Translating a Visual LEGO Manual to a Machine-Executable Plan

Ruocheng Wang[1], Yunzhi Zhang[1], Jiayuan Mao[2],
Chin-Yi Cheng[3⋆], and Jiajun Wu[1]

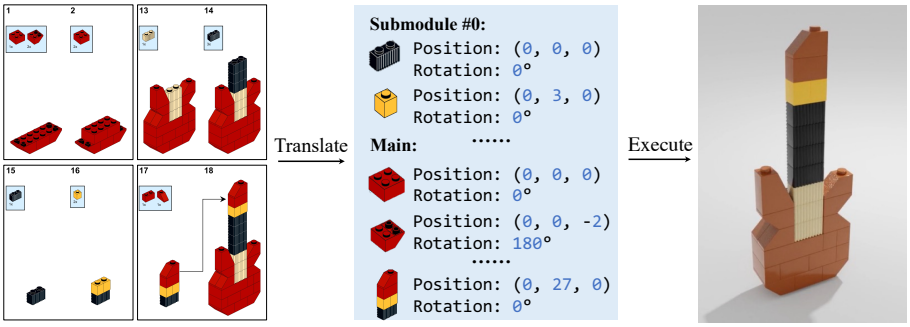
[1] Stanford University
[2] Massachusetts Institute of Technology
[3] Google Research

Abstract. We study the problem of translating an image-based, step-by-step assembly manual created by human designers into machine-interpretable instructions. We formulate this problem as a sequential prediction task: at each step, our model reads the manual, locates the components to be added to the current shape, and infers their 3D poses. This task poses the challenge of establishing a 2D-3D correspondence between the manual image and the real 3D object, and 3D pose estimation for unseen 3D objects, since a new component to be added in a step can be an object built from previous steps. To address these two challenges, we present a novel learning-based framework, the Manual-to-Executable-Plan Network (MEPNet), which reconstructs the assembly steps from a sequence of manual images. The key idea is to integrate neural 2D keypoint detection modules and 2D-3D projection algorithms for high-precision prediction and strong generalization to unseen components. The MEPNet outperforms existing methods on three newly collected LEGO manual datasets and a Minecraft house dataset.

1 Introduction

As a community, we would like to build machines that can assist humans in constructing and assembling complex objects, such as block worlds [7], LEGO models [9], and furniture [35]. The assembly task involves a sequence of actions that move different 3D parts to desired poses. Tackling such a long-horizon task with machines requires significant engineering effort [35]. On the other hand, humans usually rely on visual manuals to guide assembly procedures. These manuals are built by expert designers to decompose the task into a sequence of short steps that can be executed smoothly and efficiently. In this paper, we aim to facilitate the assembly tasks for machines by building a model that translates manuals into machine-interpretable plans. Fig. 1a shows an example of LEGO manuals that guides the user to build a guitar. Each step in the manual involves multiple building components presented in 2D images, and our goal is to extract the pose of each component in order to inform a downstream autonomous agent to execute the step to build the target object.

⋆ Work done when working at Autodesk AI Lab.



(a) LEGO Manual (b) Translated Machine-Executable Plan (c) LEGO Built From the Plan

Fig. 1: We study the problem of translating a LEGO manual to a machine-executable plan that can be executed to build the target shape. (a) Screenshots from an original LEGO manual. (b) A machine-executable plan generated by our model MEPNet. (c) LEGO built by executing the generated plan.

We identify two key challenges of interpreting visual manuals. First, it requires identifying the correspondence between a 2D manual image and the 3D geometric shapes of the building components. Since each manual image is the 2D projections of the desired 3D shape, understanding manuals requires machines to reason about the 3D orientations and alignments of components, possibly with the presence of occlusions.

The second challenge is the rich library of assembly components. Taking LEGO as an example, while most LEGO shapes can be built from a finite set of primitives, these primitives can be flexibly composed into more complex subparts that are added to the main body as a whole (e.g., the head of the guitar in Fig. 1) in a step. The compositionality of primitive bricks greatly increases the diversity of LEGO components, and as a result increases the difficulty for machines to interpret LEGO manuals: it requires inferring 3D poses of *unseen* objects composed of seen primitives.

In this work, we develop a method that tackles this challenging problem. Concretely, we formulate the problem of translating manuals into machine-executable plans as a sequential task. For each step, the inputs consist of 1) a set of primitive bricks and parts that have been built in previous steps represented in 3D, and 2) a target 2D image showing how components should be connected to each other. The expected output is the (relative) poses of all components involved in this step, as shown in Fig. 1b.

There are roughly two groups of solutions to parsing a single step of a manual. The first group is search-based methods (i.e., "analysis-by-synthesis") [45,3]. These methods make use of a given forward synthesis model, i.e., the underlying manual image renderer, for pose inference. They search over possible 3D poses of new components, render manual images based on the candidate poses, and select the pose that maximizes the matching score between the input and the rendered images. This approach is simple and accurate but assumes a given renderer. It is also computationally expensive as we need to search for the 3D poses of multiple components jointly in a single assembly step. The second group is learning-based,

using end-to-end neural networks that predict the 3D pose of each component. This approach does not require an image renderer and is fast, but typically suffers from poor generalization to unseen 3D shapes.

Inspired by these observations, we propose the Manual-to-Executable-Plan Network (MEPNet), a hybrid approach that combines the best of both worlds. The MEPNet has two stages. In the first stage, a convolutional neural network takes as input the current 3D LEGO shape, the 3D model of new components, and the 2D manual image of the target shape. It predicts a set of 2D keypoints and masks for each new component. In the second stage, 2D keypoints predicted in the first stage are back-projected to 3D by finding possible connections between the base shape and the new components. It also refines component orientation predictions by a local search. Our approach does not require the groundtruth image renderer during training or inference. Experiments show that our proposed approach maintains the efficiency of learning-based models, and generalizes better to unseen 3D components compared to end-to-end learning-based approaches.

We evaluate MEPNet on two benchmarks: one in the LEGO domain and another in a Minecraft-style house crafting domain [8]. Our results show that MEPNet enables more accurate pose estimation and, more importantly, generalizes better to unseen novel 3D components compared with several baselines. Most notably, we demonstrate that MEPNet is capable of generalizing to real-world LEGO manuals by training solely on synthetically generated manuals. We will release all code and data for full reproducibility.

2 Related Work

Parsing human-designed diagrams. A diagram is a fundamental and commonly-used tool for humans to communicate concepts and information [1]. There have been a number of works on parsing different types of diagrams like engineering drawings [13], cartographic road maps [26] and sewing patterns [2] into machine-understandable data. We focus on the task of parsing assembly manuals into instructions that can be executed by machines. Shao et al. [34] proposes a technique to parse assembly instructions into executable plans, but focuses manuals in vector-graphic formats, while we work on LEGO manuals in RGB image format with no known visual primitives like edges and polygons. Li et al. [23] proposes a method to parse the 3D poses of parts from a single RGB image, while our work focuses on sequentially inferring 3D poses from a series of manual images.

Inverse 3D modeling. Inferring the geometry procedures that reconstruct a 3D shape is useful in many domains like robotic assembly [35,19], shape synthesis [12,6,21] and computer-aided design [40,20]. A line of research focuses on using different geometry operations like poses of 3D primitive parts [37], shape programs [36,18], constructive solid geometry [10] and CAD operations [44]. Different kinds of information are studied to guide the inverse inference process: final 3D shape [27,16], single image [30,23] or multi-view images [15,9]. When it comes to image-guided inverse modeling, previous works often assume access to images of the final 3D shape, where the inverse problem is inherently ambiguous.

To tackle this issue, a shape prior is learned from a corpse of 3D shapes like ShapeNet [5], which can not be directly transferred to distribution different from the training data. Our work focuses on recovering the poses of LEGO bricks from a series of manual images that progressively specify building operations, which are intended to guide the assembly of diverse shapes of objects.

Pose estimation. To parse the information from manual images, we need to infer the poses of primitive parts to be assembled. Estimating the 3D poses of objects is a fundamental problem in 3D vision. A line of research uses convolutional neural networks to directly localize objects in a scene and regress their 6D poses [4,41,22,38]. Other works adopt a two-stage approach where 2D keypoints of objects are first extracted and then poses are inferred from them [33,31,32]. Most works focus on detecting objects from the same category of the training objects, while our work aims to build models that can estimate the pose of known primitives as well as novel shapes composed from them. Xiao et al. [43,42] proposes a CNN architecture to estimate the pose of a single object in the image with a known 3D model. On the other hand, assembly manuals often require estimating poses of multiple objects. Furthermore, to understand manuals, models need to ignore the parts that are assembled in previous steps and only detect parts of interest, which is not addressed in previous works.

3 Problem Formulation

Throughout the paper, we will be using LEGO manuals as our example, although many of the ideas generalize to other types of assembly manuals such as Minecraft and furniture. A LEGO manual is composed of a sequence of images. The images specify a step-by-step instruction sequence of adding new components to an existing LEGO shape (called the *base shape*, which is typically the target shape in the previous step). Each new component is either a primitive brick with specified type and quantities, (in which case the image will specify the type and the number of new bricks) or a pre-built component (called a *submodule*, such as the guitar head shown in Fig. 1). The main diagram in each image will specify the poses of the new components w.r.t. the *base shape*, by showing a 2D view of the target shape at this step.

Sequential manual parsing. We formulate the task as a sequential prediction task of T steps. At each step, a manual parser receives the following inputs.

1. A 3D representation of the *base shape* at this step. In this paper, we focus on a voxel-based representation V_i^{cur}. It might be an empty voxel grid when we start building a new shape. In LEGO manuals, this usually comes from the target shape of the previous step.
2. A set of components to be added at this step, where each component is either a primitive brick or a pre-built shape composed of multiple primitive bricks. In either cases, their shape will be represented as a set of 3D voxels V_{ij}^{new}, where $j = 1, 2, \cdots, C_i$ and C_i is the number of components for step i. In a LEGO manual image, these are usually specified in a small diagram at a corner as shown in Fig. 1a.

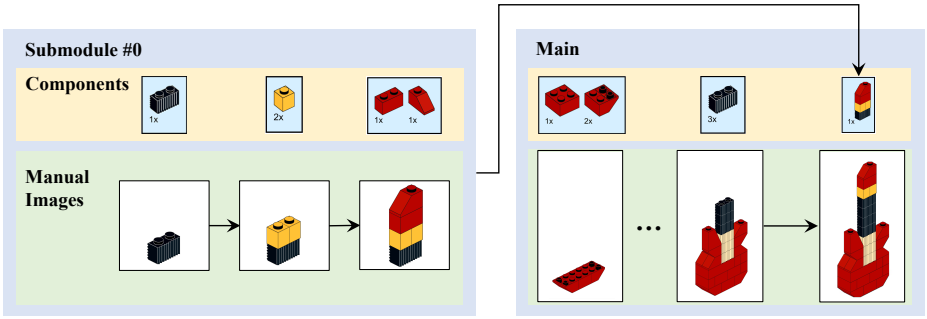


Fig. 2: Our factorized representation of LEGO manuals, which is a tree-structured plan specified by manual images. Each component is either a primitive brick or a submodule, and each submodule is recursively constructed from a sequence of manual images with corresponding components.

3. A 2D image I_i specifying the target shape after we compose all components at step i. In LEGO manuals, this corresponds to the main diagram of a manual image.

Our goal is to predict the 3D translation and rotation w.r.t. V_i^{cur} for each added component $\{(t_{ij}, r_{ij})\}_{j=1}^{C_i}$ for each step i, where $t_{ij} \in \mathbb{R}^3, r_{ij} \in \mathbb{R}^{3\times3}$.

Once we have a model that predicts 3D poses of components at each step, they serve as a plan that can be executed by machines to assemble a complete object (called a LEGO *set*) in an iterative manner. In general, the dependency between steps is a tree because of *submodules* in assembly—a step may depend on the resulting shapes from multiple previous steps, as shown in Fig. 2.

The connection constraint. An important feature of object assembly is the inherent connection constraints among object parts, such as the studs in LEGO shapes and the pegs and holes in furniture assembly. This enables designers to simply use a 2D image to specify how parts should be connected, in contrast to specifying the exact 3D dimensions and poses of objects. This constraint brings us benefits in both model design and evaluation. For example, in model design, it is generally hard to accurately infer the 3D continuous pose of an object from a 2D image, but it is easier to infer a set of discrete connections between objects. On the evaluation side, this allows us to reconstruct a physically plausible (stable and no inter-penetration) target shape by simulating all steps.

4 Manual-to-Executable-Plan Network

In this section, we will be focusing on developing a learning-based method for solving the one-step prediction task. It can be applied iteratively to each assembly step to reconstruct the full 3D shape. Recall that the input to our model is the base shape at this step V^{cur}, a set of new components, $\{V_j^{new}\}$, and a 2D image I (omitting the step index i for all variables for clarity). Our goal is to infer the 3D poses of all components $\{V_j^{new}\}$.

Our model, the Manual-to-Executable-Plan Network (MEPNet), consists of two stages. In the first stage, we use a neural pose estimation model to predict the

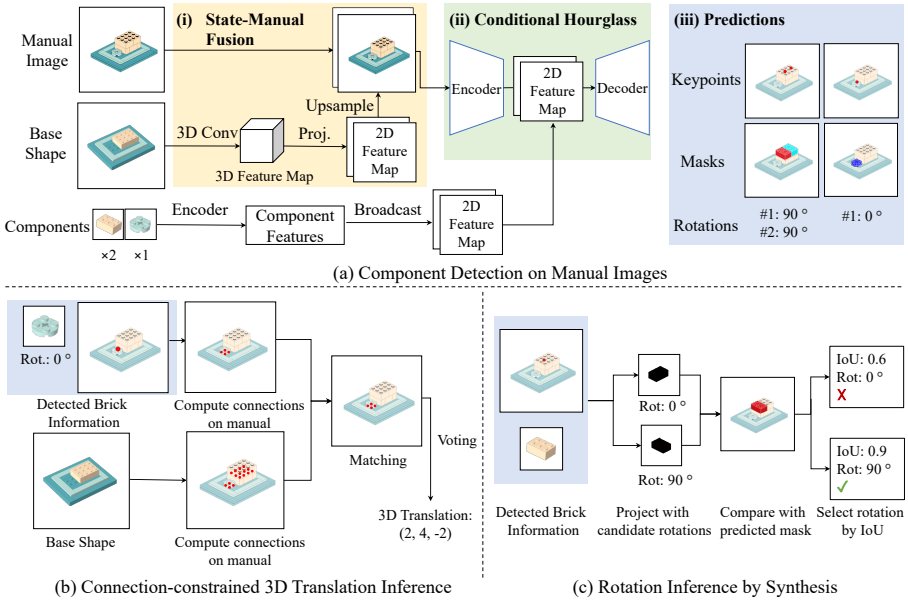


Fig. 3: MEPNet consists of two stages. (a) In the first stage, the model detects the 2D center keypoints, the masks, and the rotations for all added components on the manual. (b) and (c): We use a connection-constrained inference subroutine and an inference-by-synthesis subroutine to recover 3D poses of components from 2D information predicted in the first stage.

2D keypoint (corresponding to the center of the object), mask, and rotation for each new component to be added. In the second stage, we use two deterministic algorithms to infer the 3D poses of new components by fusing the predictions from the first stage.

Our two-stage approach is primarily motivated by the difficulty of directly estimating 3D poses from 2D images, especially considering the generalization to unseen components. Here, we leverage the connection constraint between components and the idea of analysis-by-synthesis to design post-processing algorithms that refine the raw prediction made by neural networks. In the experiment section, we will compare our two-stage approach with other alternatives, including a single-stage neural network baseline and an analysis-by-synthesis baseline based on mask predictions.

4.1 Neural Pose Estimation

Our neural module for estimating component poses follows an encoder-fusion-decoder pipeline. It combines convolution-based 3D shape encoders and a 2D Hourglass network—a state-of-the-art model for 2D keypoint and mask prediction. We will use separate 2D and 3D encoders to extract features for V^{cur}, V_j^{new}, and I, fuse the latent representations, and predict the center keypoint, mask, and rotation for each individual V_j^{new}.

Our first module, shown in Fig. 3a (i) is a state-manual encoder, which fuses the information of $V^{cur}$and the image I representing the target shape. Specifically, the state-manual encoder takes the voxel representation of V^{cur} as input, and extracts a 3D representation with two 3D Convolution-BatchNorm-ReLU modules. Let f_{3d} denote the output tensor of shape $\mathbb{R}^{C_1 \times H \times W \times D}$, where (H, W, D) is the 3D dimension of the input voxel and C_1 is the number of channels. Then, we use the camera parameters of the manual image to transform this voxel to the camera frame. This is computed by a differentiable spatial transformation based on [17]. Next, we project this voxel to the camera plane by rasterization, that is, for each pixel on the sensor plane, select the first non-empty voxel hit by a ray shooting from the sensor pixel towards the camera center. This gives us a 2D feature map $f^{2d} \in \mathbb{R}^{C_1 \times H \times W}$. Then the feature map is upsampled with bilinear interpolation to have the same resolution as the manual image I. Then we concatenate this feature map and the image along the feature channel to generate an "augmented" image representation, i.e., $I^a \in \mathbb{R}^{(C_1+3) \times H_{\text{img}} \times W_{\text{img}}}$.

Our second module, shown in Fig. 3a (ii) is a component-conditioned Hourglass model for predicting the center keypoint, mask, and rotation for each component. Specifically, for each new component V_j^{new}, also represented as a 3D voxel, we use five 3D Convolution-BatchNorm-ReLU layers, followed by an average pooling layer to extract the corresponding feature, denoted as $f_j^{new} \in \mathbb{R}^{C_2}$. Note that, for multiple components that have the same shape (w.r.t. SO(3)), we only encode one of them. We order all components based on their order in the input manual and get an sequence of component feature embeddings $\{f_j^{new}\}$. We concatenate all embeddings along the channel dimension into a vector f^{new*}, whose number of channels is $K \times C_2$, where K is the total number of distinct-shaped components, which we call "*component types*". We pad this vector by adding 0's into a vector of length $K_{max} \times C_2$, where $K_{max} = 5$ is the maximum number of distinct components considered in MEPNet.

Given the state-aware manual image I^a and component features $\{f_j^{new}\}$, we predict the 3D poses of the added components using an adapted implementation of stacked Hourglass Networks [29,46]. We first use a top-down network to process the I^a into a low-resolution 2D feature map I^{aenc} of resolution $(H_{\text{img}}/32, W_{\text{img}}/32)$. Next, we tile the concatenated component embeddings f^{new*} along spatial dimensions into a 2D feature map of resolution $(H_{\text{img}}/32, W_{\text{img}}/32)$, and concatenate this feature map to I^{aenc}. Then we use a bottom-up decoder network to output a high-resolution feature map of resolution $(H_{\text{img}}/4, W_{\text{img}}/4)$.

Then we use three separate small fully-convolutional neural networks to extract the center keypoint, mask, and rotation for each input component based on the feature map output by the Hourglass decoder.

1. For the center keypoint, we employ the structure of CenterNet [46]. For each component type, the model output is a tuple of three 1D feature maps: (h, dx, dy). h is a heatmap of centers, and dx and dy form a two-dimensional offset prediction which is the difference between the actual center of an object and the pixel location of the heatmap. This helps mitigate the discretization error caused by downsampling.

2. For the component mask, we employ the structure of Associative Embedding [28]. For each component type, the output is a tuple of two feature maps (m, emb), where m is a segmentation mask of the component type and emb is a 2D feature map where each pixel is associated with a vector embedding (called the "associative embeddings"). The L2 distance between pixels associated with different instances should be large, while the distance between pixels of the same instance should be small. To get the instance-level segmentation, we perform a pixel clustering based on emb.
3. For the rotation, our model output is a 4-dimensional vector, with a Softmax nonlinearity[†], since the component will only have rotations chosen from (0°, 90°, 180°, 270°) around the vertical axis in the LEGO problem we considered.

It is important to note that since we have merged components of the same shape into one component type, there will be multiple instances detected for each component type. For example, shown in Fig. 3a, the center keypoint heatmap for the first component type has two peaks, corresponding to two instances of this component type.

Training and losses. We train MEPNet with full supervision on a synthetically generated dataset where we have the groundtruth keypoint, mask, and rotation information. The entire neural network module is trained end-to-end with gradient descent. Our objective function is computed by

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{keypoint}} + \beta \cdot \mathcal{L}_{\text{mask}} + \gamma \cdot \mathcal{L}_{\text{rotation}}.$$

We adopt the keypoint loss adapted from [46]: $\mathcal{L}_{\text{keypoint}} = \mathcal{L}_{\text{heatmap}} + \mathcal{L}_{\text{offset}}$, where $\mathcal{L}_{\text{heatmap}}$ is a focal loss [24] computed based on the predicted heatmap and a groundtruth heatmap generated by Gaussian kernels. $\mathcal{L}_{\text{offset}}$ is an L1 loss for the regression task of dx and dy (the offsets). The mask loss is adapted from [28]: $\mathcal{L}_{\text{mask}} = \mathcal{L}_{\text{semantic}} + \mathcal{L}_{\text{pull}} + \mathcal{L}_{\text{push}}$, where $\mathcal{L}_{\text{semantic}}$ is a cross-entropy loss applied to the predicted mask, and $\mathcal{L}_{\text{pull}} + \mathcal{L}_{\text{push}}$ is the contrastive loss for learning the associative embeddings. Finally, we use a cross-entropy loss to train the rotation prediction module. More details are in the supplementary material.

4.2 3D Pose Inference

Based on the 2D predictions from the first stage, we infer 3D poses for each component. Here, we will exploit two important ideas: the connection constraint in assembly domains, and the idea of analysis-by-synthesis.

Connection-constrained 3D translation inference. Given the center keypoint of each component, our goal is to find the 3D XYZ position of the component. Here, we will rely on the connection constraints, that is, there should be at least one position where the new component is attached to another existing brick. In LEGO, the attachment is achieved by inserting a "stud" of the existing brick into an "anti-stud" of the new component. As illustrated in Fig. 3b, our inference has three steps. First, given the center keypoint of the new component, we infer the 2D location of all anti-studs of the new component (this process is deterministic

[†] In the implementation we used a slightly more complex scheme to handle symmetries. Details are included in the supplementary material.

and does not require any depth information because images in LEGO manuals are orthographically projected, see a detailed proof in the supplementary material). Next, we also project all studs in the base shape V^{cur} onto the same 2D plane. Finally, we perform a matching between all possible studs and anti-studs followed by a majority voting. Then we can predict the 3D position of the new component based on the 3D position of existing studs, which is known.

Rotation inference by synthesis. We empirically found that directly predicting the rotation of a *submodule* is hard. Thus, instead of using the rotation predicted from Section 4.1, we employ an analysis-by-synthesis process to estimate the rotation. Illustrated in Fig. 3c, for each *submodule*, we compute all possible translations for each of the 4 rotations based on connection constraints. Then for each rotation-translation candidate (r, t), we project the component with pose (r, t) to the image, obtain a mask and compute its Intersection-over-Union (IoU) with the predicted mask from Section 4.1. We select the pose (r, t) with the highest IoU score as the final pose prediction.

4.3 Implementation Details

LEGO discretization. To enable flexible attachments between different LEGO bricks, most LEGO bricks have sizes that are the multiples of the smallest $1 \times 1 \times 1$ brick and thus are inherently voxelized. In this work, we voxelize the basic $1 \times 1 \times 1$ LEGO brick as a $2 \times 2 \times 2$ voxel grid. We double the resolution of the voxel grid because the 3D translation of a brick can be half the size of the brick.

Center keypoint. We observe that although the components of interest can be severely occluded in a manual image, their top faces often remain visible. This is a consistent design pattern across many LEGO manuals [14]. Thus, the center keypoint of a primitive brick is defined as the center of the top surface of a LEGO brick, as illustrated in Fig. 3a (iii). For submodules, we define the center keypoint to be the center keypoint of the topmost primitive brick in the submodule. If there are multiple topmost primitive bricks, we use a randomly selected brick during training and use a modified 3D pose inference algorithm during evaluation. Details are in supplementary materials.

Camera projection. Same as the real-world LEGO manuals, we model the camera projection as a weak perspective (scaled orthographic) transformation. Thus, the camera is parameterized by scale $s \in \mathbb{R}$, translation $t \in \mathbb{R}^2$, and rotation r represented by three Euler angles. In this paper, we assume known camera parameters for all methods including baselines. The camera parameters are predicted by a pretrained pose estimation model [43] which takes the input base shape and manual image as input. Details are in supplementary materials.

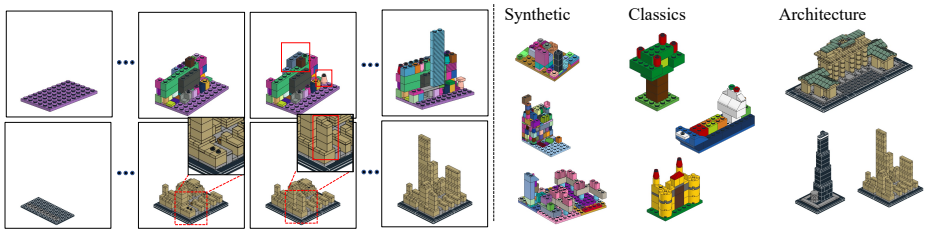
5 Experiments

We evaluate MEPNet in two assembly domains: LEGO and 3D-Craft [7].

5.1 Setup

Baselines. We compare MEPNet with two baselines.

PartAssembly [23] is a two-stage method designed for inferring object part poses from a single image. It first predicts the masks for individual components.



(a) Examples of manual images to build a LEGO set. (b) Examples of complete LEGO sets from each dataset.

Fig. 4: Example manual images and shapes in our LEGO datasets.

Then, it encodes the feature for each component and predicts the pose. We have made the following adaptations to it for our sequential prediction setting. First, to incorporate information about the current object state, we replaced the input image with the state-augmented manual image I^a using the same encoder as MEPNet. Second, we add CoordConv [25] to the model to enhance its prediction about 3D positions. Third, we add a post-processing procedure to the model prediction that quantizes the prediction based on the connection constraints. Finally, we also replaced their original point cloud encoder with our voxel-based encoder. More details can be found in the supplementary materials.

Direct3D, the second baseline, is an ablative variant of MEPNet, in which we directly predict the 3D translation and rotation for each instance (instead of predicting keypoints, masks, and rotations). We also use CoordConv and prediction quantization with connection constraints in this model.

Metrics. We evaluate MEPNet and baselines at three different levels of granularity: componentwise, stepwise and setwise. In the componentwise and the stepwise case, the input to each model is the ground truth voxel grid V^{cur} and submodules (if any). We evaluate the *3D pose accuracy* (correct or incorrect because we have quantized the model predictions using the connection constraint) and the *Chamfer distance* [11]. To account for the rotation symmetry of components, we restricted the set of possible rotations based on component shapes. For stepwise pose accuracy, we say a step is correct if the predictions for all components in this step are correct. To compute Chamfer distance, following [39], we uniformly sample 10000 points from their meshes. For stepwise Chamfer distance, we compute the metric between the union of all components in a step.

In the setwise case, each model will be run on all manual images sequentially, and auto-regressively. That is, the predicted target shape from the first step will be the input for the second step. Submodules will be built by models as well. We compute two metrics: the Chamfer distance between the ground truth final shape and the shape output by each model. We also compute a normalized *Mistakes to Complete (MTC)* score, proposed in [7]. MTC computes the average percentage of steps where a model gives wrong poses predictions. In this case, the model will be fed with the ground truth V^{cur} and submodules.

5.2 Results on LEGO

Dataset. Our first dataset is a synthetic LEGO dataset that is procedurally generated based on 72 types of primitive bricks, and rendered using a standard

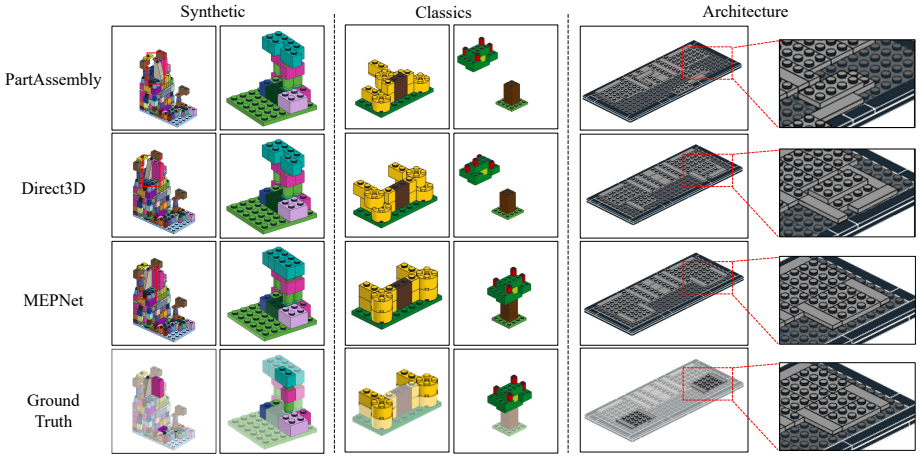


Fig. 5: Qualitative Results on the LEGO datasets. Each column contains the ground truth and the predictions from models for a single step. Components added in the step are highlighted in the manual images. To have a straightforward comparison between different models, we render their predictions in the same way as rendering the target manual.

		Componentwise		Stepwise		Setwise	
		Pose Acc ↑ (%)	CD ↓	Pose Acc ↑ (%)	CD ↓	CD ↓	MTC ↓ (%)
Synthetic	PartAssembly [23]	63.74	393.21	47.05	226.48	321.91	52.72
	Direct3D	88.51	32.60	77.39	5.08	9.18	25.62
	MEPNet	**96.96**	**11.60**	**93.41**	**1.93**	**4.74**	**8.63**
Classics	PartAssembly [23]	2.26	1171.57	0.00	296.82	775.29	100.00
	Direct3D	34.84	303.56	24.27	3.00	67.75	80.77
	MEPNet	**88.69**	**72.79**	**90.29**	**0.10**	**15.52**	**13.97**
Architecture	PartAssembly [23]	5.12	1964.31	3.24	858.58	719.88	96.13
	Direct3D	13.71	936.77	23.17	227.41	191.28	87.79
	MEPNet	**83.47**	**136.40**	**82.23**	**15.35**	**107.95**	**16.00**

Table 1: Quantitative results of models on the three LEGO datasets. Chamfer distance metrics are multiplied by a factor of 10^5. MEPNet outperforms baselines in all metrics on the three datasets.

LEGO manual renderer[‡]. Our data generation pipeline encapsulates two features of real-world LEGO manuals: 1) using submodules to assemble two components that have been built separately, and 2) stacking or tiling multiple bricks of the same shape at one step to form structures such as walls and floors. Examples of generated LEGO objects are shown in Fig. 4(b). Details of the generation pipeline, attribution of assets and their license can be found in the supplementary materials. We generate 8000 manuals for training, 10 sets for validation, and 20 sets for testing. In sum, there are 200K individual steps in training, 300 for the validation split, and 600 for the test split.

[‡] `https://trevorsandy.github.io/lpub3d/`

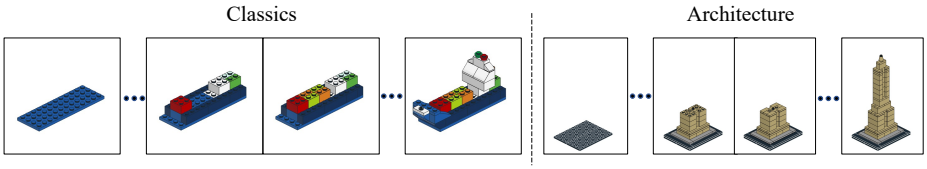


Fig. 6: Qualitative Results of MEPNet building LEGOs from scratch. We visualize several intermediate steps and final results by rendering them in the same way as the manual images. MEPNet can successfully parse manuals into executable plans with diverse target shapes.

We have also collected two datasets from real-world LEGO manuals. We select 11 sets from LEGO's Classics theme, which contains simple objects designed for children older than 4, and 5 sets from LEGO's Architecture theme, which contains more complex building-shaped LEGOs for kids older than 10. There are around 200 individual steps in each dataset. We manually exclude or replace bricks that are not in our primitive brick set, and re-render the LEGO manuals with the factorized representation. Examples of our datasets are shown in Fig. 4.

Results on the synthetic dataset. Quantitative results on the synthetic dataset are summarized in Table 1. MEPNet outperforms baseline models in all metrics considered. From visualizations in Fig. 5, we can see MEPNet is able to accurately predict 3D poses of both primitive bricks and submodules, even in cases with significant occlusions. Baseline models tend to either have a small deviation in 3D translation, or fail to infer the orientation of submodules. We also find that PartAssembly fails to predict accurate masks for target components, which are critical in its pose estimation.

Generalization to the Classics and the Architecture datasets. We directly apply models trained on our synthetically generated datasets on the Classics and the Architecture datasets. Illustrated in Table 1, there is a large performance drop for PartAssembly and Direct3D due to the distribution mismatch between these two datasets and our synthetic training dataset, while MEPNet maintains high accuracy. We found that the Direct3D model can still predict accurate keypoints on these two datasets, but fails to accurately predict their 3D poses. This shows the effectiveness of our two-stage inference procedure. Noticeably, MEPNet can successfully build sets with diverse shapes guided by manuals as illustrated in Fig. 6, although they look drastically different from the synthetic dataset and contain submodules of unseen shapes. We present several examples of the full assembly process in the supplementary materials.

The architecture dataset is the most challenging one because it contains a large number of primitive bricks and severe occlusion. MEPNet still achieves a decent prediction accuracy while both baselines fail significantly.

Ablation: submodules. We perform an ablation study comparing how different models handle submodules, as shown in Table 2. Specifically, we evaluate the componentwise pose prediction accuracy and Chamfer distance across all steps that involve submodules. The table indicates the challenge of 3D pose inference for submodules. Our rotation-inference-by-synthesis (RS) consistently improves the results across all three datasets.

	Synthetic		Classics		Architecture	
	Pose Acc ↑ (%)	CD ↓	Pose Acc ↑ (%)	CD ↓	Pose Acc ↑ (%)	CD ↓
PartAssembly [23]	0.00	1754.64	0.00	2107.11	0.00	4816.21
Direct3D	22.75	161.59	10.00	180.86	0.00	1539.04
MEPNet (w.o. RS)	22.27	192.93	20.00	25.17	23.08	**256.55**
MEPNet	**75.83**	**23.27**	**90.00**	**0.25**	**38.46**	271.35

Table 2: Quantitative results on the submodules exclusively. Chamfer distance metrics are multiplied by a factor of 10^5. Rotation inference by synthesis (RS) plays an important role in inferring the pose of submodules.

	Classics			Architecture		
	Pose Acc ↑ (%)	Time ↓ (s)	TLE ↓ (%)	Pose Acc ↑ (%)	Time ↓ (s)	TLE ↓ (%)
Direct3D	34.84	**0.233**	**0**	13.71	**0.320**	**0**
MEPNet (a-by-s)	48.42	10.94	0.9	34.14	70.77	11.65
MEPNet	**88.69**	0.311	**0**	**83.47**	0.406	**0**

Table 3: Comparison with an ablation model based purely on analysis-by-synthesis. We show the component pose accuracy and the average inference time per step (which may include multiple components). TLE (Time-Limit-Exceeded) measures the percentage of components whose prediction cannot terminate in 1 minute.

Ablation: pure analysis-by-synthesis. We also perform an ablation study of the second stage algorithm by replacing it with an algorithm that is entirely based on analysis-by-synthesis, for both primitive bricks and submodules. In contrast, our full model MEPNet only applies analysis-by-synthesis for the rotation inference of submodules. Since we do not assume access to the underlying image renderer, we will perform analysis-by-synthesis based on the 2D mask of the manual image.

Specifically, we use the mask prediction from our Hourglass model. Our goal is to set the 3D pose for new components to maximize the matching score between detected 2D masks and the 2D projections of these new components. To avoid the exponential scaling with respect to the number of components, we employed a sequential greedy algorithm. Given the current shape and a set of new components to be added, we iteratively search the pose for each new component. For each component, we enumerate all possible poses, and select the pose that minimizes the IoU between the projected mask and segmentation mask of that component type predicted by the Hourglass model. We set the maximum searching time for each component to 1 minute.

The accuracy and runtime of different models are summarized in Table 3. Direct3D is the fastest because 3D poses are directly predicted by the Hourglass model, but the performance is significantly worse than other methods. The analysis-by-synthesis baseline outperforms the Direct3D method, but it runs significantly slower. Our full model, MEPNet, performs the best. We empirically attribute the inferior performance of the analysis-by-synthesis variant to the imprecise prediction of component masks, especially for small primitive bricks.

5.3 Results on 3D-Craft

We also evaluate MEPNet on building Minecraft houses from 3D-Craft [7].

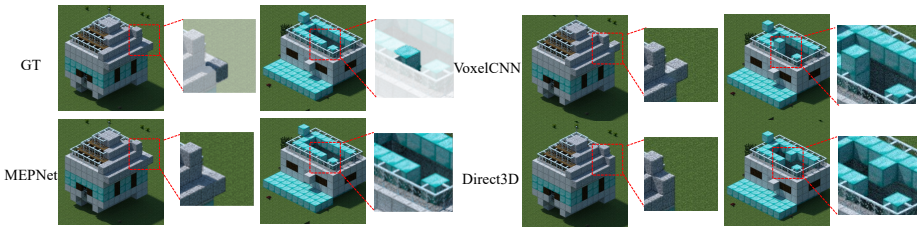


Fig. 7: Qualitative results on the 3D-Craft dataset. Ground truth and predictions of new bricks are zoomed in. Predictions of baseline models often lie in a small neighborhood of the ground truth positions, while our model is more accurate.

Data. The original 3D-Craft contains houses built with equally-sized bricks from crowdworkers in the format of building operation sequence. We leverage this sequence information to generate manuals. Each step in the manuals contains only one new brick. And our goal is to predict the 3D translation of the new brick in each manual image. We select 80 houses (20k steps) for training, 5 (1.5k steps) for validation, and 5 (1.5k steps) for testing. As the original dataset has a heavily imbalanced distribution of brick types, we select 5 frequently-used types in the dataset. In general houses in 3D-Craft contains more bricks than LEGO, and the appearance of bricks will be affected by lighting.
Setup. To migrate MEPNet to this setting, we use one-hot embedding to encode different types of bricks occupying each voxel. Because there is no rotation involved, we also remove the rotation prediction modules for all methods. Finally, we modify the 2D-to-3D algorithm according to Minecraft's connection constraints. Details are in the supplementary material.
Results. MEPNet achieves translation accuracy of 86.3% on the 3D-Craft dataset, while VoxelCNN and Direct3D only achieve 49.8% and 75.8%. Based on the visualizations shown in Fig. 7, we can also see that MEPNet can yield more accurate results across all different lighting conditions.

6 Discussion

We study the problem of translating an image-based, step-by-step assembly manual created by human designers into machine-interpretable instructions. We propose Manual-to-Executable-Plan Network (MEPNet), a model that reconstructs the assembly steps from a sequence of manual images. The key idea behind our model is to combine learning-based methods and inference-by-synthesis algorithms to wire in the connection constraints in assembly domains. Results show that our model outperforms existing methods on three newly collected LEGO manual datasets and a Minecraft house dataset.

Acknowldegements: We thank Joy Hsu, Chengshu Li, and Samuel Clarke for detailed feedback on the paper. This work is partly supported by Autodesk, the Stanford Institute for Human Centered AI (HAI), the Stanford Center for Integrated Facility Engineering (CIFE), ARMY MURI grant W911NF-15-1-0479, NSF CCRI #2120095, the Samsung Global Research Outreach (GRO) Program, and Amazon, Analog, Bosch, IBM, Meta, and Salesforce.

References

1. Agrawala, M., Li, W., Berthouzoz, F.: Design principles for visual communication. Communications of the ACM **54**(4), 60–69 (2011)
2. Berthouzoz, F., Garg, A., Kaufman, D.M., Grinspun, E., Agrawala, M.: Parsing sewing patterns into 3d garments. ACM TOG **32**(4), 1–12 (2013)
3. Bever, T.G., Poeppel, D.: Analysis by synthesis: a (re-) emerging program of research for language and vision. Biolinguistics **4**(2-3), 174–200 (2010)
4. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: ECCV (2014)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. arXiv:1512.03012 (2015)
6. Chaudhuri, S., Kalogerakis, E., Guibas, L., Koltun, V.: Probabilistic reasoning for assembly-based 3d modeling. ACM TOG **30**(4), 35 (2011)
7. Chen, Z., Guo, D., Xiao, T., Xie, S., Chen, X., Yu, H., Gray, J., Srinet, K., Fan, H., Ma, J., et al.: Order-aware generative modeling using the 3d-craft dataset. In: ICCV (2019)
8. Chu, H., Wang, S., Urtasun, R., Fidler, S.: Housecraft: Building houses from rental ads and street views. In: ECCV (2016)
9. Chung, H., Kim, J., Knyazev, B., Lee, J., Taylor, G.W., Park, J., Cho, M.: Brick-by-brick: Combinatorial construction with deep reinforcement learning. In: NeurIPS (2021)
10. Du, T., Inala, J.P., Pu, Y., Spielberg, A., Schulz, A., Rus, D., Solar-Lezama, A., Matusik, W.: Inversecsg: Automatic conversion of 3d models to csg trees. ACM TOG **37**(6), 1–16 (2018)
11. Fan, H., Su, H., Guibas, L.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)
12. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM TOG **23**(3), 652–663 (2004)
13. Haralick, R.M., Queeney, D.: Understanding engineering drawings. Computer Graphics and Image Processing **20**(3), 244–258 (1982)
14. Heiser, J., Phan, D., Agrawala, M., Tversky, B., Hanrahan, P.: Identification and validation of cognitive design principles for automated generation of assembly instructions. In: Proceedings of the working conference on Advanced Visual Interfaces. pp. 311–319 (2004)
15. van den Hengel, A., Russell, C., Dick, A., Bastian, J., Pooley, D., Fleming, L., Agapito, L.: Part-based modelling of compound scenes from images. In: CVPR (2015)
16. Huang, J., Zhan, G., Fan, Q., Mo, K., Shao, L., Chen, B., Guibas, L., Dong, H.: Generative 3d part assembly via dynamic graph learning. In: NeurIPS (2020)
17. Jaderberg, M., Simonyan, K., Zisserman, A.: Spatial transformer networks. In: NeurIPS (2015)
18. Jones, R.K., Barton, T., Xu, X., Wang, K., Jiang, E., Guerrero, P., Mitra, N.J., Ritchie, D.: Shapeassembly: Learning to generate programs for 3d shape structure synthesis. ACM TOG **39**(6), 1–20 (2020)
19. Lee, Y., Hu, E.S., Lim, J.J.: Ikea furniture assembly environment for long-horizon complex manipulation tasks. In: ICRA (2021)
20. Li, C., Pan, H., Bousseau, A., Mitra, N.J.: Sketch2cad: Sequential cad modeling by sketching in context. ACM TOG **39**(6), 1–14 (2020)

21. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.: Grass: Generative recursive autoencoders for shape structures. In: SIGGRAPH (2017)
22. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV (2018)
23. Li, Y., Mo, K., Shao, L., Sung, M., Guibas, L.: Learning 3d part assembly from a single image. In: ECCV (2020)
24. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
25. Liu, R., Lehman, J., Molino, P., Such, F.P., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. arXiv:1807.03247 (2018)
26. Mena, J.B.: State of the art on automatic road extraction for gis update: a novel classification. Pattern recognition letters **24**(16), 3037–3058 (2003)
27. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N.J., Guibas, L.J.: Structurenet: hierarchical graph networks for 3d shape generation. ACM TOG **38**(6), 1–19 (2019)
28. Newell, A., Huang, Z., Deng, J.: Associative embedding: End-to-end learning for joint detection and grouping. In: NeurIPS (2017)
29. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV (2016)
30. Niu, C., Li, J., Xu, K.: Im2struct: Recovering 3d shape structure from a single rgb image. In: CVPR (2018)
31. Oberweger, M., Rad, M., Lepetit, V.: Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In: ECCV (2018)
32. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6dof pose estimation. In: CVPR (2019)
33. Rad, M., Lepetit, V.: Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: CVPR (2017)
34. Shao, T., Li, D., Rong, Y., Zheng, C., Zhou, K.: Dynamic furniture modeling through assembly instructions. In: ACM TOG. vol. 35. Association for Computing Machinery (2016)
35. Suárez-Ruiz, F., Zhou, X., Pham, Q.C.: Can robots assemble an ikea chair? Science robotics **3**(17) (2018)
36. Tian, Y., Luo, A., Sun, X., Ellis, K., Freeman, W.T., Tenenbaum, J.B., Wu, J.: Learning to infer and execute 3d shape programs. In: International Conference on Learning Representations (2018)
37. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: CVPR (2017)
38. Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., Savarese, S.: Densefusion: 6d object pose estimation by iterative dense fusion. In: CVPR (2019)
39. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. arXiv:1804.01654 (2018)
40. Willis, K.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W.: Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. ACM TOG **40**(4), 1–24 (2021)
41. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In: RSS (2018)
42. Xiao, Y., Marlet, R.: Few-shot object detection and viewpoint estimation for objects in the wild. In: ECCV (2020)

43. Xiao, Y., Qiu, X., Langlois, P.A., Aubry, M., Marlet, R.: Pose from shape: Deep pose estimation for arbitrary 3d objects. In: BMVC (2019)
44. Xu, X., Peng, W., Cheng, C.Y., Willis, K.D., Ritchie, D.: Inferring cad modeling sequences using zone graphs. In: CVPR (2021)
45. Yuille, A., Kersten, D.: Vision as bayesian inference: analysis by synthesis? TiCS **10**(7), 301–308 (2006)
46. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)