The Visual Computer manuscript No.
(will be inserted by the editor)

Generative Design of Decorative Architectural Parts

Yuzhe Zhang · Chan Chi Ong · Jianmin Zheng · Seng-Tjhen Lie · Zhendong Guo

Received: date / Accepted: date

Abstract This paper presents a method for generative design of decorative architectural parts such as corbel, moulding and panel, which usually have clear structure and aesthetic details. The method is composed of two components: offline learning and online generation. The offline learning trains a 2D CurveInfoGAN and a 3D VoxelVAE that learn the feature representations of the parts in a dataset. The online generation proceeds with an evolution procedure that evolves to product new generation of part components by selecting, crossing over, and mutating features, followed by a feature-driven deformation that synthesizes the 3D mesh representation of new models. Built upon these technical components, a generative design tool is developed, which allows the user to input a decorative architectural model as a reference and then generates a set of new models that are "more of the same" as the reference and meanwhile exhibit some "surprising" elements. The experiments demonstrate the effectiveness of the method and also showcase the use of classic geometric modeling and advanced machine learning techniques in modeling of architectural parts.

This work is supported by the Ministry of Education, Singapore, under its MoE Tier-2 Grant (2017-T2-1-076).

Y. Zhang, C.C. Ong, J.Zheng (corresponding author), Z. Guo
School of Computer Science and Engineering, Nanyang Technological University, Singapore
E-mail: zhang.yz@ntu.edu.sg, c170117@e.ntu.edu.sg, asjmzheng@ntu.edu.sg, zhendong.guo@ntu.edu.sg

S.-T. Lie
School of Civil and Environmental Engineering, Nanyang Technological University, Singapore
E-mail: cstlie@ntu.edu.sg

Keywords Decorative architectural parts · Generative design · 3D modeling · Machine learning · Evolution principle

1 Introduction

This paper considers the design of decorative architectural parts, which are often created beautifully and delicately to decorate or reflect the characteristic of the design in industry [1]. Particularly, we focus on three typical architectural categories: corbel, moulding and panel. A corbel is a structural piece protruding from a wall to support a superincumbent weight. A moulding is a shaped strip of material with various profiles used to cover transitions between surfaces, especially in a cornice. A panel is a flat or curved component to capture a "floating" decoration within a sturdy frame that sets into the surface of a door, wall or ceiling. Usually these three models have clear structure and aesthetic details.

While CAD tools are widely used for traditional design and manufacturing, they are not necessarily convenient for exploration of decorative architectural shapes, especially in terms of cost efficiency and creativity essential for design. Considering the shortage of adequate software/tools for exploration, inspiration and creation in modeling and the availability of well-designed models with varying shapes and styles in online warehouse, we aim to develop efficient design techniques for creating 3D decorative architectural models from examples. This may provide a cost-effective and easy-to-use solution to the design of architectural models.

With advance of machine learning and data analytics, high level geometric design techniques such as explorative modeling, example-driven synthesis and generative design are emerging and becoming popular. In particular, with simple design specifications, generative design is able to produce a series of design options using artificial intelligent techniques [2]. This thus provides a convenient way for exploration, inspiration and creation. When we apply generative design, we want the generated models to contain elements of unexpectedness or surprise, which brings inspiration to us. Meanwhile we want the new models to be more of the same as the examples and specifications, particularly retaining geometric structure and style of architectural parts. These two requirements actually contradict each other, hence imposing a technical challenge [3].

This paper is an extension of "creative corbel modeling" we presented at 2020 International Conference on Cyberworlds [4]. We propose a framework to realize the generative design of decorative architectural parts that include corbel, moulding and panel. The framework combines example-based modeling, geometric processing, machine learning and evolution. Specifically, we collect a few well-designed decorative architectural parts and decompose them into semantically meaningful components to build a dataset. The dataset serves as an initial design space. The user can provide an architectural part or specify an example in the dataset as input. The evolution based algorithm will generate a set of novel models. To make the generated models meet the above-mentioned two requirements, we carefully construct high level representations of the collected part models, design a 2D curve Information Maximizing Generative Adversarial Net (InfoGAN) and a 3D Variational Autoencoders (VAE) model for learning the features of the models, and design a generation algorithm involving evolution principle and feature-driven deformation for generating new models. The main contributions of the paper include the novel framework for generative design of architectural parts and its underlying evolution based generation algorithm.

The rest of the paper is organized as follows. Section 2 reviews some relevant work. Section 3 gives an overview of the proposed method, followed by model processing in Section 4 and evolution-based design in Section 5. Section 6 presents the experiments to evaluate the proposed method and Section 7 concludes the paper.

2 Related Work

Shape design and 3D modeling have been extensively studied. Various techniques have been developed. This section briefly reviews example-based synthesis, generative design and creative modeling, which are relevant to our work.

2.1 Example-driven synthesis

The idea of synthesizing new 3D models from parts segmented from example shapes was initially proposed by Funkhouser et al. [5]. It was then extended to the processes of part retrieval and composition, and further to the evolution from a set of 3D models to obtain new generations of fit and diverse offsprings [6]. Here the word "fit" emphasizes plausibility with which a design will generate for instance airplane-like shapes from airplane models, and "diversity" implies sort of surprising designs that are not stuck in an elite population. In this approach, however, machine does not really learn knowledge.

Kalogerakis et al. [7] proposed a generative probabilistic model of shape structure, which relates probabilistic relationships between geometric and semantic properties of shape components to learned latent causes of structural variability. Thus plausible new shapes can be synthesized by sampling in the learned spaces. Sung et al. proposed "CompletementMe", an incremental synthesis system for constructing a shape by progressively suggesting complementary components and their placement [8]. To use unlabeled parts and predict a probability distribution over the space of part embeddings, embedding and retrieval networks are jointly trained, which first index parts by mapping them to a low-dimensional feature space and then map partial assemblies to appropriate complements. As a result, individual new parts are suggested one by one to complement the partially constructed shape with minimal user input.

2.2 Generative design

Machine learning, particularly deep neural networks, has achieved great success in imaging processing, computer vision and natural language processing [9,2]. A steady stream of methods applying deep learning models in the geometric field has also been proposed for generative design of 3D shapes. A simple extension from 2D image synthesis to 3D shape generation is to directly apply machine learning to 3D voxels. In 3D-GAN [10], Wu et al. combined volumetric CNN and GAN to map a 64^3 volume to a 200D latent vector. The voxel based representations usually require huge memory and computation costs especially when the volumetric resolution is high. When the geometry is sparse, octrees are often used to reduce the costs [11].

Compared to standard GAN [2], InfoGAN (Information Maximizing GAN) [12]), which introduces additional parameters – latent codes automatically learned by the networks, allows some control over the generated data. VAE [9] is another class of generative models, which encodes latent variables to describe the data. Both InfoGAN and VAEs could recreate the input from the latent space in a compressed way. Compared to InfoGAN where the discriminator is calibrated on latent codes, VAE is on real data and good at feature dimension reduction.

Shape representation and encoding are important in design. Various schemes have been proposed for the representation and encoding of 3D shapes. Su et al. [13] converted 3D shapes into multi-view images by multiple projections and developed a neural network for 3D shape recognition. Qi et al. proposed PointNet [14] and PointNet++ [15] for 3D classification and segmentation. Learning from irregular point clouds is challenging, especially for relatively complex geometry. In [16], Chen and Zhang proposed to learn implicit fields for generative shape modeling. A signed distance function is reconstructed to distinguish the inside and outside regions of a closed shape. The generative model can be applied to various applications including shape autoencoding, generation, interpolation, completion, and single-view reconstruction. A shape could also be encoded as the deformation of a mesh template model [17]. This simplifies the specification of vertex connectivity of the shape, but meanwhile limits the topological and geometric complexity of generated shapes. With multi-chart representations [18], AtlasNet generates a shape as a collection of patches, each of which is parameterized to a 2D domain as an atlas.

Man-made shapes are often highly structured. This motivates decoupling of geometry and structure. The structure-aware generative design pays attention to high-level shape abstractions [19,20,21,22]. Li et al. trained independent networks for structure and geometry and particularly proposed a generative recursive autoencoder for structure using Recursive Neural Networks [19]. Wang et al. [20] introduced a global-to-local generative architecture where GAN is built for structure and conditional autoencoder is augmented to refine the structure at part level. Wu et al. [21] proposed a structure-aware generative model (SAGNet) for 3D shape design, in which the part structure and geometry are jointly learned and fused into a single latent code to intertwine two types of features. Gao et al. proposed a two-level VAE that consists of a PartVAE learning a deformable model of part geometry and a structured part VAE jointly learning part structure and geometry [22].

2.3 Creative modeling by evolution

Evolutionary algorithms (EAs) are a class of popular algorithms in computational intelligence. They mimic biological evolution in nature and are good at producing surprise via the operations of selection, mutation and cross-over. Thus EAs are good tools for creative modeling. To realize this, the contents should be encoded and the cross-over and mutation operators should be appropriately designed to allow the contents to evolve and produce surprises. Moreover, controllability is implemented by the selection process, where a fitness function determines whether a new creation is allowed to survive and further produce offsprings [3].

Xu et al. [6] proposed to perform EA-based stochastic object modeling with a design gallery to achieve set evolution. The set evolution starts with an initial population of 3D objects belonging to a category. Then stochastic mutation of object components and cross-over between objects drive the evolution and produce offspring generations. During the process, user preference is used to determine the fitness for the evolution in selecting shapes from the gallery that are deemed to be fit to breed the next generation.

3 Overview of the Proposed Method

The basic problem addressed in the paper can be described as follows: given a 3D decorative architectural model inputted by the user, we want to develop an algorithm that automatically generates a group of new models that are similar to the input and also show some "unexpected" effects.

Our proposed solution is illustrated in Figure 1 and consists of model collection and processing, offline learning, and online generation.

- We first collect a set of decorative architectural models designed by the professional and convert them into triangular mesh representation. Then we decompose the models into components. Each corbel model is decomposed into base, main body and decoration. Each moulding or panel model is decomposed into main body and decoration. The decoration components are shared across the three architectural categories. Finally we extract features for each component. The components together with the features form the underlying representation for our design task.
- In the offline learning stage, we develop 2D CurveInfoGAN and 3D VoxelVAE. The 2D CurveInfoGAN model is trained for feature curves of main body for corbel, moulding and panel, respectively. New feature curves could be generated and controlled by

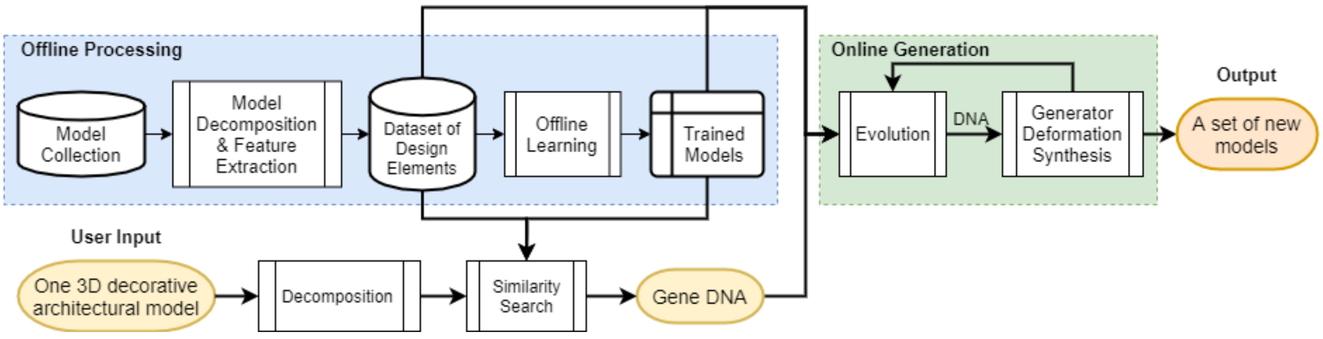


Fig. 1: The workflow of the proposed design.

sampling and adjusting the variables in the latent space. The 3D VoxelVAE model is trained to encode voxelized decoration component into a 64D latent space where similar and dissimilar decoration models could be easily retrieved.

– In the online generation stage, we encode architectural models by gene vectors using the dataset and two feature latent spaces created earlier. We apply evolution algorithm to generate new models via crossover, mutation and selection from the one provided by the user. The new models are expected to meet the requirements of fitness and diversity. The final mesh models are synthesized by applying feature-driven deformation to generated main bodies, decorations and bases.

The detailed processes of these stages are presented in the next two sections.

4 Model Processing

We have collected 125 corbel models, 254 moulding models and 193 panel models. All these models are triangular meshes in OBJ format, designed by industry designers. To facilitate generative design, we decompose parts, and extract structure and feature information.

4.1 Decomposition

According to the characteristics of decorative architectural parts, the parts can be decomposed into semantically meaningful components. The decomposition can be done manually or by partitioning algorithms [8]. In particular, a corbel can be decomposed into base, main body and decoration. The base usually has very regular shape and is used to touch the supporting structure such as a wall or ceiling. The main body stands for the overall shape of the corbel. The decoration accounts for geometric texture or details (see Figure 2).

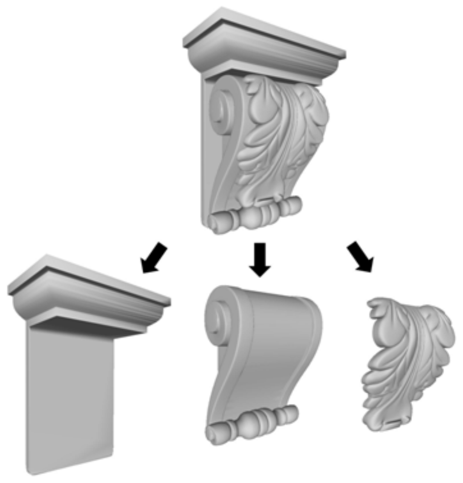


Fig. 2: Decomposition of a corbel model into base (left), main body (middle) and decoration (right) components.

A moulding model can be decomposed into main body and decoration as shown in Figure 3. Similarly, a panel model can also be decomposed into main body and decoration as shown in Figure 4.

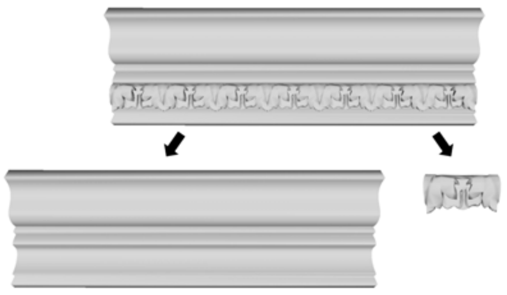


Fig. 3: Decomposition of a moulding model.

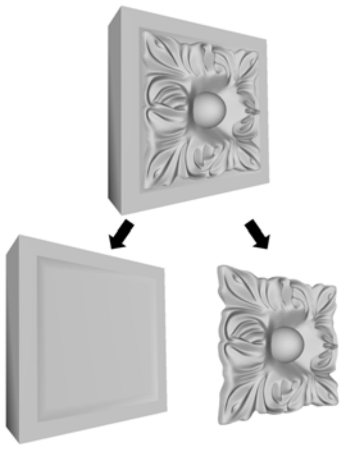


Fig. 4: Decomposition of a panel model.

4.2 Geometric Feature Extraction

For a main body, its front face corresponds to the decoration component and is usually characterized by a curved profile. We propose to fit a planar cubic B-spline curve to the curved profile, as shown in Figure 5. The starting and ending points of the B-spline curve are used to guide the alignment of the decoration to the main body.

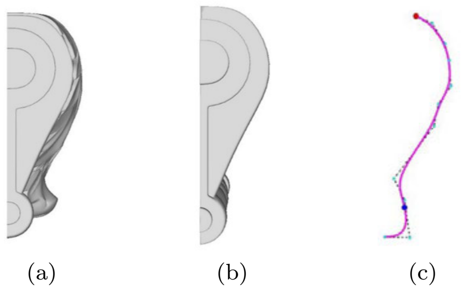


Fig. 5: Feature extraction of a corbel's main body: (a) Main body with decoration; (b) Main body without decoration; (c) a B-spline feature curve, and the starting and ending points (in red and blue, respectively) for aligning decoration.

Similarly, for the decoration, we can also fit a cubic B-spline curve to its back face that corresponds to the main body (see Figure 6). These B-spline curves serve as the feature curves for the main body and decoration components. They will be used to deform both components in the process of synthesizing new models.

For a base component that only corbel models have, the feature is captured by the width, height and depth of the component, together with a reference point as illustrated in Figure 7. The base component works as a connector to attach the main body to a wall or ceiling. The reference point is a pivot point for the main body to be aligned to the base. The height and depth values are not necessarily always available since some base components do not have a top or back.

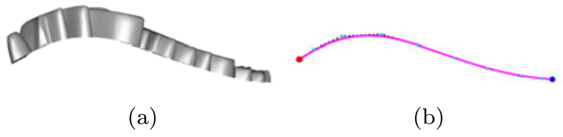


Fig. 6: Feature extraction of a decoration component: (a) Decoration component; (b) Feature curve of the decoration component.

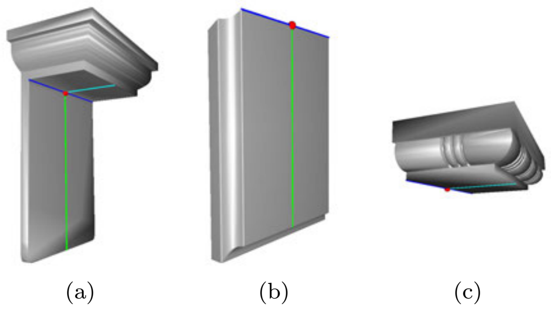


Fig. 7: Feature extraction of 3 base components: (a) Features of a base component; (b) Features of a base component without top; (c) Features of a base component without back.

4.3 Design Space

Thereafter, each model can be decomposed into components. Each of these components has corresponding feature. We treat each component and feature as a design element or building block, and assembly them according to their categories to form a design space. Specifically, we have three decorative architectural part collections $\{M_i^{corbel}\}_{i=1}^{125}$, $\{M_i^{moulding}\}_{i=1}^{254}$ and $\{M_i^{panel}\}_{i=1}^{193}$. Our design space is composed of the following elements:

- Main body: a set of triangular meshes $\{Mmb_i^x\}$ and a set of feature curves in B-spline representation $\{Cmb_i^x\}$, where the superscript 'x'={corbel, moulding, panel} denotes the part category.
- Decoration: a set of triangular meshes $\{Md_i\}$ and a set of feature curves in B-spline representation $\{Cd_i\}$.

– Base: a set of triangular meshes $\{Mb_i\}$ and a set of features in the form of reference points together with width, height and depth $\{Cb_i\}$.

5 Evolution-Based Design

We now present our evolution-based design, the core techniques of which include machine learning, generation by evolution and feature-driven deformation.

5.1 Offline Learning

To facilitate generating new models by evolution principle, we develop 2D CurveInfoGAN and 3D VoxelVAE models and train low dimensional vectors to represent feature curves and decorations.

5.1.1 2D CurveInfoGAN

In the standard GAN, the generator G maps an arbitrary noise distribution to the data distribution while the discriminator D distinguishes candidates produced by the generator G. G and D improve during training by competing with each other. However, the noise input z of the standard GAN has no clear interpretation. To solve this issue, the Information Maximizing Generative Adversarial Nets (InfoGAN) [12] was proposed, which splits the input of the Generator into two parts: the random noise vector and a new latent code vector c. InfoGAN is a useful framework that learns a latent code representation to control varying factors related to the input dataset. InfoGAN's objective can be expressed as the following equation by adding a regularization term L_I to GAN's objective function.

$$\min_{G,Q}\max_{D} V_I(D,G) = V(D,G) - \lambda_I L_I(G,Q) \tag{1}$$

where

$$\begin{aligned} V(D,G) &= E_{x\sim P_{data}}[logD(x)] + E_{z\sim P_z}[log(1-D(G(z)))] \\ L_I(G,Q) &= E_{x\sim P_G}[E_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c), \end{aligned}$$

Q is the auxiliary distribution for approximating $P(c|x)$, $H(c)$ is the entropy of the latent codes, and λ_I is the regularization constant and is typically set to 1. InfoGAN regularizes c by maximizing mutual information L_I between c and the generated data.

The structure of our 2D CurveInfoGAN adapts from the BézierGAN [23], where there is an additional parameter transforming layer and a B-Spline layer in the generator network (see Figure 8). For each feature curve, we sample a sequence of 192 points. The discriminator takes 2D coordinates of the sample points as data representation x. The 2D CurveInfoGAN adds a regularization term $R(G)$ to the InfoGAN objective:

$$\min_{G,Q}\max_{D} V(D,G) - \lambda_I L_I(G,Q) + \mu_R R(G) \tag{2}$$

where μ_R is a tradeoff factor (we set $\mu_R = 0.5$ in our experiments), and the term $R(G)$ is to regularize the cubic B-Spline's control points and knots, which is expressed as

$$R(G) = \frac{1}{N}\sum_{i=1}^{N}\|P_i - P_{i-1}\| + \max_i \|P_i - P_{i-1}\| + \frac{1}{N+4}\sum_{i=0}^{N+5}\left|u_i - u_i^0\right| \tag{3}$$

Here we assume that the B-spline curve has $N+1$ control points P_i, and u_i are the knots. The first two terms of $R(G)$ are the average and maximum Euclidean distances between two adjacent control points and the last term is to make knots close to the initial knots u_i^0. With the B-Spline layer, the B-Spline curve is learned and then passed into the discriminator. This ensures that both the synthesized data and real data are the same when passed into the discriminator for identification.

In our database, there are 78 main body feature curves for corbel collection, 125 for moulding collection and 61 for panel collection. For each category, we will learn an InfoGAN model. However, the number of main body feature curves is insufficient for training a good model. Hence data augmentation is carried out to increase the diversity of feature curves. Specifically, for each control point of a B-spline feature curve, we first assign it a vector that is the normal of the curve at the point nearest to the control point. Then we move the control point along the normal direction by an offset value in a range between -2 to 2 units. This results in the change of the control points and thus the shape of the feature curve. The augmentation randomly selects control points for offsetting. The augmentation process is applied to all feature curves. Eventually we produce 2989 feature curves for corbel, 4187 for moulding and 2413 for panel.

The dimension of the latent code L_{2D} is specified to be 3. Then 3 control variables are provided to control the shape of the generated curve. As shown in Figure 9, the latent codes provide a controller for generating the feature curves. Figure 9(a) shows a series of curves generated by setting $L_{2D}^0 = (0.0, 0.1, ..., 1.0)$ with $L_{2D}^1 = L_{2D}^2 = 0.5$. Similarly, in Figure 9(b) and (c), we sample the value of the second latent code, and the third latent code from 0.0 to 1, respectively. Each latent code has different controls on the shape of the generated curves, which help to generate desired curves

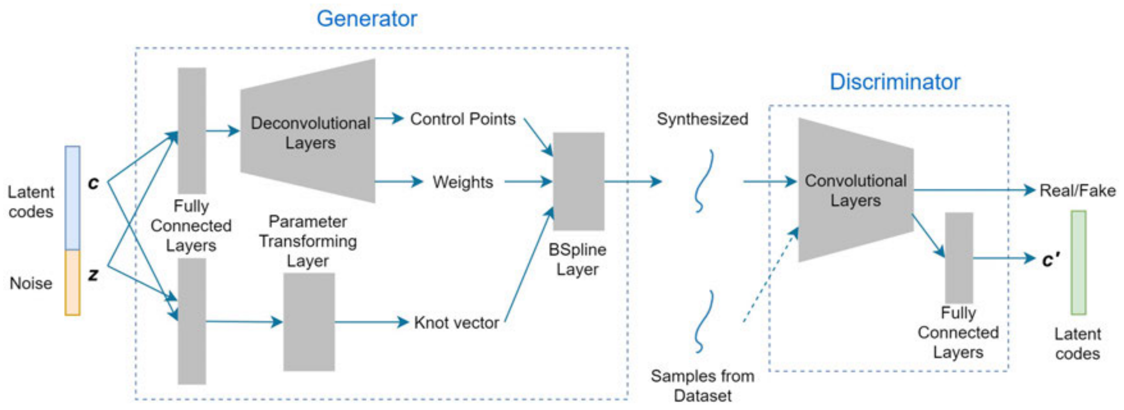


Fig. 8: Architecture of 2D CurveInfoGAN.

in a high level and also narrow down the sampling space for the evolution process.

5.1.2 3D VoxelVAE

VAE is a neural network that allows machine to learn a compact representation of data and generate new instances or reproduce the input by sampling in the latent space. To compress the representation of decoration component into a low dimensional space so that a quick enquiry of similar or dissimilar decoration could be made, we design a 3D VoxelVAE that is a variant model of VAE. In VoxelVAE, there is an additional voxel conversion layer before the data is passed into the encoder. The meshes of all decorations are first converted into 3D voxel representations. The dimension of the voxel grid is chosen to be $128 \times 128 \times 16$ for width, height and depth, respectively. Each grid is associated with the number of sampled points of the surface mesh that are within the voxel, which we call the voxel density.

In our dataset, we have 379 decoration components (102 from corbel, 121 from moulding and 156 from panel), which are used to train the VAE model. The goal is to make the encoding and decoding of the data with minimum information lost. Figure 10 shows the architecture of our 3D VoxelVAE model. The input voxel is represented as x of $128 \times 128 \times 16$ dimension. Let $Enc(\cdot), Dec(\cdot)$ denote the encoder and decoder, $z = Enc(x)$ represent the latent encoding vector and $x' = Dec(z)$ be the reconstructed voxel of the decoration component. The relationship between the input x and the latent vector z can be defined by prior $p_\theta(z)$, likelihood $p_\theta(x|z)$ and posterior $p_\theta(z|x)$. The estimated posterior $q_\phi(z|x)$ should be very close to $p_\theta(z|x)$. The 3D VoxelVAE aims to minimize the following loss:

$$L_{3DVoxelVAE} = \lambda L_{recnt} + \mu L_{KL} + L_{Reg} \quad (4)$$

where $L_{recnt} = \frac{1}{n}\sum \|x - x'\|^2$ measures the reconstruction error in mean squared error which encourages $x' \approx x$, $L_{KL} = D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$ is the Kullback-Leibler divergence to promote Gaussian distribution in the latent space, L_{Reg} is the regularization term of the network parameters in L_2 norm, and λ and μ are the tradeoff factors. In our experiments, we empirically set $\lambda = 1.0$ and $\mu = 0.5$.

With a trained 3D VoxelVAE model using the dataset of $n = 379$ decorations, the representation of the decorations is learned. The output is a 64D vector where 64 is the dimension of the latent space. In the latent space, similar decorations will have a small Euclidean distance and dissimilar decorations will have a big Euclidean distance.

5.2 Online Generation

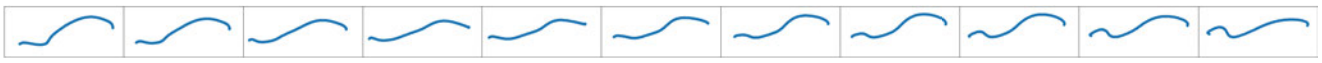
In online generation, the user provides one model or specifies an example in the dataset. Then the algorithm automatically generates a set of new models. The process involves evolution and deformation.

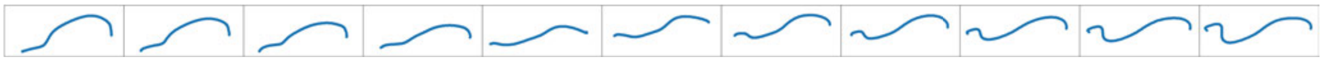
5.2.1 Generation by Evolution

To realize generative design, we propose an evolution algorithm. With all the design elements in the dataset and two trained models, we encode each model by a gene vector DNA which is defined as follows:

$$DNA =< Id_{mb}, Id_d, Id_b, LC^{0,1,2}_{mbfc}, Par_{(start,end)} > \quad (5)$$

where Id_{mb}, Id_d and Id_b are indices of main body, decoration and base in the dataset, respectively, LC_{mbfc} is the 3D latent code of the main body feature curve learned by 2D CurveInfoGAN, and $Par_{(start,end)}$ represents the starting/ending points of the alignment. Id_b is set to -1 if there is no base for moulding or panel.


(a) Latent code: $L^0_{2D} = (0.0, 0.1, ..., 1.0)$; $L^1_{2D} = 0.5$; $L^2_{2D} = 0.5$


(b) Latent code: $L^0_{2D} = 0.5$; $L^1_{2D} = (0.0, 0.1, ..., 1.0)$; $L^2_{2D} = 0.5$

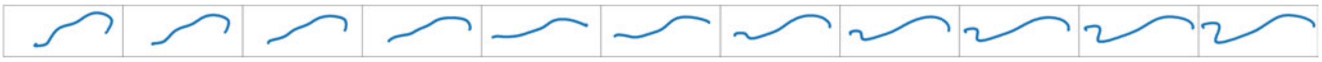

(c) Latent code: $L^0_{2D} = 0.5$; $L^1_{2D} = 0.5$; $L^2_{2D} = (0.0, 0.1, ..., 1.0)$

Fig. 9: Feature curves generated by adjusting control variables in the latent space.

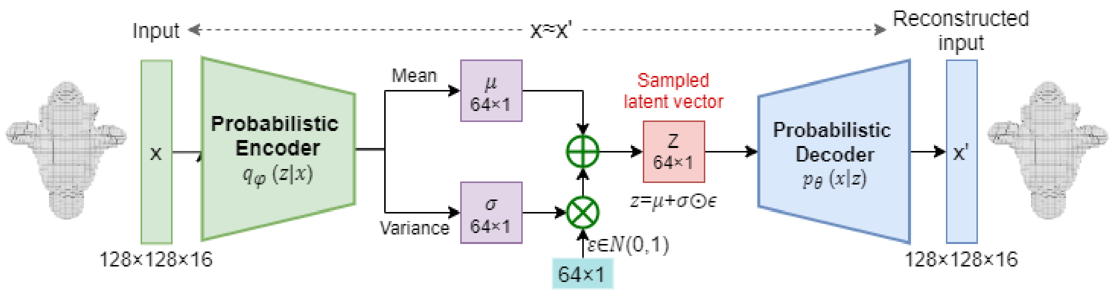


Fig. 10: Architecture of 3D VoxelVAE.

The outline of the algorithm is given in Algorithm 1. We first generate the gene vector *DNA* for the input reference model. Then the best-fit candidates are selected by *selectParents* function. By evolution, a set of new gene vectors is produced by cross-over and mutation. For each new gene vector, a decorative architectural model is synthesized using the feature-driven deformation that will be described in Section 5.2.2. The generated models are evaluated by function *EvalPopulation* and only the models in good geometric and physical quality survive. The evolution continues until the number of generated models reaches the target number or the number of evolution reaches the threshold specified by the user.

The functions in the algorithm are explained below.

– $DNA_{ref} \leftarrow encodeDNA(M_{ref})$
The reference model could be provided by the user, which is not in the database, or be selected from the existing database. For the former case, the input model M_{ref} is decomposed and its features are extracted using the method described in Section 4. Id_{mb} and $LC^{0,1,2}_{mbfc}$ are obtained from a main body component in the database which is most similar to that of the input model. The latent code for the decoration component of M_{ref} is calculated by the encoder of the learned 3D VoxelVAE. Id_d is obtained by finding a decoration component in the database with minimal difference in the latent space, and Id_b is obtained by finding a base having the most similar dimensions. $Par_{(start,end)}$ is inferred during feature extraction.

Algorithm 1: Evolution-Based Generation

```
Input: initialize population $\mathcal{G}_0 = \{M_i\}$
Input: 2D CurveInfoGAN latent space $\mathcal{L}_{2D}$
Input: 3D VoxelVAE latent space $\mathcal{L}_{3D}$
Input: user input model $M_{ref}$
Input: the target number of generated models $N$
Input: the maximum number of iterations $IteMAX$
//Encode gene vector for the input model
$DNA_{ref} \leftarrow encodeDNA(M_{ref})$ ;
Background set $\mathcal{B} \leftarrow \mathcal{G}_0$ ;
$i = 0$ ;
$\mathcal{O} \leftarrow NULL$;
while $TRUE$ do
    //select the best-fit individuals for reproduction
    $P_i \leftarrow selectParents(\mathcal{B}, \mathcal{L}_{3D}, DNA_{ref})$ ;
    //breed new generations via crossover & mutation
    $\mathcal{G}_{i+1} \leftarrow Reproduce(P_i)$ ;
    //evaluate the fitness of new individuals
    $\mathcal{G}_{i+1} \leftarrow EvalPopulation(\mathcal{G}_{i+1})$ ;
    $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{G}_{i+1}$ ;
    if $\#(\mathcal{O}) \geq N$ or $i \geq IteMAX$ then
        return $\mathcal{O}$;
    else
        $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{G}_{i+1}$ ;
        $i = i + 1$ ;
```

– $P_i \leftarrow selectParents(\mathcal{B}, \mathcal{L}_{3D}, DNA_{ref})$
This function selects $0.2N$ best-fit models from $\mathcal{B}$.

They are used with the user input model as parents for crossover and mutation as well. In addition, based on the user input M_{ref}, we inquire top $0.5N$ similar and $0.25N$ dissimilar decorations in the latent space $\mathcal{L}_{3D}$ for mutation. We also select $0.6N$ bases from $\mathcal{B}$ for mutation.

- $\mathcal{G}_{i+1} \leftarrow Reproduce(P_i)$
  The cross-over operation is applied on DNA_{ref} and selected $0.2N$ models' DNA. The mutation operation is then applied by randomly selecting Id_{mb}, Id_d, and Id_b in DNA and replacing them with the selected candidates. Latent code $LC^{0,1,2}_{mbfc}$ may be perturbed in the latent space $\mathcal{L}_{2D}$ for generating new feature curves. The values of the start and end points for aligning decoration are inferred from the corresponding main body and also undergo random minor adjustments.
- $\mathcal{G}_{i+1} \leftarrow EvalPopulation(\mathcal{G}_{i+1})$
  This function is to evaluate the fitness of generated model. The geometric quality is evaluated by checking the existence of self-intersection and the physical quality is evaluated by checking the centroid of the side section of the main body. If the centroid locates near the boundary or outside of the side section, it implies that the shape of the model will likely lead to a high gravitational torque, which should be avoided.

5.2.2 Synthesis by Feature-Driven Deformation

We finally describe how to synthesize a new 3D model for a given gene vector.

Given a new main body feature curve, we apply deformations to the main body and the decoration such that their feature curves match the new curve.

Let us begin with the deformation for the main body. The deformation for the decoration can be done similarly. We use direct manipulation of free form deformation (FFD) and sample points on the feature curves as manipulating points. The xyz-coordinate system and the 3D FFD grid for the main body are set up such that the plane the main body feature curve lies on is parallel to the xy-plane and the projection of the grid on the xy-plane is a 2D FFD for the feature curve (see Figures 11(a) and (b) for an example). If we constrain the FFD control points lying on a row parallel to the z-axis to have the same offsets, the 3D FFD is simplified to a 2D FFD. Without ambiguity, our 2D FFD function of degree $m \times n$ can be written

$$P(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} B_i^3(s) B_j^3(t) \tag{6}$$

where $P(s,t)$ represents the point corresponding to local coordinates (s,t), P_{ij} are the 2D FFD control points, and $B_i^3(s)$ and $B_j^3(t)$ are cubic B-spline basis functions.

The deformation is driven by deforming 100 sampled points $P(s_k,t_k)$ on the original feature curve (in purple color) to Q_k on the target feature curve (in blue color). This is achieved by solving the following con-

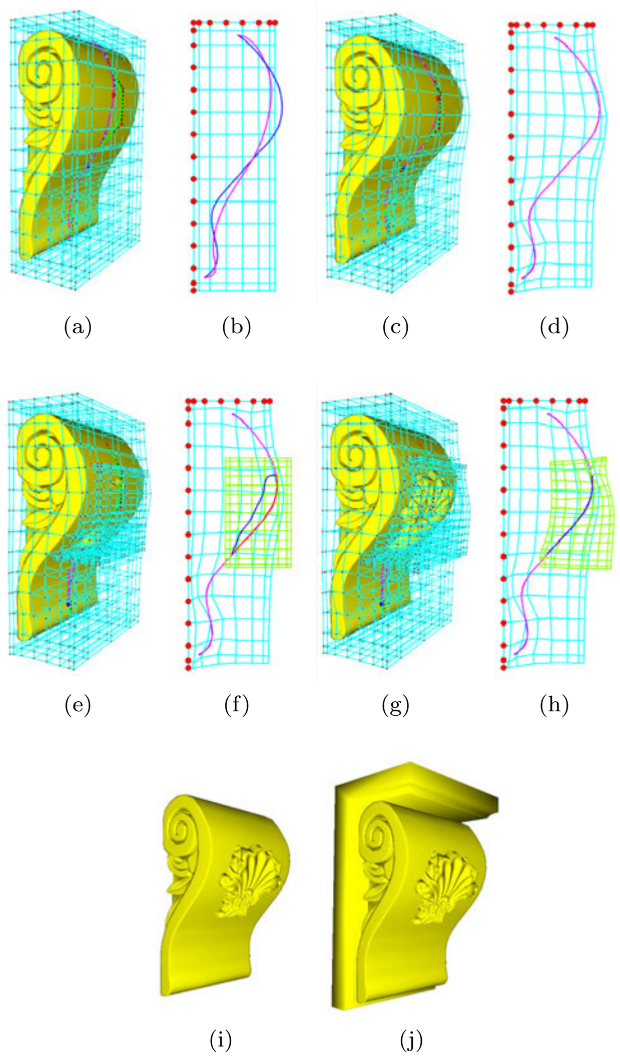


Fig. 11: Feature curve driven deformation. (a&b): 3D&2D views of initial FFD grids for the main body; (c&d): the deformed FFD grids for the main body; (e&f): 3D&2D views of initial FFD grids (in green) for the decoration; (g&h): the deformed FFD grids (in green) for the decoration; (i): combination of the deformed main body and decoration; (j): combination of deformed main body, decoration and base.

strained minimization problem:

$$\min_{\{P_{ij}\}} \sum_{k=1}^{100} \|P(s_k, t_k) - Q_k\|^2 + \lambda E_S \tag{7}$$

subject to

$$P_{ij} = P_{ij}^{ori} \quad \text{for} \quad i = 0 \text{ or } j = 0 \tag{8}$$

where λ is the trade-off parameter, P_{ij}^{ori} represents the original position of P_{ij}, the constraints are introduced to make sure that the shape of the model at top and left does not change much, and

$$\begin{aligned} E_S = &\sum \left\|P_{i,j}P_{i+1,j} - P_{i,j}^{ori}P_{i+1,j}^{ori}\right\|^2 \\ &+ \sum \left\|P_{i,j}P_{i,j+1} - P_{i,j}^{ori}P_{i,j+1}^{ori}\right\|^2 \end{aligned} \tag{9}$$

is to constrain the edge length between two neighboring control points so as to reduce the chance of self-intersection of the deformed model.

The new positions of control points P_{ij} are the solution of the minimization problem (7). Refer to Figures 11(c) and (d) for the new FFD grids for the main body in 3D and 2D. It can be seen that the main body is deformed such that its feature curve matches the target one.

To deform a decoration, the mesh Md_j and its feature curve Cd_j are first scaled and translated to match the starting and ending points on the feature curve of the main body. Then a similar direct manipulation FFD method of (7) without the constraint (8) is applied to the decoration component as shown in Figures 11(e)-(h). Figure 11(i) is the combination of the deformed main body and decoration. Finally we simply scale the base component according to the size of the bounding box of the main body and add it to the main body, which gives Figure 11(j).

After deformation is performed on each component, all the components are ready to be combined into a single mesh model. This is achieved by a union operation and then a new model is finally generated.

6 Experiments

We have implemented the proposed algorithm and developed a generative design tool. In this section, we conduct a series of experiments to evaluate the underlying techniques. We run our tool on a x64-based PC with Intel Core i9-9900K CPU3.60GHz, 64G RAM, Samsung 1TB SSD and NVIDIA GeForce RTX 2080 Ti × 2 CPU. We train the 2D CurveInfoGAN and 3D VoxelGAN on two parallel NVIDIA 2080T GPUs with Tensorflow-GPU 2.4.1.

6.1 2D CurveInfoGAN

We first evaluate the 2D CurveInfoGAN model. The parameters are set as follows: $latent\_dim = 3$, $train\_step = 20000$ and $batch\_size = 32$. It takes 35 minutes to train our 2D CurveInfoGAN model for 2989 corbel main body feature curves; 49 minutes for 4187 moulding feature curves and 28 minutes for panel collection, respectively. Figure 12a gives 64 feature curves of main bodies of the original corbel collection. These curves show the characteristics of the main bodies of the corbel models in the dataset. Using the feature curve generator learned by 2D CurveInfoGAN, we create 64 new feature curves by randomly adjusting the values of the control variables, which are displayed in Figure 12b. It can be seen that the curves in Figure 12b are generally different from those in Figure 12a, but both sets have similar overall curvature variation.

To evaluate the 2D CurveInfoGAN quantitatively, we use the metrics introduced in papers [23] and [24], which include likelihood, smoothness, consistency and diversity.

- **Likelihood**: The mean log likelihood (MLL) is widely used to measure the distribution of generative models. A higher MML value manifests that the generator could approximate the real data distribution better.
- **Smoothness**: Smoothness is an important metric for a 2D curve generator. A 2D curve could be sampled and represented by a sequence of points denoted by x. The variance of difference is expressed by $VOD(x) = \frac{1}{m-1}\sum_{i=1}^{m-1} Var(x_{i+1} - x_i)$. A relative varaiance of difference (RVOD) calculated by $RVOD = \mathbb{E}_{x \sim P_{data}} VOD(x) / \mathbb{E}_{x \sim P_G} VOD(x)$ could be used to measure the relative smoothness between the generated 2D curves and those in the datasets. A larger $RVOD$ value means better performance.
- **Consistency**: Latent Space Consistency (LSC) is a quantitative measure of shape change consistency along any direction in the latent space. Distance D_c between two samples c^i and c^j along a certain direction in the latent space should be consistent with the dissimilarity D_x between the generated curves $x^i = G(c^j)$ and $x^j = G(c^j)$. Pearson correlation coefficient is computed to measure the consistency: $LSC(c, x) = \frac{cov(D_c, D_x)}{\sigma(D_c)\sigma(D_x)}$. A higher LSC indicates more consistent shape change.
- **Diversity**: Relative Diversity, $RDiv = Div(X_g)/Div(X_{data})$, measures the relative level of diversity of the synthesized designs, where X_g and X_{data} denote the set of synthesized designs and the set of designs from the dataset, and $Div(X) = \frac{1}{N} trace(cov[X, X])$. When the RDiv value is close to zero, it indicates that the

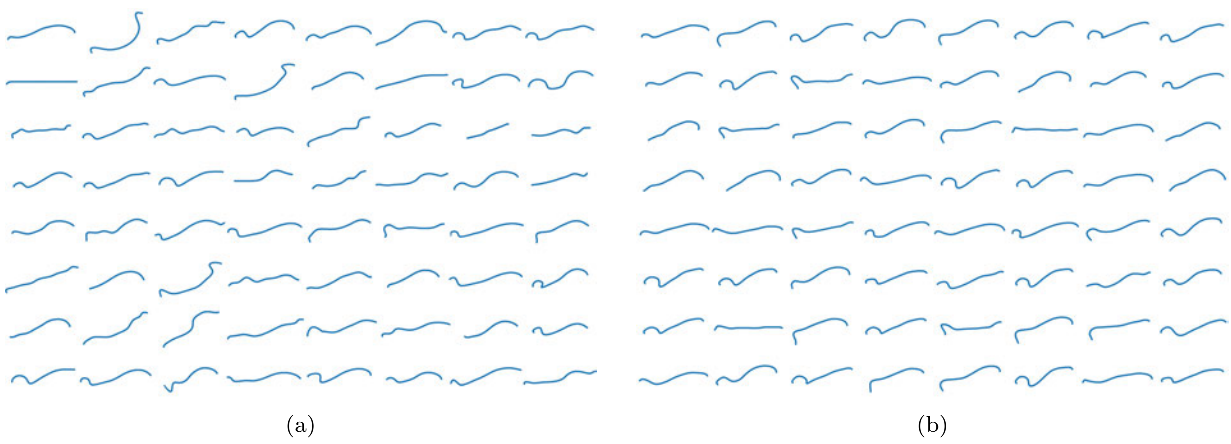


Fig. 12: Feature curves of main body Component: (a) Original feature curves; (b) feature curves generated by 2D CurveInfoGAN

diversity of the synthesized designs is very limited. When the RDiv value is close to 1.0, it indicates that the synthesized designs have a similar level of variability with the dataset. However, a higher diversity does not always indicate better performance since some unrealistic designs might be synthesized to increase diversity.

Table 1 gives the quantitative performance measures for our 2D CurveInfoGAN and the 2D CurveVAE proposed in paper [4]. It can be seen that the 2D CurveInfoGAN could generate smooth curves with more diversity and approximate the distribution of real data better. The latent space L_{2D} is also more consistent.

To demonstrate the interpretability of 2D CurveInfoGAN, we sample 10 values in the range of $[0.0, 1.0]$ for L^0_{2D} and L^1_{2D}, and the value of L^2_{2D} is fixed as 0.5, which generate 100 (10 by 10) curves displayed in Figure 13. These curves show how the first two dimensions of the latent code control the distribution of generated curves. The trained latent space L_{2D} could be used to control the shape of generated curves semantically.

6.2 3D VoxelVAE

The 3D VoxelVAE encodes each decoration component into a 64D vector. The output of the 3D VoxelVAE is a latent feature space $\mathcal{L}_{3D}$. The difference between two decoration components can be measured based on the Euclidean distance between their corresponding vectors in $\mathcal{L}_{3D}$. Similar decoration components will have a relatively small Euclidean distance and dissimilar decoration components will return a relatively large distance. The parameters of our 3D VoxelVAE model are set as follows: $latent\_dim = 64$, $train\_step = 10000$, $batch\_size = 64$. It takes about 5 hours to train the 3D VoxelVAE model for 379 voxel data in decoration collection.

Figure 14(a) shows an input that is a component with leaf decoration and an overall simple structure. Figure 14(b) lists the top 5 similar results that are retrieved from the latent space and all have a leaf-like design. The top 5 dissimilar results are displayed in Figure 14(c). We can see that they are far from leaf-like shape and have textures that are much more complicated compared to the input component.

Another example is given in Figure 15 where the top 5 similar and dissimilar decoration models returned based on the query of a flower shaped decoration are displayed. The results are reasonable visually.

Table 2 reports the Euclidean distances between the input and the returned components from the two enquiries given in Figure 14 and Figure 15. It can be observed that the distance value between two latent vectors well reflects the similarity of the corresponding two 3D components. The experiments demonstrate that the compressed low dimensional latent space $\mathcal{L}_{3D}$ can represent decorations well.

6.3 Combination of main bodies and decorations

Experiments were conducted on how the deformed main body and decoration components are combined. Figure 16 shows some results of mounting a decoration to a main body with different parameter values. It can be seen that all these results look aesthetically pleasing.

However, not all parameter values, especially the mounting positions, are suitable for combination. Further selection and fine-tuning the parameter values are

Table 1: Quantitative comparison between the 2D CurveInfoGAN and the 2D CurveVAE of [4].

Model	MLL	RVOD	LSC	RDiv
2D CurveInfoGAN	201.5 +/- 4.5	1.239 +/- 0.007	0.967 +/- 0.002	0.702 +/- 0.006
2D CurveVAE	142.7 +/- 5.7	1.060 +/ 0.006	0.422 +/- 0.010	0.301 +/- 0.003

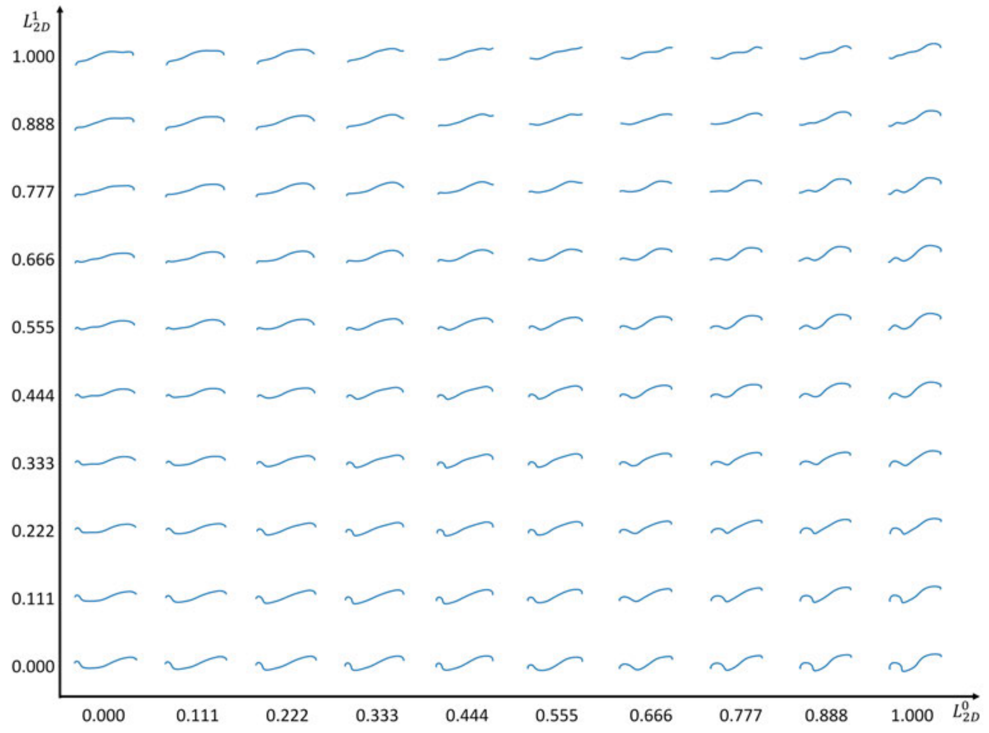


Fig. 13: Interpretability of 2D CurveInfoGAN.

Table 2: Quantitative comparisons

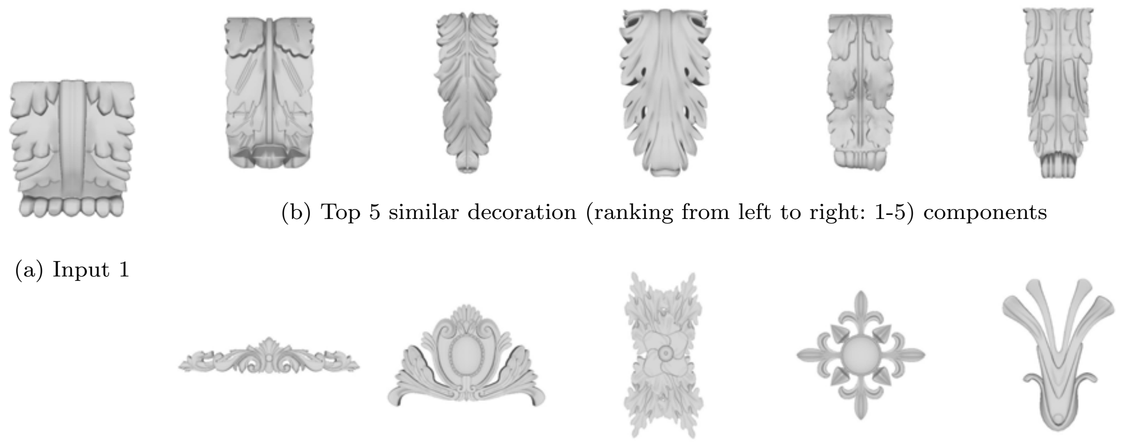
Distance Ranking	Enquiry 1 (Fig 14)		Enquiry 2 (Fig 15)	
	Similar	Dissimilar	Similar	Dissimilar
1	0.051960	0.639880	0.024021	0.591412
2	0.062439	0.605853	0.025578	0.537924
3	0.076452	0.594287	0.039865	0.518931
4	0.089057	0.583578	0.045792	0.516223
5	0.092496	0.579836	0.048035	0.507816

required during the evolution procedure. In Figure 17, we just simply align different decorations to a main body using the same parameter setting. Some of the results exhibit overlaps or intersections with elements present in the main body. These results should be rejected during the evolution.

With proper parameter setting, visually pleasing combinations could be obtained, as shown in Figure 18 for some examples with one decoration and different main body components. These examples demonstrate the importance of the positions of the starting and ending point on the main body component. Note that the positions of the starting and ending points on the main body will automatically affect the size of the decoration in the output.

6.4 Evolution

Figure 19 shows our evolution design from an input corbel model displayed in Figure 19(a). In the first round, 12 new corbel models are generated as shown in Figure 19(b). Then the user selects one as his/her preferred model, which is highlighted with a red box. It can be observed that several of these generated models have the main body components that are similar to that of the input model in style, but vary in curvatures and shapes. Meanwhile there are also a few models that exhibit other styles of the main body components in the database. The decoration components look in a similar style, but have different details or textures. The starting and ending points for the mounting of decorations vary from model to model. The base components are ran-

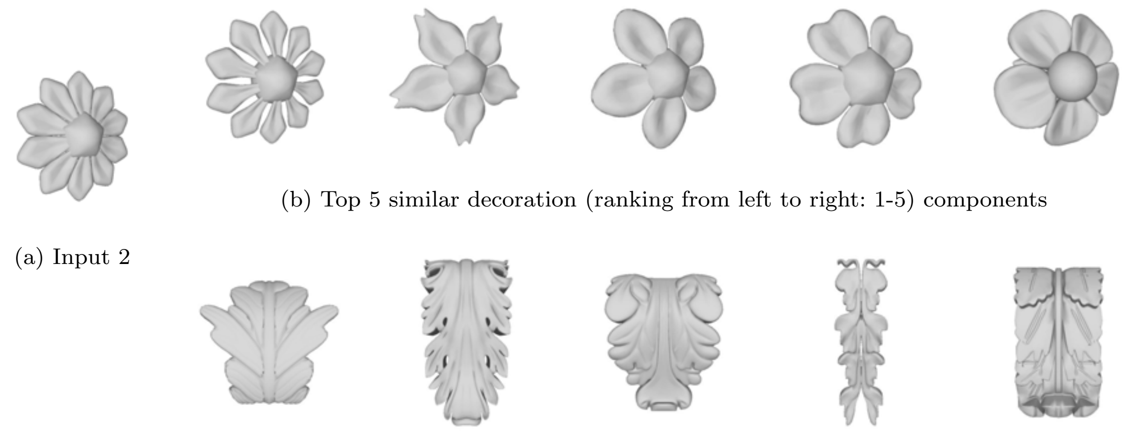


(a) Input 1

(b) Top 5 similar decoration (ranking from left to right: 1-5) components

(c) Top 5 dissimilar decoration (ranking from left to right: 1-5) components

Fig. 14: The enquiry results of the decoration components from the latent space.



(a) Input 2

(b) Top 5 similar decoration (ranking from left to right: 1-5) components

(c) Top 5 dissimilar decoration (ranking from left to right: 1-5) components

Fig. 15: Another example. The enquiry results of the decoration components from the latent space.

domly selected for cross-over. In the second round, our evolution algorithm lets 50% of generated models have the main body similar to that of the selected model in the previous round, 30% of the models have the main body similar to that of the models generated in the previous round, and 20% of the models randomly choose the main body components in the database. If two user-selected models have the similar body components, the chance of occurrence of the similar component in the next round will increase. It can be seen in Figure 19(d) that our algorithm generates corbels by synthesizing similar main bodies and decorations with different feature curve latent codes and parameters for placing the decoration. When the user repeatedly selects corbels with L-shaped bases, the algorithm will follow the user's preference and select L-shaped bases from the dataset with a high probability to do cross-over.

In moulding design, multiple decorations are allowed to be combined to a main body to form a pattern. In Figure 20, various patterns are randomly chosen to generate new moulding models.

We also generate 10 panel models according to an input panel using our generative design tool, as shown in Figure 21. We can see that six generated panels have a square-shaped main body, which is similar to the input model, and 4 panels have the main bodies with a round or irregular shape. Similarly, the decoration components of the new panels are mostly similar in style except for three panels.

The above experiments with corbel, moulding and panel models demonstrate that our developed generative design tool is able to produce new models that are of "fit and diversity". This is due to the evolution process with proper selection, crossover and mutation.

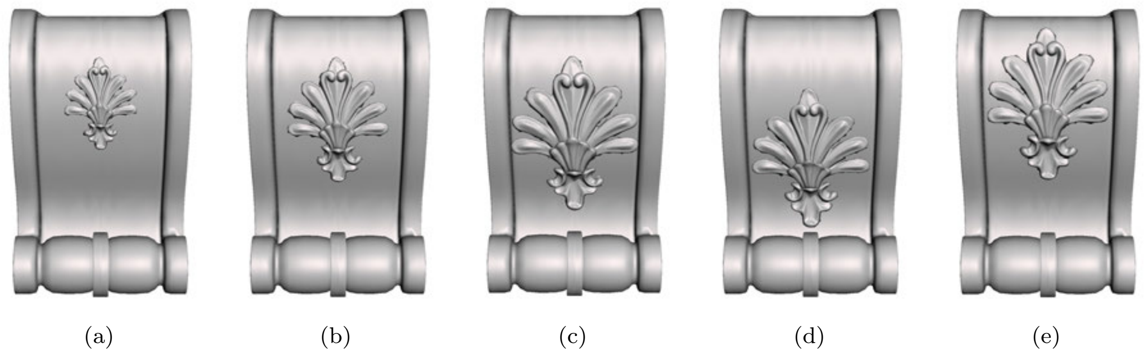


Fig. 16: Aligning a decoration component to a main body with different parameter values

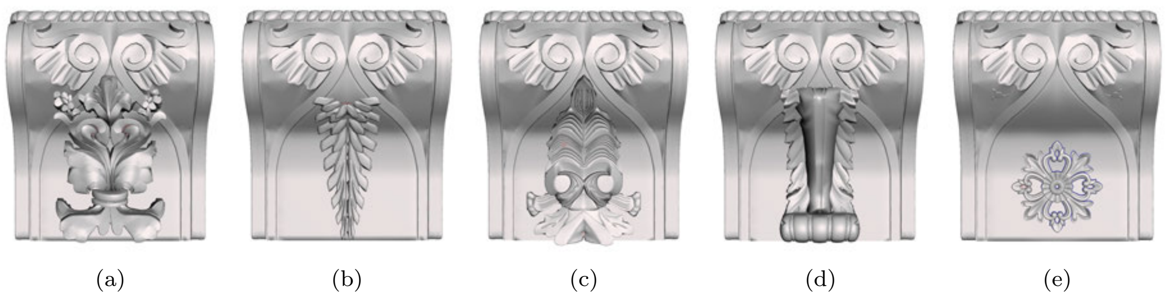


Fig. 17: Results of combining different decoration components to a main body using the same parameter values.

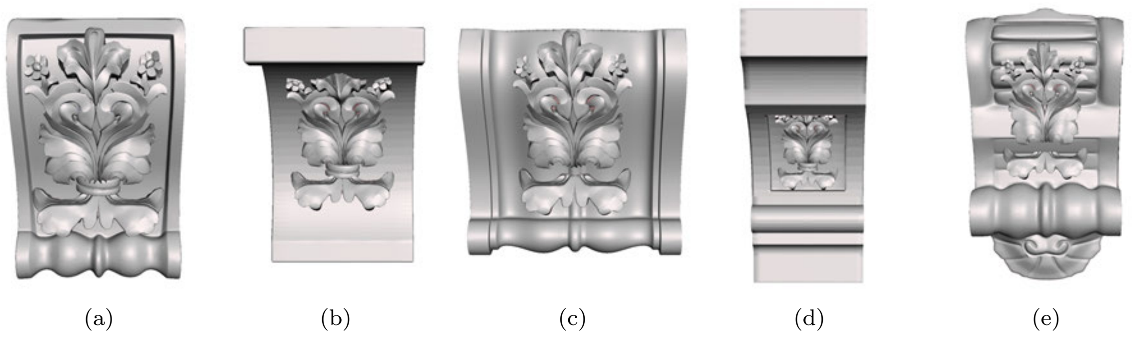
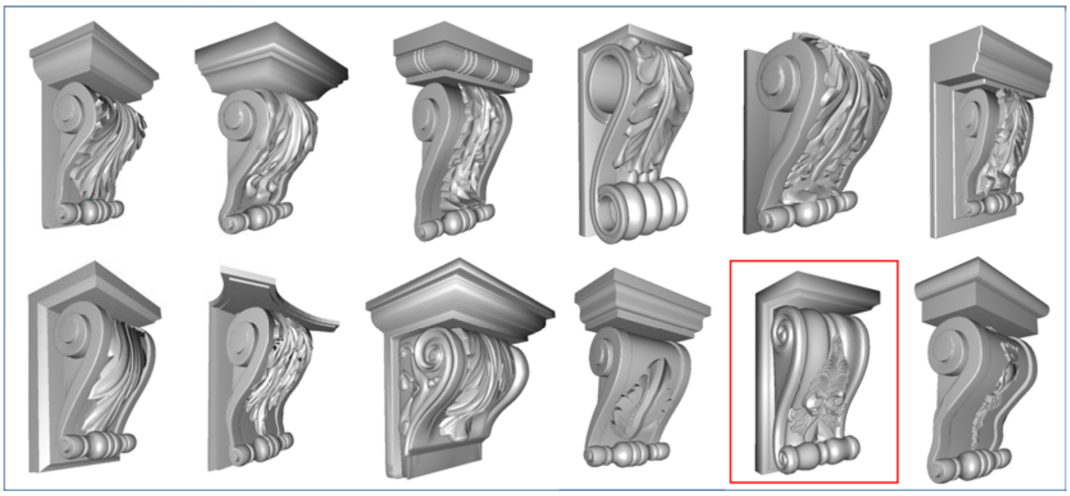


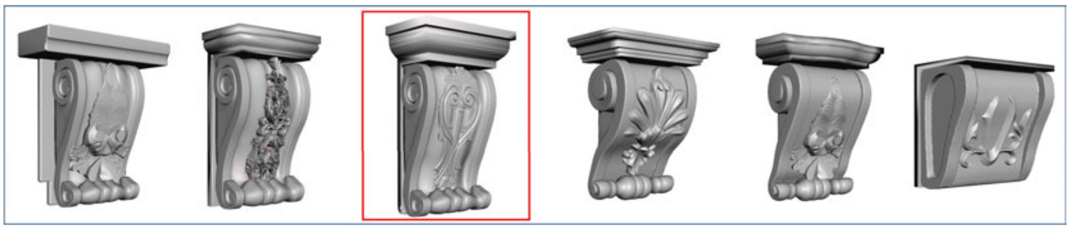
Fig. 18: Results of aligning a decoration component to several main bodies using different positions of the starting and ending points.

6.5 Limitations

Our method has some limitations. First, we only use the decoder of the trained 3D VoxelVAE to compress the decoration component represented in triangular mesh into a low dimensional space to achieve quick enquiry of similar or dissimilar decoration. However, we have not utilized the decoder to generate a 3D decoration component due to the difficulty of reconstructing design details in high resolution. Second, when evaluating the fitness of generated models in Section 5.2.1, we only check self-intersection and centroid bias due to a trade-off between effectiveness and efficiency of the EvalPopulation function. A more sophisticated process could be incorporated in future. Third, the feature-driven FFD method proposed in Section 5.2.2 is rather 2.5D than 3D. Hence some models as shown in Figure 22 might not be properly handled. The base models are generally characterized by extruding the side section curve along a straight line. Since the front surface of Figure 22(a) is bent, the deformed decoration component could not be attached perfectly. To resolve this issue,


(b) Round 1: 12 new corbel models are generated according to the input. The user then identifies one he/she prefers, which is highlighted with a red box.

(a) Input


(c) Round 2: 6 new corbel models are generated from the results of round 1 and the user's preference. Similarly, a model of the results is identified by the user as his/her preference.

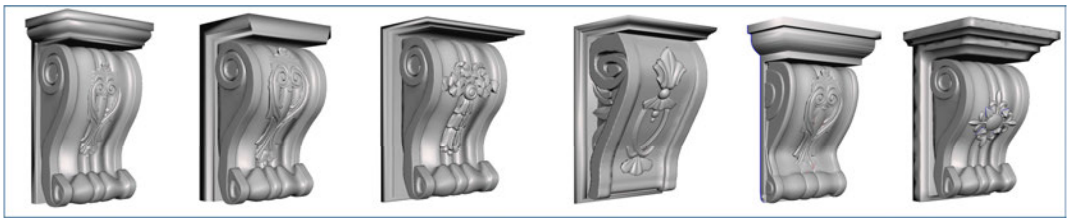

(d) Round 3: 6 new corbel models are generated from results of round 2 and the user's preference.

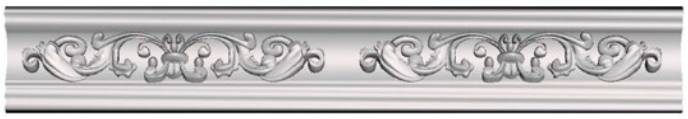
Fig. 19: The evolution design of corbel models.

the boundary of the decoration model could be projected onto the main body first and then the As-Rigid-As-Possible deformation algorithm is used to deform the decoration model. However, for the models shown in Figure 22(b)&(c), self-intersection might occur when the warping of the decoration component is too large.

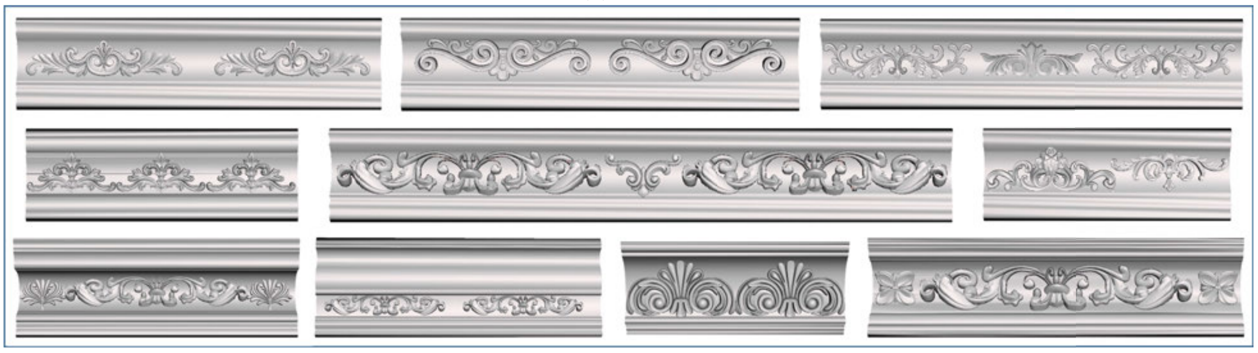
7 Conclusions

We have described a method for generative design of decorative architectural parts. The underlying techniques include geometric processing, part learning, and component evolution. In particular, InfoGAN and VAE models are constructed to learn the latent representation of feature curves and decoration shape, evolution principle is used to evolves model components to produce new offsprings, and feature-driven geometric deformation is developed to realize the synthesis of new models. Based on these techniques, a creative modeling tool is developed for designing corbel, moulding and panel shapes. The experimental results show that the developed method and tool have the capability of generating a multitude of "fit and diverse" decorative architectural models from a small dataset.

Though this paper is focused on the design of architectural parts such as corbel, moulding and panel, it should be pointed out that the developed method can be extended to handle those shapes that can be decomposed into an overall shape and a detailed shape. Hence our future work will extend the method to de-

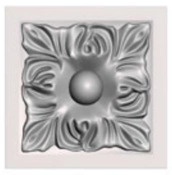


(a) Input

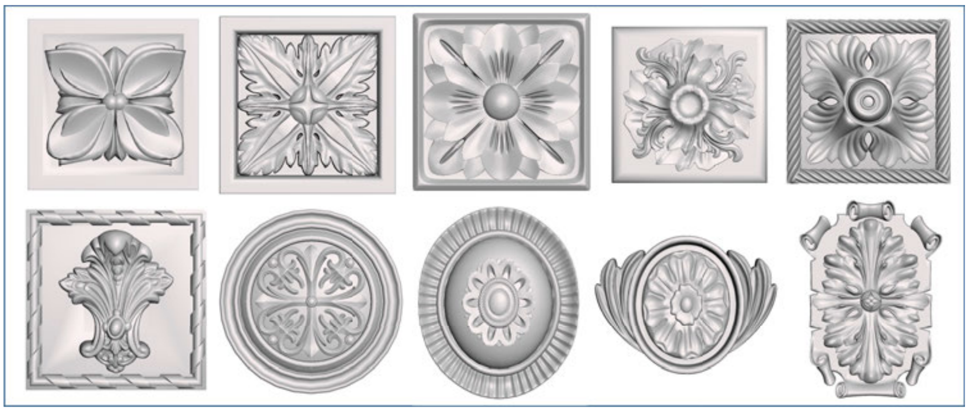


(b) Generated 10 new moulding models

Fig. 20: The results of evolution from an input moulding model.

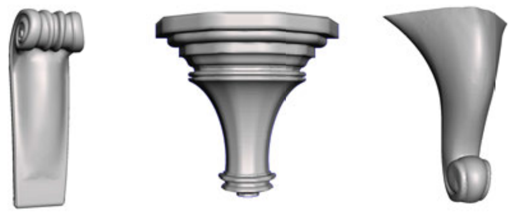


(a) Input



(b) Generated 10 new panel models

Fig. 21: The results of evolution from an input panel model.



(a) (b) (c)

Fig. 22: Failure examples of the feature-driven FFD method.

sign a broader class of shapes and also further optimize the underlying algorithms such as using more sophisticated blending methods for the union operation [25] and considering curvature continuity [26] in blending.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. K Koch. *Architectural Patterns for Woodcarvers.* Fox Chapel Publishing, East Petersburg, PA 17520, 2003.
2. I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, A Courville, and Y Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
3. D Cohen-Or and H Zhang. From inspired modeling to creative modeling. *The Visual Computer*, 32(1):7–14, 2016.
4. Y Zhang, C.C. Ong, J. Zheng, and S.-T. Lie. Creative corbel modeling using evolution principle. In A Sourin, C Charrier, C Rosenberger, and O Sourina, editors, *International Conference on Cyberworlds, CW 2020, Caen,*

France, September 29 - October 1, 2020, pages 9–16. IEEE, 2020.
5. T Funkhouser, M Kazhdan, P Shilane, P Min, W Kiefer, A Tal, S Rusinkiewicz, and D Dobkin. Modeling by example. *ACM Transactions on Graphics*, 23(3):652–663, 2004.
6. K Xu, H Zhang, D Cohen-Or, and B Chen. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics*, 31(4):1–10, 2012.
7. E Kalogerakis, S Chaudhuri, D Koller, and V Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics*, 31(4):1–11, 2012.
8. M Sung, H Su, V G Kim, S Chaudhuri, and L Guibas. Complementme: weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics*, 36(6):1–12, 2017.
9. D P Kingma and M Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
10. J Wu, C Zhang, T Xue, B Freeman, and J Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.
11. M Tatarchenko, A Dosovitskiy, and T Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.
12. X Chen, Y Duan, R Houthooft, J Schulman, I Sutskever, and P Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.
13. H Su, S Maji, E Kalogerakis, and E Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015.
14. C R Qi, H Su, K Mo, and L Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.
15. C R Qi, L Yi, H Su, and L Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.
16. Z Chen and H Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.
17. Q Tan, L Gao, Y-K Lai, and S Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5841–5850, 2018.
18. T Groueix, M Fisher, VG Kim, BC Russell, and M Aubry. Atlasnet: a papier-mâché approach to learning 3d surface generation (2018). *arXiv preprint arXiv:1802.05384*, 11.
19. J Li, K Xu, S Chaudhuri, E Yumer, H Zhang, and L Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics*, 36(4):1–14, 2017.
20. H Wang, N Schor, R Hu, H Huang, D Cohen-Or, and H Huang. Global-to-local generative model for 3d shapes. *ACM Transactions on Graphics*, 37(6):1–10, 2018.
21. Z Wu, X Wang, D Lin, D Lischinski, D Cohen-Or, and H Huang. Sagnet: Structure-aware generative network for 3d-shape modeling. *ACM Transactions on Graphics*, 38(4):1–14, 2019.
22. L Gao, J Yang, T Wu, Y-J Yuan, H Fu, Y-K Lai, and H Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics*, 38(6):1–15, 2019.
23. W Chen, K Chiu, and M Fuge. Aerodynamic design optimization and shape exploration using generative adversarial networks. In *AIAA SciTech Forum*, San Diego, USA, Jan 2019. AIAA.
24. Wei Chen and Mark Fuge. Synthesizing designs with interpart dependencies using hierarchical generative adversarial networks. *Journal of Mechanical Design*, 141(11), 2019.
25. Y Yu, K Zhou, D Xu, X Shi, H Bao, B Guo, and H-Y Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, 2004.
26. J Zheng, G Wang, and Y Liang. Curvature continuity between adjacent rational bézier patches. *Computer Aided Geometric Design*, 9(5):321–335, 1992.