CNC-Net: Self-Supervised Learning for CNC Machining Operations

Mohsen Yavartanoo[1*] Sangmin Hong[2*] Reyhaneh Neshatavar[1*] Kyoung Mu Lee[1,2]
[1]Dept. of ECE & ASRI, [2]IPAI, Seoul National University, Seoul, Korea
{myavartanoo,mchiash2,reyhanehneshat,kyoungmu}@snu.ac.kr

Abstract

CNC manufacturing is a process that employs computer numerical control (CNC) machines to govern the movements of various industrial tools and machinery, encompassing equipment ranging from grinders and lathes to mills and CNC routers. However, the reliance on manual CNC programming has become a bottleneck, and the requirement for expert knowledge can result in significant costs. Therefore, we introduce a pioneering approach named CNC-Net, representing the use of deep neural networks (DNNs) to simulate CNC machines and grasp intricate operations when supplied with raw materials. CNC-Net constitutes a self-supervised framework that exclusively takes an input 3D model and subsequently generates the essential operation parameters required by the CNC machine to construct the object. Our method has the potential to transformative automation in manufacturing by offering a cost-effective alternative to the high costs of manual CNC programming while maintaining exceptional precision in 3D object production. Our experiments underscore the effectiveness of our CNC-Net in constructing the desired 3D objects through the utilization of CNC operations. Notably, it excels in preserving finer local details, exhibiting a marked enhancement in precision compared to the state-of-the-art 3D CAD reconstruction approaches. The codes are available at https://github.com/myavartanoo/CNC-Net_PyTorch.

1. Introduction

Manufacturing processes have undergone remarkable transformations over the past decades, driven by automation and the advancement of computational techniques. A domain that has witnessed substantial innovation is Computer Numerical Control (CNC) machining, a pivotal pillar of modern manufacturing. CNC machines have revolutionized manufacturing by producing complex products with better precision, efficiency, and robustness [1] in diverse industries, from aerospace to medical devices. Despite their numerous advantages, CNC machines still grapple with certain limitations, particularly in manual programming and adaptability. Traditional CNC programming requires intricate sets of instructions crafted by Computer-Aided Manufacturing (CAM) software that guide machine tools, including mills and drills, to produce the intended object. However, despite its effectiveness, this process introduces bottlenecks due to its labor-intensive nature and reliance on expert knowledge. Furthermore, adapting CNC machines to new tasks typically involves extensive reprogramming, hindering their agility and responsiveness in dynamic manufacturing environments. Incorporating deep learning techniques into CNC machining offers a transformative solution to address these challenges. In particular, several recent studies use deep neural networks (DNNs) to explore 3D objects using Constructive Solid Geometry (CSG) [19] operations, employing both a set of simple [31, 34] and more complex [6, 36, 37] primitives. Therefore, the ability of DNNs to learn complex patterns from data makes them an ideal candidate for revolutionizing CNC manufacturing, which can pave the way for automation, adaptive programming, and efficient utilization of CNC machines. However, the intricate search space for operations on complex objects involving NP-hard problems presents a challenge in labeling optimal solutions as ground truth (GT). Consequently, lacking a dataset with such a GT as supervision poses challenges in training a DNN model.

To mitigate these challenges, we propose CNC-Net, a DNN-based approach designed to simulate generic CNC machines in a self-supervised manner. Our approach can construct target objects without relying on the GT labels (*i.e.*, a set of sequential operations). CNC-Net is structured to incrementally learn the production of target 3D shapes, thereby determining the subsequent set of operations by implicitly modeling milling and drilling operations. This capability enables CNC-Net to generate the necessary machining steps effectively. At each operational step, the tools are represented as cylindric primitives, and the CNC-Net determines the radius of the tool and identifies the path coordinates for the subsequent milling or drilling action. To

*equal contribution

enhance the carving capabilities of a CNC machine, we introduced a feature that enables the machine to rotate the workpiece along the X and Y axes. This functionality is commonly found in advanced CNC setups. In this scenario, guided solely by the target shape and lacking prior information or labeled operations, CNC-Net models 3D shapes at each step, involving subtracting operations, represented as the union of cylindric primitives, from the outcomes of preceding steps. This enables CNC-Net to learn the essential operations to reconstruct the target shapes accurately.

In our experiments, we provide the competitive performance of CNC-Net in reproducing 3D shapes compared to state-of-the-art (SOTA) CAD reconstruction methods. To validate the effectiveness of our approach, we conduct experiments on both industrial objects from the ABC dataset [16] and more intricate objects obtained from the ShapeNet dataset [3]. Our main contributions are threefold:

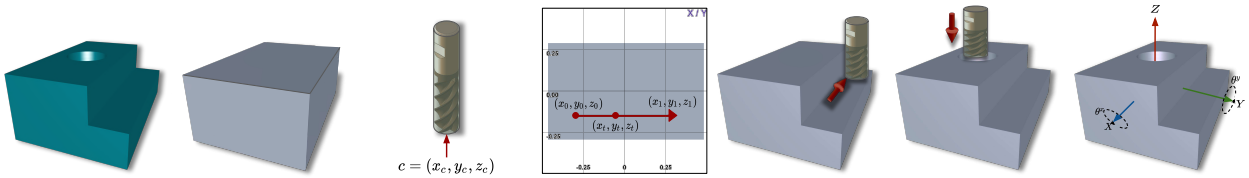
- We introduce CNC-Net, a pioneer self-supervised and DNN-based approach for simulating CNC machines.
- CNC-Net learns to automatically find the sequential operations required for sculpting a 3D shape and exhibits capability akin to expert human labor without the need for labels or any prior information.
- The experiments demonstrate that our self-supervised CNC-Net method can precisely reproduce target objects and outperform SOTA methods in terms of 3D reconstruction performance based on volume-based metrics.

2. Related Works

This section covers previous studies related to our method, divided into two categories: reverse engineering of 3D shapes and machine learning for CNC machines.

Reverse engineering 3d shapes. Reverse engineering a 3D shape refers to understanding the features and structure of the original object and learning how it is constructed. With the development of deep learning, several approaches have been proposed to investigate how a 3D shape is assembled. In recent years, there has been an exploration of the use of simple geometric primitives to approximate a 3D shape with a pre-defined set of cubes [27, 33, 40], ellipsoids [10]. More recent studies improve the representation ability and surface reconstruction by introducing more flexible and deformable primitives [6, 13, 28, 36]. These works represent a shape as a union of primitives using constructive solid geometry (CSG) [19], which relies on Boolean operations applied to the primitives [9]. On the other hand, there exist various methods [5, 8, 31] that assemble primitives using a sequence of modeling operations through reinforcement learning (RL). These methods aim to match a target shape in a reverse engineering manner. Furthermore, recent supervised primitive networks [22, 32] have been designed to detect and fit primitives within point clouds, which initially identify primitive types. Subsequently, they estimate their parameters or integrate spline patches, incorporating differentiable metric-learning segmentation. Additionally, CSGNet [31] is a neural network approach to form a CSG program from a given shape, and InverseCSG [8] solves it as a program synthesis problem. Later methods [14, 23, 29, 30, 37] learn to compact 3D computer-aided design (CAD) models via CSG operations, including intersection, union, and subtraction, without relying on any ground-truth primitive assemblies. However, primitive and implicit-based methods are often designed for static shape reconstruction rather than dynamic processes such as material removal in machining. Accordingly, none of the above-mentioned methods is applicable for modeling operations in CNC machining. In contrast, our self-supervised method can dynamically reproduce target shapes by learning sequential CNC machining operations.

Machine learning for CNC machines. Computer-aided process planning (CAPP) by utilizing machine learning (ML) approaches plays a crucial role in streamlining and automating various stages of the manufacturing process [17, 20]. It encompasses critical machining processes such as toolpath optimization, feature selection, tool selection, operation selection, etc. ML-based methods [7, 12, 21, 26, 35, 39] within CAPP have focused on improving toolpath generation and optimization, contributing to manufacturing processes. In [12], particle swarm optimization (PSO) is utilized to optimize randomly initialized toolpaths to minimize the machining time and resource consumption. Furthermore, a recent study [7] utilizes support vector machine (SVM) for error prediction in toolpath generation, ensuring that the toolpaths generated meet the quality and precision requirements. Such optimization-based methods enhance the machining performance and reduce production costs, but they lack the capability to learn the CNC operations themselves. Moreover, [2, 18] has been proposed to select the best route planning strategy from conventional approaches, helping to generate a path aligned with specific manufacturing objectives. However, these methods require data preparation involving 3D CAD models, each with corresponding labels representing the generated toolpaths, which is time-consuming. Moreover, such models are limited to learning only from these specific labels and restricting their generalization to adapt and generate toolpaths across different datasets. Consequently, direct research on automatic search and learning to generate CNC machining operations step-by-step has been relatively scarce. Therefore, we introduce a novel self-supervised framework, CNC-Net, to simulate generic CNC machines by generating toolpaths and learning the operations sequentially solely from 3D CAD models without requiring any labeled data or prior information. Our proposed method



(a) **Target Shape.** (b) **Input Material.** (c) **Tool.** (d) **Path.** (e) **Milling.** (f) **Drilling.** (g) **Rotation.**

Figure 1. **Overview of a generic CNC machine features.**

can be applied to diverse datasets containing 3D CAD models and shows the ability to learn the operations, even from a single sample, using zero-shot learning.

3. Method

In this section, we first provide an overview of a generic CNC machine by explaining its basic principles. Later, we introduce our novel framework, CNC-Net, which takes an input 3D model and attempts to generate the sequence of operations required for the CNC machine to reproduce the designated shape in a fully self-supervised manner.

3.1. CNC machine

Computer Numerical Control (CNC) is a computer-based system used to control various machining tools such as drills and mills. It operates using pre-programmed instructions, such as the G-code or M-code, to shape materials in the desired shape $\mathcal{S}$ as Fig. 1a. These instructions include a sequence of operations that can be created by individuals, computer-aided design (CAD) systems, or computer-aided manufacturing (CAM) software. In this study, we only consider CNC machines that use milling and drilling processes applied from the top of the workpiece and include rotation capability as a typical machining operation.

3.1.1 Input material

As shown in Fig. 1b, we implicitly model the material initially provided, denoted as $\mathcal{S}_0$, as the bounding box $\mathcal{B}(\mathcal{S})$ that encompasses the target object $\mathcal{S}$ as follows:

$$\mathcal{S}_0 = \mathcal{B}(\mathcal{S}) = \textbf{max}(|\frac{x}{l}|, |\frac{y}{w}|, |\frac{z}{h}|) - 1, \tag{1}$$

where l, w, and h represent the length, width, and height of the target object $\mathcal{S}$, respectively.

3.1.2 Tool

As shown in Fig. 1c, for simplicity and without loss of generality, we model milling and drilling tools $\mathcal{T}$ using the implicit form of cylindrical primitives represented as follows:

$$\mathcal{T}(c, r) = \textbf{max}\{(x - c_x)^2 + (y - c_y)^2 - r^2, c_z - z\}, \tag{2}$$

where r and $c = (c_x, c_y, c_z)$ denote the radius of the tool and the center of the base of the cylinder, respectively.

3.1.3 Path

The milling tool can move along a specified path to carve or cut through the material. As shown in Fig. 1d, we represent the path using a parametric function $\mathcal{P}$ as follows:

$$\mathcal{P}(t) = (c_x(t), c_y(t), c_z), \tag{3}$$

where $c_x(t)$ and $c_y(t)$ represent the parametric components for each axis X and Y, with the parameter t, respectively. It should be noted that the component c_z is constant, indicating the depth of penetration of the tool into the workpiece.

3.1.4 Milling operation

Milling is the process of removing material from a workpiece using a rotating cutting tool, often with multiple edges layer by layer, as shown in Fig. 1e. It is commonly used to create complex contours, pockets, and slots. Consequently, milling operation $\mathcal{O}_s^{\mathcal{M}}$ at step s can be defined as the process of a tool $\mathcal{T}_s$ traversing a path $\mathcal{P}_s(t)$, expressed as follows:

$$\mathcal{O}_s^{\mathcal{M}} = \textbf{min}\{\mathcal{T}_s(\mathcal{P}_s(t), r_s)\}_{t=0}^{1}, \tag{4}$$

where r_s is the radius of the tool at step s and the **min** operator encompasses the union of the primitives $\mathcal{T}_s$ across path parameterized by time steps t ranging from 0 to 1.

3.1.5 Drilling operation

Drilling is a machining process that uses a rotating drill bit to create holes with specific sizes and depths in a workpiece. This operation is essential to accommodate fasteners such as bolts and screws in various applications. Unlike the milling operation, which traverses along a specific path, the drilling tool $\mathcal{T}_s$ at step s penetrates a designated location $c_s = (c_{s,x}, c_{s,y}, c_{s,z})$ as shown in Fig. 1f. As a result, the drilling operation $\mathcal{O}_s^{\mathcal{D}}$ at step s can be defined as follows:

$$\mathcal{O}_s^{\mathcal{D}} = \mathcal{T}_s(c_s, r_s), \tag{5}$$

where r_s represents the radius of the drilling tool at step s.

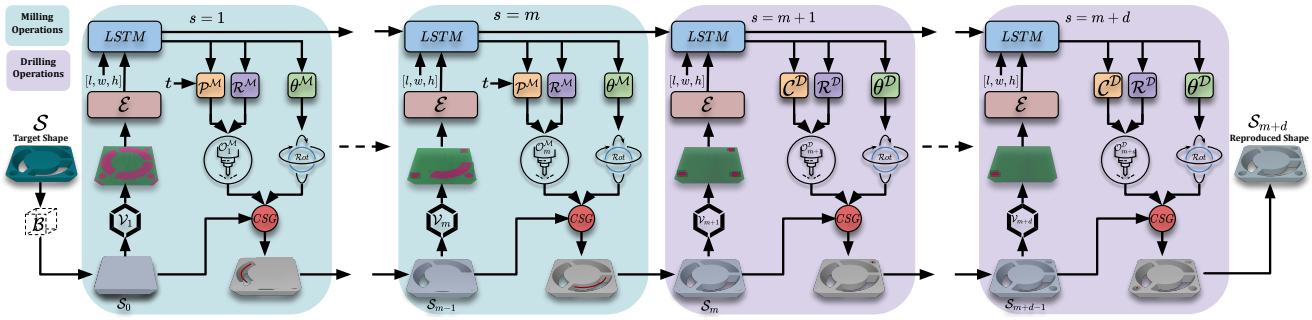


Figure 2. **Overview of our proposed framework.** Subsequent of milling and drilling operations to reproduce a target 3D CAD model.

3.1.6 Rotation operation

CNC machines are equipped with the ability to rotate the workpiece at different angles, enhancing their versatility, precision, and effectiveness in addressing complex geometries and intricate machining tasks while eliminating the need for manual repositioning. As shown in Fig. 1g, we assume that the machine can rotate the shape $\mathcal{S}_s$ at step s by the rotation transformation $\mathcal{R}ot_{\{\theta_s^x,\theta_s^y\}}$ as follows:

$$\mathcal{S}_s^{\mathcal{R}ot} = \mathcal{R}ot_{\{\theta_s^x,\theta_s^y\}}(\mathcal{S}_s), \tag{6}$$

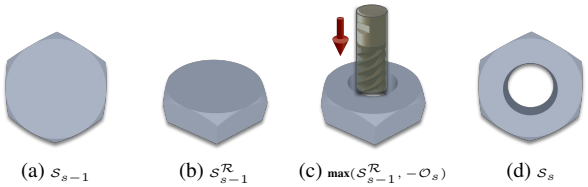
where θ_s^x and θ_s^y denote the counterclockwise rotation angles about the X-axis and Y-axis at step s, respectively.

3.2. CNC-Net

In this section, we introduce CNC-Net, an innovative self-supervised framework designed to automate the operations of generic CNC machines. CNC-Net achieves this by autonomously learning and refining essential parameters for sequential operations, including milling, drilling, and rotation, to sculpt raw materials into intricate objects precisely. Fig. 2 provides a comprehensive visual representation of our proposed self-supervised framework, CNC-Net. We start with voxelizing the input material $\mathcal{S}_0$ as the bounding box $\mathcal{B}(\mathcal{S})$ of the target shape $\mathcal{S}$ as follows:

$$\mathcal{V}_0(v) = \begin{cases} +1, & \text{if } v \in (\mathcal{S}_0 - \mathcal{S}) \\ -1, & \text{if } v \in \mathcal{S} \end{cases}. \tag{7}$$

Subsequently, we utilize an encoder $\mathcal{E}$ to extract both local and global features from $\mathcal{V}_0$, where the encoded features are concatenated with the bounding box size $[l, w, h]$ and fed into a long short-term memory (LSTM) network denoted as $\mathcal{A}$. This LSTM network generates hidden features for the subsequent operational step and produces the necessary output features to predict the parameters of the operations. Moreover, we utilize three distinct decoders $\theta^{\mathcal{M}}$, $\mathcal{R}^{\mathcal{M}}$, and $\mathcal{P}^{\mathcal{M}}$ to generate the rotation parameters (θ_1^x, θ_1^y), tool radius r_1, and path $\mathcal{P}_1(t)$ parameterized by t for the first step $s = 1$, respectively. Then, we construct the milling operation $\mathcal{O}_1^{\mathcal{M}}$ from the generated r_1 and $\mathcal{P}_1(t)$ and feed it along with $\mathcal{S}_0$ and the rotation parameters (θ_1^x, θ_1^y) into the Constructive Solid Geometry (CSG) operation, executing $\mathcal{O}_1^{\mathcal{M}}$ and the rotation $\mathcal{R}ot_{\{\theta_1^x,\theta_1^y\}}$ resulting in the generation of $\mathcal{S}_1$. We continue this process with milling operations $\mathcal{O}_s^{\mathcal{M}}$ iteratively for m steps until no further improvement is achieved and generate the shape $\mathcal{S}_m$ at step $s = m$, wherein we update the voxel representation as follows:



(a) $\mathcal{S}_{s-1}$ (b) $\mathcal{S}_{s-1}^{\mathcal{R}}$ (c) $\max(\mathcal{S}_{s-1}^{\mathcal{R}}, -\mathcal{O}_s)$ (d) $\mathcal{S}_s$

Figure 3. **CSG operation.**

$$\mathcal{V}_{s-1}(v) = \begin{cases} +1, & \text{if } v \in (\mathcal{S}_{s-1} - \mathcal{S}) \\ -1, & \text{if } v \in \mathcal{S} \cup (\mathcal{S}_0 - \mathcal{S}_{s-1}) \end{cases}. \tag{8}$$

When the milling operations are completed, we proceed to perform the drilling operations similarly, where the generated shape $\mathcal{S}_m$ is considered as the initial material for the drilling operations. Here, we input the updated voxel representation $\mathcal{V}_m$ into the same encoder $\mathcal{E}$ and feed the extracted features along with the bounding box size $[l, w, h]$ to the LSTM network $\mathcal{A}$ to extract the characteristics, but different decoders $\theta^{\mathcal{D}}$, $\mathcal{R}^{\mathcal{D}}$, and $\mathcal{C}^{\mathcal{D}}$ to produce the rotation parameters $(\theta_{m+1}^x, \theta_{m+1}^y)$, the drill radius r_{m+1}, and the drill tip coordinates c_{m+1}, respectively. Similar to the milling operation, $\mathcal{S}_{m+1}$ in the subsequent step $s = m + 1$ can be generated by feeding $\mathcal{S}_m$ along with the constructed drilling operation $\mathcal{O}_{m+1}^{\mathcal{D}}$ and the generated rotation parameters $(\theta_{m+1}^x, \theta_{m+1}^y)$ into the CSG operation. We iterate the drilling operations over d steps until a notable similarity is achieved between $\mathcal{S}_{m+d}$ and the target shape $\mathcal{S}$.

3.2.1 CSG

Fig. 3 shows the outline of the CSG module. To construct the shape $\mathcal{S}_s$ for each step s, the initial procedure involves applying the rotation transformation $\mathcal{R}ot_{\{\theta_s^x,\theta_s^y\}}$ to $\mathcal{S}_{s-1}$ as

in Eq. (6), where the resulting rotated shape is denoted as $\mathcal{S}_{s-1}^{\mathcal{R}ot}$. Accordingly, the shape $\mathcal{S}_s$ can be constructed through the subtraction of a milling operation $\mathcal{O}_s^{\mathcal{M}}$ or a drilling operation $\mathcal{O}_s^{\mathcal{D}}$ from the rotated shape $\mathcal{S}_{s-1}^{\mathcal{R}ot}$ followed by the inverse rotation $\mathcal{R}ot^{-1}_{\{\theta_s^x,\theta_s^y\}}$ to ensure that its orientation matches the initial orientation as follows:

$$\begin{aligned}\mathcal{S}_s &= \mathcal{R}ot^{-1}_{\{\theta_s^x,\theta_s^y\}}(\textbf{max}(\mathcal{S}_{s-1}^{\mathcal{R}ot}, -\mathcal{O}_s^{\mathcal{M}})) \\ \mathcal{S}_s &= \mathcal{R}ot^{-1}_{\{\theta_s^x,\theta_s^y\}}(\textbf{max}(\mathcal{S}_{s-1}^{\mathcal{R}ot}, -\mathcal{O}_s^{\mathcal{D}}))\end{aligned} \text{ or,} \quad (9)$$

where the **max** operator encompasses the subtraction.

3.3. Loss functions

We set up various loss functions for self-supervised training of our model. For simplicity, we represent $||.||$ as the L^2 norm, $v_{xyz} \in \mathbb{R}^3$ as the 3D coordinates of the corresponding voxel v, $v_{xyz}^{\mathcal{R}ot} \in \mathbb{R}^3$ and $v_{xy}^{\mathcal{R}ot} \in \mathbb{R}^2$ as the 3D and 2D coordinates of the rotated v_{xyz} by rotation operation $\mathcal{R}ot_{\{\theta_s^x,\theta_s^y\}}$, and $\sigma(x) = \tanh(wx)$ as a smooth sign function, with w serving as a large scaling factor. First, we design the milling loss $\mathcal{L}^{\mathcal{M}}$ to ensure that the implicit shape $\mathcal{S}_m$ obtained after sequential milling precisely approximates the target shape $\mathcal{S}$. Consequently, $\mathcal{S}_m$ is expected to produce negative and positive values for voxels $v \in \mathcal{V}_0$ inside and outside the target shape $\mathcal{S}$ as follows:

$$\mathcal{L}^{\mathcal{M}} = \frac{1}{|\mathcal{V}_0|}\sum_{v\in\mathcal{V}_0} ||\sigma(\mathcal{S}_m(v_{xyz})) - \mathcal{V}_0(v)||^2, \quad (10)$$

where $|\mathcal{V}_0|$ denotes the total number of voxels in $\mathcal{V}_0$.

We further define the drilling loss $\mathcal{L}^{\mathcal{D}}$ to facilitate the drilling operations to remove the remaining regions after the sequential milling operations as follows:

$$\mathcal{L}^{\mathcal{D}} = \frac{1}{d}\sum_{s=m+1}^{m+d}\frac{1}{|\mathcal{V}_{s-1}^+|}\sum_{v\in\mathcal{V}_{s-1}^+} ||\sigma(\mathcal{O}_s^{\mathcal{D}}(v_{xyz}^{\mathcal{R}ot})) + 1||^2, \quad (11)$$

where $\mathcal{V}_{s-1}^+$ is the subset of $\mathcal{V}_{s-1}$ whoes values are positive $(+1)$ and $|\mathcal{V}_{s-1}^+|$ denotes its number of voxesls.

To ease training and improve performance, we also introduce a shape loss $\mathcal{L}^{\mathcal{S}}$ and a center loss $\mathcal{L}^{\mathcal{C}}$ as complementary losses for both operations. $\mathcal{L}^{\mathcal{S}}$ ensures that the regions inside the target shape $\mathcal{S}$ are not removed by any milling or drilling operations, and therefore $\mathcal{S}$ is preserved after sequential operations. Therefore, we need to ensure that the milling or drilling operation $\mathcal{O}_s$ at each step s produces positive values for the voxels inside $\mathcal{S}$ as follows:

$$\mathcal{L}^{\mathcal{S}} = \frac{1}{|\mathcal{V}_0^-|}\sum_{s=1}^{m+n}\sum_{v\in\mathcal{V}_0^-} ||\sigma(\mathcal{O}_s(v_{xyz}^{\mathcal{R}ot})) - 1||^2, \quad (12)$$

where $\mathcal{V}_0^-$ is the subset of $\mathcal{V}_0$ whose values are negative (-1) and $|\mathcal{V}_0^-|$ denotes its number of voxels. Furthermore, $\mathcal{L}^{\mathcal{C}}$ is applied from the top of the workpiece to ensure that the decoders $\mathcal{P}^{\mathcal{M}}$ and $\mathcal{C}^{\mathcal{D}}$ generate tooltip coordinates around the remaining regions that are yet to be removed. This loss simplifies the training process by narrowing down the search space for the decoders $\mathcal{P}^{\mathcal{M}}$ and $\mathcal{C}^{\mathcal{D}}$, facilitating the generation of tip coordinates. Consequently, we minimize the Chamfer distance between the 2D coordinates C_s^{xy} of the tools tips at each step s, *e.g.*, $C_s^{xy} = \{(c_{s,x}(t), c_{s,y}(t))\}_{t=0}^{1}$ for the milling path and $C_s^{xy} = (c_{s,x}, c_{s,y})$ for the drilling tip and the 2D coordinates $v_{xy}^{\mathcal{R}ot}$ of the positive-valued voxels $\mathcal{V}_{s-1}^+$ as follows:

$$\begin{aligned}\mathcal{L}^{\mathcal{C}} = \frac{1}{m+d}\sum_{s=1}^{m+d}\Big(&\frac{1}{|C_s^{xy}|}\sum_{\substack{c_s^{xy}\\ \in C_s^{xy}}}\min_{v\in V_{s-1}^+}||c_s^{xy} - v_{xy}^{\mathcal{R}ot}||^2 \\ &+\frac{1}{|\mathcal{V}_{s-1}^+|}\sum_{v\in V_{s-1}^+}\min_{\substack{c_s^{xy}\\ \in C_s^{xy}}}||v_{xy}^{\mathcal{R}ot} - c_s^{xy}||^2\Big)\end{aligned}, \quad (13)$$

where $|C_s^{xy}|$ denotes the number of points inside C_s^{xy}.

The total loss $\mathcal{L}^{\mathrm{T}}$ is a summation of the defined loss functions $\mathcal{L}^{\mathcal{M}}$, $\mathcal{L}^{\mathcal{D}}$, $\mathcal{L}^{\mathcal{S}}$, and $\mathcal{L}^{\mathcal{C}}$, expressed as follows:

$$\mathcal{L}^{\mathrm{T}} = \mathcal{L}^{\mathcal{M}} + \mathcal{L}^{\mathcal{D}} + \mathcal{L}^{\mathcal{S}} + \mathcal{L}^{\mathcal{C}}. \quad (14)$$

3.4. Implementation details

We utilize the same encoder network of CapriNet [37] as $\mathcal{E}$ and a two-layer LSTM with a hidden size of 256 as $\mathcal{A}$. Furthermore, the decoders $\theta^{\mathcal{M}}$, $\mathcal{R}^{\mathcal{M}}$, $\theta^{\mathcal{D}}$, $\mathcal{R}^{\mathcal{D}}$, and $\mathcal{C}^{\mathcal{D}}$ are structured with a fully connected (FC) of size 256 layer followed by ReLU activation, five consecutive ResNet blocks [11] of 256, and another FC layer. To ensure that the angle values fall within the range of $[-\pi, \pi]$, we employ $\tanh$ on the output of the decoders $\theta^{\mathcal{M}}$ and $\theta^{\mathcal{D}}$ and multiply the results by π. Since there is no continuous size of the tools, we consider a predefined radius range $r \in radius = \{0.025, 0.05, 0.075, 0.1\}$ for milling tools and $r \in radius = \{0.01, 0.02, 0.03, 0.04\}$ for drilling tools. To allow the decoders $\mathcal{R}^{\mathcal{M}}$ and $\mathcal{R}^{\mathcal{D}}$ to differentially select among the tool $radius$, we employ softmax on the output of the decoders and multiply them by the vector $radius$. On the other hand, the path decoder $\mathcal{P}^{\mathcal{M}}$ has two branches, one to generate the depth of tool penetration c_z using features obtained from $\mathcal{A}$, while the second branch, by the same features, receives the time step $t \in \{0, 0.01, 0.02, \ldots, 0.99\}$ as input to generate (c_x, c_y). Both branches have network architectures similar to those of the other decoders. In all experiments, we set the resolution of voxels $\mathcal{V}_s$ as $64 \times 64 \times 64$, pick a large scaling factor $w = 1000$. We continue milling and drilling, each up to a maximum of 20 steps, until the difference of the loss $\mathcal{L}^{\mathcal{M}}$ and $\mathcal{L}^{\mathcal{D}}$ between steps s and $s-1$ exceeds the threshold of $1e-4$, respectively. All experiments are carried out with PyTorch 1.12.0 and Quadro RTX 8000 GPUs.

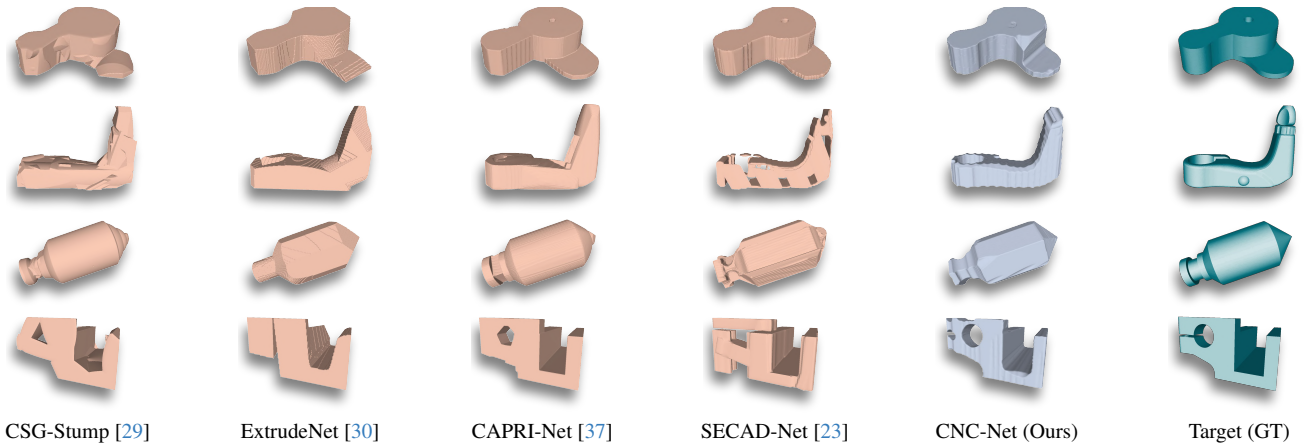


Figure 4. **Qualitative results on ABC [16] dataset.** All reproduced shapes are visualized using marching cubes (MC) with 256 resolution.

4. Experiments

This section provides comprehensive information on training and evaluation datasets, training configurations, and a deeper analysis of our proposed method, CNC-Net.

4.1. Dataset

ABC. The ABC [16] dataset comprises one million 3D Computer-Aided Design (CAD) models, particularly in manufacturing CAD objects, which serves as a valuable resource for developing geometric deep-learning methods and applications. To pre-train our method, we follow CAPRI-Net [37], sampling $5,000$ normalized single-part CAD objects. Given the time-consuming nature of fine-tuning for each shape, we randomly sample 50 shapes from $1,000$ test samples for fine-tuning and evaluation.

ShapeNet. Additionally, we utilize a broader spectrum of objects sourced from the ShapeNet Core (V1) [3] dataset. In our training and evaluation, we use the watertight shapes derived from ONet [24]. Following CAPRI-Net [37], we subsample $35k$ shapes across 13 categories for pre-training, and further, we randomly select 10 shapes from the test set of each category for fine-tuning and evaluation.

4.2. Evaluation metrics

Quantitative evaluations encompass widely used metrics, including volume-based metrics Intersection over Union (IoU) [38] and F1 [30], and surface-based metrics symmetric Chamfer Distance (CD) [25] and Normal Consistency (NC) [4]. To measure IoU and F1 for all methods, we voxelize the box $[-0.5, 0.5]^3 \subset \mathbb{R}^3$ into 256^3 voxels and evaluate their occupancies from the reconstructed meshes. For CD and NC measurements, following CAPRI-Net [37], we uniformly sample $8k$ points on the surface of each object, where all CD values are multiplied by $1,000$.

4.3. Training and evaluation

While pre-training provides foundational knowledge and learning generic features, fine-tuning adapts the model to specific samples, enhancing its effectiveness. Initially, we pre-train our model with the self-supervised objectives defined in Sec. 3.3. This pre-training phase involves 100 epochs on training samples from ABC [16] and ShapeNet [3] datasets, taking 0.5 and 1.5 hours per epoch, respectively. Later, given the fully self-supervised nature of our method, proceed to fine-tune the pre-trained model, following prior approaches [23, 29, 30, 37] for $12,000$ iterations on each test sample individually, which takes around 30 minutes per sample. We utilize ADAM [15] optimizer with a learning rate of 1×10^{-4} for both pre-training and fine-tuning experiments on both ABC [16] and ShapeNet [3] datasets.

We conduct various quantitative and qualitative experiments on both ABC [16] and ShapeNet [3] datasets to compare our reconstruction performance in contrast to state-of-the-art (SOTA) 3D CAD reconstruction methods including CSG-Stump [29], ExtrudeNet [30], CAPRI-Net [37], and SECAD-Net [23]. For fair comparisons, we use the existing pre-trained models of CAPRI-Net [37] and SECAD-Net [23], while we pre-train CSG-Stump [29] and ExtrudeNet [30] using their official implementations. Furthermore, the fine-tuning process for all methods involves the same number of iterations.

4.3.1 Results on ABC dataset

The results shown in Tab. 1 on ABC [16] dataset illustrate the superior performance of our self-supervised method in accurately reproducing target 3D CAD models by simulating CNC machining operations. In particular, compared to other 3D CAD reconstruction techniques, our CNC-Net method can improve the performance of CSG-Stump [29], ExtrudeNet [30], CAPRI-Net [37], and SECAD-Net [23]

by 4.7%, 7.2%, 7.3%, and 6.2% of IoU metric, respectively. We observe an inferior performance of our method compared to other methods when considering surface-based metrics CD and NC. This discrepancy arises because prior methods generate shapes by combining several smooth primitives, while our approach carves a cube to approximate the shape. Consequently, our approach struggles to achieve the same level of smoothness in the reproduced shapes.

Method	IoU↑	F1↑	CD↓	NC↑
CSG-Stump [29]	0.787	0.879	0.428	0.884
ExtrudeNet [30]	0.769	0.875	0.505	0.871
CAPRI-Net [37]	0.768	0.866	**0.312**	**0.914**
SECAD-Net [23]	0.776	0.867	0.398	0.900
CNC-Net (Ours)	**0.824**	**0.901**	0.509	0.893

Table 1. **Quantitative results on ABC [16] dataset.**

Moreover, we provide a visual comparison of our results with those of CSG-Stump [29], ExtrudeNet [30], CAPRI-Net [37], and SECAD-Net [23] on ABC dataset [16] depicted in Fig. 4. For equitable comparison, all reconstructed CAD models are visualized using marching cubes (MC) at a resolution of 256. Qualitative comparisons highlight the superiority of our CNC-Net in faithfully reproducing the overall shape of the target CAD models while exhibiting exceptional precision in preserving more local details, setting it apart from other methods. Specifically, taking advantage of designed carving and rotation operations, our method adeptly generates holes via drilling, a capability not achieved by CSG-Stump [29] and ExtrudeNet [30]. Furthermore, CNC-Net has the advantage of preserving parts that CAPRI-Net [37] and SECAD-Net [23] might damage. This aspect is beneficial as our reproduced shapes can be further refined through post-processing.

Therefore, the precise reproduction from 3D CAD models achieved by our CNC-Net method, as evident in quantitative and qualitative experiments, emphasizes its practical applicability in shaping desired objects from raw materials by learning CNC machine operations. We provide more qualitative results in our supplementary material.

4.3.2 Results on ShapeNet dataset

The quantitative and qualitative comparisons of our method over SOTA 3D CAD reconstruction methods [23, 29, 30, 37] on the ShapeNet [3] dataset are shown in Tab. 2 and Fig. 5, respectively. Although the objects in this dataset are generally unions of object parts, *e.g.*, chairs composed of legs, back, seat bottomland, etc., and cannot be easily processed using generic CNC machines, our method still demonstrates superior performance in terms of IoU and F1 metrics. As our method carves the shapes, it performs inferior compared to methods that construct shapes through unions in terms of surface-based metrics CD and NC. We provide more visual results in our supplementary material.

Method	IoU↑	F1↓	CD↓	NC↑
CSG-Stump [29]	0.697	0.827	0.521	0.866
ExtrudeNet [30]	0.607	0.773	0.918	0.844
CAPRI-Net [37]	0.700	0.824	**0.447**	**0.895**
SECAD-Net [23]	0.650	0.784	2.405	0.852
CNC-Net (Ours)	**0.740**	**0.850**	1.562	0.863

Table 2. **Quantitative results on ShapeNet [3] dataset.**

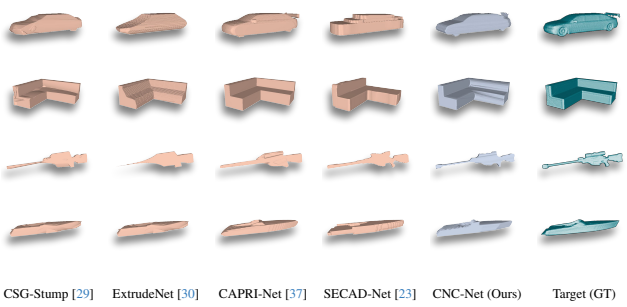


Figure 5. **Qualitative results on ShapeNet [3] dataset.** All reproduced shapes are visualized using marching cubes (MC) with 256 resolution. The 1st to 4th rows visualize the results of sampled shapes from the car, sofa, rifle, and vessel categories, respectively.

4.4. Ablation study

We further comprehensively analyze our proposed CNC-Net through a series of ablation studies. In these studies, we illustrate the learned milling paths and the zero-shot learning capability, explore the effect of various loss functions, and investigate the impact of each operation.

4.4.1 Milling paths

We visualize the learned path $\mathcal{P}_s$ for steps $s = 1, \ldots, 4$ and the generated shape $\mathcal{S}_s$ after applying the milling operation in Fig. 6. The results indicate that the decoder $\mathcal{P}^{\mathcal{M}}$ generates a path that outlines the boundary of the target object to form its overall shape in the first step and shorter paths in later steps to reach the fine details.

4.4.2 Zero-shot learning on a single shape

Based on the fully self-supervised nature of our method, we show the advantage of our CNC-Net to train in a zero-shot manner for each sample individually, eliminating the need for a large-scale training dataset. Consequently, we train the model individually for each 50 and 130 test sample from the ABC [16] and ShapeNet [3] datasets, respectively. Comparisons between zero-shot and fine-tuned results indicate that

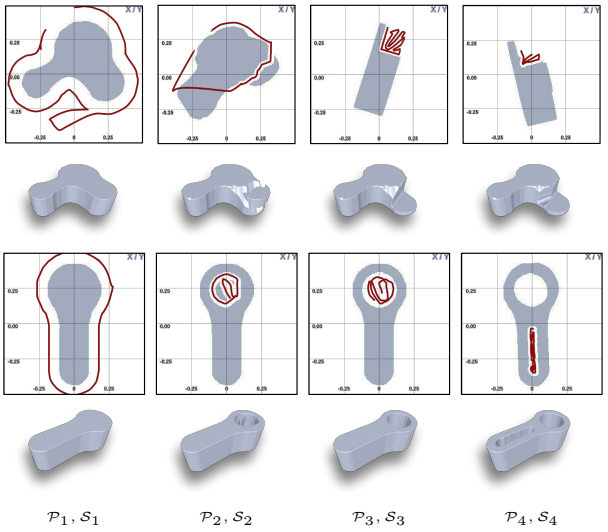


Figure 6. **Ablation study for milling paths.** The 1st and 3rd rows display the path $\mathcal{P}$ in steps $s = 1, \ldots, 4$ from the top view. The 2nd and 4th rows depict the reproduced shapes in each step.

our method can reproduce target shapes without significant degradation performance, as shown in Tab. 3. In subsequent ablation studies, we show the results for the models trained in a zero-shot manner.

Training	ABC [16]				ShapeNet [3]			
	IoU↑	F1↑	CD↓	NC↑	IoU↑	F1↑	CD↓	NC↑
Fine-tuning	**0.824**	**0.901**	**0.509**	**0.893**	**0.740**	**0.850**	**1.562**	**0.863**
Zero-shot	0.780	0.878	1.127	0.870	0.698	0.812	2.155	0.845

Table 3. **Ablation study for zero-shot training.**

4.4.3 Effect of losses

We evaluate the effect of our defined loss functions in Sec. 3.3 on the test samples of ABC [16] dataset detailed in Tab. 4. The results demonstrate that our method can be effectively trained using only the essential loss functions $\mathcal{L}^{\mathcal{M}}$ and $\mathcal{L}^{\mathcal{D}}$, while the collaboration of $\mathcal{L}^{\mathcal{S}}$ and $\mathcal{L}^{\mathcal{C}}$ serves as a guiding factor to improve the reconstruction performance.

$\mathcal{L}^{\mathcal{M}}$	$\mathcal{L}^{\mathcal{D}}$	$\mathcal{L}^{\mathcal{S}}$	$\mathcal{L}^{\mathcal{C}}$	IoU↑	F1↑	CD↓	NC↑
✓	✓	✗	✗	0.721	0.845	5.202	0.837
✓	✓	✓	✗	0.686	0.820	5.138	0.831
✓	✓	✗	✓	0.755	0.847	7.184	0.859
✓	✓	✓	✓	**0.780**	**0.878**	**1.127**	**0.870**

Table 4. **Ablation study for loss functions.**

4.4.4 Effect of operations

We further explore the individual impact of milling $\mathcal{O}^{\mathcal{M}}$, drilling $\mathcal{O}^{\mathcal{D}}$, and rotation $\mathcal{R}ot$ operations by excluding each one during both the training and testing phases. The results in Tab. 5 on the ABC dataset [16] underscore the significant role of milling and rotation operations in shaping target objects. In contrast, drilling slightly contributes to more precise shape reconstruction. We observe that removing the drilling operation results in almost twice the value for the CD metric. The reason is that to generate a hole in the shape, the milling operation has to function as a drilling process, resulting in a non-smooth shape.

$\mathcal{O}^{\mathcal{M}}$	$\mathcal{O}^{\mathcal{D}}$	$\mathcal{R}ot$	IoU↑	F1↑	CD↓	NC↑
✗	✓	✓	0.460	0.630	8.315	0.701
✓	✓	✗	0.525	0.677	10.826	0.738
✓	✗	✓	0.776	0.876	2.098	0.869
✓	✓	✓	**0.780**	**0.878**	**1.127**	**0.870**

Table 5. **Ablation study for operations.**

5. Conclusion

We propose CNC-Net, a novel self-supervised DNN-based framework designed to simulate a generic CNC machine. CNC-Net provides a sequence of learnable modeled manufacturing operations with implicit representation to construct desired objects from raw materials. Our quantitative and qualitative reconstruction results demonstrate the superior performance of our method compared to the most state-of-the-art 3D CAD reconstruction techniques.

Limitation and future work. One remaining limitation is that no dataset contains the sequential operations required to serve as a reference for our learned milling and drilling operations. This absence makes it challenging to assess the efficiency of our method and determine whether it represents an optimal solution. On the other hand, finding multi-stage execution for CNC machines is an NP-hard problem, so it is not time-efficient and requires expensive specialized human labor. In future work, we plan to enhance our method by incorporating a decision-making process to select the appropriate type of operation at each step. Although cylindrical tools are common, it is important to note that tool shapes are diverse and designed for specific machining operations and applications. Our future endeavors include exploring tools with various shapes, such as broaches, gear, and fly cutters, and incorporating a broader range of operations such as grinding, turning, and lapping. Additionally, our current focus on the input raw material as a bounding box can be expanded to encompass shapes with various geometry in future investigations.

Acknowledgement. This work was supported in part by the IITP grants [No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University), No.2021-0-02068, and No.2023-0-00156], the NRF grant [No.2021M3A9E4080782] funded by the Korean government (MSIT).

References

[1] Yusuf Altintas and AA Ber. Manufacturing automation: metal cutting mechanics, machine tool vibrations, and cnc design. *Appl. Mech. Rev.*, 2001. 1

[2] J Balic and M Korosec. Intelligent tool path generation for milling of free surfaces using neural networks. *Int. J. Mach. Tools Manuf.*, 2002. 2

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv*, 2015. 2, 6, 7, 8

[4] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *CVPR*, 2020. 6

[5] Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W Taylor, Jaesik Park, and Minsu Cho. Brick-by-brick: Combinatorial construction with deep reinforcement learning. *NeurIPS*, 2021. 2

[6] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *CVPR*, 2020. 1, 2

[7] Marc-André Dittrich, Florian Uhlich, and Berend Denkena. Self-optimizing tool path generation for 5-axis machining processes. *CIRP-JMST*, 2019. 2

[8] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Trans. Graph.*, 2018. 2

[9] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, 1996. 2

[10] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019. 2

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2015. 5

[12] Hsin-Ta Hsieh and Chih-Hsing Chu. Improving optimization of tool path planning in 5-axis flank milling using advanced pso algorithms. *Robot. Comput.-Integr. Manuf.*, 2013. 2

[13] Xiaoyang Huang, Yi Zhang, Kai Chen, Teng Li, Wenjun Zhang, and Bingbing Ni. Learning shape primitives via implicit convexity regularization. In *ICCV*, 2023. 2

[14] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. *NeurIPS*, 2020. 2

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[16] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, 2019. 2, 6, 7, 8

[17] Naofumi Komura, Kazuma Matsumoto, Shinji Igari, Takashi Ogawa, Sho Fujita, and Keiichi Nakamoto. Computer aided process planning for rough machining based on machine learning with certainty evaluation of inferred results. *Int. J. Autom. Technol.*, 2023. 2

[18] Aman Kukreja and Sanjay S Pande. Optimal toolpath planning strategy prediction using machine learning technique. *Eng. Appl. Artif. Intell.*, 2023. 2

[19] David H Laidlaw, W Benjamin Trumbore, and John F Hughes. Constructive solid geometry for polyhedral objects. In *SIGGRAPH*, 1986. 1, 2

[20] SP Leo Kumar, J Jerald, and Somasundaram Kumanan. Feature-based modeling and process parameters selection in a capp system for prismatic micro parts. *Int. J. Comput. Integr. Manuf.*, 2015. 2

[21] Bingran Li, Hui Zhang, Peiqing Ye, and Jinsong Wang. Trajectory smoothing method using reinforcement learning for computer numerical control machine tools. *Robot. Comput.-Integr. Manuf.*, 2020. 2

[22] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *CVPR*, 2019. 2

[23] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *CVPR*, 2023. 2, 6, 7

[24] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 6

[25] Himangi Mittal, Brian Okorn, Arpit Jangid, and David Held. Self-supervised point cloud completion via inpainting. *BMVC*, 2021. 6

[26] KK Natarajan and J Gokulachandran. Application of artificial neural network techniques in computer aided process planning-a review. *IJPMB*, 2021. 2

[27] Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single rgb image. In *CVPR*, 2018. 2

[28] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *CVPR*, 2021. 2

[29] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *ICCV*, 2021. 2, 6, 7

[30] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In *ECCV*, 2022. 2, 6, 7

[31] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *CVPR*, 2018. 1, 2

[32] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *ECCV*, 2020. 2

[33] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017. 2

[34] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017. 1

[35] Tim von Hahn and Chris K Mechefske. Machine learning in cnc machining: Best practices. *Machines*, 2022. 2

[36] Mohsen Yavartanoo, Jaeyoung Chung, Reyhaneh Neshatavar, and Kyoung Mu Lee. 3dias: 3d shape reconstruction with implicit algebraic surfaces. In *ICCV*, 2021. 1, 2

[37] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: learning compact cad shapes with adaptive primitive assembly. In *CVPR*, 2022. 1, 2, 5, 6, 7

[38] Jiaqian Yu, Jingtao Xu, Yiwei Chen, Weiming Li, Qiang Wang, Byungin Yoo, and Jae-Joon Han. Learning generalized intersection over union for dense pixelwise prediction. In *ICML*, 2021. 6

[39] O Zavalnyi, G Zhao, Y Liu, and W Xiao. Optimization of the step-nc compliant online toolpath generation for t-spline surfaces using convolutional neural network and random forest classifier. In *IOP Conf. Ser.: Mater. Sci. Eng.*, 2019. 2

[40] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *ICCV*, 2017. 2