KinScene: Model-Based Mobile Manipulation of Articulated Scenes

Cheng-Chun Hsu[1], Ben Abbatematteo[1], Zhenyu Jiang[1], Yuke Zhu[1], Roberto Martín-Martín[1*], Joydeep Biswas[1*]

***Abstract*— Sequentially interacting with articulated objects is crucial for a mobile manipulator to operate effectively in everyday environments. To enable long-horizon tasks involving articulated objects, this study explores building scene-level articulation models for indoor scenes through autonomous exploration. While previous research has studied mobile manipulation with articulated objects by considering object kinematic constraints, it primarily focuses on individual-object scenarios and lacks extension to a scene-level context for task-level planning. To manipulate multiple object parts sequentially, the robot needs to reason about the resultant motion of each part and anticipate its impact on future actions. We introduce KinScene, a full-stack approach for long-horizon manipulation tasks with articulated objects. The robot maps the scene, detects and physically interacts with articulated objects, collects observations, and infers the articulation properties. For sequential tasks, the robot plans a feasible series of object interactions based on the inferred articulation model. We demonstrate that our approach repeatably constructs accurate scene-level kinematic and geometric models, enabling long-horizon mobile manipulation in a real-world scene. Code and additional results are available at `https://chengchunhsu.github.io/KinScene/`**

I. Introduction

Domestic robots in human unstructured environments need to perform long-horizon tasks by reasoning about and actuating articulated objects at a *scene-level*. For example, when putting away the dishes from the dishwasher, the robot must consider actuating the dishwasher door and drawer, as well as the doors and drawers of the cabinet. Reasoning solely about individual joints prevents the robot from understanding intra-object dependencies such as that the dishwasher door needs to be actuated before the dishwasher drawer, and scene-level dependencies such as that the dishwasher door may block the path to the cabinet (Fig. 1).

Previous work has investigated in isolation individual parts of the problem of building, planning and manipulating articulation at the scene level including single-joint estimation with passive sensing [1, 17, 29], zero-shot actuation of unknown objects [11, 19, 22, 34], and task and motion planning given a scene-level articulation model [4, 12, 28], but key open questions that persist are 1) *how to integrate* these components, 2) what are the *assumptions and limitations* inherent to each individual component for effective synthesis, and 3) *how well* different solutions work when integrated in an end-to-end automated solution.

In this paper, we introduce KinScene, a model-based solution to scene-level articulation reasoning and manipulation that involves autonomously exploring the scene to generate a geometric-kinematic model, and using this model to plan and execute scene-informed manipulation plans. KinScene consists of three phases: First, in the *mapping stage*, the robot builds a static map and identifies potential articulated objects. Next, in the *articulation discovery stage*, the robot autonomously visits each articulated object and manipulates it to model its joint parameters and build a scene-level articulation model. Finally, in the *scene-level manipulation stage*, the robot leverages the scene-level articulation model to efficiently plan and execute actions to manipulate articulated objects in the scene to complete long-horizon tasks.

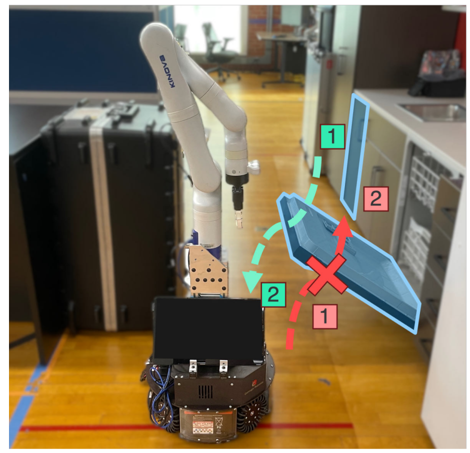


Fig. 1: **KinScene enables scene-level reasoning about articulated objects**. In this scenario, attempting to open the dishwasher would obstruct the path for subsequent interactions. KinScene constructs a scene-level articulation model and plans a feasible trajectory.

We evaluate our approach in a real-world kitchen that features diverse everyday articulated objects of varying sizes and positions. Furthermore, we perform ablations showing that autonomous exploration is essential for articulated object manipulation. The experiments demonstrate that our robot can accurately infer articulation properties through exploration. Leveraging the scene-level articulation model, our robot enhances execution speed by 60.55% and ultimately achieves a higher success in planning and manipulating the articulated objects by reasoning at a scene scale.

II. Problem Formulation

Our goal is to change the kinematic state of the scene to a desired configuration. We assume this configuration is generated by a higher-level task planner and it is needed to perform a long-horizon task. For example, to *unload the dishwasher* the robot needs to open the door of the dishwasher before picking the clean dishes, and open the cabinet doors

[1] Department of Computer Science, The University of Texas at Austin.
[*] equal advising. Correspondence to `chengchun@utexas.edu`.

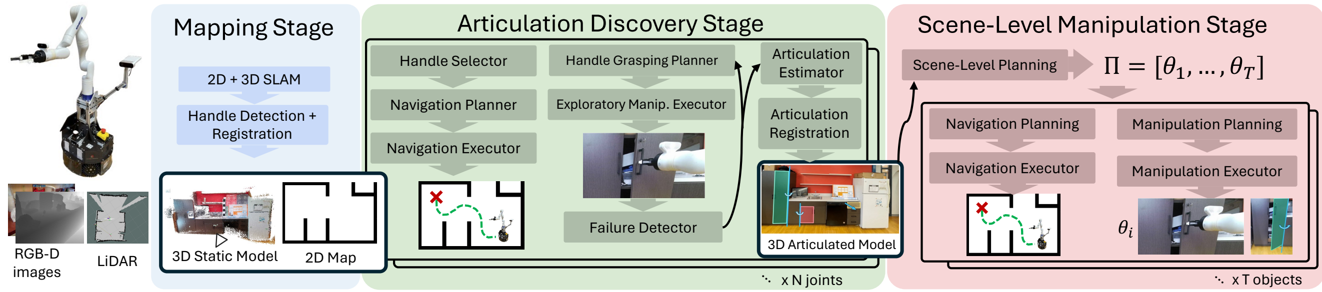


Fig. 2: **KinScene System Overview.** Our approach involves three stages: a mapping stage (left), where the robot conducts a 3D scan and detects handles; an articulation discovery stage (middle), where the robot navigates, interacts, collects observations, and estimates the scene-level articulation model; and a scene-level manipulation stage (right), where the robot plans tasks using the scene-level articulation model and executes long-horizon actions through the articulation planner.

before putting away the dishes. Our goal will be to achieve these states, *i.e.*, planning and executing a sequence of single joint interactions to bring the environment to the desired configuration, taking into account the constraints introduced by the scene-level articulation.

Formally, we represent the scene-level articulation model as a kinematic tree [29], $M = \langle B, V, E \rangle$, consisting of a static 3D base map, B, (*e.g.* including the walls and fixed kitchen counters), vertices $V = \{v_i | i \in [1, N]\}$ consisting of N 3D models of articulated moving parts, and articulated edges, $E = \{e_i | i \in [1, N]\}$, with kinematic constraints between each articulated moving part and the base map [14]. The goal will be then to change the kinematic state of the scene from an initial state, Θ_0, to a given goal state, Θ_g, that involves actuating several joints.

To achieve the desired scene configuration, the robot must reason both at the task level (i.e. which objects to manipulate) and the motion level (i.e. finding collision-free motion plans). As such, we cast this problem as a hierarchical planning problem [12]. The robot must first search for a sequence of single joint changes $\Pi = [\theta_1, \dots, \theta_T]$ using the scene level articulation model M, resulting in a geometrically feasible sequence of object motions that accomplishes the scene goal configuration Θ_g. For each interaction, the robot must then infer a collision-free trajectory τ_i in the state space $\Theta \times S_R$ that accomplishes the subgoal θ_i, including navigating to the object and manipulating it. If no such motion can be found (*e.g.* articulating one object blocks the robot's path to a second), the planner should return to the high level and seek a new object-level plan. This process should be repeated until the robot finds a task plan Π and corresponding collision-free motion plans $\{\tau_i\}_{i=1}^{T}$ that achieve the scene goal configuration Θ_g.

III. Related Work

In this section, we delve into previous research efforts related to the subproblems associated with scene-level articulated object manipulation. Previous studies in articulated object manipulation focused on estimating articulation parameters for manipulation [3, 9, 11, 15, 19, 24, 32, 33] but were limited to controlled settings, lacking generalization to realistic environments. While integrated systems for specific tasks such as door opening have been developed [8, 20], recent research has addressed interaction with unknown objects in indoor spaces [2, 7, 13, 14, 21, 35], albeit primarily focusing on single-object scenarios without extensive scene-level planning capabilities.

IV. KinScene

Our system to build and use scene-level articulated models operates in three stages (Fig. 2): in the initial *mapping stage*, the robot builds a base model with 2D and 3D maps of the scene. The 3D map is enriched with a set of detected handles that guide the physical exploration in the next stage. In the second *articulation discovery stage*, the robot aims at finding and characterizing all joints in the scene. To accurately perceive that, the robot must generate motion actuating each degree of freedom [14, 16, 18]. One by one, the robot moves to each detected handle and attempts to explore it reactively. With the observations from the interaction, it infers the possible articulation that is then registered into the scene-level 3D map for planning. In the final *scene-level manipulation stage*, the robot utilizes the reconstructed scene-level articulation model to plan and execute more efficiently a change in the environment that requires actuating multiple DoF in the right sequence. In the following, we provide details of each stage.

A. Mapping Stage

In the mapping stage, our focus is on building the static model and identifying potential interaction regions (Figure 2, left). Specifically, we employ DROID-SLAM [31] to build the static base map B, and a YOLO [27] handle detector to identify potential interaction hotspots for articulation exploration. Based on the localization of the robot, the handle poses can be converted into the map frame, so that the robot can revisit the location in subsequent stages.

B. Articulation Discovery Stage

In the articulation discovery stage (Figure 2, middle), our focus is on physically interacting with the objects, creating informative observations, and inferring the scene-level articulation model.

Interacting with the objects in this phase poses challenges due to the lack of prior understanding about the kinematic constraints. The robot must determine a base position and arm trajectory that accomplishes each manipulation without self-collision or joint limit violations. To overcome these

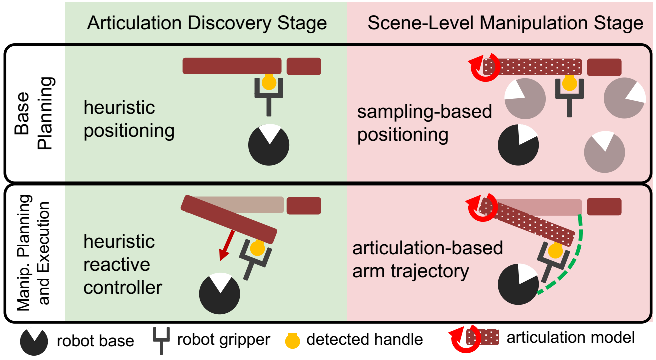


Fig. 3: **Interaction through autonomous exploration vs. articulation planning.** In the Articulation Discovery stage (green), the agent does not have information about the articulation and performs heuristic-based base positioning (in front of the interaction point) and manipulation (admittance controller with motion normal to the plane). In the Scene-Level Manipulation Stage (light red), the robot plans an efficient base positioning and an arm trajectory based on the estimated articulation model obtained during exploration. Thanks to the built model, KINSCENE achieves higher manipulation success and interacts faster than the baselines (Sec. V-B).

challenges, we propose a progressive exploration strategy and failure recovery mechanism.

Exploratory Manipulation Executor. To ensure that the applied force on the object complies with kinematic constraints, we estimate the surface normals of the point cloud observation from the wrist camera at each time step and adjust the end-effector action to pull perpendicular to the normal (Figure 3, left). These surface normal estimates predict how the object part would move under articulated motion without relying on an articulation model. This action is executed through an admittance controller to prevent damage to hardware or furniture.

Failure Detector. Without the articulation model, the robot's exploration often fails due to sub-optimal base positions, self-collisions, and joint limits. By comparing pre- and post-interaction point clouds, we detect if the mobile part has moved beyond a set threshold. Upon failure, the robot adjusts its position for another attempt. Failed attempts provide insights into joint types. Using wrist camera plane detection, joints are classified as revolute or prismatic based on observed rotations. While this doesn't replace a complete articulation model, it helps adjust the base for better motion observations. For revolute joints, the robot moves to the same side as the rotation to preserve space, while for prismatic joints, it moves away to allow for translation. The new base position is set a fixed distance away from the interaction hotspot. These strategies, though not considering object size or position, aid in building the articulation model by providing dynamic motion signals.

Articulation Estimator. During exploration, the robot collects egocentric observations of the object before and after the interactions. By analyzing these observations, KINSCENE can accurately predict the kinematics of the object by using a neural network to infer the articulation model, building upon the prior work Ditto in the House (DiTH) [14].

C. Scene-Level Manipulation Stage

In the scene-level manipulation stage, the robot utilizes the reconstructed scene-level articulation model to plan and execute sequential manipulation tasks. The key components for this stage are presented in the rightmost section of Figure 2.

Scene-Level Planning. To successfully achieve the goal scene configuration Θ_g, the robot must first determine a feasible sequence of object interactions, avoiding collisions between object parts or blocking its path to other objects. KINSCENE first searches for a sequence of single joint changes $\Pi = [\theta_1, \dots, \theta_T]$ such that each θ_i is feasible and the desired scene configuration Θ_g is achieved using a simple task and motion planning implementation [12].

KINSCENE begins by sampling object part trajectories for each object that must be manipulated to achieve Θ_g. Bounding boxes for each part are extracted using the segmentation of the predicted articulation model, and the pose of each mobile part along the articulation trajectory is computed based on articulation parameters. In practice, we sample 6 configurations for each object by interpolating from zero to the maximum state (90 degrees for revolute joints and 15cm for prismatic). The order of interactions is then determined by sampling candidate plans and checking for feasibility. If the bounding boxes of the manipulated parts overlap or the path of the robot is blocked, the plan is ruled infeasible. KINSCENE continues sampling plans until we find a feasible sequence of interactions. The robot then executes each interaction using the Manipulation Planner described below to achieve the scene goal configuration Θ_g.

Manipulation Planning and Execution. At each interaction, the robot uses the inferred articulation models to manipulate the objects. The robot grasps the interaction hotspot $p \in \mathbb{R}^3$ and applies actions that comply with the kinematic constraints until reaching the goal angle g_r for revolute joint, or goal translation g_p for prismatic joint. These sequential actions can be represented as trajectories following prior work [10, 36]. To determine a base position that enables manipulation (preventing self-collision or reaching joint limits), we use a random sampling approach. We sample a set of base positions within a specified range around the object. Among these base positions, we choose the one that can reach the most poses on the trajectory as our final base position during execution (Figure 3, right).

In contrast to the articulation discovery stage, the complete trajectory is now available, eliminating the need to recalculate each step during the actuation of the articulated part. The end-effector trajectory is executed via a position controller to achieve the desired manipulation, and the robot proceeds to the subsequent interactions in the scene-level plan.

V. EXPERIMENTS

We evaluate 1) KINSCENE's effectiveness at performing long-horizon tasks that involve scene-level articulated manipulation; and 2) the effectiveness of different controllers for single-object articulated manipulation compared to KINSCENE. We evaluate KINSCENE in a real-world kitchen scene comprising 4 drawers and 5 cabinet doors of different sizes and shapes. Our mobile manipulator robot is equipped

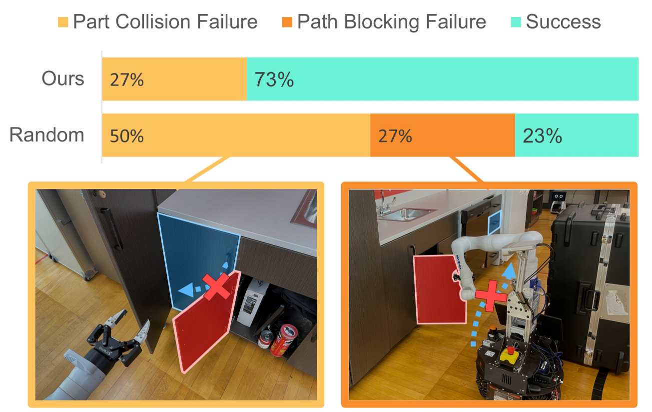


Fig. 4: **Results of Scene-level Manipulation.** KINSCENE enables planning sequences of articulated object interactions. The lower left figure illustrates an example of part collision where the rightmost cabinet blocks actuation of the middle cabinet. The lower right figure illustrates the path blocking failure, where actuating the cabinet first blocks the robot's path to other objects in the plan.

with a four-wheeled custom omnidirectional base and a Kinova Gen3 7 DoF arm.

A. *Scene-level Articulated Manipulation*

We generate five sequences wherein the robot must interact with three articulated objects of varying sizes, positions, and articulation properties within the scenes. Success in each task requires the robot to consider how the outcome of its actions would impact its future plans, similar to the scenario depicted in Fig. 1. Provided with the initial position and the objects requiring articulation over a minimum threshold, the robot autonomously navigates and infers an action sequence to actuate the objects. The entire process can be witnessed in the accompanying video. We compare our method with a *Random* baseline that samples randomly a sequence of object manipulations and executes it. To ensure a fair comparison, we conduct five runs for each attempt for both methods.

Figure 4 tabulates the results — we identify two failure modes, namely part collision failures (an object part cannot be actuated because it collides with another), and path blocking failures (the path of the robot is blocked by an object's part). The Random baseline exhibits a mere 23% success rate and suffers significantly from both object part collisions and mobility blocking failures. In contrast, our method achieves a 73% success rate with 23% fewer object part collision failures and no path blocking failures. The occasional failure of our method is due to erroneous articulation estimation that results in a failure to detect a part collision. These results empirically demonstrate the effectiveness of KINSCENE's approach in unfamiliar scenes with exploration and modeling followed by scene-level model-based planning and control.

B. *Single Object Articulated Manipulation*

We compare KINSCENE's model-based approach to state-of-the-art zero-shot articulated manipulation controllers to evaluate the importance of model-based articulation estimation on robust manipulation. We report the execution time, opening degree, and execution success rate in single object manipulation. The execution time is determined by the duration between the initiation of the action and the point at which the object has been articulated over a minimum threshold of 30° for revolute joints and 5 cm for prismatic joints. We define the *opening degree* as the ratio of the actuation degree to the maximum degree.

TABLE I: Results of Single Object Manipulation.

Method	Execution time			Opening Degree	
	Base ↓	Arm (Rev.) ↓	Arm (Pris.) ↓	Rev. ↑	Pris. ↑
FlowBot3D [10]	-	-	-	0.13	0.23
Exploration-Only	9.85	10.37	9.57	0.43	0.92
Ours	3.41	3.41	4.40	0.78	1.00
Oracle	3.39	2.99	4.68	1.00	1.00

We compare KINSCENE model-based manipulation strategy with three baselines: 1) **FlowBot3D [10]**, a zero-shot approach that generalizes the manipulation of articulated objects from a large amount of training data. 2) **Exploration-Only**, where the interaction uses only the heuristic strategy employed in our articulation discovery stage, without building articulation models. 3) **Model-Based Oracle**, where the interaction is conducted using KINSCENE's model-based approach but assuming ground-truth knowledge of the articulation model. It represents an upper-bound scenario achievable for our model-based approach.

Table I tabulates the execution results. For both revolute and prismatic joints, FlowBot3D is unable to articulate any object to the required threshold. Two factors contribute to this failure: first, FlowBot3D is trained and tested in a controlled lab environment where the optimal viewing angle is manually configured and ensured. And second, the presence of a cluttered environment, and variations in object shape, and size introduce a considerable domain gap, leading to the observed shortcomings. On the other hand, Exploration-Only effectively articulates all objects to the threshold, gathering observations used to construct the articulation model, but it requires iterative updates to its base position due to self-collision and reaching joint limitations before succeeding, adding execution time. Moreover, it only infers per-step actions throughout the interaction, also contributing to a slower execution speed.

In contrast, KINSCENE (Ours) strategically plans the base position using pre-computed action trajectories derived from the acquired articulation model. As a result, Ours achieves a base reposition speed that is 2.88 times faster and executes arm actions 3.04 times faster for revolute joints and 2.17 times faster for prismatic joints compared to Exploration-Only. It successfully reaches 78% of the maximum opening degree compared to Model-based Oracle. This gap suggests that KINSCENE can be further improved by more advanced articulation estimation methods in the future.

VI. CONCLUSION

We introduced KINSCENE, a method for model-based manipulation of articulated objects at the scene level, and showed experimentally that it can autonomously build an articulated 3D model of the scene, and use this model to plan and execute sequences of interactions.

References

[1] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," in *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[2] M. Arduengo, C. Torras, and L. Sentis, "Robust and adaptive door operation with a mobile robot," *Intelligent Service Robotics*, vol. 14, no. 3, pp. 409–425, 2021.

[3] P. R. Barragän, L. P. Kaelbling, and T. Lozano-Pérez, "Interactive bayesian identification of kinematic mechanisms," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 2013–2020.

[4] M. Baum *et al.*, "Opening a lockbox through physical exploration," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, IEEE, 2017, pp. 461–467.

[5] J. Biswas and M. M. Veloso, "Episodic non-markov localization," *Robotics and Autonomous Systems*, vol. 87, pp. 162–176, 2017.

[6] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2544–2550.

[7] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 1656–1662.

[8] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator," in *2010 IEEE International Conference on Robotics and Automation*, pp. 1799–1806.

[9] A. Curtis, L. Kaelbling, and S. Jain, "Task-directed exploration in continuous pomdps for robotic manipulation of articulated objects," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3721–3728.

[10] B. Eisner, H. Zhang, and D. Held, "Flowbot3d: Learning 3d articulation flow to manipulate articulated objects," *arXiv preprint arXiv:2205.04382*, 2022.

[11] S. Y. Gadre, K. Ehsani, and S. Song, "Act the part: Learning interaction strategies for articulated object part discovery," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 752–15 761.

[12] C. R. Garrett *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[13] A. Gupta, M. Zhang, R. Sathua, and S. Gupta, *Opening cabinets and drawers in the real world using a commodity mobile manipulator*, 2024. arXiv: 2402.17767 [cs.RO].

[14] C. Hsu, Z. Jiang, and Y. Zhu, "Ditto in the house: Building articulation models of indoor scenes through interactive perception," in *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, IEEE, 2023, pp. 3933–3939.

[15] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 272–277.

[16] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects.," in *ICRA*, 2013, pp. 5003–5010.

[17] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3706–3715.

[18] R. Martín-Martín and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2494–2501.

[19] R. Martín-Martín and O. Brock, "Coupled recursive estimation for online interactive perception of articulated objects," *The International Journal of Robotics Research*, vol. 41, no. 8, pp. 741–777, 2022.

[20] W. Meeussen *et al.*, "Autonomous door opening and plugging in with a personal robot," in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 729–736.

[21] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1647–1654.

[22] K. Mo, L. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2act: From pixels to actions for articulated 3d objects," *arXiv preprint arXiv:2101.02692*, 2021.

[23] *Nautilus: A map curation tool with autonomous loop closure*, 2020.

[24] N. Nie, S. Y. Gadre, K. Ehsani, and S. Song, "Structure from action: Learning interactions for 3d articulated object structure discovery," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 1222–1229.

[25] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 143–152.

[26] S. Qian, L. Jin, C. Rockwell, S. Chen, and D. F. Fouhey, "Understanding 3d object articulation in internet videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1599–1609.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[28] M. Stilman, "Task constrained motion planning in robot joint space," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 3074–3081.

[29] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.

[30] X. Sun, H. Jiang, M. Savva, and A. X. Chang, "Opdmulti: Openable part detection for multiple objects," *arXiv preprint arXiv:2303.14087*, 2023.

[31] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[32] Y. Wang *et al.*, "Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions," in *European Conference on Computer Vision*, Springer, 2022, pp. 90–107.

[33] R. Wu, K. Cheng, Y. Zhao, C. Ning, G. Zhan, and H. Dong, "Learning environment-aware affordance for 3d articulated object manipulation under occlusions," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[34] R. Wu *et al.*, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," *arXiv preprint arXiv:2106.14440*, 2021.

[35] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak, *Adaptive mobile manipulation for articulated objects in the open world*, 2024. arXiv: 2401.14403 [cs.RO].

[36] H. Zhang, B. Eisner, and D. Held, "Flowbot++: Learning generalized articulated objects manipulation via articulation projection," *arXiv preprint arXiv:2306.12893*, 2023.

Appendix I
System Details

A. Mapping Stage

2D+3D SLAM. To build the static map, we manually navigated the robot around the scene and created a 3D scan with DROID-SLAM [31]. We use the Nautilus [23] tool based on the scene scan for 2D navigation mapping.

Handle Detection and Registration. We are dealing with articulated objects designed for human use; therefore, we assume they incorporate handles that facilitate human-hand interaction. Consequently, we simplify the problem of finding interaction hotspots by focusing on handle detection. We use a YOLO detector [27] trained on the DoorDetect dataset [2] to detect handles. Given an image, the detector outputs the bounding box of the detected handle. We then use non-maximum suppression to remove duplicate handle detections and register the remaining to the global map.

B. Articulation Discovery Stage

The proposed progressive exploration strategy and failure recovery mechanism are shown in Algorithm 1.

Handle Selector. The robot first selects a candidate handle $h \in H$ to explore from the set of detected handle locations.

Navigation Planner and Executor. Upon identifying the candidate of interest on the map, the robot can navigate directly to the object (`PlanBase, GlobalNavTo`). We use ENML [5] for planning and localization during the navigation.

Global localization is prohibitively inaccurate for object manipulation. To precisely position the base for manipulation, we use the MOSSE object tracker [6] to track the relative pose in the end-effector frame between the detected handle and the robot from the video stream. We implement a simple PID controller for the base movement to ensure the robot achieves the desired position (`LocalNavTo`).

Handle Grasping Planner. Based on the robot's egocentric observation, we use the handle detector to retrieve the bounding box of the handle. The center point of the bounding box is selected as the interaction hotspot. This location is then converted into the end effector frame by leveraging depth sensor information and the robot executes a grasp.

Articulation Estimator. For each interaction, the robot acquires observation point clouds o_{pre} and o_{post} and interaction hotspots c_{pre} and c_{post}. To integrate knowledge from the interaction, we incorporate the contact regions into our network input by creating Gaussian heatmaps centered around c_{pre} and c_{post} over the point clouds o_{pre} and o_{post}, respectively. These heatmaps represent the contact regions of interaction and offer insights into the mobile part region and underlying kinematic constraints. Both the point cloud observations and the heatmaps of contact regions are fed into the network for articulation inference (`PredArticulation`). The model then outputs the articulation joint parameters and part segmentation of each articulated object.

Visual occlusion poses a significant challenge when estimating object geometry and kinematics, especially when the camera is often obstructed by the robot arm or object parts during the manipulation of articulated objects. To address this challenge, we collect observations only before and after interactions. In practice, the robot will move a small fixed distance away from the object to capture observation o_{post} (Algorithm 1 lines 24-27). Empirically, we have observed that this improves the observation coverage and results in more accurate articulation estimation.

Algorithm 1: Articulation Discovery Stage

Input : Handle location set H and 3D static map B
Output: Scene-level Articulation Model M

```
1  /* initialize set of object models */
2  A = ∅;
3  foreach h ∈ H do
4      b ← PlanBase(h);
5      GlobalNavTo(b);
6      o ← GetCurrentObs();
7      p ← DetectHandle(o);
8      b_pre ← PlanBase(p);
9      repeat
10         LocalNavTo(b_pre);
11         /* get pre-action observation */
12         o_pre ← GetCurrentObs();
13         /* exploratory interaction */
14         for i ← 1 to n do
15             o' ← GetCurrentObs();
16             f' ← DetectFailure(o_pre, o');
17             if f' ≠ ∅ then
18                 b_pre ← Reposition(o_pre, o') ;
19                 break
20             end if
21             a' ← GetComplianceAction(o');
22             ApplyAction(a');
23         end for
24     until f' = ∅;
25     b_post ← PlanBase(p);
26     LocalNavTo(b_post);
27     /* get post-action observation */
28     o_post ← GetCurrentObs();
29     /* predict articulation model */
30     α ← PredArticulation(o_pre, o_post);
31     A ← A ∪ α;
32 end foreach
33 M ← Register(A, B);
```

Articulation Registration. Finally, to construct the scene-level articulation model M, we register each estimated object model to the static base map B using Colored ICP [25], discarding outlier points (`Register`).

C. Scene-Level Manipulation Stage

Manipulation Planning and Execution. For a revolute joint, the action trajectory lies within the circle within a plane perpendicular to the revolute axis u^r with origin q. Given a rotation angle ϕ, the rotation matrix can be defined as follows:

$$\mathbf{R}(\phi) = \mathbf{I} + \sin\phi [u^r]_\times + (1 - \cos\phi)[u^r]_\times^2 \tag{1}$$

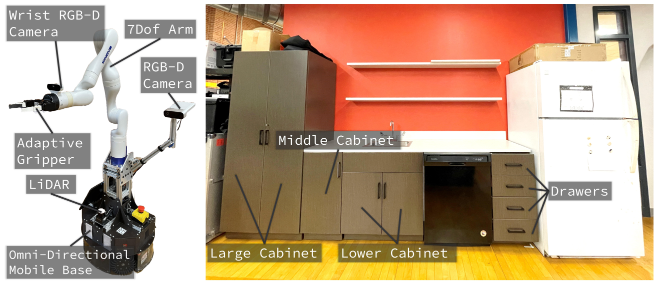


Fig. 5: **Our mobile manipulator and the indoor kitchen scene.** (Left) We integrate a custom omnidirectional base with a 7 DoF torque-controlled arm and a 2-fingered hand, combined with two RGB-D and one LiDAR sensors. (Right) Our kitchen environment contains 7 degrees of freedom including revolute and prismatic in objects of different shapes, weights and heights.

where $\mathbf{I}$ is identity matrix and $[u^r]_\times$ is the skew-symmetric matrix of u^r. By evenly sampling K steps between 0 and the goal angle g_r, the action trajectory can be defined as follows:

$$\tau_{\text{revolute}} = \left\{ \mathbf{R}\left(\frac{i}{K} g_r\right)(p-q)+q \right\}_{\forall i \in [0,K]} \tag{2}$$

For prismatic joints, the K-step trajectory is defined as the translation of the grasp point p to the goal translation g_p along the translation axis u^p.

$$\tau_{\text{prismatic}} = \left\{ p + \frac{i}{K} g_p u^p \right\}_{i \in [0,K]} \tag{3}$$

The action trajectory is then executed with a position controller.

Appendix II
Scene and Hardware Details

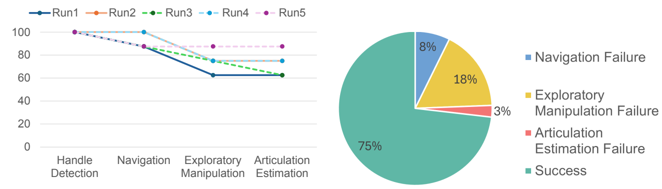
The real-world kitchen scene and our mobile manipulator robot are shown in Figure 5. The robot utilizes a forward-facing Kinect RGB-D camera, a Hokuyo UST-10LX 2D LIDAR for scene mapping and localization, and an Intel Realsense RGB-D camera on the arm wrist to guide manipulation. A Robotiq F-140 end-effector with 3D-printed soft robotic fingers offer flexibility and compliance during manipulation.

Appendix III
Extended Experimental Results

A. Accuracy of Articulation Estimation

We investigate several state-of-the-art methods for articulation estimation in KinScene, and identify key factors affecting their performance. We use the same metrics as in prior work [14]: axis orientation error (Angle Err.) for prismatic and revolute joints, and additionally axis position error (Trans Err.) for revolute joints.

Table II lists the articulation estimation errors in our real-world evaluation scene, over all articulations present in the scene. Art3D relies on RGBD sequence input and thus performs poorly due to the occlusions and imperfect view angles that occur during object interaction. Proprioception performs estimation based on the sequential end-effector pose collected during the physical interaction. This baseline outperforms the vision-based baselines but fails to recover object geometry or part segmentations necessary for long-horizon planning. OPDMulti takes only the initial RGB observation of the object without observing part motion, and thus produces erroneous results. In contrast, DiTH [14] (Ours) takes the observation before and after the interaction and accurately infers object kinematics, significantly outperforming all baselines.



(a) Success rate (in %) for each trial after completion of each step

(b) Cumulative success and sources of failures over all trials

Fig. 6: **Analysing errors during articulation discovery**

TABLE II: Quantitative results of articulation estimation.

Method: Name	Method: Input Modalities	Joint: Prismatic Angle Err.↓	Joint: Revolute Angle Err.↓	Joint: Revolute Trans Err.↓
Art3D[1] [26]	RBG-D Seq.	62.50	46.02	0.29
Proprioception	Pose Seq.	2.77	5.43	0.19
OPDMulti [30]	RGB-D	73.21	11.48	0.78
DiTH (Ours) [14]	Two Point Clouds	8.11	4.07	0.08

[1] The method fails to detect articulation in 4 out of 8 sequences

B. Analysing Errors During Articulation Discovery

We investigate the sources of errors during articulation discovery since KinScene relies on the resulting articulation models to successfully perform scene-level articulation tasks during deployment. Figure 6 shows the sources of errors over five repeated and independent runs of articulation discovery in the scene. For each run, we logged (Figure 6a) the success rates over all articulated objects in terms of the percent of successful 1) detection of handles to articulated objects (Mapping), 2) navigation to the object based on heuristic positioning (Navigation), 3) zero-shot exploratory manipulation, and 4) articulation estimation. We also report the successes and failures over all runs and all objects (Figure 6b) in terms of the success at articulation discovery; and failures of each of the aforementioned four steps. Typically, the main causes of failure stem from exploratory manipulation. These failures predominantly occur when our tracker loses track of the relative pose between the robot and the object during local navigation. Another factor contributing to failure is unreliable motion prediction, leading the arm to reach its limits and consequently fail to recover.

Appendix IV
Limitations

Despite its success, KinScene is not without limitations: we assume that 1) the scene can be represented by a single-level kinematic tree; 2) the articulated objects have detectable and graspable handles, and 3) that the objects have plane normals perpendicular to the articulation motion. Despite these limitations, KinScene is a promising initial solution to the problem of scene-level reasoning for articulation manipulation, which is necessary for robots to successfully perform long-horizon tasks in real human environments.