Neural Assembler: Learning to Generate Fine-Grained Robotic Assembly Instructions from Multi-View Images

Hongyu Yan, Yadong Mu*

Wangxuan Institute of Computer Technology, Peking University
{pku_wdyhy, myd}@pku.edu.cn

Abstract

Image-guided object assembly represents a burgeoning research topic in computer vision. This paper introduces a novel task: translating multi-view images of a structural 3D model (for example, one constructed with building blocks drawn from a 3D-object library) into a detailed sequence of assembly instructions executable by a robotic arm. Fed with multi-view images of the target 3D model for replication, the model designed for this task must address several sub-tasks, including recognizing individual components used in constructing the 3D model, estimating the geometric pose of each component, and deducing a feasible assembly order adhering to physical rules. Establishing accurate 2D-3D correspondence between multi-view images and 3D objects is technically challenging. To tackle this, we propose an end-to-end model known as the Neural Assembler. This model learns an object graph where each vertex represents recognized components from the images, and the edges specify the topology of the 3D model, enabling the derivation of an assembly plan. We establish benchmarks for this task and conduct comprehensive empirical evaluations of Neural Assembler and alternative solutions. Our experiments clearly demonstrate the superiority of Neural Assembler.

Introduction

The assembly task necessitates predicting a sequence of operations for the placement of various components. Accurate and efficient assembly algorithms play a pivotal role in robotics. These assembly challenges are pervasive in daily life, as in scenarios like constructing LEGO models (Chung et al. 2021), assembling furniture (Suárez-Ruiz, Zhou, and Pham 2018), and Minecraft (Chen et al. 2019). In previous research, Chen et al. (2019) suggested replicating human building order with spatial awareness to construct Minecraft houses without target information. Wang et al. (2022a) introduced a step-by-step approach for assembling LEGO models based on the assembly manual, while the work Zhan et al. (2020) focused on predicting the 6-DoF pose of each component based on the object's class for assembly. Li et al. (2020) predicted the poses of parts using a single image.

In this study, we define a new task of image-guided assembly. We are provided with a set of multi-view images captured from a 3-D model, assuming it is built with components from a pre-specified library. The goal of the task is to generate a sequence of fine-grained assembly instructions, encompassing all parameters—such as component types, geometric poses of each component, and assembly order—in accordance with physical rules and suitable for execution by a robotic arm. The task serves as a valuable testbed for advancing vision-guided autonomous systems, presenting a range of technical challenges. Firstly, understanding the correspondence between 2D images and 3D objects is crucial. Given that certain components in the 3D model might be entirely obscured from specific viewpoints, we employ multi-view images (e.g., typically 4 in this study) as input. The algorithm must effectively integrate information from images captured from multiple perspectives. Secondly, estimating critical information for each component is non-trivial. With a 3D library in place, the algorithm needs to segment components across all images and categorize each based on predefined object types in the library, mainly using shape and texture cues. In addition, one also needs estimate the 3-D spatial position and rotation matrix of each component. Thirdly, obtaining typological information for all components is necessary to formulate a physically feasible assembly plan.

Importantly, observations of a component in images are often incomplete, primarily due to frequent occlusions. This poses a substantial challenge in fully understanding and interpreting the scene. Such occlusions are particularly challenging when assembling complex, multi-layerd models. For this novel task, we propose an end-to-end neural network, dubbed as Neural Assembler. The computational pipeline of Neural Assembler is illustrated in Figure 1. Taking multi-view images and a 3-D component library as input, Neural Assembler not only identifies each component from images but also determines its 3D pose at each step of assembly. Leveraging images from multiple viewpoints, our method aims to enhance the overall scene understanding and accurately predict the order for placing parts in assembly tasks.

We present two datasets for the proposed image-guided assembly task, namely the CLEVR-Assembly Dataset and LEGO-Assembly Dataset. Each sample in these datasets comprises images of the object captured from various perspectives, along with the pose of each part (2D keypoints,

*Corresponding Author.

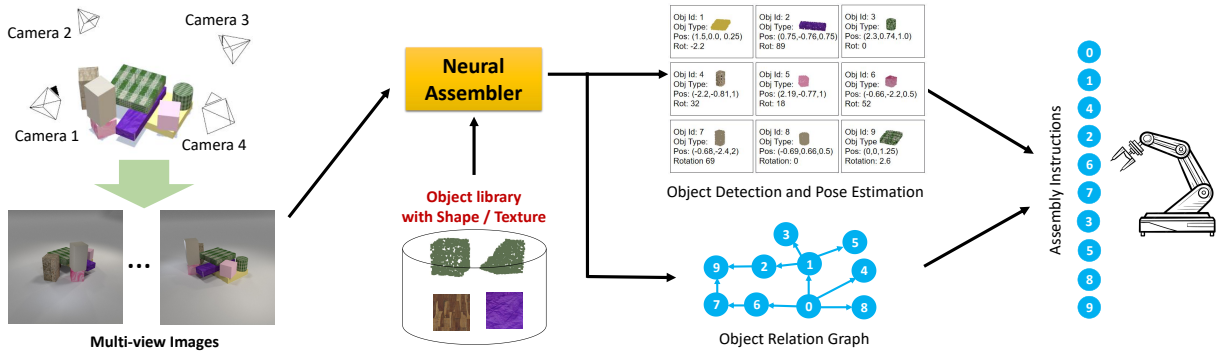

Figure 1: Schematic illustration of the proposed Neural Assembler. See Section 3 for more details.

mask, 3D position, and rotation), and the relationship graph of all components. Comprehensive experiments are conducted on both datasets. Due to the absence of prior work addressing this novel setting like Neural Assembler, we establish two robust baselines for comparison. The evaluations unequivocally demonstrate that Neural Assembler outperforms the baselines across all performance metrics.

Related Work

Assembly Algorithms

Assembly tasks have utilized computational strategies for part selection and placement, including action scoring (Yuille and Kersten 2006; Bever and Poeppel 2010), genetic algorithms (Lee et al. 2015), and voxel-based optimization (Kim et al. 2020; van den Hengel et al. 2015). Manual-driven approaches(Shao et al. 2016; Wang et al. 2022a; Chen et al. 2019; Walsman et al. 2022; Chung et al. 2021) have also been explored. LSTM Models(Walsman et al. 2022) have exhibited limitations in long sequence predictions ,and the reinforcement learning based method Chung et al. (2021) have struggled with the variety of components. Existing research into part-based 3D modeling (Zhan et al. 2020; Mo et al. 2019; Li et al. 2020; Niu, Li, and Xu 2018) and parts retrieval from 3D meshes (Chaudhuri and Koltun 2010; Shen et al. 2012; Sung et al. 2017) assumes prior part knowledge. Our novel task of reconstructing 3D models from multi-view images without prior information presents a unique challenge not yet addressed by none of above works.

Multi-View Scene Understanding

Many scene understanding tasks require multi-view inputs as single-view approaches often prove intractable. For instance, SLAM (simultaneous localization and mapping) necessitates reconstructing the 3D scene geometry and estimating camera poses from a sequence of video frames. 3D-SIS (Hou, Dai, and Nießner 2019) utilizes RGB-D images for 3D object detection and segmentation, while MVPointNet (Jaritz, Gu, and Su 2019) combines RGB-D images and point clouds for enhanced feature extraction. Murez et al. (2020) conducted 3D semantic segmentation and reconstruction using multiple RGB images. Similarly, our work involves understanding the structure of 3D brick models and predicting part poses from multi-view images.

Structural Modeling for Scenes or 3-D Models

Recent studies have focused on inferring structural representations like graphs (Johnson et al. 2015; Cong, Yang, and Rosenhahn 2023; Li, Zhang, and He 2022) and programmatic descriptions (Ellis et al. 2018; Liu and Wu 2019; Wu et al. 2017) from images. Techniques range from encoder-decoder architectures for triplet generation (Cong, Yang, and Rosenhahn 2023) to efficient graph-based methods (Li, Zhang, and He 2022), CNNs for primitive shape programs (Ellis et al. 2018). Additionally, an encoder-decoder model for scene re-rendering (Wu et al. 2017) and transformers for structural change detection (Qiu et al. 2023) have been explored. Our work differs by using structural representations to predict assembly sequences of object parts.

Method

Task Specification

Given several multi-perspective images of a 3-D brick model $\{I_k\}_{k=1}^{K}$, the corresponding camera parameters with respect to some reference coordinate system, and a predefined 3-D brick (or termed as component, as used exchangeably in this paper) library $Lib = \{b_1, b_2, \ldots, b_M\}$, our algorithm identifies the bricks present in the scene, predicts each brick's pose and constructs a relational graph $G = \{V, E\}$. The vertex set V correspond to the bricks and the directed edge set E represent spatial configuration that can further be used to derive the assembly instructions. In particular, each brick b_i in the library is denoted as $(S_i,\ T_i)$ where S_i is assumed to be the point cloud and T_i is represented as the texture image. In the relationship graph, each node $v_i \in V$ describes the i-th brick information $v_i = (S_i, T_i, Kps_i, Rot_i, M_i)$ where $Kps_i \in ([0,1] \times [0,1])^{K\times 2}$ encodes the 2D keypoints (we use this notation to refer to the planar projection of the brick center) in K views, $Rot_i \in [0,\ 2\pi]^K$ are the rotation angles in K views and $M_i \in \{0,1\}^{K\times H\times W}$ are the binary masks of the brick in K views. The edge $e_{i,j} \in E$ explicitly describes the assembly order, where $e_{i,j} = 1$ only when a brick v_j is placed into the 3-D model after brick v_i.

Here we develop a baseline for the proposed task, namely Neural Assembler, whose computational pipeline is shown in Figure 2, Our model leverages a graph convolutional network (GCN) to delineate the assembly sequence effectively. The 3-D poses of objects are inferred from 2D image data across multiple views, exploiting the geometric constraints provided by the camera parameters to ensure spatial consistency in the 3D domain. This framework showcases the capability of capturing complex relational patterns and translating them into a structured assembly protocol.

Multi-View Feature Fusion

We adopt the pretrained CLIP image encoder to get the feature maps F^k and CLIP feature vectors v^k_{CLIP}, $k = 1, 2, ...K$. Then the features of all views are fused with others, using an adapted implementation of Group-based Semantic Agreement Transformer (Xu and Mu 2023). The scene consensus (denoted as a vector g) of the images from different perspectives are extracted via mean-pooling $g = \frac{1}{K}\sum_{k=1}^{K} Norm(v^k_{CLIP})$, where $Norm(\cdot)$ is the L_2 normalization. Afterwards, the scene consensus g is dispersed to multi-view image features through channel-wise multiplication

$$\hat{F}^k = F^k \cdot g, \quad k = 1, 2, \ldots, K. \tag{1}$$

Template As Visual Prompts. Let the texture image $T_i \in \mathbb{R}^{H_{T_i} \times W_{T_i} \times 3}$, where $(H_{T_i},\ W_{T_i})$ is the texture image size, represent the texture template and point clouds $S_i \in \mathbb{R}^{N_P \times 3}$ represent the shape template. In all experiments of this work, we fix the parameter N_P to be all 1024, striking a balance between compute expense and capability of representing the bricks. For the texture image, we set both H_{T_i} and W_{T_i} to 224. A CNN backbone (*e.g.*, ResNet-18 (He et al. 2016)) generates template features $T_i \leftarrow CNN(T_i)$, and a pointnet backbone (*e.g.*, PointNet (Qi et al. 2017)) generates shape features $S_i \leftarrow PointNet(S_i)$.

Transformer Decoder. The decoder accepts as input the shape feature library $\mathbf{S} = \{S_i\}$, the texture feature library $\mathbf{T} = \{T_i\}$ and the fused image features $\hat{F}^k$. As shown in Fig. 2, the transformer decoder is designed to obtain the image conditioned object query.

The elements from the shape feature library **S** or the texture feature library **T** and the N learned object queries *O* are concatenated to define a query $O' = [\mathbf{S}, \mathbf{T}, O]$. Given the fused image features $\hat{F}^k$ and the query O', the library-based object queries f^k_i $(i = 1, 2, \ldots, N,\ k = 1, 2, \ldots, K)$ are obtained through a couple of decoding layers that employ the structure of FS-DETR (Bulat et al. 2023).

As shown in Fig. 2, for the Consensus Dispersion module and Linear Interpolation module, to obtain the object-conditioned image features, we adapt the combination of CLIPSeg (Lüddecke and Ecker 2022) and group-based semantic agreement Transformer (Xu and Mu 2023). The node features f^k_i serve as conditional prompts. Feature fusion between node features and image features is achieved through linear interpolation. Formally, in each perspective, the object-conditioned image features can be obtained by linear interpolation LI

$$F^k_i = LI(\hat{F}^k, f^k_i),\ i = 1...N,\ k = 1, 2, ...K$$

Next, the scene consensus g is dispersed to multi-view object-conditioned image features through channel-wise multiplication $\hat{F}^k_i = F^k_i \cdot g$.

Then we use an weighted averaging operation to obtain a unified, multi-view feature representation for each object

$$f^{global}_i = \sum_{k=1}^{K} \frac{c^k_i}{\sum_{k'=1}^{K} c^{k'}_i} \cdot f^k_i, \quad i = 1, 2, \ldots, N. \tag{2}$$

where $c^k_i = sigmoid(MLP(F^k_i))$ represents the predicted confidence score of the brick i in the view k . This approach effectively integrates diverse perspectives to enhance the overall understanding for the object's attribute.

Brick Number Prediction

For each view $k = 1, 2, \ldots, K$, we average the N object-conditioned image features $\hat{F}^k_{avg} = \frac{1}{N}\sum_{i=1}^{N} \hat{F}^k_i$. Then the scene feature can be obtained by $\hat{F}_{scene} = \frac{1}{K}\sum_{k=1}^{K} \hat{F}^k_{avg}$. Finally a couple of convolution layers are used to predict the number of bricks in the scene.

Relation Graph Prediction

Predicting the assembly sequence is equivalent to predicting the connections of bricks which can be described using the relationship graph. If brick A is placed on the top of B, then there is a directed edge from B to A. A graph convolutional network (Kipf and Welling 2016) is adopted to predict the existence of each edge in the graph. Given the complete graph $G = (V, E)$ with initial node features $p^0_i = f^{global}_i$. Similar to Johnson, Gupta, and Fei-Fei (2018); Wald et al. (2020), we implemented the GCN using the MLP (multi-layer perception) structure. Each iteration of message passing starts from computing edge features: $e^{t+1}_{i,j} \leftarrow MLP([p^t_i, p^t_j])$. An average-pooling is performed over all edge features connected to a node, obtaining an updated node feature

$$p^{t+1}_i = \frac{1}{|\{u|(u,i) \in E\}|} \sum_{(u,i)\in E} e^{t+1}_{u,i}. \tag{3}$$

After gathering the edge features via graph convolutions as $\{e^t_{i,j}\}_{t=1}^{T}$, we use another MLP to predict the probability of the existence of each edge, $P_{(i,j)} = MLP(e^T_{i,j})$

Finally, to determine the assembly sequence during inference, the directed edges are sorted in descending order according to the predicted probability and subsequently incorporated into the directed graph. If a loop is formed after adding an edge, this edge will not be added. The process continues until it reach a state where there exists a vertex that can reach all other vertices in the graph.

Brick Pose Estimation

The N object query features $\{f^{global}_i\}_{i=1}^{N}$ are used for shape and texture classification. The shape prediction head predicts the shape label and the texture prediction head predicts the texture label.

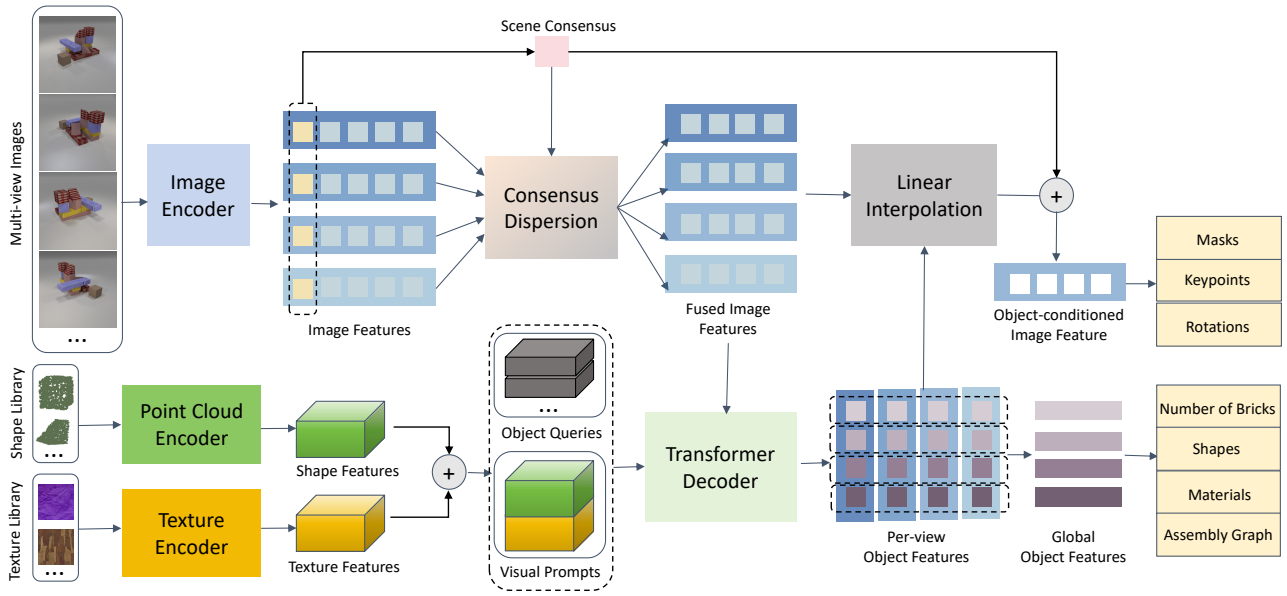

Figure 2: The proposed Neural Assembler architecture starts with an image encoder that outputs visual embeddings from multi-view images, which are then fused through the consensus module. The shape and texture library serves as visual prompts for object detection, followed by the application of a transformer decoder module to extract library-based object features. Finally, the object-conditioned image features are decoded into bricks' masks, keypoints, and rotation angles, while the global features are decoded into the bricks' textures, shapes, brick count, and the assembly graph.

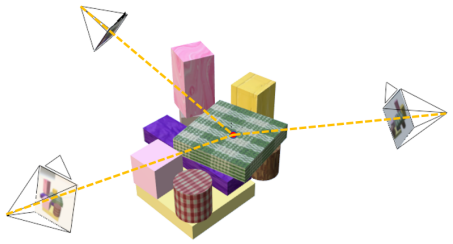

Figure 3: Illustration of 3D position prediction module.

Mask and Heatmap Prediction. We employ a simple deconvolution layer on the object-conditioned image feature $\hat{F}_i^k$ to obtain the heatmap of keypoint and mask of the object i in each perspective k.

Rotation Prediction The rotation angle is represented as a 2D vector representing the sine and cosine values of it. The rotation predictor accepts conditional image features $\hat{F}_i^k$ as input and outputs the sine and cosine value.

Confidence Score Prediction Since bricks may not be visible at all perspectives, here we predict the the confidence score c_i^k of each brick at each perspective. Specifically, c_i^k represents the IoU(Intersection Over Union) between the predicted mask and ground truth mask.

During inference, the pose of each object in 3D space is obtained by merging the poses from each perspective (see Figure 3). In more details, for 3D position prediction, our method involves detecting keypoints of the object parts from the view whose confidence score is higher than a threshold θ. Then, utilizing the camera parameters, the rays in 3D space generated by keypoints are used to infer the object's position in 3D space. Each ray R_i is represented as $r_i(t) = O_i + t \cdot D_i$ where O_i is the origin and D_i is the direction. Our objective is to find a point P that minimizes the function:

$$h(Z) = \sum_{i}^{L} d(Z, R_i), \tag{4}$$

where L is the number of rays and $d(Z, R_i)$ represents the shortest distance from the point P to the ray R_i. Here, the minimization of the objective function $h(Z)$ is approached through the gradient descent method.

Training and Loss Functions

We train Neural Assembler with full supervision on the generated dataset where each sample we have the groundtruth shape, texture, keypoint, mask, rotation information of each brick, the number of bricks and the relationship graph of bricks. The entire neural network is trained end-to-end with gradient descent. Our objective function is computed by $L = \alpha \cdot L_{count} + \beta \cdot L_{graph} + L_{pose}$, where L_{count} is the L1 Loss between the predicted number of bricks and ground truth $count_{gt}$.

Following Carion et al. (2020), bipartite matching is used to find an optimal permutation $\{\sigma_i\}_{i=1}^{N}$ to match the N object queries and ground truth bricks in the scene. The pose loss of bricks includes the loss of shape, texture, keypoint, mask and rotation.

$$L_{pose} = L_{keypoint} + L_{mask} + \gamma_1 L_{rotation} \tag{5}$$

$$+ \gamma_2 L_{shape} + \gamma_3 L_{texture} + \gamma_4 L_{confidence}, \tag{6}$$

where $L_{keypoint}$ is the focal loss (Lin et al. 2017) computed based on the predicted heatmap and ground truth heatmap generated by Kps_{σ_i}, L_{mask} is the focal and dice loss between the predicted mask and ground truth mask M_{σ_i}, $L_{rotation}$ is the L1 Loss between the prediced sine and cosine and the ground truth sine and cosine of Rot_{σ_i}, L_{shape} and $L_{texture}$ are the cross entropy loss for shape and texture classification and $L_{confidence}$ is L1 Loss between the predicted confidence score and IoU of the predicted mask and ground truth mask. Our model strategically prioritizes the hyperparameters $L_{keypoint}$ and L_{mask} due to their critical impact on object detection, essential for accurate object interaction and identification in complex scenes. In contrast, $L_{rotation}$, L_{shape}, $L_{texture}$ and $L_{confidence}$ are assigned a reduced weight of 0.1 each, a decision grounded in empirical findings that highlight their relatively minor incremental benefits to overall model efficacy.

L_{graph} is the loss for relationship graph prediction. Firstly, we define the loss for any subset of the entire edge set. For a subset $\hat{E}$ of the edge set E of the complete graph, the edge loss of $\hat{E}$ is defined as

$$L_{\hat{E}} = \sum_{(x,y)\in\hat{E}} L_{CE}(P_{(x,y)}, \hat{E}_{(x,y)}). \tag{7}$$

Then the L_{graph} is defined as $L_{graph} = L_E + L_{topK_E}$

$$L_{topK_E} = \frac{1}{K_E}\sum_{k=1}^{K_E} L_{E_{top_k}}, \tag{8}$$

where E_{top_k} is the edge set with the top k highest predicted probability. Since the entire relationship graph is a directed graph with sparse edges, the hyperparameter K_E is defined as $count_{gt} + 1$.

Experiments

Comparison With Baselines

Dataset Preparation. Experiments are conducted on two self-constructed datasets. CLEVR-Assembly Dataset is created using the CLEVR-Engine (Johnson et al. 2017), and LEGO-Assembly is synthesized with Pytorch3d. The dataset analysis is summarized in Table 1, where we report the number of bricks per sample, the visibility probability of each brick from different perspectives, the variety of brick shapes, the number of textures, and the average height of brick model per sample.

Metric	CLEVR-Assembly	LEGO-Assembly
Brick shapes	6	12
Textures	16	8
Visible Probability (%)	76.5	82.6
Avg Bricks per Sample	7.51	7.39
Avg Height	4.01	4.49

Table 1: Dataset Analysis

Baselines. In addressing this novel task, for which no direct baseline exists, we have established a comparative framework against three distinct baseline methods to demonstrate the efficacy of our approach. First, to assess the validity of our assembly order prediction methodology, we use a Long Short-Term Memory (LSTM) (Graves and Graves 2012) module as a surrogate baseline to contrast with our Graph Convolutional Network (GCN) based module. This comparison aims to highlight the enhanced predictive capabilities our GCN model brings to complex assembly sequences. Furthermore, for the object pose estimation component, our methodology is rigorously benchmarked against DETR3D (Wang et al. 2022b), a standard baseline in the field of 3D object detection. This comparison is pivotal in underscoring the adaptability and accuracy of our model in 3D pose estimation, a crucial aspect in varied application domains.

Implementation Details. Our approach is implemented in single-scale version for fair comparison with other works. It incorporates a CLIP(Radford et al. 2021) pre-trained ViT-B/16 image encoder, a PointNet-based (Qi et al. 2017) point cloud encoder, and a ResNet-18(He et al. 2016) for texture encoding. We employ a two-layer residual network for brick number prediction. The shape, material, iou prediction heads are implemented using 3-layer MLP and ReLU activations. Rotation prediction also uses a two-layer residual network, and our GCN architecture employs two message-passing layers. Training is conducted on an RTX 3090 GPU using AdamW, with an initial rate of 5e-4, decaying by 0.8 per epoch, a weight decay of 1e-3, and batch size 8 over 10 epochs for both datasets.

Evaluation Metrics. We introduce metrics to evaluate our algorithm's performance at both the per-scene and per-step levels, providing a holistic measure of the efficacy.

- **Per-Step Metrics**: Assuming the assembly order is known, we evaluate the accuracy of individual brick predictions. Key metrics include Pos Acc and Rot Acc (3D position and rotation accuracy), Shape Acc and Texture Acc (shape and texture accuracy), Kps MSE (2D keypoint error), mIoU (mask mIoU), F1-score (relation graph match), and Per-step Acc (accuracy when both brick type and pose are correctly predicted).
- **Per-Scene Metrics**: We assess the ability to predict the complete assembly sequence from multi-view images. Metrics include Model CR (completion rate of brick model), Order CR (assembly order accuracy without considering pose), Per-scene Acc (accuracy in assembling the entire brick model), and Count Acc (precision in predicting the number of bricks).

Results on CLEVR-Assembly Dataset. Per-scene quantitative results on the CLEVR-Assembly Dataset are summarized in Table 2. Neural Assembler outperforms baseline models in all metrics considered. As shown in Table 3, it locates objects more accurately than DETR3D.

The metric CCA proposed by Chen et al. (2019) is adopted here for the brick order evaluation. It denotes the probability distribution of the number of brick that a model can consecutively place from scratch. As shown in Fig.4, LSTM perform worse than GCN. It is because time dependence is not crucial for the assembly order prediction. Instead, the assembly problem requires prediction from complex spatial relationships. The adeptness of GCN in capturing spatial relation plays a critical role in understanding the assembly order.

Method	Model CR	Per-scene Acc	Count Acc	Order CR
LEGO-Assembly Dataset				
LSTM	27.5	5.3	60.3	35.1
DETR3D	25.8	2.5	61.5	63.5
Ours (w/o consensus)	43.7	18.4	69.0	64.5
Ours	**43.9**	**22.9**	**76.3**	**69.4**
CLEVR-Assembly Dataset				
LSTM	19.7	8.0	91.5	22.6
DETR3D	16.8	4.5	89.5	35.3
Ours (w/o consensus)	28.6	6.6	92.1	45.5
Ours (2 views)	22.0	4.6	88.7	38.6
Ours (3 views)	25.7	9.3	94.0	44.5
Ours (Mean Pooling)	41.5	22.5	95.5	62.1
Ours	**42.4**	**23.2**	**97.4**	**62.6**

Table 2: Comparison of baselines on Per-Scene metrics.

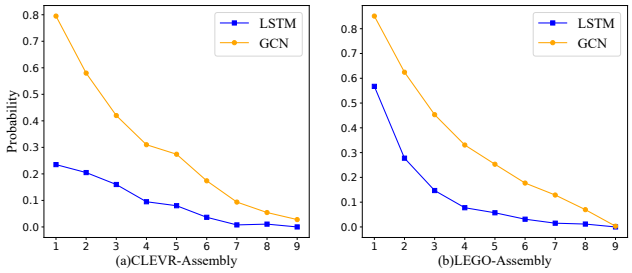

Figure 4: The probability distribution of CCA.

Fig. 5 shows the generated assembly instructions for a brick model in the CLEVR-Assembly Dataset. Perspectives with confidence score greater than 0.66 are selected to infer the brick's information. It is evident that Neural Assembler is adept at excluding perspectives where bricks are obscured by predicting confidence scores, thereby identifying optimal perspectives for predicting positional information. Concurrently, it is capable of predicting the structure between bricks to determine the appropriate assembly sequence.

Results on LEGO-Assembly Dataset. Different from CLEVR-Assembly Dataset, LEGO bricks are connected through slots. It is easier to infer the position of the bricks based on the connection constraints between the LEGO bricks, as long as the rough position of the LEGO bricks is predicted. The LEGO brick will only have rotations chosen from $(0°, 90°, 180°, 270°)$. Meanwhile, the position of LEGO bricks is discretized. However, there are many challenges in predicting the assembly sequence of LEGO brick models. For instance, the more compact assembly of LEGO bricks results in increased occlusion.

We adopt the connection-constrained inference subroutine and an inference-by-synthesis subroutine used in Wang et al. (2022a) to predict the position and rotation angle for each view, and then integrated them through voting. The results in Table 2 and Table 3 show that Neural Assembler yielsd more accurate results. Fig.6 further shows the generated assembly instructions for a LEGO model.

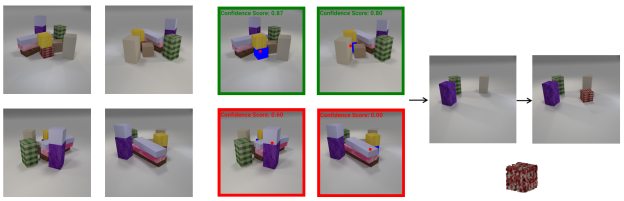

Figure 5: Result from CLEVR-Assembly Dataset.

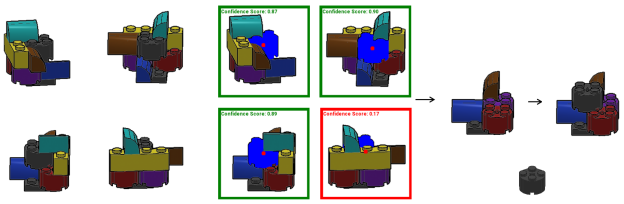

Figure 6: Result from LEGO-Assembly Dataset.

Ablation Study

Consensus Module We contrast our approach with a method that does not leverage scene consensus. As shown in Table 3, extracting the consensus can better align the information of the images from various perspectives. Without scene consensus, it is difficult for the model to integrate information from multi-view images to obtain the overall information of each brick.

Number of Views Furthermore, we compared the results obtained by accepting different numbers of images as input. As shown in Tables 2 and 3, the result shows that more perspectives as input can improve the performance. This is because each brick may not be seen from some perspectives due to the existence of occlusion. More views mean more information for prediction.

Pooling Method We compared the effects of mean pooling and weighted pooling in the Equation (2). As can be shown in Table 2 and 3, the weighted pooling method outperforms mean pooling by accounting for the visibility of each brick in different views.

Real-World Experiments

To confirm the model's generalizability, a comprehensive test dataset is constructed, complete with annotations for each brick's shape, position, and rotation. For each sample, we acquired real brick images using a Realsense camera in the real world and generated corresponding simulated images in the simulation environment employing real-world camera parameters to ensure the consistent coordinate between the simulated and real environments. The dataset encompasses 5 brick types and 7 textures, averaging 6.1 bricks per brick model.

To evaluate the Neural Assembler, we collected point clouds and textures from real bricks. This data facilitated the creation of a simulated dataset, used for fine-tuning the model initially trained on the CLEVR-Assembly Dataset.

As indicated in Table 4, Neural Assembler achieves performance in real-world experiments close to the results ob-

Method	Per-Step Acc	Pos Acc	Rot Acc	Shape Acc	Texture Acc	mIoU	Kps Mse	F1
LEGO-Assembly Dataset								
DETR3D (Wang et al. 2022b)	41.7	47.2	78.2	87.8	98.3	-	-	0.797
Ours (w/o consensus)	71.7	79.0	87.2	89.9	98.2	76.3	1.12	0.829
Ours	**73.5**	**80.4**	**88.0**	**91.5**	**98.3**	**78.5**	**0.88**	**0.820**
CLEVR-Assembly Dataset								
DETR3D (Wang et al. 2022b)	29.2	32.4	75.6	72.4	67.0	-	-	0.734
Ours (w/o consensus)	57.2	67.9	78.9	87.1	88.5	61.2	1.2	0.781
Ours (2 views)	56.1	71.7	73.0	80.0	86.4	65.4	2.5	0.721
Ours (3 views)	61.1	74.6	77.9	85.7	90.7	67.1	1.5	0.772
Ours (Mean Pooling)	69.2	79.1	84.1	91.5	**93.8**	71.1	0.78	**0.840**
Ours	**74.4**	**80.4**	**87.2**	**91.8**	93.2	**72.0**	**0.87**	0.773

Table 3: Comparison of baselines on Per-Step metrics.

Method	Per-scene Acc	Model CR
Simulated Dataset		
LSTM (Graves and Graves 2012)	16.0	27.3
DETR3D (Wang et al. 2022b)	7.3	21.8
Ours	**34.2**	**58.5**
Real-World Dataset		
LSTM (Graves and Graves 2012)	7.3	21.8
DETR3D (Wang et al. 2022b)	2.4	12.8
Ours	**22.0**	**50.5**

Table 4: The performance of the fine-tuned model on the novel simulated dataset and real-world dataset.

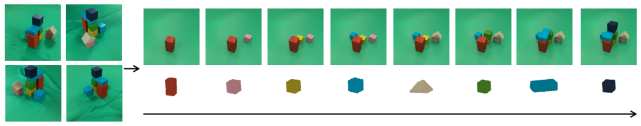

Figure 7: The result from the real-world brick model. The left side displays 4 images captured using a Realsense camera, while the right side shows the assembly order of the brick model.

tained in simulated environments, demonstrating its robustness. Fig. 7 presents the result on the real-world dataset.

Limitation and Future Work

As shown in Fig. 8, the model confidently but incorrectly predicts the highlighted block in View 1, while in View 3, despite correct keypoint identification, occlusion results in a less confident. In Fig. 9, the model incorrectly predicts the keypoint in View 2, which causes erroneous overall prediction. As can be seen from the failure cases above, the objects under the brick model will be greatly blocked by the bricks pressing on it. To alleviate this problem, in future work, we plan to enhance model performance with a deeper integration of physical scene understanding. The model is expected to not only interpret the visual aspects but also the underlying physical principles governing the scene.

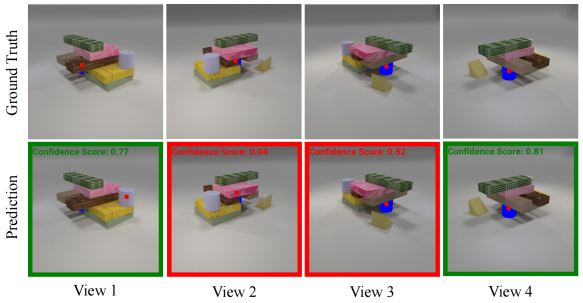

Figure 8: Failure case in CLEVR-Assembly Dataset.

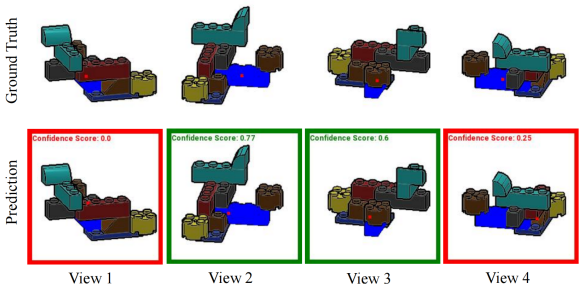

Figure 9: Failure case in LEGO-Assembly Dataset.

Conclusion

We study the problem of generating robotic assembly instructions given multi-view images and propose Neural Assembler, an end-to-end framework. The key idea behind our model is to learn the relation graph to predict the assembly order and integrate multi-view information to infer 3D poses of bricks. Results show that our model outperforms existing methods on the newly collected CLEVR-Assembly Dataset and LEGO-Assembly Dataset.

Acknowledgements

This work is supported by an internal grant of Peking University (2024JK28).

References

Bever, T. G.; and Poeppel, D. 2010. Analysis by synthesis: a (re-) emerging program of research for language and vision. *Biolinguistics*, 4(2-3): 174–200.

Bulat, A.; Guerrero, R.; Martinez, B.; and Tzimiropoulos, G. 2023. Fs-detr: Few-shot detection transformer with prompting and without re-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11793–11802.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, 213–229. Springer.

Chaudhuri, S.; and Koltun, V. 2010. Data-driven suggestions for creativity support in 3D modeling. In *ACM SIGGRAPH Asia 2010 papers*, 1–10.

Chen, Z.; Guo, D.; Xiao, T.; Xie, S.; Chen, X.; Yu, H.; Gray, J.; Srinet, K.; Fan, H.; Ma, J.; et al. 2019. Order-aware generative modeling using the 3d-craft dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1764–1773.

Chung, H.; Kim, J.; Knyazev, B.; Lee, J.; Taylor, G. W.; Park, J.; and Cho, M. 2021. Brick-by-brick: Combinatorial construction with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 5745–5757.

Cong, Y.; Yang, M. Y.; and Rosenhahn, B. 2023. Reltr: Relation transformer for scene graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ellis, K.; Ritchie, D.; Solar-Lezama, A.; and Tenenbaum, J. 2018. Learning to infer graphics programs from hand-drawn images. *Advances in neural information processing systems*, 31.

Graves, A.; and Graves, A. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, 37–45.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hou, J.; Dai, A.; and Nießner, M. 2019. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4421–4430.

Jaritz, M.; Gu, J.; and Su, H. 2019. Multi-view pointnet for 3d scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0–0.

Johnson, J.; Gupta, A.; and Fei-Fei, L. 2018. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1219–1228.

Johnson, J.; Hariharan, B.; Van Der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2901–2910.

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D.; Bernstein, M.; and Fei-Fei, L. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3668–3678.

Kim, J.; Chung, H.; Lee, J.; Cho, M.; and Park, J. 2020. Combinatorial 3D Shape Generation via Sequential Assembly. In *NeurIPS Workshop on Machine Learning for Engineering Modeling, Simulation, and Design (ML4Eng)*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lee, S.; Kim, J.; Kim, J. W.; and Moon, B.-R. 2015. Finding an optimal LEGO® brick layout of voxelized 3D object using a genetic algorithm. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1215–1222.

Li, R.; Zhang, S.; and He, X. 2022. Sgtr: End-to-end scene graph generation with transformer. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 19486–19496.

Li, Y.; Mo, K.; Shao, L.; Sung, M.; and Guibas, L. 2020. Learning 3d part assembly from a single image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, 664–682. Springer.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, Y.; and Wu, Z. 2019. Learning to describe scenes with programs. In *International conference on learning representations*.

Lüddecke, T.; and Ecker, A. 2022. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7086–7096.

Mo, K.; Guerrero, P.; Yi, L.; Su, H.; Wonka, P.; Mitra, N.; and Guibas, L. J. 2019. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*.

Murez, Z.; Van As, T.; Bartolozzi, J.; Sinha, A.; Badrinarayanan, V.; and Rabinovich, A. 2020. Atlas: End-to-end 3d scene reconstruction from posed images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, 414–431. Springer.

Niu, C.; Li, J.; and Xu, K. 2018. Im2struct: Recovering 3d shape structure from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4521–4529.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qiu, Y.; Sun, Y.; Matsuzawa, F.; Iwata, K.; and Kataoka, H. 2023. Graph Representation for Order-Aware Visual Transformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22793–22802.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Shao, T.; Li, D.; Rong, Y.; Zheng, C.; and Zhou, K. 2016. Dynamic Furniture Modeling through Assembly Instructions. *ACM Trans. Graph.*, 35(6).

Shen, C.-H.; Fu, H.; Chen, K.; and Hu, S.-M. 2012. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 31(6): 1–11.

Suárez-Ruiz, F.; Zhou, X.; and Pham, Q.-C. 2018. Can robots assemble an IKEA chair? *Science Robotics*, 3(17): eaat6385.

Sung, M.; Su, H.; Kim, V. G.; Chaudhuri, S.; and Guibas, L. 2017. ComplementMe: Weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG)*, 36(6): 1–12.

van den Hengel, A.; Russell, C.; Dick, A.; Bastian, J.; Pooley, D.; Fleming, L.; and Agapito, L. 2015. Part-based modelling of compound scenes from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 878–886.

Wald, J.; Dhamo, H.; Navab, N.; and Tombari, F. 2020. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3961–3970.

Walsman, A.; Zhang, M.; Kotar, K.; Desingh, K.; Farhadi, A.; and Fox, D. 2022. Break and make: Interactive structural understanding using lego bricks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, 90–107. Springer.

Wang, R.; Zhang, Y.; Mao, J.; Cheng, C.-Y.; and Wu, J. 2022a. Translating a Visual LEGO Manual to a Machine-Executable Plan. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*, 677–694. Springer.

Wang, Y.; Guizilini, V. C.; Zhang, T.; Wang, Y.; Zhao, H.; and Solomon, J. 2022b. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, 180–191. PMLR.

Wu, J.; Lu, E.; Kohli, P.; Freeman, B.; and Tenenbaum, J. 2017. Learning to see physics via visual de-animation. *Advances in Neural Information Processing Systems*, 30.

Xu, P.; and Mu, Y. 2023. Co-Salient Object Detection with Semantic-Level Consensus Extraction and Dispersion. In *Proceedings of the 31st ACM International Conference on Multimedia*, 2744–2755.

Yuille, A.; and Kersten, D. 2006. Vision as Bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 10(7): 301–308.

Zhan, G.; Fan, Q.; Mo, K.; Shao, L.; Chen, B.; Guibas, L. J.; Dong, H.; et al. 2020. Generative 3d part assembly via dynamic graph learning. *Advances in Neural Information Processing Systems*, 33: 6315–6326.