RESEARCH-ARTICLE

Learn to Create Simple LEGO Micro Buildings

JIAHAO GE, Chinese University of Hong Kong, Hong Kong, Hong Kong

MINGJUN ZHOU, Chinese University of Hong Kong, Hong Kong, Hong Kong

CHI WING FU, Chinese University of Hong Kong, Hong Kong, Hong Kong

Learn to Create Simple LEGO® Micro Buildings

JIAHAO GE*, MINGJUN ZHOU*, and CHI-WING FU, The Chinese University of Hong Kong, Hong Kong

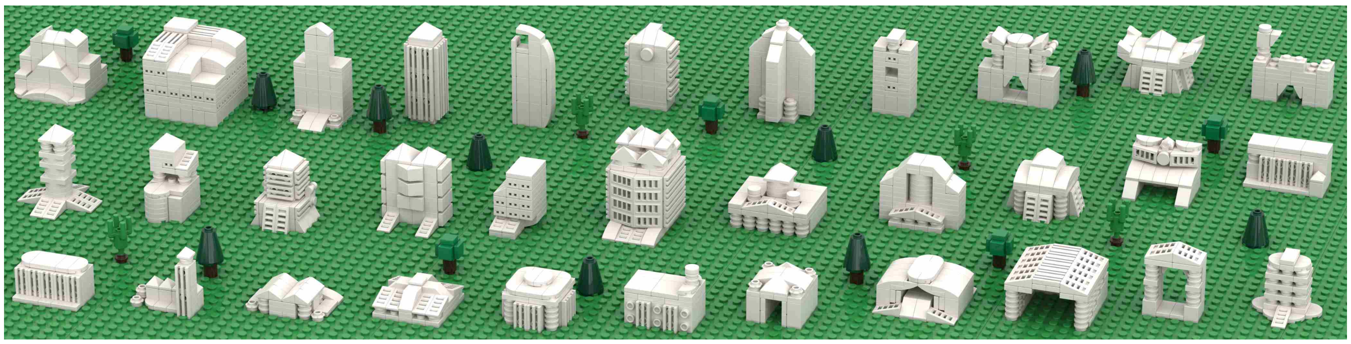


Fig. 1. LEGO® micro buildings created by our approach exhibit a rich variety of shapes and styles, ranging from modern city buildings to iconic landmarks, to simple houses, and to miniature castles, in compact but intricate forms. The LEGO® trees are manually added for decoration.

This paper presents the first learning-based generative pipeline for effectively creating 3D LEGO® [1] models. This task is very challenging due to the lack of dedicated representations and datasets for learning coherently-connected bricks arrangements, as well as an immense design space that is combinatorial in nature. We approach this task by focusing on creating LEGO® micro buildings. Our contributions are four-fold. First, we propose the LEGO® semantic volume representation to encode LEGO® models, considering the bricks types and bricks connections, while allowing back-propagation learning. Second, we further consider the transformative nature of LEGO® to atomize the semantic volume and formulate a generative model to learn the representation. Third, we build a rich dataset of micro buildings for model learning. Last, we design the progressive reconstructor to create 3D LEGO® models from the generated representations, while ensuring bricks connections. We employed our pipeline to create LEGO® micro buildings with a wide array of bricks types, demonstrating its strong capability of learning diverse micro-building styles and producing assemble-able LEGO® models. Further, we performed various quantitative evaluations, ablations, and a user study to show the compelling capability of our approach in terms of generative quality, fidelity, and diversity.

CCS Concepts: • **Applied computing → Computer-aided manufacturing**.

Additional Key Words and Phrases: LEGO®, machine learning, 3D generation, assembly

*indicates joint first authors.
[1]LEGO® is a trademark of the LEGO® Group, which does not sponsor, authorize or endorse this work. All information in this paper is collected and interpreted by its authors and does not represent the opinion of the LEGO® Group.

Authors' Contact Information: Jiahao Ge, jiahaoge@link.cuhk.edu.hk; Mingjun Zhou, mingjunzhou@link.cuhk.edu.hk; Chi-Wing Fu, cwfu@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong.

ACM Reference Format:
Jiahao Ge, Mingjun Zhou, and Chi-Wing Fu. 2024. Learn to Create Simple LEGO® Micro Buildings. *ACM Trans. Graph.* 43, 6, Article 249 (December 2024), 13 pages. https://doi.org/10.1145/3687755

1 Introduction

The word "Lego" comes from the Danish words "leg godt," which mean "play well." Since 1958, LEGO® bricks have become popular toys, enabling children and adults to build almost anything, so long as one can imagine. Especially, children can "learn through play," as this building process promotes creativity, imagination, spatial awareness, and beyond. Indeed, LEGO® bricks offer endless building possibilities, thanks to the simple but effective brick interlocking system with a rich variety of specifically-designed brick pieces.

Motivated by this astonishing possibility, in 1998, the LEGO® Group presented the following question to the academics: "Given any 3D model, how can it be built with LEGO® bricks?" [Gravesen and Hjorth 1998]. In this construction problem, only generic rectangular bricks are considered and the task is to "create a computer program to determine which brick should be put at which place." Then, after the pioneering work of [Gower et al. 1998], many researchers began to explore the answer to this question. The outcomes include various brick representations, algorithms, and meta-heuristic cost functions, such as graphs [Peysakhov et al. 2000], cellular automata [Smal 2008], force-based layout refinement [Luo et al. 2015], and so on. However, most existing methods consider only rectangular bricks perfectly inside a simple regular 3D grid, so undermining the versatile expressiveness of LEGO® bricks. Also, due to the combinatorial nature of the task, heuristic-based methods are widely used, yet they may become intractable when handling the ever-expanding variety of brick types and arrangements.

In the era of 2020s, generative models demonstrate remarkable success in creating AI-generated contents in natural languages, images, videos, and shapes. For the LEGO® construction problem, an extended question would be: "Given a family of LEGO® models, can the machine learn to generate similar models?" The question is challenging for several reasons. First, the machine has to learn

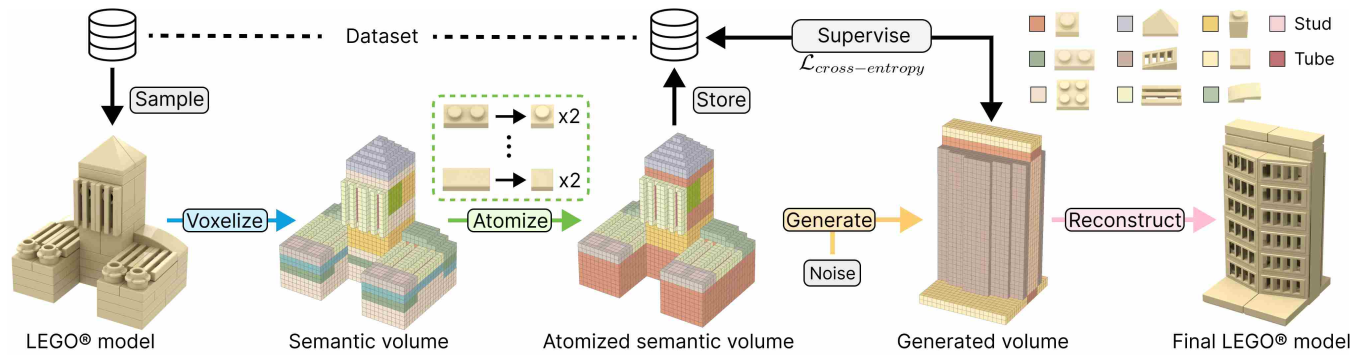


Fig. 2. Overview of our approach: (i) build a dataset of LEGO® micro buildings; (ii) voxelize each data sample into the LEGO® semantic volume that encodes both brick types and bricks connections; (iii) atomize the semantic volume representation to contain only simplex bricks; (iv) train the generative model to learn to produce the proposed representation; and (v) progressively reconstruct the final LEGO® model from the generated semantic volume.

the LEGO® system to a certain extent that it can follow the system to reproduce similar models. Yet, such a system includes a large number and also a large variety of bricks, as well as various types of bricks connection. Second, LEGO® models are not as structural as natural languages and images. They are not directly compatible with conventional neural networks, for the networks to effectively learn from. Third, following the above reason, it is thus hard to devise an effective representation of LEGO® model for machine learning, while capturing brick semantics and allowing back-propagation learning. Last, there is a lack of dedicated datasets for LEGO® models to support the machine learning.

At present, machine learning has been studied on few LEGO® -related tasks such as brick classification [Boiński et al. 2022] and assembly planning [Ma et al. 2023]. As for LEGO® model generation, we are aware of only two works: Thompson et al. [2020] use a graph to represent the model structure, whereas Lennon et al. [2021] use a voxel volume. Yet, both works consider only rectangular bricks. Thompson et al. focus on models built with 2×4 LEGO® bricks, whereas Lennon et al. convert each voxel into a 1×1 brick and then merge the small bricks into larger cuboid bricks. Hence, both methods can produce only coarse and simple voxel-like results. Besides, the two representations encode either connections or shapes while neglecting the other. However, both connections and shapes are crucial for the machine to learn the LEGO® system.

To approach the challenging task of generating LEGO® models by learning in a manageable manner, we focus our attention on unconditionally generating a specific class of models called the LEGO® micro buildings and constrain the physical size in a volume of $4.8cm \times 4.8cm \times 6.4cm$. We elaborate on the choice of size later in Section 4. Though these models are made up of limited number of bricks on small physical scales, they employ a wide variety of bricks and exhibit diverse structures and rich styles. From modern city buildings to iconic landmarks and from simple houses to miniature castles, LEGO® micro buildings showcase the essence of creating LEGO® models in compact but intricate forms. See Figure 1 for diverse micro buildings produced by our learning-based approach.

To develop our computational approach, we start by first building *a dataset of 1,000 LEGO® micro buildings*, capturing diverse building shapes and styles with different kinds of bricks. Second, to facilitate effective learning, we design a novel volumetric representation named the *LEGO® semantic volume*, which *spatially* encodes both the brick-type and brick-connection information. Particularly, our representation encodes the LEGO® studs and tubes (see Section 4) to help the neural network learn bricks connections in the LEGO® system. Third, we consider the transformative nature of building things with LEGO®, meaning that the same shape can be built by different bricks management, we propose to simplify the LEGO® semantic volume by *atomizing* (decomposing) larger bricks into simpler ones, such that the neural network can focus on learning the 3D model shapes and the connection structures. Lastly, we develop a *generative model* to learn to generate the semantic volume by formulating a discrete diffusion-based classification task, and design the *progressive reconstructor* to create coherently-connected LEGO® models from the generated semantic volume by iteratively arranging bricks while respecting the network model predictions and the bricks connections.

We demonstrate the capability of our generative pipeline by creating diverse LEGO® micro buildings of varying shapes, styles, and scales. Also, we quantitatively evaluated the quality, fidelity, and diversity of the generated micro buildings, showcasing the compelling performance of our pipeline. Besides, we conducted a user study to demonstrate that our generated micro buildings are almost indistinguishable from the human-crafted ones. Further, ablation studies and a comparison with baseline were performed to validate the effectiveness of our proposed pipeline.

Overall, this work makes the following contributions:

(i) We introduce the first generative pipeline that can effectively learn to create LEGO® micro buildings, considering a wide array of non-rectangular bricks;
(ii) We design a novel representation for LEGO® models, the LEGO® semantic volume, coupled with the atomization idea, to facilitate effective model learning;
(iii) We formulate the learning process of training the generative model as a discrete diffusion-based classification task for high-quality generation of the LEGO® semantic volumes, and develop the progressive reconstructor to create LEGO® models from the semantic volumes; and

(iv) We contribute the first LEGO® micro-building dataset with diverse shapes and design styles.

Training code and dataset can be found on the following GitHub page: https://github.com/Occulte/LEGO_Buildings_Generation.

2 Related Work

In this section, we discuss assorted related topics.

LEGO® construction. Our work aligns with research on LEGO® model construction using computer programs. The first work is Gower et al. [1998], which automatically constructs LEGO® models with regular bricks. Subsequent studies focus on constructing voxelized LEGO® models from colored 3D models. Some follow-up works include [Petrovič 2001] using an evolutionary algorithm, [Winkler 2005] using a beam search, [Testuz et al. 2013] using a graph-based algorithm, [Stephenson 2016] using a multi-phase method, and [Lee et al. 2018] using a genetic algorithm. Recently, some methods focus on constructing specific types of LEGO® models, *e.g.*, LEGO® houses with slope bricks [Zhou et al. 2019], LEGO® Technic models [Xu et al. 2021], layered LEGO® sketches [Zhou et al. 2023], and LEGO® Figurines [Ge et al. 2024].

Exploiting machine learning for LEGO® construction, Thompson et al. [2020] and Lennon et al. [2021] are early attempts to learn the bricks arrangements for generation. Thompson et al. conceptualize LEGO® structures as graphs and employ a generative graph model to learn from human-built structures. Lennon et al. employ a voxel volume and build an autoencoder to generate voxels from images. Overall, both approaches work only with basic rectangular bricks and their representations encode *either* the bricks connections *or* the 3D model shapes *but not* both. Also, without specific techniques, their results are coarse and simple voxel-like models. Very recently, Ge et al. [2024] design a new pipeline to aid the production of LEGO® figurine models. Yet, the approach makes use of machine learning mainly to match human photos and LEGO® figurine appearance and to transfer garment textures from photos to LEGO® figurines, without learning to generate 3D LEGO® structures.

Aiming at generating more expressive 3D LEGO® models, we consider a wider array of bricks types and develop a novel generative pipeline that can effectively produce diverse LEGO® micro-building models. Our method is capable of learning not only the shapes of the LEGO® models but also the bricks connections, for producing physically assemble-able LEGO® models.

Procedural modeling. Procedural modeling aims to create objects of specific categories algorithmically with rules and parameters, *e.g.*, plants [Prusinkiewicz 1986], terrain [Musgrave et al. 1989], buildings [Parish and Müller 2001], rivers [Kelley et al. 1988], *etc.* Our task is related to procedural modeling of buildings. Methods in this category mostly use rewriting systems, such as L-system [Coelho et al. 2005; Greuter et al. 2003], split grammar [Müller et al. 2006; Wonka et al. 2003], or shape grammar [Patow 2010; Silva et al. 2013]. Please refer to the survey by Smelik et al. [2014] for details.

To further control grammar-based procedural modeling, Talton et al. [2011] develop an algorithm based on Reversible jump Markov chain Monte Carlo to produce a model that conforms to a high-level specification of the desired model. Ritchie et al. [2015] propose Stochastically-Ordered Sequential Monte Carlo to receive feedback from incomplete models and handle many possible sequentializations of a hierarchical and recursive procedural model.

As mentioned in [Smelik et al. 2014], procedural modeling is instrumental for technical artists and is still far from widespread acceptance among non-technical, creative users. Yet, the recent Artificial Intelligence Generative Content (AIGC) approach [Hui et al. 2024; Rombach et al. 2022] demonstrates great success in generating high-quality images and 3D models, making the approach popular among mainstream users. Unlike grammar-based methods, AIGC leverages large datasets to learn to create contents of varying styles and categories. Hence, taking the AIGC approach to design our techniques, our method can well adapt to the high variety and complex connectivity of LEGO® models, enabling us to create LEGO® micro-building models of various shapes and styles.

Representations for 3D shape generation. Early works on 3D generation focus on voxel volumes and point clouds [Girdhar et al. 2016], but these representations have limited expressiveness. Polygonal meshes, which are well-suited for downstream tasks, have also been studied [Gao et al. 2024; Nash et al. 2020; Wang et al. 2018]. Recently, implicit representations, such as signed distance fields (SDF) [Chen and Zhang 2019; Hui et al. 2022, 2024] and neural radiance fields (NeRF) [DeVries et al. 2021; Kosiorek et al. 2021; Schwarz et al. 2020], are popular for 3D shape generation. They offer superior capabilities in modeling diverse shapes and appearances.

The aforementioned 3D shape generation tasks focus on 3D object sufaces. Yet, LEGO® models are more than 3D surfaces. They are made up of bricks with intrinsic internal connections. So, the representation should encode more than just the external surfaces, but also the internal structures, in order to enable the generative model to learn to produce feasible LEGO® models.

Learning-based assembly. Our task is related to Design for Assembly (DFA), which aims for fast and high-yield manufacturing. DFA has two main tasks: (i) find an optimal sequence to assemble component parts, given the parts and target assembly, and (ii) predict the joint of two/more parts, given the 3D representation of the parts.

For task (i), various optimization and simulation-based techniques have been studied to find the optimal assembly sequence [Chen et al. 2006, 2008; Qin and Xu 2007; Sinanoğlu and Börklü 2005; Tian et al. 2022]. However, learning-based assembly planning is relatively new. Wang et al. [2022b] present a dataset, consisting of IKEA objects paired with assembly manuals and fine-grained annotations. Wang et al. [2022a] study the problem of translating image-based, step-by-step assembly LEGO® manuals created by human designers into machine-interpretable instructions.

For task (ii), some datasets have been compiled for joint prediction. Jones et al. [2021] compile the first large-scale dataset of parametric boundary representation CAD assemblies and train a graph convolution network to predict the CAD mate type. Willis et al. [2022] present the fusion 360 assembly dataset, containing assemblies with rich information on joints, contact surfaces, and holes, with the assembly graph structure. They further train a network over a graph representation of solid models to predict joints.

However, both tasks require the target assembly/parts to be known, in order to plan the joints or to find a feasible assembly

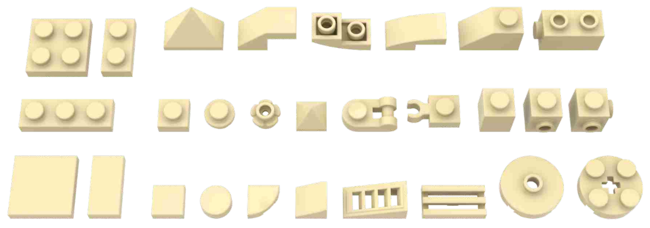


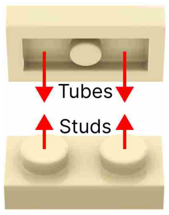
Fig. 3. The brick set employed for creating LEGO® micro buildings.

sequence. Unlike these tasks, this work aims to *generate* 3D LEGO® models assembled with diverse bricks by learning from data samples, without prior knowledge of the target assembly/parts.

3 Overview

Task definition. In this work, we aim to train a generative model that can learn the semantic structures and bricks connections in LEGO® micro buildings, such that it can produce 3D semantic information reconstructible into LEGO® models. Importantly, the resulting models should make up of coherently-connected bricks and exhibit building elements such as storeys and rooftops, over a LEGO® baseplate. To enhance the appearance and diversity of the produced LEGO® models, we consider not only rectangular bricks but also bricks of various shapes such as slopes, round plates, grilles, etc., see Figure 3 for the 28 types of bricks we employed.

As the first attempt to take a learning-based pipeline to generate LEGO® models, particularly considering assorted bricks types and bricks connections, we focus our effort on micro buildings rather than general models.



So, we consider a brick set of a manageable size for creating LEGO® micro buildings; see again Figure 3. Also, we limit the physical dimensions (width, depth, and height) of the data samples to be $4.8cm \times 4.8cm \times 6.4cm$ to control the computational overhead and consider only bricks that are inter-connected by studs and tubes; see the inset figure on the right. Further, we consider only bricks placements with axis-aligned (rather than arbitrary) orientations. See the red arrows in Figure 4; they mark bricks that are rotated by 90 degrees to connect with side studs on the modified LEGO® bricks[2]. We leave the efforts to generalize the method to bricks placements with arbitrary orientations as future works.

Challenges. Despite the above considerations to try to make the task manageable for training the generative model, to learn to create LEGO® micro buildings is still very challenging for various reasons.

First, existing widely-used 3D representations such as polygonal meshes, implicit fields, and neural volumes are designed to encode standalone 3D shapes, without considering building blocks and connections. Yet, the task requires a uniform representation that can effectively capture not only the 3D shape features but also the bricks and the bricks connections. Also, the representation should allow backpropagation for optimization, *i.e.*, differentiable.

[2]Modified LEGO® brick means a regular LEGO® brick modified with one or more studs on its side; see the brick at top-right corner and the two bricks below it in Figure 3.

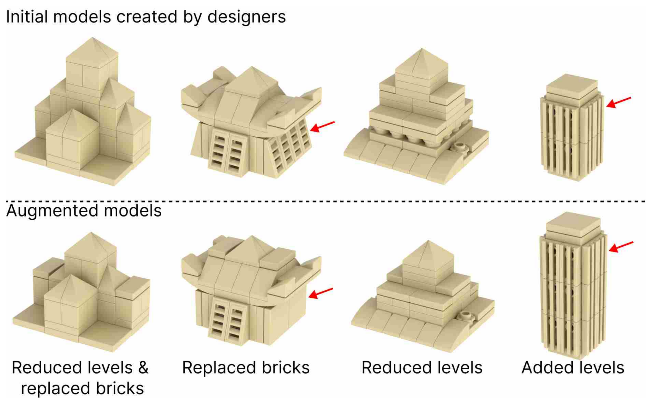

Initial models created by designers
Augmented models
Reduced levels & replaced bricks
Replaced bricks
Reduced levels
Added levels

Fig. 4. Some of the LEGO® micro buildings in our dataset. Top row: initial models created by the recruited designers. Bottom row: models augmented by brick replacement and level adjustment. Particularly, beyond the default up-facing orientation, bricks can be re-oriented sideways (see the red arrows) to connect with the side studs on the modified LEGO® bricks[1].

Second, LEGO® bricks are discretely placed and connected, generally via one/multiple pairs of studs and tubes. We will elaborate on the connection mechanism later in Section 4. How to encode brick connection information effectively in a representation for the generative model to learn from is still an open question.

Third, considering many non-rectangular LEGO® bricks types and flexible connection choices, the brick arrangement is essentially a combinatorial optimization, which has an immense and discrete solution space. A small local variation of a brick placement may propagate to large changes in the appearance and connections of the final assembly. So, we need to learn the assembly structures over a vast and discrete space, where outliers may fail to be reconstructed to buildable and aesthetic assembly structures.

Last, the lack of datasets hinders the development of data-driven methods for LEGO® generation. We remedy this by contributing the first LEGO® micro-building dataset that features a rich variety of building styles and scales.

Our Pipeline. Our pipeline has five major steps as Figure 2 depicts:

(i) Dataset preparation. We start this research by creating a LEGO® micro-building dataset. First, we studied LEGO® micro buildings and identified expressive bricks suitable for creating the micro buildings; see again Figure 3. Then, we recruited three expert designers to create LEGO® micro buildings using this brick set. In short, the designers employed the Studio Software [Bri 2024] to first create around 300 models, then expanded the set into one thousand models by adjusting the number of repeated patterns (i.e., levels or storeys), swapping partial building parts between data samples, and splitting large buildings into multiple smaller ones. Further, we examine each data sample to ensure design criteria such as scale, connection, stability, non-overlap, and diversity; see Figure 4 for example models in the dataset.

(ii) LEGO® semantic volume representation. In this step, we parse the data file of each LEGO® model sample in the dataset and convert it into a representation that can be learned by the generative

model. Importantly, we propose to use a volumetric representation to spatially encode both the bricks types and bricks connections, *i.e.*, the studs and tubes, and carefully design the voxel dimensions and granularity in our representation. See Section 4 for the details.

(iii) Semantic volume atomization. Next, to achieve efficient learning, while avoiding ambiguity in the representation, we consider the transformative nature of LEGO® and propose to atomize the LEGO® semantic volume, say, *e.g.*, by decomposing the 2×1 LEGO® plate into two 1×1 LEGO® plates (Figure 2). See Section 5.

(iv) Semantic volume generation. Next, we construct the network architecture of the generative model and formulate a classification task to train it with the cross-entropy loss to learn to predict the occupancy types in the semantic volume. See Section 5.

(v) Progressive reconstructor. Last, we design the LEGO® reconstructor to create LEGO® models from the generated semantic volumes. In short, we iteratively arrange simplex bricks from bottom to top, then merge the bricks to produce the final LEGO® model. See Figure 2 for an example and Section 6 for the details.

In the end, we employed our pipeline to generate a wide range of models and conducted a series of evaluation experiments to demonstrate the quality, fidelity, and diversity of our results.

4 LEGO® semantic volume

After constructing the dataset, we next aim to design a representation to encode necessary information for the generative model to learn from. At the beginning of this research work, we considered two possible approaches: (i) a graph representation with nodes (bricks types & locations) and links (bricks connections) to model the LEGO® assembly; and (ii) a 3D volume representation that voxelizes the LEGO® model. The graph representation is a natural choice for modeling an assembly structure but it cannot effectively encode spatial information and bricks information together for learning. Also, it cannot easily take advantage of the powerful denoising diffusion model, which has demonstrated strong capability of generating grid-structured data such as images, videos, and implicit fields. Hence, we proceed to design a volume representation.

Before looking into our representation, we first describe the LEGO® brick system, which leads us to designing the representation. Fundamentally, LEGO® bricks are connected via an interlocking system [Kirk 1961], primarily by the studs and tubes on the bricks. Studs are cylindrical protrusions on top of conventional bricks, whereas tubes are hollow cavities under many bricks for connecting with the studs; see again the inset figure in Section 3. When properly connected, they exert strong frictional forces to put the associated bricks together. Importantly, they occupy certain space at dedicated locations in the LEGO® bricks and are paired across the associated connecting bricks.

To make up the volume representation, after exploring alternative design choices, we found it important to *spatially encode both the bricks types and bricks connections* in the representation, such that the generative model can be aware of learning and generating also the studs and tubes to ensure proper bricks connections; see an ablation study in Section 7.4. On the other hand, we considered

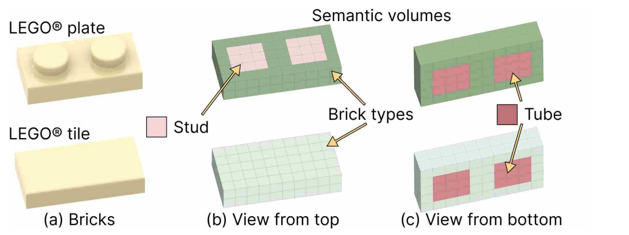


Fig. 5. (a) The 2×1 LEGO® plate brick (top) and 2×1 LEGO® tile brick (bottom). Their semantic volumes when viewed (b) from top and (c) from bottom. Each occupied voxel stores either a brick type ID or a stud/tube ID.

an alternative design of additionally predicting the *brick instances*. However, we found it hard to train the generative model to learn to produce also the brick instance IDs per voxel, so we defer the task of creating brick instances at a later stage of the pipeline.

Motivated by the above considerations, we design the *LEGO® semantic volume representation* to spatially encode brick type and brick connection information, and formulate a procedure to construct the representation with the following two steps.

(i) Extract Bricks Placements and Bricks Connections. The LEGO® model samples in the constructed dataset are stored in the LDraw file format [Jessiman 1995]. So, for each data sample, we parse its LDraw file and extract a list of LEGO® bricks in the model. For each brick, we can obtain its brick type, which is a unique ID, and also its 3D location, which is represented by a 3D transformation matrix. Since connection information is not explicitly available in the LDraw file format, we annotated the position and orientation of each stud and tube in each brick of the employed brick set. By doing so, we can obtain the locations of all the studs and tubes in the LEGO® model and locate the connecting studs and tubes.

(ii) Construct LEGO® Semantic Volume. Next, we convert each LEGO® model sample into a LEGO® semantic volume, which is a grid of $W \times D \times H$ voxels, each storing a one-hot vector $\mathbb{1}^C$, where C denotes the number of channels; W denotes width; D denotes depth; and H denotes height. In detail, $C = 1 + 28 + 2 = 31$, where the first channel is one if the voxel is empty; the next 28 channels indicate the type of the occupied brick if the voxel is not a stud or tube; and the last two channels indicate stud or tube. Note that we use one-hot vectors instead of integer values to denote brick/connection type IDs, helping to ensure that the 31 occupancy types are independently chosen by the neural network.

Procedure-wise, we start with an empty semantic volume, then fill the voxels occupied by each brick with its brick type ID, stud ID, or tube ID accordingly. Figure 5 shows example LEGO® semantic volumes for two different bricks: a LEGO® plate with studs on top and a LEGO® tile without studs on top. As the examples show, we encode one layer of LEGO® plate (thickness 0.32 cm) using two layers of voxels, such that studs are encoded at the relevant voxels in the top voxel layer whereas tubes are encoded in the bottom voxel layer of the bricks. Note that we use the same voxel coloring scheme to indicate the bricks types, studs, and tubes in the entire paper; see the legend at the top right corner of Figure 2. Also, Figure 6 shows a running example using a simple four-brick LEGO® model.

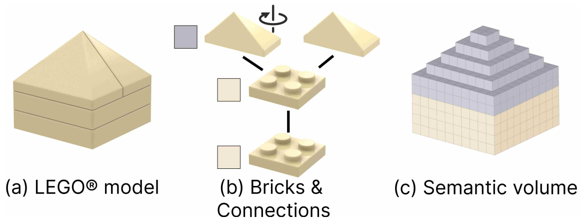


Fig. 6. A running example. From the given LEGO® model (a), we locate the bricks (b), then construct the LEGO® semantic volume accordingly (c).

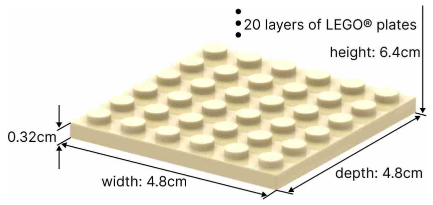
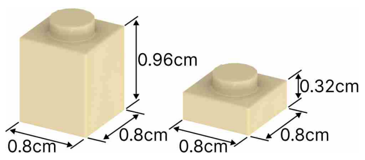


Fig. 7. Limiting physical size of LEGO® models. The max width/depth is 4.8 cm, corresponding to a grid of six-by-six units of studs. The max height is 6.4 cm, corresponding to the height of stacking 20 LEGO® plates.

Granularity of Semantic Volume. Choosing a proper granularity for the LEGO® semantic volumes is essential for preserving the geometric details of LEGO® models, while avoiding computational overhead in training and preventing non-overlapping bricks from occupying the same voxel. In practice, a simple one-by-one LEGO® brick can be represented by a volume of $0.8cm \times 0.8cm \times 0.96cm$, and a one-by-one LEGO® plate by a volume of $0.8cm \times 0.8cm \times 0.32cm$; see the inset figure. To accurately represent any model built by these simple bricks, the side length (width/depth) of the voxel cubes in the semantic volume—which also corresponds to the smallest discrete displacement of any brick in the system—should at least be the greatest common divisor of the brick dimensions. In this case, the side length is $0.16cm$. On the other hand, this number is equivalent to half the plate thickness, as the representation needs to encode studs and tubes of a plate in its two separate layers; see, *e.g.*, Figure 5 (b & c).



Based on this granularity setting, Figure 7 illustrates the limited physical size of LEGO® models considered in this work, *i.e.*, a volume of $4.8cm \times 4.8cm \times 6.4cm$. This physical volume proportionally corresponds to a LEGO® semantic volume of $W = D = 30$ and $H = 40$, *i.e.*, six-by-six LEGO® studs horizontally and a stack of 20 LEGO® plate bricks vertically. Also, with such a granularity, each LEGO® plate unit corresponds to a sub-grid of $5 \times 5 \times 2$ voxels; see again the simple examples in Figures 5 and 6.

Discussion. The connection voxels for studs and tubes help encourage the neural network to learn to locate the studs and tubes more accurately, bringing us the following two benefits. First, guided by the stud and tube voxels, the generative model can better arrange

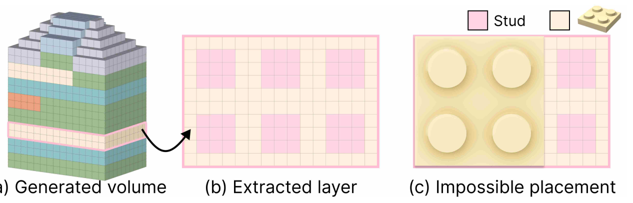


Fig. 8. When trained with the LEGO® semantic volumes, the generative model may produce results with non-fill-able brick type IDs. For the 3×2 layer extracted above, its voxels are filled with the ID of the 2×2 LEGO® plate. We may use the 2×2 brick to fill any voxel in this layer. However, we cannot fill the entire layer with only 2×2 bricks, without bricks overlap.

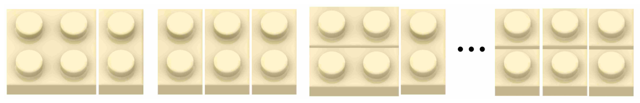


Fig. 9. Transformative nature of building things with LEGO® bricks. There are multiple ways of filling/building the 3×2 layer shown in Figure 8.

inter-connected bricks. Second, the connection voxels provide additional cues for the neural network to differentiate individual bricks from a cluster of voxels of the same brick type.

5 LEGO® model learning

With the LEGO® semantic volume prepared for each dataset sample, we can start to train a generative model to learn to produce such a volume. However, we experimentally found that the generated semantic volumes may contain voxel layers that cannot be filled by the bricks types recommended by the generative model.

Issue with the LEGO® semantic volumes. Figure 8 reveals the issue. For the 3×2 layer extracted in the generated semantic volume shown in Figure 8(b), we can see that the voxels contain the brick ID of the 2×2 LEGO® plate. While it is feasible to use the 2×2 plate to fill any voxel in this layer, if we are allowed to use only the 2×2 plate, we cannot fill the entire layer without bricks overlap. The underlying reasons are two-fold. First, when the generative model is trained to assign brick type IDs, it can only predict bricks placements that are locally feasible. It cannot have a global view to ensure compatible bricks placements over the entire layer. Second, LEGO® is transformative by nature. As Figure 9 shows, the same single layer, or more generally a shape, can be built using different bricks arrangements. So, it is a huge burden for the generative model to further learn to ensure compatible bricks arrangements.

Atomization. Inspired by the transformative nature of LEGO®, we propose the idea of atomizing the LEGO® semantic volumes and train the generative models on the atomized volumes. Overall, we first identify the bricks in the brick set that can be atomized; see the leftmost five bricks in Figure 3. Among them, the top three LEGO® plates can be atomized into four, two, and three 1×1 LEGO® plates next to them, whereas the bottom two LEGO® tiles can be atomized into four and two 1×1 LEGO® tiles next to them in the figure. Hence, after the atomization, the number of channels C in the semantic volume can be reduced from 31 to 26. Besides, Figure 2

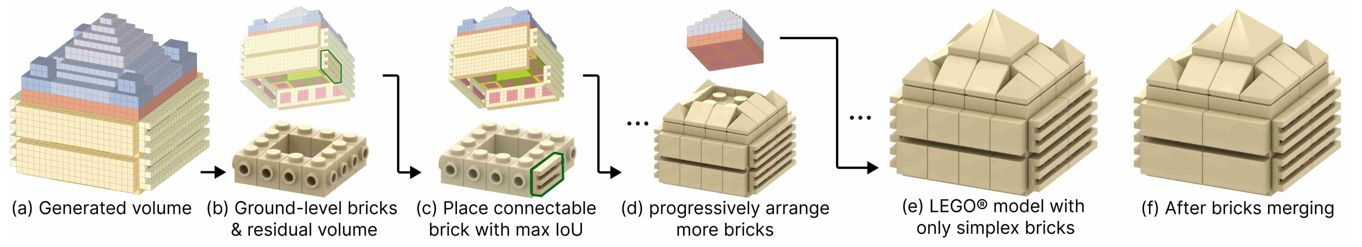

(a) Generated volume
(b) Ground-level bricks & residual volume
(c) Place connectable brick with max IoU
(d) progressively arrange more bricks
(e) LEGO® model with only simplex bricks
(f) After bricks merging

Fig. 10. A running example of LEGO® reconstruction. From (a) a generated semantic volume, we (b) first arrange ground-level bricks, then (c & d) progressively place connectable bricks with maximized IoU until (e) a LEGO® model is reconstructed. Further, we merge simplex bricks to enhance brick connectivity (f).

presents a full example of atomizing a LEGO® semantic volume, which is simplified to contain only the *simplex* LEGO® bricks, while still representing the same overall 3D shape.

The atomization process is important. First, it reduces the ambiguity, stemming from the transformative nature of LEGO®. For the different bricks arrangements shown in Figure 9, although they represent the same shape, they have different semantic volumes. Since the generative model should first be tasked to learn the shapes and bricks connections in the dataset, this ambiguity poses an unnecessary burden for effective shapes and connection learning. Second, it remedies the aforementioned incompatible bricks arrangements. Without brick-instance information, a region in the generated semantic volume may not be filled with the assigned brick type; see again Figure 8. The atomization allows to decouple the problem, so that the generative model can focus fully on the overall shape and bricks connections, while the LEGO® reconstructor later on (see Section 6) will help to arrange the actual brick instances.

Network Architecture. We formulate the generative model based on denoising diffusion [Ho et al. 2020], adopting a U-Net architecture with 3D convolutions to predict the voxel labels. Overall, the encoder network has four downsample blocks, each of three layers, whereas the decoder network has four upsample blocks, each of four layers. Also, there are three downsampling and three upsampling operations, so we zero-pad the semantic volume from $30 \times 30 \times 40$ to $32 \times 32 \times 40$ before feeding it to the U-Net. Hence, the input volume scales for the four blocks are $32 \times 32 \times 40$, $16 \times 16 \times 20$, $8 \times 8 \times 10$, and $4 \times 4 \times 5$, and their hidden dimensions are 192, 384, 576, and 768, respectively. To further learn the spatial correspondence between bricks, we adopt transformer layers after the convolution layers in the two blocks at the smallest scales.

Network Training. To learn the shape and connection distribution of the LEGO® models from the atomized semantic volumes, we propose to formulate the model training as a discrete diffusion-based classification task. First, instead of predicting the applied noises at each diffusion step, our generative model is trained to predict the semantic volume itself from random initial noises. Second, instead of predicting an integer-valued brick/connection type ID for each voxel, supervised by the widely-adopted Mean Square Error (MSE) loss, we propose to use a one-hot vector for each voxel and formulate a per-channel classification task, supervised by the cross-entropy loss. Specifically, the generative model is trained to progressively predict the semantic volume $f_\theta(V_t, t) \in \mathbb{R}^{C \times D \times H \times W}$ from a random noise volume ϵ, supervised by

$$\mathcal{L} = \mathbb{E}_{t,\epsilon}\left[-\log \frac{\exp(f_\theta(V_t, t)_{V_{gt}})}{\sum_{c=1}^{C} \exp(f_\theta(V_t, t)_c)} \right],$$

where t is the time step; V_t is the noisy semantic volume at timestep t; V_{gt} is the label of the ground-truth volume; $f_\theta(V_t, t)_c$ is the volume slice of the c-th channel of the semantic volume V_t; and $f_\theta(V_t, t)_{V_{gt}}$ is the volume where the value of the voxel at position (x, y, z) is the value of $f_\theta(V_t, t)$ at the $V_{gt_{(x,y,z)}}$-th channel and (x, y, z) position.

Network Inference. At the inference, we employ the softmax function to normalize $f_\theta(V_t, t)$, *i.e.* ,

$$\text{Softmax}(f_\theta(V_t, t)_i) = \frac{\exp(f_\theta(V_t, t)_i)}{\sum_{j=1}^{C} \exp(f_\theta(V_t, t)_j)},$$

which indicates the probability distribution of the voxel occupancy types in the generated volume. We take $\text{Softmax}(f_\theta(V_0, 0))$ as the generated semantic volume and feed it to the LEGO® reconstructor (see Section 6) to produce the final LEGO® model.

6 Progressive LEGO® Reconstructor

We design the LEGO® reconstructor to arrange bricks based on the generated semantic volume to produce a coherently-connected LEGO® model. Figure 10 shows a running example. Overall, the key idea is to progressively arrange bricks in a bottom-up manner to cover the generated semantic volume, while respecting the network-recommended bricks types and ensuring proper bricks connections. Here, we first arrange simplex LEGO® bricks, and then merge the bricks to reduce the brick count. Optionally, we analyze the building levels in the structure to assign colors to the bricks. Overall, the reconstructor has the following three steps. More details are provided in Supplementary material Part F.

Step (i) Arrange Bricks. To start, we first arrange *simplex* bricks to cover the generated semantic volume iteratively in a bottom-up manner, from ground-level bricks (Figure 10 (b)) to upper-level bricks (Figure 10 (c & d)). In each iteration, we locate a bottom-most uncovered voxel in the generated semantic voxel and select a brick with the largest IoU with the semantic volume (*i.e.*, the brick with the largest percentage of matched brick- and connection-type IDs) that covers the located voxel. Note that the selected brick, if not ground-touching, should connect with some previous brick(s) primarily via a stud-tube connection. Yet, for rectangular simplex bricks, we temporarily allow them to be placed in contact with

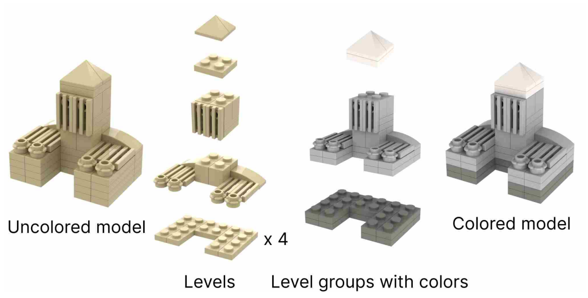


Fig. 11. Height-range decomposition and color assignment to bricks.

same-type bricks side-by-side, since they can be merged into larger non-simplex rectangular bricks later on.

Step (ii): Merge bricks. Local groups of rectangular simplex bricks (*i.e.*, 1×1 LEGO® plates/tiles) can be merged into larger rectangular bricks (the left five bricks in Figure 3). To do so, we first enumerate all possible candidate instances of these rectangular bricks that cover the arranged rectangular simplex bricks. Then, our objective is to find a minimal and non-overlapping subset of these candidate brick instances, which exactly cover the target simplex bricks. This process is formulated as a constrained integer programming and solved using [Gurobi Optimization, LLC 2024].

Step (iii): Assign Brick Colors. Optionally, as Figure 11 shows, we decompose the arranged LEGO® bricks into groups for different height ranges (levels), then assign a standard LEGO® color to each group based on predefined color palettes. We admit that this is a simple scheme without considering structures and bricks semantics. A more structure-aware color assignment strategy may necessitate a semantic segmentation of the LEGO® model. Possible methods include supervised training on the dataset annotated with brick color and differentiable rendering supervised by pre-trained models like CLIP [Radford et al. 2021]. We leave this as a future work.

7 Results and experiments

7.1 Implementation Detail

We implemented our neural network model using PyTorch and ran all the experiments on a single A40 GPU. We employed the Adam optimizer to train the generative model for 2,000 epochs, using a batch size of 20, a learning rate of $1e^{-4}$, and a total of 1,000 denoising steps. Training the generative model took around one day. Inference takes around 20 seconds per LEGO® model on an A40 GPU.

7.2 Gallery of our Results

In addition to Figure 1, we present 24 more results in Figure 12 to showcase the compelling capability of our method. These results cover a wide range of architectural styles, including arch (*e.g.*, column 1 top), castle gates (*e.g.*, column 1 middle & bottom), Greek temple (*e.g.*, column 2 top) traditional pavilions (*e.g.*, column 2 middle & bottom), modern buildings (*e.g.*, column 3), churches (*e.g.*, column 4), and sculptures (column 8). The results exhibit *diverse topologies and fine details without obvious artifacts*, in which the bricks are all coherently connected.

For example, though the buildings have layered structures, many of them contain hollowed exteriors, reproducing windows, doors, passages, and other openings. Also, the building styles are accompanied by various levels of complexity. Some results show stacks of simple bricks to mimic modern buildings, whereas some present intricate patterns, exhibiting fine details in traditional buildings. On the other hand, we found the generated models mostly exhibit symmetry without explicit constraints, showcasing the capability of our method to learn the structural patterns at a global scale.

7.3 Quality Evaluation

Metrics. Following [Chen et al. 2021; Hertz et al. 2022], we evaluate the generation quality using (i) minimum matching distance (MMD), which measures the fidelity of the generated LEGO® models, and (ii) 1-NN classifier accuracy (1-NNA), which measures how well a classifier differentiates the generated model from those in the dataset. In practice, we uniformly sample 2,048 points on each of our generated models (treated as generated shapes), and also on each model in the dataset (treated as reference shapes). Then, we use Chamfer Distance (CD) and Earth Mover's Distance (EMD) as the distance measures for MMD and 1-NNA. CD measures the average deviation between a generated shape and a reference shape, whereas EMD measures the effort required to transform a generated shape into a reference shape. Overall, a low MMD and a 1-NNA close to 50% indicate good generation quality.

Besides, we introduce two additional metrics: (iii) *Voxel Reconstruction Accuracy* (VRAcc) and (iv) *Mean Maximum Voxel Accuracy* (MMAcc). The VRAcc measures how well the reconstructed model matches the generated semantic volume. Specifically, after we obtain the final output model guided by the generated semantic volume from the network, we calculate the voxel accuracy between the final output and the generated semantic volume. A high VRAcc implies that the generated semantic volume has clean shapes and connections with little noise for model reconstruction. The MMAcc measures the shape and connection similarity between the generated model and the dataset. We determine the maximum voxel accuracy (MAcc) between each generated model and all samples in the dataset, then compute the mean of these MAcc values across all generated models as the MMAcc. This metric aims to remedy the shortcomings of CD and EMD, which consider only the shape similarity and ignore the connection similarity. In our task, LEGO® models of the same external shape but different internal connections are not identical. A high MMAcc indicates that the generated LEGO® models are highly similar to the samples in the dataset; please refer to the Supplementary material Part B for details.

Quantitative evaluation. We employ the above four metrics to quantitatively evaluate our results. First, we generated 400 LEGO® models from random noise volumes. Second, we randomly selected 400 samples from the dataset. Third, we compared our method with a naive method, as a baseline, which arranges only rectangular LEGO® plates when creating LEGO® micro buildings, similar to the voxel-like shapes produced by many existing works. To do so, we converted the LEGO® semantic volumes of the 400 generated models all to be 1×1 simplex LEGO® plates and reconstructed LEGO® models from the converted volumes.

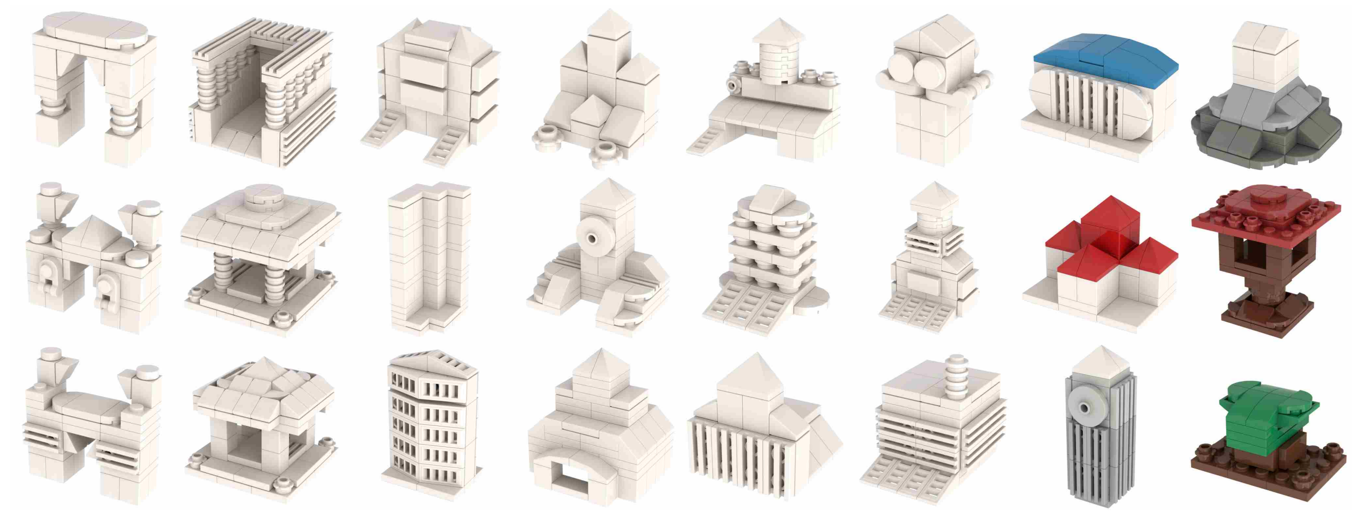


Fig. 12. A gallery of LEGO® micro buildings created by our pipeline. The results exhibit a rich variety of styles and quality, ranging from arches to city buildings, from bell towers to ancient tempers, and from gazebos to sculptures. The rightmost two columns show LEGO® models with brick color assignment.

Table 1. Qualitative evaluation of our method. The unit of CD and EMD in MMD is 10^{-4}, whereas the unit of VRAcc and MMAcc is percentage (%).

Method	MMD ↓ CD	MMD ↓ EMD	1-NNA (~50) CD	1-NNA (~50) EMD	VRAcc ↑	MMAcc ↑
Naive method	11.21	15.74	82.75	80.87	89.03	39.59
Ours	**9.74**	**12.58**	**57.28**	**53.05**	**95.34**	**70.77**

Overall, we have 1,200 LEGO® models, *i.e.*, 400 generated by our pipeline, 400 by the naive method, and 400 from the dataset. Then, we evaluated our results and the naive method results on the four metrics. As Table 1 shows, our generated models have high fidelity like the human-crafted dataset samples. In addition, Figure 13 provides some visual comparisons. Both the quantitative and visual results show that the naive method can hardly show meaningful and aesthetic shapes under at the micro-building scale, meaning that the results cannot well reproduce micro buildings with rich details. Therefore, considering a wide array of specialized LEGO® bricks helps produce LEGO® models aligned with the human designs, thanks to their versatile expressiveness.

7.4 Model Analysis

Ablation study. To evaluate the effectiveness of some of the major designs in our pipeline, we consider the following ablation cases: (i) supervising the training using the MSE loss instead of the cross-entropy loss; (ii) removing the connection (stud/tube) channels in the semantic volumes by simply considering the brick type IDs; and (iii) removing the *atomization* process before training.

In Case (i), for each atomized semantic LEGO® volume, we need to replace the one-hot vectors with the integer-valued brick/connection type IDs, so as to use the MSE loss to supervise the prediction of the noises added to the integer-valued semantic volumes. Here, the generated semantic volumes are found to possess massive noises. Thus, they cannot be reconstructed in practice. Hence, we exclude this case in the subsequent quantitative and qualitative evaluations. The underlying reason is that the MSE loss penalizes values distant from the ground truths, yet the brick/connection type IDs are pure semantic labels, carrying no distance meanings. So, the prediction of brick/connection IDs is a classification task, not a regression task.

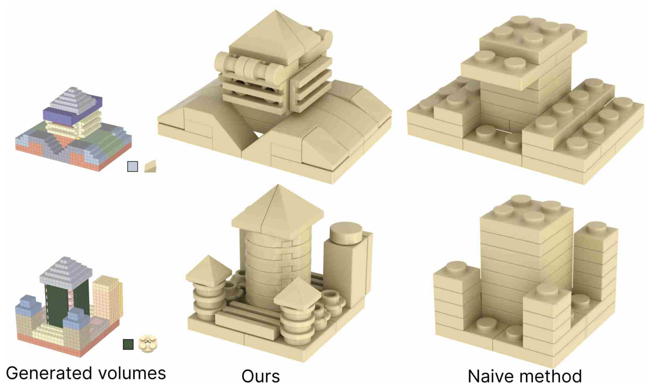


Fig. 13. LEGO® models generated by our method (middle column) and by the naive method (right column). Our method employs expressive bricks to depict nontrivial building structures, whereas the naive method simply uses rectangular bricks, thus failing to show rich details in micro buildings.

Table 2. Ablating our full pipeline. The unit of CD and EMD in MMD is 10^{-4}, whereas the unit of VRAcc and MMAcc is percentage (%).

Method	MMD ↓ CD	MMD ↓ EMD	1-NNA (~50) CD	1-NNA (~50) EMD	VRAcc ↑	MMAcc ↑
w/o atomization	13.02	17.37	62.44	59.15	80.11	55.53
w/o connection	9.82	12.78	60.98	57.75	91.87	65.99
Full pipeline	**9.74**	**12.58**	**57.28**	**53.05**	**95.34**	**70.77**

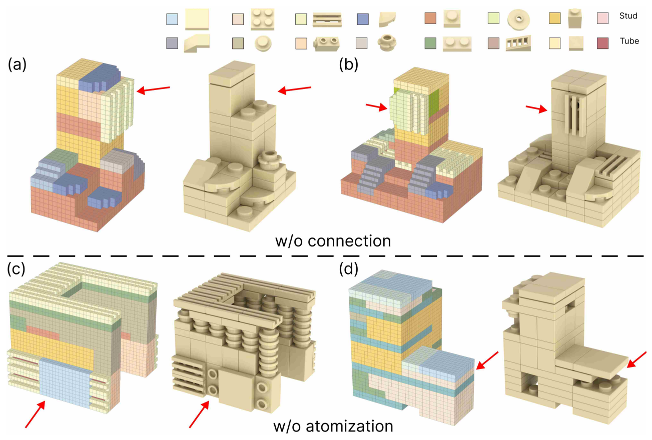


Fig. 14. Results produced by our pipeline without connection channels for studs and tubes (top) or without the atomization process (bottom). The red arrows mark obvious artifacts related to regions failed to be reconstructed.

For Cases (ii) and (iii), we generated 400 results for each case. Then, we compare them with a randomly-selected subset of the dataset, following the experiment presented in Section 7.3. Table 2 reports the quantitative results of our full pipeline against the two ablated cases. The full pipeline quantitatively outperforms the ablated ones, signifying the importance of the atomization process and the connection information encoded in the LEGO® representation.

Figure 14 visualizes the generated volumes and the final LEGO® models, where the ablated models produce obvious artifacts. First, for Case (ii), see Figure 14 (a), the generative model assigns the type ID of the $1 \times 1 \times 2$ modified brick (see the very soft orange voxels) to a region with only half of the height of the intended brick. Subsequently, the intended grille bricks (light yellow voxels) cannot be attached to the main structure. In Figure 14 (b), there are two artifacts marked by the red arrows. First, only half of the circle plate (soft light green voxels) is generated. Second, this halfly-generated circle plate is predicted to be placed on the side surface of a $1{\times}1{\times}1$ brick, where no studs nor tubes are available for making the connections. The two examples show that the neural network makes use of the connection information to better locate the individual bricks. Without such guidance information, the generative model may fail to accurately perceive the shapes of individual bricks and consequently fail to arrange proper connections.

Proceeding to Case (iii), In Figure 14 (c), the generative model created a 2×3 region and assigned the voxels with the label of the 2×2 tiles, making it impossible to find a feasible placement. In Figure 14 (d), there is also a 2×3 region below the LEGO® tiles marked by the red arrows that are intended to be filled by 2×2 plates. These failure cases further confirm the ideas in Section 5. Without atomization, the generative model may struggle to arrange non-atomic bricks to feasibly fill the predicted target shapes.

Interestingly, from Figure 14, we observe that ablating the two components may result in similarly failing cases. This indicates that the generative model jointly makes use of the connection voxels and the atomization process to effectively arrange bricks types, without explicitly requiring instance-level information.

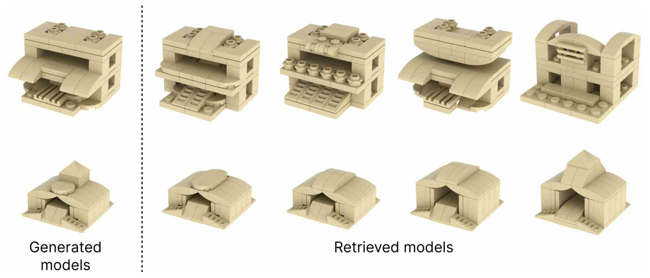


Fig. 15. Leftmost column: LEGO® models generated by our method. The other four columns: similar LEGO® models retrieved from the dataset with the Maximum Voxel Accuracy.

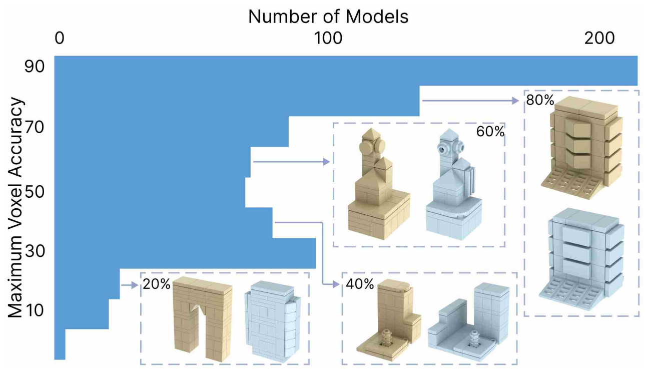


Fig. 16. Distribution of Maximum Voxel Accuracy between the 400 generated LEGO® models and the LEGO® models in dataset, showing that our pipeline can generate models highly similar and also more different relative to those in the dataset. Some example pairs (generated in yellow and retrieved in blue) at 20%, 40%, 60%, and 80% voxel accuracy are shown.

Novelty analysis. Furthermore, we study whether our pipeline can generate LEGO® models, not necessarily the same as those in the dataset. To do so, for each of the 400 generated models, we retrieve the top four most similar models in the dataset using Voxel Accuracy. The Voxel Accuracy is computed between each generated model and all dataset samples. Figure 15 shows two of the 400 cases, together with the top four most similar models found in the dataset. The two example models clearly exhibit structures and details distinct from the similar dataset samples, while being connected. More results can be found in Supplementary material Part E.

We further statistically analyze the novelty of our generated models relative to the models in the dataset. Figure 16 plots the distribution of the Maximum Voxel Accuracy between our generated LEGO® models and the samples in the dataset. Also, we show four LEGO® model pairs (generated and retrieved) at different Maximum Voxel Accuracy, *i.e.* , 20%, 40%, 60%, and 80%. From the distribution plot, we can see that our generative model can learn a distribution that covers the models in the dataset (with high voxel accuracy) and also generates novel models (with low voxel accuracy) that exhibit different features from the dataset samples.

Other experiments. We provide further experimental results in the Supplementary material: (i) *Time analysis for the LEGO® reconstructor* in Part C and (ii) *Distribution of the metric VRAcc* in Part D.

7.5 User study

To evaluate the quality of the LEGO® models generated by our method, we compare the quality of our generated models with the quality of the samples in the dataset, which contains both initial models created by designers and augmented models.

First, we use a binomial test, in which our key idea is to test if humans can differentiate our generated models from the human-crafted ones. If they cannot, it means the quality of our models is on par with the human-crafted models in the dataset. To start, we randomly picked ten models generated by our method and another ten models from the dataset. According to the hypothesis of the binomial test, if humans cannot differentiate whether a model is created by our method or by a human designer, the accuracy should be 0.5. We recruited ten participants in this user study, comprising four females and six males. Procedure-wise, we first showed them the 20 LEGO® models in a random order and told them that each model is created either by a human or by an AI tool. Then, each participant needed to mark their choice on each model. In the process, we did not tell the participant how many of the 20 LEGO® models were generated by an AI tool, nor did we limit the number of choices on AI-generated models. The overall accuracy is found to be **0.55**, which is very near to the perfect value (0.5), showing that our generated LEGO® micro buildings are almost indistinguishable from the human-crafted models.

Further, to quantitatively compare the quality of the generated models with the human-crafted ones, we asked each participant to rate each of the 20 models from one (worst) to five (best) in terms of visual quality, without informing the participant which models were designed by humans or created by our method. It turns out that the average rating for the generated models is **2.82**, and the average rating for the human-designed models is **3.18**, which is very close. Details can be found in Supplementary material Part G.

8 Conclusion, Limitations, and Future Works

This paper presents the first generative pipeline that can effectively learn and generate 3D LEGO® models, *i.e.*, micro buildings, considering a wide variety of non-rectangular bricks to enhance the expressiveness of the results. The LEGO® models generated by our pipeline demonstrate nontrivial structures, diverse styles, and high fidelity, almost indistinguishable from the human-crafted ones.

This work makes the following notable contributions: (i) the overall pipeline, which is a very first attempt to learn and generate non-rectangular LEGO® models in a data-driven manner, advancing over prior works on LEGO® construction; (ii) the LEGO® semantic volume representation, considering encoding both the bricks types and bricks connections while allowing back-propagation for optimization; (iii) the idea of atomizing LEGO® models, addressing the transformative nature of LEGO®, to facilitate effective learning; and (iv) a neural generative model partnered with the progressive LEGO® reconstructor, to generate coherently-connected LEGO® models that are physically assemble-able. Furthermore, we conducted a series of evaluations and analyses to assess the effectiveness of our generative pipeline, including the production of 400 LEGO® models using our pipeline and evaluating these models comprehensively using various metrics. In the end, we also performed an ablation study on our pipeline, studied the novelty distribution of our results, and conducted a user study to showcase the high fidelity of our results, compared with human-crafted models.

Limitations. This work still has limitations for creating more expressive and more controllable results. First, as a data-driven method, the scale of the dataset is relatively limited. Second, though we consider a wide variety of bricks types, the LEGO® system has several thousands of different types of brick [bri 2024], even excluding colors. The one-hot vector is not scalable to handle a larger brick set. Third, as an initial attempt to learn to generate LEGO® models, to ensure the network learning is manageable, we limit the physical size and connection types of the LEGO® bricks when building the dataset and the generative process is unconditional. Fourth, the LEGO® representation allows only axis-aligned bricks orientations. We can improve the expressiveness of our results if the generated bricks can have arbitrary orientations. Last, using a diffusion-based generative process sometimes introduces artifacts in the semantic volumes, resulting in generated volumes that cannot be fully reconstructed into a complete model. Details on failure cases and their causes can be found in Supplementary material Part H.

Discussion and Future works. Addressing each of the above limitations is already very challenging and will lead to the creation of a more general method with larger brick variety, arbitrary bricks orientations, and larger LEGO® models. At first glance, our benefits can be extended to other downstream tasks with extra conditions, *e.g.* , LEGO® model generation from photos, 3D shapes, or user sketches. Also, it would be interesting to explore the possibility of generating scenes, say a LEGO® city, beyond individual buildings. Following the high-quality generation applications in 2D images [Rombach et al. 2022] and 3D shapes [Hui et al. 2024], the dataset with corresponding multimodal annotations like text, multi-view images, and 3D shapes, is the key to conditional generation. Given the annotated dataset, we can develop a pipeline that can fuse information from text, images, or 3D shapes to generate contextually relevant LEGO® models. This can be achieved using existing techniques such as cross-attention and ControlNet [Zhang et al. 2023]. Additionally, considering the availability of powerful 3D shape generative tools trained on multimodal datasets, we can adapt these tools to the LEGO® generation task through transfer learning. There is a prospect to build a generative engine to aid the creation of LEGO® models given multimodal inputs with a continuous effort in creating a dedicated dataset at a larger scale.

Acknowledgments

We extend our sincere thanks to the anonymous reviewers for their invaluable comments and feedback. Our gratitude also goes to the LEGO® designers for their assistance in crafting the dataset. We would like to specifically acknowledge Wenrui Bao and Shaoxuan Yuan for their help in processing the LEGO® micro buildings dataset.

Finally, we thank all the participants who took part in our user study. This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK 14201921).

References

2024. BrickLink. https://www.bricklink.com/v2/main.page

2024. BrickLink - Studio Software. https://www.bricklink.com/v2/build/studio.page

Tomasz Boiński, Konrad Zawora, and Julian Szymański. 2022. How to Sort Them? A Network for LEGO Bricks Classification. In *Computational Science – ICCS 2022*. 627–640.

Guanlong Chen, Jiangqi Zhou, Wayne Cai, Xinmin Lai, Zhongqin Lin, and Roland Menassa. 2006. A framework for an automotive body assembly process design system. *Computer-Aided Design* 38, 5 (2006), 531–539.

Wen-Chin Chen, Pei-Hao Tai, Wei-Jaw Deng, and Ling-Feng Hsieh. 2008. A three-stage integrated approach for assembly sequence planning using neural networks. *Expert Systems with Applications* 34, 3 (2008), 1777–1786.

Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5932–5941.

Zhang Chen, Yinda Zhang, Kyle Genova, Thomas Funkhouse, Sean Fanello, Sofien Bouaziz, Christian Haene, Ruofei Du, Cem Keskin, and Danhang Tang. 2021. Multiresolution Deep Implicit Functions for 3D Shape Representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 13087–13096.

António Fernando Coelho, António Augusto de Sousa, and Fernando Nunes Ferreira. 2005. Modelling urban scenes for LBMS *(Web3D)*. 37–46.

Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. 2021. Unconstrained Scene Generation With Locally Conditioned Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14304–14313.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. 2024. GET3D: a generative model of high quality 3D textured shapes learned from images. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. 31841–31854.

Jiahao Ge, Mingjun Zhou, Wenrui Bao, Hao Xu, and Chi-Wing Fu. 2024. Creating LEGO Figurines from Single Images. *ACM Trans. Graph. (SIGGRAPH 2024)* 43, 4 (2024), 1–16.

Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. 2016. Learning a Predictable and Generative Vector Representation for Objects. In *European Conference on Computer Vision (ECCV)*. 484–499.

Rebecca A. H. Gower, Agnes E. Heydtmann, and Henrik G. Petersen. 1998. LEGO: Automated Model Construction. In *European Study Group with Industry*. 81–94.

Jens Gravesen and Poul G. Hjorth (Eds.). 1998. *32nd European Study Group with Industry, Final Report*.

Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. 2003. Real-time procedural generation of 'pseudo infinite' cities *(GRAPHITE)*. 87–ff.

Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. https://www.gurobi.com

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. SPAGHETTI: editing implicit shapes through part aware generation. *ACM Trans. Graph. (SIGGRAPH 2022)* 41, 4 (2022), 1–20.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. 6840–6851.

Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. 2022. Neural wavelet-domain diffusion for 3D shape generation. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.

Ka-Hei Hui, Aditya Sanghi, Arianna Rampini, Kamal Rahimi Malekshan, Zhengzhe Liu, Hooman Shayani, and Chi-Wing Fu. 2024. Make-a-shape: a ten-million-scale 3D shape model. In *Proceedings of the International Conference on Machine Learning (ICML)*. 20660–20681.

James Jessiman. 1995. LDraw.Org - LDraw File Format Specification. https://ldraw.org/article/218.html.

Benjamin Jones, 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. (SIGGRAPH Asia 2021)* 40, 6 (2021), 1–18.

Alex D. Kelley, Michael C. Malin, and Gregory M. Nielson. 1988. Terrain simulation using a model of stream erosion. *ACM Trans. Graph. (SIGGRAPH 1988)* 22, 4 (1988), 263–268.

Christiansen Godtfred Kirk. U.S. Patent 3005282, Oct. 1961. Toy building brick.

Adam R Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokra, and Danilo Jimenez Rezende. 2021. NeRF-VAE: A Geometry Aware 3D Scene Generative Model. In *Proceedings of the International Conference on Machine Learning (ICML, Vol. 139)*. 5742–5752.

Seung-Mok Lee, Jae Woo Kim, and Hyun Myung. 2018. Split-and-Merge-Based Genetic Algorithm (SM-GA) for LEGO Brick Sculpture Optimization. *IEEE Access* 6 (2018), 40429–40438.

Kyle Lennon, Katharina Fransen, Alexander O'Brien, Yumeng Cao, Yamin Beveridge, Matthewand Arefeen, Nikhil Singh, and Iddo Drori. 2021. Image2lego: Customized lego set generation from images. *arXiv preprint arXiv:2108.08477* (2021).

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. 2015. Legolization: optimizing LEGO designs. *ACM Trans. Graph. (SIGGRAPH ASIA 2015)* 34, 6 (2015), 1–18.

Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. 2023. Planning assembly sequence with graph transformer. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 12395–12401.

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. Procedural modeling of buildings. *ACM Trans. Graph. (SIGGRAPH 2006)* 25, 3 (2006), 614–623.

F. K. Musgrave, C. E. Kolb, and R. S. Mace. 1989. The synthesis and rendering of eroded fractal terrains. *ACM Trans. Graph. (SIGGRAPH 1989)* 23, 3 (1989), 41–50.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. 2020. PolyGen: an autoregressive generative model of 3D meshes. In *Proceedings of the International Conference on Machine Learning (ICML)*. 7220–7229.

Yoav I. H. Parish and Pascal Müller. 2001. Procedural modeling of cities. *ACM Trans. Graph. (SIGGRAPH 2001)*, 301–308.

Gustavo Patow. 2010. User-friendly graph editing for procedural modeling of buildings. *IEEE Computer Graphics and Applications* 32, 2 (2010), 66–75.

Pavel Petrovič. 2001. Solving LEGO Brick Layout Problem using Evolutionary Algorithms. In *Proceedings of the Norsk Informatikkonferanse (NIK)*. 87–97.

Maxim Peysakhov, Vlada Galinskaya, and William C. Regli. 2000. Representation and Evolution of Lego-based Assemblies. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. 1089.

Przemyslaw Prusinkiewicz. 1986. Graphical applications of L-systems. In *Proceedings of graphics interface*, Vol. 86. 247–253.

yong-fa Qin and zhi-gang Xu. 2007. Assembly Process Planning Using a Multi-objective Optimization Method (ICMA). In *Proceedings of the International Conference on Mechatronics and Automation*. 593–598.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*. 8748–8763.

Daniel Ritchie, Ben Mildenhall, Noah D. Goodman, and Pat Hanrahan. 2015. Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo. *ACM Trans. Graph. (SIGGRAPH 2015)* 34, 4 (2015), 11 pages.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. 2020. GRAF: generative radiance fields for 3D-aware image synthesis. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. 20154–20166.

Pedro Brandão Silva, Pascal Müller, Rafael Bidarra, and António Coelho. 2013. Node-based shape grammar representation and editing. *Proceedings of the workshop on Procedural Content Generation in Games (PCG)* (2013), 1–8.

Cem Sinanoğlu and H. Rıza Börklü. 2005. An assembly sequence-planning system for mechanical parts using neural network. *Assembly Automation* 25, 1 (2005), 38–52.

Eugene Smal. 2008. *Automated Brick Sculpture Construction*. Thesis. Stellenbosch University.

Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum* 33, 6 (2014), 31–50.

Ben Stephenson. 2016. A Multi-Phase Search Approach to the LEGO Construction Problem. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*. 89–97.

Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Trans. Graph. (SIGGRAPH 2011)* 30, 2 (2011), 1-14 pages.

Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. 2013. Automatic Generation of Constructible Brick Sculptures. In *Eurographics (short paper)*. 81–84.

Rylee Thompson, Ghalebi Elahe, Terrance DeVries, and Graham W Taylor. 2020. Building LEGO Using Deep Generative Models of Graphs. *Machine Learning for Engineering Modeling, Simulation, and Design Workshop at Neural Information Processing Systems* (2020).

Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D.D. Willis, and Wojciech Matusik. 2022. Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly. *ACM Trans. Graph. (SIGGRAPH Asia 2022)* 41, 6 (2022), 11 pages.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *European Conference on Computer Vision (ECCV)*. 55–71.

Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Chin-Yi Cheng, and Jiajun Wu. 2022a. Translating a Visual LEGO Manual to a Machine-Executable Plan. In *European Conference on Computer Vision (ECCV)*. 677–694.

Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Ran Zhang, Chin-Yi Cheng, and Jiajun Wu. 2022b. IKEA-Manual: Seeing Shape Assembly Step by Step. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. 28428–28440.

Karl D.D. Willis, 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15828–15839.

David V. Winkler. 2005. Automated Brick Layout. In *Proc. BrickFest*. 145–166.

Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. 2003. Instant architecture. *ACM Trans. Graph. (SIGGRAPH 2003)*, 669–677.

Hao Xu, Ka-Hei Hui, Chi-Wing Fu, and Hao Zhang. 2021. Computational LEGO Technic Design. *ACM Trans. Graph. (SIGGRAPH Asia 2019)* 38, 6, Article 196 (2021), 14 pages.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 3836–3847.

Jie Zhou, Xuejin Chen, and Ying-Qing Xu. 2019. Automatic generation of vivid LEGO architectural sculptures. In *Computer Graphics Forum*, Vol. 38. 31–42.

Mingjun Zhou, Jiahao Ge, Hao Xu, and Chi-Wing Fu. 2023. Computational Design of LEGO® Sketch Art. *ACM Trans. Graph. (SIGGRAPH Asia 2023)* 42, 6 (2023), 1–15.