CADTrans: A Code Tree-Guided CAD Generative Transformer Model with Regularized Discrete Codebooks

Xufei Guo
Institute of Artificial Intelligence,
Xiamen University
Xiamen, Fujian, China
feixuguo@126.com

Xiao Dong
Zhuhai Key Laboratory of IRADS,
United International College,
BNU-HKBU
Zhuhai, Guangdong, China
xiaodong@uic.edu.cn

Juan Cao
School of Mathematical Sciences,
Xiamen University
Xiamen, Fujian, China
juancao@xmu.edu.cn

Zhonggui Chen
Department of Computer Science and Technology,
Xiamen University
Xiamen, Fujian, China
chenzhonggui@xmu.edu.cn

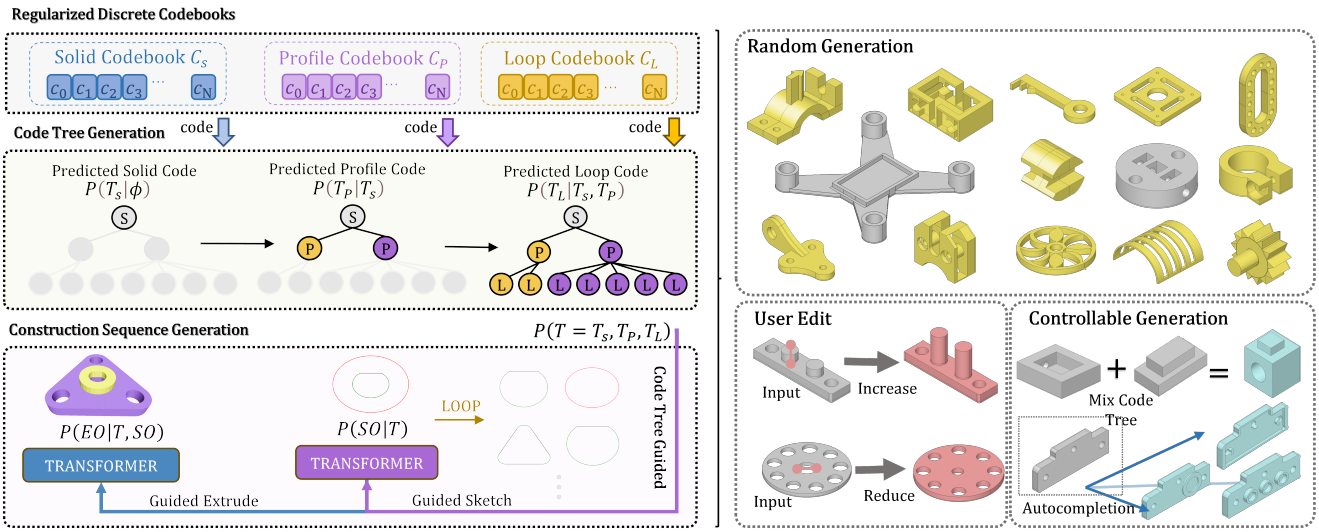

Figure 1. CADTrans is a transformer-based autoregressive generation model that leverages a sketch-and-extrude strategy guided by a code tree representation encoded via three trained discrete regularized codebooks (left). Random generation results are showcased in the top right. CADTrans introduces a number of new controlled generation methods (bottom right), enabling (i) **User Edit**, which allows parameter modification while preserving design consistency, and (ii) **Controllable Generation**, which facilitates design based on specific controls, such as code tree mixing, and automatic completion and refinement of partial models.

Abstract

The creation of computational agents capable of generating computer-aided design (CAD) models that rival those produced by professional designers is a pressing challenge in the field of computational design. The key obstacle is the need to generate a large number of realistic and diverse models while maintaining control over the output to a certain degree. Therefore, we propose a novel CAD model generation network called CADTrans which is based on a code tree-guided transformer framework to autoregressively generate CAD construction sequences. Firstly, three regularized discrete codebooks are extracted through vector quantized adversarial learning, with each codebook respectively representing the features of Loop, Profile, and Solid. Secondly, these codebooks are used to normalize a CAD construction sequence into a structured code tree representation which is then used to train a standard transformer network to reconstruct the code tree. Finally, the code tree is used as global information to guide the sketch-and-extrude method to recover the corresponding ge-

ometric information, thereby reconstructing the complete CAD model. Extensive experiments demonstrate that CADTrans achieves state-of-the-art performance, generating higher-quality, more varied, and complex models. Meanwhile, it provides more possibilities for CAD applications through its flexible control method, enabling users to quickly experiment with different design schemes, inspiring diverse design ideas and the generation of a wide variety of models or even inspiring models, thereby improving design efficiency and promoting creativity. The code is available at https://effieguoxufei.github.io/CADtrans/.

Keywords: Vector quantization, Shape representations, Autoregressive generation model

1. Introduction

Computer-aided design (CAD) tools have a significant impact on industries such as engineering, architecture and manufacturing. Users can create complex models by following simple step-by-step instructions, which brings a number of benefits: design efficiency increased, error and cost reduced, interdisciplinary collaboration enhanced, and product development cycles accelerated. However, the process of creating models remains challenging for many users. Even with specialized training and experience, any unexpected action can cause a model to fail, especially when creating accurate and detailed 3D models. As a result, there are relatively few available 3D models compared to the abundance of 2D data, which limits the research and application of deep learning methods in this area.

To overcome the difficulties of generation, research efforts nowadays focus on designing computational agents to create CAD models like a professional designer. But generating CAD models directly is a challenging task. The key challenge lies in constructing a unified representation that can describe the different geometric operations of a CAD model. Many existing generation methods use 3D data representations such as point clouds [11, 24, 25, 42], voxels [21, 35, 36], meshes [9, 12] and boundary representation (B-rep) [14, 40] to avoid this problem.These representation methods based on the final geometric form generally ignore the core feature of parametric CAD modeling, that is, the generation process of building a model through feature sequence and design history. As a result, the reconstructed model loses the hierarchical feature structure, parameter constraint relationship and editable characteristics of the original CAD design, making it difficult to support design backtracking and iterative optimization in engineering scenarios.

Recently, [37, 38, 39, 41] adopts a new representation, the sketch-and-extrude approach, which effectively preserves the design sequence and parameters. This procedural creation process is roughly as follows: the user first creates 2D sketches, which are subsequently extruded to form 3D solids, and finally Boolean operations are applied between these 3D solids to generate the final 3D model. However, despite the advances made in this approach, there are still some limitations. In the study of [37, 38], only one surface is involved in each generation process, which greatly limits the complexity and diversity of the generated results. While [39, 41] improve the complexity of generation to a certain extent by including multiple facets. However, it still fails to address several critical issues. It generates results that lack diversity, and during the model generation process, scattered and incoherent jumbled extrusion often occurs. Consequently, the generated models frequently exhibit fragmentation, flaws, or redundancy, which can be clearly seen in Figure 2.

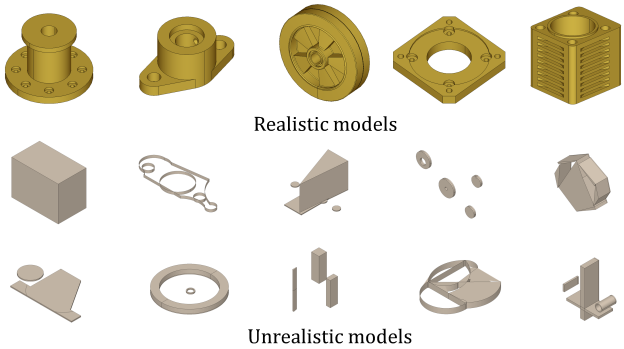

Figure 2. The contrast between realistic and unrealistic models. Realistic models more closely resemble everyday physical objects and usually have a high degree of complexity; unrealistic models, on the other hand, tend to contain a large number of fragments, a jumbled or extruded appearance, or oversimplification.

One of the primary limitations contributing to these issues is directly related to the use of vector quantization (VQ) [8, 26, 31] method. Although the embedding vectors of VQ input are clustered and represented as code information stored in the codebook, which significantly improves the recovery ability and enhances the semantic understanding of the model structure compared to traditional methods that rely on simple feature embeddings, during the VQ training process, the embedding vectors are aligned with the most similar vectors in the codebook through nearest neighbor search, resulting in most input vectors being mapped to only a few codes instead of being evenly distributed in the codebook space. Consequently, only a small portion of the codebook is updated and utilized, leading to a phenomenon known as codebook collapse [29]. This collapse restricts the model's ability to learn all potential patterns in the input data because it relies on a limited number of codes, making it difficult to capture the full range of design variations in the dataset. The reliance on fewer codes results in decreased generation quality, increased reconstruction errors,

and weakened generalization ability. Ultimately, this leads to poor fit with the real data distribution. Another limitation is that previous methods rely on non-adversarial training, which refers to training with simple loss functions such as mean squared error or cross entropy. To minimize the risk of wrong predictions, such training tends to average out regions of uncertainty, and the results are often coarse. In addition, they are prone to mode collapse, where the generator converges to a solution that can only produce a limited set of outputs. This not only leads to loss of details, but also causes the generated 3D models to deviate from the true data distribution.

To address the limitations of existing methods, we propose CADTrans, a novel approach. CADTrans generates CAD reconstruction sequences autoregressively through the Transformers [32] framework, guided by a regularized code tree. Moreover, the code tree guidance provides a degree of control over the generation process. The generated sequences can then be compatible with other CAD tools for modification and redesign.

Specifically, the root node of the code tree represents the solid code, and the other elements such as profile and loop code are gradually transformed into child nodes from the top to the bottom, as shown on the left side of Figure 3. The codes of these nodes are represented by a regularized discrete codebook extracted by vector quantization [8, 26, 31] adversarial learning [10]. By leveraging regularization, we encourage the model to make more comprehensive use of the codebook, thus enabling the generation of more realistic and diverse results. The codes obtained using this approach largely alleviate the lack of diversity caused by codebook collapse.

At the same time, in order to further promote the realism of the generated results, it is natural to create a competitive environment for the generator. Therefore, we introduced a discriminator network to evaluate the realism of the generated results, forcing the generator to learn the distribution patterns of various geometric shapes, structural relationships, and feature combinations in real CAD models, and then adjust its generation model to more effectively approximate the real data distribution in the model and generate more accurate and realistic CAD models.

Extensive experimental results and ablation experiments show that CADTrans can generate more complex, realistic and diverse models compared to existing methods. The performance is beyond the level of the best baseline models. We also try to explore more applications and design options as shown in Figure 1. In summary, we make the following contributions:

- We propose a transformer-based autoregressive CAD generative network guided by a code tree encoded from three regularized codebooks to generate realistic, high-quality, well-structured and diverse outputs.
- We design a vector quantized adversarial network to extract codebooks and propose a regularization strategy to mitigate codebook collapse. The regularized codebook facilitates the passage of more information through codebook bottlenecks, enabling the learning of richer data representations.
- We achieve state-of-the-art performance in CAD model generation compared to previous approaches and provide greater design flexibility and geometric diversity in applications.

2. Related Work

Various approaches are developed for constructing CAD models, including constructive solid geometry (CSG) and construction sequence modeling (CSM) techniques. These methods enable the representation of CAD models by tracing their design history, with each step containing rich parametric data. This not only affords more effective control over modifications but also facilitates further design iterations using the recovered design elements. Based on these two forms, researchers have extended many applications to explore a wider range of CAD model construction. For example, [17] greatly broadens the way users interact with CAD systems by parsing hand drawings into CAD commands. In view of this, researchers are increasingly interested in recovering the design history of CAD models [7, 15, 28, 37]. In this section, we will review current research on CSG and CSM methods.

2.1. CSG-Based Approaches

As a popular reverse-engineering representation for 3D models, learning exploration for CSG tree reconstruction gains traction [6, 18, 30, 43]. Break down 3D models into simple elementary solids, such as box, cylinder, cone and sphere. Subsequently reassemble them using Boolean operations (union, intersection, and difference) to reconstruct the CSG tree. Initial explorations of the CSG-based approach, as discussed in [16], focus on the process of incrementally constructing complex models through sequential Boolean operations on simple solids. However, this approach is hindered by two primary limitations. Firstly, the reliance on basic solids restricts the accurate representation of intricate 3D models. Secondly, learning a deep CSG tree has significant challenges, and further constrains the representation of complex geometries. Recent advancements [3, 27, 43] seeks to address these limitations. CSG-Stump [27] proposes a three-layered architecture consisting of complementary, cross, and connection layers, which effectively streamlines the complexity and depth of the CSG tree. Building upon this, CAPRI-Net [43] leverages quadratic surface elements to construct CSG trees, thereby improving the representation of detailed shapes.While this

approach surpasses previous reconstruction strategies in terms of detail shape representation, it introduces new challenges related to designer editability and model reconstruction due to the implicit equations and parameter complexities inherent to quadratic surface elements.

2.2. CSM-Based Approaches

Hence, simulating the 3D modeling process that begins with 2D sketches and progresses through extrusions and Boolean operations offers a promising solution. This approach, first introduced by [38], represents 3D models as CAD sequences, describes shapes through a series of constructive operations that mimic the natural shape constructing process in CAD software.This approach provides a more intuitive and flexible representation of 3D models, [22, 28, 37] leverage this expression. Building upon these foundations, subsequent work [19, 39, 41] directly targets the learning of reconstruction sequences for reconstructing 3D shapes, utilizes elementary sketch-and-extrude operations which contain more component details. The prediction process preserves the sequential and parametric nature of the modeling. Previous methods are prone to collapse when generating 3D models using sketch-and-extrude operations, resulting in unrealistic generated models such as poor structure and too simple. In our approach, we employ a code tree represented by three regularized codebooks to guide the generation process, enhancing the controllability of the CAD model during generation and improving the quality, realism, and diversity of generated results.

3. Method

CADTrans guides the generation of a CAD sequence based on a transformer by using a code tree consisting of codes from three regularized discrete codebooks as global information. Before training, we first represent the CAD model in a unified CAD construction sequence format (Section 3.1). Our training process consists of three stages. Firstly, we train three regularized discrete codebooks for the subsequent code tree representation (Section 3.2). Secondly, we train a code tree generation network (Section 3.3). Thirdly, we train conditional transformers to gradually generate the geometric information of the model (Section 3.4).

3.1. CAD Construction Sequence Representation

Since it is difficult to learn the entire construction sequence directly, we follow [39], which splits the sequence of CAD models into three primitive structures: loop, profile, and solid. In order to simplify the complex construction sequence of the model, we define these three primitives as 2D geometry, 2D arrangement, and 3D arrangement, and train the codebook for each primitive.

Primitive Definition. The basic elements of 3D model are primitive curves (lines, arcs, and circles). A line contains two coordinate points: a start point and an end point. An arc consists of three coordinate points: a start point, a midpoint, and an end point. A circle consists of four equally spaced coordinate points.

A loop is defined as a closed path comprising multiple curves, which is represented by the set (L) of x-y coordinate points along the curve. Notably, L includes a special $\langle s \rangle$ tag to demarcate individual curves within the loop.

One or more loops are encapsulated within a 2D region called a face. And for simplicity, we only use the set (P) of 2D bounding boxes $\left(x_i^t, y_i^t, x_i^b, y_i^b\right)$ of all the loops within a face, called the profiles, to represent the information on a face briefly and illustrate the distribution of the locations of the loops. Where (x_i^t, y_i^t), (x_i^b, y_i^b) denote the top left and bottom right coordinate point of the 2D bounding box respectively. Here i is the sequence number of a loop.

Solids are formed by extruding faces, and multiple solids can be combined into a 3D model through basic Boolean operations. Here we simply use the set (S) of 3D solid bounding boxes $\left(x_i^t, y_i^t, z_i^t, x_i^b, y_i^b, z_i^b\right)$ to characterize the relationship between multiple solids, where $\left(x_i^t, y_i^t, z_i^t\right)$, $\left(x_i^b, y_i^b, z_i^b\right)$ denote the top left and bottom right corner of the 3D bounding box respectively. Here i is the sequence number of a solid.

Construction Sequence. A CAD construction sequence is understood from a user operation point of view to be from 1) Drawing a sketch, that is, drawing a face in a plane. Here we use L tokens and P tokens to simply represent the information contained in a face. This information will be used as a part of the code tree to guide the recovery of more detailed sketch operation parameters SO. 2) Then perform a series of 3D operations on the sketch to turn it into a model, here we only consider simple 3D operations: extrusion and Boolean operations. We only use the solid S tokens to simply represent the solids information of the model. This information will be used as a part of the code tree to guide the recovery of more detailed 3D operation parameters EO. In this way, a CAD construction sequence is represented as a long sequence containing only L, P, S tokens. The CAD reconstruction sequence is represented as a sequence SO, EO that recovers the geometric operation information (Section 3.4).

3.2. Regularized Discrete Codebooks

For codebook training, we use the VQ method [8, 26, 31] to compress continuous data in the latent space into discrete vectors and propose a new vector quantization adversarial network to train the regularized codebook.

Since we use the same network structure for training loops, profiles, and solids, for simplicity we denote the sequence as $\mathbf{A}$, which can be a loop sequence L, a profile sequence P or a solid sequence S. We use encoder E to encode the input sequence $\mathbf{A}$ and use the predicted sequence

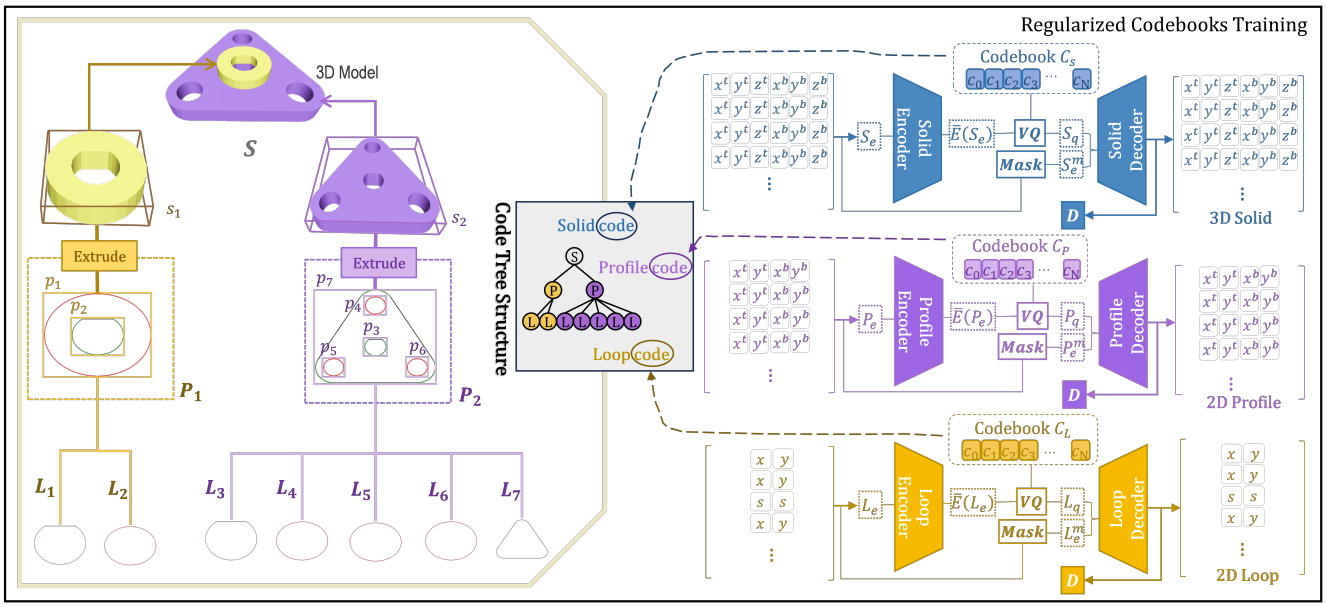

Figure 3. Our code tree is constructed from the codes in three discretized regularized codebooks, which are trained using vector quantization adversarial network. Specifically, we decompose the reconstructed sequence of the model into three distinct primitives: solid (S), profile (P), and loop (L), then train the network to extract three codebooks (right). Each codebook training network contains an encoder (E), a decoder (G), a discriminator (D), and a codebook (C). Then we represent the model as a tree structure, wherein each node is encoded with a specific code from the codebook. The code tree is constructed in a top-down manner $S-P-L$ (left).

of decoder G to perform adversarial training with the discriminator D to obtain the codebook C.

Encoder & Quantization. For the loop we quantize it to a plane of 64 $\times$ 64 pixels, so each token can be represented as a 64-dimensional one-hot vector, but the loop set L contains $\langle s \rangle$ tokens to separate the coordinate of the curves, so we need to add an extra dimension to differentiate them, and therefore use a 65-dimensional one-hot vector to represent each token in the loop. Each L is denoted as h^i_{L}. The symbol i denotes the index of the token in the loop, and this vector is used to uniquely identify the token information. For the P and S, we quantize each of their coordinate dimensions to 64 pixels, so each token can be represented using a 64-dimensional one-hot vector, denoted h^i_{P} and h^i_{S}, respectively.

Then, we convert these one-hot vectors h^i_A into embedding A^i_e by a multilayer perceptual network and utilize the sequence encoder E to obtain a 256-dimensional embedding $E(A^i_e)$. Next, we quantize the average pooling latent space embedding $\overline{E}(A^i_e)$ by the minimum distance matching technique in the codebook $\mathrm{C_A}$ and map the embeddings to the corresponding quantized code indices. This leads to the quantization code $A_{\mathbf{q}}$ as a component of the decoder input, as detailed in Equation 1.

$$A_{\mathbf{q}} \leftarrow c_k, k = \arg\min_{c_j \in \mathcal{C}_{\mathcal{A}}} \left\| \overline{E}(A^i_e) - c_j \right\|^2. \qquad (1)$$

Regularized Codebook. Through the above quantization process, we obtain a discretized representation of the potential space. However, in existing VQ-based methods [8, 26, 34, 39, 44], codebook collapse is often encountered when optimizing the code. In this case, only a few code vectors receive useful updates, whereas most remain largely unchanged or unused, as evident from the green "dead" spots in the bottom row of Figure 4. These "dead" code vectors are quantized only for specific vectors with little or no use. This phenomenon limits the effectiveness of VQ in learning larger code sets and the expressive power of code sets in complex reconstruction tasks that require high-capacity representations.

To address the above issues, we are inspired by the entropy regularization technique [4] from unsupervised clustering tasks [1, 20] and propose a generalized regularization approach to mitigate codebook collapse. The method continuously updates codes by minimizing the entropy of each code, which effectively avoids the problem of codebooks learning only specific vector representations, and thus improves the expressive and generalization capabilities of the codebook set. At the same time, we also increase the differences between codebooks by maximizing the entropy between different codes, which in turn improves the accuracy and diversity of the learned codebook features.

Specifically, we minimize the L_2 distance $\mathcal{D}_{i,j}$ between the feature embedding $\overline{E}(A^i_e)$ and the codebook $\mathrm{C_A}$ =

$\{c_j\}_{j=1}^{C}$. Subsequently, by applying a softmax normalization on $\mathcal{D}_{i,j}$ and then calculating the mean of these probabilities produced a frequency vector $\overline{\mathcal{D}}_j = \frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_{i,j}$ representing the overall utilization of each codebook item. Minimizing the entropy $H(\overline{\mathcal{D}})$ of this vector ensures the usage of the codebook is more balanced, which in turn enhances the generation quality and diversity:

$$H(\overline{\mathcal{D}}) = -\sum_{j}\overline{\mathcal{D}}_j \log \overline{\mathcal{D}}_j. \tag{2}$$

As shown in Figure 4, our approach generally improves the utilization of codebook under the same dataset and learns codebook shapes that are closer to the true data distribution. This dual strategy not only significantly improves the separability of the codebook, but also enhances the representation capability of per code, ensuring that each entry in the codebook adequately represents valid features and information, and avoids the problem of learning only specific information that fails to capture the true data distribution. By equalizing the representation capabilities of codebooks, our approach helps to learn feature representations more comprehensively, further mitigates codebook collapse, and greatly enriches the overall utility and performance of codebook.

Figure 4. The usage of codebook. We evaluated our method against the HNC-CAD [39] method on the DeepCAD [38] dataset. There are a large number of "dead" vectors (green dots) on the HNC-CAD [39], with some of the low-use vectors circled in black. In contrast, our method is able to effectively cluster similar features and separate dissimilar features by using entropy regularization, which enables our network to learn a codebook distribution that represents the true data distribution and significantly improves the utilization of the codebook.

Decoder & Adversarial Learning. To improve the generalization and robustness of the model, we use a noise injection mask [2, 5, 23, 33, 39] on the input of decoder G by adding noise to the initial input sequence $\mathbf{A}$, randomly masking 20% to 80% of the elements in the sequence, and encoding the modified sequence using a three-layer perceptron network to transform it into a 256-dimensional embedding to obtain A_e^m. We concatenate the quantization code $A_{\mathbf{q}}$ with the mask embedding sequence A_e^m and input it into the decoder G for predicting the sequence $\mathbf{A}$:

$$\hat{\mathbf{A}} = G(A_{\mathbf{q}} \,||\, A_e^m). \tag{3}$$

Since codebook quantization is a non-backpropagable process, we use $\mathbf{sg}$ to stop the gradient operator [8, 26, 31], which acts as an identifier in the forward computation but stops the flow of the gradient in the backward computation. This mechanism ensures that the gradient is more stable, thus facilitating model convergence and backpropagation during training. Additionally adding our entropy regularization loss term $\mathbf{H}(\overline{\mathcal{D}})$ in Equation 2 to the codebook loss optimizes the codebook distribution to improve the codebook representation. We can then obtain the end-to-end loss of our improved reconstructed codebook:

$$\begin{aligned}\mathcal{L}_{\mathrm{VQ}}(E,G,C) = \mathbf{EMD}(\mathbf{A},\hat{\mathbf{A}}) + \left\|\mathbf{sg}[\overline{E}(A_e)] - A_{\mathbf{q}}\right\|_2^2 \\ +\beta\left\|\mathbf{sg}\left[A_{\mathbf{q}}\right] - \overline{E}(A_e)\right\|_2^2 + \rho\mathbf{H}(\overline{\mathcal{D}}).\end{aligned} \tag{4}$$

The reconstruction loss is minimized using the Earth Moving Distance (EMD) loss to ensure that the distribution of the sequence $\hat{\mathbf{A}}$ generated by the decoder G is similar to the distribution of the original input sequence $\mathbf{A}$. The commitment loss is denoted by $\left\|\mathbf{sg}\left[A_{\mathbf{q}}\right] - \overline{E}(A_e)\right\|_2^2$. The hyperparameter β is set to 0.25 to scale the commitment loss. We fix ρ to 0.01 in our experiments to maintain balance and optimize performance.

We add a discriminator D to the decoded output, and use adversarial learning [10] to make the decoder G and discriminator D compete and collaborate with each other to optimize. This enables the generated results to better approximate the real distribution and is also helpful in alleviating the situation where the codebook only learns coarse-grained information, resulting in overall local inconsistency in the generated results, improving the realism of the generated results, and also enhancing the generalization ability of the model. The adversarial loss function is as follows:

$$\mathcal{L}_{\mathrm{GAN}}(\{E,G,C\},D) = [\log D(\mathbf{A}) + \log(1 - D(\hat{\mathbf{A}}))], \tag{5}$$

where $D(\mathbf{A})$ signifies the likelihood that $\mathbf{A}$ is deemed real by the discriminator, and $D(\hat{\mathbf{A}})$ signifies the likelihood that $\hat{\mathbf{A}}$ is considered real by the discriminator. In this equation, the goal of maximizing D is to maximize both terms, whereas the minimization of E, G, C seeks only to minimize the second term in Equation 5.

Loss Function. We train three codebooks $\mathrm{C_L}$,$\mathrm{C_P}$ and $\mathrm{C_S}$ respectively, in which the codebook size N is 4000, and

the training loss item is as follow:

$$\mathcal{L}(E,G,C,D) = \arg\min_{E,G,C}\max_{D}\mathbb{E}_{A\sim p(A)}[\mathcal{L}_{\mathrm{VQ}}(E,G,\mathcal{C}) + \lambda\mathcal{L}_{\mathrm{GAN}}(\{E,G,\mathcal{C}\},D)]. \quad (6)$$

The min-max portion reflects an adversarial training process, where the encoder E, decoder G, and codebook C collectively minimize, and the discriminator D maximizes separately.

Here, λ acts as a hyperparameter that adjusts the relative importance of the adversarial loss compared to the VQ loss. The determination of λ is contingent upon the generator's sensitivity to gradients in both $\mathcal{L}_{\mathrm{EMD}}$ and $\mathcal{L}_{\mathrm{GAN}}$. with a small positive constant δ to prevent division by zero. The $\mathcal{L}_{\mathrm{EMD}}$ term represents the reconstruction loss, computed as the EMD loss. $\nabla_{G_L}[.]$ represents the gradient of the last layer L of the generator.The equations for the computation of λ are as follows:

$$\lambda = \frac{\nabla_{G_L}\left[\mathcal{L}_{\mathrm{EMD}}\right]}{\nabla_{G_L}\left[\mathcal{L}_{\mathrm{GAN}}\right] + \delta}. \quad (7)$$

3.3. Code Tree Representation & Generation

After completing the above training, and before recovering the detailed geometric information through the codebook information, we follow the idea in SkexGen [41], HNC-CAD [39] to represent the sequence into the code sequence first.The input constructed sequence can be transformed into a code sequence using our trained encoder and regularized codebook quantization, taking the model on the left side of Figure 3 as an example, it can be transformed into the following code sequence: $[\langle ss\rangle, S, \langle sp\rangle, P_1, \langle sp\rangle, P_2, \langle sl\rangle, L_1, L_2, \langle sl\rangle, L_3, L_4, L_5, L_6, L_7, \langle eo\rangle]$, where L_i, P_i, and S are from the code in the codebook C_L,C_P,C_S, as well as the beginnings of the profile, loop, solid token $\langle sp\rangle$, $\langle sl\rangle$, $\langle ss\rangle$ and the end-of-sequence token $\langle eo\rangle$.We use the standard transformer's architecture [32], which generates sequences of code that is, each layer of the code tree, sequentially using 6 layers of attention mechanisms and 8 multiple heads of attention and guided by minimizing the loss of cross-entropy, so that the generation of the code tree T can be expressed as:

$$P(T) = P(T_L|T_S,T_P)P(T_P|T_S)P(T_S|\phi), \quad (8)$$

where T_S represents the code sequence at the solid layer of the code tree, T_P represents the code sequence at the profile layer of the code tree and T_L represents the code sequence at the loop layer of the code tree.

For code sequence generation we use kernel sampling [13] autoregressive generated sequences. The generated quantized sequences can be easily transformed into a tree structure using the separators in the sequences, as shown on the left side of Figure 3, which will serve as a guide for subsequent learning of the detailed model information.

3.4. CAD Construction Sequence Generation

Directly generating all geometric information simultaneously poses a significant challenge, even when utilizing a rough code tree as a foundation. Therefore, we adopt a sketch-and-extrude generation strategy that gradually recovers the parameters of the sketch and extrusion operations during model creation. We use the code tree as a guid and condition the step-by-step generation results. Specifically, we decompose the sequence of construction operations X into a product of conditional estimates for the stepwise generation with the following equation:

$$P(X) = P(EO|T,SO)P(SO|T), \quad (9)$$

where SO represents the recovered 2D sketch operation parameters, including curve type and curve parameters; EO represents the recovered 3D operation parameters, including extrusion and Boolean operation parameters. Since the networks for recovering 2D sketch operation parameters and 3D operation parameters share the same network architecture, with only slight differences in input, output data, and guided information, we will only provide a detailed introduction to the transformer network for sketch operation parameter recovery.

Specifically, we positionally encode the sketch operation parameters and convert them to a 256-dimensional embedding. An additional layer normalization is applied to the standard transformer encoder layer, while the remaining structures remain unchanged. Our encoder consists of 6 layers, each equipped with 8 multi-head attention mechanisms. For the decoder, we utilize the code tree as guided information to inform the recovery of sketch operation parameters. Similarly, an additional layer normalization is added to the standard transformer decoder layer, with the other structures remaining unchanged. The decoder consists of 6 layers, also with 8 multi-head attention mechanisms, as illustrated in Figure 5. For the recovery of 3D operation parameters, the input and output are the parameters of the 3D operation, with SO and the code tree serving as guidance information. Finally, we employ a combined cross-entropy loss to optimize two transformer networks.

During the generation process, we generate the geometric operation parameters based on the generated code tree as a guide and autoregressively recover the CAD construction sequences by kernel sampling method [13]. Subsequently, we transform the recovered CAD construction sequences into STEP files.

Loss Function. Our loss function is as follows, where $CE(.)$ is the cross-entropy loss, $\hat{SO}$, $\hat{EO}$ are the prediction parameters of sketch operation and 3D operation respectively:

$$\mathcal{L}_{\mathrm{CAD}} = \mathbf{CE}(SO,\hat{SO}) + \mathbf{CE}(EO,\hat{EO}). \quad (10)$$

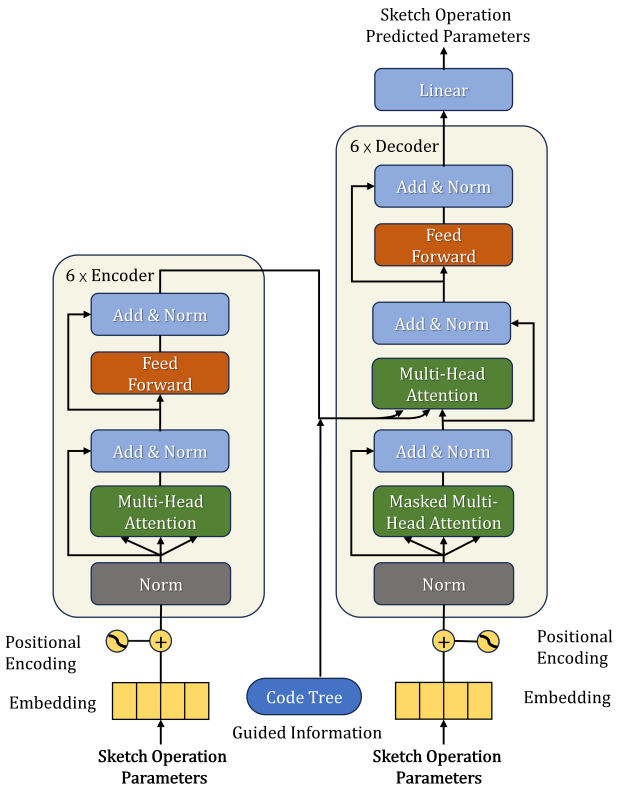

Figure 5. Improved conditionally guided transformer architecture for sketch operation parameters recovery.

4. Experiments

In this section we give the results of unconditional and conditional control generation. Numerous experimental results demonstrate that we can generate more realistic, high-quality, and more complex and diverse models compared to the state-of-the-art approach. We also provide richer means of conditional control generation including user editing, autocompletion and code tree controlled generation.

We implement the model using PyTorch and conduct experiments on an NVIDIA GeForce RTX 3090 GPU. We trained the improved 2D sketch transformer and the 3D transformer backbone with 256 batch sizes each for 350 epochs. We utilized the Adam optimizer for both generators, which was initialized with a learning rate of 0.001. This was followed by a linear warmup strategy over 2000 steps to gradually increase the learning rate.

4.1. Experimental Settings

Training Dataset. Our research makes full use of the DeepCAD dataset [38], which combines 178,238 sketch-and-extrude models. Divided into 90% training, 5% validation, and 5% test sets. To ensure the quality of the data, some preprocessing steps are performed. Duplicate models are detected and removed from the training set. In addition, the model is centered and rotated 45 degrees each time for a total of 8 rotations, starting from an isometric viewpoint. After each rotation, the image of one viewpoint and the model contour map in the picture are rendered and saved, from which the broken and invalid models containing multiple outer contours are removed. Post filtering, the training set retains 116,102 instances of sketch-and-extrude sequences and 94,191 solid sequences, 53,502 profile sequences, 135,588 loop sequences for codebooks' learning.

4.2. Evaluation Metrics

In order to quantitatively evaluate the generative models, five metrics were used to comprehensively evaluate the different generative methods. We convert the model to point cloud to compute three metrics - coverage (COV), minimum matching distance (MMD), and Jensen-Shannon divergence (JSD) employed in previous works [38, 39, 41]. The metrics Novelty (Nov) and uniqueness (Uniq), on the other hand, are improvements on the metrics of previous works, aiming to partially address the anomalous lowering of the metrics in [39, 41], and to further improve the accuracy and reasonableness of the assessment.

We observe that during training, sequences are transformed from a preprocessed real model, so that each sequence corresponds to a real and valid model. However, the situation is different during the generation process. Not all generated CAD sequence models are successfully transformed into valid CAD models. This also explains why sometimes the generated distributions appear to have been optimized and enhanced, but the results of the generated CAD models instead appear to be degraded on the Nov and Uniq metrics. To conduct a more precise evaluation of the generated models and address the anomalous experimental results mentioned earlier, we refined the evaluation criterion for the Nov and Uniq metrics by shifting the focus from the generated sequences to the generated valid models.

- COV measures the percentage of generated data that have at least one match to the real data after assigning each generated data to its nearest neighbor in the real data by Chamfer Distance (CD).
- MMD measures average minimum matching distance between generated sets and their corresponding real sets.
- JSD assesses similarity in marginal point distributions between true distributions and generated distributions.
- Novel quantifies the percent of successfully generated valid CAD models which is not found in the training set, reflecting the generative model's capacity to develop new designs.

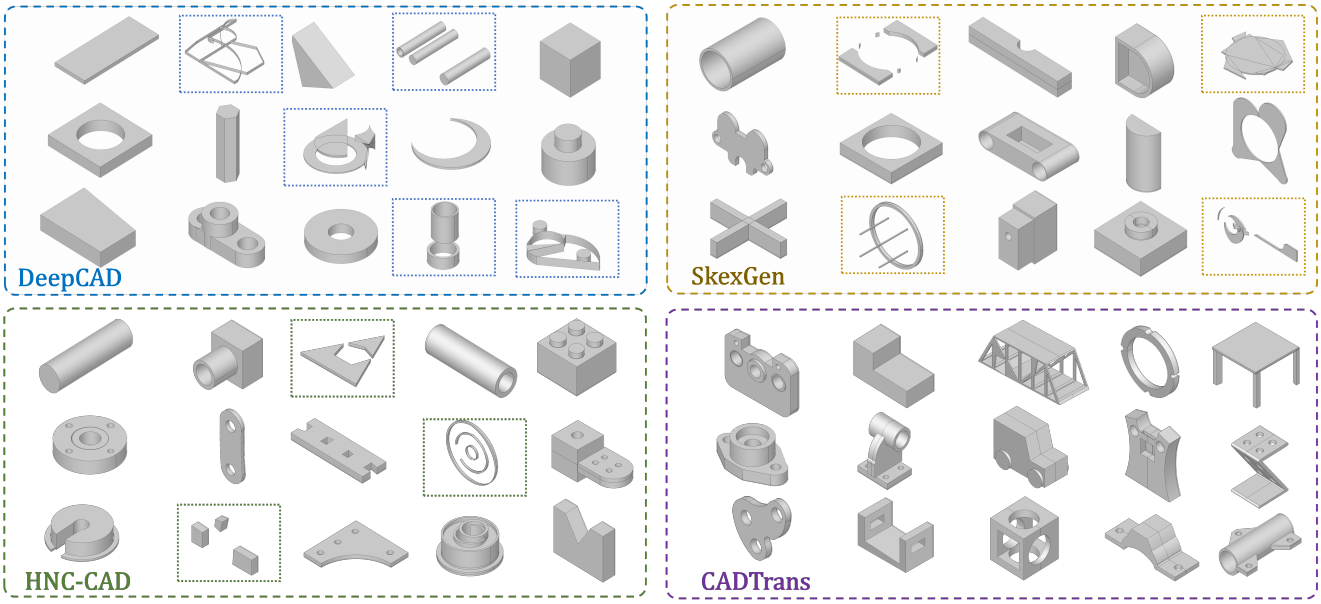

Figure 6. Randomly generated 3D models by DeepCAD [38], SkexGen [41], HNC-CAD [39] and CADTrans. A visual comparison shows that our method produces more complex and diverse models than existing methods and significantly reduces the probability of broken or damaged models. The unrealistic model is framed by the dashed box.

- Uniqueness represents the ratio of uniquely occurring instances in successfully generated valid CAD models, evaluating the generative data's diversity.

4.3. Unconditional Generation

We performed a quantitative and qualitative comparison of unconditional generation results with DeepCAD [38], SkexGen [41] and HNC-CAD [39]. To ensure the reliability of the comparison results, 10,000 CAD building sequences are randomly generated for each method and converted to B-rep format. At the same time, 2,500 real CAD models are randomly selected from the available test dataset and a series of quantitative metrics are calculated for these 10,000 generated models.

Ablation study. To further demonstrate its effectiveness, we also conducted ablation study, where NR stands for not applying regularization and NG stands for not applying adversarial training. Table 1 experimental results show that by employing adversarial training, our method is able to improve the quality and generalization ability of the codebook more effectively, and this improvement significantly boosts the COV metric. At the same time, the introduction of a regularized codebook makes the generated code distribution closer to the real distribution and improves the representation ability of the codebook, which leads to a significant decrease in the JSD metric.

Qualitative Evaluation. Figure 7 presents the qualitative results of our sketch generation, which produces more complex structures and diverse outputs compared to the best baseline model. Similarly, Figure 6 shows that our approach generates well-structured CAD models that closely resemble real-world models, with increased complexity and diversity in geometries and combinations. In particular, when the generator tries to generate models with unrealistic features or lacks coherent geometric features, both of which may lead to fragmentation and are considered unrealistic models. Our adversarial learning strategy will force the generator to learn to generate samples that are more

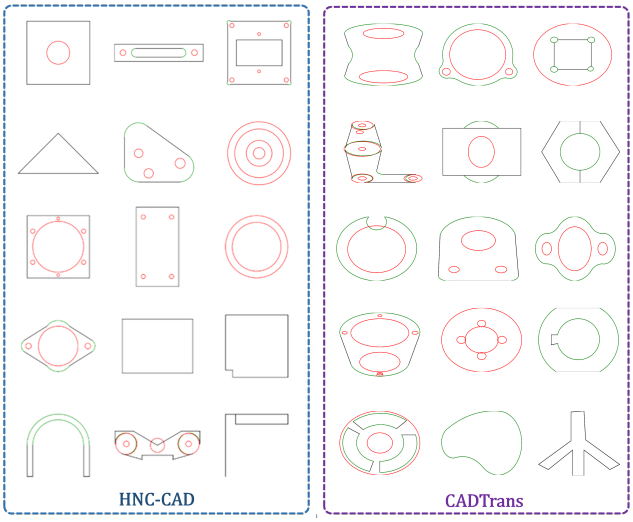

Figure 7. Random sketch generation results by HNC-CAD [39] and CADTrans. Our method utilizes richer elements in the generation of each sketch, resulting in more complex and realistic outcomes.

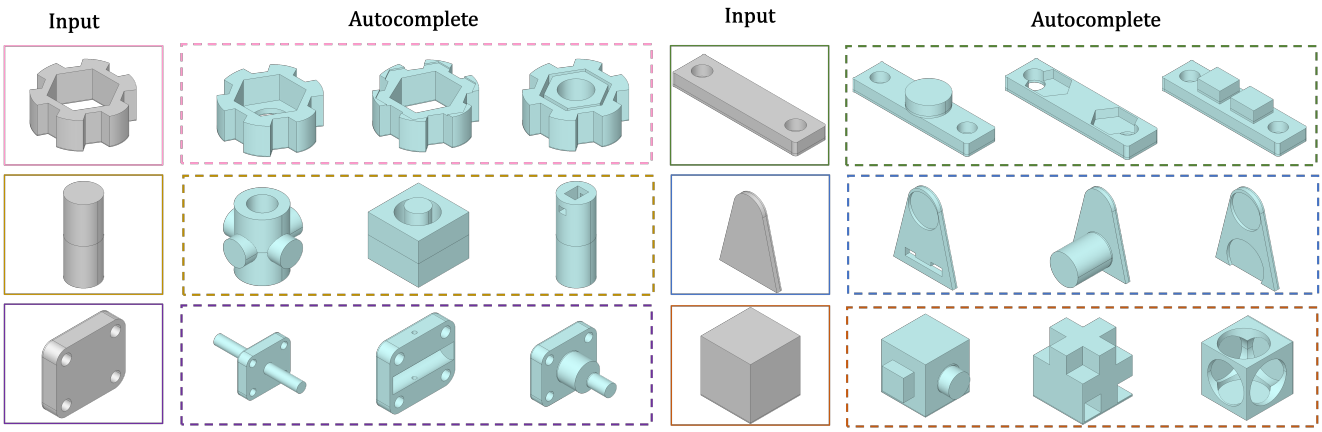

Figure 8. Autocompletion generations based on the provided model, maintaining the structure of the provided model while generating reasonable and human-like design.

consistent with the true distribution, become more complete, and reduce the generation of fragmented geometry. To fully demonstrate our approach, we provide more results for generated CAD models and sketchs in https://effieguoxufei.github.io/CADtrans/.

Method	COV (%) ↑	MMD ↓	JSD ↓	Nov (%)↑	Uniq (%)↑
DeepCAD	79.83	1.49	3.91	83.59	83.61
SkexGen	85.19	1.06	0.82	85.20	85.18
HNC-CAD	87.55	0.95	0.69	79.32	97.65
Ours(NR)	89.16	0.92	0.64	74.60	97.60
Ours(NG)	88.56	0.95	**0.58**	**88.49**	**99.21**
Ours	**89.54**	**0.94**	0.59	80.64	99.05

Table 1. Quantitative evaluations on CAD generation task with metrics: coverage percentage (COV), minimum matching distance (MMD), Jensen-Shannon divergence (JSD), novelty percentage (Nov), uniqueness (Uniq) percentage. The best result **bolded**.

Quantitative Evaluation. Table 1 shows the quantitative evaluation results on CAD generation. Our method outperforms existing works on all value matrices, indicating that our method has improved both quality and diversity.

4.4. Conditional Generation

The generation strategy of discrete regularized codebooks and autoregression of the transformer allows for conditional generation, so we give three main heuristics for the controlled generation: user editing, autocompletion, and code tree controlled generation.

4.4.1 User Editing

Figure 9 demonstrates the editing capabilities that allow the user to directly intervene and modify the model parameters in order to adjust various parts of the generated CAD model. During the iterative editing process, the modified parameters are encoded through the embedding layer along with other parameters of the original model and fed again into the generative network. The output of the network is the fine-tuned 3D model.

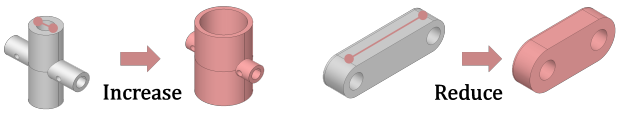

Figure 9. Edit model parameters such as radius and length. Editing changes the size in a responsive manner and aligns with the relevant structure.

For example, on the left side of Figure 9, the rest of the model is automatically adapted when the inner radius is changed. Throughout the operation, the network is able to sensitively change the dimensions and intelligently adapt the associated structures to optimize the hole locations while avoiding causing unnecessary occlusions or affecting the overall aesthetics and consistency. In addition, users can directly import the generated CAD model into existing CAD tools for further modification and editing.

4.4.2 Autocompletion

The autocompletion capability enables further design and detail refinement based on user input of an initial model of one or more parts, resulting in a 3D model with more complex geometry structures and details. We complement the code tree based on the user's input and input it with the existing CAD geometry sequence as conditional information into our code tree-guided transformer network to generate a new CAD model. As shown in Figure 8, it is possible to form more complex models based on a given model while maintaining the basic structure of the input model. The process and results of autocompletion of the model are

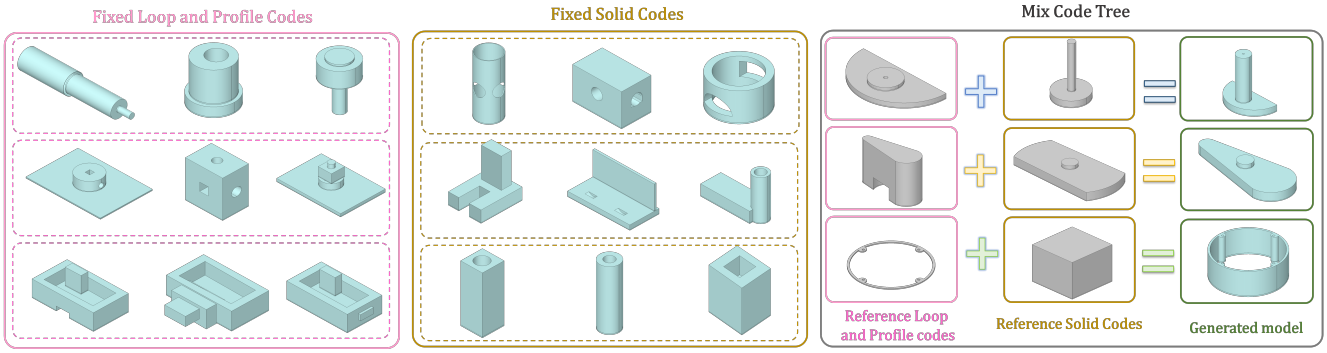

Figure 10. Mixed code tree showcase. Code generation results for fixed loop and profile codes (left), fixed solid codes (middle), and mixed code tree (right).

reasonable and produce results that are more in line with real-world design concepts.

4.4.3 Code Tree Controlled Generation

Target Code Tree Structure Generation. Figure 11 shows the generation results guided by the target code tree structure. For example, the first row in Figure 11 contains only solids consisting of two profiles for Boolean manipulation, and each profile contains only one loop. With this setup, we can manipulate the code structure under certain code constraints to achieve targeted model generation. This allows us to manipulate the code structure within certain code constraints, thus enabling targeted model generation.

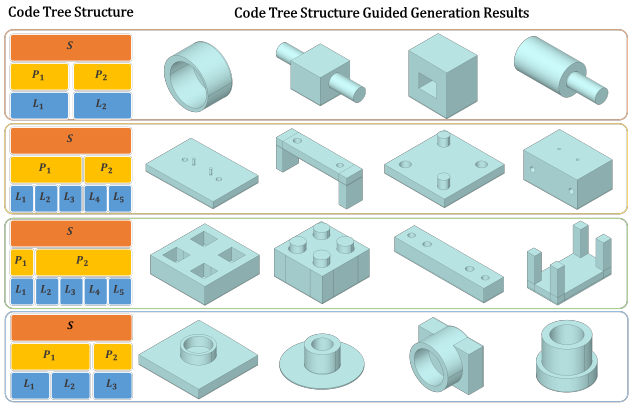

Figure 11. Target code tree structure generation. The first column is the structure of the code tree.

Mixed Code Tree Code Generation. Mixing codes from different code trees can generate entirely new 3D models. Locking down some of the code tree information helps to mimic a particular design style, thus generating similar design models and widening the design boundary. As shown in Figure 10, the " fixed loop and profile codes" illustrates a series of models that share the same 2D geometry parameters but have different extension parameters. For example, the sketches in the first row are all circles, but with different 3D operation parameters (including extension height and Boolean operation type). The "fixed solid codes" shows another set of models with similar 3D extrusion parameters. For example, the models in the third row all use the same 3D operations. In addition, the models shown in the " mixed code tree" are a mixture of loop and profile codes from different models as well as 3D solid codes, resulting in a new model that has both sketched features inherited from the supplied loop and profile codes and 3D operation parameters inherited from the supplied solid code. For example, the hybrid model generated at the end of the first line has the same sketched geometric elements as the model with the provided loop and profile code, and at the same time has the same 3D operations as the provided solid code.

5. Conclusion

In summary, we introduce CADTrans, a transformer-based autoregressive generative network guided by a code tree using three regularized discrete codebooks.Extensive experimental evaluations demonstrate the superiority of CADTrans over the state-of-the-art baseline, showcasing its ability to generate more realistic and diverse outputs. CADTrans supports both unconditional and conditional autoregressive generation, and its rich interactivity and control greatly improve the user experience.

Limitation. Since our method quantizes the model data to 64 pixels, data smaller than a certain threshold is treated as identical during the learning reconstruction process resulting in loss of detail. This limitation is similar to other sketch-and-extrude generation methods such as [39, 41].

Future Work. We provide various generation control methods that can provide generation results based on the user's explicit design intent. However, for users with unclear design goals, their code trees for result control are correspondingly vague. In order to fully utilize the

spark of inspiration, many methods are currently being investigated. [34] attempts to use a textual approach to control generation, but it only guarantees the inclusion of inspirational elements in the generated result, but does not guarantee the generation of meaningful results. [45] attempts to use semantic annotations of CAD programs to control CAD models, but is unable to reorganize the ordering of the input program. We envision that future research directions will focus on developing more effective control mechanisms, such as generating professional and realistic CAD models from textual descriptions or 2D images, in an effort to bridge the gap between design intent and generated results.

Acknowledgement

This work was supported by the National Key R&D Program of China (No. 2022YFB3303400), National Natural Science Foundation of China (Nos. 62272402, 62372389), Natural Science Foundation of Fujian Province (Nos. 2024J01513243, 2022J01001), and Fundamental Research Funds for the Central Universities (No. 20720220037).

References

[1] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 5

[2] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022. 6

[3] Z. Chen, A. Tagliasacchi, and H. Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 45–54, 2020. 3

[4] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 5

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186, June 2019. 6

[6] T. Du, J. P. Inala, Y. Pu, A. Spielberg, A. Schulz, D. Rus, A. Solar-Lezama, and W. Matusik. InverseCSG: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018. 3

[7] E. Dupont, K. Cherenkova, A. Kacem, S. A. Ali, I. Arzhannikov, G. Gusev, and D. Aouada. CADOps-Net: Jointly learning cad operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*, pages 114–123. IEEE, 2022. 3

[8] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 2, 3, 4, 5, 6

[9] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao. MeshNet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8279–8286, 2019. 2

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 3, 6

[11] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 2

[12] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. MeshCNN: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 2

[13] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. 7

[14] P. K. Jayaraman, J. G. Lambourne, N. Desai, K. D. Willis, A. Sanghi, and N. J. Morris. SolidGen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research (TMLR)*, 3, 2023. 2

[15] P. K. Jayaraman, A. Sanghi, J. G. Lambourne, K. D. Willis, T. Davies, H. Shayani, and N. Morris. UV-Net: Learning from boundary representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11703–11712, 2021. 3

[16] K. Kania, M. Zieba, and T. Kajdanowicz. UCSG-NET-Unsupervised discovering of constructive solid geometry tree. *Advances in neural information processing systems*, 33:8776–8786, 2020. 3

[17] C. Li, H. Pan, A. Bousseau, and N. J. Mitra. Free2CAD: Parsing freehand drawings into cad commands. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 3

[18] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019. 3

[19] P. Li, J. Guo, X. Zhang, and D.-M. Yan. SECAD-Net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16826, 2023. 4

[20] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8547–8555, 2021. 5

[21] D. Maturana and S. Scherer. VoxNet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 2

[22] C. Nash, Y. Ganin, S. A. Eslami, and P. Battaglia. PolyGen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020. 4

[23] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 6

[24] C. R. Qi, H. Su, K. Mo, and L. J. 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. 2

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[26] A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 2, 3, 4, 5, 6

[27] D. Ren, J. Zheng, J. Cai, J. Li, H. Jiang, Z. Cai, J. Zhang, L. Pan, M. Zhang, H. Zhao, et al. CSG-Stump: A learning friendly csg-like representation for interpretable shape parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12478–12487, 2021. 3

[28] D. Ren, J. Zheng, J. Cai, J. Li, and J. Zhang. ExtrudeNet: Unsupervised inverse sketch-and-extrude for shape parsing. In *European Conference on Computer Vision*, pages 482–498. Springer, 2022. 3, 4

[29] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, pages 1–3, 2018. 2

[30] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, and S. Maji. CSGNet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018. 3

[31] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, pages 6306–6315, 2017. 2, 3, 4, 6

[32] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Neural Information Processing Systems*, pages 5–13, 2017. 3, 7

[33] P. Vincent Pascalvincent, L. Larocheh, and H. S. D. Autoencoders. Learning useful representations in a deep network with a local denoising criterion pierre-antoine manzagol. *J. Mach. Learn Res*, 11:3371–3408, 2010. 6

[34] H. Wang, M. Zhao, Y. Wang, W. Quan, and D.-M. Yan. VQ-CAD: Computer-Aided Design model generation with vector quantized diffusion. *Computer Aided Geometric Design*, 111:102327, 2024. 5, 12

[35] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-CNN: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017. 2

[36] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong. Adaptive O-CNN: A patch-based deep representation of 3d shapes. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 2

[37] K. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24, 2021. 2, 3, 4

[38] R. Wu, C. Xiao, and C. Zheng. DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021. 2, 4, 6, 8, 9

[39] X. Xu, P. K. Jayaraman, J. G. Lambourne, K. D. Willis, and Y. Furukawa. Hierarchical neural coding for controllable cad model generation. In *International Conference on Machine Learning*, pages 38443–38461, 2023. 2, 4, 5, 6, 7, 8, 9, 11

[40] X. Xu, J. Lambourne, P. Jayaraman, Z. Wang, K. Willis, and Y. Furukawa. BrepGen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024. 2

[41] X. Xu, K. D. Willis, J. G. Lambourne, C.-Y. Cheng, P. K. Jayaraman, and Y. Furukawa. SkexGen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022. 2, 4, 7, 8, 9, 11

[42] Y. Ye, Y. Wang, J. Cao, and Z. Chen. Watertight surface reconstruction method for cad models based on optimal transport. *Computational Visual Media*, 10(5):859–872, 2024. 2

[43] F. Yu, Z. Chen, M. Li, A. Sanghi, H. Shayani, A. Mahdavi-Amiri, and H. Zhang. CAPRI-Net: Learning compact cad shapes with adaptive primitive assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11768–11778, 2022. 3

[44] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu. Vector-Quantized Image Modeling with Improved VQGAN. 2, 2022. 5

[45] H. Yuan, J. Xu, H. Pan, A. Bousseau, N. J. Mitra, and C. Li. CADTalk: An algorithm and benchmark for semantic commenting of cad programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3753–3762, 2024. 12