A Data-driven Pattern Design Method for Watertight Plush Toy Models

Yinghan Jin[1], Jin Wang[2], Jituo Li[3] and Dongliang Zhang[4]

[1] College of Computer Science, Zhejiang University, yh.jin@zju.edu.cn
[2] School of Mechanical Engineering, Zhejiang University, dwjcom@zju.edu.cn
[3] School of Mechanical Engineering, Zhejiang University, jituo_li@zju.edu.cn
[4] College of Computer Science, Zhejiang University, dzhang@zju.edu.cn

Corresponding author: Dongliang Zhang, dzhang@zju.edu.cn

Abstract. This paper presents an automatic pattern design method for plush toys using a data-driven approach. For a watertight toy model, we use a divide-and-conquer strategy to convert it into a component-based model, then automatically design patterns of each component. Firstly, a segmentation and componentization method based on deep learning is proposed to decompose a 3D watertight model into several meaningful components. Then, the most similar component template is retrieved and used to generate seam lines of the model. Finally, 2D patterns of the toy model are obtained by flattening 3D surface patches. The segmentation and componentization method can also be used to easily edit the shape and structure of a watertight model. Experimental results show that reasonable patterns can be designed automatically and efficiently using the proposed method.

Keywords: Computer-aided Pattern Design, 3D Plush Toy Design, 3D Decomposition, Component-based Template, Surface Flattening, Watertight Plush.
DOI: https://doi.org/10.14733/cadaps.2023.28-43

1 INTRODUCTION

Designing 2D patterns is one of the most important and difficult steps in the whole process of plush toy design. Traditionally, 2D patterns are designed manually by designers. The manual design method is time-consuming, and it has high requirements for the designers' spatial imagination, practical experience and aesthetic ability. With the development of computer graphics and computer-aided design, patterns of toys can be obtained by flattening the surface of a 3D model. The key to designing patterns of a toy model is to draw appropriate seam lines on the 3D surface, which divide the surface into several patches. The efficiency of pattern design can be dramatically improved if seam lines of a toy model can be generated automatically.

However, automatic generation of seam lines is a challenging task. On the one hand, the shapes and structures of toy models are diverse, leading to many different design schemes of seam lines.

On the other hand, there may exist several reasonable design schemes for one toy since there is no standard design for a plush toy. Moreover, the knowledge of seam line design is related to manufacturability and aesthetics, so it is non-standard, non-formulaic and non-structured. Therefore, the problem of generating seam lines automatically is hard to be solved by the computational optimization method.

In our previous work [1], we proposed an automatic pattern design method of generating seam lines for component-based plush toy models. Its main idea is to reuse the design knowledge using a component-based template, which retrieves the best matched design template from the library and mapping seam lines to the component. However, the method is limited to the ***component-based model***, which is consisted of several components. In practice, some toy models are non-component models. A non-component model is a ***watertight model*** which is consisted of only one closed surface. The template-based method can not be applied to the watertight model directly.

To solve the problem, we adopt a divide-and-conquer strategy to convert a watertight model into a component-based model through a segmentation and componentization method, and use design template to automatically generate seam lines of each component. To achieve a high accuracy of segmentation, a deep learning method is utilized to segment the 3D model based on a dataset of plush toy models.

Fig. 1 shows the workflow of the proposed method of designing patterns for a watertight model. Firstly, a 3D model (Fig. 1(a)) will be segmented into parts (Fig. 1(b)), and each part will be componentized into a component (Fig. 1(c)). Next, special seam lines named the structure lines are generated between each pair of connected parts (Fig. 1(d)). Then, template retrieval is conducted for each component to search the best matched template to generate seam lines for the component (Fig. 1(e)), and the seam lines of each component will be mapped to the original watertight model (Fig. 1(f)). Finally, the surfaces divided by seam lines are flattened to obtain 2D patterns (Fig. 1(g)), and a physical product can be manufactured according to these patterns (Fig. 1(h)).

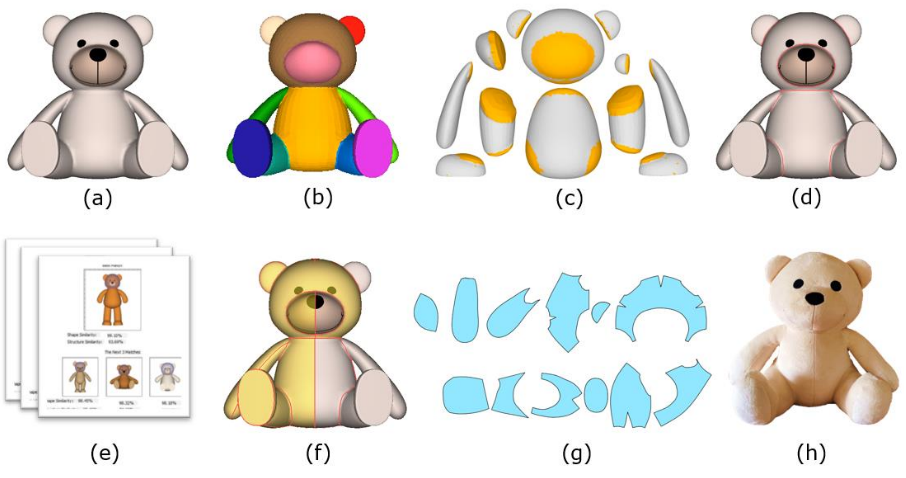


Figure 1: Workflow of the proposed method. (a) 3D toy model. (b) 3D segmentation. (c) Componentized part. (d) Generated structure lines. (e) Template retrieval. (f) Mapped seam lines. (g) Flattened 2D patterns. (h) Manufactured physical toy.

The main contributions of the study are as follows:

(1) An automatic method of generating seam lines for watertight plush toy models is presented. We employ a divide-and-conquer strategy to decompose a watertight model into a component-based

model, and use component-based design templates to automatically create seam lines for the toy model.

(2) A segmentation and componentization approach is proposed to exactly decompose a 3D watertight model into parts consistent with the structures of the toy model. A deep learning method is utilized to segment the 3D model based on a dataset of toy models. In addition, the segmentation and componentization method can be used to easily modify the shape and structure of the watertight model.

The article is organized as follows: Section 2 reviews the related work on the pattern design of soft products and part-based 3D segmentation. In section 3, we briefly introduce our previous pattern design method. In Section 4 and 5, we elaborate on the proposed method of 3D model segmentation, componentization and pattern design using templates. Experimental results are presented in Section 6. Finally, we summarize the paper and discuss the future work in Section 7.

2 RELATED WORKS

Pattern Design of Soft Products: Several interactive systems have been proposed to facilitate the pattern design process of soft products, including plush/inflatable toys, garments, etc. Igarashi and Igarashi [2] proposed *Pillow*, a pattern design system for plush toys. After specifying the segmentation boundaries interactively of an imported 3D model, 2D patterns are generated by flattening segmented regions, and patterns are simulated into a sewn product. Mori and Igarashi [3] presented *Plushie*, a pattern design framework for nonprofessional users to create plush toys and 2D patterns from a blank canvas. The user creates and edits the 3D model by sketching, then 2D patterns are generated automatically, and the 3D model is updated by the result of applying sewing simulation to the patterns. Umetani et al. [4] presented *Sensitive Couture*, offering bidirectional design capabilities between 2D patterns and corresponding simulated 3D garments. Using parameterized pattern templates, 3D user interactions are mapped to parameter changes and then 2D patterns are edited according to the updated parameters. Aric et al. [5] presented a direct 3D garment editing method, and 2D patterns can be generated after free-form 3D editing, which is different from the parameter restriction of *Sensitive Couture*. In their method, a fixed-point optimization scheme is proposed to adjust 2D patterns to ensure the simulated garments retain the desired shape.

Few studies have been conducted on automatic pattern design, and most of them aim at optimization patterns. Skouras et al. [6] provided a physics-based pattern optimization method for the pattern design of inflatable structures. Montes et al. [7] proposed a computational automatic pattern design method specially for skintight clothing based on a physics-driven optimization algorithm. These methods require an initial guess of the segmentation boundaries, which is often unavailable. Previously we presented an automatic pattern design method by directly generating segmentation boundaries using component-based templates [1]. However, this method restricts the input model to be component-based. In this study, we extend it to deal with watertight models.

Part-based 3D segmentation: Part-based model segmentation is a fundamental research topic in computer graphics and geometry processing, whose objective is to decompose a given 3D object into visually meaningful parts. So far, several classical approaches of mesh segmentation have been proposed, aiming to formulate geometric criteria to form meaningful segments or segment boundaries. Representative algorithms include fuzzy clustering with graph cuts [8], spectral clustering [9], region growing [10,11], clustering based on prominent feature points [12-14], and skeleton extraction [15]. In the last decade, instead of analyzing the individual shape, more research focused on data or knowledge driven approaches, which is known as co-segmentation. After learning from a multitude of shapes, co-segmentation will produce more consistent segmentation results of a family of meshes. Kalogerakis et al. [16] proposed the first co-segmentation algorithm based on supervised learning, which learns an objective function from the labeled training meshes and outputs the segmentation result via the minimization of the objective function. The segmentation method based on a deep convolution neural network presented by Guo et al. [17] extracts the information of 3D features using a convolution kernel to realize model segmentation. To speed up the training performance of supervised learning, Xie et al. [18] proposed a segmentation algorithm with real-time online sequential learning based on the extreme learning machine (ELM). In recent years, a variety of segmentation algorithms utilizing deep learning were proposed to understand the 3D models of different representations, e.g. multi-view 2D images [19, 20], point clouds [21-23], and triangular meshes [24-26]. In this paper, we take advantage of co-segmentation based on deep learning and utilize geometric features of the toy models to achieve fine decomposition.

3 COMPONENT-BASED DESIGN METHOD

In [1], we presented a pattern design method for component-based models using component-based templates. In this paper, we extend it to design patterns for the watertight models. For the integrity of the paper, we briefly introduce the component-based pattern design method.

The method is based on component-based models, meaning the model is represented by several watertight 3D meshes. In toy design, component-based modeling is proved to be an easy and effective modeling method. A variety of toy models can be built quickly and intuitively by transforming and assembling components. Fig. 2a shows a watertight model of a teddy bear in Princeton segmentation benchmark [27]. We create a similar component-based model (Fig. 2b) using sketch modeling [28], and its corresponding components are shown in Fig. 2c. Different teddy models can be designed easily by transforming and assembling the same components, as shown in Fig. 2d.

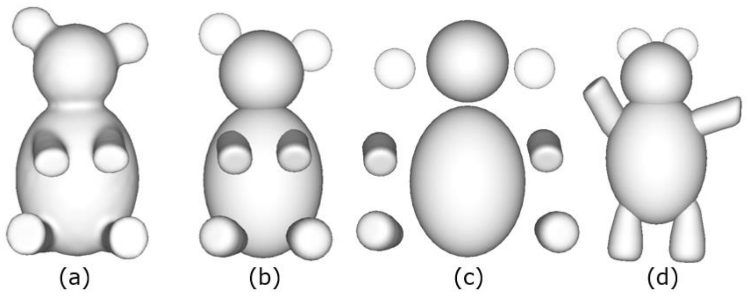


Figure 2: 3D toy models. (a) A watertight model. (b) A component-based model. (c) Corresponding components. (d) A different component-based model assembled by the same components.

In the method, the component-based template is proposed to reuse the design of seam lines. It is inspired by the fact that, toy parts with similar shapes usually have a similar design of seam lines. The template stores the information of professionally designed seam lines of classified toy components. The design of seam lines of a new component can be realized by retrieving the most similar component template from the library and mapping the seam lines of the matched template to the target component.

The process of automatically designing seam lines of a component-based model is as follows. Firstly, intersection lines between connected components are generated automatically. Next, for each component, the component descriptor is computed and the best match template is retrieved by comparing the shape and structure similarity with templates in the library. Then, the seam lines of the best-matched template are mapped onto the target component. At last, 2D patterns are obtained by flattening the surface patches which are divided by seam lines.

4 SEGMENTATION AND COMPONENTIZATION OF 3D MODELS

The component-based template works well to generate seam lines for the component-based model. However, it falls short in handling watertight models. To address this, we propose a method of decomposing the watertight model into several components by model segmentation and componentization. Thus, an equivalent component-based model is created for the watertight model so that it makes the component-based template suitable for the watertight model.

4.1 Segmentation

In the study, segmentation is used to automatically divide a watertight model into several meaningful parts consistent with the structure of the toy model. General segmentation algorithms for dealing with the individual model sometimes have some shortcomings, such as over-segmentation, pre-definition of the number of clusters, and unsatisfactory hierarchical segmentation (e.g. at a certain hierarchical level, the nose part is extracted but the ear part is not) [29]. To achieve a stable and consistent segmentation result for the toy model, we divide the segmentation process into two steps. The first step is to obtain a rough segmentation result using the deep learning based co-segmentation, and the second step is to refine the segmentation exactly using local geometry features.

The rough segmentation we adopted is the segmentation branch of PointNet++ [21], which is an improvement in the ability of processing non-uniform point clouds of PointNet [23]. The architecture of the PointNet++ network is shown in Fig.3. In the set abstraction level, point features are extracted hierarchically by a group of set abstraction layers (SA), and each SA layer includes three sublayers: sampling, grouping, and PointNet. In the sampling sublayer, the input points are sampled by the farthest point sampling (FPS) method, and several central points are selected from points. In the grouping sublayer, the ball query method is used to find points in local regions within a certain radius of the selected center points. In the PointNet sublayer, each local region is input into the PointNet, and features of local points are obtained through multilayer perceptron network (MLP) and max pooling. In the feature propagation level, features are hierarchically propagated from subsampled points to the original points, thus the segmentation labels for all points are obtained. Finally, the predicted segment label of each point is mapped to the original model.

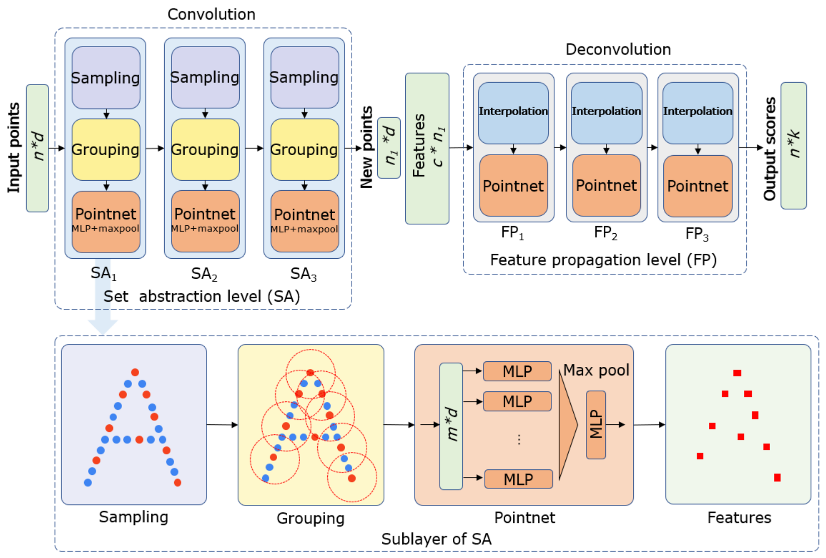


Figure 3: The structure of PointNet++ for segmentation.

To improve the accuracy of segmentation, we create a dataset of plush toy models to train the model, which will be detailed in Section 6. The predicting result from the deep learning network is semantic segmentation. After training, the overall accuracy of segmenting is 88.07%, which can predict the segmentation of the 3D model and decompose meaningful parts. However, the boundaries between connected parts cannot be gotten exactly, as shown in Fig. 4(a).

To refine the segmentation boundaries, we utilize the graph cut method after the first step of rough segmentation. The exact segment of the coarse boundaries between segments can be formulated as a Graph cut problem, which can be solved by applying the maximum flow (minimum cut) algorithm [30]. To do it, it is necessary to define the search region and the capacity on the arcs of the flow graph. For the mesh $G = (V, E)$ (where *V* is the vertex, and *E* is the edge) formed by each pair of connected parts, the search region is found by searching for the faces within the distance $0.25\,\beta$ (where β is the diagonal length of the minimum bounding box of the smaller part) from the rough segment boundary in *G*. The search region (*Green*) is formed between two certain regions (marked as *Red* and *Blue*), as shown in Fig. 4(b). Let f_i, f_j be the dual faces in *G*, α_{ij} be the angle between the normals of dual faces and $AngDist(\alpha_{ij}) = \mu(1 - cos\alpha_{ij})$, the capacity between f_i and f_j are defined as:

$$\mathrm{Cap}(f_i, f_j) = \begin{cases} \frac{1}{1+\frac{AngDist(\alpha_{ij})}{avg(AngDisit)}} & if\{f_i, f_j \neq \; Red, Blue\}, \\ \infty & else, \end{cases} \tag{1}$$

where $\mu = 1$ or 0.1, for concave and convex angles, respectively. Since the minimum cut tends to pass through arcs with small capacities, for each pair of connected parts, the minimum cut turns to be the segmented boundary between them, as shown in Fig. 4(c).

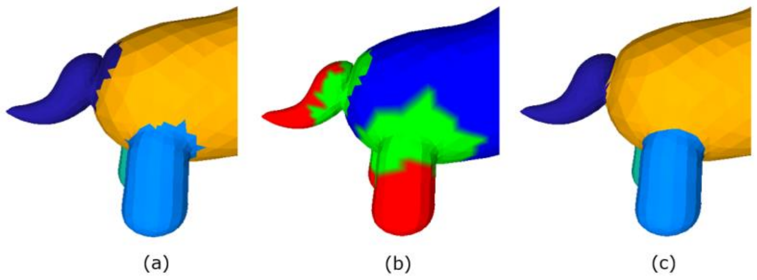


Figure 4: 3D model segmentation. (a) Segmentation based on deep learning. (b) Search regions of Graph cut. (c) Exact segmentation using Graph cut.

4.2 Componentization

Componentization is used to convert a segmented part of a watertight model into a component by repairing the holes on the part and retaining its original shape as much as possible. The process of componentization produces an equivalent component-based model for the watertight model.

Before repairing the hole of a given mesh *M*, the hole should be located. The boundary of a hole can be easily identified by searching a closed loop of boundary edges. We use two steps to realize the hole repairing to create a watertight mesh with a smooth surface, as shown in Fig. 5.

The first step is to fill the hole. The mesh of a hole is formed based on its boundary points and edges. After creating the hole mesh, it will be stitched to the mesh *M*, then it will be remeshed to obtain the mesh with the approximately uniform edge length. The remesh process takes the target edge length ε as input, and repeatedly splits edges longer than $4/3\,\varepsilon$, collapses edges shorter than $4/5\,\varepsilon$, and relocates vertices until the length of all edges is approximate to ε. In the study, we set the target edge length ε as $\frac{1}{|E(M)|}\sum_{e\in E(M)} length(e)$, where E(*M*) is the set of edges of the original mesh *M*.

The second step is to smoothen the created mesh using a surface fairing algorithm [31]. Its aim is to create a smooth surface patch that naturally follows the curvature around the hole. In the algorithm, let $\mathbf{x}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$ and Ω be the parameter domain, the energy function of the curvature of the created patch is defined as the following equation:

$$\mathrm{E}(\mathbf{x}) = \iint_{\Omega} k_1^2 + k_2^2 dudv\,, (u,v) \in \Omega, \tag{2}$$

where $\mathbf{x}(u,v)$ is subject to boundary constraints, k_1 and k_2 denote the principal curvatures. For fairing the surface, we need to minimize $E(\mathbf{x})$. Since Equation 2 is nonlinear, curvatures are replaced by second derivatives for linearization, leading to

$$\widetilde{\mathrm{E}}(\mathbf{x}) = \iint_{\Omega} \|\mathrm{x}_{uu}\|^2 + 2\|\mathrm{x}_{uv}\|^2 + \|\mathrm{x}_{vv}\|^2 dudv. \tag{3}$$

The corresponding Euler-Lagrange equation is $\Delta^2 \mathrm{x}(u,v) = 0$ in Ω, with suitable boundary constraints prescribing positions $\mathbf{x}(u,v)$ and normals $\mathbf{n}(u,v)$. Then translate it to a discrete triangle mesh by replacing $\mathbf{x}(u,v)$ with the vector of vertex coordinates $\mathbf{x} = [\mathbf{x}_1, \dots \mathbf{x}_n]^T$ and using the discretized Laplace-Beltrami operator. This leads to a linear bi-Laplacian system:

$$\mathbf{L}^2\mathbf{x} = 0 \tag{4}$$

that is solved for the optimal vertex positions $\mathbf{x}$, as show in Fig.5(d).

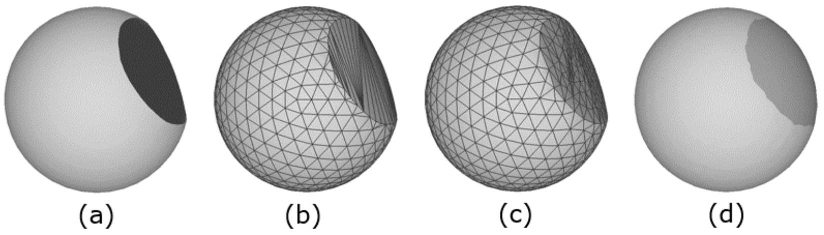


Figure 5: Surface repairing. (a) A part with a hole. (b) Fill the hole by connecting boundary points. (c) Remesh the hole. (d) Optimized shape after surface fairing.

For a segmented part with multiple holes, the holes will be repaired one by one, as shown in Fig. 6. Since the curvature of the hole boundary has a significant effect on the result of surface repairing, boundaries will be refined by adjusting the concave points near the boundary to obtain good repairing. The componentization is completed by repairing all holes of the segmented part, and eventually a watertight component is obtained, as shown in Fig.6.

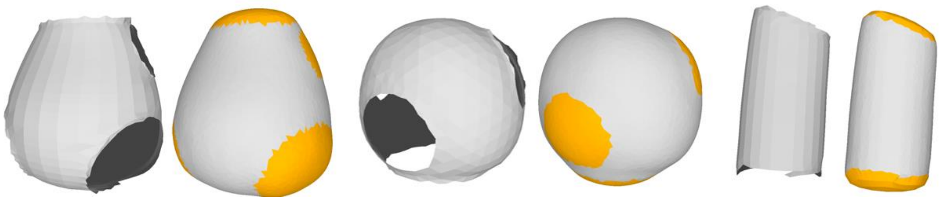


Figure 6: Repairing multiple holes (yellow color represents the created patch and grey color represents the original mesh).

5 PATTERN DESIGN WITH TEMPLATES

After componentization, the seam lines will be created based on the segmented components and design templates. At first, structure lines will be generated between each pair of connected parts.

Then, the best match template will be retrieved from the template library to generate seam lines on each component. Next, seam lines will be mapped to the original model, and finally 2D patterns will be obtained using surface flattening.

5.1 Structure Line Generation

The structure line is a special kind of seam line used to separate two parts of a toy model, such as the head and the body. It has the same function as the intersecting lines of the component-based model [1]. The intersecting line is gotten by calculating the intersection of two intersecting components. In the study, we can simply convert the boundary points of the segmented parts to the structure line. Similar to the intersection line, a structure line is formed by projecting a cubic spline curve to the 3D model surface. Therefore, it is necessary to set its control points.

For a boundary line of two connected segmented parts, we can set a random point as the first control point of the structure line. The next control point on the boundary can be located with a length L= $\sum\|E\|/N_c$, where E is the edge of the boundary, and N_c is the expected number of control points of the structure line. To make sure that some control points can be located at the regions with high curvatures so as to obtain a smooth structure line, we use the chord height as the constraint while searching the next control point. Fig. 7(a) shows the generated structure lines of a watertight model.

5.2 Template Retrieval

Template retrieval is to find the most similar template from the library so that the design knowledge of seam lines can be reused. Since components of 3D toy models are various, the similarity is considered both in shape and structure. For preprocessing, the principal component analysis (PCA) [32] is used to normalize the component, which makes components to be position and posture irreverent. A component descriptor $d(M)$ is used to describe the shape features f and the structure information of its related structure lines s of a component [1]:

$$d(M) = \{f\!:\{f_1,\ f_2, \ldots\ f_{20}\}, \{s\!:\{s_1, s_2, \ldots s_m\}.\}, \tag{5}$$

where f_i is the farthest distance from the center of the component along the i-th direction, s_i is the control points of the i-th structure line, and m is the number of related structure lines of component M. Therefore, the similarity between two components A, B is calculated by the distance between descriptor $d(A)$ and $d(B)$:

$$\text{shape similarity} = 100\left(1-\frac{\sum_{i=1}^{20}|f_i^A - f_i^B|}{2\sqrt{10}}\right), \tag{6}$$

$$\text{structure similarity } = 100\left(1-\frac{1}{2N_A}\sum_{n=1}^{N_A}\min(|C_n^A C_m^B|)\right), n, m = 1,2,\ldots,N_Q, \tag{7}$$

where C_n^A is the center of the n-th structure line s_n^A, and C_m^B is the center of the structure line s_m^B which is the closest to C_n^A in B.

To retrieve the most similar template of a segmented part of a model, we use the shape of its watertight component and its related structure lines (Fig. 7(b)) to generate the query descriptor $d(Q)$ and find its nearest $d(T)$ with max similarity in the template library. Fig. 7(c) shows the retrieved result of the selected part of the model. In the figure, four most matched templates with similar shapes and structures are found, and the user can select one template which is regarded as the most matched template.

5.3 Seam Line Mapping

After retrieving the most similar template, the seam lines of the template can be mapped to the segmented part of the original watertight model, and we use the watertight component as an intermediate. Firstly, the seam lines are mapped from the template to the component, then the seam lines of the component are mapped to the original model to generate the final design of seam lines.

To map seam lines from the template to the component, both of them are parameterized into the unit sphere to establish the mapping relation. Here, we adopt a robust parameterization method [33] to improve the quality of spherical parameterization. The method firstly simplifies the mesh into a tetrahedron by the flat-to-extrusive decimation strategy and maps the tetrahedron onto the sphere. Then the vertices are alternately inserted under the distortion control and global distortion optimization. Once spherical parameterization is completed, mapping relation will be established by projecting seam lines onto the unit sphere, and the seam lines can be mapped from the template to the component. Fig. 7(d) shows the seam lines of the component mapped from the most matched template. At last, the seam lines of the component will be transferred to the original model by mapping the control points to the nearest face of the model, as shown in Fig. 7(e).

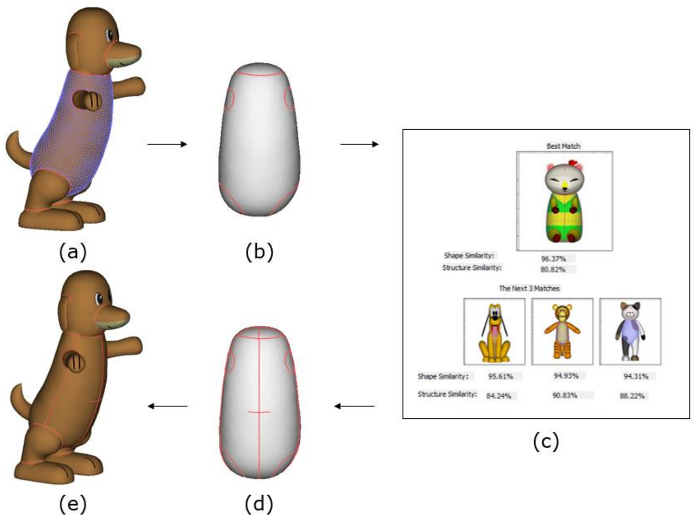

(a)
(b)
Best Match
Shape Similarity: 96.37%
Structure Similarity: 80.82%
The Next 3 Matches
Shape Similarity: 95.61% 94.93% 94.31%
Structure Similarity: 84.24% 90.83% 88.22%
(c)
(d)
(e)

Figure 7: Process of generating seam lines. (a) Structure lines of a watertight model. (b) The component converted from the selected part. (c) Template retrieval. (d) Mapping seam lines to the component. (e) Seam lines generated on the original model.

5.4 Pattern Development

Based on the above method of seam line generation, the seam lines of a toy model can be created automatically or interactively. For each segmented part, if the template with the highest shape and structure similarity is set as the most match one by default, seam lines of all segmented parts can be generated automatically. Otherwise, the process is an interactive way if the user wants to choose the template from several most matched templates by herself/himself. The generated seam lines can be further adjusted manually to improve the design of seam lines.

After all seam lines are generated, the surface of a model is divided into several closed surface patches. To obtain 2D patterns, we adopt a surface flattening method based on a mass-spring model with multilevel-mesh acceleration [34]. The quality of flattened patterns can be checked using the deformation rate.

6 EXPERIMENTAL RESULTS

We have implemented our method on the Intel i5 CPU core and GTX 1660 super graphics card under Windows 10 (64-bit). The training and testing of rough segmentation use Python 3.6, PyTorch 1.1.0, and CUDA 10.0, and other processes of our method are implemented with C++ and OpenGL. Some experiments of deep learning-based segmentation, componentization and pattern design were conducted. In addition, we presented a method of editing the watertight model based on segmentation and componentization.

6.1 Dataset, Training and Testing

In the study, we collected 1,254 different component-based models of plush toys designed by EasyToy software [28]. These models were designed by professional designers, some of which are shown in Fig. 8. Among the data, some models were built from another model by manually adjusting the shapes and postures of components. To augment the data, the model is up-sampled and down-sampled into different triangle resolutions, and thereby generating 1,254×3=3762 data in total. These models were split into 8:2 for training and test. Since the model is consists of components, we can easily classify its components into 10 categories, including the head, body, tail, arm, palm, leg, foot, ear, nose, and others. Then they can be semantically labeled according to their categories. Since we focus on the segmentation of the model surface, the invisible surfaces caused by the intersection of components are ignored while generating training and test data. Thus, the data of component-based models can be used for training the segmentation of watertight models.

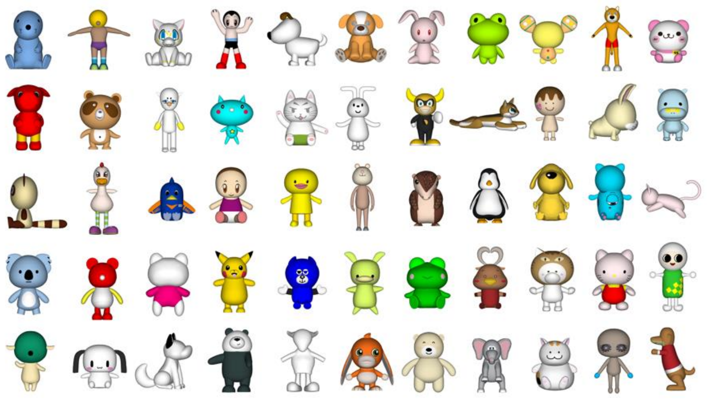


Figure 8: Some toy models in dataset.

For training, the optimizer selected in the network is the ADAM optimizer [35], which designs independent adaptive learning rates for different parameters by calculating the first and second moment gradient. The batch size is set to 4, and the step size is set to 20. The learning rate is set to 5×10^{-4} initially and decays in each step size, not less than 10^{-5}, and the decay rate is set to 0.8. We use the cross-entropy as loss function, which can be calculated by the code "total_loss = torch.nn.functional.nll_loss(predict, target)" in pytorch. It took around 6 minutes to train for each epoch and we trained the model for 500 epochs in total. The changes of loss value during the training epochs is shown in Table 1.

Epochs	*Loss*
10	0.7940
200	0.3646
400	0.1589
500	0.1352

Table 1: Loss value during the training epochs.

After training the model, the weights of the model are stored as a ".pth" file. To segment a watertight mesh, firstly, the weights of the trained model are loaded and the points are extracted from the mesh. After the predicted result is obtained by the model, the rough segmentation result is then mapped from points to the mesh surface. Finally, the segmentation is refined by graph cut. The abridged code is shown in code listing 1:

```
torch.save(model, “model_path.pth”) #save trained model
#for testing
point_data = mesh.get_vertex()
model = torch.load(“model_path.pth”) #load trained model
model.eval() #begin prediction
predict_result = model(point_data)

mesh.map_to_faces(predict_result) #map the prediction to mesh
mesh.graph_cut() #refine segmentation by graph cut
```

Listing 1: Main process of PointNet++-based segmentation.

Some results of segmentations are shown in Fig. 9. We can find that satisfactory segmentation results can be obtained using the methods of deep learning and graph cut. In the figure, the running time of segmentation and the number of faces of the model are shown. It shows that the runtime of segmenting a model is usually less than 15 seconds. We found that the time of the segmentation is dominated by the computation of the minimum cut. It is possible to shorten the segmentation time if the algorithm of graph cut is further optimized. For the runtime of componentization, it is less than 1.0s for each segmented part.

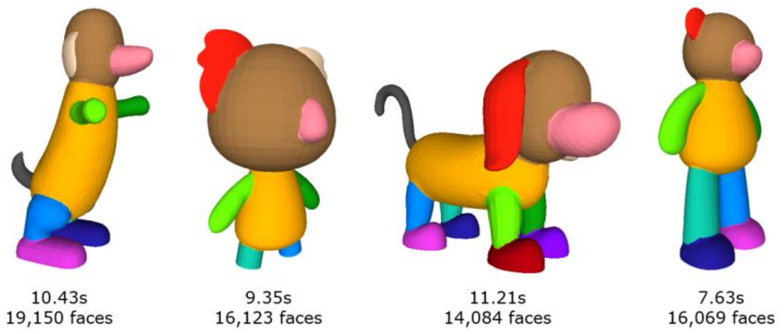
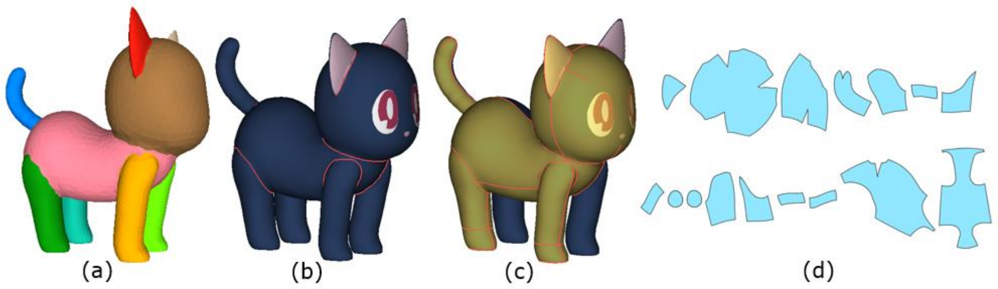


Figure 9: Some examples of segmentation.

6.2 Pattern Design of Watertight Models

The process of pattern design for a watertight model is shown in Fig. 10. For a 3D model to be designed, firstly the segmentation and componentization process is conducted to create segmented parts and watertight components (Fig. 10(a)), then structure lines are generated on the original model (Fig. 10(b)). For each component, the most matched template is retrieved (Fig. 10(e)). Next, seam lines are mapped from the template to the component, and then transferred to the original model (Fig. 10(c)). Finally, 2D patterns are obtained by flattening 3D surface patches (Fig. 10(d)). The system is able to deal with both the component-based model and the watertight model. For a component-based model, the procedure of segmentation and componentization is unnecessary, and the process is the same as our previous method.



	Left ear	Head	Left hand	Body	Left leg	Tail
Created component						
Template mapped						
Shape similarity	91.01%	84.18%	90.56%	89.34%	91.08%	92.53%
Structure similarity	72.89%	86.35%	90.72%	84.26%	70.07%	97.12%

(e)

Figure 10: Process of pattern design. (a) Segmentation and componentization. (b) Generation of structure lines. (c) Generation of seam lines using templates. (d) Designed 2D patterns. (e) Templates selected for all components.

The toy model in Fig. 10 is a merged model from a component-based model used in our previous work. The result of template retrieval and seam line mapping might be different from the original model, since much more templates have been added to the library in current experiments.

To verify the proposed method, various types of watertight models were selected for pattern design, as shown in Fig. 11. We chose two of them to produce plush toys, as shown in Fig. 12. The physical toys were manufactured based on the flattened patterns without any modification. In the figure, we can see that the shape of the real toy is very similar to that of the 3D model, demonstrating the effectiveness and feasibility of our method.

6.3 Watertight Model Editing

Editing a watertight model is usually complex and time-consuming. With the aid of segmentation and componentization, a watertight model can be easily edited. The editing process is illustrated in Fig. 13. Firstly, a watertight model will be segmented, then select a segmented part of the model to be modified (highlighted in blue in Fig. 13(b)), and the part will be componentized into a watertight component, and the hole of the remaining part will be repaired. The watertight component can be edited freely by translating, rotating and scaling (Fig. 13(c)). Finally, the component will be merged into the remaining part to form a new model (Fig. 13(d)). Fig. 13(e) and 13(f) show two examples of editing watertight models. The experiments demonstrated that, the shape and posture of a watertight model can be adjusted conveniently based on the method of segmentation and componentization.

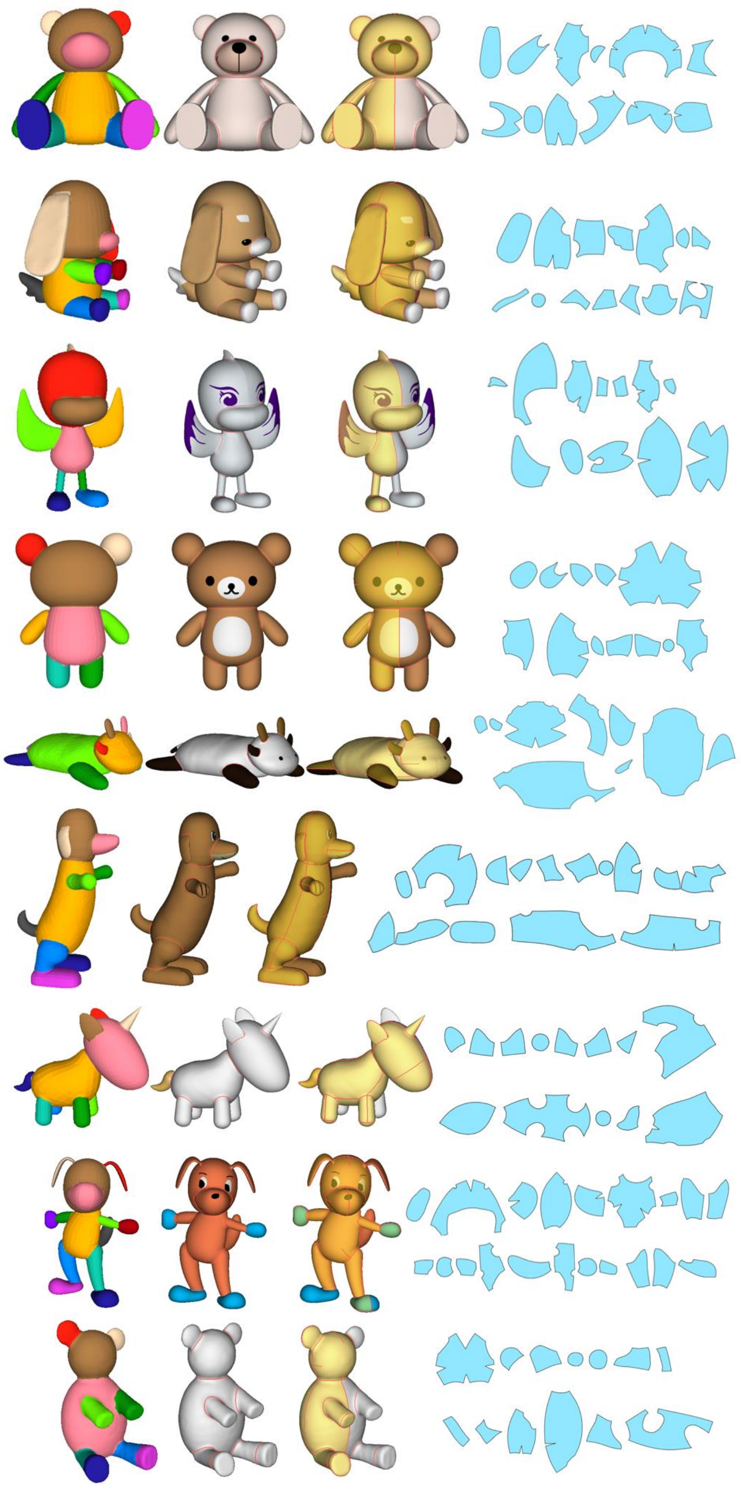


Figure 11: Some examples of pattern design. (From left to right, each row shows the model with segmentation, structure lines, seam lines, and 2D patterns).

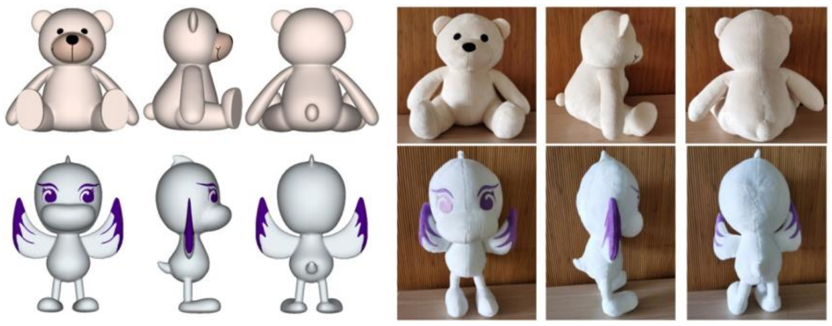


Figure 12: Two physical plush toys produced based on patterns designed by our method.

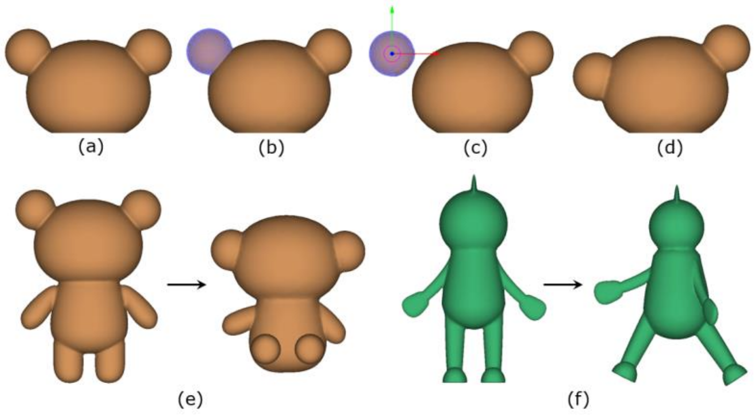


Figure 13: Editing a watertight model. (a) A watertight model. (b) A segmented part is selected. (c). Componentize the selected part and repair the hole of the remaining part. (d) Modified model. (e, f) Two examples.

7 CONCLUSIONS

In this paper, a data-driven method is presented to design patterns for watertight models of plush toys. We use a divide-and-conquer strategy to convert a watertight model into a component-based model so that its patterns can be designed using component-based templates. A segmentation and componentization method is proposed to decompose the watertight model into components, and the method can also be used to modify the shape and structure of a watertight model freely and easily. Experimental results demonstrated the feasibility and usability of the proposed method and the produced physical toys are satisfactory.

Future work remains to be done. At the current stage, deep learning is used for the segmentation in the process of pattern design, whereas its potential in pattern design has not been fully explored. For example, deep learning might be able to generate seam lines directly from the 3D model without the aid of design templates, or generate multiple design schemes for one model. Another problem is that the influence of textures of the 3D model is ignored in the study. If the textures are recognized and taken into account while generating seam lines, a more reasonable design will be obtained.

ACKNOWLEDGEMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant 61972340, 61732015 and 51775492.

Yinghan Jin, http://orcid.org/0000-0002-2778-4314

REFERENCES

[1] Jin, Y.; Feng, W.; Zeng, L.; Zhang, D.: An Efficient Pattern Design Method for Plush Toys Using Component-Based Templates, Computer-Aided Design, 128, 2020, 102911. https://doi.org/10.1016/j.cad.2020.102911

[2] Igarashi, Y.; Igarashi, T.: Pillow: Interactive flattening of a 3D model for plush toy design, International Symposium on Smart Graphics. Springer, Berlin, Heidelberg, 2008, 1-7. https://doi.org/10.1007/978-3-540-85412-8_1

[3] Mori, Y.; Igarashi, T.: Plushie: an interactive design system for plush toys, ACM SIGGRAPH 2007 papers, 2007, 45-es. https://doi.org/10.1145/1275808.1276433

[4] Umetani, N.; Kaufman, D. M.; Igarashi, T.; et al.: Sensitive couture for interactive garment modeling and editing, ACM Trans. Graph., 30(4), 2011, 90. https://doi.org/10.1145/2010324.1964985

[5] Bartle, A.; Sheffer, A.; Kim, V. G.; et al.: Physics-driven pattern adjustment for direct 3D garment editing, ACM Trans. Graph., 2016, 35(4), 2016, 50:1-50:11. https://doi.org/10.1145/2897824.2925896

[6] Skouras, M.; Thomaszewski, B.; Kaufmann, P.; et al.: Designing inflatable structures, ACM Transactions on Graphics (TOG), 33(4), 2014, 1-10. https://doi.org/10.1145/2601097.2601166

[7] Montes, J.; Thomaszewski, B.; Mudur, S.; Popa, T.: Computational design of skintight clothing, ACM Transactions on Graphics (TOG), 39(4), 2020, 1-12. https://doi.org/10.1145/3386569.3392477

[8] Katz, S.; Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts, ACM transactions on graphics, 22(3), 2003, 954-61. https://doi.org/10.1145/882262.882369

[9] Liu, R.; Zhang, H.: Segmentation of 3D meshes through spectral clustering, in 12th Pacific IEEE Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings. 2004.

[10] Rodrigues, R.S.; Morgado, J. F.; Gomes, A. J.: A contour-based segmentation algorithm for triangle meshes in 3D space, Computers & Graphics, 49, 2015, 24-35. https://doi.org/10.1016/j.cag.2015.04.003

[11] Lavoué, G.; Dupont, F.; Baskurt, A.: A new CAD mesh segmentation method, based on curvature tensor analysis, Computer-Aided Design, 37(10), 2005, 975-87. https://doi.org/10.1016/j.cad.2004.09.001

[12] Wang, Y.-Y.; Zhang, H.; Wang, Y.; Mo, T.: Mesh Segmentation Based on Feature Point and Core Extraction, Modern Computer, 2009, 01.

[13] Zhou, Y.; Huang, Z.: Decomposing polygon meshes by means of critical points, in 10th IEEE International Multimedia Modelling Conference, 2004. Proceedings. 2004.

[14] Agathos, A.; Pratikakis, I.; Perantonis, S.; Sapidis, N. S.: Protrusion-oriented 3D mesh segmentation, The Visual Computer, 26(1), 2010, 63-81. https://doi.org/10.1007/s00371-009-0383-8

[15] Au, O.K.-C.; Tai, C.-L.; Chu, H.-K.; Cohen-Or, D.; Lee, T.-Y.: Skeleton extraction by mesh contraction, ACM transactions on graphics, 27(3), 2008, 1-10. https://doi.org/10.1145/1360612.1360643

[16] Kalogerakis, E.; Hertzmann, A.; Singh, K.: Learning 3D mesh segmentation and labeling, ACM Transactions on Graphics, 29(4), 2010, 1-12. https://doi.org/10.1145/1778765.1778839

[17] Guo, K.; Zou, G.; Chen, X.: 3d mesh labeling via deep convolutional neural networks, ACM Transactions on Graphics, 35(1), 2015, 1-12. https://doi.org/10.1145/2835487

[18] Xie, Z.; Xu, K.; Liu, L.; Xiong, Y.: 3D shape segmentation and labeling via extreme learning machine, in Computer Graphics Forum, 2014. https://doi.org/10.1111/cgf.12434

[19] Kalogerakis, E.; Averkiou, M.; Maji, S.; Chaudhuri, S.: 3D shape segmentation with projective convolutional networks, in proceedings of the IEEE conference on computer vision and pattern recognition, 2017. https://doi.org/10.1109/CVPR.2017.702

[20] Han, Z.; Lu, H.; Liu, Z.; Vong, C.-M.; Liu, Y.-S.; Zwicker, M.; et al.: 3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation. IEEE Transactions on Image Processing, 28(8), 2019, 3986-99. https://doi.org/10.1109/TIP.2019.2904460

[21] Qi, C.R.; Yi, L.; Su, H.; Guibas, L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in Advances in neural information processing systems, 2017.

[22] Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems, 31, 2018, 820-30.

[23] Qi, C.R.; Su, H.; Mo, K.; Guibas, L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[24] Lahav, A.; Tal, A.: MeshWalker: deep mesh understanding by random walks, ACM Transactions on Graphics, 39(6), 2020, 1-13. https://doi.org/10.1145/3414685.3417806

[25] Feng, Y.; Feng, Y.; You, H.; Zhao, X.; Gao, Y.: Meshnet: Mesh neural network for 3d shape representation, in Proceedings of the AAAI Conference on Artificial Intelligence, 2019. https://doi.org/10.1609/aaai.v33i01.33018279

[26] Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; Cohen-Or, D.: Meshcnn: a network with an edge, ACM Transactions on Graphics, 38(4), 2019, 1-12. https://doi.org/10.1145/3306346.3322959

[27] Chen, X.; Golovinskiy, A.; Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Transactions on Graphics, 28(3), 2009, 1-12. https://doi.org/10.1145/1531326.1531379

[28] Liu, Y.-J.; Ma, C.-X.; Zhang, D.-L.: EasyToy: plush toy design using editable sketching curves. IEEE Computer Graphics and Applications, 31(2), 2009, 49-57. https://doi.org/10.1109/MCG.2009.147

[29] Rodrigues, R.S.; Morgado, J. F.; Gomes, A. J.: Part-based mesh segmentation: a survey, in Computer Graphics Forum. 2018. https://doi.org/10.1111/cgf.13323

[30] Cormen, T.; Leiserson, C.; Rivest, R.: Introduction to Algorithms, Second Edition. 2012.

[31] Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P.; Lévy, B.: Polygon Mesh Processing, 2010: Polygon Mesh Processing. https://doi.org/10.1201/b10688

[32] Vranic, D.V.; Saupe, D.: 3D model retrieval (Ph.D. thesis), University of Leipzig; 2004.

[33] Hu, X.; Fu, X.-M.; Liu, L.; Advanced hierarchical spherical parameterizations. IEEE transactions on visualization and computer graphics, 24(6), 2017, 1930-41. https://doi.org/10.1109/TVCG.2017.2704119

[34] Zhang, D.; Li, J.; Wang, J.: Design Patterns of Soft Products Using Surface Flattening. Journal of Computing and Information Science in Engineering, 18(2), 2018, 021011. https://doi.org/10.1115/1.4039476

[35] Kingma, D.P.; Ba, J. A.: A method for stochastic optimization. Computer Science, 2014.