G-FARS: Gradient-Field-based Auto-Regressive Sampling for 3D Part Grouping

Junfeng Cheng
Imperial College London
London, UK
junfeng.cheng20@imperial.ac.uk

Tania Stathaki
Imperial College London
London, UK
t.stathaki@imperial.ac.uk

Abstract

This paper proposes a novel task named "3D part grouping". Suppose there is a mixed set containing scattered parts from various shapes. This task requires algorithms to find out every possible combination among all the parts. To address this challenge, we propose the so called Gradient Field-based Auto-Regressive Sampling framework (G-FARS) tailored specifically for the 3D part grouping task. In our framework, we design a gradient-field-based selection graph neural network (GNN) to learn the gradients of a log conditional probability density in terms of part selection, where the condition is the given mixed part set. This innovative approach, implemented through the gradient-field-based selection GNN, effectively captures complex relationships among all the parts in the input. Upon completion of the training process, our framework becomes capable of autonomously grouping 3D parts by iteratively selecting them from the mixed part set, leveraging the knowledge acquired by the trained gradient-field-based selection GNN. Our code is available at: https://github.com/J-F-Cheng/G-FARS-3DPartGrouping.

1. Introduction

Assuming that you purchase multiple unassembled IKEA chairs and carelessly mix all the parts together, it can quickly become a nightmare to sort through and assemble each chair. The task can be especially daunting if the pieces from different chairs are mixed together, making it challenging to identify the correct components for each chair. Similarly, in the field of archaeology, recovering broken cultural relics can be incredibly difficult, as the fragments are often intermingled with the pieces from other relics. In such cases, archaeologists must carefully separate the mixed fragments and piece them together to reconstruct the original relics. In a similar vein, the field of LEGO automatic assembly requires AI agents to select different combinations of parts from massive LEGO blocks and assemble them into a shape. All of these examples contain two goals: The first goal is to identify the correct combinations from the mixed part set (*i.e.* part grouping) and the second one is to assemble them into reasonable shapes (*i.e.* part assembly). To achieve these two objectives, algorithms must first be capable of comprehending the geometric relationships among all the parts. Next, they should be able to separate the parts by their shapes, and finally, assemble the chosen parts into reasonable shapes.

For the part assembly, previous works have researched some methods for assembling a given group of parts. DGL-Net [15] is the first work to explore the assembly problem without prior instruction. The DGL-Net algorithm can predict the 6-DoF poses for each input part, enabling translation and rotation of the parts to their expected positions. RGL-Net [27] is another part assembly work that utilizes sequential information among all the input parts. By assembling shapes in a specific order (e.g., top-to-bottom), RGL-Net achieves more accurate assembly. IET [41] is a recently proposed algorithm that utilizes an instance encoded transformer and self-attention mechanisms [30, 32, 36, 37, 40] to enhance the network's assembly ability.

However, part grouping still remains an unsolved problem. As previously mentioned, the goal of part grouping is to use algorithms to identify all possible combinations in a mixed part set. To address this, we introduce the 3D part grouping task. The definition of this task is presented in Fig. 1. Suppose we have a set of mixed parts from N different shapes. The 3D part grouping task mandates the algorithms to process all these parts and categorize them into groups based on their originating shapes.

Our proposed task 3D part grouping is challenging for two main reasons. First, the algorithms must understand the relationships among all the parts. Second, the exact number of potential groups, N, is unknown. This uncertainty complicates both the problem formulation and the creation of effective algorithms. To tackle these challenges, we introduce Gradient-Field-based Auto-Regressive Sampling (G-FARS) framework in this paper. This framework integrates a gradient-field-based graph neural network for the encoded parts, aiding in understanding the relationships among all

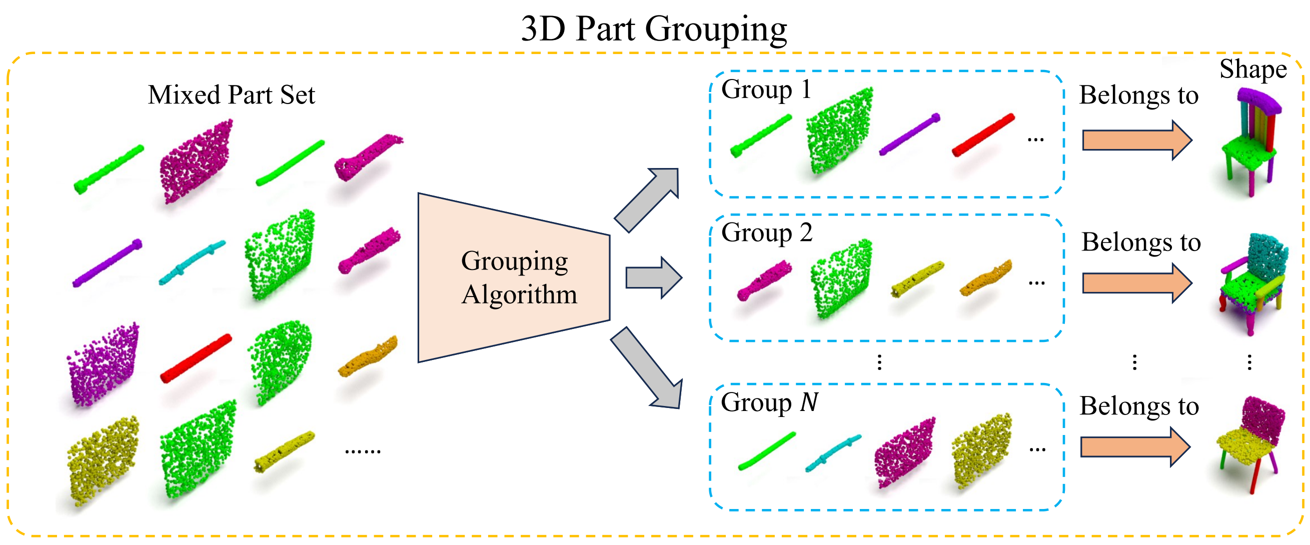

Figure 1. The definition of our proposed 3D part grouping task. Assuming we have a set which contains mixed parts from N shapes. Our goal in this task is to use a grouping algorithm to separate these mixed parts and group them by their respective shapes.

the input parts. Moreover, it can auto-regressively sample new groups from the mixed set, enabling the algorithm to identify all the groups, regardless of the number of potential groups N. More details about G-FARS are in Sec. 4.

Our proposed task and algorithm hold significant potential for the industrial world. In manufacturing environments, where parts from multiple products are intermixed, a robotic system utilizing this method can aid in automated sorting, leading to more efficient production lines. The 3D part grouping algorithm can also generalize to other domains. For instance, in recycling facilities, this method could help segregate mixed materials into appropriate categories for processing, thus improving waste management practices. Our contributions are concluded as follows:

- We introduce a novel setting termed 3D part grouping, which necessitates algorithms to identify all possible combinations of the input mixed parts.
- We present the Gradient-Field-based Auto-Regressive Sampling (G-FARS) for the 3D part grouping task. Our framework addresses two primary challenges in this task: 1. understanding the relationships among parts and 2. managing the uncertainty of the potential groups.
- Utilizing PartNet [25], we establish benchmarks for our introduced task and showcase the effectiveness of G-FARS in the 3D part grouping domain.

2. Related Works

2.1. Combinatorial Optimization

Combinatorial Optimization is an area that studies the task of finding the best object from a finite set of objects. The 3D part grouping task aims to identify every possible combination in a given set of parts. From this perspective, the 3D part grouping task aligns with combinatorial optimization problems, and we can glean insights from combinatorial optimization algorithms.

Classic algorithms Research in combinatorial optimization can be traced back to the 1960s. Some hallmark algorithms include Dijkstra's shortest path algorithm [4, 16–18, 29] and Kruskal's minimum spanning tree algorithm [12]. These algorithms provided a robust foundation for understanding the structure and intricacies of combinatorial problems.

Heuristic methods have introduced a new dimension to combinatorial optimization. Genetic algorithms [9–11, 14, 34], inspired by natural selection processes, stand as prominent methods for these problems. They employ mechanisms like mutation, crossover, and selection to pinpoint solutions. Another significant method is Ant Colony Optimization [3, 8, 19, 20, 42], which emulates the behavior of ants when finding a path from their colony to food sources.

Reinforcement learning-based methods Recently, some reinforcement learning-based methods also demonstrate their ability in combinatorial optimization problems [2, 24]. The basic idea behind these methods is to learn a policy to search the solution space.

Graph neural networks are also making their mark in addressing combinatorial optimization problems. They transpose the problem graph into continuous space, aiding in the prediction of optimal combinatorial solutions. The essence of graph neural networks revolves around the use of message passing and aggregation operations on nodes and edges to unearth and understand the graph's layout. Graph Convolutional Networks (GCNs) are a notable type of graph neural network that applies convolutional operations on graphs, capturing local and global structural nuances [21]. Edge Convolution, another variant, focuses on the features of edges between nodes, amplifying the model's capacity to depict complex graphs [38].

2.2. 3D Part Assembly

Although this work does not consider the task of part assembly, it is still worthwhile to review these works, as they provide insights into identifying the relationships among the input parts.

3D part assembly is a task proposed by Huang et al. [15] which aims to assemble separate parts into a complete shape without any external guidance. The goal of 3D part assembly is to predict a translation vector and a rotation vector for each part [6, 15, 27, 41]. We introduce two categories of 3D part assembly algorithms, graph neural network based algorithms and transformer based algorithm:

Graph neural network based methods Huang et al. [15] propose Dynamic Graph Learning algorithm to achieve the goal of 3D part assembly task. DGL-Net includes a PointNet [28] for part feature extraction. They also propose an iterative graph neural network backbone for message passing among all the part features. Besides, they propose dynamic relation reasoning modules to learn the relationship among all the encoded parts. They also have dynamic part aggregation modules for more direct information exchanges among geometrically-equivalent parts.

RGL-Net, described in [27], is also a GNN-based algorithm designed for the part assembly task. The key concept behind RGL-Net is sequential assembly, where the separate parts are assembled in a specific order, such as a top-to-down approach. In particular, the authors employ GRUs [7] to learn the order information, which significantly improves the assembling performance. However, one potential limitation of RGL-Net is that its performance may be suboptimal in non-ordered assembly settings. In other words, when the parts are assembled in a random or unstructured order, RGL-Net may not perform as well as it does in the ordered assembly setting.

Transformer based method Zhang et al. [41] propose a method that employs a transformer-based framework and self-attention mechanism to model the relationships between different parts, while resolving the ambiguity issue using instance encoding. The method includes four modules: a shared PointNet for feature extraction, a transformer encoder for reasoning the relationships between parts, an MLP predictor for pose estimation, and an instance encoder for handling the ambiguity between parts. The experimental results show that IET achieves better performance than DGL-Net and RGL-Net.

3. Backgrounds of Score-based Modeling Through Stochastic Differential Equations

Our proposed G-FARS is built upon the score-based modeling through stochastic differential equations (SDEs). It is necessary to discuss the basic principles of the score-based modeling before introducing G-FARS.

Score-based modeling through SDEs [33] is a recently proposed technique for generation tasks. The basic idea behind this modeling method is estimating the gradients of the data distribution. The new data can be generated by sampling the estimated data distribution. Score-based models with SDEs are trained to estimate a time-dependent score $S_\theta(\mathbf{x}(t), t)$ of a given probability density function $p_t(\mathbf{x})$, which can be described by $S_\theta(\mathbf{x}(t), t) = \nabla_\mathbf{x} \log p_t(\mathbf{x})$. To successfully achieve a score-based model, we need to set up a diffusion process which can be represented by a continuous-time stochastic process $\{\mathbf{x}(t) \in \mathbb{R}^d\}_{t=0}^T$, where $t \in [0, T]$. We choose a diffusion process such that $\mathbf{x}(0) \sim p_0$ and $\mathbf{x}(T) \sim p_T$, where p_0 is the data distribution, p_T is the prior distribution, and both are uncorrelated after the perturbation by the diffusion process. We can describe our diffusion process by using the following equation:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}. \tag{1}$$

In the above equation, $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient of the SDE, $g(t) \in \mathbb{R}$ represents the diffusion coefficient, and $\mathbf{w}$ means the standard Brownian motion. To generate new data, we can sample the score function by reversing the diffusion process:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_\mathbf{x} \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}, \tag{2}$$

where $\bar{\mathbf{w}}$ denotes a Brownian motion with time flowing in the reverse direction, and dt is an infinitesimal time step with a negative sign. To successfully train a score-based model, we optimize the following objective function:

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{\mathbf{x}(0)\sim p_0(\mathbf{x})}\mathbb{E}_{\mathbf{x}(t)\sim p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))} [\|S_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2]], \tag{3}$$

4. Method

This section introduces our proposed G-FARS framework. We start by discussing the working principle of the framework. We then explain how we establish its key component, referred to as the gradient-field-based selection GNN.

Following that, we explore the training algorithm for our framework. Finally, we discuss the sampling algorithm within the G-FARS framework.

4.1. G-FARS: Gradient-Field-based Auto-Regressive Sampling Framework

We propose a model, denoted as G-FARS, to solve the 3D part assembly problem, the workings of which are illustrated in Fig. 2. Before we discuss our framework, we introduce the definitions of the mathematical symbols first.

Mathematical symbol definitions Let n denote the index for iteration, and P_n represents the mixed part set at iteration n. Naturally, P_0 is the initial input part set before any processing by the algorithm. Each part in the mixed part set is a point cloud which has the dimension of 1000×3. The Boolean vector $\mathbf{c}_n$ is used for part selection. At iteration n, this vector determines how to select parts from the mixed part set P_n to form a new group n. The ultimate goal of our algorithm is to identify all possible groups.

Working principle Our framework consists of two key components: a PointNet and a gradient-field-based selection graph neural network (GNN). The PointNet is employed to encode all the input parts. The gradient-field-based selection GNN is constructed with Edge Convolution layers [38], which enables the framework to understand the relationships among all the input parts. We can use this network to sample new selection vectors for the encoded parts. Our framework operates as an auto-regressive algorithm. Assuming we are at iteration n, we begin with using the PointNet to obtain the encoded per-part features F_P^n. Following that, our gradient-field-based selection GNN is used to sample a selection vector $\mathbf{c}_n$. The sampled vector $\mathbf{c}_n$ allows us to identify group n, while the complementary vector $\overline{\mathbf{c}_n}$ (obtained via a bitwise NOT operation) determines the unselected parts. These unselected parts then form the input parts P_{n+1} for the subsequent iteration $n+1$.

Auto-Regressive Sampling As stated in the Introduction section, one critical issue in the 3D part grouping task is that the number of groups N is not certain. Our framework is able to solve this problem as it can auto-regressively sample new groups from the mixed part set. As stated in the working principle part, our framework is able to sample a new group at each iteration. In this case, our algorithm does not care the number N in the mixed part set, and it only stops when the mixed part set is cleaned or the algorithm reaches the maximum iterations.

4.2. Gradient-Field-based Selection Graph Neural Network

We have already discussed the main working principle of G-FARS. The problem remaining here is how to obtain the key component (*i.e.*, the gradient-field-based selection graph neural network $S_\theta^{\mathbf{c}}$) in our proposed framework. In our designed GNN, the nodes are the per-part features F_p^n encoded by the PointNet. These nodes are fully connected to facilitate message passing.

The learning objective Designing a suitable learning objective for the gradient-field-based selection graph is key to achieving success in auto-regressive sampling. Recalling the principle of G-FARS, we expect our designed model to predict the correct selection for the input parts P_n at iteration n. To achieve this, we design our gradient-field-based GNN to learn the distribution of selections conditioned on the input parts. Mathematically, our gradient-field-based selection GNN learns a conditional probability $p(\mathbf{c}_n \mid F_p^n)$. After the training process, the GNN learns all the ways of selecting for the mixed part set. In this case, at each iteration, we can use the trained GNN to sample a correct selection for the mixed part set.

4.3. Training

After confirming our learning objective, our next task is to achieve our goal of estimating the distribution $p(\mathbf{c}_n \mid F_p^n)$. Specifically, our gradient-field-based selection GNN is designed to approximate the gradients of the target log conditional probability density. Mathematically, we expect our model to satisfy $S_\theta^{\mathbf{c}} = \nabla_{\mathbf{c}} \log p_t(\mathbf{c}_n \mid F_P^n)$.

To achieve this goal, we basically follow the training steps proposed by Song et al. [33], and we have discussed this in the Backgrounds section (Sec. 3). However, we still need to modify the training objective function to the conditional form. We present the general form of the loss function in the following equation:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \lambda(t) \| S_\theta^{\mathbf{c}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0), \mathbf{y}) \|_2^2, \tag{4}$$

where $\mathbf{x}(0)$ is the original data, $\mathbf{x}(t)$ is the perturbed data, $\mathbf{y}$ represents the condition and t is time index. The next step is to train G-FARS through Algorithm 1. In the training algorithm, we first use PointNet to get the per-part feature F_P^n. Following that, we sample time index t from the uniform distribution $\mathcal{U}(0, T)$. Then the $\mathbf{c}_n(t)$ can be obtained through the perturbation. After that, we calculate the loss function and perform back propagation. At the end of the iteration, we optimize the parameters for both $PointNet$ and $S_\theta^{\mathbf{c}}$.

4.4. Sampling Algorithm

As stated above, our framework auto-regressively samples new selections to choose parts from the given part set. In this case, a suitable sampler for our framework is the key to achieve high performance of 3D part grouping. We use

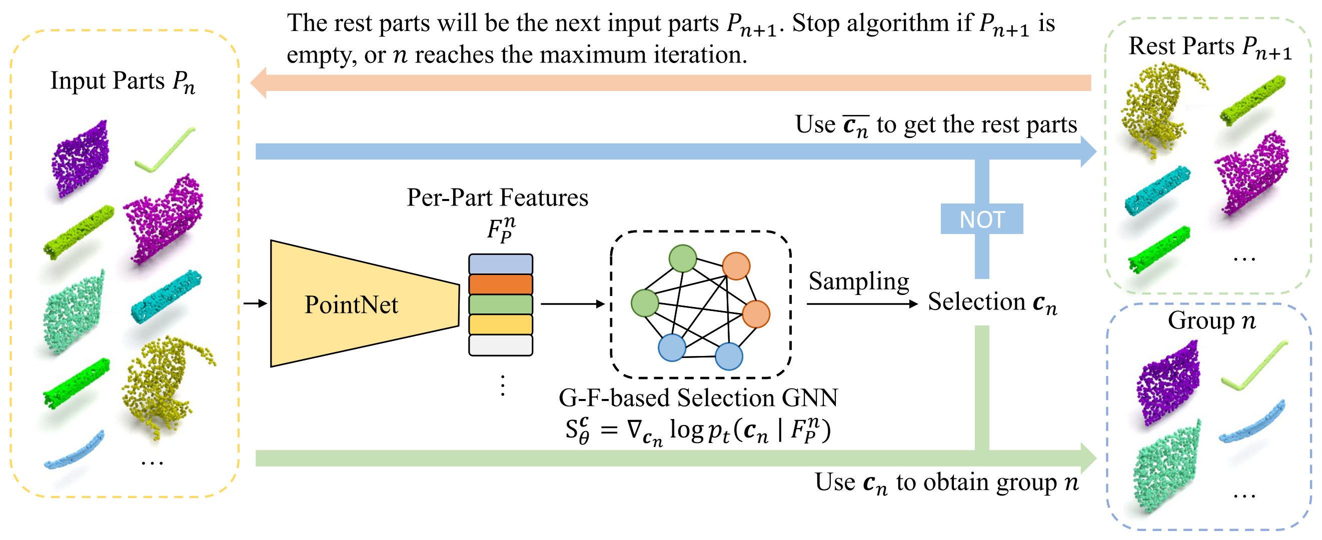
The rest parts will be the next input parts P_{n+1}. Stop algorithm if P_{n+1} is empty, or n reaches the maximum iteration.
Input Parts P_n
Rest Parts P_{n+1}
Use $\overline{c_n}$ to get the rest parts
NOT
Per-Part Features
F_P^n
PointNet
Sampling
Selection c_n
G-F-based Selection GNN
$S_\theta^c = \nabla_{c_n} \log p_t(c_n \mid F_P^n)$
Group n
Use c_n to obtain group n

Figure 2. The auto-regressive sampling procedure of our proposed framework. First, we obtain the per-part feature by using a PointNet [28] to encode all the input parts P_n at the iteration n. Then, we use the gradient-field-based (G-F-based) selection GNN to sample a selection vector $\mathbf{c}_n$ to obtain part group n, and use $\overline{\mathbf{c}_n}$ to get the rest parts P_{n+1}. P_{n+1} will be the next input parts at the next iteration. The auto-regressive sampling stops when P_{n+1} is empty, or n reaches the maximum iteration.

Algorithm 1 The training algorithm

```
Input: Training dataset $\mathcal{D}_{train}$
Parameter: $T$, $N$ epochs
Output: $S_\theta^{\mathbf{c}}$, $PointNet$
1: for $N$ epochs do
2:     for each $P_n, \mathbf{c}_n$ in $\mathcal{D}_{train}$ do
3:         $F_P^n \leftarrow PointNet(P_n)$;
4:         Sample $t \sim \mathcal{U}(0, T)$;
5:         Add perturbation on $\mathbf{c}_n(0)$ to obtain $\mathbf{c}_n(t)$;
6:         Calculate loss $\mathcal{L}(\mathbf{c}_n, F_P^n)$;
7:         Perform back propagation;
8:         Optimize the parameters of $PointNet$ and $S_\theta^{\mathbf{c}}$;
9:     end for
10: end for
11: return G-F Selection GNN $S_\theta^{\mathbf{c}}$, $PointNet$;
```

Algorithm 2 Predictor-Corrector sampler

```
Input: $F_P^n$
Parameter: $T, \sigma, N, C$
Output: the sampled result $\mathbf{c}_n(0)$
1: Sample $\mathbf{c}_n(T) \sim \mathcal{N}(\mathbf{0}, \frac{1}{2\log\sigma}(\sigma^{2T} - 1)\mathbf{I})$.
2: for $n \leftarrow N - 1$ to $0$ do
3:     $t_p \leftarrow \frac{(n+1)T}{N}$ $t \leftarrow \frac{nT}{N}$
4:     for $i \leftarrow 1$ to $C$ do
5:         $\mathbf{c}_n(t_p) \leftarrow Corrector(\mathbf{c}_n(t_p), F_P^n)$
6:     end for
7:     $\mathbf{c}_n(t) \leftarrow Predictor(\mathbf{c}_n(t_p), F_P^n)$
8: end for
9: return the sampled result $\mathbf{c}_n(0)$
```

Predictor-Corrector (PC) [33] as our sampler. We demonstrate a simplified version in Algorithm 2. This method ensures that the generated samples are close to the desired distribution. Since our application requires a conditional PC sampler, we have provided the algorithm in 2. After the sampling procedure, we use a threshold $T_h = 0.5$ to transform the selection vector $\mathbf{c}_n(0)$ to a boolean vector.

5. Experiment

5.1. Datasets

As 3D processing task, PartNet [25] can be a good option for us to conduct our experiments. We apply 6,323 chairs, 8,218 tables and 2,207 lamps from PartNet [25] in our experiments. We apply random mixing method to create our mixed part sets. We illustrate our mixing method in Fig. 3. First, we randomly selected N shapes from the PartNet dataset (N is also a random number). Next, we mixed all the parts into a single part set. Finally, we shuffle all the parts to obtain our mixed set. For more statistics of our mixed datasets, please refer to our Supplementary.

5.2. Evaluation Metrics

We use precision, recall and F1 score to evaluate the performance of the algorithms in 3D part grouping task.

Definitions for TP, FP and FN in 3D part grouping task Before the evaluation, it is necessary to define the TP, FP

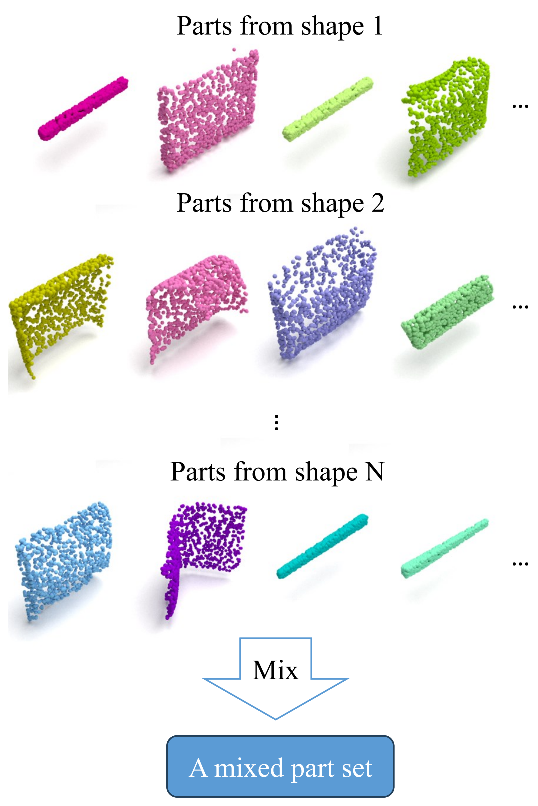

Figure 3. The random mixing method to create a mixed part set. We randomly selected N shapes from the PartNet dataset. We then mix all the parts into a single set. The sequence of parts are shuffled.

and FN in 3D part grouping task. We here assume we have one predicted group and the corresponding ground truth group. TP, FP and FN are defined as:

- TP: The parts selected by the predicted group are also in the ground truth group.
- FP: The parts selected by the predicted group are NOT in the ground truth group.
- FN: The parts which are NOT selected by the predicted group are in the ground truth group.

Single set average VS overall average To fully investigate the performance of the algorithms, we provide two averaging methods in the evaluation. We present the definitions as follows:

- Single set average: We calculate the TP, FP and FN in one part set, and calculate and record the corresponding precision, recall and F1 score of this single part set. After all the input part sets are processed, we calculate the average precision, recall and F1 score.
- Overall average: We calculate the TP, FP and FN for all the part sets and record them. After all the input part sets are processed, we calculated precision, recall and F1 score based on the accumulated TP, FP and FN.

Jaccard similarity matching In the evaluation procedure, a critical issue is that the sequence of the predicted part groups is not certain. In this situation, for fair evaluation, we apply Jaccard similarity to match the ground truth group with the predicted group. The formula of Jaccard similarity is shown as follows:

$$J(P, G) = \frac{|P \cap G|}{|P \cup G|} \tag{5}$$

where P is the predicted group, and G is the ground truth group. In our task, $|P \cap G|$ represents the number of parts which are included by both P and G, while $|P \cup G|$ represents the total number of unique parts in both P and G. In our evaluation program, each ground truth group is matched with the predicted group which has the highest Jaccard similarity.

5.3. Baselines

As 3D part grouping is a newly proposed task, there are no existing baselines specific to this task. Therefore, we propose some alternative methods for comparisons:

- **GRU-Mask:** We propose GRU-Mask which uses a GRU to generate a binary mask for the given part set. This mask indicates how to select parts to form groups. GRU-Mask is inspired by RNN-based methods [39, 43]. More details about GRU-Mask can be found in the supplementary.
- **Comp-Net:** Comp-Net is another method proposed by us, capable of classifying whether two parts can be grouped to form a shape. Our idea is inspired by RL-based approaches [23, 35]. We provide more information about Comp-Net in our supplementary.
- **Variants of G-FARS:** Inspired by [1, 5, 22, 26, 31], we propose three variants of G-FARS: G-FARS-CG, G-FARS-R, and G-FARS-T. We use the same training and inference algorithms (*i.e.,* Auto-Regressive sampling) as G-FARS in these variants. G-FARS-CG applies a GNN for message passing among all the encoded parts. Besides, it also includes a separate MLP to learn the score function for part selection. G-FARS-R and G-FARS-T use a ResNet [13] and a transformer [36] to learn the selection score function respectively. For these variants, we employ a score function modeling way that differs from the one used in G-FARS, which is introduced in the supplementary materials.

5.4. Experimental Details

Hardware and software details We conduct all the experiments on a personal computer with CPU R9 3900x and RTX3090. The memory of the PC is 32 GB. The operating system is Ubuntu Linux.

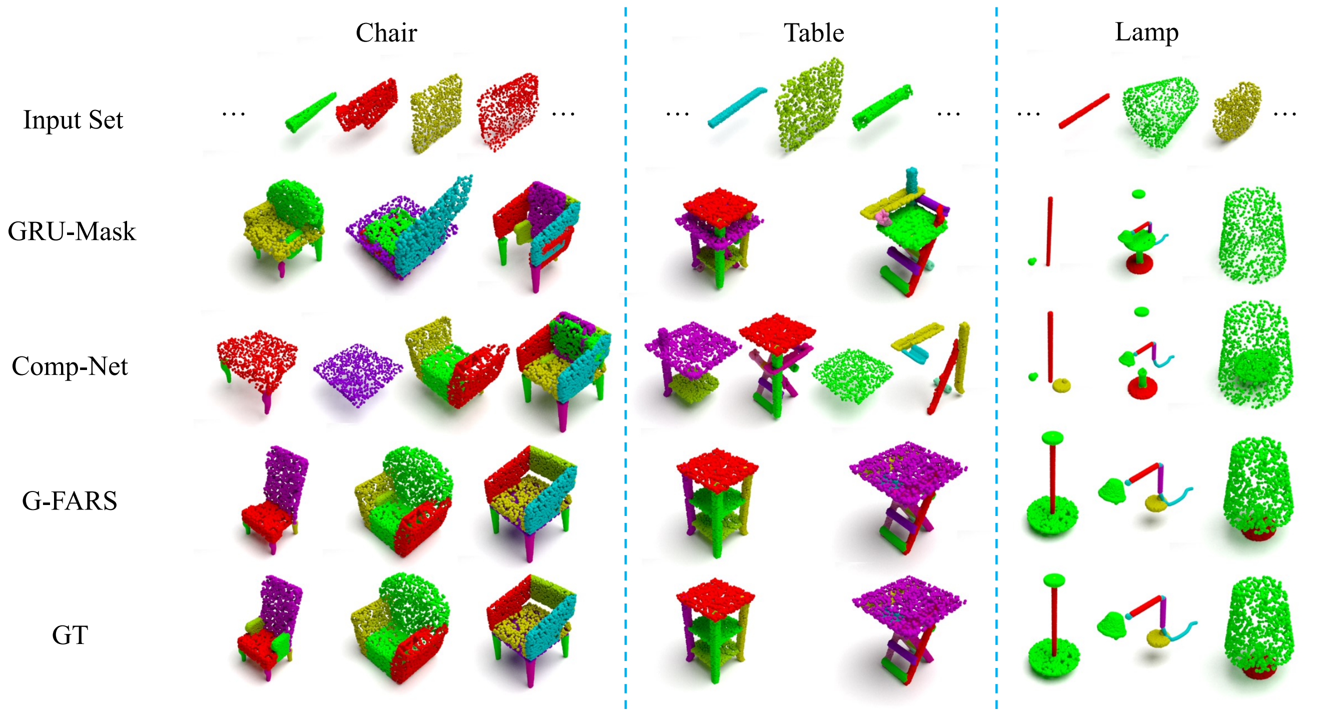

Figure 4. The qualitative comparisons. To intuitively demonstrate the effect of grouping, we rotate and translate all the parts by using their ground truth poses after the grouping procedure. The results show that only G-FARS is able to correctly group the 3D parts. Some baselines even predict an incorrect number of groups (*e.g.*, Comp-Net). Due to the page limit, we here only present the results of G-FARS and the two most competitive baselines (*i.e.*, GRU-Mask and Comp-Net). We have included the full comparison and an additional qualitative comparison figure in the supplementary materials.

About hyper-parameters Our framework contains many hyper-parameters. For the sampler and the sampling steps, we discuss them in Table 3. For other details about hyper-parameters, please refer to our supplementary.

5.5. Comparisons and Discussions

We present the quantitative comparisons in Table 1. Overall, our framework G-FARS outperforms other baselines by a large margin on all datasets. This proves the effectiveness of our proposed G-FARS. We also demonstrate the qualitative results in Fig. 4 (the full comparison is in the supplementary). To intuitively show our performance of the grouping results, we rotate and translate the parts in all the groups with their ground truth poses after the grouping procedure. **Please Note that the rotation and translation are only used for demonstration purpose. We DO NOT use the ground truth poses information in the grouping procedure or the training procedure.** The figure shows that our G-FARS is able to group most of the mixed part sets. However, the other baselines hardly manage to predict the correct groups accurately. Besides these experiments, we also demonstrate category mixing testing, and generalization testing to unseen categories in the supplementary. These tests further prove the effectiveness of G-FARS.

5.6. Ablation Study

To further prove the effectiveness of our algorithm, we conducted ablation studies. These experiments were carried out on the Chair dataset. Our ablation experiments comprise two parts: architecture ablation (see Table 2) and sampling ablation (see Table 3).

In the architecture ablation, we removed two key components from our G-FARS. For G-FARS w/o GF, we use a deterministic loss (*i.e.,* Binary Cross Entropy) instead of a score-matching loss (refer to Equation 4) [33] to train a network with the same architecture, adding a Sigmoid activation at the output. In the case of G-FARS w/o Graph, we replace the GNN with an MLP to learn $S_\theta^{\mathbf{c}}$. The results in Table 2 prove the effectiveness of both modules.

For the sampling ablation, we propose a variant of our G-FARS framework, where the sampler in our framework is replaced by Euler-Maruyama (EM) sampler [33]. The table shows that the performance of PC sampler is better than the performance of the EM sampler. The main reason why PC sampler outperforms the EM sampler is that the PC sampler uses both the numerical SDE solver and the Langevin MCMC as the corrector, while the EM sampler only contains numerical SDE solver. The Langevin MCMC

Metrics	Category	GRU-Mask	Comp-Net	G-FARS-CG	G-FARS-R	G-FARS-T	G-FARS
Precision ↑	Chair	0.641 / 0.608	0.696 / 0.629	0.592 / 0.532	0.586 / 0.522	0.6 / 0.539	**0.828 / 0.793**
	Table	0.647 / 0.601	0.758 / 0.694	0.62 / 0.555	0.608 / 0.545	0.596 / 0.528	**0.848 / 0.811**
	Lamp	0.613 / 0.576	0.732 / 0.661	0.691 / 0.616	0.684 / 0.619	0.718 / 0.637	**0.751 / 0.711**
Recall ↑	Chair	0.67 / 0.652	0.687 / 0.674	0.449 / 0.441	0.476 / 0.473	0.452 / 0.446	**0.753 / 0.744**
	Table	0.689 / 0.667	0.694 / 0.677	0.448 / 0.439	0.463 / 0.453	0.492 / 0.484	**0.798 / 0.784**
	Lamp	0.657 / 0.633	0.641 / 0.634	0.547 / 0.531	0.568 / 0.543	0.509 / 0.491	**0.728 / 0.71**
F1 Score ↑	Chair	0.651 / 0.629	0.678 / 0.651	0.499 / 0.483	0.512 / 0.496	0.503 / 0.488	**0.781 / 0.768**
	Table	0.662 / 0.632	0.712 / 0.685	0.507 / 0.49	0.514 / 0.495	0.525 / 0.505	**0.814 / 0.797**
	Lamp	0.626 / 0.603	0.666 / 0.647	0.592 / 0.571	0.603 / 0.578	0.574 / 0.555	**0.73 / 0.711**

Table 1. The quantitative comparisons among all the algorithms. We show both the single set average (before the slash) and overall average (after the slash) results in the table.

approach can help the algorithm to reduce the error produced by the numerical SDE solver [33]. We also test the best sampling steps for both samplers, and we find that the best sampling steps for PC sampler is 500.

	Precision	Recall	F1 Score
w/o GF	0.706 / 0.568	0.317 / 0.316	0.386 / 0.406
w/o Graph	0.578 / 0.52	0.479 / 0.472	0.512 / 0.495
G-FARS	**0.828 / 0.793**	**0.753 / 0.744**	**0.781 / 0.768**

Table 2. The results of architecture ablation. We show both the single set average (before the slash) and overall average (after the slash) results in the table.

Sampler	Step	Precision	Recall	F1 Score
EM	100	0.659 / 0.607	0.566 / 0.552	0.599 / 0.578
	200	0.825 / 0.791	0.719 / 0.711	0.76 / 0.749
	300	0.814 / 0.779	0.724 / 0.716	0.758 / 0.746
	400	0.822 / 0.789	0.719 / 0.71	0.758 / 0.747
	500	0.825 / 0.795	0.716 / 0.708	0.76 / 0.749
	600	0.824 / 0.785	0.719 / 0.711	0.76 / 0.746
PC	100	0.658 / 0.608	0.568 / 0.559	0.6 / 0.582
	200	0.827 / 0.792	0.736 / 0.729	0.77 / 0.759
	300	0.816 / 0.782	0.734 / 0.724	0.764 / 0.752
	400	0.826 / 0.791	0.74 / 0.733	0.771 / 0.761
	500	0.828 / 0.793	**0.753 / 0.744**	**0.781 / 0.768**
	600	**0.833 / 0.797**	0.747 / 0.74	0.779 / 0.767

Table 3. The ablation study for samplers and sampling steps. The ablation is conducted on the Chair dataset. We show both the single set average (before the slash) and overall average (after the slash) results in the table.

5.7. Noisy Part Removal

We surprisingly find that our algorithm is able to achieve the application of noisy part removal in a zero-shot manner. We demonstrate this application in Fig. 5. Assume you have a set of parts which belongs to a chair. However, you carelessly mix some noisy parts into this set, and you want to remove these parts. Our framework G-FARS can achieve your goal. The framework can directly output the correct selection for the parts of your chair.

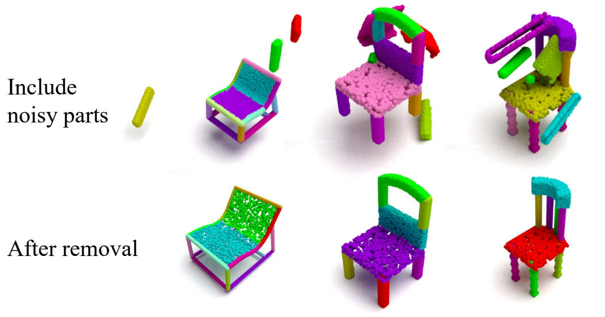

Figure 5. Three examples of noisy part removal.

6. Conclusion and Future Work

In conclusion, we have introduced a novel task termed ”3D part grouping”, which entails identifying all possible combinations from a mixed set. To address this, we began by randomly mixing shapes from PartNet [25] to construct our training and testing datasets. Subsequently, we unveiled a unique framework named G-FARS to fulfill our grouping objective. We validated our method using a series of benchmarks, illustrating that our algorithm displays commendable performance on the introduced task.

Future Work One constraint of our study is its confinement to virtual environments. Moving forward, we aim to investigate the viability of our proposed technique in real-world settings. As an example, our algorithm could be integrated into a robotic system, allowing robots to discern all feasible combinations from a real mixed part set.

References

[1] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. *arXiv preprint arXiv:2112.03126*, 2021. 6

[2] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016. 2

[3] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005. 2

[4] Jing-Chao Chen. Dijkstra's shortest path algorithm. *Journal of formalized mathematics*, 15(9):237–247, 2003. 2

[5] Peng Chen, Chaojun Xu, Zhigang Ma, and Yaqiang Jin. A mixed samples-driven methodology based on denoising diffusion probabilistic model for identifying damage in carbon fiber composite structures. *IEEE Transactions on Instrumentation and Measurement*, 2023. 6

[6] Junfeng Cheng, Mingdong Wu, Ruiyuan Zhang, Guanqi Zhan, Chao Wu, and Hao Dong. Score-pa: Score-based 3d part assembly. *arXiv preprint arXiv:2309.04220*, 2023. 3

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 3

[8] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006. 2

[9] Stephanie Forrest. Genetic algorithms. *ACM computing surveys (CSUR)*, 28(1):77–80, 1996. 2

[10] Mitsuo Gen and Lin Lin. Genetic algorithms and their applications. In *Springer handbook of engineering statistics*, pages 635–674. Springer, 2023.

[11] Shifen Han and Li Xiao. An improved adaptive genetic algorithm. In *SHS Web of Conferences*, page 01044. EDP Sciences, 2022. 2

[12] V Haripriya et al. The performance optimization of approximate minimum spanning tree for the different mobility model. In *2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pages 1–6. IEEE, 2023. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992. 2

[15] Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas J Guibas, Hao Dong, et al. Generative 3d part assembly via dynamic graph learning. *Advances in Neural Information Processing Systems*, 33:6315–6326, 2020. 1, 3

[16] Nadira Jasika, Naida Alispahic, Arslanagic Elma, Kurtovic Ilvana, Lagumdzija Elma, and Novica Nosovic. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In *2012 proceedings of the 35th international convention MIPRO*, pages 1811–1815. IEEE, 2012. 2

[17] Jehn-Ruey Jiang, Hsin-Wen Huang, Ji-Hau Liao, and Szu-Yuan Chen. Extending dijkstra's shortest path algorithm for software defined networking. In *The 16th Asia-Pacific Network Operations and Management Symposium*, pages 1–4. IEEE, 2014.

[18] Donald B Johnson. A note on dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20(3):385–388, 1973. 2

[19] Fereshteh Karimi, Mohammad Bagher Dowlatshahi, and Amin Hashemi. Semiaco: A semi-supervised feature selection based on ant colony optimization. *Expert Systems with Applications*, 214:119130, 2023. 2

[20] R Kavitha, D Kiruba Jothi, K Saravanan, Mahendra Pratap Swain, José Luis Arias Gonzáles, Rakhi Joshi Bhardwaj, Elijah Adomako, et al. Ant colony optimization-enabled cnn deep learning technique for accurate detection of cervical cancer. *BioMed Research International*, 2023, 2023. 2

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

[22] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4583–4592, 2021. 6

[23] Tiange Luo, Kaichun Mo, Zhiao Huang, Jiarui Xu, Siyu Hu, Liwei Wang, and Hao Su. Learning to group: A bottom-up framework for 3d part discovery in unseen categories. *arXiv preprint arXiv:2002.06478*, 2020. 6

[24] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021. 2

[25] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019. 2, 5, 8

[26] Shentong Mo, Enze Xie, Ruihang Chu, Lewei Yao, Lanqing Hong, Matthias Nießner, and Zhenguo Li. Dit-3d: Exploring plain diffusion transformers for 3d shape generation. *arXiv preprint arXiv:2307.01831*, 2023. 6

[27] Abhinav Narayan, Rajendra Nagar, and Shanmuganathan Raman. Rgl-net: A recurrent graph learning framework for progressive part assembly. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 78–87, 2022. 1, 3

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3, 5

[29] KAFA Samah, Burairah Hussin, and ASH Basari. Modification of dijkstra's algorithm for safest and shortest path during emergency evacuation. *Applied Mathematical Sciences*, 9(31):1531–1541, 2015. 2

[30] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 1

[31] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, pages 9558–9568. PMLR, 2021. 6

[32] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 1

[33] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 3, 4, 5, 7, 8

[34] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. *computer*, 27(6):17–26, 1994. 2

[35] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 6

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 6

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 3, 4

[39] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020. 6

[40] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019. 1

[41] Rufeng Zhang, Tao Kong, Weihao Wang, Xuan Han, and Mingyu You. 3d part assembly generation with instance encoded transformer. *IEEE Robotics and Automation Letters*, 7(4):9051–9058, 2022. 1, 3

[42] Xiangbing Zhou, Hongjiang Ma, Jianggang Gu, Huiling Chen, and Wu Deng. Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. *Engineering Applications of Artificial Intelligence*, 114:105139, 2022. 2

[43] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017. 6