Learning a Causal Transition Model for Object Cutting

Zeyu Zhang[1,2*] Muzhi Han[1,2*] Baoxiong Jia[1,2]
Ziyuan Jiao[1,2] Yixin Zhu[3] Song-Chun Zhu[2,3,4] Hangxin Liu[2†]

***Abstract*— Cutting objects into desired fragments is challenging for robots due to the spatially unstructured nature of fragments and the complex one-to-many object fragmentation caused by actions. We present a novel approach to model object fragmentation using an attributed stochastic grammar. This grammar abstracts fragment states as node variables and captures causal transitions in object fragmentation through production rules. We devise a probabilistic framework to learn this grammar from human demonstrations. The planning process for object cutting involves inferring an optimal parse tree of desired fragments using the learned grammar, with parse tree productions corresponding to cutting actions. We employ Monte Carlo Tree Search (MCTS) to efficiently approximate the optimal parse tree and generate a sequence of executable cutting actions. The experiments demonstrate the efficacy in planning for object-cutting tasks, both in simulation and on a physical robot. The proposed approach outperforms several baselines by demonstrating superior generalization to novel setups, thanks to the compositionality of the grammar model.**

I. Introduction

Representing object states and understanding how they change with actions are fundamental for robots to manipulate the physical world. In the literature, the primary focus is restricted to *rigid* objects whose states can only be altered spatially, represented with reconstructed 3D geometry [1–3], estimated 6D poses [4, 5], semantic keypoints [6, 7], or extracted appearance features [8, 9]. Recently, *articulated* object understanding in terms of kinematics estimation [10, 11] and part-level object modeling [12–14], and *deformable* object understanding empowered by physics-based simulation [15, 16] further expand a robot's manipulation capabilities towards handling drastic appearance and geometry changes of objects. However, either a rigid, articulated, or deformable object can be treated as a single whole object or a fixed collection of rigid parts when manipulated by a robot; modeling objects with topology changes, *i.e.*, object fragmentation in a cutting task, is still largely unexplored.

The challenges of modeling object fragmentation are twofold: (i) An object or its fragments exhibit considerable variation in terms of their *configurations* (*i.e.*, the layout, fragment number, pose, and shape of each fragment) during fragmentation. (ii) An object fragmentation process intrinsically involves one-to-many transitions (*i.e.*, an object breaking into multiple fragments), and there are many ways an object might be potentially fragmented. As a result, it is nontrivial to find a proper state representation, hindering the direct employment of methods such as neural networks [9, 17], probabilistic graphical models [18, 19], symbolic logic [20, 21], *etc.*, to model such complex causal transitions during fragmentation. To overcome these challenges, a desired state representation should be reconfigurable and extendable to account for the drastic variations in object fragmentation while being abstract enough to reduce the number of possible transitions for efficient planning.

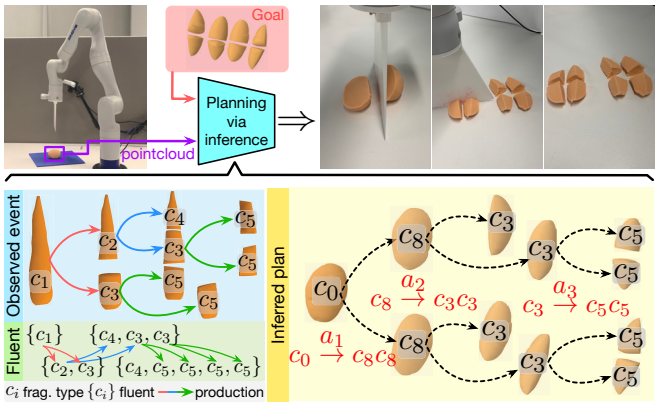

Fig. 1: **Planning for object cutting with a stochastic grammar of object fragmentation.** The grammar reveals the underlying fluent space of object fragmentation and captures causal transitions in a compositional manner with production rules. An observed fragmentation process is represented as a parse tree derived from the grammar; planning for object cutting is to infer an optimal parse tree that describes the desired fragmentation. Observing cutting a carrot could support planning actions for cutting a potato into the same by sharing the production rule $c_3 \to c_5 c_5$.

In this paper, based on the stochastic grammar model, we develop a *fluent* notation [22] to represent the state of an object–or that of its fragments–during cutting and derive a *fluent space* to describe the possible fragmentation (*i.e.*, the states and their causal transitions). Being successful in modeling scenes [23–25] and dynamic events [19, 26, 27], a stochastic grammar consists of a set of production rules that generate terminal or non-terminal variables from existing non-terminal ones, akin to the process of an object breaking into pieces—the original object generates newly appeared fragments. Specifically, the *grammar* itself incorporates all possible states an object may finally be as fragmentation repeats, and the *production rules* of the grammar indicate all valid one-to-many transitions. Together, they form the fluent space of object cutting. Furthermore, each *parse tree* derived from the grammar reflects a specific fragmentation process

* Zeyu Zhang and Muzhi Han contributed equally to this work. Emails: {zeyuzhang, muzhihan}@ucla.edu.
† Corresponding author. Email: liuhx@bigai.ai.
[1] Center for Vision, Cognition, Learning, and Autonomy (VCLA), Statistics Department, UCLA.
[2] National Key Laboratory of General Artificial Intelligence, Beijing Institute for General Artificial Intelligence (BIGAI).
[3] Institute for Artificial Intelligence, Peking University.
[4] Department of Automation, Tsinghua University.

produced by a sequence of cutting actions, whose *terminal nodes* correspond to fragments in the resulting configuration and collectively define the resulting fluent.

Fig. 1 presents the proposed stochastic grammar that models the object fragmentation process of cutting a carrot into chunks and supports planning cutting actions for a novel situation of cutting a potato into the same. We extract shape features for fragments and cluster them to obtain a much smaller set of variables to represent fluents and to induce production rules that describe the causal transitions between fluents. Crucially, the cluster number is determined such that the resulting grammar seeks to reduce its complexity by having fewer types of variables while preserving the necessary discriminability of fragments for consistency of transitions. More importantly, grammar's recursive and compositional nature allows us to model the fluent and fluent space compactly and flexibly and achieve better generalization.

In the experiments, we collect a dataset of human cutting demonstrations in simulation, from which we induce the grammar model and learn to generate action parameters for cutting. During the test phase, we demonstrate the efficacy of the proposed grammar-based representation and planning method on a series of object-cutting tasks, including those under novel setups. A preliminary real robot experiment also shows that our method can be applied to real-world object-cutting scenarios.

A. Related Work

Planning a sequence of actions to alter objects' states towards a goal is a long-standing problem in robotics and artificial intelligence. Task planning [28] efficiently searches for a sequence of discrete actions to reach the goal based on a known planning domain, usually defined in Planning Domain Definition Language (PDDL) [29]. While this approach provides a general and practical solution, it is limited to a fixed number of objects and relies on known transition models and hand-crafted state and action abstractions [21, 30–32].

An alternative approach is Model-based Reinforcement Learning, which effectively learns a transition model from interaction data, potentially with a learned state representation [17, 33, 34]. While this method achieves impressive performance, it requires extensive exploration and may not generalize well in complex scenarios.

We adopt an approach inspired by Task and Motion Planning (TAMP), accommodating a changing number of objects, while learning a stochastic grammar model to represent an abstract planning domain for object cutting. Notably, the production rules in the grammar model effectively bridge the gap between planning discrete cutting actions and generating continuous action parameters (see Sec. III-C).

B. Overview

The remainder of this paper is organized as follows. Sec. II formally models the object fragmentation process in object cutting using stochastic grammar and provides insights into learning such a model from human demonstrations. Sec. III formulates the plan-to-cut problem as probabilistic inference, leveraging the learned grammar model, and presents an algorithm for online planning. Furthermore, in Sec. IV, we demonstrate the efficacy of our method through a series of experiments, including novel scenarios. Finally, we conclude the paper and discuss future research directions in Sec. V.

II. Modeling Object Fragmentation with an Attributed Stochastic Grammar

An object fragmentation process $r_o : \Omega_o \to \Omega_o$ transforms a set of object fragments $\mathcal{I}^{\text{pre}} \in \Omega_o$ into another set of fragments $\mathcal{I}^{\text{post}} \in \Omega_o$, where $\mathcal{I} = \{o_i\}$ represents the configuration of the object fragments, $1 \leq |\mathcal{I}^{\text{pre}}| \leq |\mathcal{I}^{\text{post}}|$, and o_i represents an initial whole object or a fragment by its shape (*e.g.*, point cloud), pose, *etc*. Considering the complex nature of Ω_o, where each fragment could vary in shape, we instead regard some fragments $o_i \in \mathcal{I}$ as the same type $c_j \in C$ via clustering, where $S = \{c_j\}$ defines an object fluent of the configuration $\mathcal{I}$. As such, we obtain a simplified fluent space $\Omega_s = \{S\}$ that depicts a fragmentation process $r_s : \Omega_s \to \Omega_s$ with better abstraction.

A. Grammar representation of object fragmentation

We adopt an attributed stochastic grammar [35] to model causal transitions in object fragmentation, where terminal variables with their attributes represent the configuration of fragments, and production rules capture the valid causal transitions that an object breaks into multiple fragments. Formally, the attributed stochastic grammar is defined by a 5-tuple $\mathcal{G} = \langle V_{NT}, V_T, v_S, R, \mathbb{P} \rangle$, where $v_{NT} \in V_{NT}$ is a non-terminal variable that denotes a fragment type $c \in C$, $v_T \in V_T$ is a terminal variable that denotes a fragment type $c \in C$ with pose $q \in SE(3)$ and shape feature z as its attributes, v_S is the start symbol, $\mathbb{P}$ is the probability of the production rules defined over the grammar, and $r_i \in R$ is the production rule $r_i : V_{NT} \to (V_{NT} \cup V_T)^*$, where $(\cdot)^*$ is the Kleene star operation, enabling a production rule to describe an arbitrary fragmentation within the domain of $V_{NT} \cup V_T$. A fluent S is defined by terminals of a parse tree pt generated from $\mathcal{G}$, and the fluent space is defined by $\Omega_s = L(\mathcal{G})$, where $L(\mathcal{G})$ represents the set of all possible fluents generated by $\mathcal{G}$. Intuitively, a parse tree pt derived from $\mathcal{G}$ represents a plausible fragmentation sequence: the collection of terminals corresponds to the resulting fragments, and the non-terminals indicate the intermediate fragments in the past that subsequently fragment into the final configuration due to the sequence of applied productions (*i.e.*, cutting actions).

B. Grammar induction from human demonstrations

We propose to learn the stochastic grammar from object-cutting sequences generated by human demonstrations; please refer to Sec. IV-A for details of data collection.

Corpus generation: We extract a shape feature z for an object or fragment $o_i \in \mathcal{I}$ using a pre-trained point cloud encoder (see Sec. III-D), and cluster all features $\{z\}$ into k fragment types $\{c\}$. Then a corpus $\mathcal{D}_c^k = \left\{ c_i^{\text{pre}} \to \{c_{i,j}^{\text{post}}\} \right\}$ is obtained by recording the fragment type before and after each cutting action.

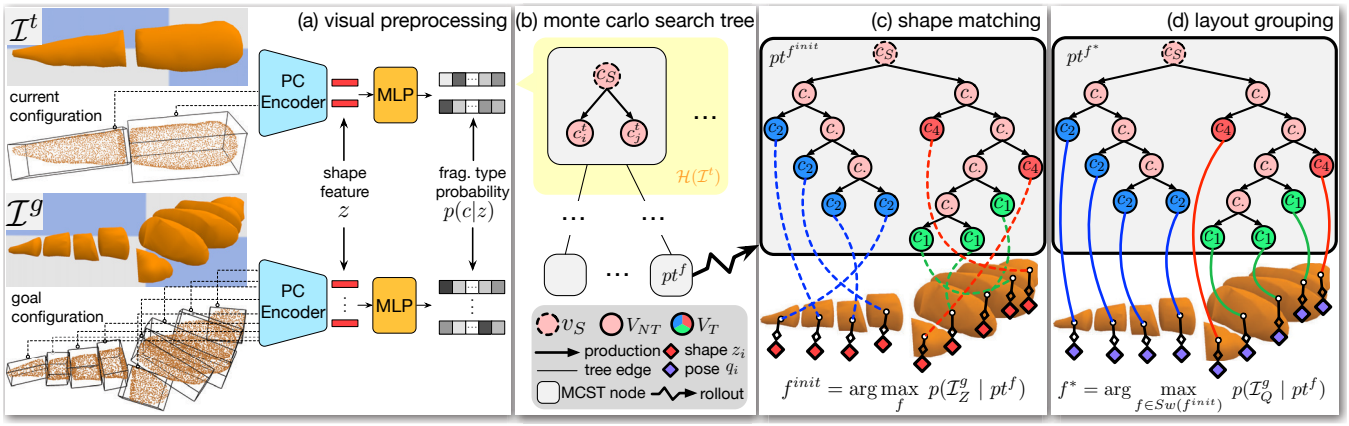

Fig. 2: **An illustration of the inference process to obtain an optimal parse tree pt^* through MCTS.** (a) Given fragment point clouds in the current or goal configuration, we extract a shape feature for each fragment with a pre-trained point cloud encoder and process it with an MLP to classify the fragment type $p(c|z)$ (the vector shows probability in greyscale). (b) We show an example of a Monte Carlo search tree where the state of a search node is a parse tree derived from the grammar. The expansion of a search node is to apply production rules to its parse tree. The yellow region $\mathcal{H}(\mathcal{I}^t)$ is a set of search nodes whose states (*i.e.*, parse trees) are sampled from fragments in $\mathcal{I}^t$ according to $p(c|z)$. (c-d) To evaluate rollout results, we find the best assignment that grounds each terminal node to a fragment in $\mathcal{I}^g$. The dotted lines in (c) represent an optimal assignment that maximizes the shape matching likelihood in Eq. (5), which is further refined to maximize the layout grouping likelihood in Eq. (6), shown in solid lines in (d).

A critical question is how to determine the proper number of fragment types k to reduce grammar complexity while maintaining sufficient discriminability among fragments. We solve it by balancing the data likelihood and model complexity in grammar induction; see the details below.

Grammar induction: Given corpus $\mathcal{D}_c^k$, we use Maximum a Posteriori (MAP) estimation to induce an optimal grammar:

$$\begin{aligned}\mathcal{G}^* &= \operatorname*{argmax}_{\mathcal{G}^k} p(\mathcal{D}_c^k \mid \mathcal{G}^k)\, p(\mathcal{G}^k) \\ &= \operatorname*{argmax}_{\mathcal{G}^k} \underbrace{\prod_{(\alpha_i \to \beta_i) \in \mathcal{D}_c^k} p(\alpha_i \to \beta_i \mid \mathcal{G}^k)}_{\text{data likelihood}} \cdot \underbrace{e^{\gamma |\mathcal{G}^k|}}_{\text{model prior}}, \end{aligned} \tag{1}$$

where $\alpha_i \to \beta_i$ is the i-th production in $\mathcal{D}_c^k$, γ a scalar coefficient, $|\mathcal{G}^k|$ the model size only depending on k, and $p(\alpha_i \to \beta_i \mid \mathcal{G})$ the branching probability of the production $\alpha_i \to \beta_i$ defined in $\mathbb{P}$.

We adopt an iterative non-parametric clustering approach, similar to DP-means [36], to solve for $\mathcal{G}^*$ in Eq. (1) by alternating two steps: search for a better k, and estimate the best production rules. With a fixed k, the best production rule probability aligns with the frequency of each alternative choice [23]:

$$p(\alpha \to \beta_i) = \#(\alpha \to \beta_i) / \sum_{j=1}^{n(\alpha)} \#(\alpha \to \beta_j), \tag{2}$$

where $\#(\alpha \to \beta)$ is the number of productions following $\alpha \to \beta$ in the corpus, and $n(\alpha)$ is the number of productions whose left-side (the non-terminals) is α. For ease of planning with $\mathcal{G}^*$, we also fit a classifier on the clustered fragments to model $p(c|z)$, the probability of a fragment's type c given its shape feature z.

III. Planning for Object Cutting

We aim to plan a sequence of cutting actions to achieve the goal configuration $\mathcal{I}^g$ from an initial configuration of fragments $\mathcal{I}^t$. Each cutting action involves cutting one object or fragment using a 3D cutting plane represented as $\boldsymbol{\pi} = [\boldsymbol{n}^T, d]^T \in \mathbb{R}^4$, where $\|\boldsymbol{n}\|_2 = 1$ is a unit plane normal vector, and $\boldsymbol{n}^T \cdot \boldsymbol{v} + d = 0$ represents the cutting plane constraint. We represent cutting planes in the canonical frame of the fragment to cut.

The planning problem is transformed into inferring an optimal parse tree of desired fragments given the learned grammar model that captures all possible causal transitions in object cutting. The planning is solved online using MCTS, detailed in Sec. III-B. Each production in the parse tree corresponds to a cutting action, and given the inferred parse tree, we generate the cutting plane $\boldsymbol{\pi}$ for each action with a sampling-based method (see Sec. III-C).

A. The posterior probability of parse trees

We derive the posterior probability of a parse tree pt, representing a fragmentation sequence or a plan of cutting actions, given the goal configuration $\mathcal{I}^g$ and the grammar $\mathcal{G}$. For each fragment in $\mathcal{I}^g$, we extract shape feature z and pose q, resulting in $\mathcal{I}_Z^g = \{z_i\}$ and $\mathcal{I}_Q^g = \{q_i\}$.

The posterior probability is given by:

$$p(pt \mid \mathcal{I}^g, \mathcal{G}) \propto \underbrace{p(pt \mid \mathcal{G})}_{\text{grammar prior}} \underbrace{p(\mathcal{I}_Z^g \mid pt)}_{\text{shape matching likelihood}} \underbrace{p(\mathcal{I}_Q^g \mid pt)}_{\text{layout grouping likelihood}}, \tag{3}$$

where the first term is the prior probability of the parse tree pt given $\mathcal{G}$, and the second and third terms describe the likelihood of observing $\mathcal{I}^g$ given pt in terms of fragment shape and pose. The overall posterior probability measures the alignment between pt generated by $\mathcal{G}$ and the goal configuration $\mathcal{I}^g$.

Grammar prior: The grammar prior captures possible causal transitions of object or fragment types. It is based on the learned production rules and branching probability:

$$p(pt \mid \mathcal{G}) = \prod_{(\alpha_i \to \beta_i) \in R^{pt}} p(\alpha_i \to \beta_i \mid \mathcal{G}), \tag{4}$$

where R^{pt} is the set of productions in the parse tree pt, and $p(\alpha_i \to \beta_i \mid \mathcal{G})$ is the conditional probability of choosing the production $\alpha_i \to \beta_i$ given the non-terminal node α_i.

Shape matching likelihood: The shape matching term evaluates the alignment between pt and the goal configuration $\mathcal{I}^g$ in terms of fragment geometry:

$$p(\mathcal{I}_Z^g \mid pt) = \prod_{i=1}^{N} p(z_i \mid c_i) \propto \prod_{i=1}^{N} p(c_i \mid z_i)\ p(z_i), \tag{5}$$

where c_i is the fragment type of the i-th terminal node in pt, z_i is the shape feature of the corresponding fragment, and N is the number of fragments in $\mathcal{I}_Z^g$. The prior $p(z_i)$ is a normal distribution fitted on the train set, and $p(c_i|z_i)$ is obtained from the classifier based on the shape feature z_i.

Layout grouping likelihood: The layout grouping term measures the alignment between pt and $\mathcal{I}^g$ in terms of fragment layout:

$$\begin{aligned} p(\mathcal{I}_Q^g \mid pt) &= \prod_{(\alpha_i \to \beta_i) \in R^{pt}} p\left(\beta_i \mid \alpha_i \to \beta_i\right) \\ &= \prod_{(\alpha_i \to \beta_i) \in R^{pt}} \prod_{v_j^{\beta_i} \in \beta_i} p\left(v_j^{\beta_i} \mid \alpha_i \to \beta_i\right), \end{aligned} \tag{6}$$

where $\alpha_i \to \beta_i$ is the i-th production in R^{pt}, α_i is the non-terminal node being expanded, and β_i represents the produced nodes from the rule. $v_j^{\beta_i}$ is the j-th produced node in β_i, and $p(v_j^{\beta_i}|\alpha_i \to \beta_i)$ gives the probability that production $\alpha_i \to \beta_i$ produces node $v_j^{\beta_i}$.

Assuming that the closer the fragments, the more likely they come from the same piece, we define the distribution $p(v_j^{\beta_i} \mid \alpha_i \to \beta_i)$ by an energy function:

$$p\left(v_j^{\beta_i} \mid \alpha_i \to \beta_i\right) = \frac{1}{Z} \exp\left(-\text{dist}(q^{\alpha_i}, q_j^{\beta_i})\right), \tag{7}$$

where Z is the partition function, $q_j^{\beta_i}$ the averaged pose of fragments in descendants under the node $v_j^{\beta_i}$, q^{α_i} the averaged poses of descendants in α_i, and $\text{dist}(\cdot,\cdot)$ the distance function that measures the distance between two poses. In practice, we calculate the Euclidean distance between the positions of two nodes and adopt dynamic programming when computing q^{α_i} and $q_j^{\beta_i}$ to avoid redundant computations.

B. Inference of the optimal parse tree

Given the current configuration $\mathcal{I}^t$, we aim to plan an optimal sequence of cutting actions that leads to a desired configuration $\mathcal{I}^g$. We formulate the planning process as inferring the optimal parse tree via an MAP estimate:

$$\begin{aligned} pt^* &= \underset{pt \in \mathcal{H}(\mathcal{I}^t)}{\text{argmax}}\ p(pt \mid \mathcal{I}^g, \mathcal{G}) \\ &= \underset{pt \in \mathcal{H}(\mathcal{I}^t)}{\text{argmax}}\ p(pt \mid \mathcal{G})\ p(\mathcal{I}_Z^g \mid pt, \mathcal{G})\ p(\mathcal{I}_Q^g \mid pt, \mathcal{G}), \end{aligned} \tag{8}$$

where $\mathcal{H}(\mathcal{I}^t)$ is a set of parse trees whose expansions from the start variable are sampled from $p(c|z)$ for each fragment in $\mathcal{I}^t$ after extracting shape feature z.

Since the computation of pt^* in Eq. (8) is intractable, we approximate pt^* via Monte Carlo Tree Search (MCTS) as shown in Fig. 2b. Initially, the algorithm starts with the root node of the search tree, which contains the start variable v_S of the grammar. The expansion and simulation step of MCTS is a process of applying feasible production rules (*i.e.*, possible causal transitions) on the parse tree of the search node, and the rollout results in each round are evaluated by measuring the objective function in Eq. (8). During the backpropagation step, we use the objective function value as the score to update the nodes on the path from the root to the rollout result. Finally, the best rollout result among all rounds in MCTS will be selected as pt^*.

To evaluate the objective function, we need to align every terminal node with a unique fragment in $\mathcal{I}^g$, as described in Sec. III-A. Hence, for the i-th round of rollout, we compute an optimal assignment function $f_i^* : V_T \to O$ that grounds each terminal node v_T in pt_i to a unique fragment o in $\mathcal{I}^g$, such that the resulting parse tree $pt_i^{f^*}$ maximizes the objective in Eq. (8) as well. Since the grammar prior term is irrelevant to the assignment, we have:

$$f^* = \underset{f}{\text{argmax}}\ p(\mathcal{I}_Q^g \mid pt_i^f)\ p(\mathcal{I}_Z^g \mid pt_i^f), \tag{9}$$

where pt_i^f denotes the parse tree whose terminal nodes are grounded to fragments in $\mathcal{I}^g$ by the assignment function f.

Since directly computing f^* is intractable (factorial to the number of fragments), we obtain an approximate solution in two steps: (i) Compute an assignment function f^{init} that maximizes the shape matching likelihood $p(\mathcal{I}_Z^g \mid pt^f)$ in Eq. (5); see the dotted lines in Fig. 2c. (ii) Refine f^{init} into f^* that maximizes the layout grouping likelihood $p(\mathcal{I}_Q^g \mid pt^f)$ in Eq. (6) while conserving the optimality obtained in the previous step; see solid lines in Fig. 2d.

The first step formulates a linear assignment problem that can be solved in polynomial time using the Hungarian algorithm [37]. Then we adopt the simulated annealing algorithm [38] to maximize $p(\mathcal{I}_Q^g \mid pt^f)$, where we randomly swap the matched terminal nodes of two fragments. As we do not want to violate the established optimality of $p(\mathcal{I}_Z^g \mid pt^f)$, we only swap terminal nodes with the same fragment type.

C. Cutting plane generation

With an inferred optimal parse tree whose productions correspond to cutting actions, we generate cutting planes to make the actions executable. We model the cutting plane $\boldsymbol{\pi}$ of an action as a Gaussian Mixture Model (Gaussian Mixture Model (GMM)) with its parameters depending on the production rule r and the shape feature z of the fragment to cut. The GMM parameters are regressed using a two-layer Mixture Density Network [39], learned from human demonstration data. During planning, we compute GMM parameters with a forward pass of the neural network and sample cutting planes from the corresponding GMMs for

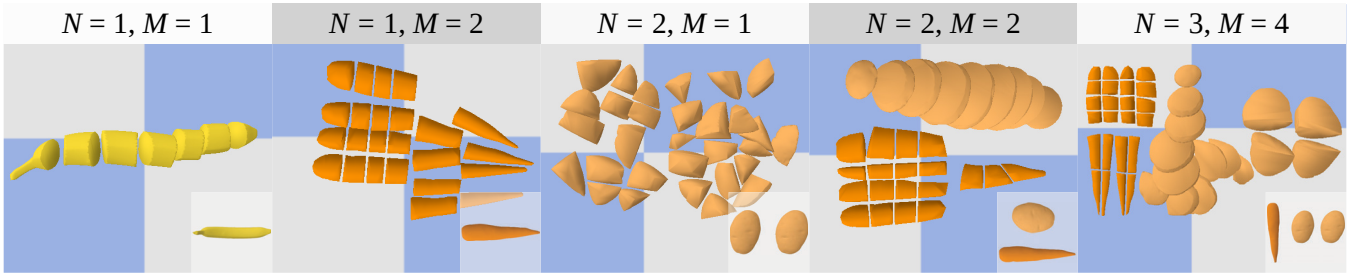

Fig. 3: **Examples of collected data with different levels of task complexity.** N is the initial number of objects, and M the number of fragment categories in the goal configurations. The bottom right corner of each subfigure shows the initial configuration.

execution. We generate cutting planes for each production separately (*i.e.*, cut one object or fragment at a time).

D. Implementation details

We define a consistent *canonical frame* for each fragment to match and distinguish between object fragments presented in different poses. The canonical frame is defined on a shape so that its projection along the z-axis is maximized, its projection along the x-axis is minimized, and its volume in the first octant is the largest. In practice, we compute the canonical frame of a fragment by principal component analysis.

We use a 17-dimensional vector $z = (z_{\text{shape}}, z_{\text{scale}})$ as the *shape feature*, where $z_{\text{shape}} \in \mathbb{R}^{16}$ encodes the normalized shape represented in its canonical frame with a point cloud encoder. The scalar $z_{\text{scale}} \in \mathbb{R}$ represents the scale. We adopt a naive encoder that processes all point coordinates and normals with a shared Multi-layer Perceptron (MLP) followed by an average pooling layer; the encoder is trained on all fragments in the train set of human cutting data following IMNet [40].

IV. Simulations and Experiments

We developed an object-cutting simulator based on BulletPhysics [41] to collect human demonstrations and test our method and baselines. Specifically, we implemented a Slice function in BulletPhysics that slices an object with a 3D plane. All methods were trained and evaluated in the simulator. Furthermore, we demonstrated that our proposed model, trained in the simulation environment, can effectively handle real-world object-cutting tasks a physical robot executes.

A. Data preparation

To collect human demonstrations, we asked human subjects to cut virtual objects presented in the simulator into one of the four fragment categories (*i.e.*, chunks, slices, cubes, and strips) or their combinations, using an intuitive Graphical User Interface (GUI) offered by the simulator. A cutting action is applied as a human subject specifies a 3D cutting plane by clicking two points on the GUI. We recorded each trail of demonstration as a sequence of fragment configurations and cutting actions; the ground-truth 3D geometry of each fragment and its pose can be directly retrieved from the simulator. A total of 110 object-cutting trails were collected and partitioned according to the initial number of objects N and the number of fragment categories in the goal configurations M; see Fig. 3 for some examples. We split the collected data, using a subset (40%) of $N=1, M=1$ as the train set and test on the remaining trails (*i.e.*, the rest of partition $N=1, M=1$ and partitions $N>1, M>1$).

B. Experimental setup

We test our method against various baselines to compute a sequence of cutting actions that reach a goal configuration $\mathcal{I}_g$ from a current fragment configuration $\mathcal{I}_t$, retrieved from test set trials. Instead of planning and executing all actions at once, we execute one action at a time and re-plan from the resultant configuration. This process repeats until we achieve the target number of fragments in the goal configuration. In our approach, we randomly select one production and execute the corresponding cutting action when multiple productions are available from the derived parse tree. Below, we describe the baseline methods and evaluation metrics that measure goal achievement.

1) Baselines: Given the success of learning-to-plan methods in handling complex state spaces, we design two baselines using a *state embedding* s, obtained by projecting shape features with an MLP followed by sum-pooling: (i) **Behavioral Cloning (BC)**: It learns a goal-directed policy with a two-layer MLP to mimic human actions in collected demonstrations. The policy predicts the cutting probability per fragment based on its shape feature z_i, current state embedding s, and goal state embedding s_g. The fragment to be cut is sampled based on the obtained probability, and cutting planes are sampled from a learned GMM conditioned on z_i and s_g. (ii) **Offline Deep Q Network (QNet)**: This model-free reinforcement learning approach is trained on logged data. We approximate a goal-conditioned action value function (Q function) using a two-layer MLP. During training, we assign sparse rewards to state-action pairs that achieve the goal, where an action is to choose a fragment to cut. Since the train set only contains positive demonstrations, we add a large margin loss term to encourage assigning a higher value to actions seen during training [42]. At test time, we select the best fragment to cut according to the Q function and sample a cutting plane identical to BC. Furthermore, we recruit (iii) **Human** participants to perform cutting tasks under the same setup, which serves as the performance upper bound.

TABLE I: **Quantitative results of planning for object cutting under various task setups.** We evaluate all methods using the best-matched Intersection over Union (IoU) and Human Rating (HR) on test sets with different N, M combinations, averaged across five runs; $\pm$ denotes standard deviation.

Task Setup		BC		QNet		Ours		Human	
		IoU	HR	IoU	HR	IoU	HR	IoU	HR
Seen	N=1, M=1	0.37 ± 0.11	2.19 ± 1.07	0.40 ± 0.16	2.14 ± 1.21	$\mathbf{0.58} \pm 0.08$	$\mathbf{4.32} \pm 0.77$	0.57 ± 0.03	4.48 ± 0.96
Unseen	N=1, M=2	0.35 ± 0.08	1.76 ± 0.87	0.32 ± 0.12	1.95 ± 0.87	$\mathbf{0.49} \pm 0.06$	$\mathbf{3.60} \pm 1.02$	0.62 ± 0.07	4.86 ± 0.35
	N=2, M=1	0.44 ± 0.08	1.64 ± 0.65	0.34 ± 0.16	1.19 ± 0.39	$\mathbf{0.56} \pm 0.03$	$\mathbf{3.69} \pm 0.89$	0.62 ± 0.09	4.83 ± 0.37
	N=2, M=2	0.42 ± 0.03	2.07 ± 0.86	0.29 ± 0.09	1.24 ± 0.43	$\mathbf{0.52} \pm 0.04$	$\mathbf{3.74} \pm 0.90$	0.56 ± 0.04	4.79 ± 0.56
	N=2, M=3	0.38 ± 0.03	1.73 ± 0.99	0.28 ± 0.09	1.52 ± 0.92	$\mathbf{0.52} \pm 0.03$	$\mathbf{3.21} \pm 0.86$	0.60 ± 0.04	4.81 ± 0.55
	N=3, M=4	0.38 ± 0.04	1.57 ± 0.62	0.22 ± 0.08	1.26 ± 0.49	$\mathbf{0.52} \pm 0.02$	$\mathbf{3.21} \pm 0.86$	0.56 ± 0.04	4.81 ± 0.55

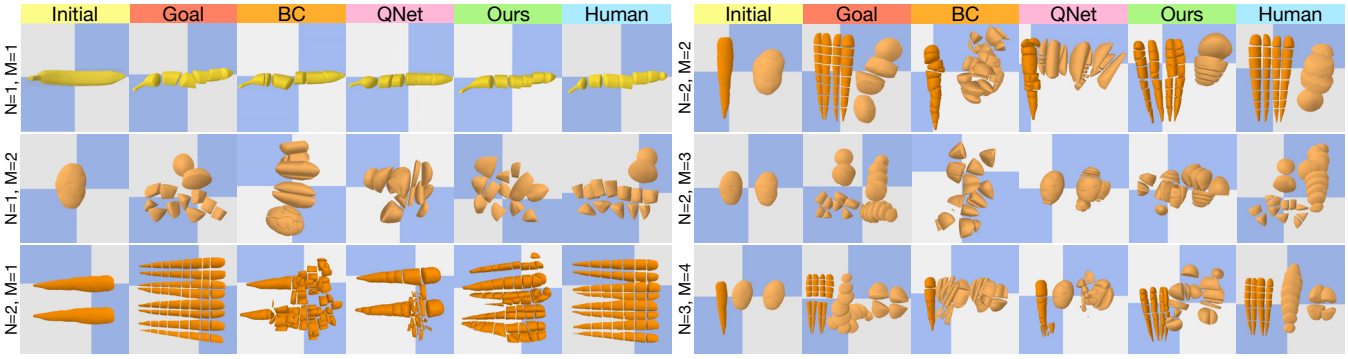

Fig. 4: **Planning cutting actions.** Qualitative results of planning cutting actions to achieve a desired set of fragments. Each row shows sample results of different methods under specific task setups.

2) Evaluation metrics: Due to geometric similarities between two fragment configurations despite different layouts, we design two metrics to evaluate how well the produced fragments match the goal configuration: (i) **Mean best-matched IoU**: This *objective* metric is the averaged IoU between the best-matched fragment pairs (see Fig. 5 dotted lines) in the produced final fragments and fragments in the goal configuration. We compute best-matched fragment pairs as a linear assignment problem using the Hungarian algorithm [37] in polynomial time. (ii) **Human Rating**: We recruit human participants to *subjectively* rate the fitness of the produced fragments against the goal. The rating ranges from 1 to 5 in discrete values, with higher scores indicating a better match.

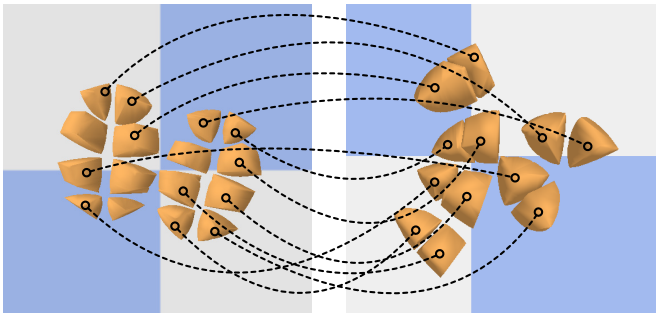

Fig. 5: **An example of best-matched fragments for evaluation.** Each dotted line connects a pair of best-matched fragments. Since the number of fragments is unbalanced between two sets of fragments, some fragments in the larger set remain unmatched.

C. Simulated results

We present the results of our method and the baselines under different test setups in Tab. I, and a qualitative comparison in Fig. 4. Our method outperforms BC and QNet in both objective and subjective metrics across all six setups, demonstrating superior generalization capabilities in scenarios involving more objects to cut ($N > 1$) and/or a composition of fragment types in the goal ($M > 1$). While BC performs on par with QNet in relatively simple task instances ($N = 1, M = 1$), it outperforms QNet in most generalization setups. Our method excels in learning a valid planning domain with a small amount of data and generalizing to novel task setups, thanks to the grammar model that effectively abstracts the fluent space and represents causal transitions in a compositional manner. The qualitative results and superior human rating of our method further demonstrate that the grammar models the fluents and causal transitions in a semantically meaningful way, aligning well with humans' mental abstraction of fragmenting objects.

D. Real-world robot experiment

We conduct a real robot experiment with a Kinova Gen 3 manipulator with a knife-like end-effector to cut an object into desired fragments. The goal configuration is given as fragment point clouds, while the robot observes a single point cloud of the current configuration by fusing outputs from two third-person-view depth cameras. The observed point cloud is segmented to produce per-fragment point clouds using a point cloud segmentation model [43] trained on the

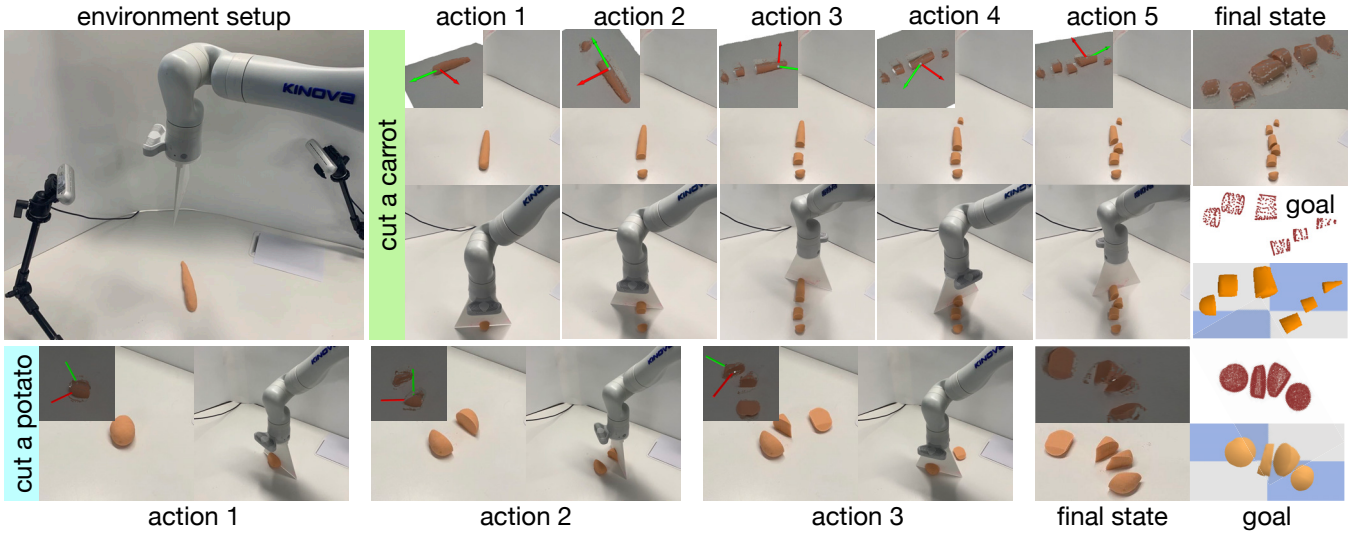

Fig. 6: **Real-world object cutting experiment.** Experiment on object cutting with a Kinova Gen 3 manipulator. The top-left figure illustrates the environment setup, and examples of the robot cutting a carrot and a potato are shown. The sequence of actions applied and the resulting fragments demonstrate a good alignment with the goal.

simulation dataset. We plan and execute a sequence of cutting actions following Sec. III. The grammar and neural networks are fully trained on simulated data as described in Sec. IV-C.

Fig. 6 presents the environment setup and keyframes of the robot executing planned actions to cut a carrot and a potato. The robot can generate meaningful cutting actions and produce fragments well-aligned with the goal. The experimental results demonstrate our method's ability to handle perception uncertainties and its potential in real-world object-cutting tasks.

V. Conclusion

In this work, we introduced a stochastic grammar of object fragmentation, which abstracts the state of fragments as terminal variables and accommodates causal transitions in object fragmentation through production rules. The proposed representation is powerful for modeling causal transitions in object fragmentation, enabling an agent to plan actions that cut an object into desired fragments. The planning problem is formulated as inferring an optimal parse tree of the desired configuration, where terminal nodes are ground to the produced final fragments and the productions indicate cutting actions. Our method achieved remarkable performance in planning for object-cutting tasks, even when applied to novel test setups with significant variations compared to the training set. Moreover, we conducted a preliminary real robot experiment utilizing a model trained in simulation, demonstrating the robustness of our method in the physical world. This work introduces a new perspective on object modeling and explores a new dimension of robot manipulation capability.

Limitations: While our method effectively abstracts the state space with fluents and models a simplified set of causal transitions, it still has some limitations. One notable limitation is the computational complexity of the MCTS method, especially when the goal configuration involves a large number of fragments, MCTS often requires considerable time to arrive at an optimal solution. Integrating Reinforcement Learning techniques with tree search, similar to Alpha-Go [44], could be a potential avenue to address this issue and further improve planning efficiency. Additionally, our method only permits cutting a single object in its object-centric frame. Extending the proposed approach to cut multiple objects simultaneously and incorporating interactions between objects during cutting remain an open challenge for future research.

Discussion: One of the key strengths of our approach is the ability to generalize to unseen scenarios and handle various cutting tasks with different numbers of objects and fragment categories. The grammar model, representing fragments as terminal variables and causal transitions through productions, allows the agent to abstract the object's state and plan for actions accordingly. This enables our method to effectively infer an optimal parse tree that guides the cutting process toward the goal configuration. Moreover, our approach is capable of learning the planning domain from a relatively small amount of simulated data, which makes it a practical and efficient solution for real-world object-cutting tasks on a physical robot.

Furthermore, the real-world experiment conducted with a Kinova Gen 3 manipulator showcased the applicability of our method in a physical setting. The robot was able to generate meaningful cutting actions and produce fragments that aligned well with the desired configuration. However, limitations in the modeling of complex contact dynamics could introduce uncertainty in the execution phase. Addressing these challenges and achieving more precise execution in real-world scenarios will be critical for practical deployment.

Acknowledgement: The authors thank Zaijin Wang (BIGAI) for setting up the real-world experiment platform. This work is supported in part by the National Key R&D

Program of China (2021ZD0150200) and the Beijing Nova Program.

REFERENCES

[1] M. Han, Z. Zhang, Z. Jiao, X. Xie, Y. Zhu, S.-C. Zhu, and H. Liu, "Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments," in *International Conference on Robotics and Automation (ICRA)*, 2021.

[2] M. Han, Z. Zhang, Z. Jiao, X. Xie, Y. Zhu, S.-C. Zhu, and H. Liu, "Scene reconstruction with functional objects for robot autonomy," *International Journal of Computer Vision (IJCV)*, vol. 130, no. 12, pp. 2940–2961, 2022.

[3] W. Agnew, C. Xie, A. Walsman, O. Murad, Y. Wang, P. Domingos, and S. Srinivasa, "Amodal 3d reconstruction for robotic manipulation via stability and connectivity," in *Conference on Robot Learning (CoRL)*, 2021.

[4] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[5] B. Wen and K. Bekris, "Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models," in *International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[6] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, R. Hadsell, L. Agapito, and J. Scholz, "S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency," in *Conference on Robot Learning (CoRL)*, 2021.

[7] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," in *International Symposium on Robotics Research (ISRR)*, 2022.

[8] C. Devin, P. Abbeel, T. Darrell, and S. Levine, "Deep object-centric representations for generalizable robot learning," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[9] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. Tenenbaum, and S. Levine, "Entity abstraction in visual model-based reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2020.

[10] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[11] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "Screwnet: Category-independent articulation model estimation from depth images using screw theory," in *International Conference on Robotics and Automation (ICRA)*, 2021.

[12] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas, "Captra: Category-level pose tracking for rigid and articulated objects from point clouds," in *International Conference on Computer Vision (ICCV)*, 2021.

[14] Z. Zhang, L. Zhang, Z. Wang, Z. Jiao, M. Han, Y. Zhu, S.-C. Zhu, and H. Liu, "Part-level scene reconstruction affords robot interaction," in *International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[15] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[16] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan, "Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools," in *International Conference on Learning Representations (ICLR)*, 2022.

[17] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning (ICML)*, 2019.

[18] H. Liu, C. Zhang, Y. Zhu, C. Jiang, and S.-C. Zhu, "Mirroring without overimitation: Learning functionally equivalent manipulation actions," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[19] M. Edmonds, F. Gao, H. Liu, X. Xie, S. Qi, B. Rothrock, Y. Zhu, Y. N. Wu, H. Lu, and S.-C. Zhu, "A tale of two explanations: Enhancing human trust by explaining robot behavior," *Science Robotics*, vol. 4, no. 37, p. eaay4663, 2019.

[20] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, IEEE, 2011.

[21] Z. Jiao, Z. Zhang, W. Wang, D. Han, S.-C. Zhu, Y. Zhu, and H. Liu, "Efficient task planning for mobile manipulation: a virtual kinematic chain perspective," in *International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[22] I. Newton, *The Method of Fluxions and Infinite Series: With Its Application to the Geometry of Curve Lines*. Nourse, 1736.

[23] S.-C. Zhu, D. Mumford, *et al.*, "A stochastic grammar of images," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2007.

[24] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu, "Holistic 3d scene parsing and reconstruction from a single rgb image," in *European Conference on Computer Vision (ECCV)*, 2018.

[25] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] S. Qi, B. Jia, S. Huang, P. Wei, and S.-C. Zhu, "A generalized earley parser for human activity parsing and prediction," *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 43, no. 8, pp. 2538–2554, 2020.

[27] Y. Zhu, Y. Zhao, and S.-C. Zhu, "Understanding tools: Task-oriented object modeling, learning and recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[29] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson, *et al.*, "Pddl| the planning domain definition language," tech. rep., Yale Center for Computational Vision and Control, 1998.

[30] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *International Conference on Robotics and Automation (ICRA)*, 2014.

[31] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[32] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *International Journal of Robotics Research (IJRR)*, vol. 37, no. 1, pp. 104–136, 2018.

[33] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[34] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[35] S. Park, B. X. Nie, and S.-C. Zhu, "Attribute and-or grammar for joint parsing of human pose, parts and attributes," *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 7, pp. 1555–1569, 2017.

[36] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via bayesian nonparametrics," in *International Conference on Machine Learning (ICML)*, 2012.

[37] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[38] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[39] C. M. Bishop, "Mixture density networks," tech. rep., Aston University, 1994.

[40] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2021.

[42] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Conference on Robot Learning (CoRL)*, 2020.

[43] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[44] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.