., Lin, C. C., & Yen, H. C. (2011). Optimized topological surgery for unfolding 3d meshes. In Computer Graphics Forum (Vol. 30, No. 7, pp. 2077-2086). Oxford, UK: Blackwell Publishing Ltd. https://doi.org/10.1111/j.1467-8659.2011.02053.x

Tangelder, J. W., & Veltkamp, R. C. (2008). A survey of content based 3D shape retrieval methods. Multimedia tools and applications, 39(3), 441-471. https://doi.org/10.1007/s11042-007-0181-0

Taubin, G. (1995, June). Estimating the tensor of curvature of a surface from a polyhedral approximation. In Proceedings of IEEE International Conference on Computer Vision (pp. 902-907). IEEE.

Theologou, P., Pratikakis, I., & Theoharis, T. (2015). A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation. Computer Vision and Image Understanding, 135, 49-82.

Tung, T., & Schmitt, F. (2008). Shrec'08 entry: Shape retrieval of noisy watertight models using amrg. In 2008 IEEE International Conference on Shape Modeling and Applications (pp. 229-230). IEEE. https://doi.org/10.1109/SMI.2008.4547981

Typke, R., Veltkamp, R. C., & Wiering, F. (2006). A measure for evaluating retrieval techniques based on partially ordered ground truth lists. In 2006 IEEE International Conference on Multimedia and Expo (pp. 1793-1796). IEEE.

Vanek, J., Galicia, J. A. G., & Benes, B. (2014). Clever support: Efficient support structure generation for digital fabrication. In Computer graphics forum (Vol. 33, No. 5, pp. 117-125).

Vyas, V., Palmer, L., Mudge, R., Jiang, R., Fleck, A., Schaly, B., et al. (2013). On bolus for megavoltage photon and electron radiation therapy. Medical Dosimetry,

38(3), 268-273.

Wang, C. C. (2008). Towards flattenable mesh surfaces. Computer-Aided Design, 40(1), 109-122.

Wang, C. C., Smith, S. S., & Yuen, M. M. (2002). Surface flattening based on energy model. Computer-Aided Design, 34(11), 823-833. https://doi.org/10.1016/S0010-4485(01)00150-6

Wang, H., Lu, T., Au, O. K. C., & Tai, C. L. (2014). Spectral 3D mesh segmentation with a novel single segmentation field. Graphical models, 76(5), 440-456. https://doi.org/10.1016/j.gmod.2014.04.009

Wang, J., & Yu, Z. (2011). Geometric decomposition of 3D surface meshes using Morse theory and region growing. The International Journal of Advanced Manufacturing Technology, 56(9-12), 1091-1103. https://doi.org/10.1007/s00170-011-3259-9

Wang, P., Gan, Y., Shui, P., Yu, F., Zhang, Y., Chen, S., & Sun, Z. (2018). 3D shape segmentation via shape fully convolutional networks. Computers & Graphics, 70, 128-139.

Wang, W. M., Zanni, C., & Kobbelt, L. (2016). Improved surface quality in 3D printing by optimizing the printing direction. In Computer graphics forum (Vol. 35, No. 2, pp. 59-70).

Xi, Z., Kim, Y. H., Kim, Y. J., & Lien, J. M. (2016). Learning to segment and unfold polyhedral mesh from failures. Computers & Graphics, 58, 139-149. https://doi.org/10.1016/j.cag.2016.05.022

Yao, M., Chen, Z., Luo, L., Wang, R., & Wang, H. (2015). Level-set-based partitioning and packing optimization of a printable model. ACM Transactions on Graphics (TOG), 34(6), 1-11. https://doi.org/10.1145/2816795.2818064

Ying, S., Peng, J., Du, S., & Qiao, H. (2009). A scale stretch method based on ICP for 3D data registration. IEEE Transactions on Automation Science and Engineering, 6(3),

559-565. https://doi.org/10.1109/TASE.2009.2021337

Zabalza, J., Ren, J., Zheng, J., Zhao, H., Qing, C., Yang, Z., et al. (2016). Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. Neurocomputing, 185, 1-10.

Zhang, D., Li, J., & Wang, J. (2018). Design Patterns of Soft Products Using Surface Flattening. Journal of Computing and Information Science in Engineering, 18(2), 021011.

Zhang, H., Wu, C., Deng, J., Liu, Z., & Yang, Y. (2018). A new two-stage mesh surface segmentation method. The Visual Computer, 34(11), 1597-1615. https://doi.org/10.1007/s00371-017-1434-1

Zhang, W., & Zhou, L. (2018). Topology optimization of self-supporting structures with polygon features for additive manufacturing. Computer Methods in Applied Mechanics and Engineering, 334, 56-78. https://doi.org/10.1016/j.cma.2018.01.037

Zhang, X., Le, X., Panotopoulou, A., Whiting, E., & Wang, C. C. (2015). Perceptual models of preference in 3D printing direction. ACM Transactions on Graphics (TOG), 34(6), 1-12. https://doi.org/10.1145/2816795.2818121

Zhang, Y., Luo, X., & Jia, J. (2019). A Compact Face-Based Topological Data Structure for Triangle Mesh Representation.

Zheng, Y., & Tai, C. L. (2010). Mesh decomposition with cross‐boundary brushes. In Computer Graphics Forum (Vol. 29, No. 2, pp. 527-535). Oxford, UK: Blackwell Publishing Ltd. https://doi.org/10.1111/j.1467-8659.2009.01622.x

Zheng, Y., Tai, C. L., & Au, O. K. C. (2011). Dot scissor: a single-click interface for mesh segmentation. IEEE transactions on visualization and computer graphics, 18(8), 1304-1312. https://doi.org/10.1109/TVCG.2011.140

Zou, W., Fisher, T., Zhang, M., Kim, L., Chen, T., Narra, V., et al. (2015). Potential of 3D printing technologies for fabrication of electron bolus and proton compensators. Journal of applied clinical medical physics, 16(3), 90-98.