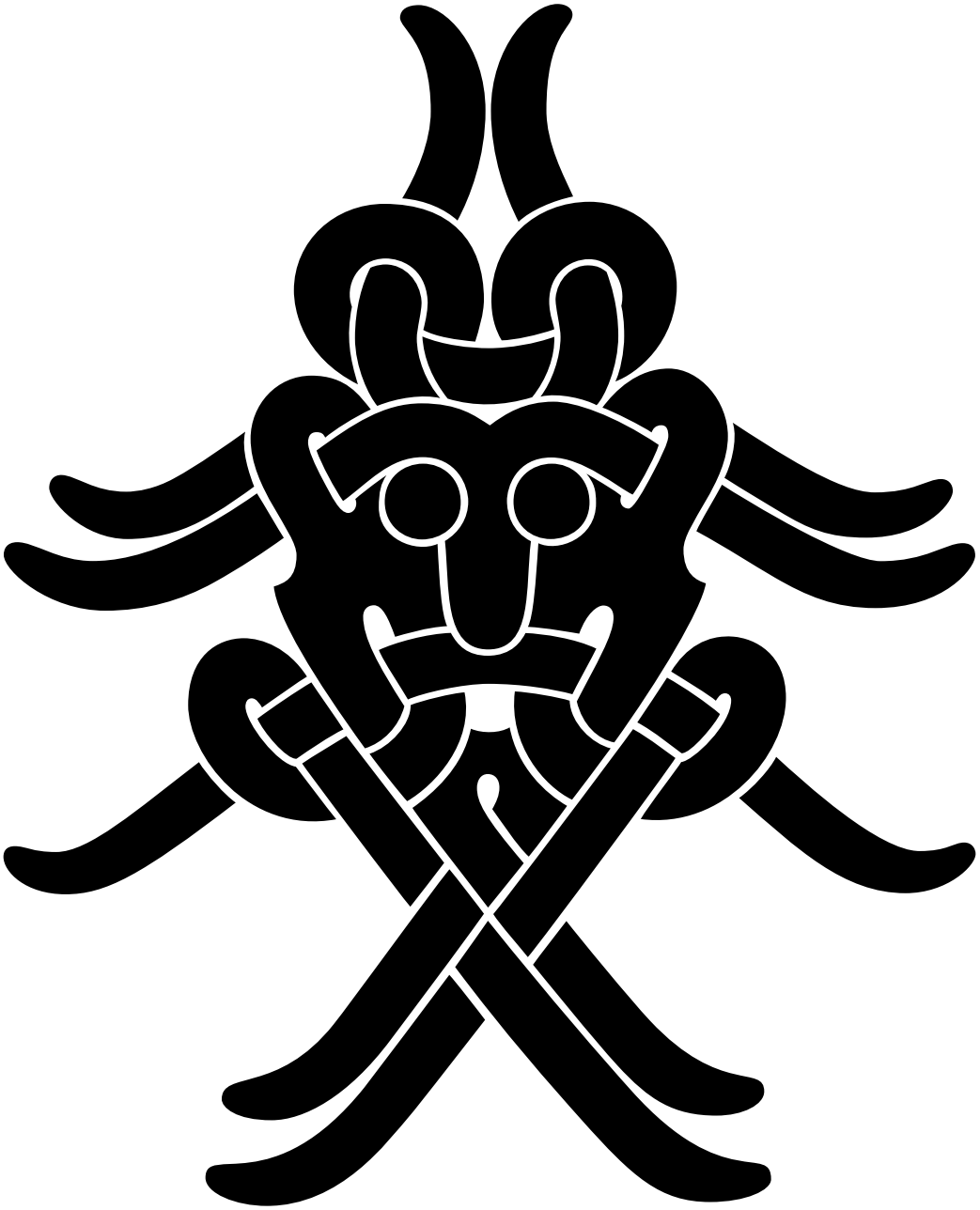
COAST RUNNER TOOLS

Research & Development



The CRTools framework aims to solve AI-orchestrated tool path creation for subtractive manufacturing use cases for 3-axis horizontal CNC machines like the Coast Runner. This technical report covers all important options for implementing the CRTools framework and covers all steps and tradeoffs for the project's deployment.

First, we review relevant work and research. We cover state of the art publications for CAM manufacturing, as well as the general research from the Fusion 360 team, which centers around CAD model generation. B-Rep graphs and set-based neural approaches like graph neural networks and transformers are found to be the cornerstone of the most of the SoA approaches. The Fusion 360 Gallery dataset and corresponding gym environment for reinforcement learning are important hints for the direction this project needs to take. In the third section we provide a mathematical setup of our problems, as well as proposed algorithms and tradeoffs involved in choosing or avoiding to use them. The fourth section is a report on the current state of the framework, and reasoning for the next steps before data collection. The fifth section analyzes hardware and software requirements for pushing this project to production, and the sixth chapter gives the overview on how to publish to the Fusion 360 Store.

Relevant work

CloudNC is UK-based startup with over 120 employees specializing in AI CAM solutions. They promise automation of 80% of CAM processing via their "CAM Assist" package. They support Fusion, Mastercam and SiemensNX platforms, and 3 or 3+2 axis CNC machines. No papers published by this company and its employees were found, neither by general paper search, nor by searching AI engineers and data scientists found on LinkedIn.

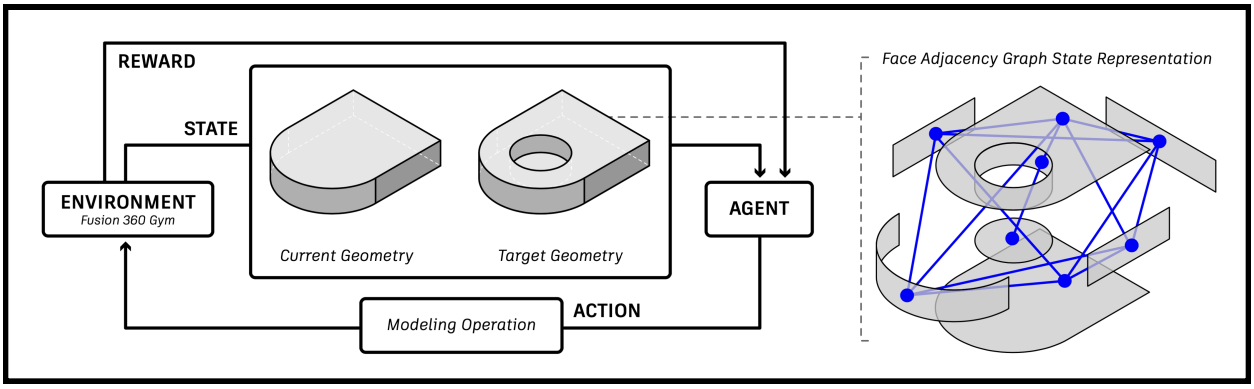
A recent review of tool path optimization techniques (Noor et al. 2020) for CNC machines highlights the following algorithms used for tool path planning and machining parameter optimization: artificial intelligence (AI) or hybrid methods, for example, genetic algorithms (GA), ant colony optimization (ACO), artificial neural networks (ANN), and particle swarm optimization (PSO). Schäfer et al. (2021) have created a combinatory, logic-based tool path generator. Most of the existing solutions are low-level and don't use the latest software components as a part of their problem solving.

The Autodesk Research team has built an open source machine learning framework and four public datasets for CAD modeling in Fusion 360 published in the repository under a non-commercial research license.

In three main papers behind this work, Willis et al. (2021, 2022), and Lambourne et al. (2021) have used graph neural networks (GNN) to exploit the topological structure of BRep CAD models for automated CAD model generation, CAD model segmentation, and assembly of CAD-modeled joints.

Willis et al. (2021) has described CAD model generation as a sequential task. They have collected 8,625 human generated sequences of operations to create a training dataset. Data generation was limited to the “extrude” and “sketch” operations to reduce the problem search space, noticing that the most designs contain at least one of these operations, and that most shapes could be approximated with only these two operations.

The authors also created a training environment called Fusion 360 gym which is used by reinforcement learning frameworks for orchestrating communication between the environment and agent in training and inference time. An agent was trained to make the decision about the location and the type of operation based on comparison of the current and target graph representation of a CAD model:



Since the model was trained from a pre-built dataset, imitation learning was used to model the loss. They have also explored how different search strategies (random sampling, greedy, beam search) affect the final performance of trained models. As a baseline for evaluation, they used uniformly sampled random decision, and surprisingly it did not perform poorly. They note:

> "The success of the rand agent demonstrates that short construction sequences can be solved by a naive approach. This is due to several factors:
>
> 1) our action representation that uses B-Rep faces
>
> 2) oursearch procedure discarding invalid actions
>
> 3) designs in the dataset with a low number of planar faces andextrude steps."

This is promising because it shows that “smart heuristics” can perform well before any true learning enters the picture.

Aside from the Fusion 360 Gallery dataset, recent progress in CAD generation has been made with the recent DeepCAD (We et al., 2021) and AutoMate (Jones et al., 2021) publications, both being B-Rep based. Multiple machining datasets exist, like Bosch's (Tnani et al., 2022), but not many datasets exist for tool path generation. To our knowledge, the only set similar to these was created by Feng et al. (2024) for their work in neural network-based parametrizing B Spline tool paths for 3-axis CNC machines, but it is not public and does not contain target tool paths to learn.

More modern approaches are taken by the Autodesk research team since open sourcing the Gallery dataset. Xu at al. (2022) are using transformer architecture to generate CAD models. Xu et al. (2023) combine a vector quantized variational autoencoder approach and hierarchical representation of CAD models for CAD model generation, with the main limitation being proneness to physically infeasible solutions.

Xu et al. (2024) use transformer-based diffusion models (work based on structured latent geometry representation) that transform any B-rep into a tree data structure. Tian et al. (2024) plan robotic assembly based on geometric heuristics or graph neural networks (GNNs) trained on a simulation-based dataset of 1,906 assemblies.

Problem setting

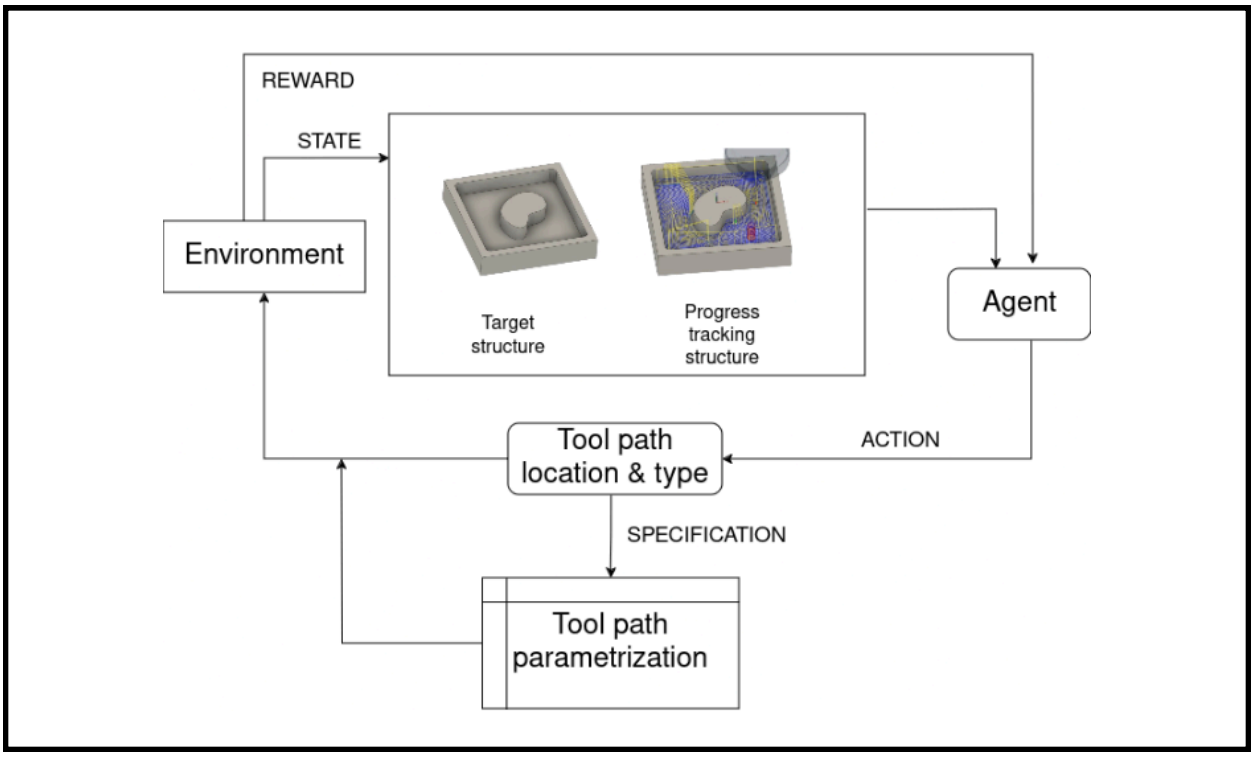
As our work setting is Fusion 360, we will maximize the usage of the Fusion API, namely the manufacturing API and its tool path generation capabilities. Like Willis et al. (2021) have done with CAD model generations, subtractive manufacturing can also be represented as a sequence of decisions which constitute a Markov Decision Process (MDP). While CAD creation tasks can start from an empty graph and add surfaces and volumes comparing it to a target, CAM cannot reduce a search space to just two operations. CAM also has inherent complications such as parallelization of chosen operations, and tool switching.

For this reason, our solution will have two necessary steps:
a. Global choice of the Fusion 360 CAM operation and its location on the B-Rep graph
b. Parametrizing the chosen operation.

For this purpose, two sets of models will be developed:
1. A global model which chooses the next operation based on-
2. A set of local models which learn to parametrize operations: one model for each operation.

While one Markov decision process step consists of both steps a) and b), one can decouple their learning and simplify the MDP as a sequence of global step 1) decisions. In this setting, local parameterization models are learned separately as a combination of classification and regression tasks. Parameterization policy is fixed in the MDP and treated as an environmental dynamic.

Machine Learning



The problem formulation above puts us in the situation where step b) is first learned from the data via classical regression and classification algorithms, and step a) is then optimized.

Data representation

Literature review and Fusion 360 API capabilities suggest that the best default option for storing data is graph representation of B-Rep objects extended with generated tool-paths. This representation allows use of graph neural networks, but also local and global summarization for more old-fashioned machine learning techniques.

Our approach will build upon Willis et al. (2021), who have built a graph from B-Rep adjacency structures where faces of the graph were cast into graph nodes and edges were cast into graph edges. Only nodes had features, regarding three 10x10 grids: a grid of sampled 3D points, a grid of sampled 3D normal vectors, and a trimming mask: a 10Å~10 grid of binary values representing samples that are inside/outside the B-Rep face trimming boundary. The Surface types - Cone, Cylinder, Elliptical, Elliptical Cylinder, Nurbs, Plane, Sphere, Torus – were also stored as one-hot encoded vector. Besides CAD models, we also need to represent generated tool paths and their adjacencies.

Open question: Since this simulation is not yet supported by the Fusion API, we should come to understand how much structure can be deduced automatically and what needs to be calculated.

It is essential that tool paths are included into a graph representation as different kinds of nodes. From the current literature, features can be the surface type, which is being processed, operation type, as well as parameters for the operation. Data analysis and experimentation will determine if different operations need to have a different node structure, as well as which parameters are truly important to keep as features.

To replicate Willis' approach, we also need to track two graphs in parallel: One for target structure, and one for the workpiece and tool-paths, later being constructed to mirror Willis' approach. Their graph was built from the bottom up adding new vertices, while ours will be building the negative from the work piece downwards.

Local step: parametrizing the operation

Using the Fusion 360 API makes this step very simple. The requirement for this step is enough data for each supported operation and parameter. The input for this step is all the chosen face local geometrical data stored as a node feature, but it can be enhanced with the geometric data calculated from the API. E.g. Minimal and maximal distance from the target model's face to the surface (min, max). For this step, old school algorithms like linear regression (for a particular parameter) and logistic regression (e.g. for tool choice) should work very well. Simple neural networks should work even better. The biggest advantage of a neural network approach is that parts of the network are shared between all operations (e.g. tool path classifier), and the trunk can be shared with different outputs for the same operation type.

Since CNC machining is a precise undertaking, replacing regression with classification either totally or as in Farebrother et al. (2024) is a reasonable decision to make, and could stabilize learning and improve performance.

Global step: order and type of operations

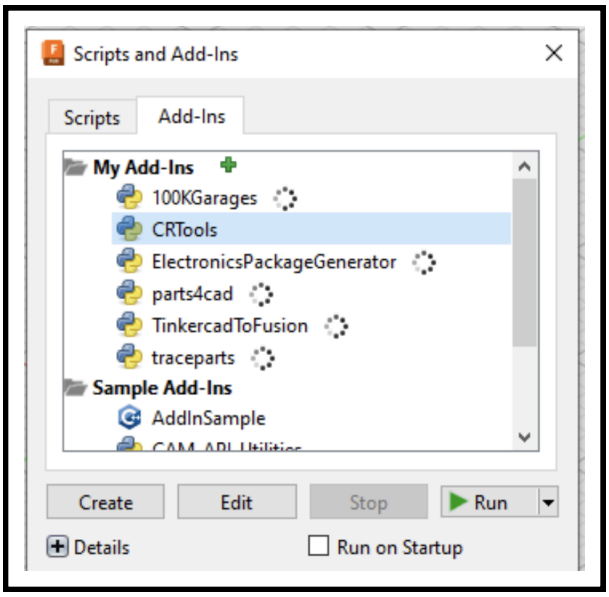
Here there are two significant approaches: Heuristic and graph neural network classification. Heuristic is the simpler approach, and it can be crafted from a modest amount of data together with some domain knowledge and common-sense reasoning. Facing, adaptive, and finishing operations seem to be the common order of operation types.

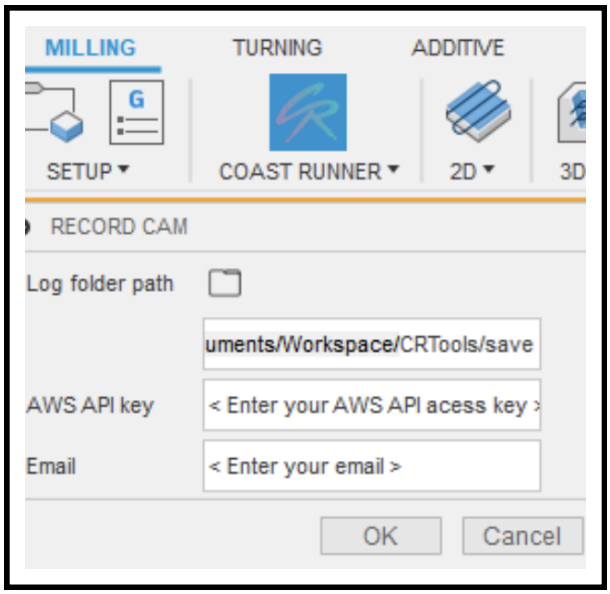
To expand:

- Hand-crafted (greedy) heuristics
- Graph neural network-based classification
  - Reason why hand-crafted might be good solution
- Data size compared to simpler Gallery problem.
- Expanding graph with holes and pockets recognized by the Fusion API

Framework

The current state of the CRTools framework is the automated recording of unstructured manufacturing process data. It already supports automatic installation of third party Python libraries, which is essential for executing machine learning workloads.





In the screenshot above, "AWS API key" and "Email" are placeholders for credentials when data collection scales up, and especially approaches full-scale cloud and back-end infrastructure.

The next step is making an equivalent of the environment from the Fusion 360 Gallery gym, but for CAM instead of CAD use cases. Since gym is not commercially licensed, even the CAM model needs to be redone. In this step it is important to achieve two-way transformation: From CAM model and generated paths to B-Rep based extended graphs and back again. Without this, it is possible that some bugs will be introduced and some data not properly or completely recorded. This can make the early data collecting step futile for serious processing, although early data collection might play fine with the dirty heuristic approach.

Hardware and software requirements

Fusion 360 provides three ways to use its API: Python, JavaScript and C++. For this project Python and C++ are the most obvious options, with a tradeoff between those two languages being in development speed on the Python support, and execution speed and code base privacy on the C++ side. Our current solution is based on Python. Since Fusion has its own Python interpreter, the challenge was installing external dependencies like machine learning libraries for the inference, which the current draft has solved and automated.

Fusion is proprietary software, so for running any algorithm which includes calculations with geometry either everything needs to be re-implemented manually, or one needs to have Fusion installed with the appropriate license to use its Python library. Since Fusion is tied to either Windows or MacOS, at least this part of the tool chain needs to be run on a Windows Desktop environment. The rest can be run in the cloud.

Currently, add on logs are stored locally. Cloud infrastructure should be built where data will be automatically uploaded, and from there used for training. While, depending on the final decision on the algorithm, a part of the training workflow will require a Windows environment for calculations in Fusion, training alone can be done in the cloud and communicate with the Windows machine via HTTP protocol.

Publishing add-in

Advanced steps

While the current work reflects SoA techniques, all parts of the process can and should be refined iteratively. Approaches like decision transformers (Chen et al., 2021) and other transformer, diffusion models or QV-VAE based hierarchical architectures (Xu at al., 2022, 2023, 2024) should be considered long term goals.

Standards and Technical Literature

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. “Decision Transformer: Reinforcement Learning via Sequence Modeling.”

Farebrother, Jesse, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. 2024. “Stop Regressing: Training Value Functions via Classification for Scalable Deep RL.”

Feng, Yi-fei, Hong-Yu Ma, Li-Yong Shen, Chun-Ming Yuan, and Xin Jiang. 2024. “Real-Time Tool-Path Planning Using Deep Learning for Subtractive Manufacturing.” IEEE Transactions on Industrial Informatics 20(4):5979–88. doi: 10.1109/TII.2023.3342474.

Jones, Benjamin, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G. Kim, and Adriana Schulz. 2021. “AutoMate: A Dataset and Learning Approach for Automatic Mating of CAD Assemblies.” ACM Trans. Graph. 40(6):227:1-227:18. doi: 10.1145/3478513.3480562.

Lambourne, Joseph G., Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. “BRepNet: A Topological Message Passing System for Solid Models.”

Noor, Hatem, Yusri Yusof, Kadir AiniZuhraA, and Mohammed M.A. n.d. “A Review of Tool Path Optimization in CNC Machines: Methods and Its Applications Based on Artificial Intelligence.” Retrieved November 18, 2024 (https://www.researchgate.netpublication/342470169_A_Review_of_Tool_Path_Optimization_in_CNC_Machines_Methods_and_Its_Applications_Based_on_Artificial_Intelligence).

Schäfer, Tristan, Jim A. Bergmann, Rafael Garcia Carballo, Jakob Rehof, and Petra Wiederkehr. 2021. “A Synthesis-Based Tool Path Planning Approach for Machining Operations.” Procedia CIRP 104:918–23. doi: 10.1016/j.procir.2021.11.154.

Tian, Yunsheng, Karl D. D. Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, Hui Li, Sachin Chitta, and Wojciech Matusik. 2024. “ASAP: Automated Sequence Planning for Complex Robotic Assembly with Physical Feasibility.”

Tnani, Mohamed-Ali, Michael Feil, and Klaus Diepold. 2022. “Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring.” Procedia CIRP 107:131–36. doi: 10.1016/j.procir.2022.04.022.

Willis, Karl D. D., Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2022. “JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints.”

Standards and Technical Literature

Willis, Karl D. D., Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. “Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences.”

Wu, Rundi, Chang Xiao, and Changxi Zheng. 2021. “DeepCAD: A Deep Generative Network for Computer-Aided Design Models.”

Xu, Xiang, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Karl D. D. Willis, and Yasutaka Furukawa. 2023. “Hierarchical Neural Coding for Controllable CAD Model Generation.”

Xu, Xiang, Joseph G. Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl D. D. Willis, and Yasutaka Furukawa. 2024. “BrepGen: A B-Rep Generative Diffusion Model with Structured Latent Geometry.”

Xu, Xiang, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. 2022. “SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled
Codebooks.”