Design Automation for Customized Apparel Products

Charlie C. L. Wang*

Department of Automation and Computer-Aided Engineering, Chinese University of Hong Kong,

Shatin, N.T., Hong Kong, P. R. China

Yu Wang **Matthew M. F. Yuen**

Department of Mechanical Engineering, Hong Kong University of Science and Technology,

Clear Water Bay, N.T., Hong Kong, P. R. China

Abstract

This paper presents solution techniques for a three-dimensional *Automatic Made-to-Measure* (AMM) scheme for apparel products. Freeform surface is adopted to represent the complex geometry models of apparel products. When designing the complex surface of an apparel product, abstractions are stored in conjunction with the models using a non-manifold data structure. Apparel products are essentially designed with reference to human body features, and thus share a common set of features as the human model. Therefore, the parametric feature-based modeling enables the automatic generation of fitted garments on differing body shapes. In our approach, different apparel products are each represented by a specific feature template preserving its individual characteristics and styling. When the specific feature template is encoded as the equivalent human body feature template, it automates the generation of made-to-measure apparel products. The encoding process is performed in 3D, which fundamentally solves the fitting problems of the 2D tailoring and pattern-making process. This paper gives an integrated solution scheme all above problems. In detail, a non-manifold data structure, a constructive design method, four freeform modification tools, and a detail template encoding/decoding method are developed for the design automation of customized apparel products.

Keywords: automation, made-to-measure, fitting, apparel products, and three-dimensional solution.

* Corresponding Author: Charlie C. L. Wang; E-mail: cwang@acae.cuhk.edu.hk; Tel: (852) 2609-8052; Fax: (852) 2603-6002

1. Introduction

The purpose of this paper is to provide an integrated solution for the design automation of customized apparel products; in other words, *Automatic Made-to-Measure* (AMM) technology. After designing a garment on a standard size mannequin, AMM automatic generates the same garment styling to be fitted on different body shapes and guarantees the three-dimensional fitting. This greatly improves the efficiency of pattern generation for the apparel industry. At present, 2D CAD systems are widely used in the cloth industry for pattern design and generation. However, this process remains a bottleneck in garment manufacturing, especially when the garment patterns of the same style are graded using empirical two-dimensional grading rules to fit different body shapes. Current commercial garment CAD systems [1-2] provide such 2D grading tools to generate patterns of different sizes from the basic pattern set. Their 2D grading rules are usually offset curves generated in plane; therefore it is not intuitive to preserve the fit of final dressing in spatial space. 2D approaches can hardly generate the fitted clothes for different human bodies. The only way to fundamentally solve the fitting problem of clothes is to design products in 3D. In the *DresssingSim* [3] solutions, some 3D design tools are provides. However, they remain relatively simple operations.

Cloth simulation techniques [4-10] provide a way of testing patterns by assembling 2D patterns in a computer system and draping them on a virtual human body. However, the tools for modifying the shape of patterns according to different human body shapes are not available. Recently, an online made-to-measure system was presented by Cordier et al. [11]. Their system allows interactive adjustment of the 3D mannequin according to the shopper's body measurements, online resizing of the garment to fit the mannequin, and real-time simulation of the garment corresponding to the body motion. However, their approach is also based on 2D pattern design and draping simulation; and their major purpose is for visualization and animation purpose, not for design. Our approach provides tools to construct and modify patterns directly on virtual human bodies in 3D space with focus on solving the 3D fitting problem. The patterns for a garment product with a particular designer styling are represented as a feature template. The easing relationships between the feature template and the feature-based human body model are encoded and stored. When the same garment product is required for a customer with different human shape, a decoding process is performed to reconstruct the 3D cloth patterns preserving the easing relationships related to the human model. Fig. 1 gives an illustration of this concept by using four human models H, H_1, H_2, and H_3. Using human model H as the reference for generating the 3D cloth pattern using the constructive design approach, the design related feature template is encoded. The same designer styled garment product can be generated automatically on H_1, H_2 and H_3. Based on the constructive

design result, we can further modify its freeform surfaces to create more complex styles. By detail template encoding/decoding, the modified results can also be regenerated automatically on different body shapes.

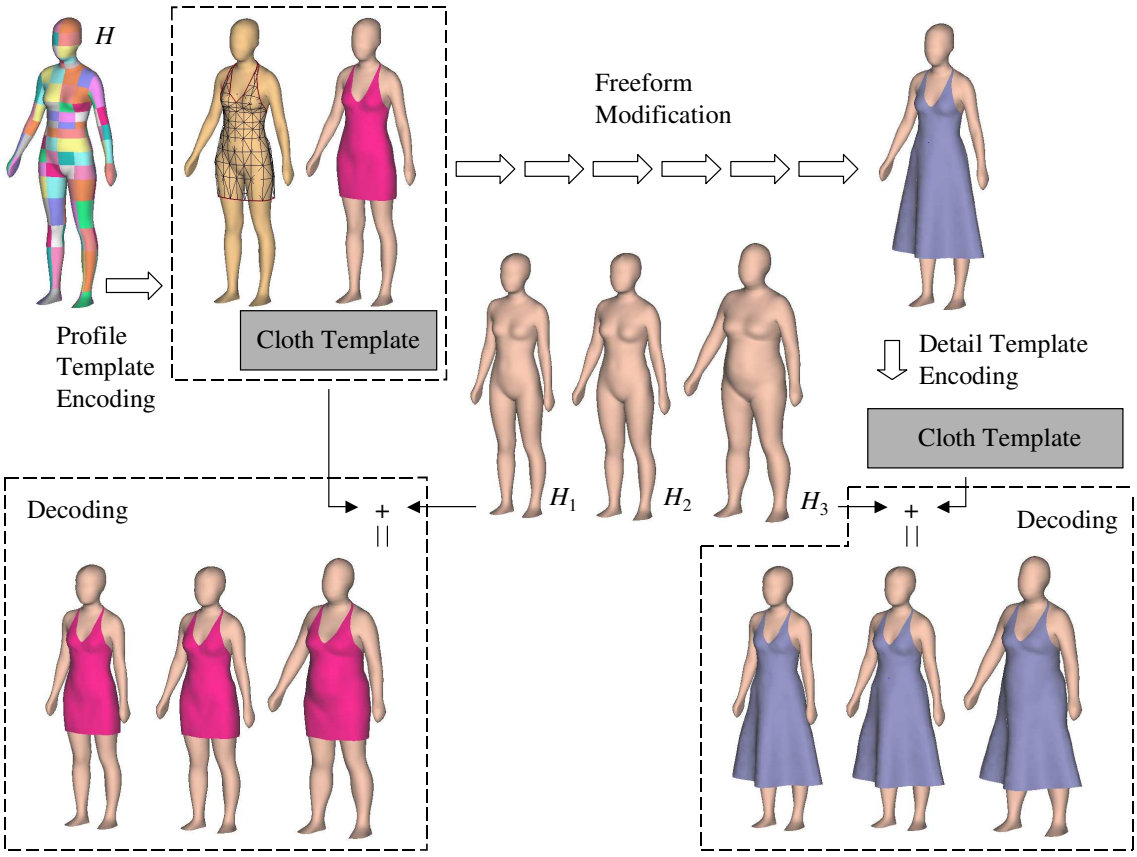


Fig. 1 The process of encoding and decoding cloth template on parameterized human bodies

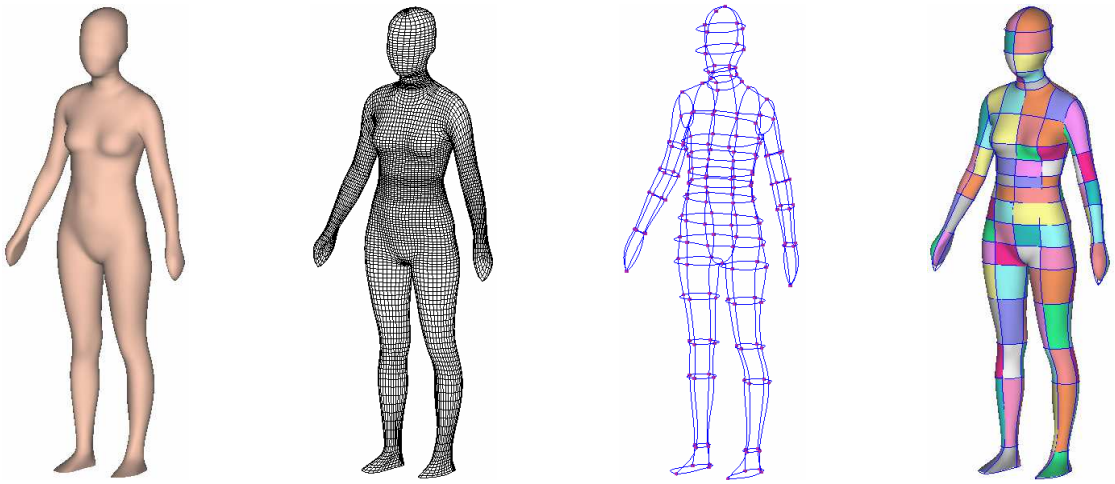
The garment products are represented by freeform surfaces because of their geometric complexity. To simplify the complex surface representation of garment products, during the design phase, abstractions are stored along with the model, which lead to the use of non-manifold data structure and operators. Garment design has its own characteristic: garments are related to and encompass human bodies, and therefore share the same set of features. Thus, the feature-based modeling enables the automatic construction and fitting of garment products on differing human body shapes. Each garment product is represented by a unique feature template in our approach. The major contribution of this paper is 1) an integrated solution scheme is presented for generating *automatic made-to-measure* (AMM) apparel products, where the products designed on one human body can be automatically regenerated on different body shapes; 2) detail techniques for supporting AMM, including a non-manifold data structure, a constructive design method, four freeform modification tools, and a detail template encoding/decoding method, are developed.

The rest of the paper are organized as follows. After reviewing related works in section 2, section 3 gives the necessary non-manifold data structure and its related operators to manipulate complex surfaces. Based on this data structure, the profile template encoding/decoding method is introduced in section 4 so that a garment product can be stored as 3D cloth pattern template and regenerated on different human body shapes. Freeform modification is usually required to change the style of clothes – section 5 presents four tools supporting the freeform modifications of apparel products. Also, designers may want to automatically regenerate the freeform modification result on different bodies, so we develop a detail template encoding/decoding technique in section 6 to convert a modified freeform surface into an apparel product template.

2. Related Works

In the area of computer graphics, Terzopoulos et al. [12] were the first to develop a physical model for cloth simulation. Volino et al. [4] developed a cloth model based on elasticity theory and used a Newtonian formulation instead of a Lagrangian formulation. Fan et al. [5] also presented a cloth simulation system on elasticity theory. Since Baraff and Witkin [6] introduced the semi-implicit method, it has become a popular technique for numerically solving the equations of motion in cloth simulation. The aforementioned physical models were found to give fairly realistic cloth motion; however Choi and Ko [7] revealed that those models suffer from a post-buckling instability that can be particularly problematic in wrinkle formation. This instability is an inherent physical instability and is therefore independent of the numerical method employed. Noting that the buckling behavior of cloth differs from that of other thin materials, Choi and Ko assumed that application of a compressive force on cloth immediately initiates buckling rather than compression. Collision detection and responds takes an important role in the cloth simulation since such simulation usually runs around human bodies. Zhang et al [8] has provided a multilevel temporal coherence collision detection technique for both object collision and self-collision detection. Bridson et al. [9] proposed a robust collision handling technique that combines repulsive forces, geometric treatment of collisions, and rigid impact zones. The combination of those techniques made the collision resolution process run very efficiently. They additionally presented a subdivision scheme that avoids collisions during the subdivision steps. Recently, in [10], Bridson et al. provided mixed implicit/explicit modeling techniques to produce a cloth simulation with many folds and wrinkles to improve the realism. In the above techniques, no matter how real the simulation result is, the patterns of clothes are unchanged. They can only be utilized to test patterns, but not to modify patterns which is the purpose of our approach.

In our approach, every human model must be parameterized based on a set of features. A lot of researches developed several parameterization algorithms for human models. The human body modeling methodologies in literature can be classified into the creative approaches and the reconstructive approaches. Anatomically based modelers [13, 14] can simulate underlying muscles, bones, and generalized tissue. They fall into the creative category of human modeling approaches. The interactive design is allowed in the anatomy-based modelers; however, these modelers require a relatively slow production time. A lot of the reconstruction approaches has been investigated to build 3D geometry of human model automatically by capturing existing shape [15-22]. Here, we adopt the parametric feature-based human model as proposed in [21, 22]. The feature-based human model H by the method of [21] gives a point-to-point correspondence among a set of human models with a same common topological structure for the defined features. Every human model is a feature-based model, which contains not only feature vertices and curves but also feature patches. Thus, the cloth pattern template can be encoded on the different levels of feature entity defined on the human model which include vertices, curves, and patches (e.g. Fig 2).



(a) parameterized human body **(b) mesh representation** **(c) feature curves and vertices** **(d) checkerboard to verify feature patches**

Fig. 2 An example parameterized feature-based human body

As mentioned above, the non-manifold data structure and related operators are utilized in our approach. In previous literature, B-rep based data structures [23-25] usually assigned respectively the *face-use*, *loop-use*, *edge-use*, and *vertex-use* topological entities in association with the *face*, *loop*, *edge*, and *vertex* entities. A more recent research on the representation of non-manifold models was the Partial Entity Structure (PES) [26], which used a compact non-manifold boundary representation. The storage size of the PES is reduced to half of the radial edge structure (RES). However, incomplete boundaries cannot be represented in this data structure, which is vital to maintain in a product's conceptual design. Some other approaches are complex-based representation

[27-29]. Since the complex-based data structure, unlike the afore-mentioned data structure, is based on a simple incidence graph that has no ordering information, it does not enable easy computation of certain important properties (orientability, for instance). The data structure adopted here (derived from [30]) is a combination of the boundary representation and the complex-based representation, which can overcome the above inadequacies.

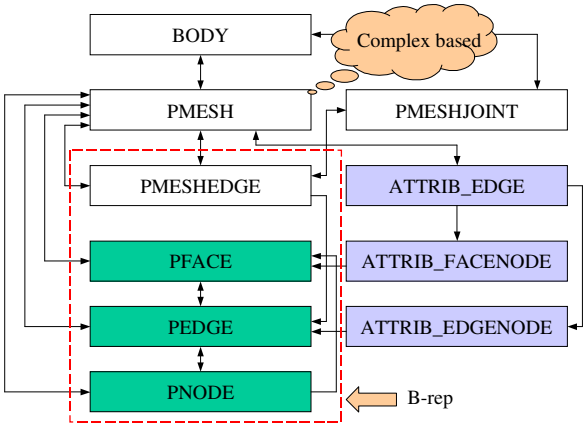
Many freeform modeling approaches have also been developed. Some of them are related to surface construction, some are interactive modification methods, and others are deformation techniques. Meyers et al.'s work [31] was concerned with the problem of reconstructing the surfaces of three-dimensional objects, given a collection of planar contours representing cross-sections through the objects. Bruyns et al. [32] developed a method that allows the user to directly sketch the desired cut contour on a three-dimensional surface in a manner assimilating the steering of scissors through fabric. Teddy system [33] can extrude a mesh to form a new closed mesh surface, and it can smooth a surrounded area by projecting this area onto a plane, triangulating the projected area using the constrained Delaunary triangulation algorithm, and finally dragging and pulling the new vertices in the surrounded area. The freeform modification part of our paper borrows some ideas from above techniques. Free-form deformations (FFDs) [34] and its variants [35-38] were popular and provide a high level of geometric control over the deformation. FFDs are useful for coarse-scale deformations but not finer-scale deformations, even if a very dense lattice or customized lattice shape is defined. Our detail template encoding/decoding method is indeed a deformation technique. The underlying technique is akin to the most recent t-FFD approach [39], which adopts the triangles as deformation primitives. Here, the polygonal facets of human bodies are utilized as deformation drivers.

3. Non-manifold Data Structure and Operators

In order to integrate the representation of geometric abstractions and the incomplete topological information, a non-manifold data structure for geometric object modeling by polygonal meshes is presented in this section. This data structure is a more generic version of the non-manifold data structure for triangular meshes presented in [30]. The proposed data structure is actually a hybrid of the boundary representation data structure and the complex-based data structure. Fig. 3 depicts the framework of our data structure.

The data structure can globally be considered as a tree with BODY as the root. A BODY has a collection of PMESHs, each of which contains complexes of a number of PMESHEDGEs, PFACEs, PEDGEs, and PNODEs; and a BODY also has a collection of PMESHJOINTs, each of which is related to some ordered PMESHEDGEs. A PFACE has *n* PEDGEs, and every PEDGE is a line segment ended by two PNODEs. Each PMESHEDGE has a collection of PEDGEs; and each PEDGE has its own direction flag in the PMESHEDGE. The adjacency

information of PMESHs at some PMESHEDGEs is stored in a new entity – PMESHJOINT. Each PMESHJOINT has a collection of PMESHEDGEs, which contain the same number of PEDGEs, and the PEDGEs are one to one connected (as shown in Fig. 4). If a PMESHEDGE is in the same direction with a PMESHJOINT, it is defined as a positive one in the PMESHJOINT; otherwise, it is defined as a negative one. The PMESHEDGEs in a PMESHJOINT are stored in a clockwise order by the right-hand rule (as shown in Fig. 4; where M1, M2, and M3 are three PMESHs, E1, E2, and E3 are their related PMESHEDGEs which contain the same number of PEDGEs, and EO is the PMESHJOINT containing the adjacent information). The detail description of each entity is shown in Appendix A. Using the data structure, it is easy to carry out any topological and geometrical manipulation on the manifold or non-manifold polygonal mesh models.



BODY	entire model complex	PMESH	mesh surface
PMESHEDGE	mesh surface edge	PFACE	polygonal face
PMESHJOINT	mesh surfaces joint	PEDGE	polygonal face edge
ATTRIB_EDGE	curves on mesh surface	PNODE	polygonal face node
ATTRIB_EDGENODE	nodes on polygonal edge	ATTRIB_FACENODE	nodes on polygonal face

Fig. 3 Non-manifold data structure frame

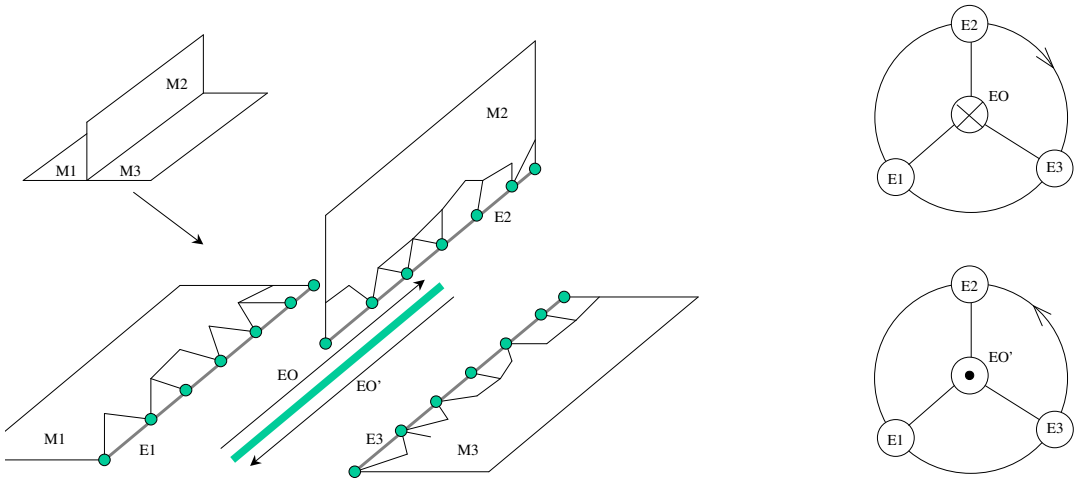


Fig. 4 Clockwise list of PMESHEDGEs in a PMESHJOINT

We define four attributes in our data structures. They include ATTRIB_NODE, ATTRIB_EDGE, ATTRIB_EDGENODE, and ATTRIB_FACENODE, where ATTRIB_EDGENODE and ATTRIB_FACENODE are derived from ATTRIB_NODE. ATTRIB_EDGENODE is the attribute node on a PEDGE, and ATTRIB_FACENODE is the attribute node in a PFACE. Their coordinates depend on the position of PEDGE's nodes or the position of PFACE's nodes. In detail, the coordinate of an ATTRIB_EDGENODE is represented by a parameter u related to the nodes of a PEDGE; and the coordinate of an ATTRIB_FACENODE is represented by (u,v) – the local coordinate of a PFACE. An ATTRIB_EDGE is an ordered collection of ATTRIB_NODEs, which can be either ATTRIB_EDGENODEs or ATTRIB_FACENODEs. The detail description of each attribute is shown in Appendix A.

The construction of a valid geometric model is achieved through the use of a proper set of topological operators. In geometric modeling, the fundamental topological operators are Euler operators [24, 40] that are consistent with the *Euler-Poincarè formula*. Likewise, the extended topological operators for non-manifold geometric modeling have to satisfy the same formula. Since no volume is included in our approach, only eight extended Euler operators are utilized (shown in Appendix B), and they are restricted to the polygonal meshes. When editing a model, often several repeated sequences of the extended Euler operators are used. These sequences are formulated as high level editing operations. Five of these sequences are formulated as high level editing operations, these include *edge collapse*, *edge split*, *edge swap*, *face split*, and *face triangulation*. These high level operators are provided to automate the performance of the extended Euler operator sequences and increase the efficiency of topological operations, and they are frequently used in polygonal mesh processing algorithms. The detail description of these operators and their sequences of extended Euler operators can be found in [30].

4. Constructive Design: Profile Template Encoding/Decoding

Based on the non-manifold data structure presented in the above section, and the parameterized feature-based human model representation described in [21], a constructive design technique is developed in this section. By this method, we can construct different feature-based garment profile templates, each representing a different garment product styling, in direct correspondence to feature vertices, curves and patches of a parameterized feature-based human model by incorporating the easing relationships. Then when applying to different body shape, by our decoding method, the easing relationships preserving garment product will be

generated. We can also control the final shape on edges through changing the profile curves on the encoded templates.

Feature-based profile template encoding

When building the feature-based profile template T_P for a garment product, two steps of interactivities are involved. Every feature node in the profile template should first be encoded in relationship to either the feature-based parameterized human model or other nodes that have already been encoded in T_P. After all feature nodes in the profile template are encoded, the topological graph linking the nodes should be interactively input by users. The processes of profile template encoding more or less like using the interactive tools to build a coarse freeform surface.

When encoding a feature node on a parameterized human model H, which is also represented by a feature-based polygonal mesh, the feature node can be encoded on a vertex, an edge, or a face on the mesh of human model – they are called *reference elements*. Every feature node is first created by specifying its (*x, y, z*) coordinate. One then can choose the encoding mode: 1) by vertex, 2) by edge, or 2) by face. When the mode is chosen, one could pick the reference element, which one wants to encode the feature node, on the surface of H. Then, the relationship between the feature node and a human model is encoded. The relationship is actually the relative coordinate of the feature nodes on the selected element. A feature node V_{T_P} in T_P can be exactly encoded on a feature node of H in the vertex mode, be encoded on a feature curve of H in the edge mode (since each feature curve is actually a set of linked edges), and be encoded on a feature patch by a face of the patch. The detail encoding processes are as follows.

In the vertex-encoding mode, if a vertex V_b is selected as the reference element, it is easy to obtain the unit normal vector n_{V_b} at the vertex; getting the first vertex V_b^1 adjacent to V_b in its adjacent vertices list, we can have

$$X_V = n_{V_b},\ Y_V = \frac{V_b^1 - V_b}{\left\| V_b^1 - V_b \right\|} \times n_{V_b},\ Z_V = X_V \times Y_V . \tag{1}$$

If Y_V is degenerated, by $V_b^1 - V_b = 0$ or $(V_b^1 - V_b) // n_{V_b}$, we can replace V_b^1 by V_b^2 in (1). Thus, when a feature node V_{T_P} in T_P is encoded on V_b, the encoded information includes the index of V_b on H, and the scalars of $u_V = (V_{T_P} - V_b) \cdot X_V$, $v_V = (V_{T_P} - V_b) \cdot Y_V$, and $w_V = (V_{T_P} - V_b) \cdot Z_V$. In the edge-encoding mode, an edge E_b serves as the reference element; the normal n_{E_b} at E_b is usually computed by averaging the normal

vectors of its left and right faces. After determining the nearest point V_{E_b} on E_b to V_{T_P}, a local frame at V_{E_b} can be determined by

$$X_E = n_{E_b},\ Y_E = t_{e_b} \Big/ \left\| t_{e_b} \right\|,\ Z_E = X_V \times Y_V, \tag{2}$$

where t_{e_b} is the direction vector of edge E_b. Since the normals on E_b's left and right faces are perpendicular to v_{e_b}, they form a plane perpendicular to E_b; n_{E_b} is on the plane, so $n_{E_b} \perp v_{e_b}$. The encoded information in the edge mode includes the index of E_b on H, the parameter t of V_{E_b} on E_b ($t \in [0,1]$), the scalars of $u_E = (V_{T_P} - V_{E_b}) \cdot X_E$, $v_E = (V_{T_P} - V_{E_b}) \cdot Y_E$, and $w_E = (V_{T_P} - V_{E_b}) \cdot Z_E$. In the face-encoding mode, after a face F_b is chosen to be a reference element, its centroid V_{F_b} and its normal n_{F_b} can be easily determined. Getting the first vertex $V_{F_b}^1$ in the vertices list of F_b, the local frame at V_{F_b} is as

$$X_F = n_{F_b},\ Y_F = \frac{V_{F_b}^1 - V_{F_b}}{\left\| V_{F_b}^1 - V_{F_b} \right\|},\ Z_F = X_F \times Y_F. \tag{3}$$

Thus, the encoded information includes the index of F_b on H, and the scalars of $u_F = (V_{T_P} - V_{F_b}) \cdot X_F$, $v_F = (V_{T_P} - V_{F_b}) \cdot Y_F$, and $w_F = (V_{T_P} - V_{F_b}) \cdot Z_F$.

When encoding a feature node V_{T_P} in T_P by other nodes in T_P, we have 1) one-to-one mode and 2) *n*-to-one mode defined. The one-to-one mode encode V_{T_P} on another feature node $V_{T_P}^* \in T_P$ by storing the information of $V_{T_P}^*$'s index in H and the vector $\delta = V_{T_P} - V_{T_P}^*$. The *n*-to-one mode has more than one feature nodes in T_P to determine the position of V_{T_P}, they are $V_{T_P}^i$ (i = 1, …, n). We encode them by the indexes of $V_{T_P}^i$ s, the vectors $\delta_i = V_{T_P} - V_{T_P}^i$, and the weights ω_i corresponding to the ith $V_{T_P}^i$. The weights determine the contribution of every $V_{T_P}^i$ on the final position of V_{T_P}. It is computed by

$$\omega_i = \frac{1}{\varepsilon + \left\| V_{T_P} - V_{T_P}^i \right\|^3}, \tag{4}$$

where $\varepsilon = 10^{-8}$ to avoid the singularity when $\left\| V_{T_P} - V_{T_P}^i \right\| = 0$. After the positions of $V_{T_P}^i$ s are changed to $\tilde{V}_{T_P}^i$, the new position of V_{T_P} can be determined by

$$\tilde{V}_{T_P} = \frac{\sum_i \omega_i (\delta_i + \tilde{V}_{T_P}^i)}{\sum_i \omega_i}. \tag{5}$$

It is not hard to prove that $\tilde{V}_{T_P} = V_{T_P}$ when $\tilde{V}^i_{T_P} = V^i_{T_P}$.

After all the feature nodes in T_P are encoded, they need to be linked with edges and faces using interactive tools (i.e., specifying the topological graph T_P^f of T_P interactively). The topological graph is a collection of PMESHs that are connected by PMESHJOINTs. For example, Fig. 5a shows the topological graph of a T_P stored in a BODY. Since our data structure is complex-based, the incomplete topology information generated by the construction process is easily stored. Fig. 5b-5h shows some fragments of the construction process of the topological graph. In Fig. 5b, the feature nodes have been defined around the reference model. An interactive tool is utilized to connect the feature nodes by edges as shown in Fig. 5c-5d; and Fig. 5e shows the feature template after creating all edges. Polygonal faces can also be created one by one interactively (Fig. 5f and 5g). Fig. 5h shows the final result.

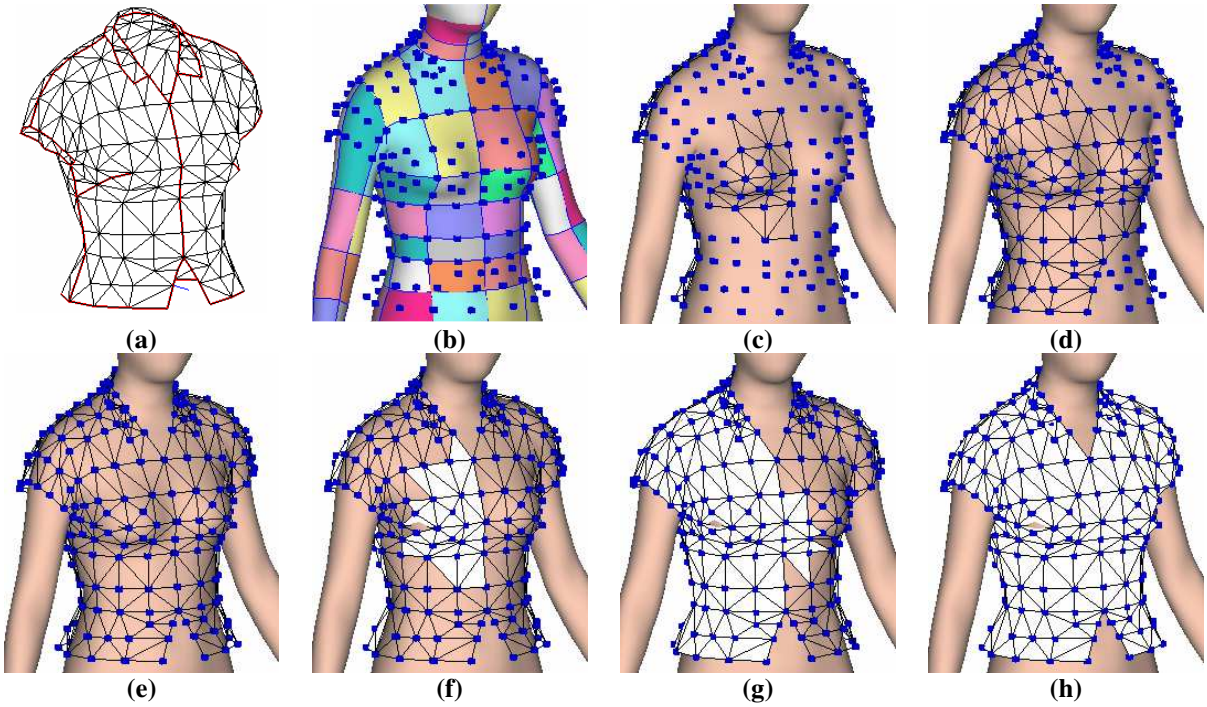


Fig. 5 An example of topological graph construction process

Using profile curves to control shape

In the above encoded T_P, the shape on an edge is not controlled. Here, we use the profile curves to control the shape of final surfaces. A profile curve is a parametric curve $C_p(u)$ on a PEDGE E_p in T_P, where $C_p(0)$ and $C_p(1)$ are coincident to the two ending vertices on E_p. Typically, it is represented as a 4^{th} order Bézier curves in our prototype system. The curve $C_p(u)$ can be specified by the traditional 3D curve input methods in

CAD systems or by the sketched-input as shown in [41]. The following profile template decoding process will persuade the final refined surface interpolating the specified profile curves. Thus, the profile curves are utilized to control local shape of the final surface. A profile curve $C_p(u)$ is encoded on its PEDGE by

$$ep(u) = C_p(u) - E_p(u). \tag{6}$$

After encoding the polynomial of $ep(u)$, the relationship between $C_p(u)$ and E_p is stored. When the positions of the endpoints of E_p are adjusted, we can construct a new parametric line segment $E_p^*(u)$ for it. Thus, the new profile curve can be obtained by

$$C_p^*(u) = ep(u) + E_p^*(u). \tag{7}$$

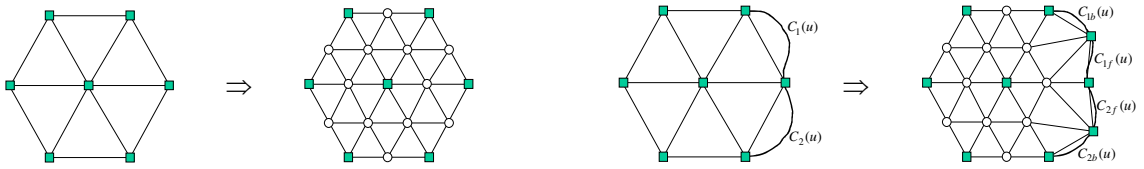
Feature-based profile template decoding

After a feature-based profile template T_P has been encoded, it can be applied to any parameterized feature-based human model H*. This involves the profile template decoding process. A decoding process includes the steps of relocating the positions of feature nodes, reshaping the profile curves, and surface refinement.

For a feature node V_{T_P} encoded on a vertex, by the encoded vertex index, the vertex V_b^* on H* is determined. A local frame can be established at V_b^* as given by the formula in eq.(1). By the stored (u_V, v_V, w_V) and the reconstructed local frame at V_b^* on H*, it is straightforward to determine the new position of V_{T_P}. For a feature node encoded on an edge, by index and *t*, the edge E_b^* on H* and the vertex $V_{E_b}^*$ on E_b^* are easily to be obtained. After the local frame at $V_{E_b}^*$ is determined by eq.(2), the new position of V_{T_P} is obtained using (u_E, v_E, w_E). Also, for the feature nodes encoded on faces, we relocate their position in the same way: calculate the centroid of the face on H*; computer the local frame at the centroid by eq.(3); finally relocate the feature node by the stored (u_F, v_F, w_F). If a feature node is encoded on another feature node $V_{T_P}^*$, its new position is $\delta + V_{T_P}^*$ with $V_{T_P}^*$ providing its position on H*. If a feature node V_{T_P} is encode on *n* other feature nodes in T_P, applying their new positions in eq.(5) gives the decoding result of V_{T_P}. After all feature nodes have been relocated, the new parametric representation of the profile curves are computed by eq.(7). They and the feature nodes are interpolated during the following surface refinement.

The surface refinement step of profile template decoding is to provide detail and smooth freeform surfaces for representing the shape of an apparel product in computer system. The method applied here is from [41] – the

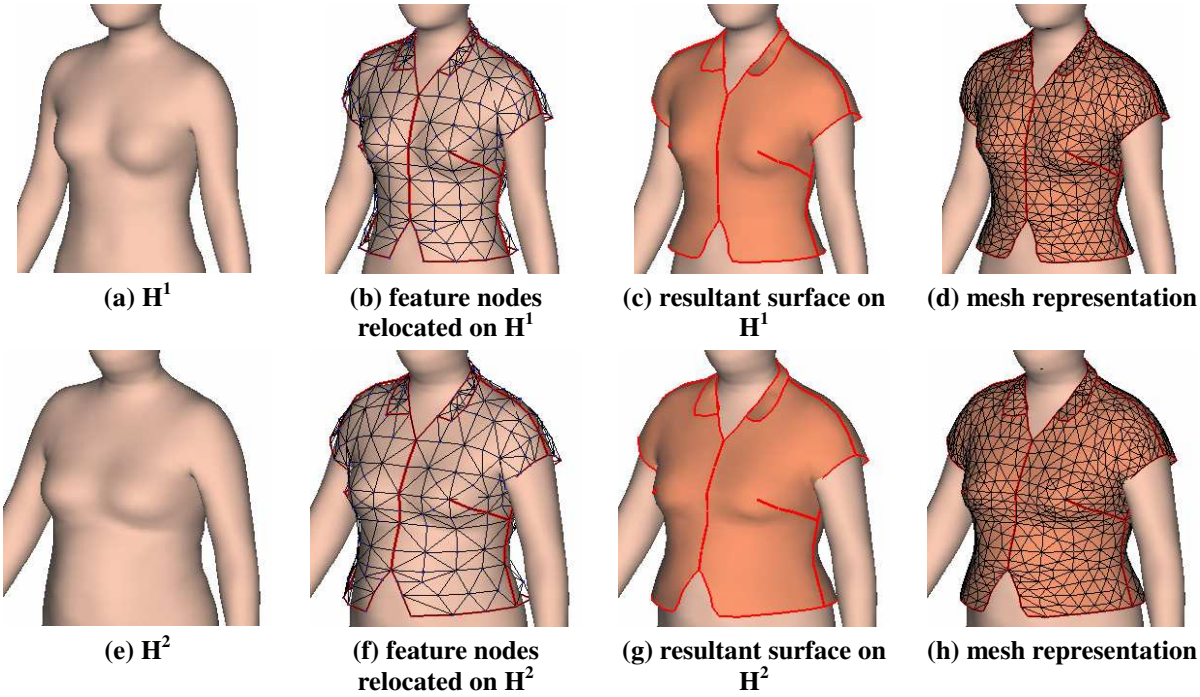
modified variational subdivision scheme, which iteratively applies a topological splitting operator to introduce new vertices to increase the degrees of freedom, followed by a discrete fairing operator to increase the overall smoothness. The constructed mesh surfaces interpolate not only the initial vertices but also the specified profiles. The topological splitting operator inserts new control vertices into the mesh. The split operation is chosen to be uniform so that all the new vertices are regular (valance is equal to 6, as shown in Fig. 6a). If non-triangular faces are involved, we just simply triangulate them before refinement. When inserting a new vertex on a edge, its position is located at the middle of the edge if there is no profile curve attached; if a profile curve $C_p(u)$ is attached on this edge, the inserted vertex is located at $C_p(0.5)$. The curve $C_p(u)$ is divided into two curves and attached on the split two edges (see Fig. 6b). The smoothing operator moves the control vertices according to the weighted averages of neighboring vertices. The positions of vertices in the refined mesh are changed to achieve a global energy functional minimization. Here, we implement the 2^{nd} order umbrella operator as an iterative solver of the problem [42]. In order to guarantee that the resultant fine mesh interpolates the originally given vertices, the umbrella operator must not be applied to those vertices belonging to the initial mesh. Also in order to guarantee that the resultant fine mesh interpolates the 3D profiles, the umbrella operator must not update the positions of the vertices lying on the profiles. As mentioned in [41], collision detection should also be incorporated to prevent vertices moving inside human models.



(a) without profile interpolation **(b) with profile interpolation**

Fig. 6 Topological splitting operator

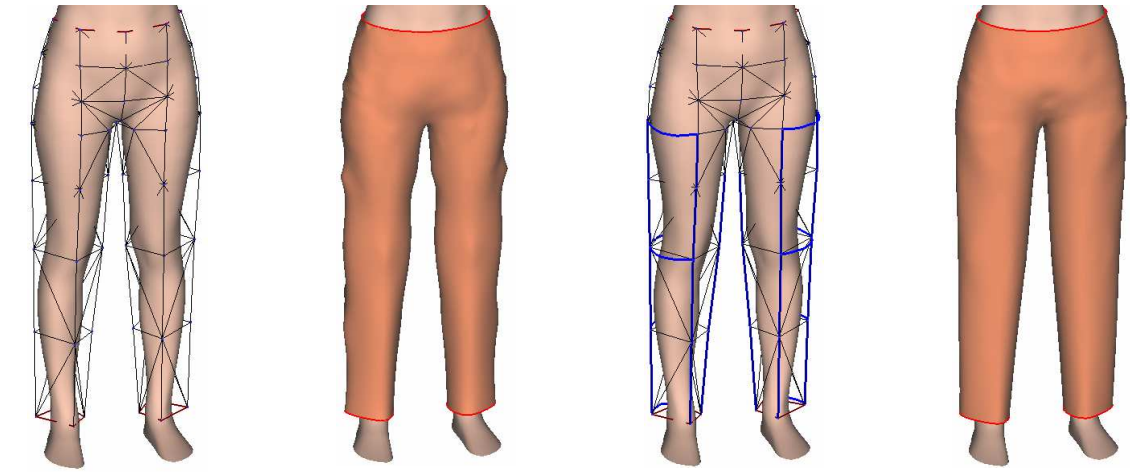
Fig. 7 shows the profile template decoding results on different human models. Fig. 7a and 7e are the new human models H^1 and H^2; Fig. 7b and 7f show the decoding processing with all feature nodes relocated; Fig. 7c and 7g give the final surface of the garment product on human models generated by the modified variational subdivision scheme, and Fig. 7d and 7h are related mesh representations. Fig. 8 explains how the profile curves (blue curves in the figure) control the final shape of refined surfaces. Fig. 8a is a template of pants without profile curves, and Fig. 8b gives its resultant shape. Fig. 8c and 8d shows the template and final shape with profile curve control, which has a much smoother surface but with no shrinkage.



(a) H^1 (b) feature nodes relocated on H^1 (c) resultant surface on H^1 (d) mesh representation

(e) H^2 (f) feature nodes relocated on H^2 (g) resultant surface on H^2 (h) mesh representation

Fig. 7 Profile template decoding on different human models



(a) template without profile curves (b) resultant shape without profile control (c) template with profile curves (d) resultant shape with profile control

Fig. 8 Using profile curves to control final shape

5. Freeform Modification

As mentioned at the beginning of this paper, the freeform surface on the decoding result of profile templates might need to be further modified. In this section, four most useful freeform surface modification tools for apparel products are introduced.

Mesh painting

Users can conduct this tool to specify curves on the surface of products by 2D strokes. Our algorithm creates 3D line segments by projecting each line segment of the input 2D stroke onto the surface meshes of the

model along the view direction. The overall procedure is: for each line segment of the 2D stroke, first determine a bounding plane containing the projection of the line segment from the viewing position; then the system finds all intersections between the plane and each polygon of the object, and splices the resulting 3D line segments together (see Fig. 9). The actual implementation searches for the intersections efficiently using polygon connectivity information. If a ray from the viewing position crosses multiple polygons, only the polygon nearest to the viewing position is used for the surface painting. If the resulting 3D segments cannot be sliced together (e.g., if the stroke crosses a "fold" of the object, as shown in Fig. 10), the algorithm fails. The painted curves are stored by the ATTRIB_EDGEs in the data structure. There is another kind of painting, called penetrated painting, in which, all the polygons crossed by the rays are used to compute intersections. A "fold" does not influence the penetrated painting. The painting result is also stored by the ATTRIB_EDGEs. Both these paintings are implemented, and examples are shown in Fig. 11.

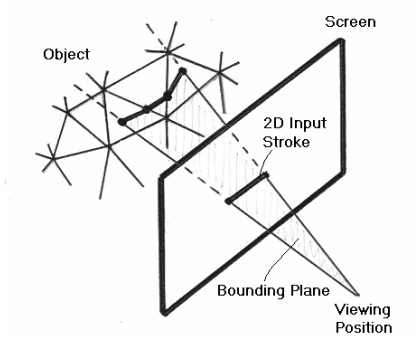


Fig. 9 Painting illustration

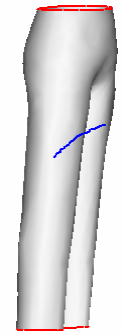
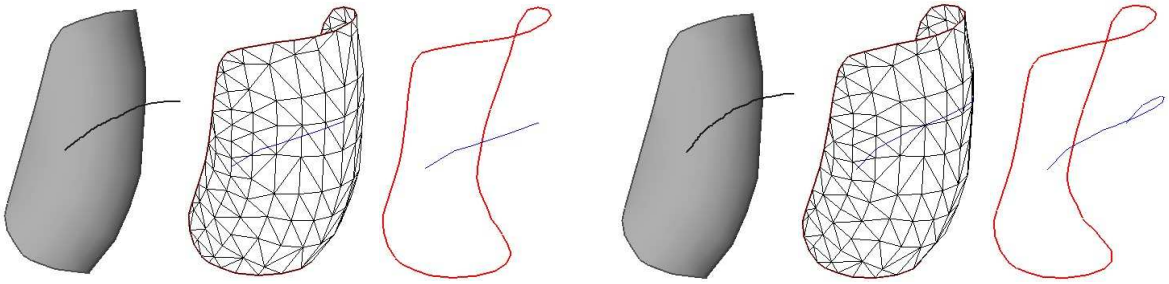


Fig. 10 Stroke across a "fold" leads algorithm fail



(a) painting result **(b) penetrated painting result**

Fig. 11 Examples of painting

Mesh cutting

The mesh cutting is to remove some parts of the given mesh surface by input 2D strokes. Similar to the Teddy system [33], the cutting tool is based on the painting algorithm. After painting a curve on the surface of a model, the constrained Delaunay triangulation algorithm [43] makes the painted curves to form the triangle edges of the model. Removing the triangles on the user-selected side of the painted curve (specified by another

stroke) from the model, the cutting result is obtained (illustrated in Fig. 12). Fig. 13 shows an example for using the mesh cutting tool to modify an evening dress.

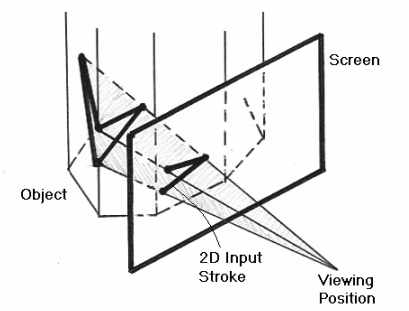


Fig. 12 Mesh cutting illustration

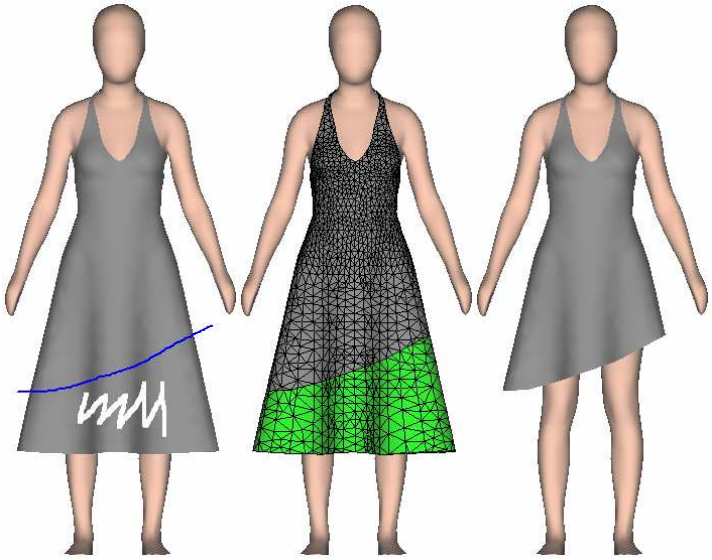
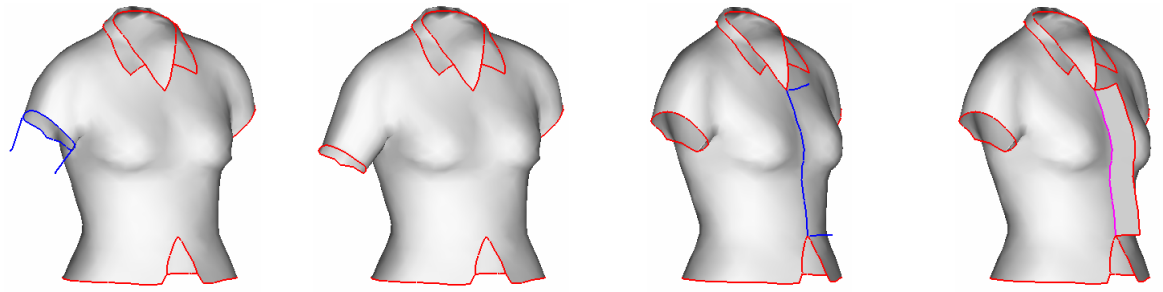
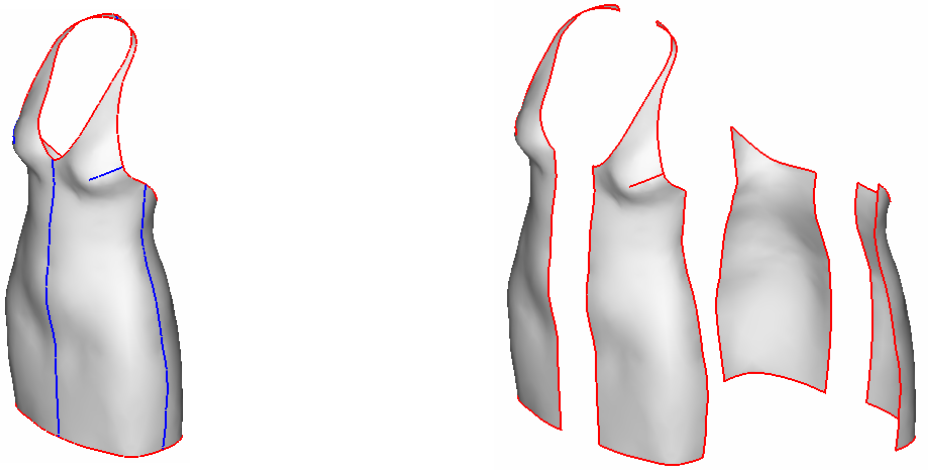


Fig. 13 Example of mesh cutting



(a) extrude a ring out **(b) the band sleeve elongated** **(c) extrude a curve out** **(d) a hanging patch formed**

Fig. 14 Mesh extrusion



(a) parting curves are defined on the dress **(b) partitioned 3D pieces**

Fig. 15 Mesh partitioning

Mesh extrusion

The extrusion operation is applied in our approach to create new polygonal meshes from base surface line segments (called the base curve) and extruding strokes. The mesh extrusion method implemented here is from our previous development in [44]. The mesh extrusion method is best illustrated by examples such as that shown in Fig. 14. In these examples, the given initial model is the freeform surface of a shirt; the mesh extrusion tool allows the user to sketch 2D input strokes (one stroke on the surface of given mesh, and other strokes depicting the profile curves of the extruded surface) to extrude a surface from the given mesh. Firstly the user draws a stroke on the object surface; then rotates the model to bring the stroke sideways and draws silhouette lines to extrude the surface. A sweep operation is applied to construct the 3D shape by moving the surface base curve, which is obtained by projecting the first stroke onto the surface of the given mesh, along the skeleton of the profile curves.

Mesh partitioning

The same as the cutting tool, the partitioning operation is also based on the painting algorithm. After painting the separating curves on the surface of a model (Fig. 15a), we apply the constrained Delaunay triangulation algorithm [43] to convert the painted curves into triangle edges of the model. After re-triangulation, the whole model can be divided to several sets of triangles; each set is a component of the product model (see Fig. 15b).

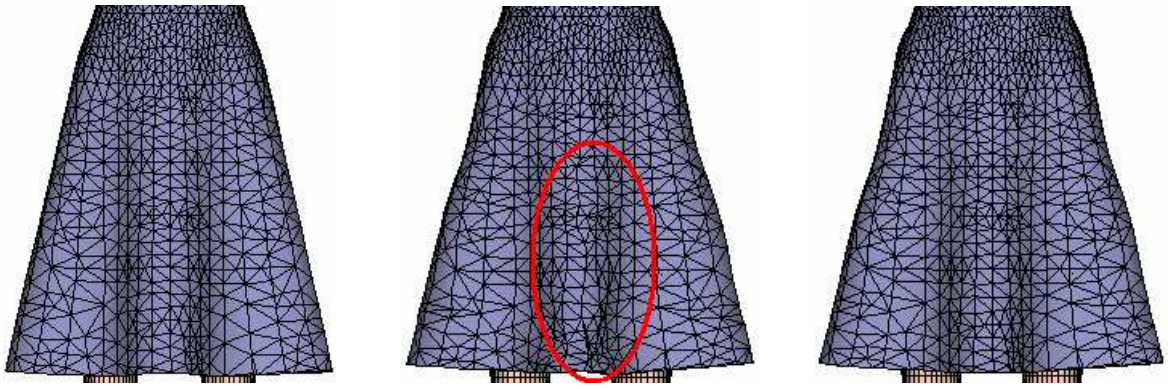
6. Detail Template Encoding/Decoding

When the freeform surface of an apparel product is modified, it is impossible to maintain the relationship between the product and a human model since lots of vertices have been inserted and removed. In this case, we need to rebuild the relationship between the product and a parameterized human model, so the detail template of a product's surface is created. The same as profile template, the detail template is also stored in our BODY data structure; however, comparing to a profile template, a detail template usually contains much more vertices and faces, and each vertices is encoded on several polygons of the parameterized human body.

The basic problem of encoding/decoding a detail template is actually how to parameterize a vertex on a freeform surface M by the polygons of a human model H. After the parameters of each vertex on M are determined, the detail template M can be deformed with the shape change of H by a mapping. This is similar to the manner of Free-Form Deformation (FFD) [34]. Nevertheless, as mentioned in [39], the parameterization method of FFD, which uses volumetric lattice to control a deformation, cannot be directly applied when using

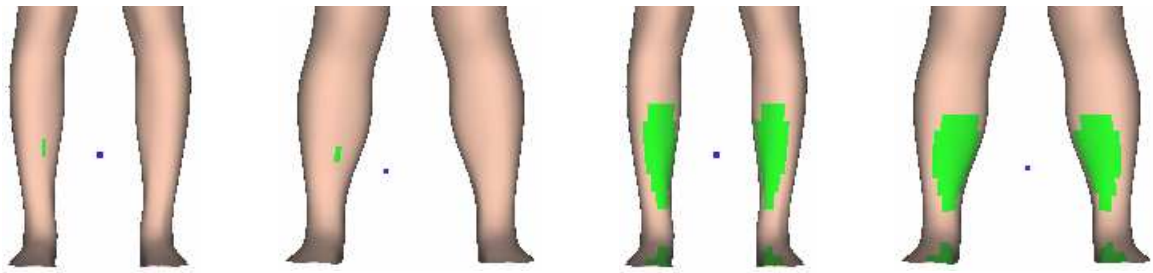
polygonal surfaces to control a deformation. Using FFD usually needs high computational cost to solve non-linear equations. Here, we adopt a method to blend linear mapped points given in the axial deformation methods [45-47]. This gives a very efficient and effective parameterization of vertices in M on a given H. Each vertex in M is parameterized and weighed by a number of polygons of a human model H. These values are stored and used for mapping to the new position when the shape of H is changed.

When parameterizing a vertex $q \in$ M, the following problems should be solved in the process: 1) what are the polygons on H should q be encoded on; 2) what are the parameters of q on a polygon $P_i \in$ H; 3) how to determine the weights for mapping the position of q by the new shape of H.



(a) detail template to be encoded **(b) result of single *p*-polygon encoding/decoding** **(c) result of multiple *p*-polygons encoding/decoding**

Fig. 16 Encoding/decoding with single *p*-polygon vs. multiple *p*-polygons



(a) single *p*-polygon mode **(b) multiple *p*-polygons mode**

Fig. 17 Encoding/decoding of a vertex with single *p*-polygon vs. multiple *p*-polygons manners

Once a point q on M is given, a fixed number of polygons on H are utilized to determine its new position when the shape of H is changed, they are called *p*-polygons. Why not just simply use the nearest polygon on H as the *p*-polygon? It is because that, as proposed in [39, 47], this simple solution gives significant ramps and bumps (as shown in Fig. 16, especially the circled regions). The blending result of multiple p-polygons blurs the above effects. The blending of multiple *p*-polygon mappings also preserves symmetry. As shown in Fig. 17, the encoding/decoding in single *p*-polygon mode did not maintain the vertex on the centerline of H, but it is

centered in the multiple *p*-polygons mode. By observation, in single *p*-polygon mode, the vertex is encoded on a polygon on the left leg of H (the green color region shows the related *p*-polygons). When the legs on another human body spreading out more, the vertex will be dragged to the left part. This is the cause of asymmetric occurrences.

During our tests, we found that choosing N_q = ROUND(N_H / 100) *p*-polygons usually gives good results, where N_H is the total number of polygons on H. We adopt the N_q nearest polygons on H to parameterize *q*. The nearest here does not mean the distance from *q* to a polygon's plane, but the distance from *q* to the centroid of a polygon. A voxel-based algorithm is developed to determine the *p*-polygons. The space around H is divided into $M \times N \times L$ boxes; each box $B_{i,j,k}$ contains a list of polygons whose centroid falls in the region of $B_{i,j,k}$. Then, the following algorithm using a minimum heap Ψ is adopted to determine the *p*-polygons, where Ψ uses the distance from *q* to the centroid of a polygon as the measurement parameter.

Algorithm determine_*p*-polygons()

1. $\Psi \leftarrow \phi$ and $h \leftarrow 1$;
2. Determine the box B_{i_0,j_0,k_0} containing *q*;
3. Insert all polygons in B_{i_0,j_0,k_0} into Ψ;
4. Set the checked flag of B_{i_0,j_0,k_0} as *true*;
5. *for* (*di*= -*h*; *di*<=*h*; *di*++)
6. *for* (*dj*= -*h*; *dj*<=*h*; *dj*++)
7. *for* (*dk*= -*h*; *dk*<=*h*; *dk*++)
8. *if* (the checked flag of B_{i_0+di,j_0+dj,k_0+dk} is *false*) {
9. Insert all polygons in B_{i_0+di,j_0+dj,k_0+dk} into Ψ;
10. Set the checked flag of B_{i_0+di,j_0+dj,k_0+dk} as *true*;
11. }
12. *if* (the number of polygons in $\Psi < N_q$) {
13. $h \leftarrow h+1$;
14. Go back step 5;
15. }
16. *return* the top N_q polygons in Ψ;

The voxel-based technique greatly reduces the time of skipping all polygons on H to determine the N_q nearest *p*-polygons. After the *p*-polygons are determined, we need to consider about the parameterization method of each polygon.

For a polygon $P_i =< p_1 p_2 \cdots p_n >$ on H, a linear local coordinate system Γ_i is formed at the centroid p_c of P_i (Fig. 18). Axis vectors of Γ_i are given by the following formulas:

$$X_{P_i} = n_{P_i},\ Y_{P_i} = \frac{p_1 - p_c}{\|p_1 - p_c\|},\ Z_{P_i} = X_{P_i} \times Y_{P_i}, \tag{8}$$

where n_{P_i} is the normal of P_i. If P_i is degenerated as a line segment or a point, it is simply ignored for the parameterization. Thus, the local coordinate (u_i, v_i, w_i) of a point $q \in$ M is

$$u_i = (q - p_c) \cdot X_{P_i},\ v_i = (q - p_c) \cdot Y_{P_i},\ w_i = (q - p_c) \cdot Z_{P_i}. \tag{9}$$

Besides (u_i, v_i, w_i), a weight ϖ_i of P_i should also be determined for the mapping process. The weight ϖ_i has a meaning of relative "strength" of Γ_i against other Γ_j s, and is calculated by an effect function. Without loss of general, the value of an effect function should be non-negative and decrease monotonously according to the distance between q and P_i. Also, the distance we adopted is the Euclidean distance between q and the centroid of P_i, p_c. The effect function conducted in our approach is

$$\varpi_i = \frac{1}{\varepsilon + \|q - p_c\|^3} \tag{10}$$

with $\varepsilon = 10^{-8}$ is utilized to avoid the singularity when $\|q - p_c\| = 0$. In summary, the encoded information of a vertex $q \in$ M is N_q indexes of the related p-polygons on H and the $(u_i, v_i, w_i, \varpi_i)$ s of each p-polygon.

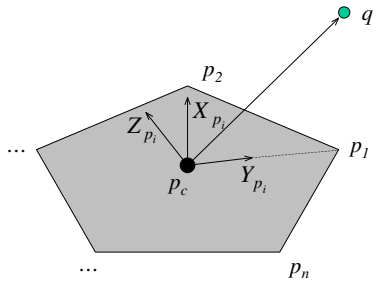


Fig. 18 Local frame Γ_i on P_i

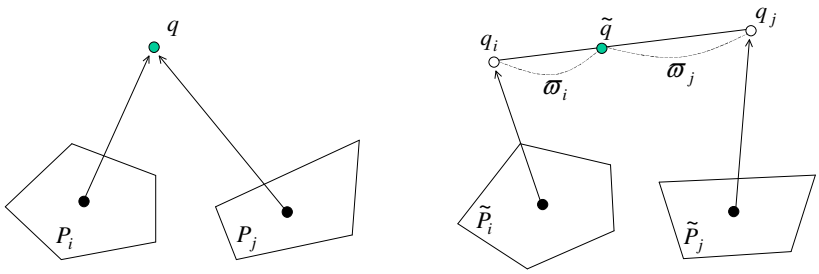


Fig. 19 Blending of mappings

The mechanism of blending the N_q mapping of p-polygons is shown in Fig. 19. After a human model $\tilde{\mathrm{H}}$ different from H is applied for decoding. The geometry of each polygon P_i is changed to $\tilde{P}_i$. The new centroid $\tilde{p}_c$ and axes $\tilde{X}_{P_i}$, $\tilde{Y}_{P_i}$, $\tilde{Z}_{P_i}$ of $\tilde{P}_i$ are computed by the same method described in eq. (9). Then, q is mapped to q_i with $\tilde{P}_i$ as

$$q_i = \tilde{p}_c + u_i \tilde{X}_{P_i} + v_i \tilde{Y}_{P_i} + w_i \tilde{Z}_{P_i} \tag{11}$$

The new position q_j of q mapped by $\tilde{P}_j$ can be calculated in the same way. Since (u_i, v_i, w_i) represents the relative position of q to the polygon P_i, generally, q_i and q_j are not coincident. The final mapped point $\tilde{q}$ is calculated by the following formula, which blends the points q_i with the weights ϖ_i:

$$\tilde{q} = \frac{\sum_i \varpi_i q_i}{\sum_i \varpi_i}. \tag{12}$$

By changing the position of each $q \in \mathrm{M}$ as $\tilde{q}$, the deformed mesh $\tilde{\mathrm{M}}$ is determined. The decoding process of a detail template is finished.

Examples of using detail template for the design automation of customized apparel products are shown in Fig. 20 and Fig. 21, where the customized apparel products are designed on the body of H, after the detail template encoding and decoding, the fitted products for different body shapes – H^1, H^2, and H^3 are generated.

7. Patterns for Manufacturing

The garment manufacturing industry needs 2D patterns to be used in the manufacturing processes. The energy-based surface-flattening algorithm presented in [48] is integrated to generate the corresponding 2D patterns of a 3D apparel model by using a spring-mass model. This procedure consists of triangles flattening and planar mesh deformation. During the triangles flattening phase, triangles are flattened one by one; and a partial spring-mass system containing flattened triangles is deformed to release the strain energy during the flattening. After all the triangles are flattened, the spring-mass system will have all the triangles of the given surface. The planar triangular mesh deformation process is directed by the energy function of the spring-mass system. By releasing the energy function, we can obtain the 2D pattern related to the given 3D mesh surface. Fig. 22 shows an example of the corresponding 2D patterns of the dress shown in Fig. 15.

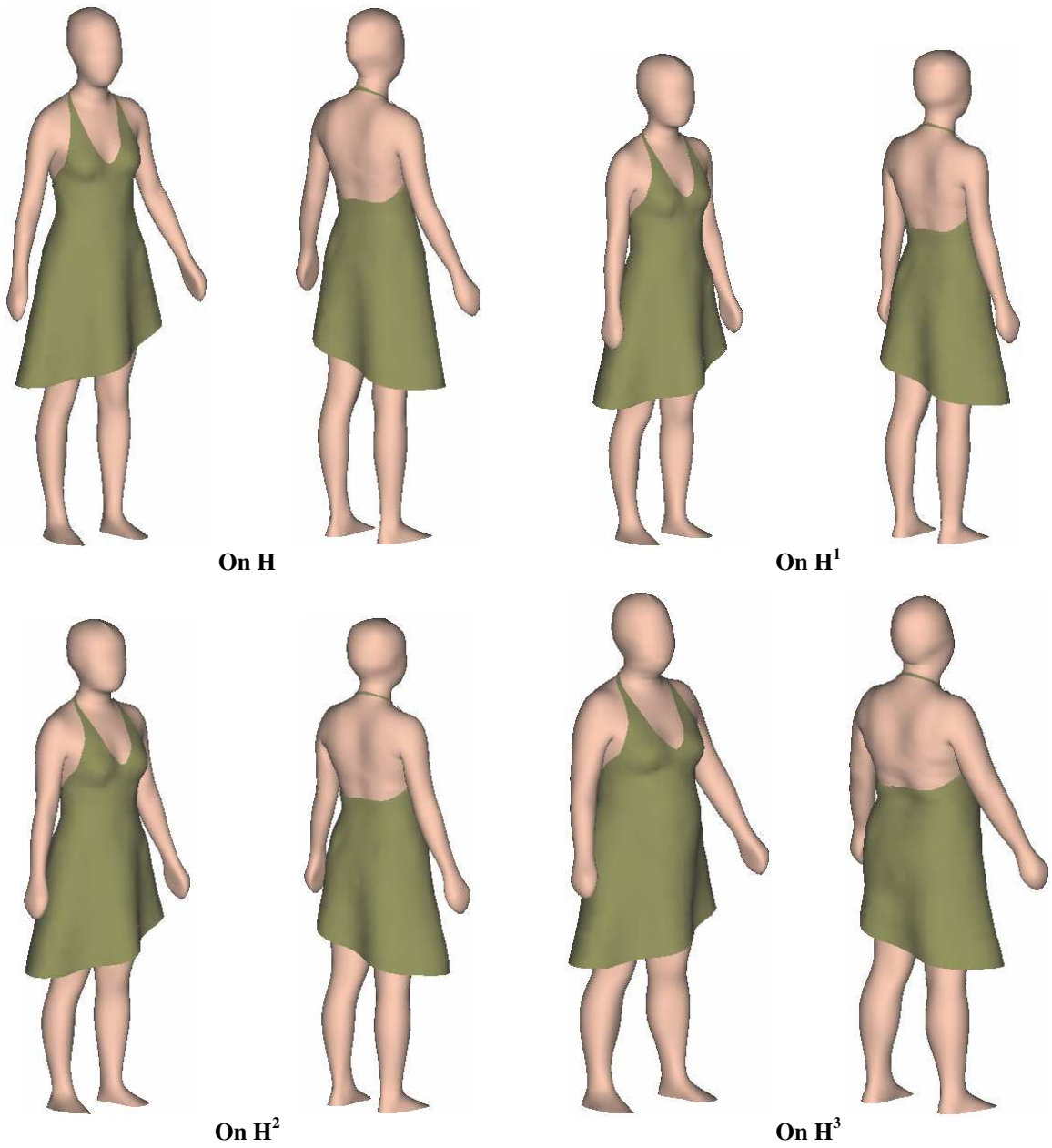


Fig. 20 Example I of detail template encoding/decoding: design automation of a trimmed dress

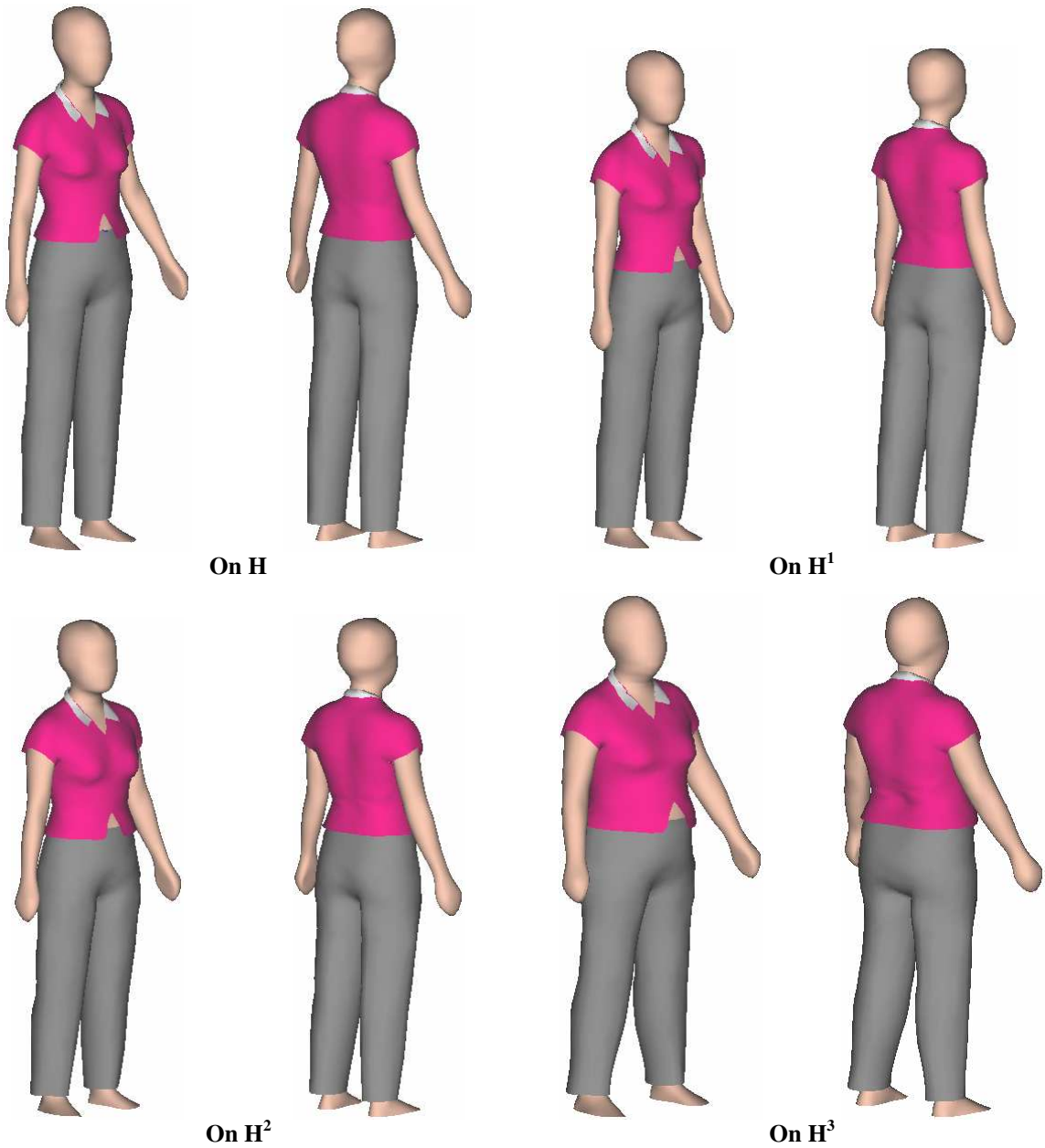


Fig. 21 Example II of detail template encoding/decoding: a set of shirt and pants

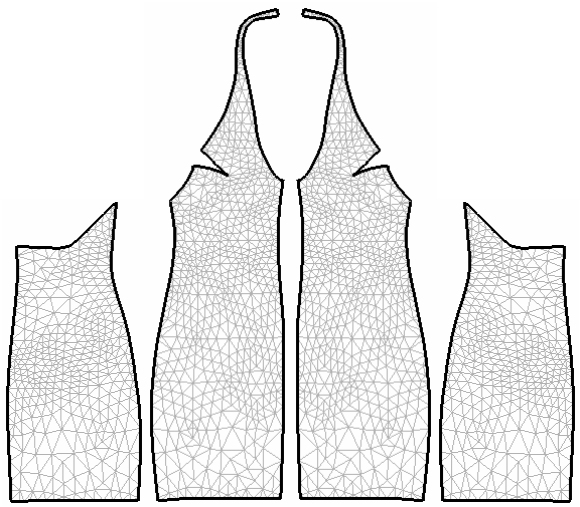


Fig. 22 2D patterns for the 3D dress previously shown in Fig. 16

8. Conclusion and Discussion

This paper provides three-dimensional solution techniques to achieve the automatic made-to-measure (AMM) scheme for apparel products. With the help of AMM, the fitting guaranteed three-dimensional clothes of a same style can be automatically generated around the human bodies with different shapes in the computer system. This can greatly improve the efficiency of pattern generation in apparel industry. To overcome the limitation of solving fitting and grading in 2D, we develop our AMM in 3D. Compared to other cloth simulation based approaches, our solution solves the problem in a reverse way – directly design product in 3D space. Different styles of clothes are represented by different 3D feature templates which are well encoded on the parameterized human bodies. The easing relationships between the feature template and the parameterized human body are encoded and stored in the template of clothes. When clothes on different human bodies are required, a decoding process is performed to reconstruct the 3D cloth patterns preserving easing relationships around human bodies. Based on the constructive design result, we can modify its freeform surfaces to achieve more complex style. By detail template encoding/decoding, the modified result can also be applied automatically on different body shapes. In summary,

- an integrated solution is presented for achieving the *automatic made-to-measure* (AMM) for apparel products, where the products designed on one human body can be automatically regenerated on different bodies according to their own morphology;
- to represent the complex surface and the uncompleted model of an apparel product during design, a non-manifold data structure, that is a hybrid of the B-rep and the complex-based ones, is given;

- a constructive design approach by profile templates encoding/decoding is presented for the design of a new product (i.e., start from none);
- after giving four intuitive freeform modification tools, a novel detail template encoding/decoding technique is developed to implement the AMM of an apparel product with detail freeform surfaces.

In one word, our approach gives an integrated solution for the design automation of customized apparel products.

The modified variation subdivision scheme applied in the profile template decoding has the shrinkage effect that influences the final shape of a constructive product. Although the profile curves can more or less resist shrinkages, we are still seeking other better surface interpolation techniques to take place of the modified variation subdivision scheme. More freeform modification tools are still under research for generating decorative elements on the surface of an apparel product (e.g., wrinkles, laces, and plies). Another possible further research relates to the detail template decoding. The detail template decoding process may take the adaptive subdivision scheme into consideration so the surfaces with better qualities could be generated at the place with high curvatures.

References

[1] *Gerber*. http://www.gerbertechnology.com.

[2] *Lectra*. http://www.lectra.com.

[3] *DressingSim*. http://www.dressingsim.com.

[4] Volino P., Courchesne M., and Thalmann N.M., Versatile and efficient technique for simulating cloth and other deformation objects, SIGGRAPH 95 Proceeding, ACM., 1995, pp.137-144, New York, USA.

[5] Fan J., Wang Q.F., Chen S.F., Yuen M.M.F., and Chan C.C., A spring-mass model-based approach for warping cloth patterns on 3D objects, The Journal of Visualization and Computer Animation, Vol. 9, No. 4, October/December 1998, pp. 215-227.

[6] Baraff D., and Witkin A., Large steps in cloth simulation, Proceedings of SIGGRAPH 98, pp.43–54, 1998.

[7] Choi K.J., and Ko H.S., Stable but responsive cloth, Proceedings of SIGGRAPH 2002, pp.604–611, 2002

[8] Zhang D.L., and Yuen M.M.F., Cloth simulation using multilevel meshes, Computers & Graphics, vol.25, no.3, pp.383-389, 2001.

[9] Bridson R., Fedkiw R.P., and Anderson J., Robust treatment of collisions, contact, and friction for cloth animation, Proceedings of SIGGRAPH 2002, pp.594–603, 2002.

[10] Bridson R., Marino S., and Fedkiw R., Simulation of clothing with folds and wrinkles, Eurographics/SIGGRAPH Symposium on Computer Animation 2003, pp.28-36, 2003.

[11] Cordier F., Seo H., and Thalmann N.M., Made-to-measure technologies for an online clothing store. IEEE Computer Graphics and Applications, vol.23, no.1, pp.38–48, January 2003.

[12] Terzopoulos D., Platt J., Barr A., and Fleischer K., Elastically deformable models, In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp.205–214, 1987.

[13] Scheepers F., Parent R.E., Carlson W.E., and May S.F., Anatomy-based modeling of the human musculature, Computer Graphics Proceedings, SIGGRAPH 97. ACM. 1997, pp.163-172. New York, NY, USA.

[14] Wilhelms J., and Van Gelder A., Anatomically based modeling, Computer Graphics Proceedings, SIGGRAPH 97. ACM. 1997, pp.173-180. New York, NY, USA.

[15] Wang C.C.L., Cheng T.K.K., and Yuen M.M.F., From laser-scanned data to feature human model: a system based on fuzzy logic concept, Computer-Aided Design, vol.35, no.3, pp.241-253, 2003.

[16] Wang C.C.L., Wang Y., Cheng T.K.K., and Yuen M.M.F., Virtual human modeling from photographs for garment industry, Computer-Aided Design, vol.35, no.6, pp.577-589, 2003.

[17] Dekker L., 3D human body modeling from range data, Ph.D. Thesis, University College London, 2000.

[18] Hilton A., Beresford D., Gentils T., Smith R., Sun W., and Illingworth J., Whole-body modelling of people from multiview images to populate virtual worlds, Visual Computer, vol.16, no.7, 2000, pp.411-436.

[19] Lee W.S., Gu J., and Magnenat-Thalmann N., Generating animatable 3D virtual humans from photographs, Computer Graphics Forum, vol.19, no.3, 2000, pp.1-10.

[20] Allen B., Curless B., and Popović Z, The space of human body shapes: reconstruction and parameterization from range scans, ACM Transactions on Graphics, vol.22, no.3, pp.587-594.

[21] Wang C.C.L., Parameterization and parametric design of mannequins, Computer-Aided Design, accepted.

[22] Au C.K., and Yuen M.M.F., A semantic feature language for sculptured object modeling, Computer-Aided Design, vol.32, no.1, pp. 63-74, 2000.

[23] Weiler K., The radial edge structure: a topological representation for non-manifold geometric boundary modeling, edited by Wozny M. J., McLaughlin H. W., and Encarnacao J. L., Geometric modeling for CAD applications, North-Holland, pp.3-36, 1986.

[24] Choi Y., Vertex-based boundary representation of non-manifold geometric models, PhD. Thesis, Carnegie Mellon University, 1989.

[25] Gursoz E.L., Choi Y., and Prinz F.B., Vertex-based boundary representation of non-manifold boundaries, edited by Wozny M. J., Turner J. U., and Preiss K., Geometric Modeling for Product Engineering, North-Holland, pp.107-130, 1990.

[26] Lee S.H., and Lee K., Partial entity structure: a compact non-manifold boundary representation based on partial topological entities, Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, Ann Arbor, Michigan, pp.159-170. 2001.

[27] Rossignac J., and O'Conner M. A., SGC: a dimensional–independent model for pointsets with internal structures and incomplete boundaries, edited by Wozny M. J., Turner J. U., and Preiss K., Geometric Modeling for Product Engineering, North-Holland, pp.145-180, 1990.

[28] Lienhardt P., Topological models for boundary representation: a comparison with n-dimensional generalized maps, Computer-Aided Design, vol.23, no.1, pp.59-82, 1991.

[29] Hubeli A., and Gross M., Multiresolution methods for nonmanifold models, IEEE Transactions on Visualization and Computer Graphics, vol.7, no.3, pp.207-221, 2001.

[30] Wang C.C.L., Wang Y., and Yuen M.M.F., Feature-based 3D non-manifold freeform object construction, Engineering with Computers, vol.19, no.2-3, pp.174-190, 2003.

[31] Meyers D., Skinner S., and Sloan K., Surface from Contours, ACM Transaction on Graphics, vol. 11, no. 3, pp.228-258, 1992.

[32] Bruyns C. D., and Senger S., Interactive cutting of 3D surface meshes, Computers & Graphics, vol.25, pp.635-642, 2001.

[33] Igarashi T., Tanaka H., and Matsuoka S., Teddy: a sketching interface for 3D freeform design, SIGGRAPH 1999 Conference Proceedings, 1999

[34] Sederberg T. and Parry S., Free-form deformations of solid geometric models, Computer Graphics, 20: 151-160, 1986.

[35] Coquillart S., Extended free-form deformations: A sculpting tool for 3D geometric modeling, Computer Graphics, 24(4): 187-196, 1990.

[36] Chang Y. K. and Rockwood A. P., A generalized de Casteljau approach to 3D free-form deformation, Computer Graphics, 28(4): 257-260, 1994.

[37] Hsu W., Hughes J., and Kaufmann H., Direct manipulations of free-form deformations, Computer Graphics, 26(2): 177-184, 1992.

[38] MacCracken R. and Joy K., Free-form deformations with lattices of arbitrary topology, Computer Graphics, 181-189, 1996.

[39] Kobayashi K.G., and Ootsubo K., t-FFD: freeform deformation by using triangular mesh, ACM Symposium on Solid Modeling and Application 2003, pp.226-234, 2003.

[40] Masuda H., Topological operators and Boolean operations for complex-based non-manifold geometric models, Computer-Aided Design, vol.25, no.2. 1993.

[41] Wang C.C.L., Wang Y., and Yuen M.M.F., Feature based 3D garment design through 2D sketches, Computer-Aided Design, vol.35, no.7, pp.659-672, 2003.

[42] Kobbelt L., Discrete fairing and variational subdivision for freeform surface design, The Visual Computer, vol.16, no.3/4, pp.142-158, 2000.

[43] de Floriani L., and Puppo E., An online algorithm for constrained delaunay triangulation, CVGIP-Graphical Models & Image Processing, vol.54, no.4, pp.290-300, USA.

[44] Wang C.C.L., and Yuen M.M.F., Freeform extrusion by sketched input, Computers & Graphics, vol.27, no.2, pp.255-263, 2003.

[45] Lazarus F., Coquillart S., and Jancene P., Axial deformations: an intuitive deformation technique, Computer-Aided Design, vol.26, no.8, pp.607-613, 1994.

[46] Singh K. and Fiume E., Wires: a geometric deformation technique, SIGGRAPH 98 Conference Proceedings, pp.405-414, 1998.

[47] Wang C.C.L., and Yuen M.M.F., View-dependent deformation with sketching input, 2001 ASME DETC/CIE, 27th Design Automation Conference, Pittsburgh, Pennsylvania, September, 2001.

[48] Wang C.C.L., Smith S.S.F., and Yuen M.M.F., Surface flattening based on energy model, Computer-Aided Design, vol.34, no.11, pp.823-833, 2002.

Appendix A Entities and Attributes

The detail descriptions of each entity and attribute in our data structure are listed below.

Table A-1 Representational Entities

Entity	Representation	Description
BODY	Complex of PMESHs and PMESHJOINTs	Highest level entity in a model.
PMESH	Complex of PMESHEDGEs, PFACEs, PEDGEs, and PNODEs	A portion of BODY's surface. It defines the shape of BODY, and is represented by many polygons.
PMESHJOINT	Complex of PMESHEDGEs	Assembly information of PMESHs.
PMESHEDGE	Complex of PEDGEs	A collection of PEDGES, every PEDGE has its own direction flag.
PFACE	Complex of *n* PEDGEs	Portion of a PMESH.
PEDGE	Complex of two PNODEs	Boundary of a PFACE, holds the model together with adjacency information. (+ve, clockwise; -ve, anti-clockwise)
PNODE	A point	Boundary of a PEDGE.

Table A-2 Representational Attributes

Attribute	Representation	Description
ATTRIB_NODE	A point	An attribute point on the surface of a PMESH.
ATTRIB_EDGE	Complex of ATTRIB_NODEs	An attribute curve lying on the surface of a PMESH. It is a list of ATTRIB_NODEs, and passes polygonal faces of the PMESH
ATTRIB_EDGENODE	A point	An attribute point on a polygonal edge. Its position depends on the positions of the two endpoints of the edge.
ATTRIB_FACENODE	A point	An attribute point in a polygonal face. Its position depends on the positions of the nodes of the face.

Appendix B Extended Euler Operators

The eight extended Euler operators are listed in Table B-1 and Fig. B-1. Reverse operators are enclosed in brackets, and *Chole* denotes a hole in a complex.

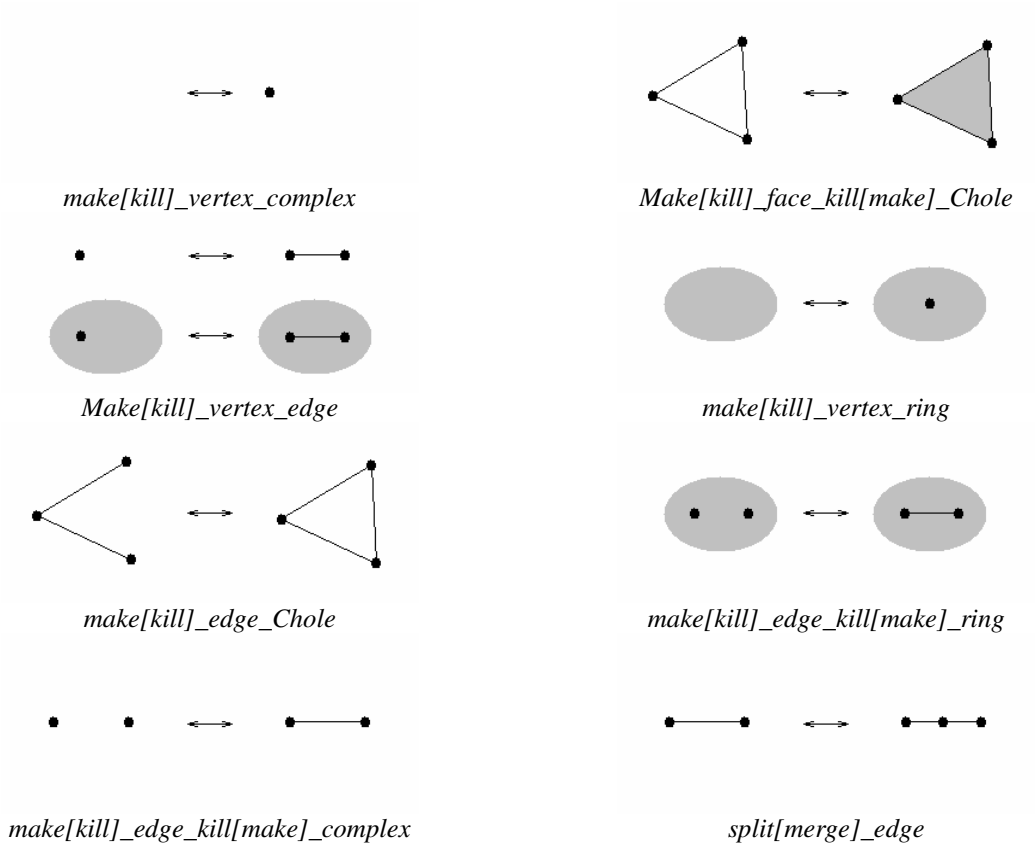


Fig. B-1 Extended Euler operators

Table B-1 Function Description of Extended Euler Operators

Operator	Function of operator
make[kill]_vertex_complex	Create a single vertex complex
make[kill]_vertex_edge	Create a vertex and an edge connecting to an existed vertex (the operator can be carried out freely or inside a face)
make[kill]_edge_Chole	Connect two vertex by a new edge to form a hole in a complex
make[kill]_edge_kill[make]_complex	Connect two vertex complexes by a new edge (since the two complexes are connected, one complex should be removed)
make[kill]_face_kill[make]_Chole	Create a new face on a complex hole and remove the hole
make[kill]_vertex_ring	Create a stand-alone vertex on a face (the single vertex forms a ring)
make[kill]_edge_kill[make]_ring	Connect two stand-alone vertices by a new edge on a face (two rings are merged into one ring, thus one ring should be removed)
Split[merge]_edge	Split one edge into two edges by adding a new vertex on the original edge