Title

Understanding Geometry and Topology Fluent for Robot Planning in Daily Scenes

Author
Zhang, Zeyu

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Understanding Geometry and Topology Fluent for Robot Planning in Daily Scenes

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zeyu Zhang

2023

ABSTRACT OF THE DISSERTATION

Understanding Geometry and Topology Fluent for Robot Planning in Daily Scenes

by

Zeyu Zhang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2023
Professor Song-Chun Zhu, Chair

This dissertation rethinks the problem of robot perception from an embodied agent's perspective: While the classic view focuses on perceiving the semantics and geometry of objects (*e.g.*, this piece of point cloud is a fridge), our new perspective emphasizes perceiving the *fluent* (a condition that can change over time) that provides actionable information for enabling an agent to reason about actions an object affords as well as the potential outcomes of actions for planning in daily scenes. We address this challenging problem by understanding (i) the *geometry fluent* that accounts for the changes in object pose, (ii) the *topology fluent* that accounts for the changes in object form, and (iii) the interconnection between the geometry and topology fluent. Considering the task of chopping garlic, one needs to transform whole garlic into minced and transport them from one place to another. An agent that only recognizes geometry and semantics can hardly accomplish such a task. Therefore, a scene reconstruction framework is proposed to reconstruct a functionally equivalent and interactive scene from RGB-D data streams to afford finer-grained interactions of geometry fluent. To further understand the interaction of topology fluent, a probabilistic framework is devised to induce an attributed stochastic grammar that models the space of object form

changes. This learned grammar and its probability model serve as a new indication of object status regarding topology fluent and are useful for planning downstream tasks. Finally, we study the interconnection between the geometry and topology fluent via a tool-use example where we learn the essential physical properties contributing to the effects of a tool-use event. By understanding potential actions in a scene, this dissertation aims to enable a robot to perceive the geometry and topology fluent and to plan their actions in daily scenes.

The dissertation of Zeyu Zhang is approved.

Yizhou Sun

Demetri Terzopoulos

Yingnian Wu

Song-Chun Zhu, Committee Chair

University of California, Los Angeles

2023

To my parents

For their endless love, support, and encouragement

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am indebted to my advisor, Prof. Song-Chun Zhu, for his tremendous supports throughout my Ph.D. program, for guiding and inspiring me to explore the cutting edge of Artificial Intelligence, and for providing me many great opportunities in my career development. I would also like to express my gratitude to Professors Yizhou Sun, Demetri Terzopoulos, and Ying Nian Wu for serving on my doctoral dissertation committee, not only for their time and infinite patience, but for their intellectual contributions and professional advices.

During my Ph.D. journey at UCLA, I have been fortunate enough to work with two great mentors: Dr. Yixin Zhu and Dr. Hangxin Liu, who guide me through the different stages of research, show me how to formulate ideas and design rigorous experiments, and instruct me how to publish scientic papers, down to each line and each figure, as well as presentations. I really appreciate their mentorship, and the work presented in this dissertation benefited enormously from their insightful discussions.

Thanks also go to a collection of remarkable collaborators at UCLA: Ziyuan Jiao, Muzhi Han, Baoxiong Jia, Dr. Xie Xu, Shuwen Qiu, Dr. Xin Jiang, Weiqi Wang, and many others, for our close collaborations, great friendships, and valuable discussions. A special thanks goes to Dr. Feng Gao, who helps us maintain the lab hardwares and equipments for years.

Finally, I dedicate this dissertation to my parents for their endless love, support, and encouragement.

Portions of this work were supported by ONR grant N00014-19-1-2153, ONR MURI grant N00014-16-1-2007, DARPA XAI grant N66001-17-2-4029, ARO grant W911NF-18-1-0296, and an NVIDIA GPU donation grant.

VITA

2017-2022	Graduate Research Assistant, UCLA, Dr. Song-Chun Zhu
2017-2019	M.S. in Computer Science, UCLA
2015-2016	Exchange Student, Arizona State University
2013-2017	B.S. in Computer Science, Hunan University, China

PUBLICATIONS

(* denotes joint first authors)

M. Han*, **Z. Zhang***, Z. Jiao, X. Xie, Y. Zhu, S.-C. Zhu, H. Liu. Scene Reconstruction with Functional Objects for Robot Autonomy. *International Journal of Computer Vision (IJCV)*, 2021, DOI: 10.1007/s11263-022-01670-0

Z. Zhang*, Z. Jiao*, W. Wang, Y. Zhu, S.-C. Zhu, H. Liu, Understanding Physical Effects for Effective Tool-use, *IEEE Robotics and Automation Letters (RA-L)*, 2022, DOI: 10.1109/LRA.2022.3191793

Z. Jiao, Y. Niu, **Z. Zhang**, S.-C. Zhu, Y. Zhu, H. Liu, Sequential Manipulation Planning on Scene Graph, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022

Z. Jiao*, **Z. Zhang***, W. Wang, D. Han, S.-C. Zhu, Y. Zhu, H. Liu, Efficient Task Planning for Mobile Manipulation: a Virtual Kinematic Chain Perspective, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021

Z. Jiao*, **Z. Zhang***, X. Jiang, D. Han, S.-C. Zhu, Y. Zhu, H. Liu, Consolidating Kinematic Models to Promote Coordinated Mobile Manipulations, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021

M. Han*, **Z. Zhang***, Z. Jiao, X. Xie, Y. Zhu, S.-C. Zhu, H. Liu. Reconstructing Interactive 3D Scenes by Panoptic Mapping and CAD Model Alignments. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

S. Qiu*, H. Liu*, **Z. Zhang**, Y. Zhu, S.-C. Zhu. Human-Robot Interaction in a Shared Augmented Reality Workspace. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

Z. Zhang, H. Liu, Z. Jiao, Y. Zhu, S.-C. Zhu. Congestion-aware Evacuation Routing using Augmented Reality Devices. *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

H. Liu*, **Z. Zhang***, Y. Zhu, S.-C. Zhu. Self-Supervised Incremental Learning for Sound Source Localization in Complex Indoor Environment. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

CHAPTER 1

Introduction

Perception of man-made environments and the objects within inevitably leads to the course of actions [Gib50, Gib66], which naturally form the basis for a human agent to interact with the environment and accomplish complex tasks. Crucially, what we "see" is much more than pixels and semantic labels [KR96]. Instead, we further "see" *how* to interact with them for our task purposes. Likewise, an embodied AI agent or a robot must possess a similar perceptual capability to achieve a wide range of task goals in the physical world. However, this critical perspective is mostly unexplored by prior scene reconstruction literature in computer vision or Simultaneous Localization and Mapping (SLAM) methods in robotics. Oftentimes, prior art only captures scenes' occupancy information and is evaluated primarily by reconstruction accuracy in the euclidean space. Without incorporating the *actionable* information—actions a semantic entity could afford and the associated physical constraints among entities—in a reconstructed scene, a robot can only perform relatively simple navigation or pick-and-place tasks, hindering its capability in planning and executing complex tasks with a long horizon.

On the other hand, modeling and understanding objects are the crux of computer vision and robot manipulation. Prior methods primarily focus on treating objects as a whole, which have made tremendous success recently by discriminating object shape (e.g., recognition) or tracking object pose (e.g., manipulation). However, objects can sometimes break into pieces (i.e., object fragmentation), violating the assumption of "object-as-a-whole". This common phenomenon has been largely neglected in recent literature.

To address these shortcomings in prior work, in this dissertation, we propose a new perspective that emphasizes perceiving the fluent (a condition that can change over time) that provides actionable information for enabling an agent to reason about actions an object affords as well as the potential outcomes of actions. We address this challenging problem by understanding (i) the geometry fluent that accounts for the changes in object pose, (ii) the topology fluent that accounts for the changes in object form, and (iii) the interconnection between the geometry and topology fluent.

First, in Chapter 2 we propose a scene reconstruction framework to reconstruct a functionally equivalent and interactive scene from RGB-D data streams, where the objects within are segmented by a dedicated 3D volumetric panoptic mapping module and subsequently replaced by part-based articulated CAD models to afford finer-grained robot interactions. The object functionality and contextual relations are further organized by a graph-based scene representation that describes the *geometry fluent* of the perceived scene. Additionally, such a graph-based representation can be readily incorporated into robots' action specifications and task definition, facilitating their long-term task and motion planning in the scenes. In Chapter 3 we further introduce a new perspective that performs planning in the geometry fluent space via a VKC.

To understand the topology fluent space, in Chapter 4, we model the events of object form changes using an attributed stochastic grammar model. A probabilistic framework is devised to induce such a grammar from observation; this learned grammar and its probability model serve as a new indication of object status during topology fluent changes. We further propose a probabilistic inference algorithm over the grammar model to perform planning and reasoning in the topology fluent space.

Finally, in Chapter 5, we study the interconnection between the geometry and topology fluent in a tool-use scenario. We present a robot learning and planning framework that learns the essential physical properties contributing to the effects of a tool-use event (*e.g.*, how a hammer cracks a walnut) and produces an effective tool-use strategy with the least joint efforts. Specifically, leveraging a Finite Element Method (FEM)-based simulator that reproduces fine-grained, continuous visual and physical effects given observed tool-use events, the essential physical properties contributing to the effects are identified through the proposed Iterative Deepening Symbolic Regression (IDSR) algorithm. We further devise an optimal control-based motion planning scheme to integrate robot- and tool-specific kinematics and dynamics to produce an effective trajectory that enacts the learned properties.

This dissertation is intended to provide a new perspective on robot perception, where perception is guided by the understanding of actions afforded in a scene. As such, the acquired geometry and topology fluent provide actionable information that enables a robot to reason about the potential outcomes of actions while planning for daily tasks in a scene and further have an intelligent robot reach a higher level of autonomy.

CHAPTER 2

Understanding the Geometry Fluent via a Contact Graph

In this chapter, we rethink the problem of scene reconstruction from an embodied agent's perspective: While the classic view focuses on the reconstruction accuracy, our new perspective emphasizes the underlying functions and constraints of the reconstructed scenes that provide *actionable* information for simulating *interactions* with agents. Here, we address this challenging problem by reconstructing a *functionally equivalent* and interactive scene from RGB-D data streams, where the objects within are segmented by a dedicated 3D volumetric panoptic mapping module and subsequently replaced by part-based articulated CAD models to afford finer-grained robot interactions. The object functionality and contextual relations are further organized by a graph-based scene representation that can be readily incorporated into robots' action specifications and task definition, facilitating their long-term task and motion planning in the scenes. In the experiments, we demonstrate that (i) our panoptic mapping module outperforms previous state-of-the-art methods in recognizing and segmenting scene entities, (ii) the geometric and physical reasoning procedure matches, aligns, and replaces object meshes with best-fitted CAD models, and (iii) the reconstructed functionally equivalent and interactive scenes are physically plausible and naturally afford actionable interactions; without any manual labeling, they are seamlessly imported to ROS-based robot simulators and VR environments for simulating complex robot interactions. The materials in this chapter have been published in [HZJ22].

2.1 Introduction

Having the *actionable* information in a scene is crucial for the training and testing of modern embodied AI agents [BCC20]. Existing research efforts are mainly devoted to develop simulation platforms that provide (i) photorealistic views (*e.g.*, Habitat [SKM19], RoboTHOR [DHH20]) for navigation, (ii) articulated and interactive objects (*e.g.*, iGibson [XSL20], SAPIEN [XQM20]) for interaction, and (iii) physical simulation engines (*e.g.* VRGym [XLZ19]) for fine-grained fluent changes. While the *actionable* information can be explicitly specified and embedded in the simulation setup, or be recognized from a physical scene using dedicated vision modules, such as part-based object pose estimation [LWY20], functionality [ZZ13] and affordance [MLZ16] recognition *etc.*, it is non-trivial to organize this information and unclear about how an agent could utilize such information for various tasks.

Take the scene in Fig. 2.1 as the example, wherein the robot is tasked to pick up a frozen meal from the fridge, microwave it, and serve it. The challenges of processing *actionable* information are three-fold. First, it needs to recognize the semantics and geometry information of objects (*e.g.*, this piece of point cloud is a fridge). Although typical semantic mapping and segmentation techniques can achieve this goal [HLS20, NSI19], a more robust and accurate approach is still in need to better handle the complexity in clustered real environments given a first-person-view RGB-D video stream. Second, mere semantics are inadequate to reflect the actions an object affords (*e.g.*, whether or how the fridge can be opened). While some existing work attempted to identify the associations between symbolic actions and objects [MTF15, LLK19] or the underlying the object's kinematics [SSB11, CD17, MB19], they are insufficient for robots to execute complex tasks with multiple steps at the motion level. Third, we quest for a more fundamental question: How to devise a scene representation with a succinct action specification and task definition to account for the action opportunities and the accumulated outcome of executed actions. Without addressing these challenges, a

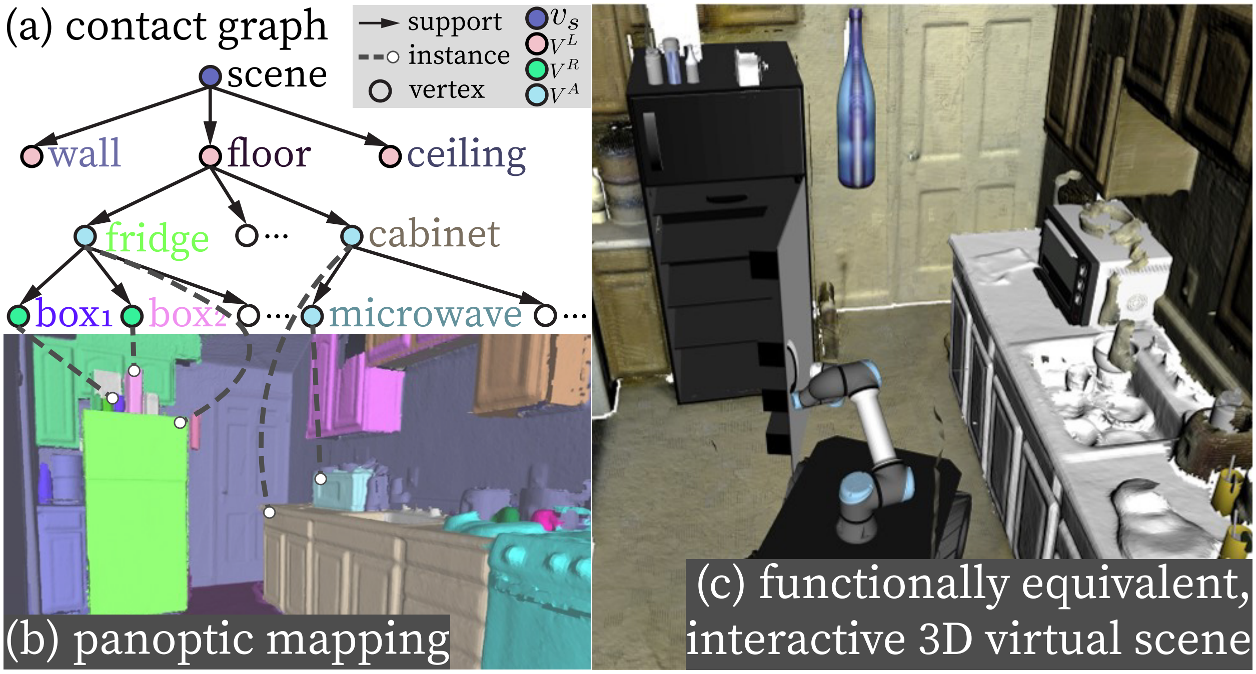


Figure 2.1: **The reconstruction of a functionally equivalent, interactive 3D scene.** (a) A contact graph is constructed by the supporting relations that emerged from (b) panoptic mapping. By reasoning their affordance, functional objects within the scene are matched and aligned with part-based interactive CAD models. (c) The reconstructed functionally equivalent scene enables a robot to simulate its task execution with comparable outcomes in the physical world.

robot can hardly plan for the given task or verify whether its plan is valid before executing in the physical world.

In this dissertation, we propose a new task of reconstructing *functionally equivalent* and interactive scenes by representing the *actionable* information of scene entities to support agents' planning and simulation. Here we argue that a scene's functionality is composed by the functions of objects within the scene. Therefore, the essence of a *functionally equivalent* scene is to preserve most objects' four characteristics with a decreasing propriety: (i) their semantic class and spatial relations with nearby objects, (ii) their affordance, *e.g.* what interactions they offer, (iii) similar geometry in terms of size and shape, and (iv) similar appearance. To address this new task, we devise a perception system with three unique components; see an illustration in Fig. 2.3:

A) A robust 3D volumetric panoptic mapping module, detailed in Section 2.3, accurately segments and reconstructs 3D objects and layouts in clustered scenes based on potentially noisy per-frame segmentation. The term "panoptic," introduced in [KHG19], refers to jointly segmenting *stuff* and *things* in semantic and instance levels. In this dissertation, we regard objects as *things* and layouts as *stuff*. This module produces a volumetric panoptic map using a novel per-frame panoptic fusion strategy and a global data fusion procedure performing data association, map integration and regularization; see Fig. 2.1b and Fig. 2.3a for examples of results.

B) A physical reasoning module, detailed in Section 2.4, replaces the potentially noisy and incomplete object meshes segmented from the panoptic map with functional (rigid or articulated) CAD models. This step is achieved by a ranking-based CAD matching and an optimization-based CAD alignment, which accounts for both geometric and physical constraints. We further introduce a global physical violation check to ensure that the resulting reconstructed interactive scene is physically plausible.

C) A contact graph cg **representation**, detailed in Section 2.2 and illustrated in Fig. 2.2, is constructed in accordance with the supporting and proximal relations among objects and imposes physical constraints as well as kinematic information for a robot's task execution. After retrieving *actionable* information annotated in CAD models, this novel representation indicates how an object can be moved or manipulated (*e.g.*, a table can be moved in 3D space) and how nearby objects would move correspondingly (*e.g.*, a box on the table would go through a similar transformation if not slid or tilted). The cg can be interpreted as and converted to a kinematic tree, which is updated following the robot's actions so that it can support long-horizon task and motion planning. As such, it serves as an ideal representation that bridges robot perception (scene reconstruction) with robot planning. Part of this work is published in [HZJ21]; comparing with it, this dissertation

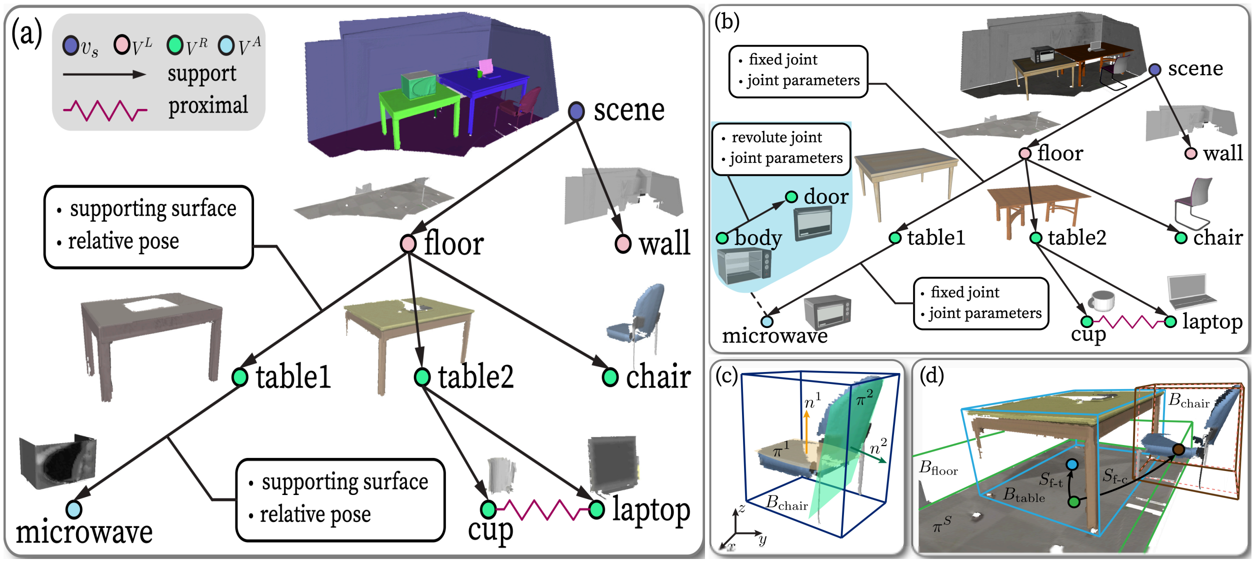


Figure 2.2: **3D scene representations and relations within.** (a) The contact graph representation. Each node denotes an object or a piece of layout, reconstructed and segmented as meshes from the RGB-D stream using the proposed panoptic mapping module. The directed edges indicate supporting relations—The parent node supports the child node. (b) The object meshes are replaced by best-fitted CAD models to create a functionally equivalent and physically plausible reconstructed scene. The directed edges and the constructed kinematic relations define the action space for robot planning. By updating the kinematic relations, various action effects can be easily integrated. (c) The supporting relations can further facilitate a reasoning process that refines (d) the 3D bounding box estimation. Initial: dashed line. Refined: solid line.

highlights the conversion from a sensed *cg* to the URDF, conducts experiments and analysis in real-world setting, and further evaluations including a new study of evaluating resulted *cg* using GED.

To our knowledge, ours is the first work that introduces a comprehensive system that reconstructs a full 3D scene from an embodied agent's perspective to provide *actionable* information for simulating robot interactions. It makes three major contributions:

1. We introduce a novel scene representation using a contact graph, whose structure is determined by the supporting and proximal relations among scene entities. It imposes physical constraints for a physically plausible scene and kinematic information that indicates whether and how an object can be interacted with. This contact graph representation is constructed and maintained for the scene reconstruction, and converted to a kinematic tree, which reflects the full geometric state of a scene and updates to keep track of every interaction. As such, our contact graph representation can facilitate the functionally equivalent scene reconstruction, as well as the robot learning and planning for complex long-horizon tasks.
2. Leveraging (i) local geometric similarity on the basis of relative sizes and surfaces of each object, and (ii) global physical constraints regarding the plausibility of stable support and non-penetration, we align rigid or articulated CAD models to object meshes to generate a physically plausible, fully interactive scene.
3. We develop a volumetric panoptic mapping module based on [GFN19], and introduce new designs to improve the accuracy in per-frame segmentation and the consistency in global data fusion. We show that this implementation is more robust against noisy input data and generates more accurate panoptic segmentation results, especially suitable for challenging and clustered indoor scenes.

2.1.1 Related Work

Scene datasets are crucial for providing supervisions of existing data-driven methods for a plethora of scene reconstruction and scene understanding tasks. In literature, the development of such datasets follows three stages. Early work, such as NYU-Depth [SHK12] and SUN RGB-D [SLX15], provides single view RGB-D images with densely annotated object segmentation, bounding boxes, *etc.* These types of 2.5D data are primarily designed to support recognition and prediction tasks in computer vision. In the second stage, datasets provide full 3D (in contrast to 2.5D) scene data in the form of annotated meshes for more holistic computer vision tasks [HPN16, CDF17, DCS17]. More recently, researchers start to construct synthetic scene datasets [YYT11, SYZ17, QZH18, JQZ18] to overcome the tedious and error-prone labeling process and obtain scene data at a much larger scale. Despite success in all three stages, they still fall short for robot learning or planning due to the lack of a proper means that converts a scanned or synthetic scene to an interactive one for robot task execution. In comparison, the proposed system can reconstruct interactive scenes from RGB-D streams and directly import them into simulators for robot training and testing of complex task execution.

To gather the scene semantics, modern **semantic mapping** [NSI19, GFN19, PHN19] and **object SLAM** [YS19a, MCB18] methods can retrieve object semantic segmentation, 6 DoF poses, and 3D bounding boxes during reconstruction. Physical cues, such as support and collision [YS19b, WSJ20, SCX20] and robot proactive actions [XHS15, LXS18], can be further integrated to better estimate and refine the scene semantics. In parallel, significant efforts have been made for object instance segmentation from point clouds [ZZC19]; *e.g.*, [YZW19] can segment an object with fine-grained part instances, and [PNH19] jointly perform semantic and instance segmentation. The above work, however, could only produce incomplete objects (in contrast to full 3D) due to confined viewpoints in the physical world, which prohibits the complex robot interaction and task execution in the reconstructed scenes. To alleviate this issue, researchers have recently attempted to **align CAD models** to these incomplete objects based on single RGB image [HQZ18, CHY19], single RGB-D image pair [GAG15, ZGL19], and scanned scene meshes [DCS17, ADD19, ADN19] to incorporate richer scene semantics. Following this trend, our system further introduces a physical reasoning procedure to align (part-based) CAD models to segmented objects to enable robot manipulation and interaction.

Devising an appropriate **scene representation** for scene reconstruction remains an open problem [CCC16]. Existing SLAM and semantic mapping approaches reviewed above oftentimes represent a reconstructed scene and its entities as sparse landmarks [PJ12, YS19a], surfels [MHD17, HLS20], volumetric voxels [GFN19, MCB18], or semantic objects [YS19a, MCB18]. Such a paradigm only provides geo-information of what and where to a robot without any actionable information for its interactions or planning. Meanwhile, graph-based representations for 3D scene further identify the hierarchical and relational structure among the scene entities [ZM07, ZZ11, ZZ13, ZZY15, HQX18, JQZ18, CHY19, AHG19, WDN20, RGA20], providing better structural and contextual information of the reconstructed scenes. In particular, [RGA20] explicitly incorporate actionable information to support robot planning, though limited to navigation and traversal tasks as the representation only models the connectivity between entity nodes. [RGA20] is also limited in that it is conducted in a simulated environment without accounting for real perception challenges. By leveraging the advantages of prior arts and addressing the shortcomings, the proposed system takes a real RGB-D stream as input and produces a contact graph representation based on the identified supporting relations among scene entities. This representation for scene reconstruction indicates how an entity can be interacted with and what the effect would be after an interaction, capable of supporting more complex manipulation planning.

2.2 Contact-Based Scene Representation

We devise a graph-based representation, *contact graph* cg, to represent a 3D indoor scene and the relations among scene entities. Formally, a contact graph $cg = (pt, E)$ contains (i) a parse tree (pt) that hierarchically organizes the scene entities [ZM07], and (ii) the proximal relations E among entities represented by undirected edges; see an example in Fig. 2.2a.

2.2.1 Representation

Scene Parse Tree $pt = (V, S)$ has been used to represent the hierarchical decompositional relations (*i.e.*, the edge set S) among entities (*i.e.*, the node set V) in various task domains, including 2D images and 3D scenes [ZM07, ZZ11, ZZ13, QZH18, JQZ18, HQZ18, HQX18, CHY19], videos and activities [ZZZ15, ZJZ16, QJH20, JCH20], robot manipulations [EGX17, LZS18, EGL19, LZZ19, ZZZ20], and theory of mind [YLF20]. In this paper, we adapt *pt* to represent supporting relations among entities instead of their decomposition. A *pt* is dynamically built and maintained during the reconstruction based on the identified supporting relations among segmented scene entities; for instance in Fig. 2.2a, the `table1` is the parent node of the `microwave`. Supporting relation is quintessential in scene understanding as it reflects the omnipresent physical plausibility; *i.e.*, if the table were moved, the microwave would move together with it. This perspective of physical common sense goes beyond occupancy information (*i.e.*, the geometric location of an object); in effect, it further provides actionable information and the potential outcome of actions for robot interactions and task executions in the scene.

Scene Entity Nodes $V = \{v_s\} \cup V^L \cup V^R \cup V^A$ include: (i) the scene node v_s, serving as the root of *pt*, (ii) layout node set V^L, including floor, ceiling, and the walls that bound the 3D scene, (iii) rigid object set V^R, wherein each object has no articulated part (*e.g.*, a table), and (iv) articulated object set V^A, wherein each object has articulated parts to be interacted for robot tasks (*e.g.*, fridge, microwave). Each non-root node $v_i = \langle o_i, c_i, M_i, B_i(\boldsymbol{p}_i, \boldsymbol{q}_i, \boldsymbol{s}_i), \Pi_i \rangle$ encodes a unique instance label o_i, a semantic label c_i, a full geometry model M_i (*e.g.*, a triangle mesh or a CAD model), a 3D bounding box B_i (parameterized by its center position $\boldsymbol{p}_i$, orientation $\boldsymbol{q}_i$, and size $\boldsymbol{s}_i$, all in $\mathbb{R}^3$), and a set of surface planes $\Pi_i = \{\boldsymbol{\pi}_i^k, k = 1 \cdots |\Pi_i|\}$, where a plane $\boldsymbol{\pi}_i^k$ is represented by a homogeneous vector $[\boldsymbol{n}_i^{k^T}, d_i^k]^T \in \mathbb{R}^4$ in the projective space [HZ03] with unit plane normal vector $\boldsymbol{n}_i^k$, where any point $\boldsymbol{v} \in \mathbb{R}^3$ on the plane satisfies a constraint: $\boldsymbol{n}_i^{k^T} \cdot \boldsymbol{v} + d_i^k = 0$; see Fig. 2.2c for an illustration. Compared to other geometric primitives like generalized cylinders, planes are advantageous in that they can be extracted robustly from corrupted object meshes and are effective features in downstream computations.

Supporting Relations S is the set of directed edges in *pt* from parent nodes to their child nodes. Each edge $s_{p,c} \in S$ imposes physical common sense between the parent node v_p

and the child node v_c. These constraints are necessary to ensure that v_p supports v_c in a physically plausible fashion:

(1) Geometrical plausibility. The parent node v_p should have a plane $\boldsymbol{\pi}_p^s = [\boldsymbol{n}_p^{sT}, d_p^s]^T$ that is horizontal and is in contact with the bottom surface of the child v_c:

$$\begin{aligned} &\exists \boldsymbol{\pi}_p^s \in \Pi_p, \boldsymbol{n}_p^{sT} \cdot \boldsymbol{g} \leqslant a_{th}, \\ &s.t.\ \mathcal{D}(v_c, \boldsymbol{\pi}_p^s) = p_c^g - (-d_p^s + s_c^g/2) = 0, \end{aligned} \tag{2.1}$$

where $\boldsymbol{g}$ is a unit vector in the gravity direction, $a_{th} = -0.9$ is a tolerance coefficient ($a_{th} = -1$ for a perfect horizontal plane), and p_c^g and s_c^g denote the position and size of the v_c's 3D bounding box along the gravity direction, respectively.

(2) Sufficient contact area for stable support. Formally,

$$\mathcal{A}(v_p, v_c) = \mathrm{A}(v_p \cap v_c)/\mathrm{A}(v_c) \geqslant b_{th}, \tag{2.2}$$

where $\mathrm{A}(v_c)$ is the bottom surface of the v_c's 3D bounding box, and $\mathrm{A}(v_p \cap v_c)$ is the area of the overlapping rectangle containing the mesh vertices of v_p near $\boldsymbol{\pi}_p^s$ within v_c's 3D bounding box. We set threshold $b_{th} = 0.5$ for a stable support.

Proximal Relations E introduce links among entities in the pt. It imposes additional constraints by modeling spatial relations between two non-supporting but physically nearby objects v_1 and v_2: Their meshes should not penetrate with each other, *i.e.*, $\mathrm{Vol}(M_1 \cap M_2) = 0$. Note that we only assign a proximal relation between two objects with overlapping 3D bounding boxes, *i.e.*, when $\mathrm{Vol}(B_1 \cap B_2) > 0$, instead of between every pair of objects to reduce computation cost. The non-penetration constraints will be applied when selecting physically plausible scene configurations, as detailed in Section 2.4.4.

2.2.2 Constructing Contact Graphs

For each scene entity x extracted from the volumetric panoptic map (see details on obtaining panoptic map in Section 2.3.4), we initialize a scene entity node v_x of cg by: (i) acquiring its

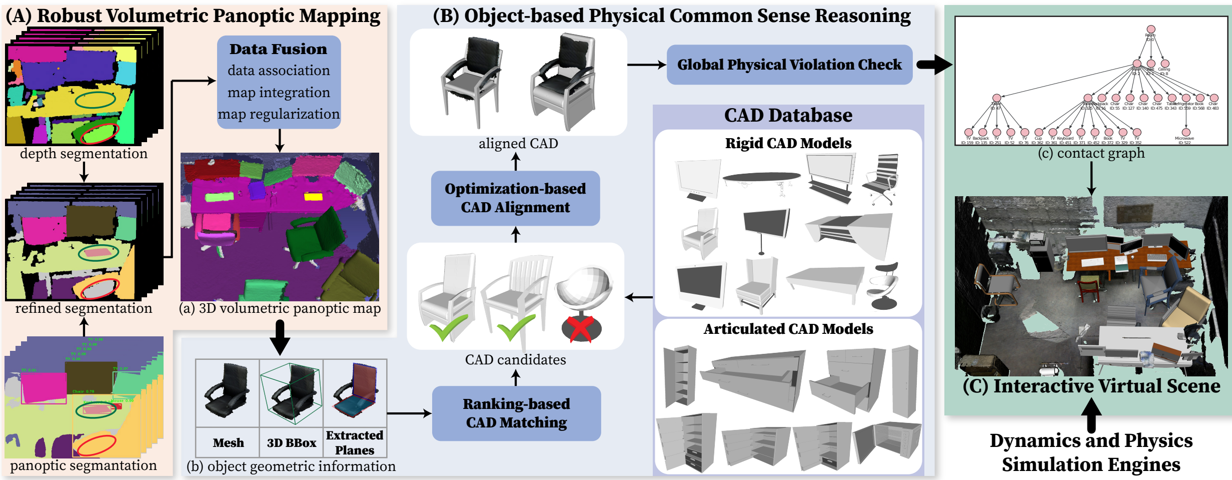


Figure 2.3: **System architecture for reconstructing a functionally equivalent scene.** (A) Per-frame segmentation and global data fusion produce (a) a 3D volumetric panoptic map with fine-grained semantics and geometry, served as the input for (B) physical common sense reasoning that matches, aligns, and replaces segmented object meshes with functionally equivalent CAD alternatives. Specifically, (b) by geometric similarity, a ranking-based matching algorithm selects a shortlist of CAD candidates, followed by an optimization-based process that finds a proper transformation and scaling between the CAD candidates and object mesh. A global physical violation check is further applied to finalize CAD replacements to ensure physical plausibility. (C) This CAD augmented scene can be seamlessly imported to existing simulators; (c) contact graph encodes the kinematic relations among scene entities in a scene and reflects the planning space for a robot.

o_x, c_x, M_x from the panoptic map, (ii) estimating a gravity-aligned, minimal 3D bounding box $B_x(\boldsymbol{p}_x, \boldsymbol{q}_x, \boldsymbol{s}_x)$ based on M_x using the method in [MB02], (iii) detecting a set of surface planes Π_x on M_x by iteratively applying RANSAC [TJR13] and removing plane inliers. We further classify each initialized scene entity node v_x as a layout node, a rigid object node, or an articulated object node based on its semantic class c_x.

Given a set of scene entity nodes initialized on-the-fly, we apply a bottom-up process to build up the structure of cg by estimating supporting relations among the entities. Specifically, for each node v_c, we find a parent node v_p with a supporting plane $\boldsymbol{\pi}_p^s$ that best satisfies the constraints described in Eqs. (2.1) and (2.2). We consider all nodes $\{v_i\}$ whose bottom planes are spatially below the 3D bounding box of v_c as v_p candidates, and acquire their gravity-opposed surface planes $\{\boldsymbol{\pi}_i^k\}$ as potential supporting planes. Then the most likely supporting relation is determined by maximizing the following score function:

$$S(v_c, v_i, \boldsymbol{\pi}_i^k) = \left\{1 - \min\left[1, \|\mathcal{D}(v_c, \boldsymbol{\pi}_i^k)\|\right]\right\} \times \mathcal{A}(v_i, v_c), \tag{2.3}$$

where the first term indicates the alignment between the v_c's bottom surface and the supporting plane, and the second term reflects an effective supporting area, both normalized to $[0, 1]$. We may also uncover an invisible supporting plane (*e.g.*, a fully occluded tabletop). When v_c is well-overlapped with v_i but v_i has no valid supporting plane, the bottom plane of v_c with be registered as a new supporting plane of v_i. This advantage is however hard to guarantee at all time due to the complexity of real-world scenarios. Finally, we construct cg and assign the attributes for each supporting edge based on the estimated supporting relations.

We further refine the 3D bounding box B_i of each scene entity node v_i such that Eq. (2.1) is strictly satisfied and the cg is feasible. This step also compensates for the error of extracting geometric features directly from incomplete reconstructed mesh. Fig. 2.2d illustrates an example of the refinement process. The reconstructed scene only produces a partial mesh of the chair; its legs are captured incompletely. Consequently, its 3D bounding box (in dashed line) only encloses the detected portion of the chair, which is floating in the air. By determining the supporting relation between the floor and the chair, our system automatically extends the bounding box (in solid line) to the supporting plane on the floor, thus reconstructed a physically plausible scene. In experiments, we also quantitatively evaluate this refinement process; see the result in Table 2.4. As the last step of cg construction, we determine the proximal relations by comparing pairwise 3D bounding boxes of scene entities.

2.2.3 Interpreting a Contact Graph

As shown in Fig. 2.2a and described above, a *cg* hierarchically organizes segmented scene entities with corresponding semantics, meshes, and extracted geometric features. To convey richer *actionable* information, we convert the *cg* to a functionally equivalent cg' by maintaining the overall graph structure and replacing each object mesh with a CAD model while preserving its semantic class, instance label, relative dimension, and surface planes; see Fig. 2.2b.

The functionally equivalent cg' with CAD models naturally encodes the full (detected) geometry state of the scene. It can be interpreted as a kinematic tree, where nodes represent links, and edges represent joints connecting two links with assumed joint type, range, and joint value. Depending on the semantic class, individual objects may be replaced by articulated CAD models. For instance, the CAD model for the microwave in Fig. 2.2b consists of two parts, the body and the door, connected by a revolute joint. The cg' (the kinematic tree) is an ideal representation to support robot planning; its joint specifications reflect the possible ways a robot can change environment states and naturally define the task goal for a robot to achieve. Although the knowledge of the object structure is injected when designing the CAD model and is not likely to match with the real one strictly, it nevertheless provides an approximation for most of the possible actions an agent can take and what the actions like, sufficient for the agent's long-term planning.

2.3 Robust Panoptic Mapping

Robust and accurate mapping of scene entities and segmenting them from clustered environments are essential for constructing a *cg* and serving our downstream tasks. We develop a robust 3D panoptic mapping module to generate object and layout segments in the form of meshes from RGB-D streams; see the pipeline in Fig. 2.3A. Based on the architecture of Voxblox++ [GFN19], our mapping module incorporates crucial modifications to improve the robustness of mapping against noisy and inconsistent segmentation at each frame.

Voxblox++ [GFN19] builds a volumetric object-centric semantic map by (i) generating per-frame segments in point cloud form by combining RGB-based instance segmentation and depth-based geometric segmentation, and (ii) associating the segments across different frames and integrating them into a Truncated Signed Distance Field (TSDF)-based object-level global map. Each per-frame segment is obtained by assigning a semantic label and an instance label produced by instance segmentation to a geometric segment produced by geometric segmentation. Assuming that segments computed using geometry cues are consistent across different frames, Voxblox++ [GFN19] associates those per-frame segments from different views with global map segments by their 3D overlapping ratio and integrates them into the global map, while recording the history of predicted semantic and instance labels for each global map segment.

However, we observe two major limitations of the Voxblox++ [GFN19]. First, the generated per-frame segments may not preserve all predicted instances and some segments of far-away background may be labeled as foreground objects, negatively affecting the mapping performance. We design two extra steps to handle this limitation, as detailed in Section 2.3.1. Second, Voxblox++ separately tracks semantic and instance labels in data association and map integration processes, making it less coherent when identifying instance and recognizing semantics for the same global map segment. Our solution is to jointly account for semantic and instance labels throughout the procedure to build a more consistent global map. We describe our implementation of this strategy in data association (Section 2.3.2), map integration and regularization (Section 2.3.3), and scene entity extraction (Section 2.3.4).

2.3.1 Per-frame Segmentation and Fusion

Following Voxblox++ [GFN19], we perform RGB-based panoptic segmentation and depth-based geometric segmentation for each frame and then combine the two sets of segments. Given a RGB-D image as the input, we use an off-the-shelf panoptic segmentation tool provided by Detectron2 [WKM19] to produce panoptic segments in RGB domain. A convexity-based depth segmentation approach [FNF18] can segment the corresponding depth image following geometric boundaries. We denote each predicted 2D panoptic segment as M_i with semantic label c_i and instance label o_i (whereas each stuff class has only one instance label), and each 3D geometric segment (in point cloud) as G_j. Then the goal is to fuse the segmentation from two sources to generate per-frame point cloud segments $\{(P_k, c_k, o_k)\}$, which preserve the predicted geometric and semantic information.

Voxblox++ [GFN19] generates $\{(P_k, c_k, o_k)\}$ by assigning semantic and instance labels to geometric segments $\{G_j\}$ greedily based on the 2D overlap between the 2D projection of each G_j and $\{M_i\}$ on the image coordinate. In practice, this strategy leads to two drawbacks. The first one is that predicted instances will be ignored if they are not recognized geometrically in depth images. Fig. 2.3A shows an example, the missing keyboard marked by a green circle in depth segmentation would be discarded by Voxblox++. We instead split a geometric segment G_j to extract the point cloud corresponding to a panoptic segment M_i if the 2D projection of G_j fully contains M_i when aligned. Then we assign semantic and instance labels for all G_j as well as the extracted point cloud segments as [GFN19] does to get $\{(P_k, c_k, o_k)\}$. Secondly, an inaccurately segmented object in RGB image may consist of far-away geometric segments in depth, *e.g.*, the floor marked by a red circle is regarded as part of the chair in the panoptic segmentation in Fig. 2.3A. Our modification addresses this issue by adding an extra step of Euclidean clustering. We compute pairwise Euclidean distances among all geometric segments that belong to the same object instance, and applying Euclidean clustering to obtain clusters of segments. Then we retrieve the largest cluster defined as having the largest total number of points in its segments, and keep the segments within as part of the instance. The rest of segments are regarded as outliers and assigned to the background.

The above implementation relies on some defined heuristics that could limit the generalizability of our panoptic segmentation approach; one direction to overcome this limitation is to introduce data-driven methods, which is beyond the scope of the paper. Nevertheless, the two proposed steps are useful practice that significantly improves the per-frame segmentation. As an example shown in Fig. 2.3a, our method (i) correctly segments the keyboard and divides the two monitors when they are geometrically under-segmented, (ii) obtains geometrically refined panoptic segmentation of the table, chair, and floor, and (iii) excludes the far-away ground from the segmentation of the chair.

2.3.2 Data Association

We associate each per-frame point cloud segment to a global 3D segment (or global segment for short) in the global map, while associating its panoptic prediction with a global panoptic entity. Note that the global segments and panoptic entities are maintained and updated throughout the entire mapping process. Following Voxblox++ [GFN19], we first draw the correspondence between per-frame segments and global segments greedily based on their 3D overlaps given the camera trajectory. We denote that each global segment is indexed with a unique segment label $l \in \mathbb{L}$.

For each per-frame segment (P_k, c_k, o_k) associated with a global segment l_i, we aim to find its associated global instance label p_m by looking at the past panoptic predictions of segment l_i. We introduce a triple-wise count $\Phi(l, c, p)$ over a segment label l, a semantic label c, and an instance label p in the global map to jointly track the semantic and instance predictions. This is inspired by the observation that the prediction of instances and their semantic labels are inter-dependent in typical object detection and segmentation algorithms [RHG16, HGD17]. Specifically, p_m is assigned with the instance label p that maximizes the count $\Phi(l_i, c_k, p) > 0$. When $\sum_p \Phi(l_i, c_k, p) = 0$, we assign a new global instance label $p_m = p_{new}$. We further prevent assigning multiple labels with the segments that have the same instance labels.

2.3.3 Map Integration and Regularization

We integrate per-frame segments into the 3D volumetric panoptic map by (i) integrating the segments into a TSDF volume [OTF17] with each TSDF voxel labeled with a global segment label l, and (ii) recording the associated panoptic entities. For any per-frame segment associated with (l_i, c_k, p_m), we increase the triple-wise count:

$$\Phi(l_i, c_k, p) = \Phi(l_i, c_k, p) + 1. \tag{2.4}$$

We also introduce a two-stage process to regulate the map by merging global segment labels and instance labels. Specifically, we first merge global segment labels pairwise if they share voxels over a certain ratio [GFN19]. Next, we merge two global instance labels $p_1, p_2 \in \mathbb{P}$ with the same semantic class $c \in \mathbb{C}$ if the duration of association with common segment labels exceeds a threshold:

$$\sum_{l \in \mathbb{L}_\cap} [\Phi(l, c, p_1) + \Phi(l, c, p_2)] \geqslant m_{th} \cdot \sum_{l \in \mathbb{L}} [\Phi(l, c, p_1) + \Phi(l, c, p_2)], \tag{2.5}$$

where $\mathbb{L}_\cap = \{l \in \mathbb{L} | \Phi(l, c, p_1) > 0, \Phi(l, c, p_2) > 0\}$. This step merges incorrectly split instances, which can be introduced by the overcautious filtering step when generating per-frame point cloud segments. We note that this map regularization process can be regarded as a delayed data association that corrects potentially wrong association of global segments and instances. It helps improve the consistency and scalability of the global map; *i.e.*, it reduces the map size.

2.3.4 Panoptic Entities Extraction

After the above mapping process, we extract the panoptic entities (*i.e.*, objects and layouts) from the global map as triangle meshes. For each global segment l, its semantic class $\hat{c}_l$ and global instance label $\hat{p}_l$ are determined following a greedy strategy:

$$\begin{aligned} \hat{c}_l &= \arg\max_{c \in \mathbb{C}} \sum_{p \in \mathbb{P}} \Phi(l, c, p), \\ \hat{p}_l &= \arg\max_{p \in \mathbb{P}} \Phi(l, \hat{c}_l, p). \end{aligned} \tag{2.6}$$

For each global instance label $p \in \mathbb{P}$, we group all global segments in the map with labels in the set $L_p = \{l \in \mathbb{L} | \hat{p}_l = p\}$ and extract the corresponding TSDF volume, from which a mesh is created. In a nutshell, our system outputs a set of scene entities in the form of triangle meshes with their instance labels and semantic labels.

2.4 Scene Reconstruction with CAD Replacement

Due to occlusion or limited camera angle, the reconstructed scene and the segment meshes are oftentimes incomplete and non-interactive before recovering them as full 3D models; Fig. 2.5a and Fig. 2.6a show some examples of incomplete meshes. We introduce a multi-stage framework to replace a segmented object mesh with a CAD model through (i) an object-level CAD matching, (ii) pose alignment of the CAD model, and (iii) a scene-level, global physical violation check; see Fig. 2.3B for an illustration of the framework.

2.4.1 CAD Pre-processing

We collect a CAD database consisting of both rigid and articulated CAD models, organized by semantic classes. The rigid CAD models are obtained from ShapeNetSem [CFG15], whereas articulated ones are first assembled and then properly transformed into one model. Each CAD model is transformed to have its origin and axes aligned with its canonical pose. Fig. 2.3B shows some instances of CAD models in the database, and Fig. 2.4 highlights some articulated CAD examples with coordinate frames on the articulated parts. All the objects can be uniformly scaled while persevering transformation and kinematic information for the subsequent matching and alignment. Similar to a segmented scene entity x, a CAD model

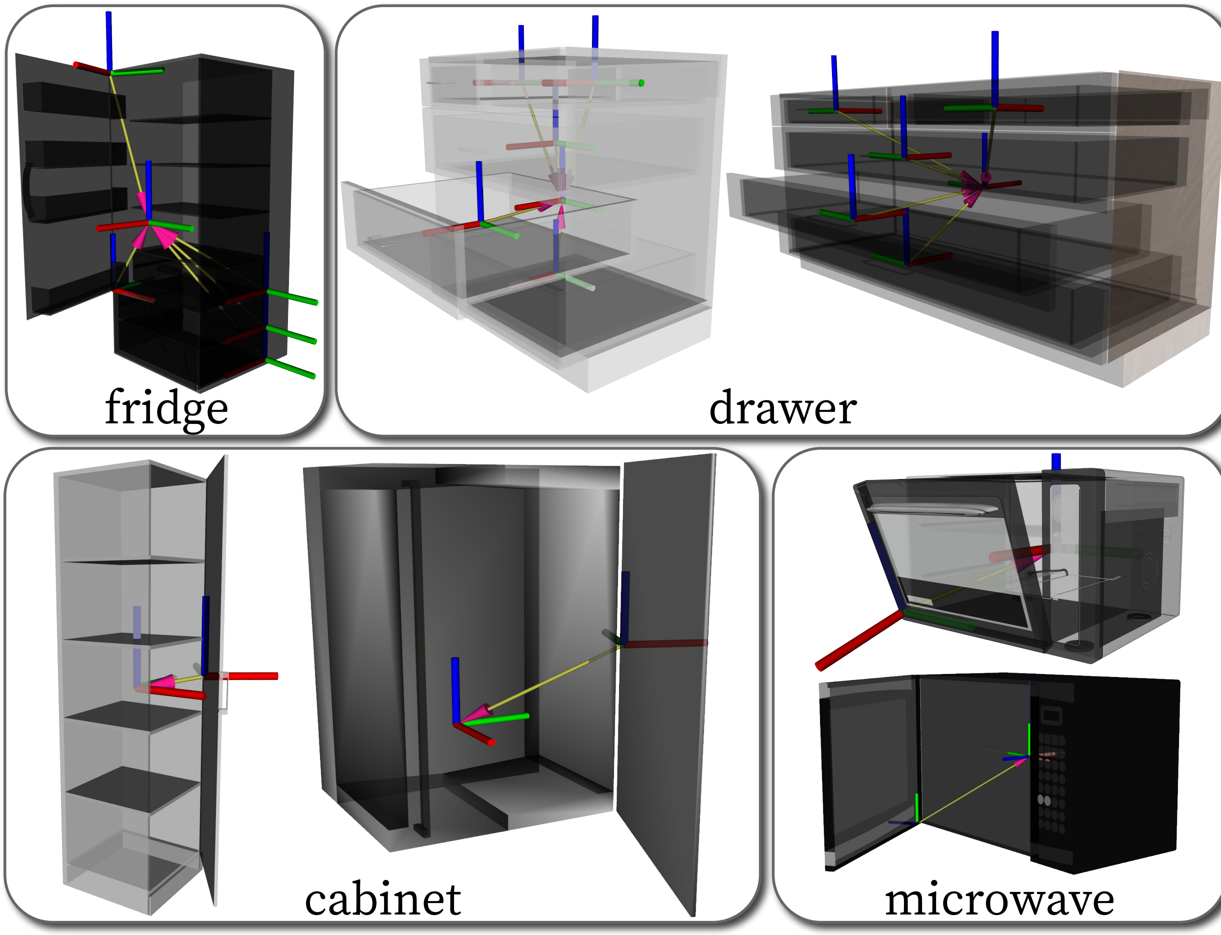


Figure 2.4: **Examples of articulated CAD models in the database.**

y is parameterized by o_y, c_y, M_y, while we further extract its $B_y(\boldsymbol{p}_y, \boldsymbol{q}_y, \boldsymbol{s}_y)$, and Π_y.

2.4.2 Ranking-based CAD Matching

Take the chair in Fig. 2.3b as an example: Given a segmented object entity x, the algorithm retrieves all CAD models in the same semantic category (*i.e.*, chair) from the CAD database to best fit x's geometric information. Since the exact orientation of x is unknown at this step yet, we uniformly discretize the orientation space into 24 possible orientations. For each rotated CAD model y that aligned to one of the 24 orientations, the algorithm computes a

Matching Error (ME):

$$D(x, y) = \omega_1 \cdot d_s(x, y) + \omega_2 \cdot d_\pi(x, y) + \omega_3 \cdot d_b(y), \tag{2.7}$$

where $\omega_1 = \omega_2 = 1.0$ and $\omega_3 = 0.2$ are the weights of three terms, set empirically. We detail these terms below.

(1) d_s computes the difference of relative 3D bounding boxes sizes between the segmented mesh and the CAD model:

$$d_s(x, y) = \left\| \frac{\boldsymbol{s}_x}{\|\boldsymbol{s}_x\|_2} - \frac{\boldsymbol{s}_y}{\|\boldsymbol{s}_y\|_2} \right\|. \tag{2.8}$$

(2) d_π penalizes the misalignment between their surface planes in terms of plane normal and relative distance:

$$\begin{aligned} d_\pi(x, y) = \min_{f_\Pi} \sum_{\boldsymbol{\pi}_i \in \Pi_x} \Bigg[\left\| \frac{d({T_x}^T \boldsymbol{\pi}_i)}{\|\boldsymbol{s}_x\|_2} - \frac{d(f_\Pi(\boldsymbol{\pi}_i))}{\|\boldsymbol{s}_y\|_2} \right\| \\ +1 - \boldsymbol{n}(\boldsymbol{\pi}_i)^T \cdot \boldsymbol{n}(f_\Pi(\boldsymbol{\pi}_i)) \Bigg], \end{aligned} \tag{2.9}$$

where T_x denotes the homogeneous transformation matrix from the map frame on the ground to the frame of the bounding box B_x, $d(\cdot)$ the offset of a plane, $\boldsymbol{n}(\cdot)$ the normal vector of a plane, and $f_\Pi : \Pi_x \to \Pi_y$ a bijection function denoting the assignment of feature planes between x and y. Note that f_Π is also constrained to preserve supporting planes as defined in Eq. (2.1). As computing d_π involves solving an optimal assignment problem, we adopt a variant of the Hungarian algorithm [JV87] to identify the best f_Π between the set of surfaces extracted from a segmented object mesh and that from a candidate CAD model. Then we can calculate the misalignment error term $d_\pi(x, y)$ that candidate CAD introduces.

(3) $d_b(y)$ is a bias term that adjusts the overall matching error for less preferable CAD candidates:

$$d_b(y) = 1 + \boldsymbol{g}^T \cdot \boldsymbol{z}(y), \tag{2.10}$$

where $\boldsymbol{z}(y)$ denotes the up-direction of the CAD model in the oriented CAD frame, and $\boldsymbol{g}$ is a unit vector along the gravity direction. Generally, we prefer CAD candidates that are upright instead of leaning aside.

Fig. 2.5b illustrates the matching process. Empirically, we observe that the discarded CAD candidates of "chair" and "table" due to large Matching Error (ME) are indeed more visually distinct from the segmented object meshes. Moreover, the "fridge" model with a wrong orientation leads to a much larger ME and is thus discarded. These results demonstrate that our ranking-based matching process can select visually more similar CAD models with a roughly correct orientation. Our system maintains the top 10 orientated CAD candidates with the lowest ME for more accurate alignment in the next stage.

2.4.3 Optimization-based CAD Alignment

The overarching goal of this step to find an accurate transformation (instead of 24 discretized orientations in the previous step) that aligns a given CAD candidate y to the original object entity x, achieved by estimating a homogeneous transformation matrix between x and y:

$$T = \begin{bmatrix} \alpha R & \boldsymbol{p} \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \ s.t. \ \min_T \mathcal{J}(x, T \circ y), \tag{2.11}$$

where $\circ$ denotes the transformation of a CAD candidate y, $\mathcal{J}$ is an alignment error function, α is a scaling factor, $R = Rot(\boldsymbol{z}, \theta)$ is a rotation matrix that only considers the yaw angle under the gravity-aligned assumption, and $\boldsymbol{p}$ is a translation. This translation is subject to the following constraint: $p^g = -d^s + \alpha \cdot s_y^g/2$, as the aligned CAD candidate is supported by a supporting plane $\boldsymbol{\pi}^s = [\boldsymbol{n}_\cdot^{sT}, d_\cdot^s]$.

The objective function $\mathcal{J}$ can be written in a least squares form and minimized by the Levenberg – Marquardt [Mor78] method:

$$\mathcal{J} = \boldsymbol{e}_b^T \Sigma_b \boldsymbol{e}_b + \boldsymbol{e}_p^T \Sigma_p \boldsymbol{e}_p, \tag{2.12}$$

where $\boldsymbol{e}_b$ is the 3D bounding box error, $\boldsymbol{e}_p$ the plane alignment error, and Σ_b, Σ_p the error covariance matrices of the error terms. Specifically: (i) $\boldsymbol{e}_b$ aligns the height of the two 3D bounding boxes while constraining the ground-aligned rectangle of the transformed B_y inside

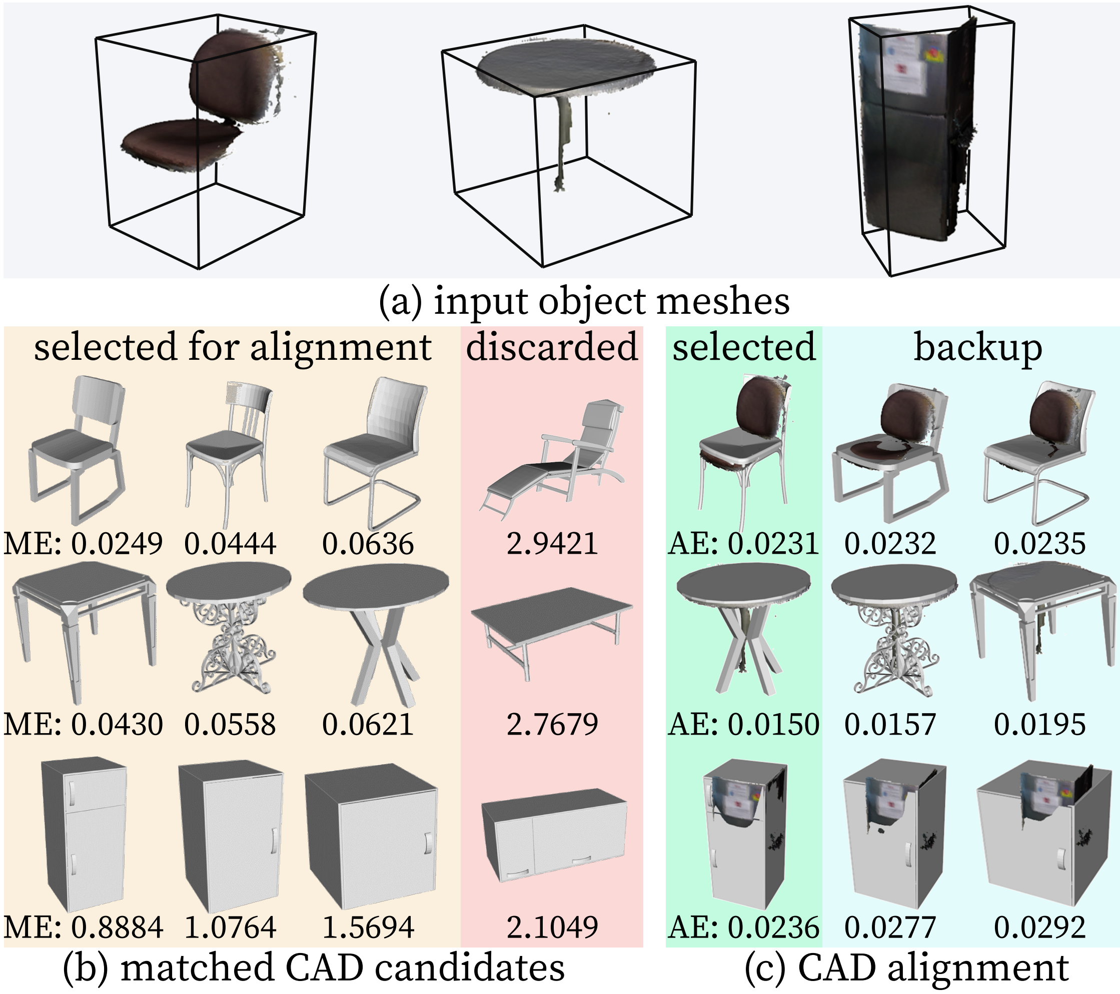


Figure 2.5: **Examples of matching and aligning CAD candidates** to (a) input object meshes. (b) All CAD models within the same semantic class as the input object are retrieved for matching. Matching Error (ME) indicates the similarity in terms of both shape and the proximity in orientations. After selecting the CAD candidates with smallest MEs, (c) a fine-grained CAD alignment process selects the best CAD model with a proper transformation based on Alignment Error (AE).

that of B_x:

$$\boldsymbol{e}_b = [\mathrm{A}(T \circ y) - \mathrm{A}(x \cap T \circ y), \alpha \cdot \boldsymbol{s}_y^g - \boldsymbol{s}_x^g]^T, \tag{2.13}$$

and (ii) $\boldsymbol{e}_p$ aligns all the matched feature planes as:

$$\begin{aligned}
\boldsymbol{e}_p &= [\Delta\boldsymbol{\pi}_1, ..., \Delta\boldsymbol{\pi}_{|\Pi_x|}]^T, \\
\Delta\boldsymbol{\pi}_i &= [-d(\boldsymbol{\pi}_i) + d(T^{-T} \cdot f_\Pi(\boldsymbol{\pi_i})), \\
&\qquad 1 - \boldsymbol{n}(\boldsymbol{\pi}_i)^T \cdot \boldsymbol{n}(T^{-T} \cdot f_\Pi(\boldsymbol{\pi}_i))],
\end{aligned} \tag{2.14}$$

where some of the notations are detailed in Section 2.2.

To evaluate how well an aligned CAD candidate fits the object mesh, we compute an AE defined as the root mean square distance between the object mesh vertices and the closest points on aligned CAD candidate; Fig. 2.5c shows both qualitative and quantitative results. The CAD candidate with the smallest AE will be selected, whereas others are potential substitutions if the selected CADs violate physical constraints, detailed next.

2.4.4 Global Physical Violation Check

Given a shortlist of matched and aligned CAD candidates, we propose a global physical violation check to finalize the CAD replacement and generate a physically plausible cg'. We first validate supporting relations and object-layout proximal relations for CAD candidates of each object. Specifically, for an object node v_p and its segmented object entity x, we discard an aligned CAD candidate y if it fails to satisfy Eq. (2.2) with any supporting child v_c of v_p. We also discard aligned CAD candidates that violate the proximal constraints with layout entities.

After early discard of invalid CAD candidates, we check the inter-object proximal constraints and jointly select CAD candidates for each object entity. We address this by formulating a constraint satisfaction problem; starting with a CAD candidate with the minimum AE for each segmented object, we adopt the min-conflict algorithm [MJP92] to obtain a global solution of CAD replacement. Finally, as the CAD alignment step cannot guarantee the precise alignment of supporting planes, we adjust the position of CAD models so that Eq. (2.1) is strictly satisfied for each supporting relation. Then we obtain a finalized cg' with

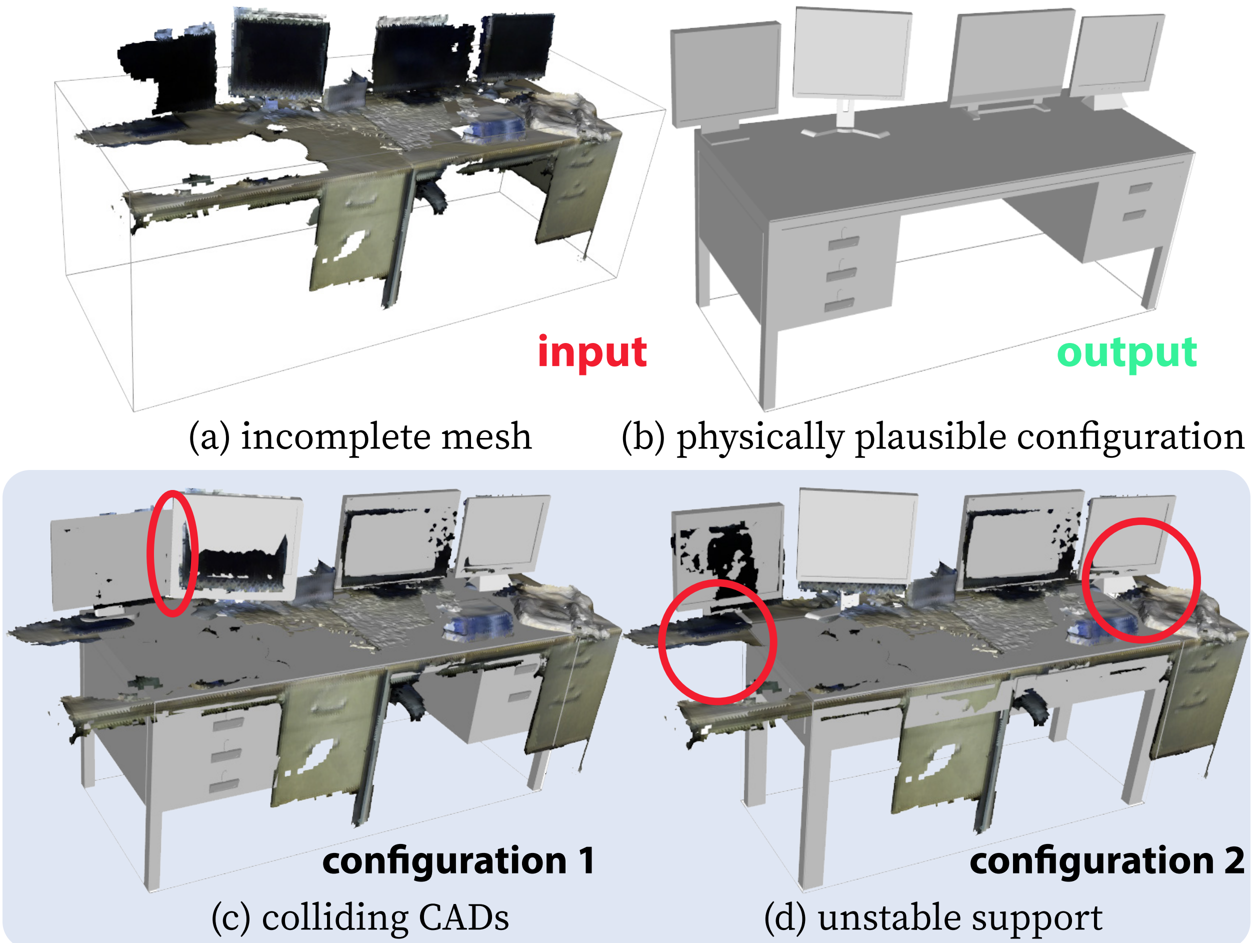


Figure 2.6: **Physical common sense reasoning for CAD replacement.** Given (a) incomplete object meshes, our physical common sense reasoning for CAD replacement (b) generates a functionally equivalent and physically plausible configuration. Specifically, the CAD matching and alignment algorithms select and rank a shortlist of CAD candidates. A global physical violation check prunes invalid configurations, such as (c) collision and (d) unstable support.

CAD models.

Fig. 2.6 illustrates a typical example, where specific configurations of CAD replacements lead to unstable support or colliding geometry. Then the abovementioned global physical violation check prunes invalid configurations and outputs a physically plausible one.

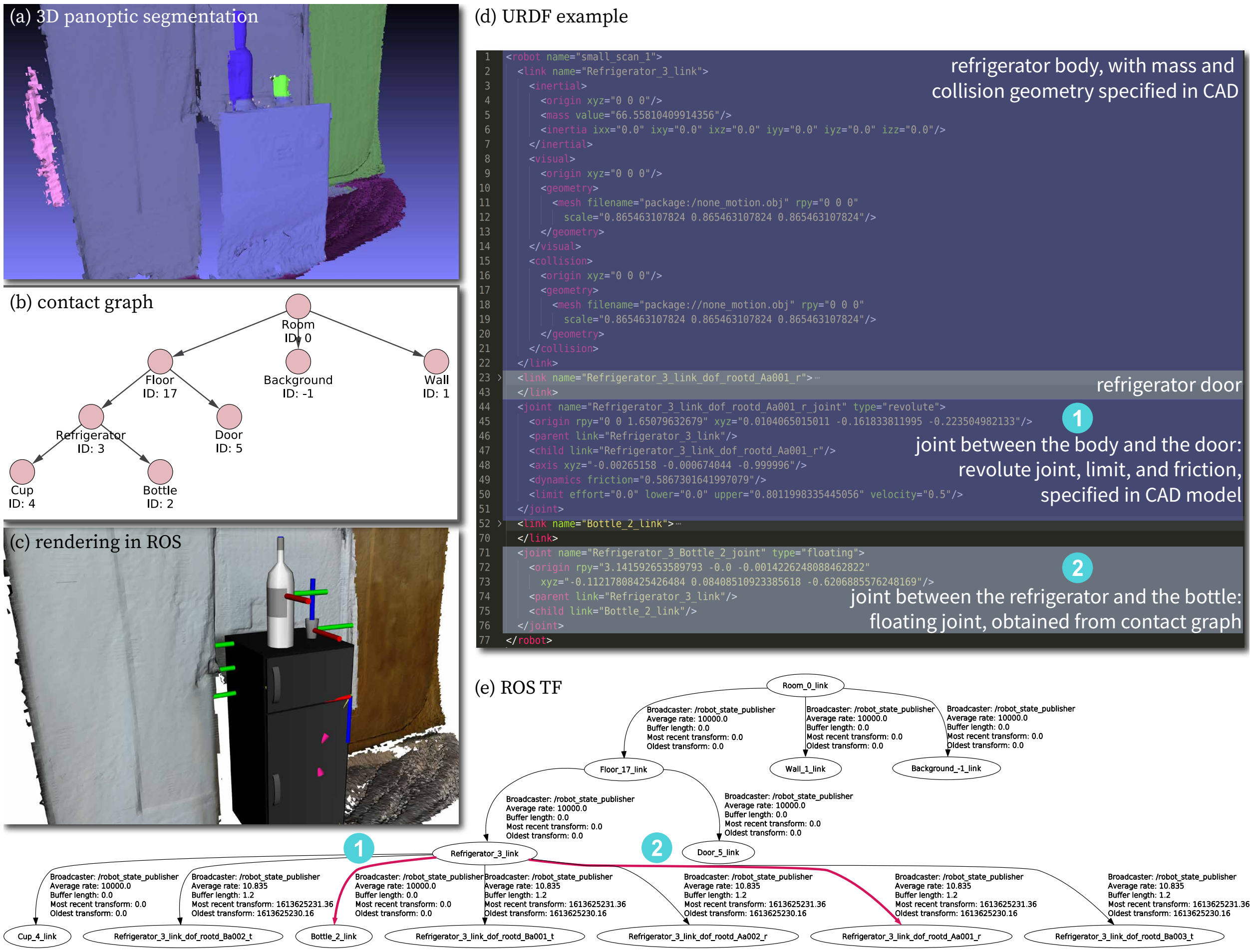


Figure 2.7: **Convert a contact graph cg′ to a kinematic tree.** (a) Given the 3D panoptic segmentation produced by our mapping module, (b) a contact graph is built and converted to (d) Unified Robot Description Format (URDF) with CAD models, which can be seamlessly (c) imported to and visualized in ROS Rviz; (e) the corresponding ROS TF describes the world states to robots.

2.4.5 Kinematic Tree Conversion

The finalized cg' can be readily converted into a kinematic tree to support various robot planning tasks. In this work, we develop an interface to generate a kinematic tree in the form of Unified Robot Description Format (URDF), which is commonly used in the robotics community.

A kinematic tree contains rigid bodies (links) as nodes, and joints connecting two bodies as edges. Each node in the kinematic tree can be created from either a scene root node, a layout node, a rigid object node, or a rigid part of an articulated object node in cg'. We preserve the joints within articulated CAD models in the kinematic tree, but alter the supporting edges in cg' to either fixed joints (no translation or rotation allowed) or floating joints (allow 3D translation and 3D rotation unless is constrained by collision) based on the semantics of the scene entity pairs. For example, a cup is connected to a table using a floating joint as a robot can freely manipulate it, and a table is linked to the floor via a fixed joint as it cannot be moved.

We show a detailed example of the kinematic tree conversion process in Fig. 2.7. Based on the 3D panoptic segmentation and the contact graph, our interface generates a kinematic tree in URDF, which can be further visualized as ROS TF and rendered in ROS Rviz. In this example, the fridge is connected to the floor via a fixed joint, and the bottle to the fridge via a floating joint. A revolute joint is inserted to connect the fridge body and the fridge door as specified by the CAD model.

2.5 Experiments and Results

2.5.1 Dataset and Implementation

We evaluate our system primarily on the SceneNN dataset [HPN16]; it contains RGB-D sequences of various room-size indoor scenes and ground-truth scene meshes annotated with instance-level segmentation. We pick 20 test sequences/scenes that contain diverse object categories to quantitative evaluate the robust panoptic mapping module and demonstrate the interactive scene reconstruction. For baselines that require training on 3D segmentation data, we roughly follow the train/test split in [HTY18] while using the test set we pick.

In our work, we choose the baseline panoptic segmentation model in Detectron2 [WKM19], pre-trained on the COCO panoptic class [LMB14] for segmentation on RGB. We use [FNF18] as the baseline geometric segmentation method for depth images. Of note, our system is designed in a modularized manner so that it is flexible enough to incorporate more powerful models when available. For instance, the segmentation module is designed as a server-side service that will be requested by a client in the perception system when a new image frame arrives and produce a list of segmented masks with labels in the response. Any segmentation methods being wrapped as a service following this protocol could be connected to our system.

2.5.2 Robust Panoptic Mapping

We evaluate our robust panoptic mapping module on three aspects: (i) 3D panoptic mapping quality, (ii) 3D object instance segmentation, and (iii) oriented 3D bounding box estimation. The first aspect focuses on how well the system reconstructs the scene and segments the objects and layouts within, whereas the latter two emphasize individual objects. Such a protocol design provides a holistic evaluation of the fundamental component of the proposed system: The accuracy of object segmentation and bounding box estimation are crucial for the overall quality of scene reconstruction when matching and aligning CAD models. An ablation study (noted as "w/o joint fusion") is also conducted, where we disable our modifications of jointly processing semantic and instance labels in data fusion, *i.e.* the procedure described in Sections 2.3.2 and 2.3.3. This study will not only better demonstrate how much the introduced modifications influence the overall mapping performance, but also verify the effectiveness of the per-frame segmentation and fusion technique by comparing the ablated results with those from baselines.

For each sequence used in the experiment, our mapping module processes incoming RGB-D frames with ground-truth camera poses provided by the dataset. We consider 10 semantic classes including 2 *stuff* classes (wall and floor) and 8 most common *thing* classes (bed, table, chair, monitor, sofa, bag, cabinet, and fridge) for evaluation.

3D Panoptic Mapping This experiment evaluates the overall segmentation performance for panoptic mapping, following the criteria defined in [KHG19] and [NSI19]:

$$PQ = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{SQ}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{RQ}}, \tag{2.15}$$

where the Segmentation Quality (SQ) is the averaged Intersection over Union (IoU) of *p*redicted and *g*round-truth panoptic masks on all matched predictions in the same class, and the Recognition Quality (RQ) is the F_1 score [MFM04] of object recognition for the aforementioned 10 semantic classes. Panoptic Quality (PQ) is simply the product of SQ and RQ, which better reflects the overall segmentation results.

We compare our panoptic mapping module with the Voxblox++ [GFN19]. Table 2.1 (white columns) shows their corresponding PQ, RQ, and SQ of 7 individual SceneNN sequences, averaged on 10 classes. Table 2.2 further tabulates per-class panoptic segmentation results of all 20 sequences. Of note, we compute PQ, RQ, and SQ in category-level for each semantic class (Table 2.2), and average the PQ, RQ, and SQ of all classes to obtain those values in scene-level (Table 2.1).

Overall, our panoptic mapping module significantly outperforms the baseline as indicated by higher PQ for individual sequences and most of the semantic classes. Without applying joint fusion, our system still performs better than the baseline Voxblox++, showing the efficacy of our per-frame segmentation. But it is not as good as our full module, which further demonstrates that our proposed strategies positively contribute to objects and layouts recognition (higher RQ value indicates higher accuracy) and segmenting them well (higher SQ value). The extra performance gain our modifications bring is very crucial for the subsequent processes.

3D Instance Segmentation We also evaluate the performance of 3D instance segmentation on 8 *thing* classes using the mAP@0.5 metric, *i.e.*, the Mean Average Precision (mAP)

Table 2.1: **Quantitative class-averaged results of 3D panoptic segmentation and 3D instance segmentation on individual sequences in the SceneNN dataset [HPN16].** Note that ProgressFusion [PHN19] accounts for more classes than the other two methods. All values are in percentage.

	Ours				Voxblox++ [GFN19]				ProgressFusion [PHN19]
	Panoptic			Instance	Panoptic			Instance	Instance
ID	PQ	SQ	RQ	mAP	PQ	SQ	RQ	mAP	mAP
011	**45.5**	60.4	50.0	58.3	34.3	64.3	40.0	**80.8**	52.1
030	**50.4**	55.6	64.5	**58.3**	23.4	34.7	26	33.5	56.8
061	**43.0**	52.0	46.3	33.6	25.7	53.1	32.2	38.6	**59.1**
078	**54.7**	54.7	62.5	**50.0**	26.3	52.5	31.7	43.9	34.9
086	**27.3**	39.6	34.6	**40.8**	19.4	32.9	25.2	37.6	35.0
096	**12.5**	21.4	14.6	23.0	7.3	11.9	8.3	14.6	**26.5**
223	**49.5**	60.2	63.3	**60.0**	21.7	40.2	26.7	34.1	40.9

Table 2.2: **Per-class 3D panoptic segmentation results in the SceneNN dataset [HPN16].** All values are in percentage.

		all	stuff	thing	wall	floor	bed	table	chair	monitor	sofa	bag	cabinet	fridge
Voxblox++ [GFN19]	PQ	24.5	10.9	27.9	4.0	17.8	**18.0**	14.4	35.5	48.5	**46.0**	**24.0**	7.2	29.5
	SQ	77.6	73.7	78.6	69.3	78.0	72.0	71.3	77.0	81.4	82.8	84.0	86.0	73.9
	RQ	31.2	14.3	35.4	5.7	22.9	25.0	20.3	46.0	59.6	55.6	28.6	8.3	40.0
Ours (w/o joint fusion)	PQ	27.8	12.6	31.6	5.6	19.5	8.7	26.7	31.7	48.8	45.7	16.1	**21.9**	**53.4**
	SQ	77.5	71.8	78.9	64	79.6	65.9	73.8	76	89	82.2	72.6	78.5	93.4
	RQ	34.2	16.6	38.6	8.7	24.5	13.3	36.1	41.8	54.9	55.6	22.2	27.9	57.1
Ours	PQ	**35.4**	**44.2**	**33.2**	**25.2**	**63.1**	11.5	**27.4**	**40.1**	**65.7**	34.3	17.4	20.1	48.7
	SQ	80.5	79.3	80.9	73.5	85.0	77.6	76.1	79.1	88.8	80.0	78.3	81.7	85.2
	RQ	43.1	54.3	40.3	34.3	74.3	14.8	36.0	50.6	73.9	42.9	22.2	24.6	57.2

computed using an Intersection over Union (IoU) with a threshold of 0.5. The evaluation is two-fold. First, we report the class-averaged results in the progressive mapping manner on 7 individual sequences compared with Voxblox++ [GFN19] and ProgressFusion [PHN19], another online semantic mapping framework; see the grey columns in Table 2.1. Our approach performs better than Voxblox++ on almost all the sequences. Note that the ProgressFusion accounts for all NYUDv2 [SHK12] classes available in the dataset, and we evaluate the performance only on the 8 *thing* classes for our method and Voxblox++. While it's possible to re-train our panoptic segmentation module to incorporate more classes, we believe the current experiment is sufficient to demonstrate the advantage of our panoptic mapping module without defeating its purpose of leveraging pre-trained perception models.

Second, in Table 2.3, we study the per-class mAP@0.5 of our approach compared with Voxblox++ [GFN19] and two learning-based works [PNH19, HZX20] that directly segment 3D instances from the full point cloud of scenes instead of continual RGB-D data stream. As the input formats are different, the results are not directly comparable. They nevertheless provide a better sense about how well our approach performs. We re-train [PNH19] and report the results of its two variants on our test set, and adopt the results reported by in [HZX20]. Overall, our method performs significantly better than Voxblox++ in most classes, and our variant without joint fusion can still slightly outperform Voxblox++. OccuSeg appears to perform the best for object classes that are less likely to be severely occluded in the dataset, but our approach also poses a unique advantage of handling partially-visibly objects such as cabinets and fridges that usually attached to a wall.

Oriented 3D Bounding Box Estimation We further evaluate the accuracy of oriented (gravity-aligned) 3D bounding boxes of object instances, which serve as essential geometric cues for physical reasoning and CAD replacement. Similarly, the mAP@0.5 metric is adopted to evaluate the oriented 3D bounding box estimation on the 8 *thing* classes. Table 2.4 tabulates results using the baseline method [GFN19], two variants described in [PNH19], our ap-

Table 2.3: **Per-class 3D instance segmentation results on the SceneNN dataset [HPN16].** The numbers in bold and numbers in underscore indicate the best and the second best results, respectively. All values are in percentage.

	Input Format	bed	table	chair	monitor	sofa	bag	cabinet	fridge
MT-PNet [PNH19]	Full point cloud	0.0	12.5	42.8	26.5	0.0	0.0	0.0	0.0
MLS-CRF [PNH19]	Full point cloud	0.0	27.3	50.9	38.6	0.0	0.0	0.0	0.0
OccuSeg [HZX20]	Full point cloud	**66.7**	**50.0**	**91.3**	**76.9**	50.0	-	5.7	-
Voxblox++ [GFN19]	RGB-D stream	39.4	22.3	55.6	63.6	72.4	**56.4**	8.5	51.6
Ours (w/o joint fusion)	RGB-D stream	17.4	40.7	51.3	48.1	**82.8**	53.2	35.4	**94.5**
Ours	RGB-D stream	27.5	46.6	65.3	69.4	64.3	53.2	**43.9**	**94.5**

proach, and our approach with supporting-based refinement (detailed in Section 2.2.2). Note that since there is no native support for evaluating oriented 3D bounding boxes in [PNH19], we re-train the models on the SceneNN dataset for this experiment. The results indicates that our approach predicts their oriented 3D bounding boxes accurately for most object classes compared with the baselines. The refinement process further improves the performance by completing the partially-observed object boxes. Looking at the two variants in [PNH19], while MLS-CRF introduces an extra post-processing step using a Conditional Random Field (CRF) on top of the MT-PNet, its 3D bounding box estimation accuracy drops as extra points from the background are merged into the foreground objects in CRF regularization. An interesting disparity between [PNH19]'s instance segmentation results (Table 2.3) and its bounding box estimation (Table 2.4) appears—having a zero-score in one place and turning to positive in another. This is because a subtle change in segmenting instances may lead to a large error in estimated bounding boxes.

In summary, the above three quantitative evaluations demonstrate that our robust panoptic mapping module well suited for (i) recognizing and segmenting scene entities progressively during mapping and (ii) estimating objects' 3D oriented bounding boxes in complex and clus-

Table 2.4: **Per-class oriented 3D bounding box estimation results on the SceneNN dataset [HPN16] based on mAP@0.5 metric.** All values are in percentage.

	all	bed	table	chair	monitor	sofa	bag	cabinet	fridge
MT-PNet [PNH19]	10.4	25.8	12.8	19.3	25.0	0.0	0.0	0.0	0.0
MLS-CRF [PNH19]	5.7	0.0	12.6	33.0	0.0	0.0	0.0	0.0	0.0
Voxblox++ [GFN19]	24.1	**39.4**	19.5	31.8	37.0	47.9	0.0	4.0	13.4
Ours (w/o joint fusion)	28.5	17.4	21.4	36.6	29.4	55.8	**53.2**	14.1	0
Ours	45.3	27.5	54.9	44.6	**42.5**	53.7	**53.2**	**29.8**	**56.4**
Ours (refined)	**47.2**	22.9	**68.2**	**49.2**	38.7	**59.1**	**53.2**	**29.8**	**56.4**

tered real indoor environments. The former capability is essential for selecting a proper CAD model to replace a segmented object, and the latter determines the size and scale of that CAD. The ablation study highlights the performance gain introduced by our data fusion procedure, demonstrating the success of jointly dealing with semantic and instance predictions during mapping.

Table 2.5: **GED of four scenes between annotated cg_{gt} and inferred contact graph from our panoptic mapping results cg_{ours} (*i.e.* Fig. 2.9b) and from ground-truth maps cg_{map} (*i.e.* Fig. 2.9a).** Note that editing a wrong support will need two operations, removing an edge and adding an edge, resulting a graph distance of 2.

Scene	Total nodes		Total distance		Wrong support		Missing detection		Wrong detection	
	cg_{gt}	*v.s.*	cg_{ours}	cg_{map}	cg_{ours}	cg_{map}	cg_{ours}	cg_{map}	cg_{ours}	cg_{map}
225	20		12	4	1	2	5	0	5	0
231	29		9	4	0	2	2	0	7	0
249	11		7	0	3	0	1	0	0	0
322	17		5	2	1	1	2	0	1	0

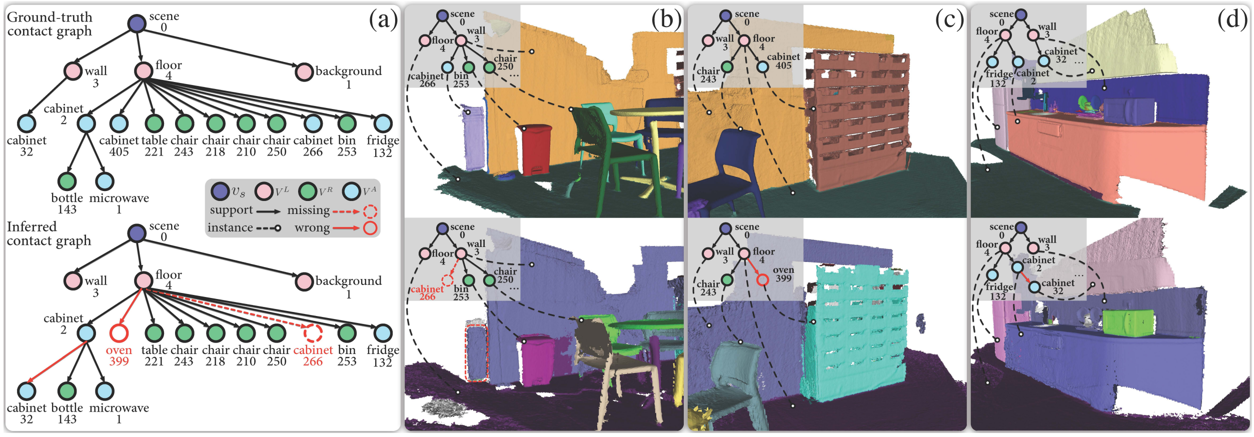


Figure 2.8: **Comparison between ground-truth and inferred contact graph**. (a) The annotated cg_{gt} *v.s.* the cg_{ours} inferred from our panoptic mapping results for scene 322. (b)(c)(d) highlight a missing detection (cabinet 266 is not detected), a wrong detection (cabinet 405 is detected as oven 399), and a wrong support (cabinet 32 is supported by wall instead of supported by cabinet 2), respectively.

2.5.3 Inferred Contact Graph

Having extracted object and layout meshes from the volumetric panoptic map, a contact graph cg can be built based on inferred supporting relations before using it to bridge the actual scene to a virtual one. It is worthwhile to evaluate the structure of an inferred cg as it collectively reveals the performance of object recognition, supporting relation identification, and overall results. To conduct this evaluation, we annotate the contact graphs for four scenes in the SceneNN dataset [HPN16] based on their ground-truth segmentation shown in Fig. 2.9a. Then, a Graph Editing Distance (GED) [ZS89] metric is applied to evaluate the distance between an annotated contact graph and an inferred graph from a segmented map. Specifically, GED measures the dissimilarity of two graph by how many graph editing operations (here we consider insertion, removal of a node or an edge, and substitution of a node ID, a total of five operations) are needed to convert one graph to the other.

The results are reported in Table 2.5, where we compare the GED between (grey columns) the annotated contact graph cg_{gt} and that inferred from our mapping results cg_{ours}, and between (white columns) cg_{gt} and that inferred from ground-truth segmentation map cg_{map}. The Total nodes column indicates the size of cg_{gt}, *i.e.* the number of scene entities a scene has. The Total distance column shows the total editing operations required to covert cg_{ours} or cg_{map} to cg_{gt}, indicating the overall quality of the inferred cg. A qualitative illustration between two graphs is also shown in Fig. 2.8a. Moreover, the GED can be broken down to three types of errors appeared in an inferred graph: (i) Wrong support (or wrong edge): a supporting relation is not assigned correctly, *i.e.* the parent node of an entity should be another; (ii) Missing detection (or missed node): an entity is not detected or segmented and thus not included in the graph; (iii) Wrong detection (or extra node): an entity that is not supposed to appear in the graph, and the reasons for having extra nodes could be having a wrong semantic label, one entity is segmented as multiple ones, or both. Fig. 2.8bcd depict some examples of error in scene 322.

In Table 2.5, we observe that our system mainly suffers from the clustered scene 225 and scene 231 with lots of small objects, indicated by the high costs of Missing and Wrong detection. On the other hand, the relatively low cost caused by Wrong support indicates that our criteria of determining supporting relations is effective.

2.5.4 Interactive Scene Reconstruction

Fig. 2.9 showcases the qualitative results for reconstructing functionally equivalent and interactive scenes. Given a volumetric panoptic map (Fig. 2.9b) and a constructed contact graph, our system reconstructs a high-quality, functionally equivalent, interactive scene by replacing incomplete meshes with CAD models and perform physical reasoning on the contact graph, as shown in Fig. 2.9c. Nevertheless, we find that our system performs poorly or fails under two circumstances: (i) The incomplete object mesh has misguided or no feature planes, resulting in the misalignment of the CAD model; (ii) The object is not supported by its bottom face (*e.g.*, cabinets on the wall), resulting in the incorrectly reconstructed scene due to the wrong estimate of the supporting relations. Section 2.6 provides a more in-depth discussion of the system limitations.

By converting the scene contact graph into a kinematic tree in URDF, we are able to seamlessly import the reconstructed functionally equivalent and interactive scene into various existing simulators. Practically, we also specify physical proprieties (such as link mass, collision geometry, joint friction) in URDF to facilitate more sophisticated simulations. We demonstrate the usage of our reconstructed interactive scenes with several examples: (i) Fig. 2.9d shows the reconstructed scenes in the ROS environment, which subsequently connects the reconstructed scenes and robot Task and Motion Planning (TAMP). Detailed planning schemes and implementations could be found in the authors' parallel work [JZW21, JZJ21]. (ii) Fig. 2.9e demonstrates that the reconstructed scenes can be loaded into the VR environment [XLZ19] for interactions with both virtual agents and human users, which opens a new avenue for future studies. (iii) Fig. 2.10 presents keyframes of a robot executing a long-horizon mobile manipulation task that involves interactions with articulated objects.

2.5.5 Reconstruction of Physical Scenes

To further evaluate our system under a real-world setting, we conduct experiments to reconstruct physical scenes using a handheld Kinect v2 sensor. We obtain accurate camera poses with a state-of-the-art feature-based SLAM system [MT17] based on RGB-D streams. The resulting 3D volumetric panoptic map, reconstructed functionally equivalent and interactive scene, and an example of robot interaction are shown in Figs. 2.11a to 2.11c, respectively. This result reveals a huge potential of applying the proposed system to facilitate robot task execution in the physical world.

We further analyze scene reconstruction results using three typical cases that highlight the advantages and failure conditions. In case 1 (Fig. 2.11d), the table is occluded by the chair and thus is identified as two instances floating in the air. These two tables are determined as floor-supported, and their 3D bounding boxes are further refined on the basis of the

(a) Ground-truth segmentation for scene 225, 231, 249, and 322 [HPN16]

(b) Segmentation results produced by the proposed panoptic mapping

(c) Scene reconstruction with functional and actionable CAD objects

(d) Robot interaction with functionally equivalent reconstructed scenes

(e) VR interaction with functionally equivalent reconstructed scenes

Figure 2.9: **Qualitative results of four reconstructed scenes with actionable CAD models.** With functionally equivalent reconstruction, both robots and human users can virtually enter the scene for Task and Motion Planning (TAMP) and VR applications.

supporting relations. The system eventually outputs two separate tables in the reconstructed interactive scene, where their poses aligned with the oriented 3D bounding boxes of the partial meshes. Case 2 (Fig. 2.11e) shows an example of a better reconstructed workspace. Given the incompletely segmented table and chair point cloud, our system can correctly estimate the supporting relations and their orientations, replace each mesh with a similar CAD model, and finally produce a functionally equivalent and physically plausible workspace, although the dimension of the table is not ideal as part of the point cloud behind the chair is not detected and segmented correctly. Case 3 (Fig. 2.11f) provides a more challenging example. The fridge and microwave are segmented and replaced by articulated CAD models, whereas the chair is not successfully detected and is removed from the reconstructed scene. Similar to case 1, the table is identified and replaced with two instances. To avoid mesh penetration, the proximal constraints incorporated by the *cg* helps the CAD replacement process to select a rounded table on the left side, but it is not a satisfactory replacement due to the large discrepancy in shapes.

2.6 Discussion

We now discuss in greater depth six topics related to the presented work.

2.6.1 Scene Functionality

Most computer vision tasks focus on devising new methodologies and representations that are beneficial within the scope of computer vision. However, this paper seeks to address a new task of building a representational system with the emphasis of facilitating robot activities. The core of the system is to represent the scene *functionality*, one of the key common senses governing our understanding of a scene [ZGF20]. This goal is achieved by associating high-level cues from object semantics (*e.g.*, whether they can be moved, opened, or can support other interactions) and low-level cues (*i.e.*, replacing the object meshes with CAD

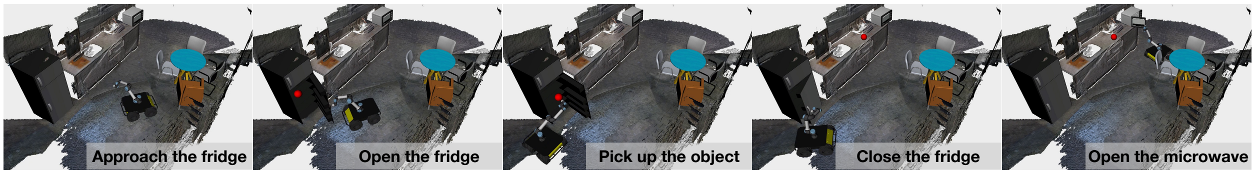


Figure 2.10: **Robot executing a mobile manipulation task with multiple steps**: microwaving an item (indicated by the red ball) by first retrieving it from the fridge.

models, whose underlying kinematic indicate how exactly they interact). Additional object attributes, affordance, or task-dependent information can be annotated to CAD models to depict the scenes more comprehensively. A subsequent, interesting open question is how to quantify the divergence between the actual scene and the reconstructed one with CAD replacements.

2.6.2 Scene Representation

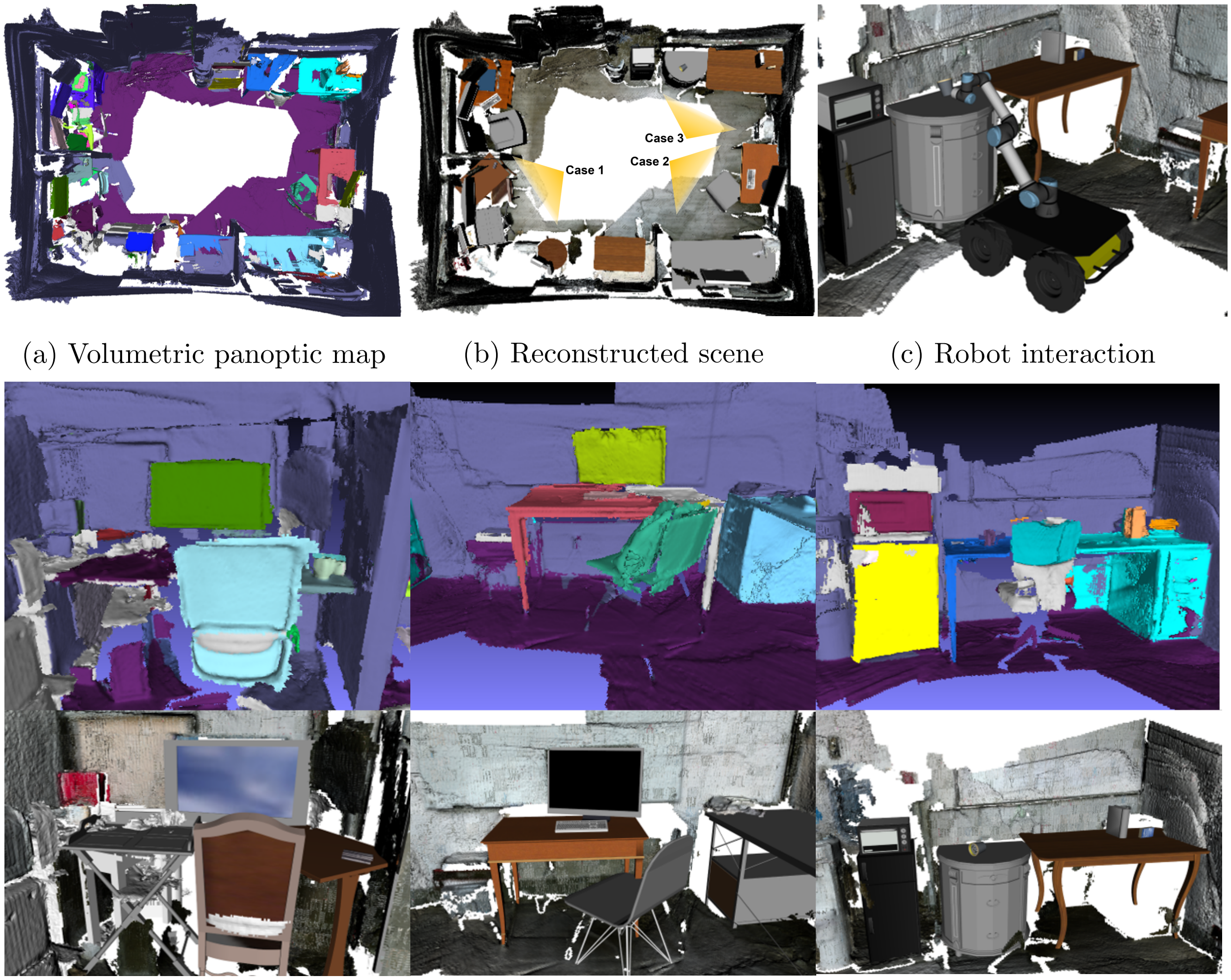
The contact graph *cg* produced by the proposed system is a holistic, but approximate scene representation. By itself is indeed insufficient for robot task executions where more precious local scene representations are needed. Although the *cg* does not seem directly beneficial, its importance is two-fold when considering a robot designed to operate over a long period of time. Firstly, the representation maintains a global belief of the scene, helps a robot to anticipate the effects of (sequence of) actions, and incorporates the actual action effects back to the *cg*. This is essential for the robot to forward search for a task plan over a long horizon [Kae20]. Secondly, given the variety of tasks a robot may anticipate, our *cg* can serve as a carrier for those necessary local representations that can be annotated, trained beforehand or build online with proper perception modules. Otherwise, different task-driven representation are standalone, lacking proper organizations.

2.6.3 Task and Motion Planning (TAMP)

Existing TAMP frameworks are oftentimes too brittle to handle a large variety of the environment for interactions. [KL11] and [SFR14] propose new TAMP frameworks, making planning long-horizon manipulation tasks possible. Still, the framework focuses on pick-and-place tasks with carefully defined environmental constraints, making it difficult for complex indoor manipulation tasks. [GPL20] devise a framework for a complex problem, which requires interaction with articulated objects. Similarly, this work is still limited to carefully designed environments with limited variety in the setup. A key factor to this problem is the lack of simulation environments that support various interactive actions (*e.g.*, door opening, object picking) and semantic relations among objects. Crucially, it could be time-consuming to generate these environments manually. In comparison, our framework can automatically generate interactive environments from real sensory data of challenging physical world in the wild and demonstrate a certain capability to support more complex TAMP study in the future.

2.6.4 Embodied AI

Embodied AI researches focus on learning a policy, mostly in simulations, that can ultimately be applied to real-world applications. Therefore, a significant amount of work is to develop simulation platforms to support learning. Our perspective echoes the motivation of task-oriented vision—designing a proper vision system that better suits a given task [IH92]. Specifically, our work allows the agent to acquire a policy specific to the given environment for the given task by capturing and representing the *actionable* information in the environment from the agent's view. Thus, our work goes beyond panoptic segmentation and 3D reconstruction.



(a) Volumetric panoptic map (b) Reconstructed scene (c) Robot interaction

(d) Case 1 (e) Case 2 (f) Case 3

Figure 2.11: **Reconstructing a physical scene with a handheld RGB-D sensor**. (a) The panoptic segmentation and the overall mapping. (b) The reconstructed scene with CAD models replacing the segmented objects, which supports (c) a robot to simulate its Task and Motion Planning (TAMP). (d–f) Qualitative results of segmentation and reconstruction. Our system recognizes most of the objects and properly replaces them with CAD models that are similar to those objects in the physical scene; see Case 2 and 3. A common problem is due to occlusion, which causes inaccurate detection, *e.g.*, one desk is recognized as two as it is occluded by the chair; see case 1 and 3.

2.6.5 Supporting Relations

Inferred supporting relations define the structure of contact graph. While this paper mainly concerns about stable support, *i.e.* those satisfying Eq. (2.3), there are several other supporting configurations, such as an object is hanging on wall and supporting from behind, is supported by two adjacent tables, is placed on floor and tilted against another object *etc*. These types of supports are not explicitly modeled and may not be well handled. Our system can nevertheless reveal their supporting relations in part. For instances, the blue bottle in Fig. 2.1c is regarded as supported by the wall because no valid supporting parent is identified but it is very close to the wall. Whereas in Fig. 2.8d, the upper cabinet that is supported by the wall (and possible the ceiling as well) is wrongly considered as supported by the lower cabinet. In other cases where an object is supported by multiple entities simultaneously, only one entity would be identify as a supporting parent based on overlapping area defined in Eq. (2.2). For a tilted object on floor, only the floor would be identified as the supporting object. Hanging objects that are supported from above, are not handled in the present work either. Apparently, our strategy cannot fully address the above less common supporting relations reliably at all time, but more specific spatial relations can be modeled and incorporated into the contact graph representation as well to extend the system's capability.

2.6.6 Other Limitations

The system's performance heavily relies on 3D panoptic segmentation of scene entities and the CAD replacement of object meshes. Currently, our robust panoptic mapping module utilizes open-sourced software to generate panoptic segmentation on RGB frames. While its development is beyond this paper's scope, new models and methods are emerging in the fast-paced community, and our system is designed to easily incorporate newer methods to improve the mapping performance further and support subsequent processes by reducing error propagated in each stage.

Our CAD replacement algorithm matches and aligns CAD models to incomplete meshes based on simple geometric features, *i.e.*, 3D bounding boxes and surface planes, which are potentially fragile when the meshes are noisy and incomplete. In the future, we may integrate deep learning-based methods [ADN19, PTL18] for more robust and accurate CAD replacement.

The articulated CAD models are unlikely to match the structure of real objects exactly. One potential solution is to detect and segment object parts and estimate the kinematics to assemble more fine-grained CAD models. The PartNet dataset [MZC19] provides an initial direction to start with.

There are various actionable information and many other information an object should afford for a robot to sufficiently interact with it depending on different task specifications, while this paper only studies a few, *e.g.* inferred supporting relations and annotated kinematics information. One central question remains unanswered is how to balance manual efforts and algorithmic efforts so that an intelligent robot can better excel in ever-changing environment.

2.7 Conclusions and Future Work

This paper proposes a new task of reconstructing functionally equivalent and interactive scenes to simulate robot autonomy and develop a full system that demonstrates this new perspective. Contrasting to the classic view of scene reconstruction that focuses on the geo-information, our system captures semantics and associated actionable information in scene entities by (i) a novel panoptic mapping module that reconstructs individuals objects and layouts, (ii) a geometric and physical reasoning module to replace the incomplete objects meshes with part-based interactive CAD models, and (iii) a contact graph representation that facilitates physically plausible scene reconstruction, and reflects action opportunities and action outcomes in terms of kinematic information. In experiments, we first quantitatively demonstrate that our system can produce high-quality panoptic segmentation, a prerequisite for the subsequent processes. We further qualitatively showcase various reconstructed scenes with functional CAD model replacements, from dataset and real-world scanning, that support fine-grained interactions in ROS and VR environments.

In the future, we hope to improve the CAD matching and alignment processes by introducing more robust feature extraction and exploring learning-based methods. Another promising future direction is to incorporate sophisticated part-based object recognition and modeling. Together with a CAD assembling module, it is possible to generate a CAD model that matches a segmented object with much finer details and reflects its functionality better. Meanwhile, more functional and attribute information can be encoded to CAD models to better reveal the "Dark Matter [ZGF20]" of a scene. Finally, we will explore the feasibility of promoting the embodied AI research from navigation tasks to fine-grained manipulation tasks using our reconstruction framework.

CHAPTER 3

Planning in the Geometry Fluent Space via a Virtual Kinematic Chain

Inspired by the theory of body schema [Gal06] proposed by cognitive psychologies and philosophers: Humans maintain a body's representation during their motions and interactions with the environment; this representation is malleable and can be extended to incorporate external objects, this chapter presents a present a Virtual Kinematic Chain (VKC) perspective, a simple yet effective method, to improve task planning and motion planning for mobile manipulation in the geometry fluent space. Although the idea of the body schema has been introduced to the robotics community to represent robot structures and guide robot's behaviors [HMA10], it has left untouched whether the theory of body schema would promote a service robot's (mobile manipulation in particular) planning and execution skills in complex manipulation tasks. And if it does, what would be a proper representation at a computational level?

By consolidating the kinematics of the mobile base, the arm, and the object being manipulated collectively as a whole, this novel VKC perspective naturally defines abstract actions and eliminates unnecessary predicates in describing intermediate poses. As a result, these advantages simplify the design of the planning domain and significantly reduce the search space and branching factors in solving planning problems. Accordingly, a mobile manipulation task is represented by altering the state of the constructed VKC, which can be converted to a motion planning problem, formulated and solved by trajectory optimization. In experiments, we implement a task planner using Planning Domain Definition Language

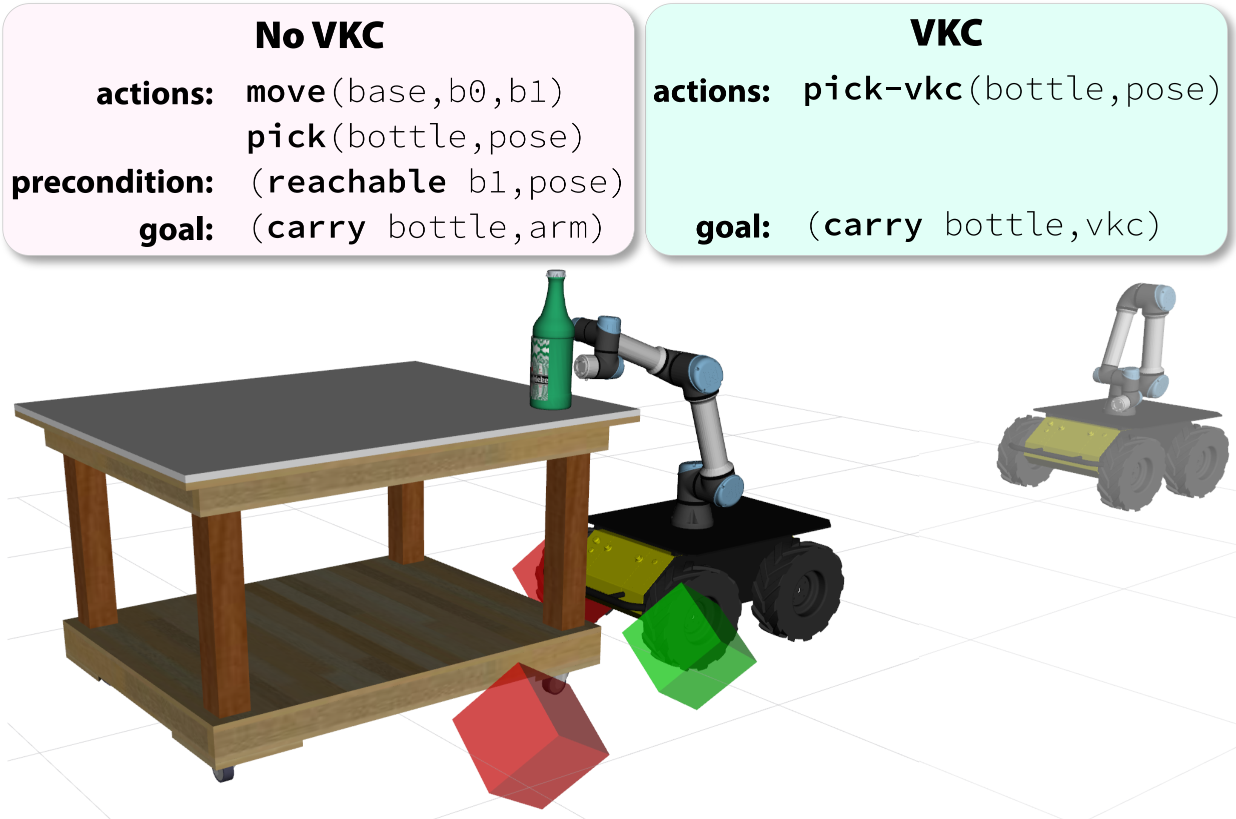


Figure 3.1: **A typical task planning setup, wherein the mobile manipulator is tasked to navigate and pick up the object on the desk.** The VKC-based domain specification reduces the search space by removing the poses of the mobile base near red cubes, resulting in a simpler and more intuitive task planning domain.

(PDDL) with VKC. Compared with conventional domain definition, our VKC-based domain definition is more efficient in both planning time and memory. In addition, abstract actions perform better in producing feasible motion plans and trajectories. We further scale up the VKC-based task planner in complex mobile manipulation tasks and validate these advantages by comparing the VKC-based approach with baselines that solely optimize individual components. Taken together, these results demonstrate that task planning using VKC for mobile manipulation is not only natural and effective but also introduces new capabilities. The materials in this chapter have been published in [JZW21, JZJ21].

3.1 Introduction

As one of the central themes in AI and robotics, task planning is typically solved by searching a feasible action sequence in a domain. Researchers have demonstrated a wide range of successful robotics applications [LaV06, KM20] with effective representations or programming languages, such as STRIPS [FN71], hierarchical task network [NAI03], temporal and-or-graph [EGL19, LZZ19], Markov decision process [Bel57], and PDDL [FL03].

An effective task planner in robotics generally possesses two characteristics. First, the planning domain must be clearly designed, which includes a set of predicates that truthfully describe the environment states, a set of actions that specify how states transit, and a goal specification that indicates the desired result. However, the definitions of these components are tightly coupled; thus, designing the planning domain could be tedious and error-prone. Second, the abstract notion of symbolic actions should be realizable by motion planners; *i.e.*, the design of these abstract symbols should have practical meaning. These two requirements pose additional challenges in task planning for mobile manipulation; the robot consists of a mobile base and an arm, which possess different motion patterns and capabilities.

To clearly illustrate the above challenges, let us take Fig. 3.1 as a concrete example, wherein a mobile manipulator is tasked to navigate and pick up the bottle on the desk. A dedicated set of predicates and actions must be specified for the mobile base and the arm; for instance, moving the *base* (`move(·)`) to a configuration, such that the *arm* can pick up the object (`pick(·)`). Of note, finding such a pose oftentimes requires to specify the mobile base and the arm *individually*. However, this separation in the planning domain is *artificial* in nature and ineffectively introduces an *unnecessarily* larger planning space: The valid poses of the mobile base near the goal (*i.e.*, the bottle) must be specified (indicated by the cubes in Fig. 3.1) in advance, and exactly one (*e.g.*, the green cube) must be selected via sampling or searching under pre-defined heuristic or criteria. This deficiency becomes increasingly evident as the task sequence grows longer and prohibits natural motions that require foot-

Figure 3.2: **Diverse interactions a service robot needs to perform in a household environment.** By abstracting the objects' kinematic structures and forming a VKC, a service robot can plan and act more efficiently with improved foot-arm coordination.

arm coordination; coordinating the base and arm movements remains challenging even for existing whole-body motion planning methods [Sha16, BAK17, CCL10], let alone realizing a symbolic task plan with a feasible motion plan.

In particular, we propose a Virtual Kinematic Chain (VKC) perspective for mobile manipulation, which consolidates the kinematics of the mobile base, the arm, and the object being manipulated into a single kinematic model. By treating the robot as a whole, more abstract actions can be defined to jointly account for both the base and the arm; see `pick-vkc(·)` *vs* `move(·)` and `pick(·)` in Fig. 3.1. Such an abstraction alleviates the manually-defined *heuristic* of where the robot can reach the goal and the unnecessary definitions of *intermediate goals*, *e.g.*, predicates describing the robot's pose before reaching the goal. As a result, this modification of the planning domain reduces the branching factor, making it scalable to more complex tasks. Crucially, the abstraction introduced by VKC does not sacrifice the success rate to generate a solvable motion planning problem.

From this new VKC-based perspective, a mobile manipulation task is represented by altering the state or the structure of the VKC, which leads to a motion planning problem on VKC, formulated and solved by trajectory optimization. This new perspective enables a service robot to plan and act efficiently by allowing it to directly incorporate external objects and plan the motion as a whole to achieve better foot-arm coordination; see examples in Fig. 3.2.

In experiments, we validate the proposed VKC perspective in various mobile manipulation tasks. Our experiments show that the consolidated kinematic models are particularly suitable for robots by alleviating intermediate goal definitions for task planners and motion planners; they offer a simple yet effective intermediate representation for domain specification in task planning and promote coordinated motions among base, arm, and object.

3.1.1 Related Work

VKC in robot modeling and planning The idea of **Virtual Kinematic Chain (VKC)** could be traced back to 1997 by Pratt *et al.* [PDP97] for bipedal robot locomotion [PCT01]. This idea was later adopted to chain serial manipulators to form one kinematic chain [LNZ14] and to dual-arm manipulation tasks; for instance, connecting parallel structures via rigid-body objects [WSK15], modeling whole-body control of mobile manipulators [WSK16]. Recently, VKC is also adopted for wheeled-legged robot control [LHP19]. In this paper, we further push the idea of VKC to a mobile manipulator and demonstrate its advantages in modeling and planning complex manipulation tasks in household environments.

Motion planning is among the largest and most fundamental fields in robotics. In essence, methods can be roughly categorized into three major doctrines: search-based (*e.g.*, A⋆ [HNR68], D⋆ [Ste97]), sampling-based (*e.g.*, RRT [LK00] and its variants [KL00, KF10]), and trajectory optimization (*e.g.*, CHOMP [RZB09], TrajOpt [SDH14]). We formulate the motion planning problem on VKC following the conventions in TrajOpt, as it incorporates kinematic constraints better than sampling-based methods while avoiding searching in large spaces. Efficiently performing mobile manipulation tasks are challenging. Notable efforts have recently been dedicated to algorithms or system implementations, focusing on interactive manipulation tasks. For instance, equilibrium point control [JK10] and impedance control structure [SNT19] are introduced to open doors and drawers. To improve efficiency, Gochev *et al.* used a heuristic-based method to reduce the search space [GSL12]. Taking advantage of solving the inverse kinematics, Burget *et al.* proposed a whole-body motion planning approach for humanoid's constrained motion [BHB13], and Bodily *et al.* proposed an algorithm for jointly optimizing a robot's base position and joint motions [BAK17]. More recently, Toussaint *et al.* proposed a multi-bounded tree search algorithm to solve multi-step manipulation tasks involving tool-use [TAS18]. Despite their promising results, prior arts primarily focus on a specific problem setup (*e.g.*, opening door and drawer, using tools). In comparison, the proposed approach rethinks mobile manipulation from a more general viewpoint using VKCs and tackles a broader range of tasks.

TAMP in mobile manipulation Thanks to the development of PDDL and other planning architectures, complex symbolic task planning can be solved using standard algorithms [KM20]. Hence, the community has shifted the focus to corresponding a valid symbolic action sequence to feasible motions, which leads to the field of TAMP [GCH20]. While researchers tackle this problem from various angles, such as incorporating motion-level constraints to the task planning [EHP11, KL11, GLK18], developing interfaces that communicate between task and motion [SFR14], or inducing abstracted modes from motions [Tou15, TAS18], it remains a largely unsolved problem. In addition, movements of a mobile base and a manipulator are commanded by two or more separate actions [BKL17, GLK18, KWK19], causing increased planning time, less coordinated movements, *etc*. In comparison, the VKC perspective serves as an intermediate representation that benefits the task modeling of mobile manipulation, improves computation efficacy, and facilitates motion planning.

3.2 Virtual Kinematic Chain (VKC) Modeling

3.2.1 Notations and Problem Definition

This section introduces the notations throughout the paper and the problem setup describing a mobile manipulation task.

The physical properties and kinematics of links and joints are defined following the Unified Robot Description Format (URDF) in Robot Operating System (ROS) and organized in a tree representation $\mathcal{T}$. Table 3.1 lists all the related notations:

Below, we further summarize the above notations:

- The group *Robot* refers to notations related to the mobile manipulator, which consists of three components: mobile base, manipulator, and end-effector.
- The group *Object* refers to notations related to the manipulated objects, which could be as simple as a *rigid* link or be an *articulated* object with two or more links connected by either a prismatic, revolute, or fixed joint. We introduce a *virtual* joint defined as an *attachment*, a local transformation ${}^{at}_{ee}T$ from the object's attachable frame $\mathcal{F}^{O}_{at}$ (*i.e.*, the link a mobile manipulator can grasp on) to the robot's end-effector frame $\mathcal{F}^{R}_{ee}$.
- The group *Others* refers to constructed VKC, its state space, and other related notations in a manipulation task.

Table 3.1: **Notations used for constructing VKCs.**

Group	Notation	Description
Robot	$\mathcal{T}^R$	A tree represents the robot kinematic model
	$\mathcal{F}_b^R$	Robot base link's frame; the root of $\mathcal{C}^R$
	$\mathcal{F}_{ee}^R$	Robot end-effector link's frame
	$\mathcal{C}^R$	$\subset \mathcal{T}^R$, a kinematic chain from $\mathcal{F}_b^R$ to $\mathcal{F}_{ee}^R$
	$\mathcal{F}_i^R$	Frame of link i in the kinematic chain $\mathcal{C}^R$
Object	$\mathcal{T}^O$	A tree represents the object kinematic model
	$\mathcal{F}_b^O$	Object base link's frame; the root of $\mathcal{T}^O$
	$\mathcal{F}_{at}^O$	Object attachable link's frame
	$\mathcal{C}^O$	$\subset \mathcal{T}^O$, a kinematic chain from $\mathcal{F}_b^O$ to $\mathcal{F}_{at}^O$
	$\mathcal{F}_i^O$	Frame of link i in the kinematic chain $\mathcal{C}^O$
Others	$\mathcal{C}_n^V$	A serial VKC with n Degree of Freedom (DoF)
	$\mathbf{q}$	$\in \mathbb{R}^n$, the state of VKC in joint space
	$\mathbf{g}$	$\in \mathbb{R}^k$ $(k \leqslant n)$, the joint goal state
	${}^a_bT$	A homogeneous transformation from $\mathcal{F}_a$ to $\mathcal{F}_b$
	${}^w_iT_g$	The goal pose of $\mathcal{F}_i$ in the world frame

Constructing a VKC $\mathcal{C}^V$ requires the inputs of robot kinematic tree $\mathcal{T}^R$, object kinematic tree $\mathcal{T}^O$, and transformation from an object attachable frame to the robot end-effector frame ${}^{at}_{ee}T$. The chain's forward kinematics (FK), inverse kinematics (IK), and Jacobians can be effectively solved by existing kinematic solvers (*e.g.*, KDL [SBA11]).

Assuming a rigid connection between the end-effector and the attachable link during manipulation, performing a mobile manipulation task can be regarded as reaching desired VKC poses. As a result, we treat a mobile manipulation task as a motion planning problem on the VKC and solve it by trajectory optimization. Formally, it is equivalent to finding a collision-free path $\mathbf{q}_{1:T}$ from the initial pose $\mathbf{q}_{init}$ to goals $\mathbf{g}$ in joint space and/or goal poses ${}^{w}_{i}T_{g}$ in Euclidean space.

The objective function of the trajectory optimization can be formally expressed as:

$$\min_{\mathbf{q}_{1:T}} \sum_{t=1}^{T-1} ||W_{vel}^{1/2} \;\; \delta\mathbf{q}_t||_2^2 \; + \sum_{t=2}^{T-1} ||W_{acc}^{1/2} \;\; \delta\dot{\mathbf{q}}_t||_2^2, \tag{3.1}$$

wherein we penalize the overall weighted squared traveled distance of every joint with the finite forward difference $\delta\mathbf{q}_t \approx \mathbf{q}_{t+1} - \mathbf{q}_t$ and overall smoothness of the trajectory with the second-order finite central difference $\delta\dot{\mathbf{q}}_t \approx \mathbf{q}_{t-1} - 2\mathbf{q}_t + \mathbf{q}_{t+1}$. W_{vel} and W_{acc} are diagonal weight matrices for each joint, respectively. $\mathbf{q}_{1:T}$ represents the trajectory sequence $\{q_1, q_2, \ldots, q_T\}$, where $\mathbf{q}_t$ denotes the VKC state at the t^{th} time step.

3.2.2 VKC Construction

The proposed VKC modeling constructs a serial kinematic chain by (i) incorporating both robot and object kinematics via a virtual joint and (ii) augmenting a virtual base to the robot base; see Fig. 3.3b for a graphic illustration.

Below we formally describe the 4-step procedure of constructing the VKC, $\mathcal{C}^V$, by consolidating the robot and the object kinematics models.

Original Structure The kinematic models of the mobile manipulator $\mathcal{T}^R$ and the manipulated object $\mathcal{T}^O$ are assumed given by the perception module or by the simulator.

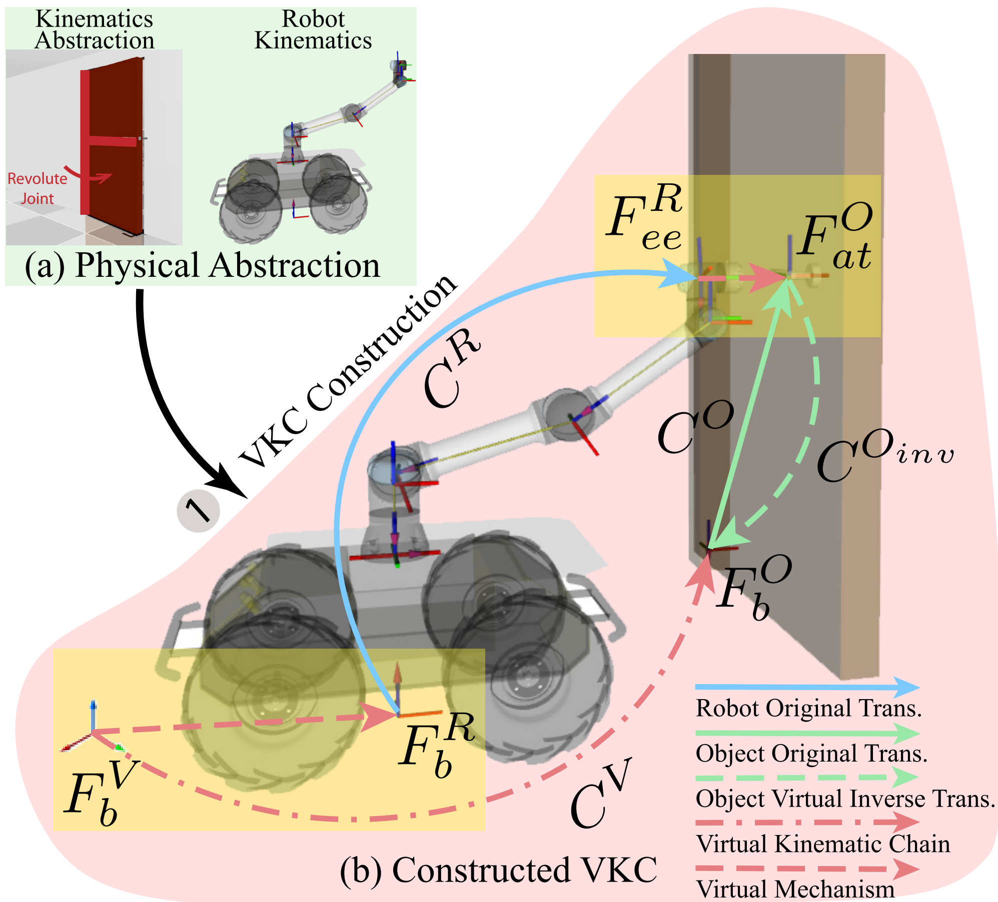


Figure 3.3: **Overview of the mobile manipulation planning schematics using the proposed VKC-based approach.** (a) After abstracting out the underlying kinematics of the manipulated object and the mobile manipulator, (b) a VKC is constructed. The yellow boxes denote where the virtual connections are established: (i) One between $\mathcal{F}_b^V$ and $\mathcal{F}_b^R$, the *virtual* base frame in the world coordinate and the robot's actual base frame, to reflect the navigational information, and (ii) another between $\mathcal{F}_{ee}^R$ and $\mathcal{F}_{at}^O$, the robot's end-effector frame and the attachable frame of the object, to transfer effects of the manipulator to the manipulated object.

Kinematic Inversion Let us take the task of opening a door as an example. In conventional kinematic notation, the door is the child link, and the door frame is its parent link in the original $\mathcal{T}^O$. To construct a VKC, this parent-child relationship needs to be inverted before it can be attached to the robot's end-effector, *i.e.*, the door becomes the parent link that "transforms" the door frame. Of note, such an inversion also requires updating the joint connecting the two links, since a joint (*i.e.*, revolute/prismatic) typically constrains the child link's motion w.r.t. the *child link's* frame.

VKC Construction After inverting the original $\mathcal{T}^O$, a virtual joint between $\mathcal{F}_b^{O_{\text{inv}}}$ and $\mathcal{F}_{ee}^R$ is inserted, whose transformation is denoted as ${}^{ee}_{at}T$. In our application, the transformation of the virtual joint is updated by the actual grasping pose right before the VKC construction to minimize kinematic discrepancies introduced by the execution error. Next, the motion planner will be invoked to plan following motions for the actual VKC. The joint type could also be determined by the grasping type between the gripper and the object (*e.g.*, revolute joint for grasping a cylindrical handle, fixed joint for grasping a rigid ball) to alleviate the inaccuracies during the execution.

Virtual Base Frame A virtual base frame $\mathcal{F}_b^V$ is further added and connected to the mobile base through two perpendicular prismatic joints and a revolute joint, enabling the mobile base's omnidirectional motions on the ground plane.

After the above procedure, the constructed VKC remains in serial and forms an equality constraint to Eq. (3.1):

$$h_{\text{chain}}(\mathbf{q}_t) = 0,\ \forall t = 1, 2, \dots, T \tag{3.2}$$

It specifies the kinematics of the VKC, which includes its forward kinematics and other physical constraints of the manipulated object; *e.g.*, the manipulated object is fixed to the ground, which leads to a closed chain: ${}^{w}_{b}T^{O}_{1:T} - {}^{w}_{b}T^{O} = 0$. Failing to account for this constraint may damage the manipulated object or the mobile manipulator.

3.2.3 Goals of VKC

The goal of the mobile manipulation can be formulated as an inequality constraint, in addition to the equality constraint introduced by the VKC construction in Eq. (3.2):

$$||f_{\text{task}}(\mathbf{q}_T) - \mathbf{g}||_2^2 \leqslant \xi_{\text{goal}}, \tag{3.3}$$

which bounds the squared $l2$ norm between the final state in the goal space $f_{\text{task}}(\mathbf{q}_T)$ and the goal state $\mathbf{g}$ with a tolerance ξ_{goal}. The function $f_{\text{task}} : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is a task-dependent function that maps the joint space of a VKC to the goal space that differs from task to task.

Again, let us take the example of opening a door. In the first phase when the robot is reaching the door handle, $f_{\text{task}}(\cdot)$ maps the joint space of a VKC to the robot's end-effector pose. In this case, the goal $\mathbf{g}$ is the robot's end-effector pose ${}^{w}_{ee}T_g$, and Eq. (3.3) can be rewritten in a simplified form $||f_{\text{fk}}(\mathbf{q}_T) - {}^{w}_{i}T_g||_2^2 \leqslant \xi_{\text{goal}}$. In the second phase when the robot is opening the door, $f_{\text{task}}(\cdot)$ maps VKC's joint space to the joint of the door's revolute axis. Hence, $\mathbf{g}$ is merely the angle θ of the revolute joint, and the trajectory of the other joints in the VKC are implicitly generated by the optimization process, together with obstacle avoidance and trajectory smoothing. Of note, Eqs. (3.2) and (3.3) are not the only forms of constraints that a VKC-based approach can incorporate; in fact, it is straightforward to add additional task constraints to the same optimization problem in Eq. (3.1), depending on various task-specific requirements.

3.2.4 Additional Constraints for VKC

During the trajectory optimization, we further impose several safety constraints. Without loss of generality, we assume an omnidirectional base and purely kinematic constraints in this paper. However, extra constraints, such as nonholonomic constraints for non-omnidirectional mobile bases or dynamic constraints for arms, could be formulated into the optimization problem by incorporating additional time, first-order, or second-order terms [RHB17].

$$\mathbf{q}^{\min} \leqslant \mathbf{q}_t \leqslant \mathbf{q}^{\max}, \quad \forall t = 1, 2, \ldots, T \tag{3.4}$$

$$||\delta \mathbf{q}_t||_\infty \leqslant \xi_{\text{vel}}, ||\delta \dot{\mathbf{q}}_t||_\infty \leqslant \xi_{\text{acc}}, \quad \forall t = 2, 3, \ldots, T-1 \tag{3.5}$$

$$\sum_{i=1}^{N_{\text{link}}} \sum_{j=1}^{N_{\text{obj}}} |\text{dist}_{\text{safe}} - f_{\text{dist}}(L_i, O_j)|^+ \leqslant \xi_{\text{dist}}, \tag{3.6}$$

$$\sum_{i=1}^{N_{\text{link}}} \sum_{j=1}^{N_{\text{link}}} |\text{dist}_{\text{safe}} - f_{\text{dist}}(L_i, L_j)|^+ \leqslant \xi_{\text{dist}}. \tag{3.7}$$

Eq. (3.4) is an inequality constraint that defines joint limits, in which $\mathbf{q}^{\min}$ and $\mathbf{q}^{\max}$ specify the lower and upper bound of every joint, respectively. Eq. (3.5) is an inequality constraint that bounds the joint velocity by ξ_{vel} and the joint acceleration by ξ_{acc} to obtain a feasible trajectory that can be executed without saturation. $||\cdot||_\infty$ denotes the infinity norm.

Eqs. (3.6) and (3.7) are inequality constraints that check link-object collisions and link-link collisions, respectively, where N_{link} and N_{obj} are the number of links and the number of objects, respectively. $\text{dist}_{\text{safe}}$ is a pre-define safety distance, and $f_{\text{dist}}(\cdot)$ is a function that calculates the signed distance [SDH14] between i-th link L_i and j-th object O_j; the function $|\cdot|^+$ is defined as $|x|^+ = \max(x, 0)$.

The inequality constraints introduced by Eqs. (3.6) and (3.7) make the preceding optimization problem highly non-convex and unsolvable by a generic convex solver. In this paper, we approximate it by a sequence of convex problems [SDH14], solved by a sequential convex optimization method.

3.2.5 Advantages of VKC

As formally derived in the above sections, solving mobile manipulation as trajectory optimization using the proposed VKC-based approach introduces two advantages:

1. **Eliminating unnecessary intermediate goals**. Let us use the example of opening a door: Only one goal—the door's angle to be opened to—is required. The final poses of

the mobile base and the manipulator are directly produced during the trajectory optimization process without manually specifying unnecessary intermediate goals. Hence, the VKC-based approach provides versatility and simplicity for modeling mobile manipulation tasks.

2. **Coordinating locomotion and manipulation**. Using VKCs, the trajectory optimization jointly generates trajectories of the mobile base and the manipulator, producing coordinated locomotion and manipulation, which is oftentimes challenging for conventional methods.

These two advantages are crucial for a robot operating in a complex domestic environment. In the following sections, we demonstrate these advantages in a series of mobile manipulation tasks.

3.3 Planning on VKC

3.3.1 Task Planning on VKC

Following the classic formalization of task planning, we describe the environment by a set of states $\mathcal{S}$. Possible transitions between these states are defined by $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$, where a transition $t = \langle s, s' \rangle \in \mathcal{T}$ alters the environment state from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$. The goal of the task planning problem is to identify a sequence of transitions that alters the environment from its initial state $s_0 \in S$ to a goal state $s_g \in S_g$, where $S_g \subseteq S$ is a set of goal states.

We primarily consider the task planning problems in mobile manipulation, which require the robot to account for its base, arm, and the object being interacted (*e.g.*, pick and place, door/drawer opening). We formulate the task planning problems and implement the planning domains using PDDL.

In PDDL, the environment state s is described by a set of *predicates* that hold true. Specifically:

- `(vkcState ?r ?q)`: A sub-chain `?r` (*e.g.*, the base, an arm, or even a VKC) of a VKC is at configuration `?q` in joint space.
- `(objConf ?o ?s)`: An object `?o` is at the configuration `?s` in SE(3).
- `(free ?v)`: The robot end-effector is free to grasp.
- `(carry ?o ?v)`: The robot end-effector is carrying an object `?o`.

In this paper, to focus on demonstrating the benefit of task planning with VKC, we pre-sampled feasible configurations `?s` for all objects and corresponding grasping poses.

Transitions in PDDL are modeled by `actions`. Each `action` takes parameters as input and can be called only when its preconditions hold true. After an `action` is called, its effect indicates how the states in the current environment change from preconditions. Thanks to the advantages introduced by the VKC, three simple action definitions—`goto-vkc`, `pick-vkc`, and `place-vkc`—are sufficient to handle various mobile manipulation tasks, from pick-and-place in different setups to foot-arm coordinated and constrained motions (*e.g.*, door/drawer opening). Below is an example of the definitions for three actions; see Figs. 3.5b and 3.5c and Section 3.4.1 for a comparison between VKC-based PDDL and a standard PDDL for mobile manipulators.

```
(:action goto-vkc
:parameters (?r ?from ?to)
:precondition (vkcState ?r ?from)
:effect (and (vkcState ?r ?to)
                (not (vkcState ?r ?from))))

(:action pick-vkc
:parameters (?o - obj ?s - state ?v - vkc)
:precondition (and (objConf ?o ?s)
                    (free ?v))
:effect  (and (carry ?o ?v)
```

```
                    (not (objConf ?o ?s))
                    (not (free ?v))))

(:action place-vkc
:parameters (?o - obj ?s - state ?v - vkc)
:precondition (and (carry ?o ?v)
                    (not (occupied ?s)))
:effect  (and (not (carry ?o ?v))
                    (objConf ?o ?s)
                    (free ?v)))
```

3.3.2 From Task to Motion

The conventional task planning setup usually assumes a robot already knows how to execute the actions defined in the task domain and, therefore, does not generate actionable motion trajectories for the robot. However, in practice, this assumption does not always hold as many abstract actions defined in the task domain are difficult to be instantiated at the motion level. This section discusses how the `actions` defined using VKC can properly form a motion planning problem solvable by existing motion planners.

We start by making the connections between the action semantics and the actual manipulation behaviors, followed by explaining how the predicates and variables in the action definitions are processed by motion planners.

`goto-vkc` (r, $\mathbf{q_1}$, $\mathbf{q_2}$) This predicate moves the VKC from the current pose q_1 to a desired pose q_2 for a chain `r`. It represents the tasks that do not require interaction with the environment, wherein the VKC structure remains unchanged. Pure navigation is a typical action falling into this category. For example, `goto-vkc` (`base`, q_1^b, q_2^b) moves the robot to the location specified in q_2^b. Another example is to manipulate a picked object from the current pose q_1 to a certain pose q_2, *i.e.*, `goto-vkc` (`vkc`, q_1, q_2)

`pick-vkc (object, s, vkc)` This predicate moves the VKC to the `object` to be manipulated and extends the current VKC structure by adding a virtual joint to connect the `object` and the arm's end-effector at state s. Here, the state could be interpreted as a grasping pose, the transformation between the robot gripper and the object to be manipulated (*i.e.*, ${}^{ee}_{at}T$). `pick-vkc` represents the group of tasks that require mobile manipulators to interact with the environment, *e.g.*, picking up an object or grasping a handle.

`place-vkc (object, s, vkc)` This predicate moves the `object` connected to `vkc` to a goal pose s, while the object to be manipulated is incorporated into the VKC and imposes kinematic constraints to the planner. Once reaching the goal pose, `place-vkc` breaks the current VKC at the virtual joint where it connects the mobile manipulator and the `object`, and the `object` will be placed at where it was disconnected from VKC. `place-vkc` represents the group of tasks that mobile manipulators stop interacting with the environment, such as placing an object on the table.

In motion planning, configuration space Q describes the environment state. Q's dimension n equals to VKCs' degrees of freedom. A collision-free subspace $Q_{\text{free}} \subseteq Q$ is the space that VKCs can traverse freely without colliding with the environment or itself. The problem of motion planning on VKC is equivalent to finding a collision-free path $\mathbf{q}_{1:T} \in Q_{\text{free}}$ from the initial pose $\mathbf{q}_1 \in Q_{\text{free}}$ to reach the final state $\mathbf{q}_T \in Q_{\text{free}}$. Each action predicate requires to form a motion planning problem due to the kinematic structure changes.

3.3.3 Optimization-based Motion Planning

Finding a collision-free path $\mathbf{q}_{1:T} \in Q_{\text{free}}$ for given tasks can be formulated by trajectory optimization, *e.g.*, CHOMP [RZB09] and TrajOpt [SDH14]. The objective function of the trajectory optimization can be formally expressed as Eq. (3.1) where we penalize the overall velocities and acceleration of every joint with diagonal weights W_{vel} and W_{acc} for each joint, respectively.

Meanwhile, the constructed VKC should also be subject to kinematic constraints of the robot and the environment described in Eq. (3.2) which includes forward kinematics and closed chain constraints. We can formulate the task goal as an inequality constraint described in Eq. (3.3) which bounds the element-wise squared ℓ^2 norm between the final state in the goal space $f_{\text{task}}(\mathbf{q}_T)$ and the task goal $\mathbf{g} \in \mathbb{R}^k$ $(k \leqslant n)$ with a tolerance ξ_{goal}. The function $f_{\text{task}} : Q \to \mathbb{R}^k$ is a task-dependent function that maps the joint space of a VKC to the goal space that differs from task to task. This definition relaxes hard constraints of goal state and optimized the other $n - k$ states with objective function Eq. (3.1). Of note, Eqs. (3.2) and (3.3) are not the only forms of constraints that a VKC-based approach can incorporate; in fact, it is straightforward to add additional task constraints to the same optimization problem in Eq. (3.1), depending on various task-specific requirements. We further impose several additional safety constraints (see Section 3.2.4), including joint limits, bounds for joint velocity and acceleration, and link-link and link-object collisions.

3.3.4 Sampling-based Motion Planning

Alternatively, motion planning on VKC can also be viewed as a search procedure in the configuration space $\mathcal{Q}_{\text{free}}$. Given a path planning problem within $\mathcal{Q}_{\text{free}}$, a sampling-based method would attempt to find a set of collision-free way points that start from an initial configuration $\mathbf{q}_0 \in \mathcal{Q}_{\text{free}}$ and end in the goal configuration $\mathbf{q}_{\text{goal}} \in \mathcal{Q}_{\text{free}}$.

Rapidly-exploring Random Tree (RRT) is a probabilistically complete search algorithm that incrementally expands a collection of directional nodes $\mathcal{T}$ to explore space [LK00]. In this paper, we adopted a RRT-connect algorithm [KL00] from the Open Motion Planning Library (OMPL) [SMK12] as our sampling-based motion planner, which initiates exploration from $\mathbf{q}_0$ and $\mathbf{q}_{goal}$ concurrently.

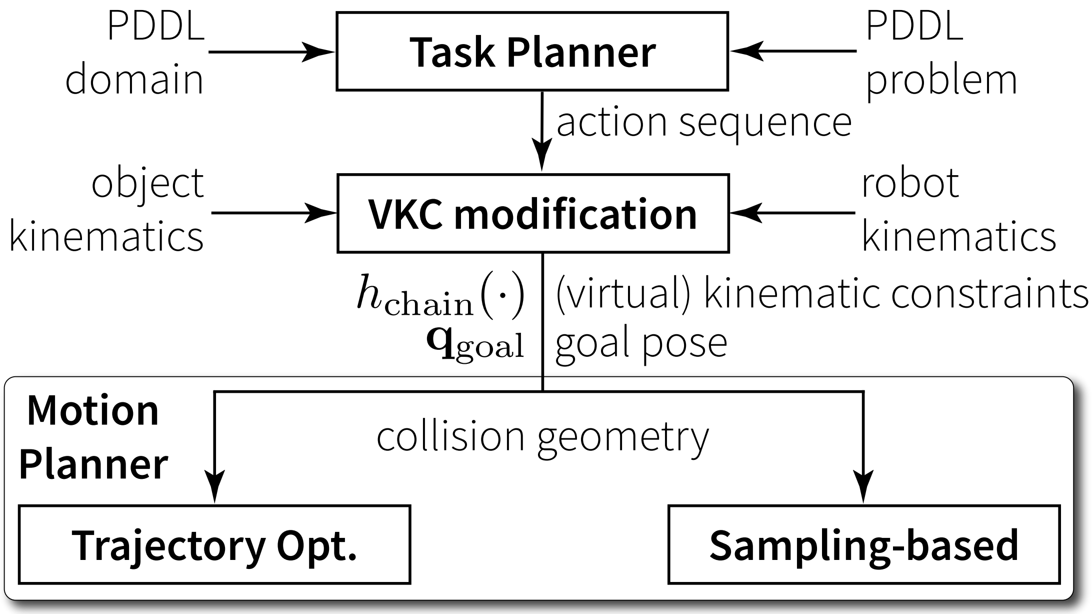


Figure 3.4: **The computing logic of instantiating the `actions` in a task plan to trajectories at the motion level.** Each action symbol encodes a (virtual) kinematic chain and a goal pose, which are sufficient for a motion planner given the environmental constraints.

Unlike the optimization-based method mentioned in Section 3.3.3, way-points collected by RRT-connect are not smoothed by an objective function during search; instead, interpolation was performed after the search is complete for a smooth trajectory to be executed on a mobile manipulator.

Fig. 3.4 summarizes the computing logic of instantiating the `actions` to motion trajectories. The action sequence produced by the task planner encodes how the VKC changes over each action and its desired goal pose. Together with environmental constraints (*e.g.*, the actual robot kinematics and the objects' geometry), the information provided by the VKC-based task planner is sufficient for a typical motion planner to produce a feasible trajectory from $\mathbf{q}_0$ to $\mathbf{q}_{\text{goal}}$.

3.4 Experiment

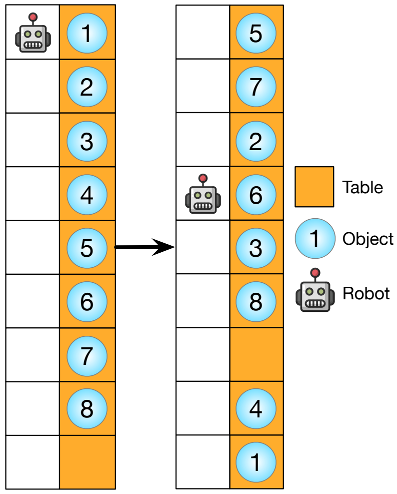
We conduct a series of experiments to evaluate the efficacy of the proposed VKC perspective for planning mobile manipulation tasks in simulations. The first experiment compares the designs of PDDL definition with VKC or without VKC and their corresponding planning efficiency. Since the action definitions can be arbitrarily abstract at the symbolic task level, we further validate the VKC-based action design in the second experiment that it indeed provides sufficient information for motion planners to produce feasible trajectories. Finally, in the third experiment, we showcase how the VKC perspective empowers more complex task planning.

3.4.1 Simplifying Task Domain

Since the VKC perspective treats the base, the arm, and the object to be manipulated as a whole, designing the planning domain becomes much simpler. In this experiment, we focus on an object-arrangement task, where the robot is tasked to re-arrange m objects on $m + 1$ tables into the desired order while satisfying the constraint that each table can only support one object. Fig. 3.5a shows a typical example of this task's initial and goal configuration with $m = 8$ objects, randomly sampled in each experimental trial.

Fig. 3.5b shows a PDDL domain designed by the actions mentioned in Section 3.3.1, which requires less predicates and provides more abstract actions compared with those designed by conventional domain definition shown in Fig. 3.5c. Specifically, the conventional method would require (i) more predicates to describe the mobile base's states and thus more complex preconditions for actions, (ii) one more action to control the mobile base, and (iii) more parameters for other actions. To solve for a task plan, we adopt the Iterated Width Search (IWS) algorithm [LG14]; it is a width-limited version of the Breadth First Search (BFS) that repeatedly runs with increasing width limits until a feasible task plan is found. If no feasible task plan could be found within the maximum width limit of the IWS, a traditional

BFS with no width limit will be deployed to search for a solution.



(a) Experimental Setup

```
(:predicates
    (objConf ?obj - block ?s - state)
    (free ?a - vkc)
    (carry ?obj - block ?a - vkc)
    (occupied ?s - state))

(:action pick_vkc
    :parameters (?obj - block ?s - state
                 ?arm - vkc)
    :precondition (and (objConf ?obj ?s)
                       (free ?arm))
    :effect (and (carry ?obj ?arm)
                 (not (objConf ?obj ?s))
                 (not (free ?arm))
                 (not (occupied ?s))))

(:action place_vkc
    :parameters (?obj - block ?s - state
                 ?arm - vkc)
    :precondition (and (carry ?obj ?arm)
                       (not (occupied ?s)))
    :effect (and (not (carry ?obj ?arm))
                 (objConf ?obj ?s)
                 (free ?arm)
                 (occupied ?s))))
```

(b) VKC-based PDDL

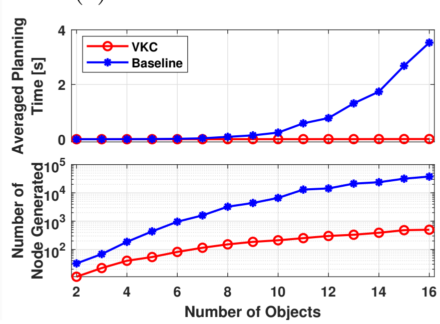
```
(:predicates
    (bConf ?qb - bstate)
    (objConf ?obj - block ?s - state)
    (free ?a - arm)
    (carry ?obj - block ?a - arm)
    (occupied ?s - state)
    (reachable ?s - state ?qb - bstate))

(:action pick
    :parameters (?obj - block ?s - state
                 ?arm - arm ?qb - bstate)
    :precondition (and (objConf ?obj ?s)
                       (bConf ?qb)
                       (reachable ?s ?qb)
                       (free ?arm))
    :effect (and (carry ?obj ?arm)
                 (not (objConf ?obj ?s))
                 (not (free ?arm))
                 (not (occupied ?s))))

(:action place
    :parameters (?obj - block ?s - state
                 ?arm - arm ?qb - bstate)
    :precondition (and (carry ?obj ?arm)
                       (bConf ?qb)
                       (reachable ?s ?qb)
                       (not (occupied ?s)))
    :effect (and (not (carry ?obj ?arm))
                 (objConf ?obj ?s)
                 (free ?arm)
                 (occupied ?s)))

(:action move
    :parameters (?from ?to - bstate)
    :precondition (bConf ?from)
    :effect (and (bConf ?to)
                 (not (bConf ?from))))
```

(c) Conventional PDDL



(d) Performance Comparison

Figure 3.5: **VKC-based domain specification improves the task planning efficacy.** (a) An example setup of re-arranging 8 objects on 9 tables; one table can only support one object. (b) The VKC-based PDDL specification has less variables and more abstract actions than (c) a conventional PDDL specification. (d) The VKC-based domain specification allows a solver to search for a feasible plan for tasks of re-arranging 2 to 16 objects with significantly less time and generated nodes in search (*i.e.*, less memory).

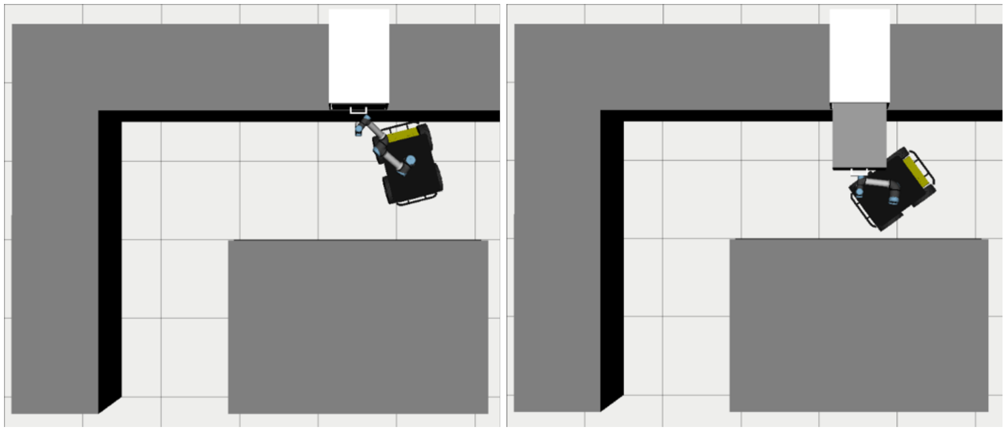
In experiments, we run 50 trials for each setup; see the result summary in Fig. 3.5d. As the task complexity increases, the average planning time and the number of nodes generated in search (*i.e.*, memory required) increase relatively slowly for the VKC-based task plan. In comparison, the baseline using conventional methods increases much more rapidly.

This result is evident. As we can see in Fig. 3.5d, planning in the non-VKC version of the task domain requires exploring more nodes at each depth level to find a plausible pose for the mobile base. It also requires more actions to accomplish the task, which further yields a deeper depth during the search. Suppose there are N nodes on average to be generated at each depth level of the search algorithm, and a feasible solution is found at depth d, the total number of nodes being generated is N^d. In theory, when the search algorithm performs in the VKC domain, the total number of generated node is $(c_1N)^{c_2d}$, where $c_1 \leqslant 1, c_2 \leqslant 1$. In the task with 16 objects, our experiment empirically finds $c_1 = 0.75$ and $c_2 = 0.22$ on average over 50 trails.

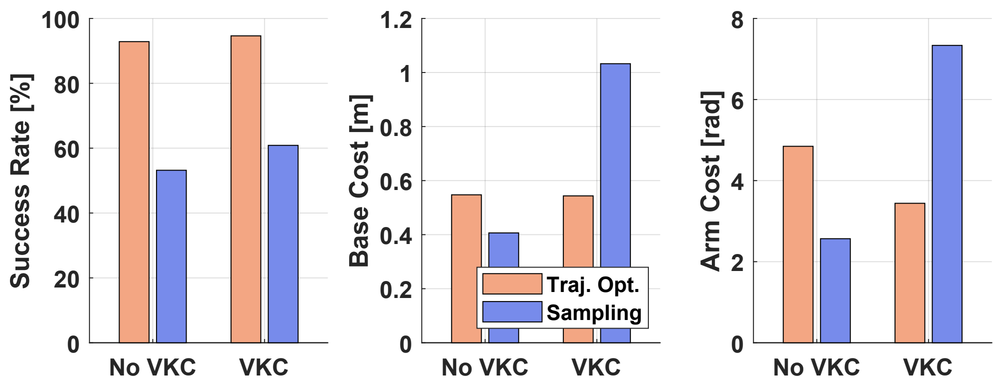
Taken together, the results in the first experiment demonstrated that VKC-based task planning requires much fewer explorations in both width and depth during the search algorithm, therefore achieving higher efficacy with less memory.

3.4.2 Improving Mobile Manipulation

In general, actions that are more abstract and with fewer variables in the planning domain specification would lead to more efficient task planning, but simultaneously could result in less success rate in generating feasible plans at the motion level. In this experiment, we validate that the VKC-based task planning provides efficacy at the task level and maintains a high success rate at the motion level. Based on the generated task plans (*i.e.*, action sequences) and the encoded information (as described in Section 3.3.2), we apply a trajectory optimization-based motion planner and a sampling-based motion planner and evaluate how well they can produce feasible motion trajectories for the given task.



(a) The drawer opening task.



(b) Success rates and the corresponding base and arm movements.

Figure 3.6: **Instantiating the task plans to motions** in (a) a drawer opening task. The domains, one with VKC and the other without, are specified similar to Figs. 3.5b and 3.5c. The generated task plans are processed by an optimization-based and a sampling-based motion planner. (b) Task success rates, and base and arm costs. Failure cases for sampling include time-out for both sub-tasks: 5 mins for reach, 50 seconds for open

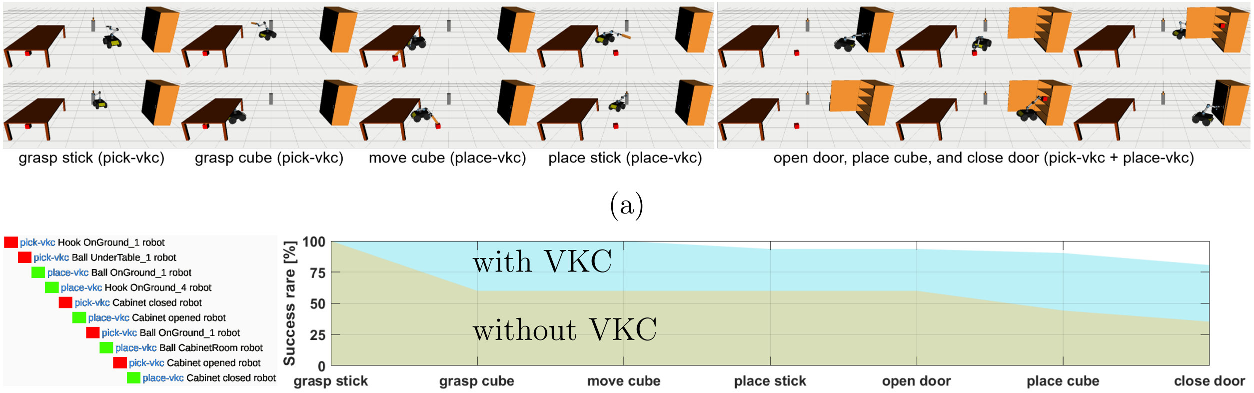
Specifically, we consider the task of pulling opening a drawer; see Fig. 3.6a. The task plans: (i) `place-vkc` (`drawer`, $s_{\mathtt{d}}^1$, `vkc`), (ii) `move` ($q_{\mathtt{b}}^0$, $q_{\mathtt{b}}^1$) + `place` (`drawer`, $s_{\mathtt{d}}^1$, `arm`, $q_{\mathtt{b}}^1$), are produced by two PDDLs specified with and without VKC, respectively. We compare the success rate of executing the trajectories planned by trajectory optimization and sampling motion planning methods described in Sections 3.3.3 and 3.3.4, as well as the base and arm cost measured by the distances they travel; see Fig. 3.6b.

Both trajectory optimization-based and sampling-based motion planners can produce feasible trajectories for the given task with high success rates. Of note, the symbolic actions that are more abstract based on VKC further guide motion planners to produce more efficient trajectories measured by the shorter arm traveling distance. The trajectory optimization-based motion planner can produce feasible trajectories for the given task with high success rates and produce more efficient trajectories measured by the shorter base and arm traveling distance. Typically, sampling-based motion planners would struggle in incorporating kinematic constraints, making it less suitable for the VKC setup. But it is still more successful in producing feasible trajectories under the VKC specification compared with that without VKC. The most significant drawback of the sampling-based motion planner we discover is that the produced trajectories are jerking, resulting in larger arm and base costs.

The trajectory optimization-based motion planner can produce feasible trajectories for the given task with high success rates; the produced trajectories are more efficient in terms of shorter base and arm traveling distances. Typically, sampling-based motion planners would struggle in incorporating kinematic and safety constraints due to naturally unconstrained configuration spaces, which need extra effort to accommodate extra kinematic constraints [KMK19], making it less suitable for such tasks. However, it is still more successful in producing feasible trajectories under the VKC specification compared with the setting without VKC. The most significant drawbacks of sampling-based motion planners are the high execution costs and violation of safety limits.

3.4.3 Solving Tasks with Multiple Steps

Complex multi-step mobile manipulation tasks with long action sequences can also be easily accomplished using the action set introduced by the VKC-based task planner described in Section 3.3.1. These actions contain high-level task semantics that could be adapted to various tasks; *e.g.*, attaching to the doorknob could be expressed by a `pick-vkc` action, and



(a) (b) (c)

Figure 3.7: **Experimental results of planning via VKC.** (a) The VKC-based task planner can easily scale up to a complex multi-step task, which can be (b) succinctly expressed by merely two actions defined based on the VKCs. (c) More abstract action definitions introduced by VKC instantiate better at the motion level, possessing an excellent foot-arm coordination in each step of the task. Without VKC, to ensure successful planning for tasks that require foot-arm coordination, several actions must be executed together to complete certain steps in the task.

open the door to a certain angle could be expressed by a `place-vkc`.

Fig. 3.7a qualitatively shows a complex multi-step task planning using the VKC-based domain specification and instantiating that to motions. For a more fair comparison, in addition to the initial and goal state of the environment, both the VKC and non-VKC methods are provided with the (identical) grasping poses for all movable objects, but not the corresponding robot state. In this task, a mobile manipulator needs to (i) grasp the stick, (ii) fetch the cube under the table using the stick, which is otherwise challenging to reach, (iii) move the cube outside, (iv) place the stick down, (v) grasp the cabinet and open it, (vi) place the cube inside the cabinet, and (vii) close the cabinet door. At each trial, the mobile manipulator is randomly placed in the environment.

Fig. 3.7b illustrates that the above complex multi-step task can be accomplished by us-

Table 3.2: **Actions and predicates in the defined planning domains.** Without VKC, more actions must be specified, and extra predicates are required for generating a feasible task plan.

Setup	Group	Notation	Description
VKC	Actions	`pick-vkc`(o, s, v) `place-vkc`(o, s, v)	see Section 3.3.1
	Predicates	`Graspable`(o, v)	Check if robot v is able to grasp object o v
		`RigidObj`(o)	Check if object o is rigid object
		`ArtiObj`(o)	Check if object o is articulated object
		`ToolObj`(o)	Check if object o could be used as a tool
		`Occupied`(s)	Check if a position s being occupied
		`Carried`(o)	Check if an object o is carried by robot
		`ContainSpace`(o, s)	Check if a position s being contained in the object o
Non-VKC	Actions	`move`(s_1, s_2)	Move robot from s_1 to s_2
		`pick`(o, s_1, s_2)	Pick the object o at location s_1 given robot state s_2
		`place`(o, s_1, s_2)	Place the object o at location s_1 given robot state s_2
		`open`(o, s_1, s_2)	Open the object o at location s_1 given robot state s_2
		`close`(o, s_1, s_2)	Close the object o at location s_1 given robot state s_2
	Extra Predicates for Non-VKC	`HasTool`(s)	Check if the robot at current state s holding a tool
		`AbleToPick`(s)	Check if the robot at state s is able to do pick action
		`Reachable`(o, s)	Check if object o is reachable by mobile base at state s

ing only two abstract actions defined based on VKC, one action in each step. Without the VKC perspective, significantly more effects must first be devoted to designing the planning domain. Furthermore, to ensure successful planning of actions that require foot-arm coordination, each step may require several actions to be executed together; see Table 3.2 for a comparison between the two setups. Even after the additional efforts of specifying base pose from a feasible region, its accumulated success rate at the motion level produced by the corresponding actions still underperforms the VKC version, shown in Fig. 3.7c. Without VKC, the motion planner particularly suffers at step 2 when the robot needs to fetch the cube in a confined space, as it requires the planner to deliver proper navigation and manipulation with excellent foot-arm coordination (*i.e.*, coordinating `move` and `place`). In sum, this experiment demonstrates that VKC-based mobile task planning for mobile manipulation tasks is advantageous by simplifying domain specification and improving motion planning.

3.5 Discussion and Conclusion

We presented a modeling method that incorporates the kinematics of a robot's mobile base, arm, and the manipulated object in VKCs. From this new perspective, a mobile manipulation task is regarded as a planning problem on VKCs. Particularly, the motion planning on VKC is solved by trajectory optimization. This approach alleviates the definition of intermediate goals and well coordinates base and arm movements, resulting in a higher success rate with more efficient trajectories in various mobile manipulation tasks. On the other hand, in task planning, more abstract action symbols become possible and fewer predicates/variables/intermediate goals are required in designing the planning domain when introducing the VKC. In a series of experiments, we demonstrate that incorporating VKC in robot planning facilitating the manipulation of geometry fluent in various environment. The VKC-based domain specification using PDDL supports more efficient task planning, works better with existing motion planners, and scales up to more complex tasks compared with the one without VKC. We argue the proposed VKC perspective has significant potential in promoting mobile manipulation in real-world daily tasks.

CHAPTER 4

Understanding the Topology Fluent via an Attributed Grammar

Modeling and understanding objects is the crux of computer vision and robot manipulation. Prior methods primarily focus on treating objects as a whole, which have made tremendous success recently by discriminating object shape (*e.g.*, recognition) or tracking object pose (*e.g.*, manipulation). However, objects can sometimes break into pieces (*e.g.*, object fragmentation), violating the assumption of "object-as-a-whole". This common phenomenon has been largely neglected in recent literature.

In this dissertation, we model the event of topology fluent changes (*e.g.*, fragmentation) using an attributed stochastic grammar model. A probabilistic framework is devised to induce such a grammar from observation; this learned grammar and its probability model serve as a new indication of object status during topology fluent changes, and are useful for downstream tasks. In the experiments, we demonstrate the efficacy of the proposed method by reasoning about the fragmentation retrospectively and by planning for object fragmentation tasks in unseen setups.

4.1 Introduction

Modeling and understanding object is one of the most fundamental problems in computer vision and robot manipulation. In literature, object modeling can be categorized into two primary schools of thoughts: (i) appearance- or geometry-based approaches, including re-

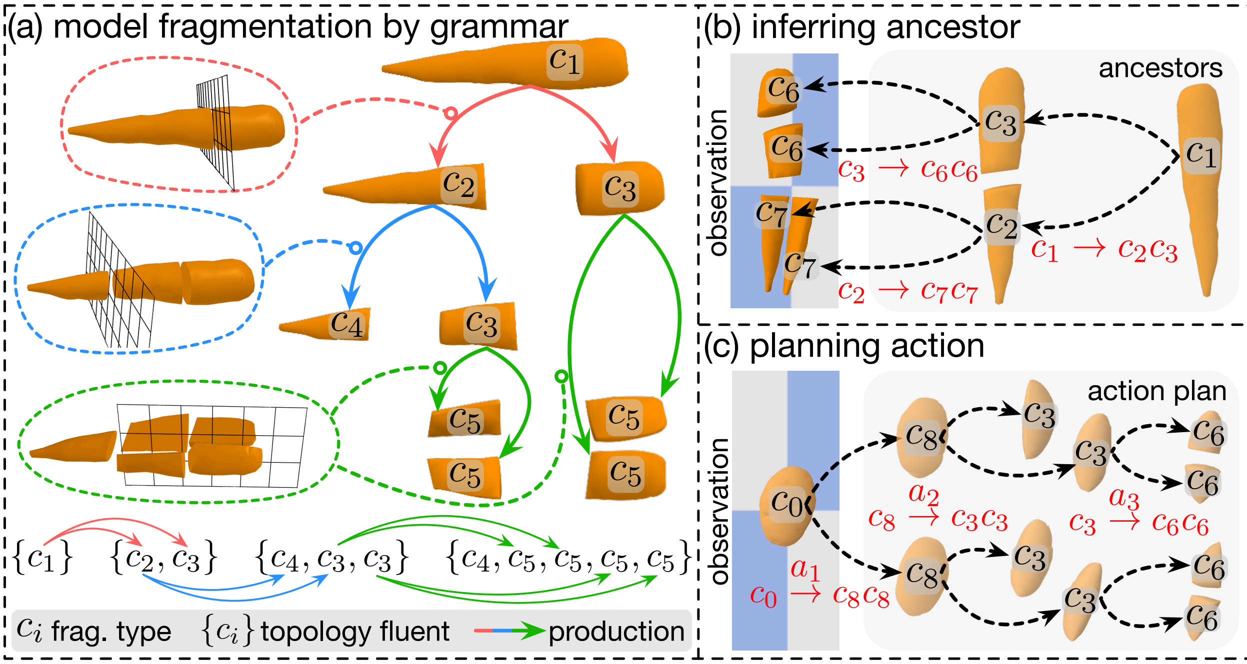


Figure 4.1: **Model object fragmentation by a stochastic grammar.** (a) The proposed grammar-based model represents not only the abstracted change of object status (fluent) by variables but also the part-whole relations and one-to-many transitions (its fluent space) by production rules, resulting in a compact and flexible description of fragmentation events. (b-c) Two tasks used to evaluate the grammar representation: inferring ancestors of fragmented objects and planning for fragmentation sequences, respectively.

construction [MSK04], object recognition and detection [HZR16], and (ii) task-oriented approaches, including object generalization [IH92, ZZZ15, LWZ17, ZJW22], robot manipulation [LZZ19], and grasping [LLJ21]. Despite rapid progress, the primary focus is restricted to **rigid** objects, represented by point clouds, meshes, voxel grids, or graphical models.

Recently, two emerging directions greatly expand an agent's manipulation capabilities: (i) **deformable** object modeling, solved by physics-based simulation [TPB87, HLS19, LHL22] or tactile sensing [SWD21], and (ii) **articulated** object modeling, including human pose estimation [FGM10, CSW17], kinematics estimation [HZJ21, JLC21], and part-based recognition and tracking [MZC19, WWZ21, HWB21].

However, either rigid, deformable, or articulated object modeling treats an object **as a whole**; modeling objects with topology changes (*i.e.*, object fragmentation) is still largely an unexplored area with its unique challenges:

1. A fragmenting object involves significant changes in **configuration**, including instance (fragment) number, shape, and even appearance. Hence, the change of object status, **fluent** [New36], is challenging to define. The corresponding **fluent space** includes both a large number of fluent values and complex one-to-many fluent transitions.
2. Our perception of object fragments is altered and transited when the fragments appear individually, collectively, or temporally, similarly to the entropy principle in natural images [WGZ08].

To overcome these challenges, a desired object representation should be reconfigurable and extendable to account for the drastic fluent changes when an object is fragmented. Further, the representation should properly discriminate fragments under different contexts, from strictly separating every fragment from each other (as texton [Jul84, ZGW05]) to loosely tracking the collection of fragments (as texture [Jul81, ZWM98]), enabling efficient computations within the large fluent space.

In this dissertation, we represent the object fluent during fragmentation and its fluent space using a stochastic grammar. Successful in modeling scenes [ZM07, HQZ18, QZH18] and dynamic events [EGL19, QJH20, ZZZ15], a grammar consists of a set of production rules that generates terminal or non-terminal variables from existing non-terminal ones, akin to an object breaking into pieces—the original object generates newly appeared fragment instances. Specifically, the grammar presents all possible **configurations** an object may finally be as fragmentation repeats; the **production rules** of the grammar indicate all valid one-to-many transitions; and each **parse tree** of the grammar reflects a specific fragmentation process, whose **terminal nodes** correspond to all the fragments in the current configuration.

Fig. 4.1a gives an example of cutting a carrot, where the carrot is fragmented multiple times due to the cutting actions. The recursive and compositional nature of grammar allows us to compactly and flexibly model its fluent and fluent spaces. In addition to encoding fragments into feature embeddings based on their geometric shape, we further cluster fragment features to obtain a much smaller set of variables and production rules. Crucially, the cluster number is determined such that the resulting grammar seeks to reduce its complexity by having less types of variables while preserving the necessary discriminability of fragments. This top-down view provides a new quantification of fragmented objects—we term it as **fragment ensemble**, wherein two groups of fragments are considered the same when their statistics are matched, akin to the Julesz Ensemble [Jul62] that defines textures.

In the experiments, we demonstrate the efficacy of the proposed grammar-based representation for object fragmentation on a perception task and an action task: (i) Reason about the fragmentation event from fragments in retrospect; see Fig. 4.1b. (ii) Plan efficient fragmentation sequence to reach goal configurations at the fragment ensemble level for far-transfer cases; see Fig. 4.1c. By providing a means to model fluent and the transition within the fluent space, our representation not only provides a new view of object modeling beyond object-as-a-whole but also enables a new capability of transforming objects with significant topology changes.

4.2 Grammar Representation

Problem definition of object fragmentation An object fragmentation event $r_o : \Omega_o \to \Omega_o$ can be regarded as transforming 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 a configuration of objects (fragments), $1 \leqslant |\mathcal{I}^{\text{pre}}| \leqslant |\mathcal{I}^{\text{post}}|$, and $\Omega_o = \{\mathcal{I}\}$ is the configuration space. An o_i represents an object or fragment by its shape (*e.g.*, point cloud), pose, *etc.*

Since the configuration space Ω_o is extremely large and complex as every fragment could have different shape, we instead regard some fragments $o_i \in \mathcal{I}$ as the same type $c_j \in C$ via a mapping $f : O \to C$, where $f(o_i) \to c_j$, and $S = \{c_j\}$ defines an object fluent (of the initial whole object or the collection of object fragments). As such, we obtain a simpler fluent space $\Omega_s = \{S\}$ that depicts a fragmentation $r_s : \Omega_s \to \Omega_s$ with a better abstraction.

Representing fragmentation by grammar We use an attributed stochastic grammar [PNZ17], wherein the terminal variables with their attributes represent the fragments, and the production rules capture the valid fragmentation. Formally, an 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 the non-terminal variable that denotes a fragment's type $c \in C$, $v_T \in V_T$ is the 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, which enables a production rule to describe an arbitrary fragmentation within the domain of $V_{NT} \cup V_T$. A fluent S is defined by terminals generated from a parse tree *pt* of $\mathcal{G}$, and the fluent space is define by $\Omega_s = L(\mathcal{G})$, where $L(\mathcal{G})$ represents the set of all possible strings generated by $\mathcal{G}$. Intuitively, a parse tree *pt* of $\mathcal{G}$ represents a plausible fragmentation sequence: the collection of terminals corresponds to current fragments, and the non-terminals indicate the intermediate fragments in the past that subsequently fragment into the current configuration due to the sequence of applied production rules.

4.3 Grammar Learning

We propose to learn stochastic grammar from object fragmentation events generated by human demonstrations.

Corpus generation Given a set of fragmentation events where object and fragment shapes are represented by point clouds, we train a point cloud encoder following IM-NET [CZ19] to extract shape feature z for each fragment $o_i \in \mathcal{I}$. A corpus $\mathcal{D}_z = \left\{ z_i^{\text{pre}} \to \{ z_{i,j}^{\text{post}} \} \right\}$ is subsequently obtained by recording the fragment features before and after each fragmentation.

Inducing a grammar directly from $\mathcal{D}_z$ would lead to an overly complex grammar by treating most fragments as unique instances, resulting in poor generalizability. Rather, we cluster all features $\{z\}$ into k fragment types $\{c\}$ and learn a grammar from this new corpus:

$$\begin{aligned} \mathcal{D}_c^k &= \left\{ f(z_i^{\text{pre}}) \rightarrow \{f(z_{i,j}^{\text{post}})\} \;\middle|\; z_i^{\text{pre}} \rightarrow \{z_{i,j}^{\text{post}}\} \in \mathcal{D}_z \right\} \\ &= \left\{ c_i^{\text{pre}} \rightarrow \{c_{i,j}^{\text{post}}\} \right\}. \end{aligned} \tag{4.1}$$

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

Grammar induction Given corpus $\mathcal{D}_c^k$, we use maximum a posteriori (MAP) estimation to learn an optimal grammar,

$$\begin{aligned} \mathcal{G}^* &= \operatorname*{argmax}_{\mathcal{G}^k} \; p(\mathcal{D}_c^k|\mathcal{G}^k)p(\mathcal{G}^k) \\ &= \operatorname*{argmax}_{\mathcal{G}^k} \underbrace{\prod_{(\alpha_i \rightarrow \beta_i) \in \mathcal{D}_c^k} p(\alpha_i \rightarrow \beta_i|\mathcal{G}^k)}_{\text{data likelihood}} \cdot \underbrace{e^{\gamma|\mathcal{G}^k|}}_{\text{model prior}}, \end{aligned} \tag{4.2}$$

where $\alpha_i \rightarrow \beta_i$ is the i-th production rule in $\mathcal{D}_c^k$, γ a scalar coefficient, $|\mathcal{G}^k|$ the model size, and $p(\alpha_i \rightarrow \beta_i|\mathcal{G})$ the branching probability of the production rule $\alpha_i \rightarrow \beta_i$ defined in $\mathbb{P}$. The production rule probability is computed via maximum likelihood estimation and aligns with the frequency of each alternative choice [ZM07]:

$$p(\alpha \rightarrow \beta_i) = \#(\alpha \rightarrow \beta_i) / \sum_{j=1}^{n(\alpha)} \#(\alpha \rightarrow \beta_j), \tag{4.3}$$

where $\#(\alpha \rightarrow \beta)$ is the number of the production $\alpha \rightarrow \beta$ is observed in demonstrations, and $n(\alpha)$ the number of production rules whose left-side (the non-terminals) is α.

We first adopt an iterative non-parametric clustering approach, similar to DP-means [KJ12], to solve for $\mathcal{G}^*$ in Eq. (4.2) by alternating two steps: search for a better k, and estimate the best production rules. Next, we add a start variable v_S to the non-terminal set V_{NT} with production rules $v_S \rightarrow V_{NT}|V_T|v_S v_S$ so that the grammar can derive all possible variables from the start variable. We also fit a classifier on the clustered fragments to model the distribution of $p(c|z)$, the probability of a fragment's type c given its shape feature z, for ease of downstream tasks.

4.4 A Probabilistic Model of Fluent Change

We define the posterior probability of a parse tree pt given a fragment configuration $\mathcal{I}^g$ (*e.g.*, a goal or an observation) and a grammar $\mathcal{G}$:

$$p(pt \mid \mathcal{I}^g, \mathcal{G}) \propto \underbrace{p(\mathcal{I}^g \mid pt, \mathcal{G})}_{\substack{\text{observation}\\ \text{likelihood}}} \underbrace{p(pt \mid \mathcal{G})}_{\substack{\text{grammar}\\ \text{prior}}}, \tag{4.4}$$

where the first term is the likelihood of observing $\mathcal{I}^g$ given pt, and the second term is a prior probability of obtaining the parse tree pt given $\mathcal{G}$. The overall posterior probability measures the alignment between pt and $\mathcal{I}^g$ according to $\mathcal{G}$.

Grammar prior The grammar prior estimates pt based on the learned production rules and branching probability:

$$p(pt \mid \mathcal{G}) = p(R^{pt} \mid \mathcal{G}) = \prod_{(\alpha_i \rightarrow \beta_i) \in R^{pt}} p(\alpha_i \rightarrow \beta_i \mid \mathcal{G}), \tag{4.5}$$

where R^{pt} represents the set of production rules contained in the parse tree pt, and $p(\alpha_i \rightarrow \beta_i \mid \mathcal{G})$ is the conditional probability of choosing the production rule $\alpha_i \rightarrow \beta_i$ given that the non-terminal node being expanded is α_i.

Observation likelihood Akin to the perception of texton [Jul84, ZGW05] and texture [Jul81, ZWM98], human perception of object fragment also falls into a continuous spectrum. Here, we measure the observation likelihood at the two ends of this continuum. At the individual level, the likelihood computes how well terminal nodes of pt match fragments in $\mathcal{I}^g$ via a one-to-one mapping, which is useful for robot planning and the reconstruction of the fragmentation sequence retrospectively. At the ensemble level, the likelihood purses the statistical difference between the distribution of fragment types in terminal nodes of pt and that of $\mathcal{I}^g$, which is useful for transferring knowledge to a similar task (*e.g.*, cutting a potato given the observation of carrot fragments). Computationally, we extract shape feature z and pose q for each fragment in $\mathcal{I}^g$ and obtain $\mathcal{I}^g_Z = \{z_i\}$ and $\mathcal{I}^g_Q = \{q_i\}$ (i refers to the i-th fragment). Below, we further detail the formulation of these likelihoods.

4.4.1 Observation likelihood at the individual level

To measure the observation likelihood at the individual level, each fragment in $\mathcal{I}^g$ is matched with a terminal node in pt. The observation likelihood can be formulated as:

$$p_{\text{idv}}(\mathcal{I}^g \mid pt, \mathcal{G}) = \underbrace{p_{\text{idv}}(\mathcal{I}^g_Z \mid pt)}_{\substack{\text{individual}\\ \text{shape matching}}} \underbrace{p_{\text{idv}}(\mathcal{I}^g_Q \mid pt)}_{\substack{\text{layout}\\ \text{grouping}}}. \tag{4.6}$$

Individual shape matching The individual shape matching term evaluates how well the fragment types in terminal nodes of pt (*i.e.*, the fluent) match the features of corresponding fragments in $\mathcal{I}^g_Z$:

$$\begin{aligned} p_{\text{idv}}(\mathcal{I}^g_Z \mid pt) &= p(\mathcal{I}^g_Z \mid \langle v^i_T \rangle) = p(\langle z_i \rangle \mid \langle c_i \rangle) \\ &= \prod_{i=1}^{N} p(z_i \mid c_i) \propto \prod_{i=1}^{N} p(c_i \mid z_i) p(z_i), \end{aligned} \tag{4.7}$$

where $\langle \cdot \rangle$ represents an ordered sequence, v^i_T refers to the i-th terminal node in the parse tree, c_i the fragment type denoted by v^i_T, z_i the shape feature extracted from the corresponding fragment, and N the number of fragments in $\mathcal{I}^g_Z$. We assume the prior probability $p(z_i)$ is a normal distribution fitted on the train set. The value of $p(c_i|z_i)$ is obtained from the classifier given the shape feature z_i.

Layout grouping The layout grouping term measures how likely the production rules in the given pt assemble the layout of fragments—the relative poses between fragments:

$$
\begin{aligned}
p_{\text{idv}}(\mathcal{I}_Q^g \mid pt) &= p(\mathcal{I}_Q^g \mid R^{pt}) \\
&= \prod_{(\alpha_i \to \beta_i) \in R^{pt}} p\left(\beta_i \mid \alpha_i \to \beta_i\right) \\
&= \prod_{(\alpha_i \to \beta_i) \in R^{pt}} \prod_{v_j^{\beta_i} \in \beta_i} p\left(v_j^{\beta_i} \mid \alpha_i \to \beta_i\right),
\end{aligned}
\tag{4.8}
$$

where R^{pt} represents the set of production rules contained in the parse tree pt. $\alpha_i \to \beta_i$ is the i-th production rule in R^{pt}, where 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 of the production rule $\alpha_i \to \beta_i$ produces the node $v_j^{\beta_i}$.

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

$$
p\left(v_j^{\beta_i} \mid \alpha_i \to \beta_i\right) = \frac{1}{Z} \exp\left(-\text{dist}(q^{\alpha_i}, q_j^{\beta_i})\right), \tag{4.9}
$$

where Z is the partition function, $q_j^{\beta_i}$ the averaged pose of objects 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.

4.4.2 Observation likelihood at the ensemble level

Fragment ensemble Different from treating fragments as individuals akin to texton modeling [ZGW05], another perspective of the observation likelihood is to consider all fragments as an ensemble akin to texture modeling [ZWM98]. Specifically, we compute the statistical difference of fragment types between the fluent in pt and the observed fragment ensemble.

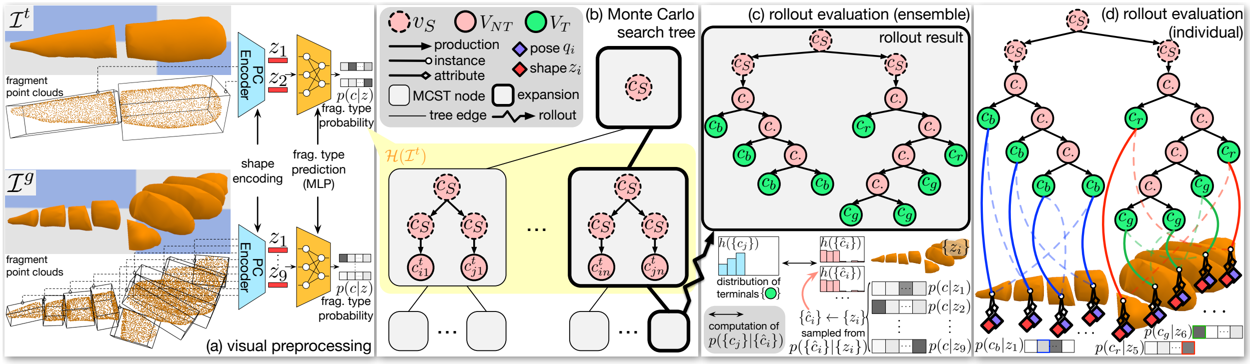


Figure 4.2: **An illustration of the inference process of the optimal parse tree pt^* through MCTS.** (a) Given fragment point clouds in an observation, the shape feature is extracted from each fragment via a pre-trained point cloud encoder, and the probability of fragment types $p(c|z)$ is estimated via an MLP. (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 on the parse tree of that node. The yellow region $\mathcal{H}(\mathcal{I}^t)$ is a set of search nodes whose states (*i.e.*, parse trees) are sampled from each fragment in $\mathcal{I}^t$ according to $p(c|z)$. (c) We evaluate the rollout at the ensemble level by measuring the statistical difference of fragment types between the parse tree and observed fragments. (d) We evaluate the rollout at the individual level by assigning each terminal node with a specific fragment in $\mathcal{I}^g$. The dotted lines represent an optimal assignment that maximizes the individual shape matching likelihood in Eq. (4.7) and are further refined to solid lines that maximize the layout grouping likelihood in Eq. (4.8) while the optimality of Eq. (4.7) is preserved.

Formally, the observation likelihood in Eq. (4.4) could be formulated as:

$$\begin{aligned} p_{\text{esm}}(\mathcal{I}^g|pt,\mathcal{G}) = p_{\text{esm}}(\mathcal{I}^g_Z|pt) &= p(\{z_i\}|\{c_j\}) \\ &\propto p(\{c_j\}|\{z_i\})p(\{z_i\}) \\ &= \mathbb{E}_{p(\{\hat{c}_i\}|\{z_i\})}\Big[p(\{c_j\}|\{\hat{c}_i\})\Big]\ p(\{z_i\}), \end{aligned} \tag{4.10}$$

where $\{c_j\}$ is the set of fragment types in the terminal nodes of pt (*i.e.*, the fluent).

The number of fragments in $\{z_i\}$ and $\{c_j\}$ are usually not necessarily the same, and it is infeasible to directly estimate $p(\{c_j\}|\{z_i\})$. Hence, we introduce a potential fluent $\{\hat{c}_i\}$, where each $\hat{c}_i$ corresponds to an observed fragment z_i, and the resulting expectation term evaluates how well the potential fluent $\{\hat{c}_i\}$ aligns with the fluent $\{c_j\}$ in pt. We formulate the alignment between two fragment ensembles $p(\{c_j\}|\{\hat{c}_i\})$ based on the statistical difference between $\{c_j\}$ and $\{\hat{c}_i\}$:

$$p(\{c_j\}|\{\hat{c}_i\}) = \frac{1}{Z} \exp\Big(-D_{KL}\big(h(\{c_j\})\big|\big|h(\{\hat{c}_i\})\big)\Big), \tag{4.11}$$

where Z is the partition function, $h(\cdot)$ the distribution of fragment types, and $D_{KL}(\cdot)$ the Kullback–Leibler divergence that measures the difference between $h(\{c_j\})$ and $h(\{\hat{c}_i\})$.

4.5 Inference of Optimal Parse Tree

The learned fluent space describes the recursive and compositional nature of object fragmentation, which affords to recognize the fluent of object fragments and plan for actions that change the object to a desired fluent or reason about a fragmentation event in retrospect. Inference in the fluent space is a parsing process that finds the optimal parse tree pt^* best aligned with a goal configuration $\mathcal{I}^g$. When a known fragment configuration $\mathcal{I}^t$ is observed (common in robot planning tasks), we generate pt^* from $\mathcal{I}^t$ to $\mathcal{I}^g$; otherwise, we generate pt^* from the start variable.

Instead of merely classifying the observed fluent, either a reasoning or planning task would further require a joint inference of fluent from $\mathcal{I}^g$ and feasible transitions. We formulate this process as an MAP estimate:

$$pt^* = \underset{pt \in \mathcal{H}(\mathcal{I}^t)}{\operatorname{argmax}} \; p(\mathcal{I}^g \mid pt, \mathcal{G}) \; p(pt \mid \mathcal{G}), \tag{4.12}$$

where $\mathcal{H}(\mathcal{I}^t)$ is a set of parse trees, whose expansions from the start variable are sequentially sampled from each fragment in $\mathcal{I}^t$ according to $p(c|z)$.

Since the computation of pt^* in Eq. (4.12) is intractable, we infer the approximately optimal parse tree via Monte Carlo Tree Search (MCTS) as shown in Fig. 4.2. 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 on the parse tree of the search node, and the rollout results in each round are evaluated by measuring the likelihood described by the objective function in Eq. (4.12). During the back-propagation step, we use the likelihood 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^*.

By substituting different observation likelihood formulations (Eqs. (4.6) and (4.10)) into Eq. (4.12), we can infer at either the individual or the ensemble level, to be detailed below.

4.5.1 Inference at the individual level

Since the rollout result (see Fig. 4.2d for examples) represents a top-down derivation from the start variable, the terminal nodes have not been grounded to fragments in $\mathcal{I}^g$. Hence, for the i-th round of rollout, we need to compute an optimal assignment function $f_i^* : V_T \rightarrow O$ that grounds each terminal node v_T in pt_i to an unique fragment o in $\mathcal{I}^g$, such that the resulting parse tree $pt_i^{f^*}$ maximizes the likelihood in Eq. (4.6):

$$f^* = \underset{f}{\operatorname{argmax}} \; p_{idv}\left(\mathcal{I}_Q^g \;\middle|\; pt_i^f\right) p_{idv}\left(\mathcal{I}_Z^g \;\middle|\; pt_i^f\right), \tag{4.13}$$

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 direct 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 individual shape matching likelihood $p_{\text{idv}}(\mathcal{I}_Z^g|pt^f)$ in Eq. (4.7); see dotted lines in Fig. 4.2d. (ii) Refine f^{init} into f^* that maximizes the layout grouping likelihood $p_{\text{idv}}(\mathcal{I}_Q^g|pt^f)$ in Eq. (4.8) while conserving the optimality obtained in the previous step; see solid lines in

Fig. 4.2d.

The first step formulates a linear assignment problem:

$$
\begin{aligned}
f^{init} &= \underset{f}{\operatorname{argmax}} \prod_{j=1}^{N} p\left(c_{v_T^j} \middle| z_{f(v_T^j)}\right) p\left(z_{f(v_T^j)}\right) \\
&= \underset{f}{\operatorname{argmax}} \sum_{j=1}^{N} \log\left[p\left(c_{v_T^j} \middle| z_{f(v_T^j)}\right) p\left(z_{f(v_T^j)}\right)\right],
\end{aligned}
\tag{4.14}
$$

where N is the number of terminal nodes of the parse tree, v_T^j the j-th terminal node in the parse tree, c_v the fragment type of node v, and $z_{f(v)}$ the shape feature of the fragment that associated with node v according to the assignment function f.

The optimization problem in Eq. (4.14) could be rewrite as an integer linear program in a matrix form:

$$
\begin{aligned}
&\max_{A} \sum_{i,j} W_{ij} A_{ij} \\
\text{s.t.} \quad &\sum_{i} A_{ij} = 1,\ \forall i,\ \ \sum_{j} A_{ij} = 1,\ \forall j \\
&0 \leqslant A_{ij} \leqslant 1,\ \forall i, j \\
&A_{ij} \in \mathbb{Z},\ \forall i, j
\end{aligned}
\tag{4.15}
$$

where $\mathbb{Z}$ represents the set of integers, A is the assignment matrix, $A_{ij} = 1$ means assigning the j-th object to the i-th terminal node, and W is a weight matrix whose entry $W_{ij} = \log[p(c_{v_i}|z_j)p(z_j)]$ represents the probability of the j-th object matches the fragment type of the i-th terminal node. We adopt the Hungarian algorithm [Kuh55] to solve this program in a polynomial time.

Of note, the program in Eq. (4.15) assumes a **balanced assignment** problem, that is, the number of terminal nodes m equals to the number of fragments n (*i.e.* A is a square matrix and $m = n$). Otherwise, the constraints $\sum_i A_{ij} = 1,\ \forall i$ and $\sum_j A_{ij} = 1,\ \forall j$ cannot be satisfied. In practice, such an assumption does not always hold (*i.e.*, the number of fragments and the number of terminal nodes are not the same). However, such an **unbalanced assignment** can be reduced to a balanced assignment. In our implementation, we add $|m - n|$ new entities to the smaller part and set their weights to 0. Such that the least-matched entities in the larger part will be matched to the newly added entities with weights of 0. Then, the optimal assignment between the terminal nodes and the fragments is obtained from the assignment of non-zero weights (*i.e.* $A_{ij} = 1$ and $W_{ij} \neq 0$).

In the second step, f^{init} is further refined to have the parse tree aligned with the layout of the fragments while preserving the optimality obtained in Eq. (4.14). We adopt the simulated annealing algorithm [KGV83] to maximize $p_{\text{idv}}(\mathcal{I}_Q^g|pt^f)$. To preserve the optimality of Eq. (4.14) while the assignment f is optimized, the key is to ensure the fragment types of fragments remain the same after swapping the terminal nodes; see Fig. 4.2d. Hence, instead of randomly swapping all terminal nodes, only terminals whose matched fragments have the same fragment type would be considered candidates to be swapped.

4.5.2 Inference at the ensemble level

For ensemble-level inference, the observation likelihood in Eq. (4.12) is substituted with $p_{\text{esm}}(\mathcal{I}_Z^g|pt)$ in Eq. (4.10). The key is to compute $p_{\text{esm}}(\mathcal{I}_Z^g|pt)$ for evaluating the rollout results during the MCTS. Since computing the expectation term in $p_{\text{esm}}(\mathcal{I}_Z^g|pt)$ is intractable, we approximate it by Monte Carlo sampling. Specifically, we draw samples from $p(\{\hat{c}_i\}|\{z_i\})$ according to the classification probability $p(c|z)$ given fragment features in $\mathcal{I}_Z^g$ and use the drawn samples to estimate the expectation; see Fig. 4.2c.

4.6 Experiments

We develop an object-cutting simulator based on BulletPhysics [CB21] to collect fragmentation events and to validate the efficacy of the grammar-based representation for understanding such events in three experimental settings; see Section 4.6.1 for details of the simulation setup. First, we show that our algorithm can recover the fluent transitions of object fragmentation—how it transit to the current fragment configuration—through retrospective reasoning on the grammar. Next, we show that a robot plans a fragmentation sequence to achieve a desired fluent value—how to cut objects into a certain goal configuration—through forward reasoning. Third, in some far-transfer cases when reaching the exact goal is infeasible, our grammar produces a plan that approximates the observed effects by matching underlying statistics.

4.6.1 Data preparation

To collect fragmentation events, 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.

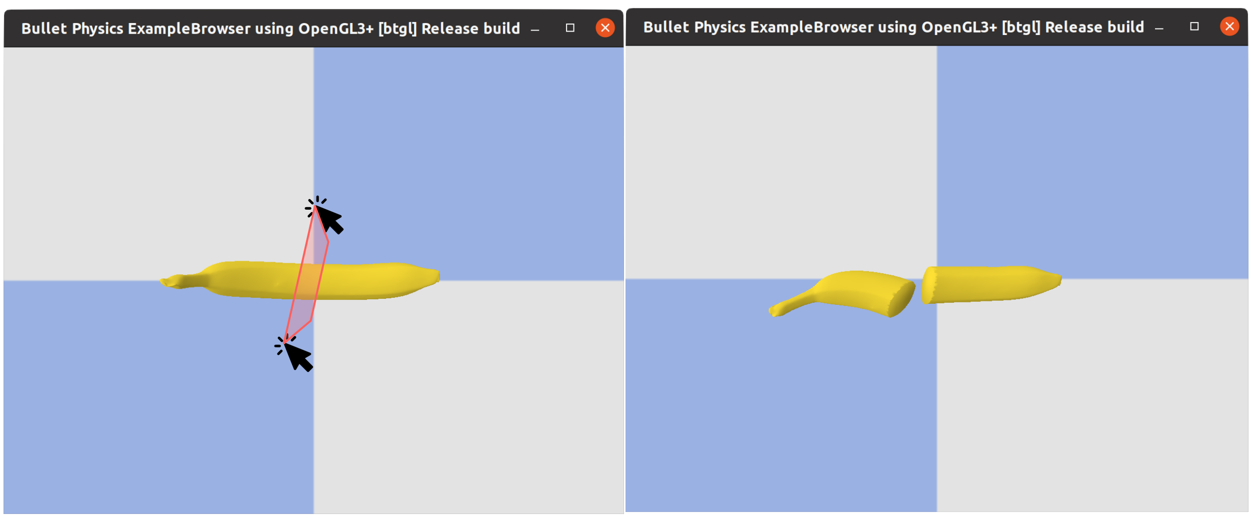


Figure 4.3: **An example of graphical user interface for object cutting.** The red translucent region indicates a 3D cutting plane determined by the two points clicked on the screen. The left figure shows the initial object configuration, whereas the right figure shows the fragment configuration after executing the cutting action.

Specifically, we develop a simulation environment based on the BulletPhysics engine [CB21], where we implement a cutting system that slice objects into pieces according to a given 3D cutting plane. Sliced objects preserve their dynamics (*i.e.*, velocity and acceleration) after being cut, and the collision and gravity system follows the original implementation of the BulletPhysics.

We design and implement an intuitive graphical user interface for collecting object cutting sequences from humans, where a 3D cutting plane is generated by two points clicked on the screen from an user, as shown in Fig. 4.3. More precisely, according to the projection matrix of synthetic rendering camera, each clicked point on the 2D screen is converted to a 3D ray that casts from the origin of camera to the clicked point in the 3D space of the simulator. These two 3D casting rays determine a 3D cutting plane, and all objects that intersect with the plane will be cut. In addition, users are able to change the angle of view in the simulator and conduct more flexible cutting actions.

Human subjects are asked 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. We recorded each fragmentation event as a sequence of fragment configurations and the corresponding cutting actions parameterized as 3D planes; the ground-truth 3D geometry of each fragment and its pose can be directly retrieved from the simulator. A total of 110 fragmentation events were collected and partitioned based on the initial number of objects N and the number of fragment categories in the goal configurations M; see Fig. 4.4 for some examples. We split the collected data, use a subset of $N = 1, M = 1$ as the train set, and test on the rest of events (*i.e.*, the rest of partition $N = 1, M = 1$ and partitions $N > 1, M > 1$).

4.6.2 Perceiving object fragments

Looking at a pile of fragments, humans can reconstruct the events retrospectively. Formally, given a current observation of object fragments $\mathcal{I}^t$, we ask the algorithm to infer their ancestors in $\mathcal{I}^{t-\Delta t}$. With the learned grammar, solving this task is to infer an optimal parse tree that reveals the fluent transitions between the two fragment configurations.

Fig. 4.5 depicts a qualitative result. The inference algorithm successfully identifies the

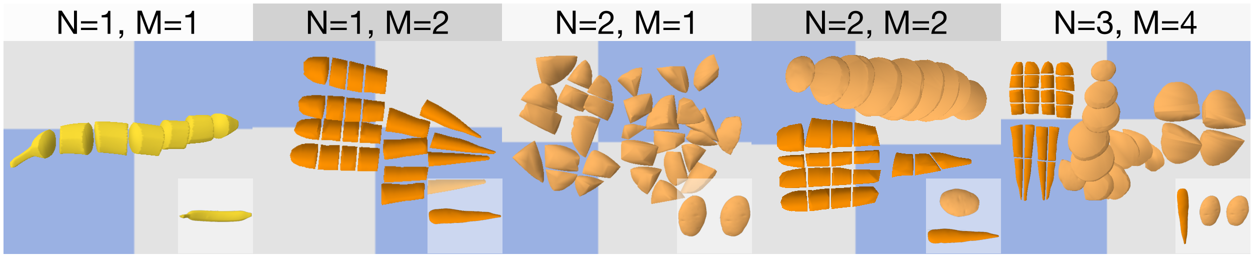


Figure 4.4: **Examples of collected data with different levels of task complexity.** N is the initial number of objects, and M the number of fragment categories in the goal configurations. The bottom right corner of each sub-figure shows the initial configuration.

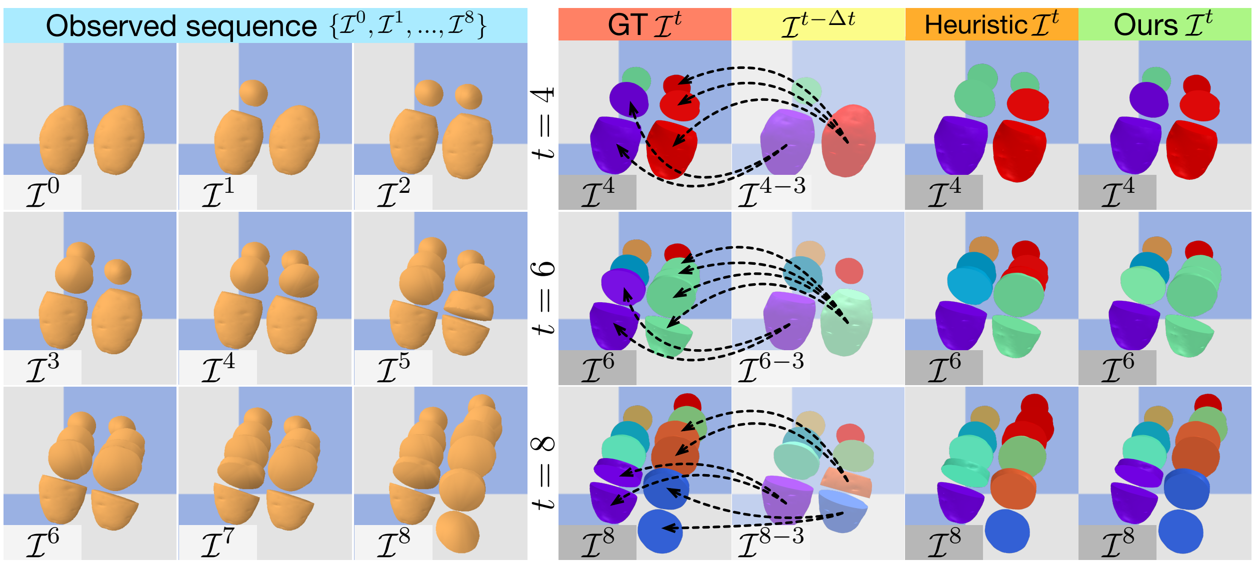


Figure 4.5: **Qualitative evaluation on inferring ancestors of fragments with interval $\mathbf{\Delta t} = \mathbf{3}$.** A fragment in $\mathcal{I}^t$ is shown in the same color as its ancestor from $\mathcal{I}^{t-\Delta t}$.

one-to-many fluent transitions and grounds each fragment in $\mathcal{I}^t$ to its ancestor in $\mathcal{I}^{t-3}$.

For comparison, we design a heuristics-based baseline due to the lack of existing baselines. Specifically, the baseline ground each fragment in $\mathcal{I}^t$ to the nearest fragment with a volume larger than it in $\mathcal{I}^{t-\Delta t}$.

Table 4.1 summarizes detailed quantitative evaluations with two ablation settings: without fragment shape matching term in Eq. (4.7) (*w/o Idv Frag*), and without the layout

Table 4.1: **Accuracy (mean and standard deviation) of fragment ancestor inference given two fragment configurations with different intervals.**

Model	$\Delta t = 1$	$\Delta t = 3$	$\Delta t = 5$
Heuristic	0.68±0.09	0.71±0.11	0.77±0.13
Ours (w/o Idv Frag)	0.71±0.08	0.63±0.14	0.69±0.19
Ours (w/o Layout)	0.80±0.18	0.78±0.14	0.79±0.19
Ours	**0.93**±0.08	**0.88**±0.10	**0.86**±0.11

grouping term in Eq. (4.8) (*w/o Layout*). Specifically, we compare the accuracy of identifying the association between fragments and their ancestors across different $\mathcal{I}$ for all four methods with Δt set to 1, 3, and 5 steps. For each fragmentation event, we repeat this evaluation multiple times for each $(\mathcal{I}^t, \mathcal{I}^{t-\Delta t})$ pair when $t \geqslant \Delta t$. Our method achieves the best performance in all cases, indicating that the complexity of object fragmentation could not be resolved solely by heuristics. Our ablation studies further show that an ideal solution must account for both individual fragment shapes and the layout of fragments.

4.6.3 Planning for exact goals

We further demonstrate the grammar's forward reasoning capability of producing a sequence of fragmentation; it is tasked to achieve a specific goal configuration based on the current object (fragment) configuration.

As the production rules in the learned grammar correspond to feasible fragmentation actions this task becomes a planning task, solved by inferring an optimal parse tree between the two fragment configurations $\mathcal{I}^t$ (current) and $\mathcal{I}^g$ (goal).

In our experiment, we have the models to cut one object at a time in the simulation. Each action requires selecting a specific object (fragment) and computing a 3D cutting

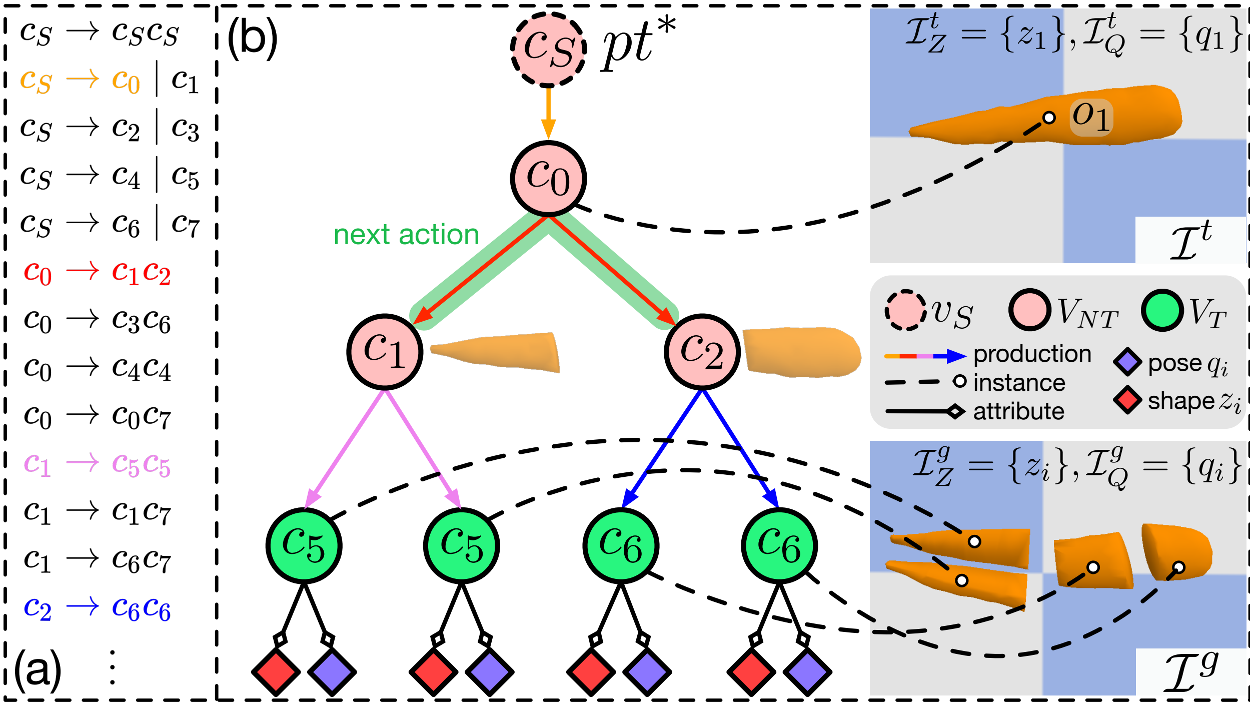


Figure 4.6: **Acquiring planned action(s) from inferred parse tree.** (a) lists the production rules of the learned grammar. (b) shows the optimal parse tree pt^* inferred at individual level. The next action (green region) is selected at the node that directly expanded from the start variable. The cutting plane $\boldsymbol{\pi}$ is acquired according to the distribution $p(\boldsymbol{\pi}|c_0 \to c_1c_2, z_1)$ given the production rule $c_0 \to c_1c_2$ and shape feature z_1 from fragment o_1.

plane $\boldsymbol{\pi}$ to cut the selected object. A cutting plane is represented as a homogeneous vector $\boldsymbol{\pi} = [\boldsymbol{n}^T, d]^T \in \mathbb{R}^4$ in the projective space with unit plane normal vector $\|\boldsymbol{n}\|_2 = 1$, and any point $\boldsymbol{v} \in \mathbb{R}^3$ on the plane that satisfies a constraint: $\boldsymbol{n}^T \cdot \boldsymbol{v} + d = 0$. As the distribution of $\boldsymbol{\pi}$ in the demonstration dataset is naturally multi-modal, we model $\boldsymbol{\pi}$ using a Gaussian Mixture Model [Rey09] with k components (we use $k = 4$):

$$p(\boldsymbol{\pi}|\cdot) = \sum_{i=1}^{k} w_i(\cdot)\mathcal{N}\big(\boldsymbol{\mu}_i(\cdot), \boldsymbol{\sigma}_i(\cdot)\big) \tag{4.16}$$

where $\cdot$ indicates potential given conditions.

Instead of directly predicting $\boldsymbol{\pi}$, all planning models regress parameters of the distribution including means $\boldsymbol{\mu}_i$, standard derivations $\boldsymbol{\sigma}_i$ and component weights $\boldsymbol{\omega}_i,\ i = 1, 2, ..., k$ using a two-layer Multi-layer Perceptron (MLP) as in the Mixture Density Network [Bis94]. Particularly, we fit the mixture model and estimate a cutting plane $\boldsymbol{\pi}$ relative to the canonical coordinate of the object to cut. Then cutting planes are sampled from the distribution for execution.

In our method, parameters of the mixture model are conditioned on the production rule $r : \alpha \rightarrow \beta$ and the shape feature z of the object (fragment) to be cut. Fig. 4.6 shows an example of how we acquire a planned action from an inferred optimal parse tree and sample a cutting plane from the distribution $p(\boldsymbol{\pi}|r, z)$.

We infer at the individual level (see Section 4.5.1) for planning for exact goals. Although this setup seems similar to that in the last experiment, where a single optimal parse tree is sufficient to identify the transitions among fragments in different time steps, planning requires the inference to happen iteratively due to the imperfect alignment between the optimal parse tree and the actual configuration and the discrepancy between the expected fluent transition and the resulted one after action execution. Therefore, after executing the action corresponding to the production rule at depth level one in the parse tree, we repeat the inference process until the number of fragments in the goal is reached.

Of note, this task involves an enormously large state and action space (see Section 4.2). Given the recent success of learning-to-plan methods [LCP21, MKS15, HLN20] in handling large spaces, we design two baselines: (i) **Behavioral Cloning (BC)** learns a goal-directed policy parameterized by a neural network to mimic human actions in collected demonstrations; (ii) **Offline Deep Q Network (QNet)**, a model-free reinforcement learning approach trained offline on logged demonstration data, where we use a neural network to approximate an action value function (Q function) of producing a certain fragment while regressing the action plane parameters. Further, we recruit (iii) **Human** participants to perform the planning tasks under the same setup, which serves as the performance upper bound.

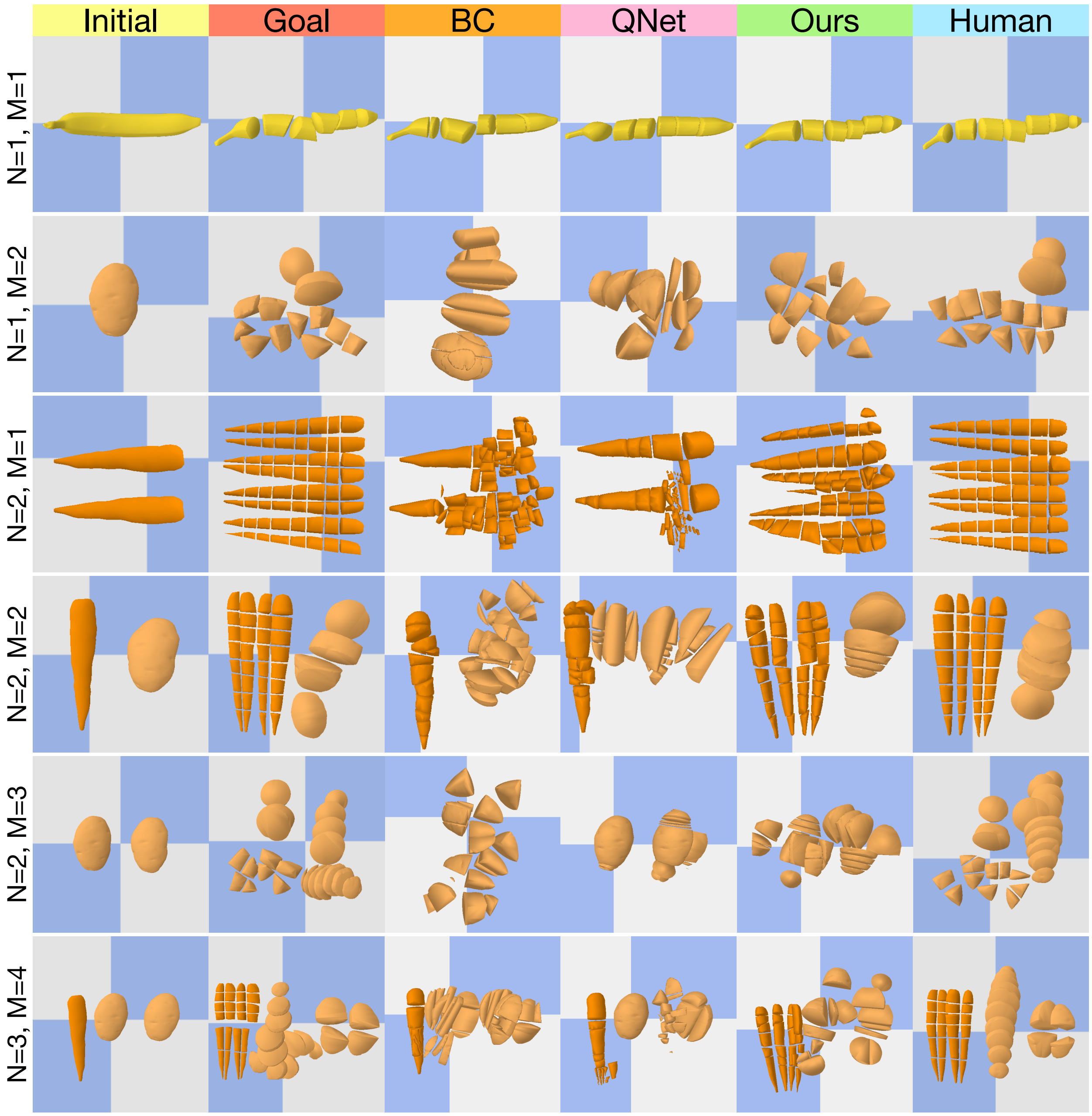


Figure 4.7: **Qualitative evaluation on planning for exact goals in object fragmentation tasks.** Each row presents a sample case of a certain combination of N and M.

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 IoU between the produced final fragments and the fragments in the goal configuration. More precisely, the

Mean best-matched IoU computes the averaged IoU of best-matched fragment pairs between two fragment configurations. (ii) **HR**. We recruit human participants to *subjectively* rate the fitness of the goal configurations. The rating ranges from 1 to 5 in discrete values; a higher score indicates a better match.

Table 4.2: **Quantitative evaluation on planning for exact goals in object fragmentation tasks.** We evaluate all methods using the best-matched IoU and HR on the test set with various N, M combinations, averaged across five runs; ± denotes standard deviation.

Method	N=1, M=1		N=1, M=2		N=2, M=1	
	IoU	HR	IoU	HR	IoU	HR
BC	0.37 ± 0.11	2.19±1.07	0.35 ± 0.08	1.76±0.87	0.44 ± 0.08	1.64±0.65
QNet	0.40 ± 0.16	2.14±1.21	0.32 ± 0.12	1.95±0.87	0.34 ± 0.16	1.19±0.39
Ours	**0.58** ± 0.08	**4.32**±0.77	**0.49** ± 0.06	**3.60**±1.02	**0.56** ± 0.03	**3.69**±0.89
Human	0.57 ± 0.03	4.48±0.96	0.62 ± 0.07	4.86±0.35	0.62 ± 0.09	4.83±0.37

Method	N=2,M=2		N=2,M=3		N=3,M=4	
	IoU	HR	IoU	HR	IoU	HR
BC	0.42 ± 0.03	2.07±0.86	0.38 ± 0.03	1.73±0.99	0.38 ± 0.04	1.57±0.62
QNet	0.29 ± 0.09	1.24±0.43	0.28 ± 0.09	1.52±0.92	0.22 ± 0.08	1.26±0.49
Ours	**0.52** ± 0.04	**3.74**±0.90	**0.52** ± 0.03	**3.21**±0.86	**0.52** ± 0.02	**3.21**±0.86
Human	0.56 ± 0.04	4.79±0.56	0.60 ± 0.04	4.81±0.55	0.56 ± 0.04	4.81±0.55

We compare the proposed method with the baselines and tabulate the results in Table 4.2; Fig. 4.7 further shows a qualitative comparison. While the testing setup ($N = 1, M = 1$) is similar to the training data, those in the other four columns require certain generalization capability by involving more objects $N > 1$ and/or a composition capability $M > 1$. Our method greatly outperforms the BC and QNet in all five setups, approaching human-level performance. These results indicate that planning for object fragmentation cannot only rely on pursuing the goal configuration as the goal-directed policy produced by BC. Associating an action and a value function fitted from data, as modeled by QNet, would also fail. Rather, it requires a proper understanding of the fluent space and the well-defined transitions within it to succeed in this complex planning task. This experiment sufficiently shows that the proposed grammar-based representation allows an effective abstraction of such a space and generalizes well to challenging settings due to its compositional nature.

4.6.4 Planning in fragment ensemble

We further evaluate the fluent space learned by the grammar in a more challenging planning setting—the desired goal configuration is infeasible to achieve from the given initial configuration (*e.g.*, cutting a potato based on the observation of carrot fragments). The inference scheme at the ensemble level (see Section 4.5.2) naturally applies to this challenging task.

As shown in Fig. 4.8, the BC performs poorly as mimicking human actions cannot address these far-transfer cases, whereas the QNet can perform slightly better as some fluent transitions still apply. In comparison, the proposed grammar-based representation enables a new planning at the fragment ensemble level–pursuing an approximate goal that shares the underlying statistics with the exact goal.

We argue that this planning objective is more similar to human's pre-attentive perception of fragments, demonstrated by the small difference in human ratings between the results generated by our method and by other human participants; see Table 4.3; two baseline methods receive much lower ratings.

Fig. 4.9 shows how the stopping threshold ϵ affects the planning results. Similar to Julesz Ensemble [Jul62], when the fragment ensemble likelihood $p_{\text{esm}}(\mathcal{I}^g|pt, \mathcal{G})$ is greater than ϵ, we assume the goal is reached as the fluent described in pt and fragments in $\mathcal{I}^g$ share adequate statistics in terms of fragment types. A larger ϵ prohibits the algorithm from finding a feasible solution that strictly matches the goal, whereas a tiny ϵ fails to capture the essence of the fragment configuration. In our experiment, we set $\log \epsilon = -0.8$.

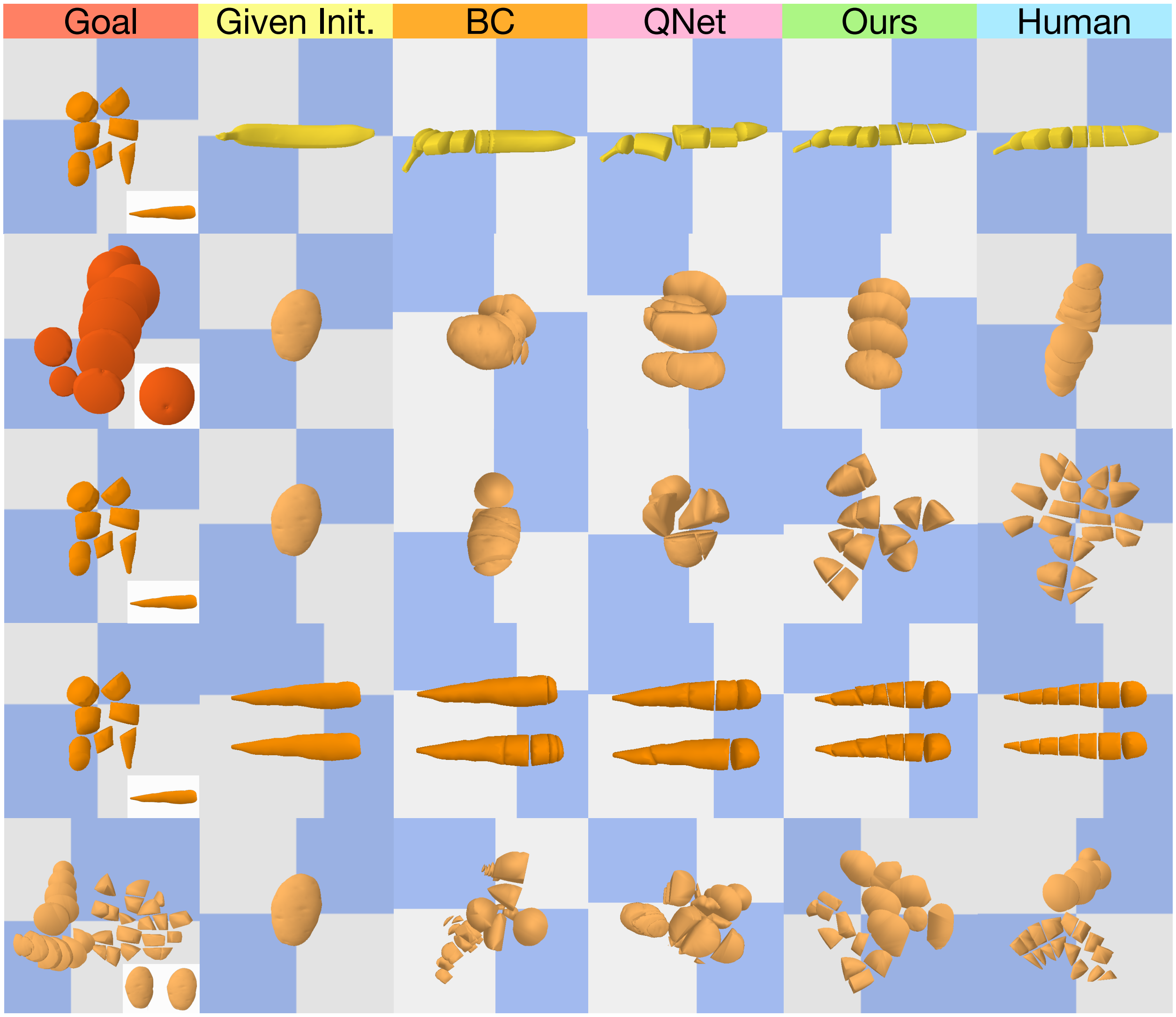


Figure 4.8: **Qualitative evaluation on planning with the ensemble goal; each row is a test case.** The bottom-right corner of the goal represents the initial configuration from which the goal is produced. In this experiment, while the goal configuration cannot be directly achieved from the initial configuration, our method produces configurations equivalent to the goal configuration at the ensemble level.

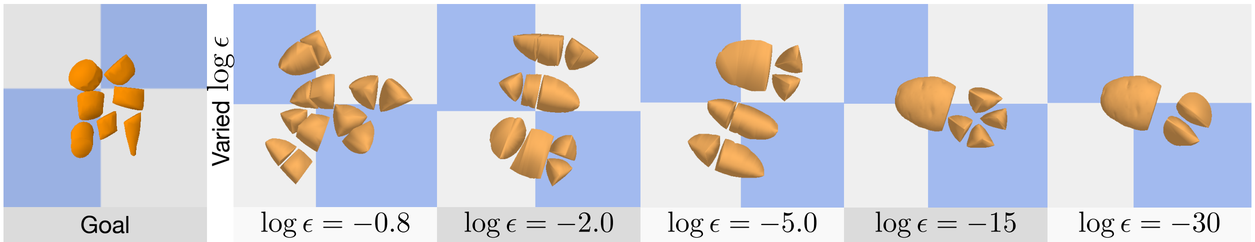


Figure 4.9: **Comparisons between the goal (left) and configurations produced by our method (right) with different stopping thresholds.**

Table 4.3: **Human evaluation of planning with ensemble goals.**

Model	BC	QNet	Ours	Human
HR	1.80±0.56	1.98±0.64	4.54±0.62	4.97±0.20

4.7 Conclusion

We presented a grammar-based representation for understanding object fragmentation events. A specific fragmentation was represented by a parse tree derived from the grammar, whose terminal nodes define the fluent of the fragment configuration, and the production rules indicate the plausible transitions within the fluent space. Given a current configuration of fragments, the grammar representation supports (i) a retrospective reasoning capability that identifies fragments' ancestors in a past configuration, and (ii) a forward reasoning scheme that plans a sequence of fragmentation to reach a goal configuration or to reach an ensemble configuration that matches the human pre-attentive perceptual experience of fragments when the goal is infeasible.

Collectively, this new perspective surpasses prior work that treats objects as a whole, introducing a new dimension for robots to perceive objects (fragments) and utilize fragmentation in complex tasks. We hope our work, as the initial effort, could shed light on future work on more complex object modeling, especially objects with topology changes.

CHAPTER 5

Understanding Physical Effects for Effective Tool-use

In this chapter, we study the interconnection between the geometry and topology fluent in a tool-use scenario. Particularly, we present a robot learning and planning framework that learns the essential physical properties contributing to the effects of a tool-use event (*e.g.*, how a hammer cracks a walnut) and produces an effective tool-use strategy with the least joint efforts. Leveraging a Finite Element Method (FEM)-based simulator that reproduces fine-grained, continuous visual and physical effects given observed tool-use events, the essential physical properties contributing to the effects are identified through the proposed Iterative Deepening Symbolic Regression (IDSR) algorithm. We further devise an optimal control-based motion planning scheme to integrate robot- and tool-specific kinematics and dynamics to produce an effective trajectory that enacts the learned properties. In simulation, we demonstrate that the proposed framework can produce more effective tool-use strategies, drastically different from the observed ones in two exemplar tasks. The materials in this chapter have been published in [ZJW22].

5.1 Introduction

A robot extends its capability to a broader range of tasks by using tools. Unlike treating a tool as a part of the end-effector that commonly appears in industrial settings [AA88, HLR19], researchers have proposed various learning-based approaches that empower more adept tool-use behaviors. However, existing learning objectives either focus on low-level motions [KOI21, SOF21] without an explicit understanding of the tasks or on higher-level

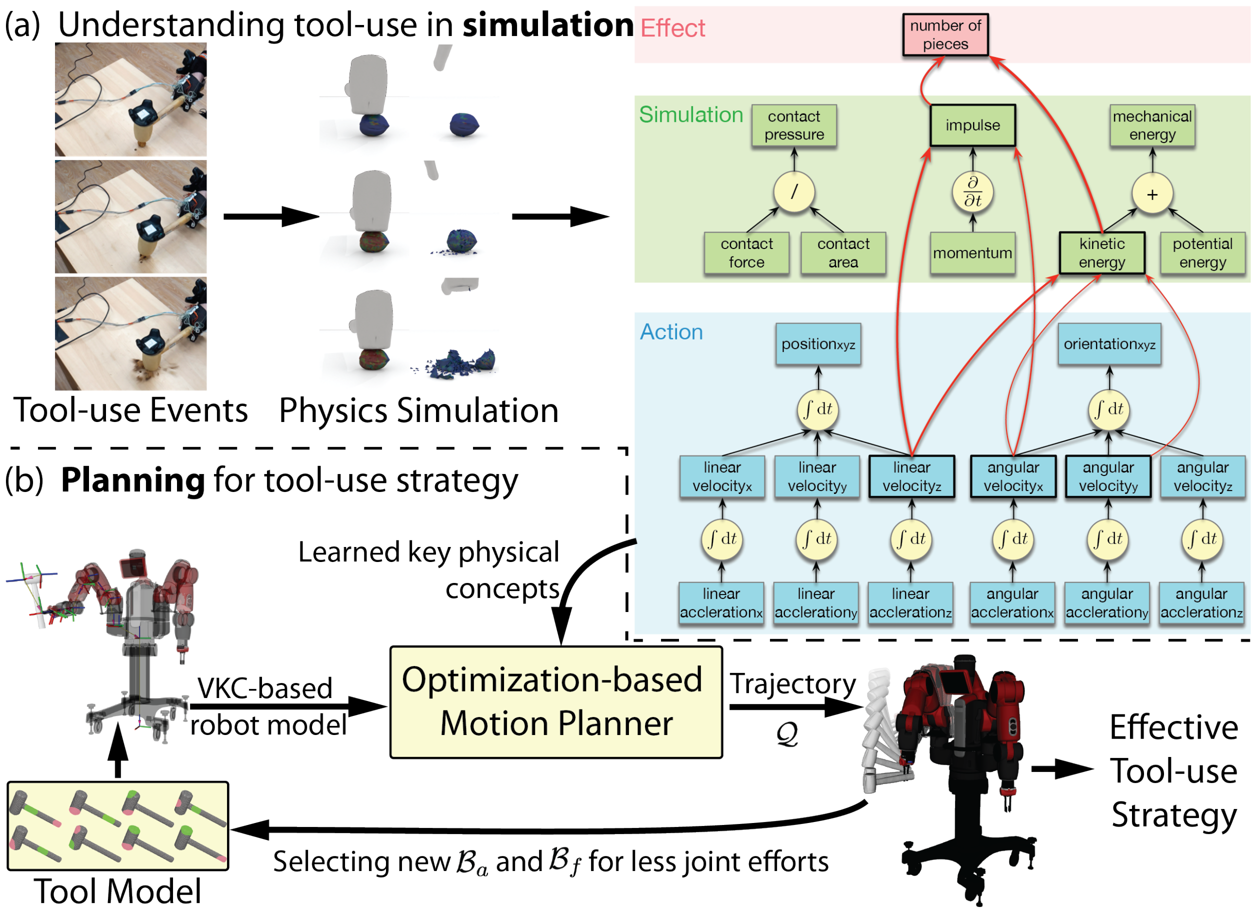


Figure 5.1: **Overview of the proposed framework.** (a) After observing tool-use events, we learn the essential physical properties involved in the processes from the effects reproduced by physics-based simulation. (b) The learned results are formulated into a motion planning scheme to produce various strategies to use an object, and the most effective strategy with minimal joint efforts among others is selected.

concepts with simplified motion patterns [AA18, QFZ20, TWT21]. As a result, robots are still far from producing situational tool-use strategies: Given a set of objects (typical tools or canonical objects), which one would be the best to accomplish the task? Once an object is chosen as the tool, how to efficiently use it given robot- and tool-specific kinematics and dynamics?

To tackle these challenges, we propose an integrated learning and planning framework wherein robots understand and produce effective tool-use strategies by reasoning about the essential physical properties that contribute to the success of the task. Fig. 5.1 shows an overview of our integrated framework. Compared to prior arts in robotics literature, our framework identifies the invariant learning objective of tool-uses at a more fundamental level; instead of using pure vision-based methods [LWZ17, TWL20], our framework focuses on the physical effects produced by the tool and learns to recognize the essential physical properties in accomplishing the task. Specifically, we adopt a state-of-the-art Finite Element Method (FEM) [LFS20] to simulate how both visual and physical effects evolve over time (*e.g.*, stress, energy, contact) in a continuous manner. A symbolic regression-based Iterative Deepening Symbolic Regression (IDSR) algorithm is devised to trace the set of physical properties produced by the simulator and to efficiently identify how much each property contributes to the effect.

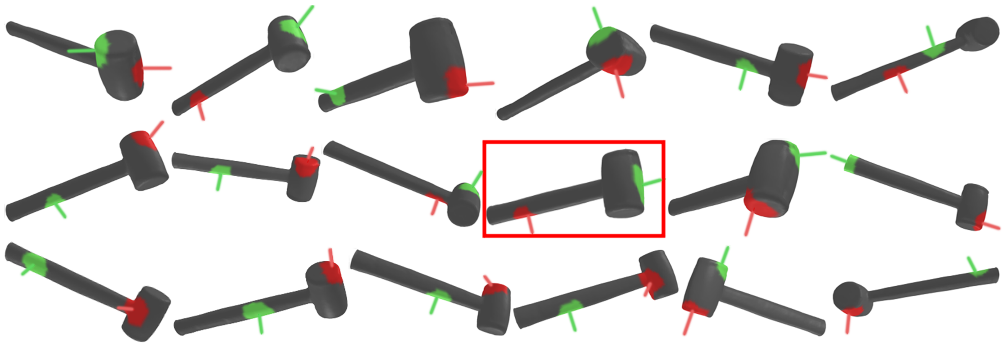
Next, we formulate the learned results into an optimal control-based motion planning scheme that allows the robot to generate various tool-use strategies whose efficiency is evaluated by joint efforts. To ease the motion planning problem and make the scheme more generic (*i.e.*, handle robots with different morphology, tools in diverse shapes, and various ways to operate tools), we introduce a VKC perspective [JZJ21, JZW21] that treats the tool as an additional link of the robot and integrates their kinematic and dynamic properties as a whole in motion planning.

In two exemplar tasks—cracking walnut and cutting carrot, we demonstrate that the proposed learning and planning framework can (i) identify the essential physical properties significant to the success of the task and (ii) produce an effective tool-use strategy that emulates the essential properties while minimizing joint efforts using seen and unseen objects as tools. As a result, the proposed framework allows the robot to better understand the physical environment by leveraging physics-based simulations and become more competent in bootstrapping novel (*i.e.*, not observed) tool-use strategies.

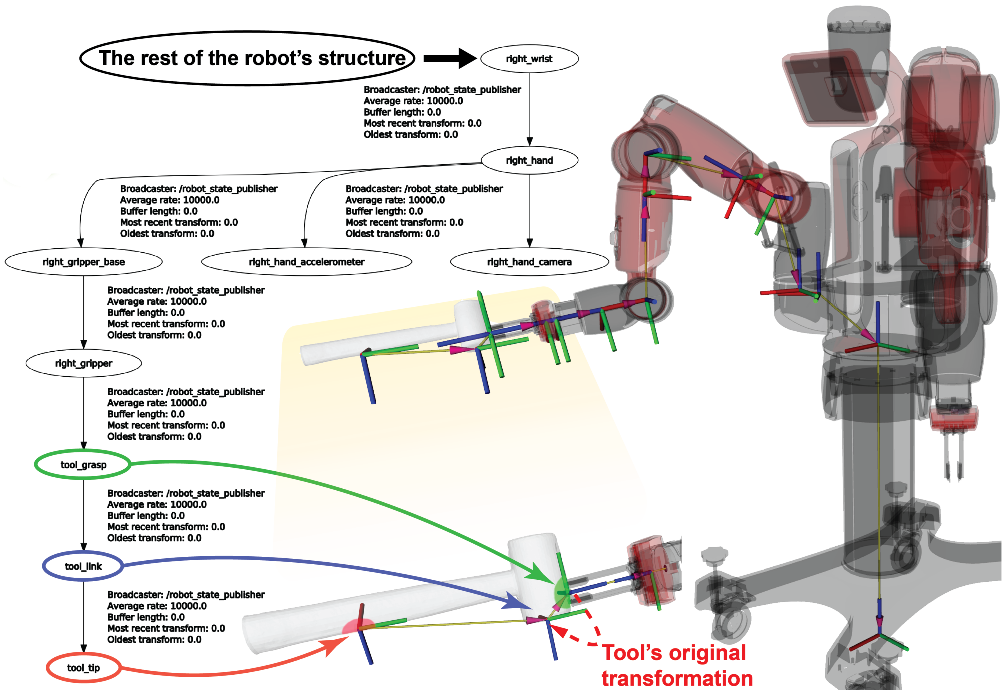
5.1.1 Related Work

Learning tool-use involves several cognitive and intelligent processes, challenging even for humans. Replicating such a skill set at the full spectrum is thus difficult, and existing literature mainly focuses at one of three different levels. Low-level **planning and control** methods track desired tool-use trajectories with impedance control [AA88], alter force and motion constraints at different stages [HLR19], or apply learning-based control [KOI21, SOF21]; robust execution is of the central interest. At mid-level, various **intermediate representations** are identified for better understanding tool-uses, such as keypoints [QFZ20, TWT21], primitive parts [NBC19, NSE19, WS15, WS20], and kinematic models [TKO17, JZW21]. Although introducing these representations facilitates learning more diverse tool-use skills, they are still restricted to the geometric association between shapes and task specifications. To capture high-level **concepts** embedded in tool-uses, researchers adopt task and motion planning [TAS18], functionality and affordance [ZZZ15, AA18, LC15], causality [BQS20], and commonsense [AST20, TBP21], achieving better generalization capabilities. Empowered by physics-based simulation, we advance this line of work by taking all three views into account: (i) learning related physical properties as the concepts from the tasks at the high-level, (ii) integrating tool's properties to robots by adopting VKC as the intermediate representation at the mid-level, and (iii) planning tool-use strategies via optimal control at the low-level.

Recently, physics-based **simulation** significantly facilitates various robotics tasks, *e.g.*, Liu *et al.* simulate forces to bridge human and robot's embodiments [LZZ19], Kennedy *et al.* plan liquid pouring [KST19], Matl *et al.* infer granular materials' properties [MNB20], Hahn *et al.* approximate soft objects' motions by estimating visco-elastic parameters [HBB19], Geilinger *et al.* develop simulation framework for rigid and soft bodies with fictional contact to promote robot locomotion [GHZ20], Li *et al.* improve UAV designs [LML21], and Heiden *et al.* optimize robot's cutting and slicing motions [HMN21]. Though sharing a similar spirit, the FEM simulator adopted in the paper [LFS20] is designed to produce a wider range of physical properties for robot learning instead of optimizing for dedicated applications.



(a) Various partitions of a hammer. The green denotes affordance bases $\mathcal{B}_a$, whereas the red denotes functional bases $\mathcal{B}_f$. Surface normals calculated at the regions' center are the directions to grasp.



(b) VKCs can be constructed by consolidating the kinematic and dynamics of the robot and tool.

Figure 5.2: **A VKC perspective that promotes motion planning.** (a) Given a sampled bases combination (highlighted in red box) of $\mathcal{B}_a$ and $\mathcal{B}_f$, (b) a VKC is constructed by assigning a virtual joint between the robot's gripper and the $\mathcal{B}_a$, and $\mathcal{B}_f$ becomes the new end-effector. This VKC conversion and construction supports efficient and optimal motion planning to produces proper tool-use trajectories by taking both kinematic and dynamic factors into account.

5.2 Problem Definition

We define a tool-use strategy $\mathcal{S} = (\mathcal{B}_a, \mathcal{B}_f, \mathcal{Q})$ by (i) an affordance basis $\mathcal{B}_a$ to be grasped by the robot gripper, (ii) a functional basis $\mathcal{B}_f$ to act on the target object, and (iii) a trajectory $\mathcal{Q}$ directing the functional basis to move towards the target object. Given a tool partitioned into a set of sub-meshes $\{\mathcal{M}_i\}$, a sampling process assigns one sub-mesh as $\mathcal{B}_a$ and another as $\mathcal{B}_f$, as illustrated in Fig. 5.2a. The surface normal vector $\boldsymbol{n}$ at the center of the corresponding sub-mesh indicates the direction for the robot's gripper to approach or for the tool to act on the target object. Assuming the robot can firmly grasp the tool at $\mathcal{B}_a$, generating a tool-use strategy $\mathcal{S}$ can be formulated as a motion planning problem that finds a collision-free trajectory $\mathcal{Q} = \boldsymbol{q}_{1:T}$ given $\mathcal{B}_a$ and $\mathcal{B}_f$.

5.2.1 VKC for Motion Planning

The theory of body schema [Gal06] suggests that humans can extend the body's representation to incorporate an external object and treat it as part of their limb for efficient motions and manipulations, which plays a significant role in tool-use [HS06]. This idea has been introduced to the robotics community to represent robot structures and guide robot's behaviors [HMA10]. Recent modeling approaches adopting VKC [JZW21, JZJ21] provide an effective means to model robot tool-uses: By inserting a virtual joint between robot end-effector and tool's $\mathcal{B}_a$, the kinematics and dynamics of the robot and the tool are integrated, and their motions are planned collectively, resulting in more coordinated motion and higher planning success rate [JZW21, JZJ21].

We first adopt an articulated body algorithm [Fea14] to compute the forward dynamics analytically for the constructed VKC. Next, the objective of the motion planning for robot tool-use is formulated by optimal control:

$$\min_{x,u,T} \int_0^T L(x(t), u(t))dt + \phi(x(T)) \tag{5.1}$$

$$L(x(t), u(t)) = \dot{q}^\top W_{\dot{q}} \dot{q} + u^\top W_u u, \tag{5.2}$$

$$\phi(x(T)) = T, \quad T \in \mathbb{R}^+, \tag{5.3}$$

where $W_{\dot{q}}$ and W_u are weight matrices for joint velocities and joint torques, $u : \mathbb{R} \to \mathbb{R}^n$ the control input consisted of joint torques, $\phi(x(T))$ measures the quality of the terminal state, particularly, we penalize the total elapsed time T. $x : \mathbb{R} \to \mathbb{R}^{2n+2m+1}$ is the state variable, which includes (i) joint positions q and velocities $\dot{q}$ of a manipulator with n DoF, (ii) q and $\dot{q}$ of underactuated joints in a tool, and (iii) the virtual joint at the grasp point with a total of m DoF. Eq. (5.1) penalizes the weighted quadratic cost on joint velocity and torques for the entire trajectory and the total elapsed time.

During the motion planning, we further impose several safety constraints:

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in [0, T] \tag{5.4}$$

$$g(x(t), u(t)) = 0, \quad t \in [0, T] \tag{5.5}$$

$$x_{lb} \leqslant x(t) \leqslant x_{ub}, \quad t \in [0, T] \tag{5.6}$$

$$u_{lb} \leqslant u(t) \leqslant u_{ub}, \quad t \in [0, T] \tag{5.7}$$

where Eq. (5.4) is the system dynamics, Eq. (5.5) is a task-dependent constraint for tool-use, Eq. (5.6) and Eq. (5.7) are safety constraints that bound the robot workspace and control limit.

5.2.2 Goal Specification

Formally, the goal for a tool-use is expressed as:

$$f_{\text{task}}(n_{\text{T}}(\mathcal{G}), \text{VKC}) \Rightarrow g(\cdot), \tag{5.8}$$

where $n_{\text{T}}(\mathcal{G})$ is a set of physical properties that are essential to the task, to be detailed in Section 5.3. f_{task} maps these physical properties and VKCs (as constructed in Fig. 5.2b) to a constraint function g for motion planning. The intuition is for the robot to emulate those essential physical properties in execution while considering the robot and tool's kinematics and dynamics.

To be more specific, let us take the walnut cracking task as an example. Given the goal position where the contact occurs p_g, the tool should act on the target object with a velocity vector $\mathbf{v}_{\text{tool}}$ and the tool's orientation $\mathbf{d}_{\text{tool}}$ (to be detailed in Section 5.3.4), both represented in world frame. Eq. (5.9) first finds a possible robot goal pose q_g through solving inverse kinematics to regulate the tool's orientation when contacting the target object:

$$\frac{f_z(q_g) \cdot \mathbf{v}_{\text{tool}}}{||f_z(q_g)|| \cdot ||\mathbf{v}_{\text{tool}}||} = \cos(\mathbf{d}_{\text{tool}}), \tag{5.9}$$

where $f_z : \mathbb{R}^n \rightarrow \mathbb{R}^3$ finds the surface normal of $\mathcal{B}_f$. Then, the goal joint velocities are computed by:

$$\dot{q}_g = J_{\text{VKC}}^{\top}(J_{\text{VKC}} J_{\text{VKC}}^{\top})^{-1} \mathbf{v}_{\text{tool}}, \tag{5.10}$$

where J_{VKC} is the geometric Jacobian from the robot's base frame to the tool's functional basis at the joint position q_g. Finally, Eq. (5.8) can be expressed in terms of joint velocity w.r.t. two constraint functions q_g and $\dot{q}_g$:

$$g_q(x(t), u(t)) = x_q(T) - q_g = 0 \tag{5.11}$$

$$g_{\dot{q}}(x(t), u(t)) = x_{\dot{q}}(T) - \dot{q}_g = 0 \tag{5.12}$$

5.3 Simulation and Learning

This section starts with the technical background of physics-based simulation, followed by how it reproduces fine-grained physical properties and helps understand tool-uses events.

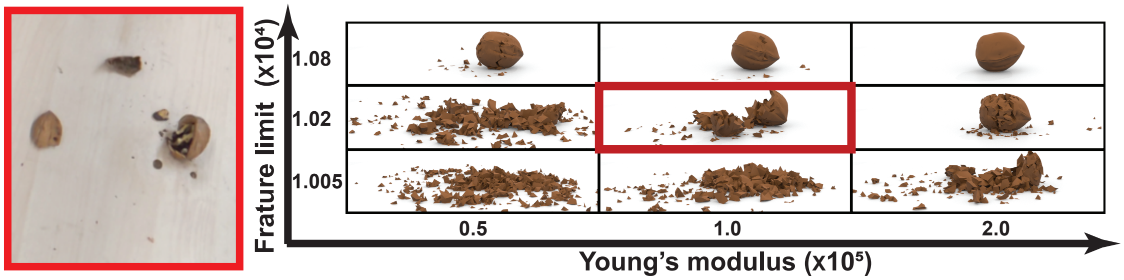
5.3.1 Background

Solid simulation approximates objects' physical status. It is oftentimes formulated with the Finite Element Method (FEM) [ZT00], which discretizes each object into small elements with a discrete set of sample points as the degree-of-freedoms. Mass and momentum conservation equations are discretized on the mesh and integrated overtime to capture the dynamics. This paper utilizes an Incremental Potential Contact (IPC) handling method [LGL19, LFS20, LKJ21], a state-of-the-art FEM-based simulator, to address the difficulty of simulating non-smooth contacts between a tool and a target object. To further support object fracture during tool-uses, our simulator measures the displacement of every pair of points that both connect to all the nodes of a triangle on the mesh. If the displacement relative to their original distance exceeds a certain strain threshold, we mark the triangle in-between as separated.

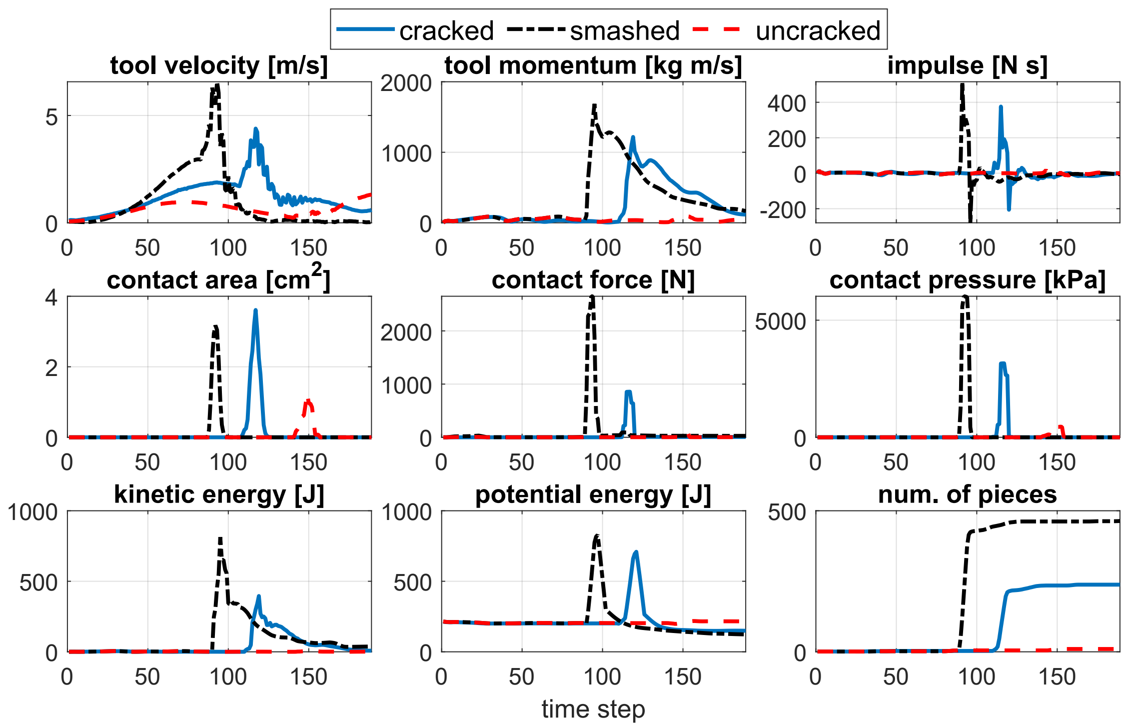
5.3.2 Reproducing Effect

To produce similar effects in the simulation that sufficiently match those in the physical world, some parameters governing an object's material property need to be appropriately set. In particular, Young's modulus reflects the object's stiffness—the stiffer the object is, the harder for it to deform or fracture, and fracture limit determines the number and the size of segments a large piece will fracture into. Fig. 5.3a qualitatively shows how the resulting effects vary given different Young's modulus and fracture limit. These two parameters are calibrated such that the simulated effects match the observation in physical world.

We use two VIVE Trackers to capture the tool-use events. One to track the movement of the tool (*e.g.*, a hammer), and another placed on the table serving as the reference point for the target object (*e.g.*, a walnut). Both VIVE trackers are calibrated such that their relative poses and captured trajectories are expressed in the same coordinate frame, with a time step of the inverse of their sampling frequency. The meshes of the target object and the tool are pre-scanned using an RGB-D camera. Combining scanned meshes and captured trajectories, we can fully reconstruct an observed event in both space and time and further simulate the effects of the target object both visually and physically. Examples of keyframes of the collected data with corresponding simulated results are shown in Fig. 5.1a. Fig. 5.3b visualizes the continuous numerical values of some notable physical properties obtained from



(a) Comparisons between the effect produced by experiments (left) and simulations (right) with different fracturing limits and Young's modulus.



(b) Fine-grained physical properties evolved in time reflect the different physical effects among uncracked, cracked, and smashed.

Figure 5.3: **Examples of qualitative and quantitative results produced by the FEM-based simulation.** (a) We qualitatively choose the parameters (in red) that best match the final effect of observed tool-use events. (b) By adopting an FEM-based simulator, the data collection process records physical properties evolved in time.

simulation. Of note, capturing how the object's status changes and its physical properties evolve over time is highly challenging, if not impossible, using visual information alone.

5.3.3 Learning Essential Physical Properties

We quantize the space of physical properties into three different levels; see Fig. 5.1a for an illustration. (i) **Action** (in blue) includes the trajectory data (position and orientation) directly observed in tool-use events and its velocity and acceleration calculated by finite-difference; these properties are usually controllable by robots. (ii) **Simulation** (in green) includes the physical properties estimated by the simulation given the observed Action. (iii) **Effect** (in red) includes the physical properties representing the tool-use effect. In the case of cracking and cutting tasks, we represent the effect by the number of pieces the target object transforms into.

Given various physical properties estimated and reproduced by the simulation, a robot has to learn how much these properties contribute to the success of the task and distill knowledge at all three levels, such that it can plan its motion in new and even unseen scenarios. To encode the connections across all three levels of physical properties, we propose to learn a **PRG** representation through symbolic regression [SL09, UT20]. Specifically, setting the Effect as the target variable y, the symbolic regression is tasked to find a valid expression of y using the set of given variables $\mathbf{x}$ in Simulation and Action: $y = f(\mathbf{x})$. To prevent over-fitting, we further balance the expression's complexity (*i.e.*, how many physical properties are involved) and accuracy (*i.e.*, how well it expresses the target variable). As such, the relations in PRG is sparse and only involve a small subset of the variables that succinctly express the target variable.

Typical symbolic regression problems oftentimes have a large search space. To tackle it, we devise an IDSR algorithm, a variant of symbolic regression, that utilizes the hierarchical information among physical properties at each level to prune the searching space. Specifically, as illustrated in Fig. 5.4a, typical symbolic regression algorithms directly explore the entire domain with all variables, whereas the proposed IDSR would iteratively deepen the domain based on the hierarchy among them. If one variable is not selected after an iteration, the domain will replace it with its child variables and reiterate the algorithm, and the resulting expression will only be updated if those child variables play a more significant role. This process continues until all the variables from one level in the domain are selected, or non-selected variables have no child. Algorithm 1 outlines the procedure.

In the case of cracking a walnut (see Fig. 5.4b), only after the set of relations between *Effect* and *Simulation* is explored would the algorithm subsequently identify the set of relations between *Simulation* and *Action*, expanding the PRG. As a result, this algorithm design saves the memory compared to conventional symbolic regression algorithms while preserving the full capability of distilling the essential relations among variables. The sub-graph highlighted in red in Fig. 5.1a shows the learned PRG of cracking a walnut, wherein the edge thicknesses are proportional to the physical properties' contribution to the effect. In another task of cutting a carrot by half using a knife (see Fig. 5.4c), the IDSR algorithm identifies the *contact area* governed by the *orientation* as an essential physical property, since the deviation from a proper orientation range may lead to the increment of contact area.

5.3.4 Reasoning about Goal Specification

The $\mathcal{G}$ identified by IDSR is still insufficient to support the proposed planning scheme because it only deduces the relation among those physical properties in a symbolic level, *i.e. velocity* for the task of cracking a walnut, and both *velocity* and *orientation* $\mathbf{d}_{\text{tool}}$ for cutting as shown in Fig. 5.4bc. The corresponding values of $\mathbf{v}_{\text{tool}}$ and/or $\mathbf{d}_{\text{tool}}$ applicable for robot planning is not determined yet.

To address this issue, we devise a sequential inference pipeline based on learned $\mathcal{G}$. As illustrated in Fig. 5.4d, by modeling the values of observed effect as a Gaussian distribution $P(E)$, a Gaussian Mixture Model (GMM) is learned to capture the joint probability between the effect and an identified physical property according to $\mathcal{G}$, *e.g.* $P(E, F)$ for effect and con-

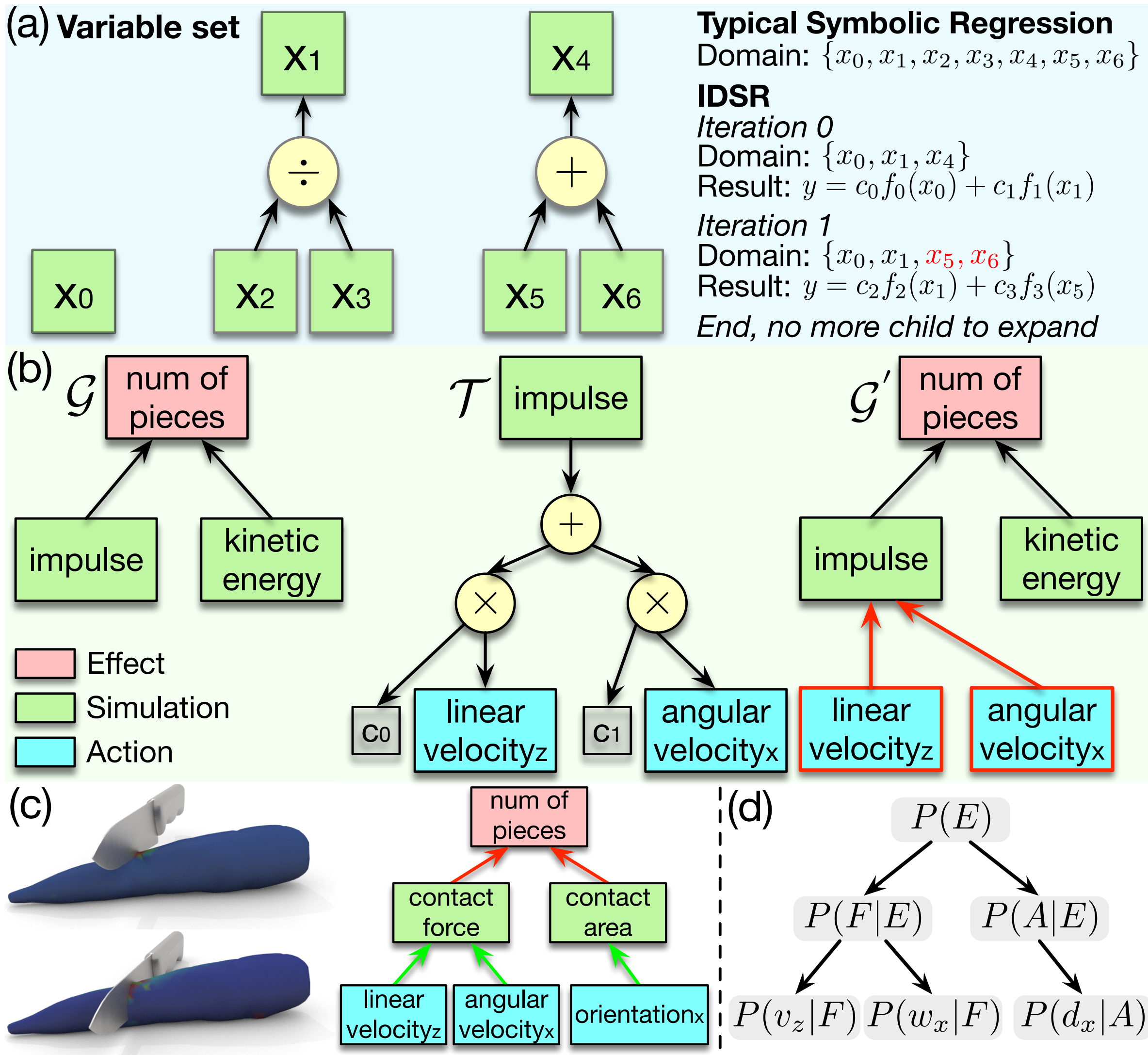


Figure 5.4: **Learning relations among physical properties using IDSR**. (a) An example of deepening the variable domain. Since x_4 is not included in the resulted expression in the iteration 0, it is thus removed, and its children are added to the domain in the next iteration. (b) An example of constructing PRG. $\mathcal{G}'$ isthe updated graph after inserting the expression $\mathcal{T}$ into the previous graph $\mathcal{G}$; newly added nodes and edges are highlighted in red. (c) The PRG constructed for the cutting task. (d) Inferring necessary values at the Action level for the goal specification in planning.

tact force in Fig. 5.4cd, using the EM algorithm [DLR77]. Specifically, the mixture models are fitted on the data obtained from the simulator that reproduces human demonstrations. Next, given a desired effect, inferring specific values of *contact force* is performed by drawing samples from the distribution $P(F|E) = P(E, F)/P(E)$ [BN06], and a velocity in z direction v_z is subsequently obtained by sampling from $P(v_z|F)$ following the same protocol. Eventually, this process produces the necessary values at the Action level ($\mathbf{v}_{\text{tool}}$ and $\mathbf{d}_{\text{tool}}$) as goal specifications for Eqs. (5.9) and (5.10).

Algorithm 1: IDSR

Data: Data samples: $\mathcal{D}$. Target variable: v_t. Variable set: $\mathcal{V}$

Result: Best matched expression tree: $\mathcal{T}$

```
Domain ← {AllRoots(𝒱)} while not terminate do
    terminate ← True;
    //Symbolic regression on Domain
    𝒯 ← SR(𝒟, v_t, Domain);
    diff ← Domain \ 𝒯.leaveSymbols();
    //Deepening the searching domain
    foreach v in diff do
        if v has child then
            Domain.add(v.children());
            Domain.remove(v);
            terminate ← False;
        end
    end
end
return 𝒯 //Return the latest 𝒯
```

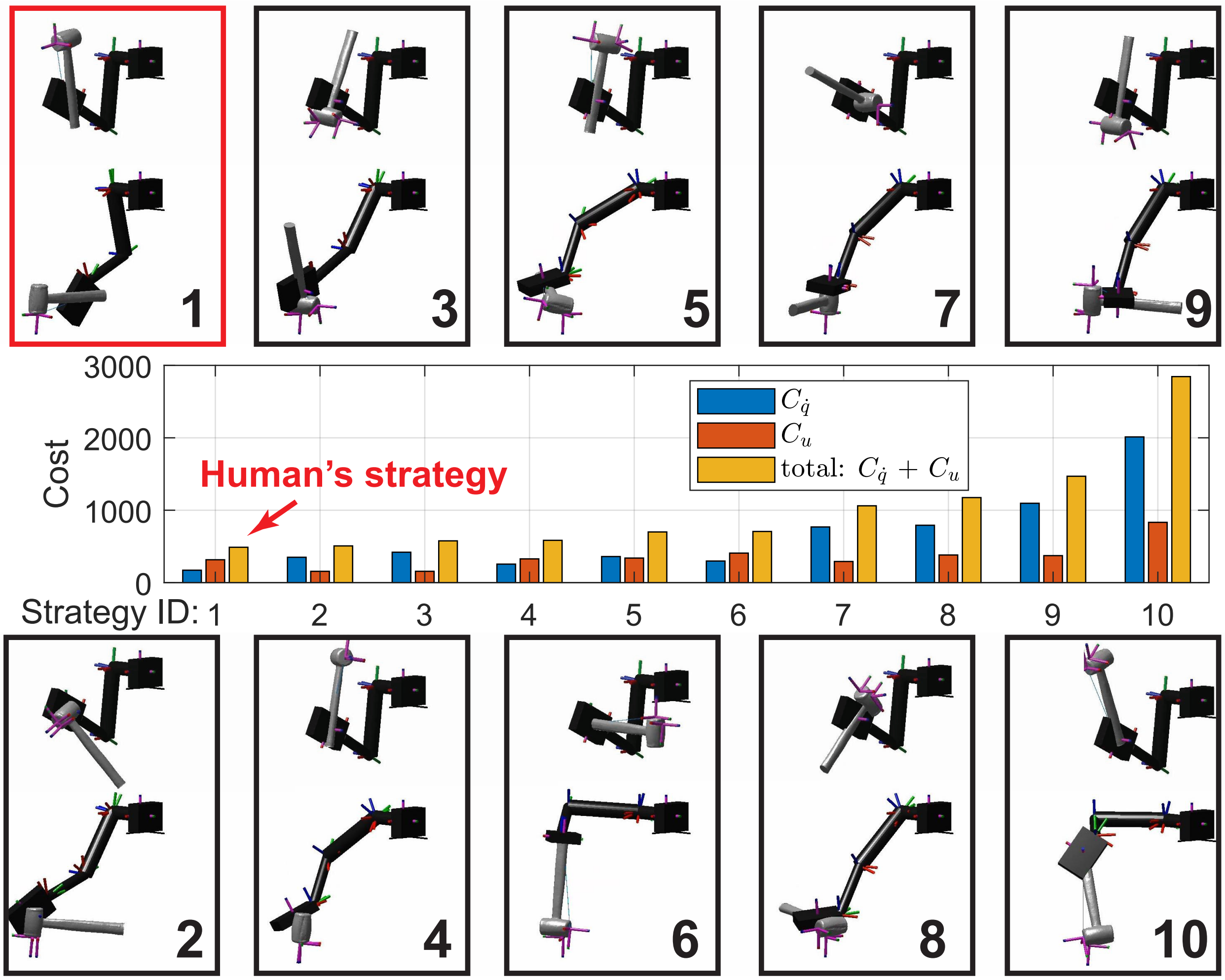


Figure 5.5: **Different strategies of tool-use using an approximated human arm model**. $\mathcal{B}_a$s and $\mathcal{B}_f$s are sampled from partitioned regions on the hammer, and the trajectory $\mathcal{Q}$ is produced by the optimal control using VKCs. The optimal strategy (in red) indeed follows human intuition of operating a hammer. $C_{\dot{q}}$ is the trajectory smoothness cost, and C_u is the joint torque effort cost.

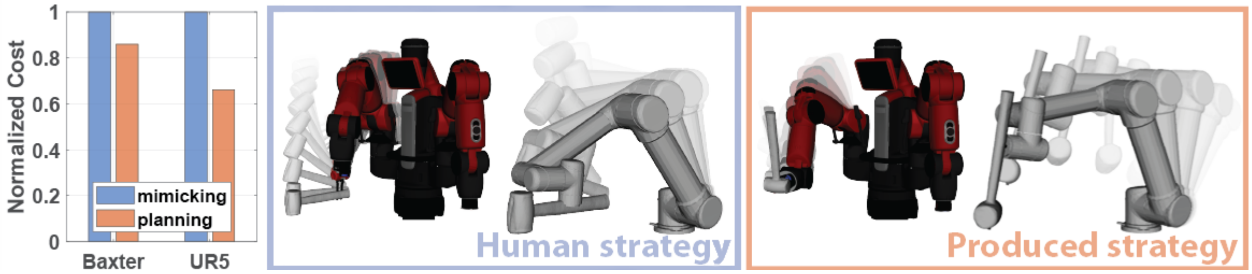
5.4 Experiments

We conduct three sets of studies regarding different types of manipulators with various settings. Using a human arm model [WHV05], we first validate that our planning scheme produces a feasible tool-use strategy identical to human choices; see Section 5.4.1. Next, we show that our proposed framework produces diverse tool-use strategies for Baxter arm and UR5 manipulator under different scenarios; the most effective ones in terms of least joint effort are demonstrated in Section 5.4.2. Finally, the produced strategies are fed to simulations for robot planning and execution; see Section 5.4.3. Experimental results verify that the framework indeed captures the essential physical properties, capable of converting these learned relations into goal specification, resulting in the success of motion planning and task completion.

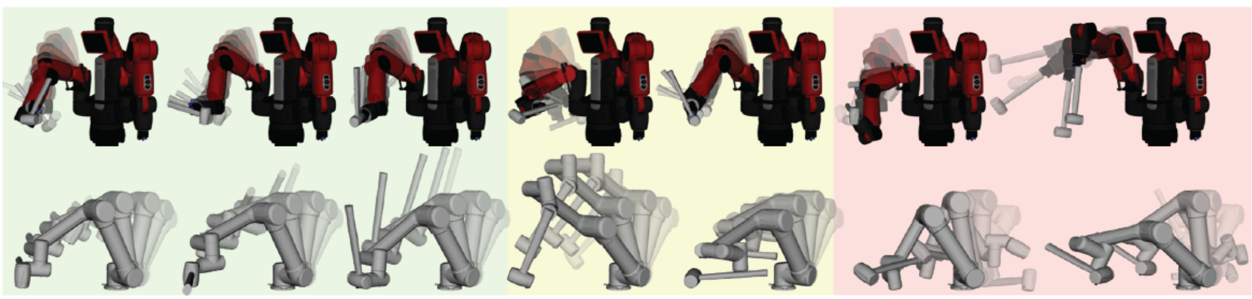
In all experiments, we solve the motion planning problem defined in Section 5.2 by CasADi [AGH19] with the OpenOCL [KLA17] support. A tool-use is considered invalid if the planner cannot produce a feasible solution. We assume the manipulators' bases are fixed. The tool structures are scanned by an RGB-D camera and reconstructed into watertight meshes, and the tool's material is homogeneous. For fair comparisons, the target object (*e.g.*, walnut) is placed at the same location within the operational space for each type of manipulator, and the initial pose of the manipulator is identical across all trials. In each trial, the target object has 1229 mesh vertices and is simulated for 200 time steps. The simulation runs on a 16-core AMD Ryzen 9 5950X machine and the average run time for one trial is 77.08 minutes with parallelization for the linear system computations [LFS20]. Algorithm-wise parallelization for FEM still remains an open problem.

5.4.1 Validating Optimal Planning by Task Efficiency

In this experiment, we evaluate whether the optimal control-based planning scheme is effective by comparing the produced tool-use strategies with that of human's rational choice, which should be regarded as near-optimal. Using the human arm model [WHV05] that consists of 7 DoFs (3 for shoulder, 2 for elbow, and 2 for wrist) with corresponding arm's physical properties (*i.e.*, mass, inertia) measured by human subjects, we sample various combinations of $\mathcal{B}_a$ and $\mathcal{B}_f$ and produce the corresponding tool-use trajectories. Fig. 5.5 shows initial and final arm postures and their computed costs of replicated human tool-uses



(a) Comparison of joint effort costs in hammering between mimicking human's strategy and the most effective one produced by our framework.



(b) Examples of various strategies to use the hammer.

Figure 5.6: **Generated various strategies in using a hammer**. (a) Given an inferred velocity vector acting on the walnut, the best tool-use strategy for each robot found by our framework is more efficient than simply mimicking human's strategy, indicated by lower cost. (b) Other strategies found by the proposed framework: low cost (in green), high cost (in yellow), and invalid with violation of constraints (in red).

and nine examples of alternative solutions.

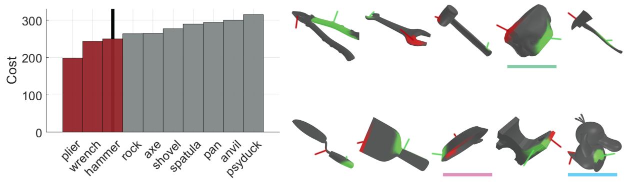
Our results show that Strategy 1 is the most efficient one. Compared with a conventional swinging action, holding hammerhead reduces the inertia compensated by actions, resulting in a lower joint torque effort costs C_us in Strategy 2 and 3. However, the trajectory smoothness costs $C_{\dot{q}}$s are higher as a larger acceleration is required to reach the goal velocity, making their total costs higher than the cost in Strategy 1. Since both Strategy 4 and 5 start from a similar $\mathcal{B}_a$ as in Strategy 1 followed by a swinging trajectory, their C_us are similar to that of Strategy 1; however, their $C_{\dot{q}}$s are higher since their $\mathcal{B}_f$s do not well aligned with arm postures. Strategy 6 to 10 are some less typical examples with high C_us and $C_{\dot{q}}$s; we seldom observe these strategies in real life. Together, these results indicate that our planning scheme can produce an efficient tool-use trajectory with underlying rationales akin to human tool-use behaviors, and thus we expect it to uncover similar insights into robot tool-uses.

5.4.2 Effective Tool-Uses

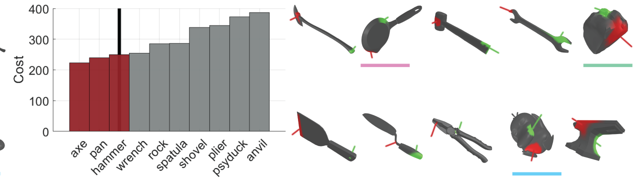
After validating our optimal control-based planning scheme, we test the efficacy of tool-use strategies derived from learned physical properties using two different robots (a Baxter robot and a UR5 manipulator) in two tasks (cracking and cutting).

Due to significant differences in kinematic structures, the observed human strategy of tool-uses may not be ideal for robots. In Fig. 5.6a, two robots first mimic human's strategy. Specifically, the robots select the observed human's $\mathcal{B}_a$ and $\mathcal{B}_f$ and mimic the observed trajectory $\mathcal{Q}$ by inverse kinematics to operate the hammer. The resulting costs are higher than those of the best strategy found by our framework; the ones found by the proposed framework are dramatically different but more effective for the robots. Fig. 5.6b further displays some other tool-use strategies with low-cost (effective), high-cost (ineffective), or are invalid by violating constraints.

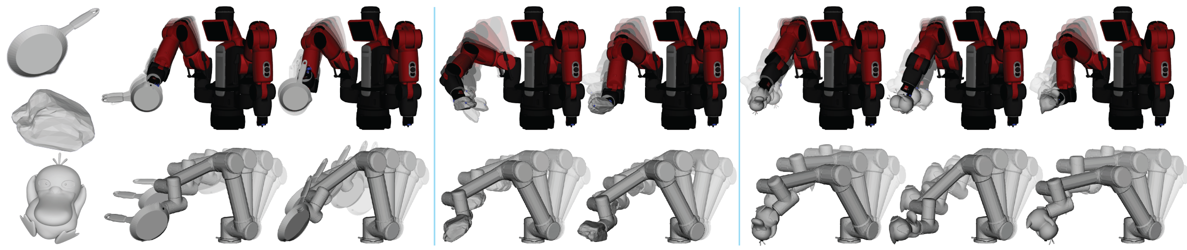
Our framework is generic and generalizable to more challenging cases. It can further generate effective strategies using unconventional daily objects. The costs of operating those objects are ranked from low to high in Fig. 5.7a (Baxter) and Fig. 5.7b (UR5). The experiment reveals some objects (piler and wrench for Baxter, and axe and pan for UR5) are surprisingly more handy for robots compared with the hammer (indicated by the black bar). We further visualize the executed trajectories in Fig. 5.7c. Of note, the same pan is more suitable for UR5 as the cost of operating it is lower than using a hammer but not that effective for Baxter. In comparison, the efficiency of using the rock and the toy (Psyduck) are similar for both robots. These results demonstrate that our learning and planning framework enables a situational tool-use skill for various robots.



(a) Best strategies found for the Baxter robot.



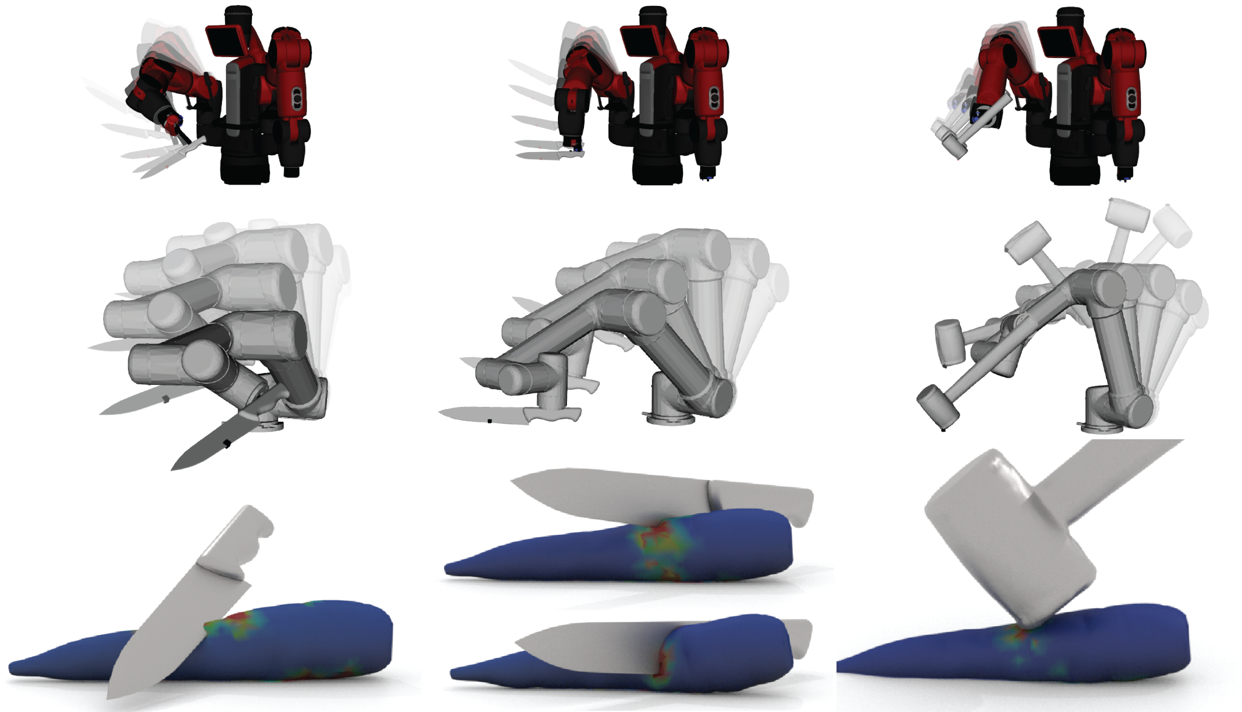
(b) Best strategies found for a UR5 manipulator.



(c) Trajectories of using some of the unconventional tools: a pan, a rock, and a toy.

Figure 5.7: **Effective tool-uses with unseen objects for the walnut-cracking task.** (a)(b) The best strategies (least cost) for ten different objects to crack a walnut use a Baxter robot and a UR5 manipulator, respectively. (c) Examples of valid trajectories of the Baxter robot (upper) and a UR5 manipulator (lower) using a pan (left), a piece of rock (middle), and a Psyduck toy (right).

In another task of cutting carrot, both robots do not perform well if concerned only about the *velocity* as they did in walnut-cracking; the target object will not align with the knife's blade properly as illustrated in Fig. 5.8a. By incorporating tool's *orientation* as uncovered in Fig. 5.4c, the robots overcome this deficiency and produce desired effects successfully; see Fig. 5.8b. Compared with the walnut-cracking task, the cutting task poses greater challenges in selecting unseen objects as tools since not all objects can lead to task completion; *i.e.*, one cannot use a hammer to successfully cut a carrot as a knife does. Yet in Fig. 5.8c, the result still demonstrates the robots' reasonable efforts in this difficult situation by choosing a sharp edge to contact with the object, showing that our framework successfully captures the essential physics in tool-uses and leverages them in producing its own strategies.



(a) (b) (c)

Figure 5.8: **Tool-use strategies for cutting the carrot**. (a) Robots fail to accomplish the task without incorporating a tool's orientation. (b) The successful use of a knife requires incorporating orientation properties as learned in Fig. 5.4c. (c) Even using an object (a hammer) unsuitable for this task, our framework still produces an effective strategy by finding a tool orientation that minimize contact.

5.4.3 Testing Robot Tool-use in Simulation

Finally, we evaluate how well the best strategy found by the proposed framework (*e.g.*, produced strategies in Fig. 5.6a) can be executed in the simulator. This step is crucial as it separates the proposed framework from purely vision-based methods.

Since no existed work can solve the proposed task, we design a kinematic-based motion planner as a baseline that accounts for the physical properties involved in the task. In the case of the walnut-cracking task, the baseline needs to plan a trajectory that moves the functional basis of the tool to the center of the walnut while keeping its surface normal aligned with the gravity direction. Fifty trials are simulated for both the Baxter robot and the UR5 manipulator using trajectories produced by the baseline and the proposed framework, and the parameters governing walnut's fracturing properties in each trial are set based on the values shown in Fig. 5.3a with a randomness of 10% for variations.

Due to the lack of quantitative evaluation of the performance of the walnut cracking task, we conducted a human study to compare the results between the baseline and the proposed framework. Ten participants were recruited online and asked to classify the total of 200 simulated execution results into one of the three statuses based on three instances shown in Fig. 5.9a. An execution is considered successful if more participants regard the walnut's status as cracked. Fig. 5.9bc show eight examples of the results based on the human study. The success rates are shown in Table 5.1, demonstrating the necessity of understanding the physics in tool-use. Together, the results show the proposed framework indeed enables a better understanding of complex physical events that occurred during the tool-uses and successful productions of tool-use behaviors for robots.

5.5 Conclusion and Discussion

We presented a learning and planning framework for robots to understand the physics behind tool-use events and generate tool-use strategies suitable for the robots' own kinematics and dynamics. A physics-based FEM simulator was developed to generate physical properties

Table 5.1: **Success Rate in Cracking Walnut in Simulator.**

Robot Type	Baseline	Proposed
Baxter	14%	62%
UR5	16%	52%

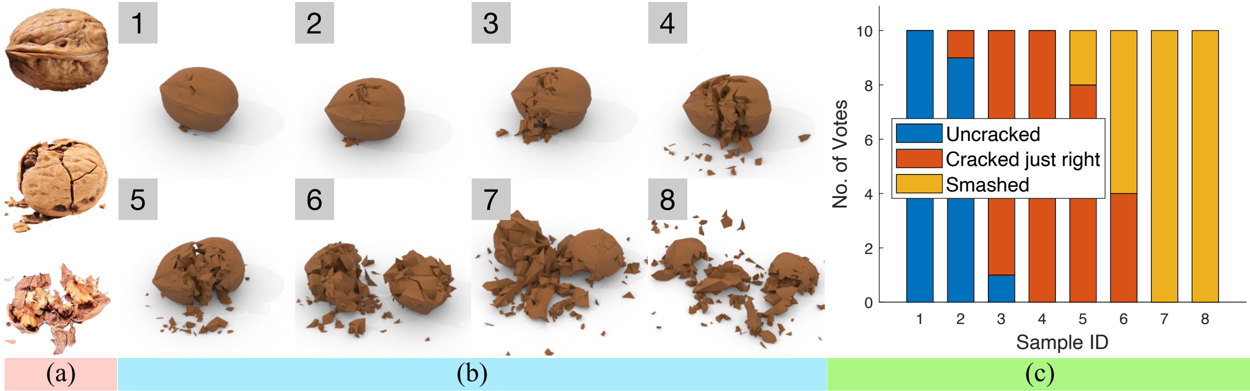


Figure 5.9: **Human evaluation of classifying the status of simulated execution results**. (a) After presenting three instances of walnut being uncracked, cracked just right, and smashed, (b) participants are asked to classify observed simulation results (eight samples for illustration) into one of these three statuses. (c) Sample 3 to 5 are considered successful as most participants regard them as cracked.

in a continuous manner, from which we devised an IDSR algorithm to learn the essential properties critical to the success of the task. By formulating the learned properties into an optimal control-based motion planning scheme, our experiments demonstrated that the proposed framework allows robots to find tool-use strategies different from human demonstrations when handling seen and unseen objects, with better efficiency measured by least joint efforts.

While our work is conducted in simulation, our planning scheme outputs torque commands that are possible for deployment on physical robots in the future. As grasping remains an unsolved problem, we plan to incorporate more sophisticated methods (*e.g.*, [LLJ22]) to generate firm grasp configurations on the tool, such that we can produce more realistic and adaptive tool-uses. The reality gap is another major challenge to realize the physical deployment of the proposed framework. Physics-based simulation is difficult to tune or match the real world precisely. However, it is still a powerful tool for robot understanding and uncovering the task goal.

CHAPTER 6

Conclusion

This dissertation is intended to provide a new perspective for robot perception, where perception is guided by the understanding of potential actions in a scene. As such, the acquired actionable fluent enables a robot to reason about actions a scene affords as well as the potential outcomes of actions and to reach a higher level of autonomy.

Understanding and perceiving the geometry fluent In Chapter 2, we propose a new perspective that emphasizes perceiving the geometry fluent that provides actionable information for enabling an agent to reason about actions an object affords as well as the potential outcomes of actions. Particularly, the geometry fluent is defined as the kinematics of a scene which reflects the underlying functions and constraints of the environment. The functionality of objects and their contextual relations are further organized by a graph-based scene representation, *i.e.*, contact graph, that describe the geometry fluent of the perceived scene.

Understanding and perceiving the topology fluent In Chapter 4, we model the events of object form changes (*e.g.*, fragmentation) using an attributed stochastic grammar model. By understanding the actions of fragmentation, we could perceive the topology fluent, *i.e.*, a new indication of object status during topology fluent changes. Specifically, we propose a probabilistic framework to induce such a grammar from observation to describe the space of topology fluent. This new perspective surpasses prior work that treats objects as a whole, introducing a new dimension for robots to perceive objects (fragments) and utilize fragmentation in complex tasks.

Planning in the fluent space In Chapter 3 and Chapter 5, we introduce how we leverage the perceived geometry and topology fluent to have a robot plan for their actions to accomplish a variety of complex tasks. In Chapter 3, we consolidate the kinematics of the mobile base, the arm, and the object being manipulated collectively as a whole via a Virtual Kinematic Chain (VKC), this novel VKC perspective naturally defines an abstract action that treats the manipulated object as an extended robot limb and incorporates the kinematics of object into that of the robot. By adopting the idea of VKC, planning in the geometry fluent space is reduced to be a planning problem on the VKC. In Chapter 5, we study the interconnection between the geometry and topology fluent in a tool-use scenario. We present a robot learning and planning framework that learns the essential physical properties contributing to the effects of a tool-use event (*e.g.*, how a hammer cracks a walnut) and produces an effective tool-use strategy with the least joint efforts.

We hope our work, as the initial effort, could shed light on future work on more complex object modeling and a more generalized action space a scene affords. In the future, ultimately, an embodied AI agent or a robot could possess a human-level perceptual capability to understand the surroundings and achieve a wide range of task goals in the physical world on its own initiative.

REFERENCES

[AA88] Haruhiko Asada and Yukio Asari. "The direct teaching of tool manipulation skills via the impedance identification of human motions." In *IEEE International Conference on Robotics and Automation (ICRA)*, 1988.

[AA18] Shuichi Akizuki and Yoshimitsu Aoki. "Tactile Logging for Understanding Plausible Tool Use Based on Human Demonstration." In *British Machine Vision Conference (BMVC)*, 2018.

[ADD19] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. "Scan2cad: Learning cad model alignment in rgb-d scans." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[ADN19] Armen Avetisyan, Angela Dai, and Matthias Nießner. "End-to-end cad model retrieval and 9dof alignment in 3d scans." In *International Conference on Computer Vision (ICCV)*, 2019.

[AGH19] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. "CasADi: a software framework for nonlinear optimization and optimal control." *Mathematical Programming Computation*, **11**(1), 2019.

[AHG19] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. "3d scene graph: A structure for unified semantics, 3d space, and camera." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[AST20] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. "Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning." *Proceedings of the National Academy of Sciences (PNAS)*, **117**(47):29302–29310, 2020.

[BAK17] Daniel M Bodily, Thomas F Allen, and Marc D Killpack. "Motion planning for mobile robots using inverse kinematics branching." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[BCC20] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. "Rearrangement: A challenge for embodied ai." *arXiv preprint arXiv:2011.01975*, 2020.

[Bel57] Richard Bellman. "A Markovian decision process." *Journal of mathematics and mechanics*, **6**(5):679–684, 1957.

[BHB13] Felix Burget, Armin Hornung, and Maren Bennewitz. “Whole-body motion planning for manipulation of articulated objects.” In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[Bis94] Christopher M Bishop. “Mixture density networks.” Technical report, Aston University, 1994.

[BKL17] Julien Bidot, Lars Karlsson, Fabien Lagriffoul, and Alessandro Saffiotti. “Geometric backtracking for combined task and motion planning in robotic systems.” *Artificial Intelligence*, **247**:229–265, 2017.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *“Conditional Gaussian distributions,” in Pattern recognition and machine learning*, chapter 2.3.1, pp. 85–88. Springer, 2006.

[BQS20] Jake Brawer, Meiying Qin, and Brian Scassellati. “A causal approach to tool affordance learning.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[CB21] Erwin Coumans and Yunfei Bai. “PyBullet, a Python module for physics simulation for games, robotics and machine learning.” `http://pybullet.org`, 2016–2021.

[CCC16] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age.” *IEEE Transactions on Robotics (T-RO)*, **32**(6):1309–1332, 2016.

[CCL10] Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. “Planning for autonomous door opening with a mobile manipulator.” In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[CD17] Hyung Jin Chang and Yiannis Demiris. “Highly articulated kinematic structure estimation combining motion and skeleton information.” *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **40**(9):2165–2179, 2017.

[CDF17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. “Matterport3D: Learning from RGB-D Data in Indoor Environments.” In *International Conference on 3D Vision (3DV)*, 2017.

[CFG15] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. “Shapenet: An information-rich 3d model repository.” *arXiv preprint arXiv:1512.03012*, 2015.

[CHY19] Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. "Holistic++ scene understanding: Single-view 3D holistic scene parsing and human pose estimation with human-object interaction and physical commonsense." In *International Conference on Computer Vision (ICCV)*, 2019.

[CSW17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. "Realtime multi-person 2d pose estimation using part affinity fields." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[CZ19] Zhiqin Chen and Hao Zhang. "Learning implicit fields for generative shape modeling." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[DCS17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. "Scannet: Richly-annotated 3d reconstructions of indoor scenes." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[DHH20] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. "Robothor: An open simulation-to-real embodied ai platform." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1977.

[EGL19] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. "A tale of two explanations: Enhancing human trust by explaining robot behavior." *Science Robotics*, **4**(37), 2019.

[EGX17] Mark Edmonds, Feng Gao, Xu Xie, Hangxin Liu, Siyuan Qi, Yixin Zhu, Brandon Rothrock, and Song-Chun Zhu. "Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[EHP11] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras. "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[Fea14] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[FGM10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. "Object detection with discriminatively trained part-based models." *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **32**(9):1627–1645, 2010.

[FL03] Maria Fox and Derek Long. "PDDL2. 1: An extension to PDDL for expressing temporal planning domains." *Journal of artificial intelligence research*, **20**:61–124, 2003.

[FN71] Richard E Fikes and Nils J Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." *Artificial intelligence*, **2**(3-4):189–208, 1971.

[FNF18] Fadri Furrer, Tonci Novkovic, Marius Fehr, Abel Gawel, Margarita Grinvald, Torsten Sattler, Roland Siegwart, and Juan Nieto. "Incremental object database: Building 3D models from multiple partial observations." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[GAG15] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. "Aligning 3D models to RGB-D images of cluttered scenes." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[Gal06] Shaun Gallagher. *How the body shapes the mind.* Clarendon Press, 2006.

[GCH20] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrated task and motion planning." *arXiv preprint arXiv:2010.01083*, 2020.

[GFN19] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. "Volumetric instance-aware semantic mapping and 3D object discovery." *IEEE Robotics and Automation Letters (RA-L)*, **4**(3):3037–3044, 2019.

[GHZ20] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. "ADD: Analytically differentiable dynamics for multi-body systems with frictional contact." *ACM Transactions on Graphics (TOG)*, **39**(6):1–15, 2020.

[Gib50] James J Gibson. *The perception of the visual world.* Houghton Mifflin, 1950.

[Gib66] James J Gibson. *The senses considered as perceptual systems.* Houghton Mifflin, 1966.

[GLK18] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. "Ffrob: Leveraging symbolic planning for efficient task and motion planning." *International Journal of Robotics Research (IJRR)*, **37**(1):104–136, 2018.

[GPL20] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. "Online replanning in belief space for partially observable task and motion problems." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[GSL12] Kalin Gochev, Alla Safonova, and Maxim Likhachev. "Planning with adaptive dimensionality for mobile manipulation." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[HBB19] David Hahn, Pol Banzet, James M Bern, and Stelian Coros. "Real2Sim: Viscoelastic parameter estimation from dynamic motion." *ACM Transactions on Graphics (TOG)*, **38**(6):1–13, 2019.

[HGD17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In *International Conference on Computer Vision (ICCV)*, 2017.

[HLN20] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. "Mastering Atari with Discrete World Models." In *International Conference on Learning Representations (ICLR)*, 2020.

[HLR19] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. "Force-and-motion constrained planning for tool use." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[HLS19] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. "Chainqueen: A real-time differentiable physical simulator for soft robotics." In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[HLS20] Dinh-Cuong Hoang, Achim J Lilienthal, and Todor Stoyanov. "Panoptic 3D Mapping and Object Pose Estimation Using Adaptively Weighted Semantic Information." *IEEE Robotics and Automation Letters (RA-L)*, **5**(2):1962–1969, 2020.

[HMA10] Matej Hoffmann, Hugo Marques, Alejandro Arieta, Hidenobu Sumioka, Max Lungarella, and Rolf Pfeifer. "Body schema in robotics: a review." *IEEE Transactions on Autonomous Mental Development*, **2**(4):304–324, 2010.

[HMN21] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. "DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting." In *Robotics: Science and Systems (RSS)*, 2021.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." *IEEE Transactions on Systems Science and Cybernetics*, **4**(2):100–107, 1968.

[HPN16] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. "Scenenn: A scene meshes dataset with annotations." In *International Conference on 3D Vision (3DV)*, 2016.

[HQX18] Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. "Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[HQZ18] Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. "Holistic 3d scene parsing and reconstruction from a single rgb image." In *European Conference on Computer Vision (ECCV)*, 2018.

[HS06] Nicholas P Holmes and Charles Spence. "Beyond the body schema: Visual, prosthetic, and technological contributions to bodily perception and awareness." *Human body perception from the inside out: Advances in visual cognition*, pp. 15–64, 2006.

[HTY18] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. "Pointwise convolutional neural networks." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[HWB21] Jiahui Huang, He Wang, Tolga Birdal, Minhyuk Sung, Federica Arrigoni, Shi-Min Hu, and Leonidas J Guibas. "Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[HZJ21] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. "Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments." In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[HZJ22] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. "Scene Reconstruction with Functional Objects for Robot Autonomy." *International Journal of Computer Vision (IJCV)*, 2022.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[HZX20] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. "Occuseg: Occupancy-aware 3d instance segmentation." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[IH92] Katsushi Ikeuchi and Martial Hebert. "Task Oriented Vision." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1992.

[JCH20] Baoxiong Jia, Yixin Chen, Siyuan Huang, Yixin Zhu, and Song-Chun Zhu. "LEMMA: A Multi-view Dataset for LEarning Multi-agent Multi-task Activities." In *European Conference on Computer Vision (ECCV)*, 2020.

[JK10] Advait Jain and Charles C Kemp. "Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[JLC21] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. "Screwnet: Category-independent articulation model estimation from depth images using screw theory." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[JQZ18] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. "Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars." *International Journal of Computer Vision (IJCV)*, **126**(9):920–941, 2018.

[Jul62] Bela Julesz. "Visual pattern discrimination." *IRE transactions on Information Theory*, **8**(2):84–92, 1962.

[Jul81] Bela Julesz. "Textons, the elements of texture perception, and their interactions." *Nature*, **290**(5802):91–97, 1981.

[Jul84] Bela Julesz. "A brief outline of the texton theory of human vision." *Trends in Neurosciences*, **7**(2):41–45, 1984.

[JV87] Roy Jonker and Anton Volgenant. "A shortest augmenting path algorithm for dense and sparse linear assignment problems." *Computing*, **38**(4):325–340, 1987.

[JZJ21] Ziyuan Jiao, Zeyu Zhang, Xin Jiang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. "Consolidating Kinematic Models to Promote Coordinated Mobile Manipulations." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[JZW21] Ziyuan Jiao, Zeyu Zhang, Weiqi Wang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. "Efficient Task Planning for Mobile Manipulation: a Virtual Kinematic Chain Perspective." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[Kae20] Leslie Pack Kaelbling. "The foundation of efficient robot learning." *Science*, **369**(6506):915–916, 2020.

[KF10] Sertac Karaman and Emilio Frazzoli. "Incremental sampling-based algorithms for optimal motion planning." *Robotics Science and Systems VI*, **104**(2), 2010.

[KGV83] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing." *Science*, **220**(4598):671–680, 1983.

[KHG19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. "Panoptic segmentation." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[KJ12] Brian Kulis and Michael I. Jordan. "Revisiting K-Means: New Algorithms via Bayesian Nonparametrics." In *International Conference on Machine Learning (ICML)*, 2012.

[KL00] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[KL11] Leslie Pack Kaelbling and Tomás Lozano-Pérez. "Hierarchical task and motion planning in the now." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[KLA17] Jonas Koenemann, Giovanni Licitra, Mustafa Alp, and Moritz Diehl. "Openocl–open optimal control library.", 2017.

[KM20] Erez Karpas and Daniele Magazzeni. "Automated planning for robotics." *Annual Review of Control, Robotics, and Autonomous Systems*, **3**:417–439, 2020.

[KMK19] Zachary Kingston, Mark Moll, and Lydia E Kavraki. "Exploring implicit spaces for constrained sampling-based planning." *International Journal of Robotics Research (IJRR)*, **38**(10-11):1151–1178, 2019.

[KOI21] Kento Kawaharazuka, Kei Okada, and Masayuki Inaba. "Adaptive Robotic Tool-Tip Control Learning Considering Online Changes in Grasping State." *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[KR96] David C Knill and Whitman Richards. *Perception as Bayesian inference*. Cambridge University Press, 1996.

[KST19] Monroe Kennedy, Karl Schmeckpeper, Dinesh Thakur, Chenfanfu Jiang, Vijay Kumar, and Kostas Daniilidis. "Autonomous precision pouring from unknown containers." *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[Kuh55] Harold W Kuhn. "The Hungarian method for the assignment problem." *Naval Research Logistics Quarterly*, **2**(1-2):83–97, 1955.

[KWK19] Beomjoon Kim, Zi Wang, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Learning to guide task and motion planning using score-space representation." *International Journal of Robotics Research (IJRR)*, **38**(7):793–812, 2019.

[LaV06] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[LC15] Martin Levihn and Henrik Christensen. "Using environment objects as tools in unknown environments." In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015.

[LCP21] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. "Reinforcement learning for robust parameterized locomotion control of bipedal robots." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[LFS20] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. "Incremental potential contact: intersection-and inversion-free, large-deformation dynamics." *ACM Transactions on Graphics (TOG)*, **39**(4):49, 2020.

[LG14] Nir Lipovetzky and Hector Geffner. "Width-based algorithms for classical planning: New results." In *Proc. of ECAI*, 2014.

[LGL19] Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M Kaufman. "Decomposed optimization time integrator for large-step elastodynamics." *ACM Transactions on Graphics (TOG)*, **38**(4), 2019.

[LHL22] Xingyu Lin, Zhiao Huang, Yunzhu Li, Joshua B. Tenenbaum, David Held, and Chuang Gan. "DiffSkill: Skill Abstraction from Differentiable Physics for Deformable Object Manipulations with Tools." In *International Conference on Learning Representations (ICLR)*, 2022.

[LHP19] Arturo Laurenzi, Enrico Mingo Hoffman, Matteo Parigi Polverini, and Nikos Tsagarakis. "An Augmented Kinematic Model for the Cartesian Control of the Hybrid Wheeled-Legged Quadrupedal Robot CENTAURO." *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[LK00] Steven M Lavalle and James J Kuffner Jr. "Rapidly-Exploring Random Trees: Progress and Prospects." In *Intenational Workshop on Algorithmic and Computational Robotics*, 2000.

[LKJ21] Minchen Li, Danny M Kaufman, and Chenfanfu Jiang. "Codimensional Incremental Potential Contact." *ACM Transactions on Graphics (TOG)*, **40**(4), 2021.

[LLJ21] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. "Synthesizing Diverse and Physically Stable Grasps With Arbitrary Hand Structures Using Differentiable Force Closure Estimator." *IEEE Robotics and Automation Letters (RA-L)*, **7**(1):470–477, 2021.

[LLJ22] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. "Synthesizing Diverse and Physically Stable Grasps with Arbitrary Hand Structures using Differentiable Force Closure Estimator." *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[LLK19] Xueting Li, Sifei Liu, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. "Putting humans in a scene: Learning affordance in 3d indoor environments." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[LMB14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context." In *European Conference on Computer Vision (ECCV)*, 2014.

[LML21] Xuan Li, Jessica McWilliams, Minchen Li, Cynthia Sung, and Chenfanfu Jiang. "Soft Hybrid Aerial Vehicle via Bistable Mechanism." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[LNZ14] Nejc Likar, Bojan Nemec, and Leon Žlajpah. "Virtual mechanism approach for dual-arm manipulation." *Robotica*, **32**(6), 2014.

[LWY20] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. "Category-level articulated object pose estimation." In *International Conference on Computer Vision (ICCV)*, 2020.

[LWZ17] Yang Liu, Ping Wei, and Song-Chun Zhu. "Jointly recognizing object fluents and tasks in egocentric videos." In *International Conference on Computer Vision (ICCV)*, 2017.

[LXS18] Ligang Liu, Xi Xia, Han Sun, Qi Shen, Juzhan Xu, Bin Chen, Hui Huang, and Kai Xu. "Object-aware guidance for autonomous scene reconstruction." *ACM Transactions on Graphics (TOG)*, **37**(4):1–12, 2018.

[LZS18] Hangxin Liu, Yaofang Zhang, Wenwen Si, Xu Xie, Yixin Zhu, and Song-Chun Zhu. "Interactive robot knowledge patching using augmented reality." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[LZZ19] Hangxin Liu, Chi Zhang, Yixin Zhu, Chenfanfu Jiang, and Song-Chun Zhu. "Mirroring without overimitation: Learning functionally equivalent manipulation actions." In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[MB02] Grégoire Malandain and Jean-Daniel Boissonnat. "Computing the diameter of a point set." *International Journal of Computational Geometry & Applications*, **12**(06):489–509, 2002.

[MB19] Roberto Martín-Martín and Oliver Brock. "Coupled recursive estimation for online interactive perception of articulated objects." *International Journal of Robotics Research (IJRR)*, pp. 1–37, 2019.

[MCB18] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. "Fusion++: Volumetric object-level slam." In *International Conference on 3D Vision (3DV)*, 2018.

[MFM04] David R Martin, Charless C Fowlkes, and Jitendra Malik. "Learning to detect natural image boundaries using local brightness, color, and texture cues." *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **26**(5):530–549, 2004.

[MHD17] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[MJP92] Steven Minton, Mark D Johnston, Andrew B Philips, and Philip Laird. "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems." *Artificial Intelligence*, **58**(1-3):161–205, 1992.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." *Nature*, **518**(7540):529–533, 2015.

[MLZ16] Huaqing Min, Ronghua Luo, Jinhui Zhu, Sheng Bi, et al. "Affordance research in developmental robotics: A survey." *IEEE Transactions on Cognitive and Developmental Systems*, **8**(4):237–255, 2016.

[MNB20] Carolyn Matl, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, and Dieter Fox. "Inferring the material properties of granular media for robotic tasks." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[Mor78] Jorge J Moré. "The Levenberg-Marquardt algorithm: implementation and theory." In *Numerical Analysis*, pp. 105–116. Springer, 1978.

[MSK04] Yi Ma, Stefano Soatto, Jana Košecká, and Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer, 2004.

[MT17] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." *IEEE Transactions on Robotics (T-RO)*, **33**(5):1255–1262, 2017.

[MTF15] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. "Affordance detection of tool parts from geometric features." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[MZC19] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[NAI03] Dana S Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J William Murdock, Dan Wu, and Fusun Yaman. "SHOP2: An HTN planning system." *Journal of artificial intelligence research*, **20**:379–404, 2003.

[NBC19] Lakshmi Nair, Jonathan Balloch, and Sonia Chernova. "Tool macgyvering: Tool construction using geometric reasoning." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[New36] Isaac Newton. *The Method of Fluxions and Infinite Series: With Its Application to the Geometry of Curve Lines.* Nourse, 1736.

[NSE19] Lakshmi Nair, Nithin Shrivatsav Srikanth, Zackory M Erickson, and Sonia Chernova. "Autonomous Tool Construction Using Part Shape and Attachment Prediction." In *Robotics: Science and Systems (RSS)*, 2019.

[NSI19] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. "PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[OTF17] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[PCT01] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. "Virtual model control: An intuitive approach for bipedal locomotion." *International Journal of Robotics Research (IJRR)*, **20**(2):129–143, 2001.

[PDP97] Jerry Pratt, Peter Dilworth, and Gill Pratt. "Virtual model control of a bipedal walking robot." In *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[PHN19] Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. "Real-time progressive 3D semantic segmentation for indoor scenes." In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[PJ12] Andrzej Pronobis and Patric Jensfelt. "Large-scale semantic mapping and reasoning with heterogeneous modalities." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[PNH19] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. "JSIS3D: joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[PNZ17] Seyoung Park, Bruce Xiaohan Nie, and Song-Chun Zhu. "Attribute and-or grammar for joint parsing of human pose, parts and attributes." *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **40**(7):1555–1569, 2017.

[PTL18] Quang-Hieu Pham, Minh-Khoi Tran, Wenhui Li, Shu Xiang, Heyu Zhou, Weizhi Nie, Anan Liu, Yuting Su, Minh-Triet Tran, Ngoc-Minh Bui, et al. "SHREC'18: RGB-D object-to-CAD retrieval." In *3DOR: Proceedings of the 11th Eurographics Workshop on 3D Object Retrieval*, 2018.

[QFZ20] Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. "Keto: Learning keypoint representations for tool manipulation." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[QJH20] Siyuan Qi, Baoxiong Jia, Siyuan Huang, Ping Wei, and Song-Chun Zhu. "A Generalized Earley Parser for Human Activity Parsing and Prediction." *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[QZH18] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. "Human-centric indoor scene synthesis using stochastic grammar." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Rey09] Douglas A Reynolds. "Gaussian mixture models." *Encyclopedia of Biometrics*, **741**(659-663), 2009.

[RGA20] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans." In *Robotics: Science and Systems (RSS)*, 2020.

[RHB17] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. "Kinodynamic trajectory optimization and control for car-like robots." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[RHG16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: towards real-time object detection with region proposal networks." *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **39**(6):1137–1149, 2016.

[RZB09] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. "CHOMP: gradient optimization techniques for efficient motion planning." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[SBA11] Ruben Smits, H Bruyninckx, and E Aertbeliën. "KDL: Kinematics and dynamics library.", 2011. `https://www.orocos.org/kdl`.

[SCX20] Zhiqiang Sui, Haonan Chang, Ning Xu, and Odest Chadwicke Jenkins. "Geofusion: geometric consistency informed scene estimation in dense clutter." *IEEE Robotics and Automation Letters (RA-L)*, **5**(4):5913–5920, 2020.

[SDH14] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. "Motion planning with sequential convex optimization and convex collision checking." *International Journal of Robotics Research (IJRR)*, **33**(9):1251–1270, 2014.

[SFR14] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. "Combined task and motion planning through an extensible planner-independent interface layer." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[Sha16] Krishna Shankar. *Kinematics and Local Motion Planning for Quasi-static Whole-body Mobile Manipulation*. PhD thesis, California Institute of Technology, 2016.

[SHK12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor segmentation and support inference from rgbd images." In *European Conference on Computer Vision (ECCV)*. Springer, 2012.

[SKM19] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. "Habitat: A platform for embodied ai research." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[SL09] Michael Schmidt and Hod Lipson. "Distilling free-form natural laws from experimental data." *Science*, **324**(5923):81–85, 2009.

[SLX15] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun rgb-d: A rgb-d scene understanding benchmark suite." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[SMK12] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. "The open motion planning library." *IEEE Robotics & Automation Magazine*, **19**(4):72–82, 2012.

[SNT19] Marvin Stuede, Kathrin Nuelle, Svenja Tappe, and Tobias Ortmaier. "Door opening and traversal with an industrial cartesian impedance controlled mobile robot." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[SOF21] Namiko Saito, Tetsuya Ogata, Satoshi Funabashi, Hiroki Mori, and Shigeki Sugano. "How to Select and Use Tools?: Active Perception of Target Objects Using Multimodal Deep Learning." *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[SSB11] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. "A probabilistic framework for learning kinematic models of articulated objects." *Journal of Artificial Intelligence Research*, **41**:477–526, 2011.

[Ste97] Anthony Stentz. "Optimal and efficient path planning for partially known environments." In *Intelligent unmanned ground vehicles*, pp. 203–220. Springer, 1997.

[SWD21] Yu She, Shaoxiong Wang, Siyuan Dong, Neha Sunil, Alberto Rodriguez, and Edward Adelson. "Cable manipulation with a tactile-reactive gripper." *International Journal of Robotics Research (IJRR)*, **40**(12-14):1385–1401, 2021.

[SYZ17] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. "Semantic Scene Completion From a Single Depth Image." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[TAS18] Marc Toussaint, Kelsey Allen, Kevin A Smith, and Joshua B Tenenbaum. "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning." In *Robotics: Science and Systems (RSS)*, 2018.

[TBP21] Shreshth Tuli, Rajas Bansal, Rohan Paul, et al. "TANGO: Commonsense Generalization in Predicting Tool Interactions for Mobile Manipulators." In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[TJR13] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. "Point-plane SLAM for hand-held 3D sensors." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[TKO17] Kuniyuki Takahashi, Kitae Kim, Tetsuya Ogata, and Shigeki Sugano. "Tool-body assimilation model considering grasping motion through deep learning." *Robotics and Autonomous Systems*, **91**:115–127, 2017.

[Tou15] Marc Toussaint. "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning." In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[TPB87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. "Elastically deformable models." In *SIGGRAPH*, 1987.

[TWL20] Haowen Tang, Ping Wei, Huan Li, and Nanning Zheng. "Inferring Tasks and Fluents in Videos by Learning Causal Relations." In *International Conference on Pattern Recognition (ICPR)*, 2020.

[TWT21] Dylan Turpin, Liquan Wang, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. "GIFT: Generalizable Interaction-aware Functional Tool Affordances without Labels." In *Robotics: Science and Systems (RSS)*, 2021.

[UT20] Silviu-Marian Udrescu and Max Tegmark. "AI Feynman: A physics-inspired method for symbolic regression." *Science Advances*, **6**(16), 2020.

[WDN20] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. "Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[WGZ08] Ying Nian Wu, Cheng-En Guo, and Song-Chun Zhu. "From information scaling of natural images to regimes of statistical models." *Quarterly of Applied Mathematics*, pp. 81–122, 2008.

[WHV05] Ge Wu, Frans CT Van der Helm, HEJ DirkJan Veeger, Mohsen Makhsous, Peter Van Roy, Carolyn Anglin, et al. "ISB recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—Part II: shoulder, elbow, wrist and hand." *Journal of Biomechanics*, **38**(5):981–992, 2005.

[WKM19] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[WS15] Handy Wicaksono and Claude Sammut. "A learning framework for tool creation by a robot." In *Proceedings of ACRA*, 2015.

[WS20] Handy Wicaksono and Claude Sammut. "A cognitive robot equipped with autonomous tool innovation expertise." *International Journal of Electrical & Computer Engineering*, **10**(2), 2020.

[WSJ20] Kentaro Wada, Edgar Sucar, Stephen James, Daniel Lenton, and Andrew J Davison. "MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[WSK15] Yuquan Wang, Christian Smith, Yiannis Karayiannidis, and Petter Ögren. "Cooperative control of a serial-to-parallel structure using a virtual kinematic chain in a mobile dual-arm manipulation application." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[WSK16] Yuquan Wang, Christian Smith, Yiannis Karayiannidis, and Petter Ögren. "Whole body control of a dual-arm mobile robot using a virtual kinematic chain." *International Journal of Humanoid Robotics*, **13**(01), 2016.

[WWZ21] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J Guibas. "Captra: Category-level pose tracking for rigid and articulated objects from point clouds." In *International Conference on Computer Vision (ICCV)*, 2021.

[XHS15] Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. "Autoscanning for coupled scene reconstruction and proactive object analysis." *ACM Transactions on Graphics (TOG)*, **34**(6):1–14, 2015.

[XLZ19] Xu Xie, Hangxin Liu, Zhenliang Zhang, Yuxing Qiu, Feng Gao, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. "Vrgym: A virtual testbed for physical and interactive ai." In *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1–6, 2019.

[XQM20] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. "SAPIEN: A simulated part-based interactive environment." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[XSL20] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. "Interactive Gibson Benchmark: A Benchmark for Interactive Navigation in Cluttered Environments." *IEEE Robotics and Automation Letters (RA-L)*, **5**(2):713–720, 2020.

[YLF20] Tao Yuan, Hangxin Liu, Lifeng Fan, Zilong Zheng, Tao Gao, Yixin Zhu, and Song-Chun Zhu. "Joint Inference of States, Robot Knowledge, and Human (False-)Beliefs." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[YS19a] Shichao Yang and Sebastian Scherer. "Cubeslam: Monocular 3-d object slam." *IEEE Transactions on Robotics (T-RO)*, **35**(4):925–938, 2019.

[YS19b] Shichao Yang and Sebastian Scherer. "Monocular object and plane SLAM in structured environments." *IEEE Robotics and Automation Letters (RA-L)*, **4**(4):3145–3152, 2019.

[YYT11] Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. "Make it home: automatic optimization of furniture arrangement." *ACM Transactions on Graphics (TOG)*, **30**(4), 2011.

[YZW19] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. "Gspn: Generative shape proposal network for 3d instance segmentation in point cloud." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[ZGF20] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, et al. "Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense." *Engineering*, **6**(3):310–345, 2020.

[ZGL19] Chuhang Zou, Ruiqi Guo, Zhizhong Li, and Derek Hoiem. "Complete 3D scene parsing from an RGBD image." *International Journal of Computer Vision (IJCV)*, **127**(2):143–162, 2019.

[ZGW05] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu. "What are textons?" *International Journal of Computer Vision (IJCV)*, **62**(1):121–143, 2005.

[ZJW22] Zeyu Zhang, Ziyuan Jiao, Weiqi Wang, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. "Understanding Physical Effects for Effective Tool-use." *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[ZJZ16] Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. "Inferring forces and learning human utilities from videos." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[ZM07] Song-Chun Zhu and David Mumford. "A Stochastic Grammar of Images." *Foundations and Trends in Computer Graphics and Vision*, **2**(4):259–362, 2007.

[ZS89] Kaizhong Zhang and Dennis Shasha. "Simple fast algorithms for the editing distance between trees and related problems." *SIAM journal on computing*, **18**(6):1245–1262, 1989.

[ZT00] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method, vol. 2*. Butterworth-Heinemann, 2000.

[ZWM98] Song Chun Zhu, Yingnian Wu, and David Mumford. "Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling." *International Journal of Computer Vision (IJCV)*, **27**(2):107–126, 1998.

[ZZ11] Yibiao Zhao and Song-Chun Zhu. "Image parsing with stochastic scene grammar." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[ZZ13] Yibiao Zhao and Song-Chun Zhu. "Scene parsing by integrating function, geometry and appearance models." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[ZZC19] Jiaying Zhang, Xiaoli Zhao, Zheng Chen, and Zhejun Lu. “A review of deep learning-based semantic segmentation for point cloud.” *IEEE Access*, **7**:179118–179133, 2019.

[ZZY15] Bo Zheng, Yibiao Zhao, Joey Yu, Katsushi Ikeuchi, and Song-Chun Zhu. “Scene understanding by reasoning stability and safety.” *International Journal of Computer Vision (IJCV)*, **112**(2):221–238, 2015.

[ZZZ15] Yixin Zhu, Yibiao Zhao, and Song-Chun Zhu. “Understanding tools: Task-oriented object modeling, learning and recognition.” In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[ZZZ20] Zhenliang Zhang, Yixin Zhu, and Song-Chun Zhu. “Graph-based Hierarchical Knowledge Representation for Robot Task Transfer from Virtual to Physical World.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.