Query as You Need: Query-centric Diffusion Policy for Generalizable Robotic Assembly

Ziyi Xu*
Department of Mechanical Engineering
Carnegie Mellon University
ziyix2@andrew.cmu.edu

Haohong Lin*
Department of Mechanical Engineering
Carnegie Mellon University
haohongl@andrew.cmu.edu

Shiqi Liu*
Department of Mechanical Engineering
Carnegie Mellon University
shiqiliu@andrew.cmu.edu

Ding Zhao
Department of Mechanical Engineering
Carnegie Mellon University
dingzhao@cmu.edu

Abstract: The robotic assembly task poses a key challenge in building generalist robots due to the intrinsic complexity of part interactions and the sensitivity to noise perturbations in contact-rich settings. The assembly agent is typically designed in a hierarchical manner: high-level multi-part reasoning and low-level precise control. However, implementing such a hierarchical policy is challenging in practice due to the mismatch between high-level skill queries and low-level execution. To address this, we propose the Query-centric Diffusion Policy (QDP), a hierarchical framework that bridges high-level planning and low-level control by utilizing queries comprising objects, contact points, and skill information. QDP introduces a query-centric mechanism that identifies task-relevant components and uses them to guide low-level policies, leveraging point cloud observations to improve the policy's robustness. We conduct comprehensive experiments on the FurnitureBench in both simulation and real-world settings, demonstrating improved performance in skill precision and long-horizon success rate. In the challenging insertion and screwing tasks, QDP improves the skill-wise success rate by over 50% compared to baselines without structured queries.

Keywords: Robotic Assembly, Diffusion Models

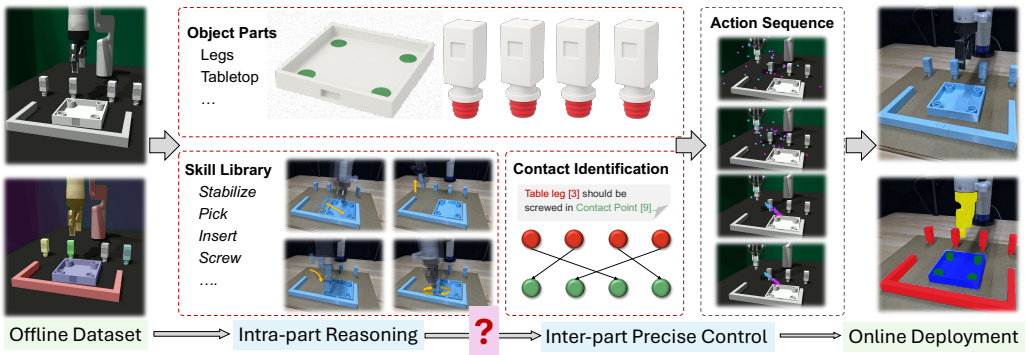

Figure 1: The overview of the furniture assembly task. The task nature comprises three parts: objects, contact points, and skill library. A good assembly policy should identify the correct contact relationship given the assembly context and generate precise action sequences for the robot arm, bridging the gap between intra-part reasoning and inter-part precise control.

*Equal contribution.

1 Introduction

Contact-rich manipulation has been widely recognized as a critical task when building generalist intelligent robots [1, 2, 3]. In this field, robot assembly [4, 5] stands out because it requires policies that are both precise and versatile to control the robot arm and interact with multiple objects. Despite the access to some offline demonstration data from human priors, robotic assembly poses two key challenges: inter-part relational reasoning and intra-part precise control in the online deployment. As is visualized in Figure 1, the first challenge arises from the long-horizon, multi-part nature of the task, which demands accurate prediction of the next skill based on current observations, as well as identifying the correct objects for interaction. The second challenge becomes especially difficult in contact-rich scenarios where successful assembly hinges on precise object alignment. Even minor noise or occlusions in raw sensory observations can completely fail the sim-to-real transfer of low-level *non-prehensile* control policies.

Various methods have been explored to address these challenges. One line of work focuses on high-level reasoning by leveraging cross-modality affordance-based approaches [6], graph-based key point reasoning [7], or skill-based retrieval [8, 9]. However, these methods rely heavily on heuristic-based low-level controllers and predefined skill libraries, thus limiting their adaptability and precision. Meanwhile, recent advances in robot learning have significantly improved the precision and adaptability of low-level policies, thanks to (i) powerful imitation learning backbones such as diffusion models [10, 11, 12] and transformer-based architectures [13, 14], (ii) enhanced learning regimes like residual policies [15] and safe failure prevention methods [16], and (iii) the integration of sensor modalities beyond vision and proprioception, such as tactile sensing [17, 9] or point clouds [18] for improved contact modeling. Still, transferring these policies from simulation to the real world remains challenging due to the inherent difficulty of accurately simulating contact dynamics.

In addition to the individual challenges of high-level and low-level policy design, integrating these two levels in a hierarchical framework introduces further complications. For instance, high-level policies may mis-specify objects or skills, causing the low-level policy to rely on the *false queries* that include irrelevant factors such as background pixels or non-impactful objects, ultimately leading to failure in real-world environments. Hence, it is crucial to establish a parsimonious proxy between high-level and low-level modules.

To address these challenges and establish a robust interface between high-level and low-level modules, we propose Query-centric Diffusion Policy (QDP). Our framework leverages powerful pre-trained foundation models to extract high-level information by specifying both the requisite skills and target objects as a query. This query then serves as a powerful precondition to guide point cloud-based low-level control, forcing the robot agent to focus on the current task-relevant components when generating the action chunks under different contexts. Our contributions are threefold:

- We introduce QDP, a query-centric diffusion policy framework that selects task-relevant queries for guiding low-level policies, enabling accurate skill selection and precise object interaction.
- We propose a query-conditioned policy learning scheme to model the complex geometry captured by point cloud observations, improving precision and facilitating sim-to-real transfer.
- We demonstrate the effectiveness of our approach on FurnitureBench in both simulation and real-world settings, showcasing robust performance under object misalignment and human intervention.

2 Related Work

Sequential decision making for robotic assembly Behavior learning methods are promising for addressing the combinatorial challenges of assembly sequence planning and fine-grained manipulation in robotic assembly tasks. Prior works have tackled the former problem by designing feasible assembly graphs [19], disassembling strategies [20], rearranging components through segmentation [21], or even utilizing manual guidance [22]. While the latter challenge of manipulation involves precise

control and adaptability, recent studies centered around FurnitureBench have employed contact-rich manipulation [9], residual policies bridging sim-to-real gap [18], [23], fine-tuning diffusion policies [24], or extracting skills from offline dataset [8]. Our approach tackles the two challenges by developing a unified hierarchical decision-making framework for robotic assembly.

Hierarchical imitation learning Instead of learning the entire task using a single policy, we can decompose the long-horizon assembly problem into reusable sub-tasks, thereby introducing a hierarchical learning framework. This typically involves a high-level policy and a set of low-level policies. On the high-level decision-making side, approaches often rely on the observable conditions [25] and accurate system dynamics [26] to realize the precondition of applying skill and its effects. On the low-level side, previous skill-based reinforcement learning methods benefit from generalization through skill discovery but face challenges in skill chaining [27] and reward design [9], especially in contact-rich environments. In this context, imitation learning can bridge the gaps between sub-tasks and avoid reward shaping by using full demonstration data. Previous data-driven hierarchical frameworks have included approaches such as sequence segmentation [28], human-in-the-loop data collection systems [29], and high-level planning based on heuristics [30]. Our focus, however, is on designing a hinge connecting high-level and low-level, generalizing across different assets and orders.

Generative diffusion model for planning Recent advancements in diffusion-based generative strategies have led to widespread adoption in robotics, particularly in imitation learning [10] and planning [11, 12]. These approaches have proven effective in capturing multi-modal distributions for planning and policy learning. Building on this diffusion-based framework, previous work explored learning visuomotor policy by integrating various conditions, including 3D representation [31], equivariant models [32], and affordance-based guidance [33], or performing hierarchical abstraction [34, 35, 36, 37].

3 Problem Formulation

Task objective Given a multi-part, long-horizon robot assembly task, our goal is to both generalize in policy-chaining and enhance the quality of robotic assembly, especially in bridging simulation and real-world environments. In general, the objective of our furniture assembly task contain improvements in two parts: higher level decision-making and lower level action-execution. Specifically, to accurately generate a feasible high-level execution sequence and precise low-level policies learned by 3D representation.

Skills: motion primitives Following the setup in FurnitureBench [5], we divide the full assembly sequence to a set of skills $\{$*stabilize, grasp, insert, screw*$\}$, where *stabilize* refers to pushing the tabletop to the corner of the stabilizer, *grasp* is to pick up the objects with propose grasp pose, *insert* requires inserting objects to the target contact point with extra precision, and *screw* focuses on threading the table legs upright into the holes.

Objects: furniture parts To equip robots with the capability to assemble a wide range of furniture, we classify furniture components into two major categories.

- **Primary Component**: These are the core structures of the furniture, such as tabletops, lamp bases, and chair seats.
- **Secondary Components**: These parts complete or assist the furniture's function, including table legs, lamp bulbs, and drawer handles.

Additionally, we define **Contact Points** as specific locations on a Primary Component where a Secondary Component is meant to be attached.

Framework formulation Given the complexity of the task, we emphasize the importance of effective *querying*. Assume we have a set of N_s skill primitives as sub-tasks, N_p furniture parts and

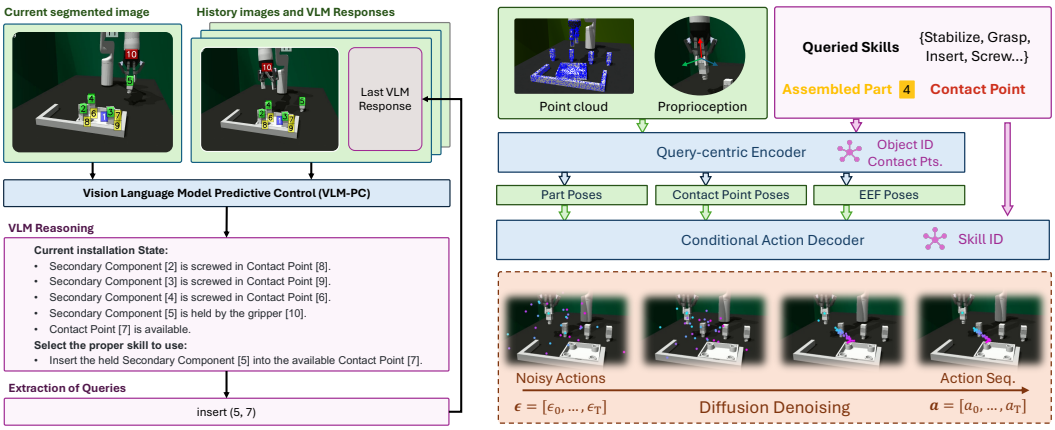

Figure 2: The proposed QDP framework adopts a hierarchical structure composed of a high-level and a low-level policy. The high-level policy processes the initial RGB image of the scene, annotates it as SoM with SAM 2 [39], and leverages a Vision-Language Model (VLM) to identify key components. Using VLM Predictive Control (VLM-PC), it takes in the current image and interaction history to dynamically select skills and objects as queries. At the core of the system, the low-level policy takes in the queries and operates directly on point cloud observations and robot proprioception. The query-centric encoder explicitly estimates the poses of the queried components in the current context and generates precise action sequences of the end effector. By conditioning a diffusion policy on these skill queries and assembly targets, the QDP generates fine-grained, context-aware action sequences.

N_c contact points available on the workbench, we define a query set $\mathcal{Q}$ as:

$$\mathcal{Q} \triangleq \left\{ q = (q_{ij}^{(\ell)}) \in \{0,1\}^{N_s \times N_p \times N_c} \;\middle|\; \sum_{\ell=1}^{N_s} \sum_{i=1}^{N_p} \sum_{j=1}^{N_c} q_{ij}^{(\ell)} = 1 \right\}. \tag{1}$$

At every sub-task performance, our system generates a query $q_{ij}^{(\ell)} \in \mathcal{Q}$, denoting a one-hot selection of furniture part i, contact point j under the sub-task ℓ.

In our task, the action space $\mathcal{A}$ consists of the end-effector's delta poses and absolute poses. The state space $\mathcal{S}$ includes the initial and goal assembly states of the full task, along with observation space $\mathcal{O}$ comprising RGB and depth inputs from external cameras. Additionally, it contains workspace-related states, including the states of individual furniture parts s_i, contact points s_j, and the end-effector s_e.

Building on the notation above, we formulate our robot manipulation task as a complex, long-horizon, partially observable **query-centric** Task and Motion Planning (TAMP) problem [38]. This formulation involves a query-centric execution sequence orderly obtained from $\mathcal{Q}$, a series of motion planning procedures for each sub-task, and the initial and goal states drawn from $\mathcal{S}$. Each motion planning procedure maps an observation input from $\mathcal{O}$ to an action output in $\mathcal{A}$.

4 Methodology

In this section, we will elaborate on the core design of our proposed query-centric diffusion policy. The methodology begins with the generation of queries from VLM-PC in the high-level part (Section 4.1), followed by the introduction of query-centric diffusion policy training in the low-level phase (Section 4.2). The entire pipeline of our method is illustrated in Figure 2.

4.1 High-level: Sequence Generator

Our high-level policy aims to mitigate a long-horizon, contact-rich furniture assembly task by commanding the robot to sequentially apply low-level skills. By leveraging VLM Predictive Control (VLM-PC) [40], our method dynamically selects appropriate low-level skills from the skill library

based on the current scene image and the history of interactions. The model incorporates a re-planning mechanism after each skill execution, allowing the system to handle external perturbations and changes in the environment effectively.

Components Identification The high-level policy identifies furniture components from the initial scene image before installation. To improve the visual grounding capabilities of the VLM, which inherently lacks precise visual grounding [41], we first segment the input image using SAM 2 [39] and annotate it with markers generated by Set-of-Mark (SoM) [42] at the beginning of the installation. The annotated image is then processed by a VLM, which is prompted to identify markers for the primary components, secondary components, contact points, and the robotic arm gripper. Since the VLM relies on both the current image and the interaction history to maintain marker consistency across different steps, we track the initial segmentation masks throughout the sequence. Figure 2 demonstrates the component identification pipeline.

Dynamic Skill Selection The high-level policy dynamically selects the most suitable low-level skill to execute next. After completing each selected skill, the VLM is prompted to re-plan the installation based on the annotated current image and previous n interaction histories, including past annotated images and VLM outputs. The VLM first reasons the current installation progress, and identifies the state of each furniture component in case of external perturbations during installation. It is then prompted to select the most appropriate skill from the skill library. The dynamic skill selection process is shown in Figure 2.

4.2 Low-level: Query-conditioned Action Generator

We develop a query-centric low-level policy that generates an action chunk under the observation input and the selected query $q_{ij}^{(\ell)}$ from the high-level policy. To bridge the sim-to-real gap, we propose to use a point cloud as the visual observation input. By using offline point cloud generated from mesh, we first train a query-centric encoder and explicitly supervise the output with pose of the queried furniture part and contact point. We then train an action decoder for each sub-task with the true state of the queried furniture part as input using the collected demonstrations in simulation under a diffusion policy framework. Together with the perception encoder, forming a two-stage imitation learning structure for the low-level policy.

Query-centric Encoder The query-centric encoder is trained using the full-task dataset. To extract useful latent representations from the point clouds, we use a PointNet [43] architecture. The skill query ℓ , furniture-part query i and contact point query j are numerically input into the encoder network through embeddings, concatenating the point cloud embedding. We also incorporate the end-effector states into a fully connected layer, hypothesizing that these end-effector states will help the encoder network focus on the specific furniture parts being manipulated at the current given moment. The network is a lightweight MLP network, with dropouts. Point clouds struggle with accurate rotation estimation. Hence, we assume the furniture parts are rotation invariant in z-axis, where the tabletop is constrained to rotate within the table plane, retaining only two euler angles representing rotation. The MSE training loss L_{EST} is computed by a weighting among the object-centric position P_i , rotation R_i and the contact-centric position P_j, where $\hat{P}_i$, $\hat{R}_i$ and $\hat{P}_j$ denotes the estimation output:

$$L_{\text{EST}} = \sum_{q_{ij}^{(\ell)} \in \mathcal{Q}} q_{ij}^{(\ell)} \cdot \left(\alpha \|\hat{P}_i - P_i\|^2 + \beta \|\hat{R}_i - R_i\|^2 + \gamma \|\hat{P}_j - P_j\|^2\right), \tag{2}$$

where α, β, γ are the coefficients of the weighted loss. The estimated poses, together with end-effector states, are also useful in sub-task transition, where we set terminal checks to chain the sub-tasks. Since this might fail in assembly check, a time-out check is set to prevent excessive steps of screwing.

Action Decoder The full-task demonstrations are first divided according to the sub-task categories, and sub-task-specific policies are trained accordingly. The sub-task action decoder are built upon the Diffusion Policy framework [10], leveraging a temporal CNN-based U-Net architecture with

FiLM conditioning layers to effectively model and generate precise action chunks. With the intuition that action modality is more pronounced in position inputs, the denoising process is conditioned on the true state of the queried furniture part, the corresponding contact point, and the additional proprioception states of the end-effector. This allows the policy to learn to handle each sub-task ℓ with the necessary precision and contextual awareness. The decoder output is in the end-effector operational space, further continued with an operational space controller (OSC) [44]. Starting from a Gaussian noise $\mathbf{a}^K$, the denoising network $\epsilon_\theta^{(\ell)}$ with parameters θ performs K iterations to gradually denoise a random noise $\mathbf{a}^K$ into the noise-free action chunk $\mathbf{a}^0$, processed in the following equation:

$$\mathbf{a}^{k-1} = \alpha_k \left(\mathbf{a}^k - \gamma_k \sum_{i=1}^{N_p} \sum_{j=1}^{N_c} q_{ij}^{(\ell)} \cdot \epsilon_\theta(\mathbf{a}^k, k, \mathbf{s}) \right) + \sigma_k \epsilon, \tag{3}$$

where $\mathbf{s} = [s_i, s_j, s_e]$ are the queried states from high-level policy, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the Gaussian noise, parameterized function α_k, γ_k and σ_k are generated by the noise scheduler. The combinatorial Mean Square Error (MSE) training loss L_{ACT} for the action generator policy is:

$$L_{ACT} = \|\epsilon^k - \sum_{q_{ij}^{(\ell)} \in \mathcal{Q}} q_{ij}^{(\ell)} \cdot \epsilon_\theta^{(\ell)}(\mathbf{a}^0 + \epsilon^k, k, \mathbf{s})\|^2 \tag{4}$$

5 Experiments

5.1 Environment Design

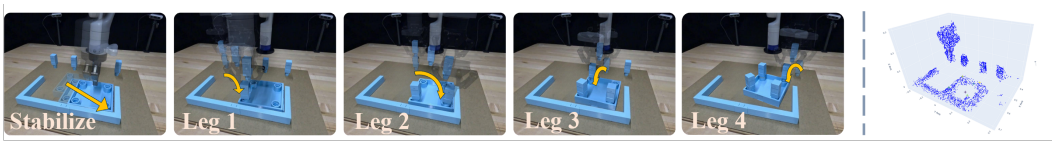

Figure 3: QDP step-by-step assembly process (left). Visualization of a real-world point cloud (right).

Tasks Design: We demonstrate our pipeline on the *Square Table* assembly task, which involves attaching four table legs to the tabletop. This task consists of three sub-tasks for each leg assembly: *grasp*, *insert*, and *screw*, along with an additional *stabilize* step. The ultimate goal is to integrate all four legs onto the tabletop in a feasible assembly sequence, which requires long-horizon decision-making as well as stability and accuracy in both state estimation and action generation.

For numerical evaluation, we prefer using the *One Leg* assembly task, containing all four basic sub-tasks as skill primitives. Notably, we define the success of the *grasp* not only by successfully lifting the leg, but also by ensuring that the grasp occurs within a small, controlled area to prevent dangerous grips or potential collisions with the other legs. We deploy our framework in simulation using Isaac Gym [45], and validate it in the real world with a furniture assembly setup on the Kinova Gen3 robotic arm. Details of the real-world harware setup are provided in Appendix A.2.

Demonstration Data Collection and Annotation We collected full-task demonstrations finishing the whole furniture assembly sequence using a heuristic policy with Finite State Machine (FSM) in simulation. During every step of the collection, we mark the queried furniture parts and contact points in a numerical order, annotate sub-task transitions, and record the full states of the assets. In the real-world deployment, policies are applied zero-shot, with no additional data collection required. Full data collection details are provided in the Appendix A.3.1.

Baselines and Evaluation Protocol We evaluate QDP through three groups of experiments ranging from high-level to low-level policies designed to assess baseline performance and overall effectiveness. 1) We evaluate low-level QDP with two common IL baselines including Behavior Cloning (**BC**) and **DP3** [31] (detailed in Appendix A.3.2). 2) For low-level QDP, we perform two ablation studies — removing the rotation invariance assumption (**w/o rot-inv**) and removing the query-centric structure (**w/o query**) — along with three breakdown studies: a shape transformation (**v/ shape**), a variation in assembly order (**v/ order**), and a tabletop perturbation (**v/ tabletop**) (detailed in Appendix A.3.3). 3) For high-level variants, we have QDP without VLM-PC (**w/o VLM-PC**), without marker (**w/o Marker**) and without interaction history (**w/o History**) (detailed in Appendix A.4.1).

5.2 Results and Analysis

In the following part, we answer the following research questions:

- **RQ1**: How is QDP's precision in both simulation and the real furniture assembly tasks?
- **RQ2**: How does each component of QDP contribute to its superior performance?
- **RQ3**: How is the robustness of QDP against multifarious external perturbations?

Table 1: Comparison of the sub-skills success rate among different baselines and ablation variants.

Method	*stabilize*	*grasp*	*insert*	*screw*
BC	0.92 ± 0.04	0.41 ± 0.14	0.00 ± 0.01	0.08 ± 0.09
DP3	0.83 ± 0.10	0.47 ± 0.02	0.09 ± 0.11	0.17 ± 0.03
QDP	0.95 ± 0.06	0.64 ± 0.14	$\mathbf{0.59 \pm 0.09}$	$\mathbf{0.80 \pm 0.03}$
QDP w/o query	$\mathbf{0.96 \pm 0.03}$	0.60 ± 0.10	0.06 ± 0.05	0.33 ± 0.09
QDP w/o rot-inv	0.96 ± 0.05	$\mathbf{0.70 \pm 0.08}$	0.50 ± 0.08	0.42 ± 0.13
QDP v/ shape	0.95 ± 0.04	0.56 ± 0.11	0.56 ± 0.12	0.76 ± 0.10
QDP v/ order	—	0.38 ± 0.07	0.28 ± 0.13	0.62 ± 0.10
QDP v/ tabletop	0.88 ± 0.08	0.68 ± 0.08	0.46 ± 0.11	—

For **RQ1**, the overall success rate of QDP in performing the *One Leg* task in simulation is 32%. As shown in Figure 7 and Table 1, our query-centric structure outperforms the baselines especially in the *insert* and *screw* sub-task. Additionally, our framework shows a faster convergence in training low-level policies compared with baseline methods. It is evident that jointly estimating all furniture parts without the query structure results in a significant performance drop in the *insert* task. This decline can be attributed to the improved attention provided by the query-centric approach, particularly in focusing on the current executing furniture part. In contrast, a better symmetry representation enhances performance in the *screw* sub-task. This is because the screw task demands precise relocation, and the full quaternion representation may introduce ambiguity, hindering the network's interpretation. In the real-world deployment, QDP successfully completed the *One Leg* task in a zero-shot manner, achieving a success rate of 7/18. The entire assembly process completed by QDP is illustrated in Figure 3(a)-(e).

For **RQ2**, we evaluate the importance of each design module in QDP low-level policy with two variants: **QDP w/o query** and **QDP w/o rot-inv**. We illustrate in Table 1 the success rate of each skill in the assembly process with these ablation variants in the simulation environments. The results show that the structured queries and rotation invariance assumptions have a greater impact on the low-level policy in the harder task, such as *insert* and *screw*.

We further evaluated 3 different variants of QDP high-level policy: **QDP w/o VLM-PC**, **QDP w/o Mark**, and **QDP w/o History**. Figure 4(a) shows the evaluation results for different QDP high-level policy variants; the result shows that removing specific design modules negatively impacts performance. Excluding the VLM-PC (**w/o VLM-PC**) significantly reduces the success rate, demonstrating the effectiveness of dynamic skill selection. Similarly, removing the SoM marks (**w/o Mark**) impairs performance; notably, more errors occur due to incorrect skill query, showing that the special reasoning capability is greatly reduced without facilitating with SoM. Finally, removing historical interactions has the most severe effect, leading to a marked increase in incorrect skill queries, emphasizing the critical role of maintaining historical context for sequential tasks. We further analyzed the effect of varying interaction history length n, and the result is shown in Figure 4(b). The success rate increases rapidly as the history length n increases and stabilizes after $n \geq 3$.

For **RQ3**, we evaluate the robustness of QDP under various perturbations. The breakdown results in the simulation are shown in Table 1(lower). 1) We found that the shape transformation (**v/ shape**) does not result in a significant drop in performance, especially in handling challenging tasks such as *insert*, which aligns with our expectations. 2) In the order variation of the final leg (**v/ order**),

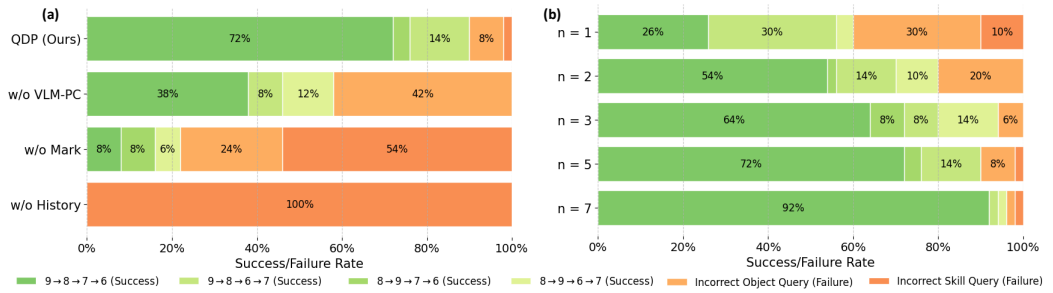

Figure 4: (a) Performance of different QDP high-level policy variants. The sequence $9 \rightarrow 8 \rightarrow 7 \rightarrow 6$ specifies the installation order of the contact points. Incorrect Skill Query indicates that the installation failed because an incorrect skill type was selected. Incorrect Object Query means failure occurs due to querying the wrong furniture component or Contact Point, even though the correct skill query is used. (b) Effect of varying history lengths on performance. $n = k$ indicates that the VLM receives the current image along with the most recent k steps of interaction history as input.

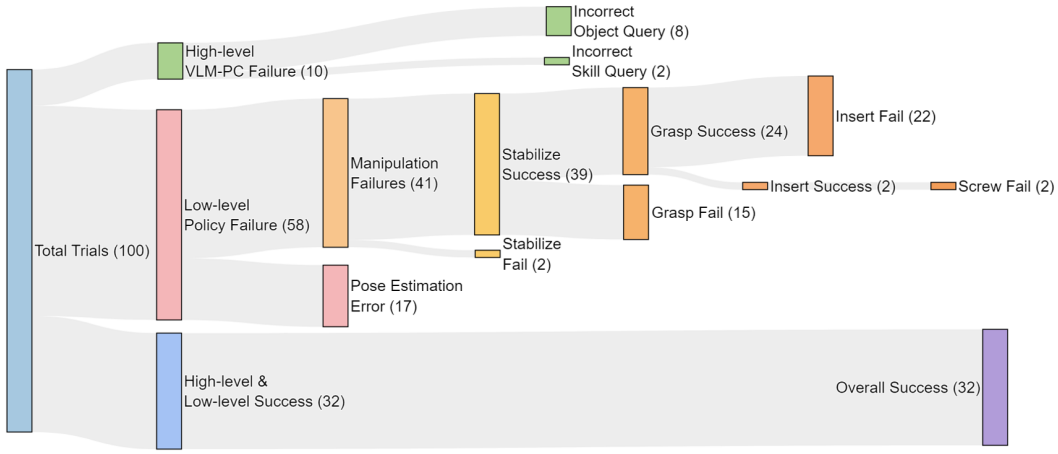

Figure 5: Sankey diagram illustrating the hierarchical flow of failure and success modes throughout the *One Leg* process. We conducted 100 simulation experiments, with most failures occurring in pose estimation and manipulation sub-tasks such as *insert* and *grasp*.

selecting the two holes closest to the camera often results in collisions with other assembled legs, leading to sub-task failure during the *insert* phase. In contrast, switching between the two more distant holes does not result in such issues. This demonstrates that our query-centric framework generalizes across different contact points while also highlighting the importance of selecting feasible contact points in generating the assembly sequence. 3) In simulation and real-world experiments, we demonstrate the robustness of our closed-loop pose estimation and action diffusion module. Upon introducing disturbances to the states of the furniture (**v/ tabletop**), the robot effectively detects these changes, re-estimates the pose, and showcases the multi-modal capabilities of the action diffusion module. A qualitative example is shown in Figure 6(left), where the robot arm is capable of adapting to human perturbation on the tabletop when inserting the leg.

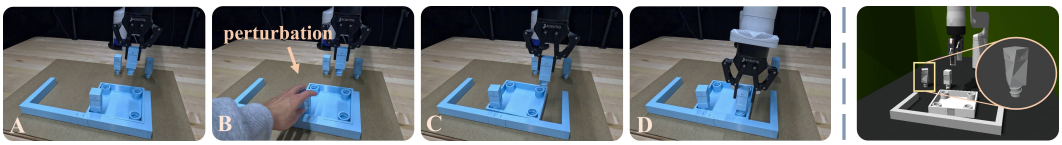

Figure 6: Robustness under perturbation, including disturbance of table-top in real (left) and shape transformation in sim (right).

6 Conclusion

In this work, we proposed QDP, a hierarchical, query-centric diffusion policy framework for long-horizon robotic assembly. By leveraging task-relevant queries, QDP effectively addresses both inter-part relational reasoning and intra-part precision control, achieving strong performance in multi-part assembly tasks. We believe QDP offers a promising step towards furniture assembly tasks, but also a hint that challenges remain in scalable and generalizable robotic assembly.

Limitations

Despite its promising results, QDP has several limitations. Both the high- and low-level modules are designed for "square-table-like" assembly scenarios, which restricts generalization to other furniture types. Furthermore, the pose-based part representations sacrifice fine-grained geometric detail, which is important for accurate low-level control. We do not further explore the sensitivity to multi-camera setups, as it falls beyond the core scope of QDP. Extending the framework to open-set assembly tasks—by enriching part representations and accommodating more diverse object geometries—remains a key direction for future research.

References

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[2] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Y. Lee, E. S. Hu, and J. J. Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. In *2021 ieee international conference on robotics and automation (icra)*, pages 6343–6349. IEEE, 2021.

[5] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *arXiv preprint arXiv:2305.12821*, 2023.

[6] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[7] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[8] J. Zhang, M. Heo, Z. Liu, E. Biyik, J. J. Lim, Y. Liu, and R. Fakoor. Extract: Efficient policy learning by extracting transferable robot skills from offline data. *arXiv preprint arXiv:2406.17768*, 2024.

[9] H. Lin, R. Corcodel, and D. Zhao. Generalize by touching: Tactile ensemble skill transfer for robotic furniture assembly. *arXiv preprint arXiv:2404.17684*, 2024.

[10] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[11] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[12] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[13] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[14] L. Wang, X. Chen, J. Zhao, and K. He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *arXiv preprint arXiv:2409.20537*, 2024.

[15] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4):389–404, 2024.

[16] R. Ahmad and P. Plapper. Safe and automated assembly process using vision assisted robot manipulator. *Procedia Cirp*, 41:771–776, 2016.

[17] K. Yu, Y. Han, Q. Wang, V. Saxena, D. Xu, and Y. Zhao. Mimictouch: Leveraging multi-modal human tactile demonstrations for contact-rich manipulation. In *8th Annual Conference on Robot Learning*, 2024.

[18] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv:2405.10315*, 2024.

[19] Y. Tian, K. D. Willis, B. Al Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. Sueda, et al. Asap: Automated sequence planning for complex robotic assembly with physical feasibility. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4380–4386. IEEE, 2024.

[20] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, and W. Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Transactions on Graphics (TOG)*, 41(6):1–11, 2022.

[21] Y. Li, A. Zeng, and S. Song. Rearrangement planning for general part assembly. In *7th Annual Conference on Robot Learning*, 2023.

[22] R. Wang, Y. Zhang, J. Mao, R. Zhang, C.-Y. Cheng, and J. Wu. Ikea-manual: Seeing shape assembly step by step. *Advances in Neural Information Processing Systems*, 35:28428–28440, 2022.

[23] L. L. Ankile, A. Simeonov, I. Shenfeld, M. T. Villasevil, and P. Agrawal. From imitation to refinement–residual rl for precise visual assembly. In *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.

[24] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[25] D. Driess, J.-S. Ha, and M. Toussaint. Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research*, 40(12-14):1435–1466, 2021.

[26] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[27] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999*, 2021.

[28] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

[29] A. Mandlekar, C. R. Garrett, D. Xu, and D. Fox. Human-in-the-loop task and motion planning for imitation learning. In *Conference on Robot Learning*, pages 3030–3060. PMLR, 2023.

[30] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel. Guided search for task and motion plans using learned heuristics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 447–454. IEEE, 2016.

[31] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[32] J. Yang, Z.-a. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024.

[33] S. Wu, Y. Zhu, Y. Huang, K. Zhu, J. Gu, J. Yu, Y. Shi, and J. Wang. Afforddp: Generalizable diffusion policy with transferable affordance. *arXiv preprint arXiv:2412.03142*, 2024.

[34] W. Li, X. Wang, B. Jin, and H. Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pages 20035–20064. PMLR, 2023.

[35] C. Chen, F. Deng, K. Kawaguchi, C. Gulcehre, and S. Ahn. Simple hierarchical planning with diffusion. *arXiv preprint arXiv:2401.02644*, 2024.

[36] X. Ma, S. Patidar, I. Haughton, and S. James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18081–18090, 2024.

[37] Z. Liang, Y. Mu, H. Ma, M. Tomizuka, M. Ding, and P. Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16467–16476, 2024.

[38] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

[39] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[40] A. S. Chen, A. M. Lessing, A. Tang, G. Chada, L. Smith, S. Levine, and C. Finn. Commonsense reasoning for legged robot adaptation with vision-language models. *arXiv preprint arXiv:2407.02666*, 2024.

[41] B. Zheng, B. Gou, J. Kil, H. Sun, and Y. Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

[42] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. *URL http://arxiv. org/abs/2310.11441*, 2023.

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[44] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[45] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

A Additional Experiment Details

A.1 Model Training Details

Table 2: Hyperparameters of Sub-Task Diffusion Models in QDP

Hyperparameter	Values
U-Net Hidden Dims	[256, 512, 1024]
U-Net Kernel Size	5
U-Net GroupNorm Num of Groups	8
Diffusion Step Emb Dim	128
Observation Horizon T_o	2
Prediction Horizon T_p	16
Action Horizon T_a	8
DDPM Training Steps	100
DDIM Inference Steps	16
Optimizaer	Adam
Learning Rate	1e-4
Learning Rate Scheduler	Cosine Annealing
Weight Decay	1e-6
Batch Size	256

Table 3: Hyperparameters of Low-Level Query-centric Encoder

Hyperparameter	Value
(1) Point Cloud Encoder & Other Embeddings	
PointNet Hidden Dimensions	[64, 128, 256]
PointNet Activation Function	GELU
PointNet Embedding Dimension	512
Query Embedding Dimension	32
End-effector Embedding Dimension	32
(2) Feature Fusion MLP	
MLP Hidden Dimensions	[512, 256, 128]
MLP Activation Function	ReLU
MLP Dropout Rate	0.1
(3) Training Configuration	
Optimizer	Adam
Learning Rate	1e-3
Learning Rate Scheduler	Cosine Annealing
Weight Decay	1e-6
Batch Size	256

The architecture of the low-level query-conditioned model begins with a PointNet-based point cloud encoder. Numerical queries from the high-level sequence generator are embedded using a lookup table. The end-effector states are encoded by a simple MLP without hidden layers. These features are then concatenated and fused through another bigger MLP, and the resulting representation is used to predict the estimated states. We then train our sub-task action decoder conditioned on these states. Hyperparameters of the low-level models are shown in Table 2 and 3.

A.2 Real-world Hardware Setup

We deploy our framework on the Kinova Gen3 collaborative robot arm, using four RealSense D435 cameras to capture real-world point clouds and video frames. The furniture parts are reset to a

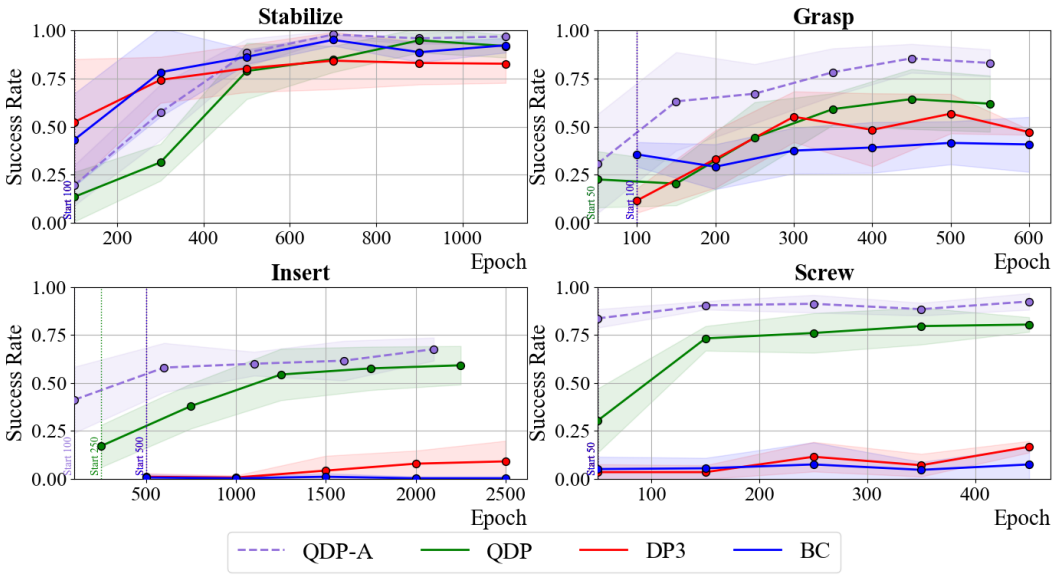

Figure 7: Learning curves for comparison of four methods: **QDP**, QDP with only the sub-task action decoder (**QDP-A**), **DP3** and **BC**. **QDP** demonstrates a better performance and faster convergence especially in bottleneck sub-tasks such as *insert* and *screw*.

relatively fixed state, with randomness introduced in their position and rotation. We use a paperboard to fix the position of the *stabilizer* relative to the robot arm. To avoid singularities during grasping, the furniture legs are repositioned to a standing position. However, we argue that this setup does not simplify the task, as a random yaw rotation is also applied, adding complexity to the scenario.

For real-world applications, we first integrate point clouds captured by all four cameras in the world frame and then clip out the workspace. Since the synthetic point clouds in the simulation are generated based on mesh surface density, it is essential to preprocess the real-world point clouds through voxel down-sampling. Additionally, we segment the ground plane and filter out any outliers. After a final down-sampling step, we retain a total of 4096 points, as is illustrated in Figure 3(f).

Our policy outputs an action chunk consisting of end-effector delta poses in simulation and absolute poses in real-world experiments. To satisfy the quasi-static assumption, the Kinova Gen3 robot arm tracks the first eight points in the action chunk orderly, naturally resulting in longer motion durations, while model inference time is tested comparable to that of the original Diffusion Policy.

A.3 Low-level Evaluation Protocol

A.3.1 Data Collection

Full task trajectories are collected using pre-defined FSM on the Isaac Gym simulation environment. To improve generalization, we introduce a jitter at each step, applying it uniformly to the states of every asset involved. For the furniture assets, we use a synthetic point cloud that is subsampled from mesh files, eliminating the need for prolonged rendering in simulation environments. For the robot arm, we mitigate the sim-to-real gap by collecting real-world point clouds of the gripper in both *Open* and *Close* modes, which replace the synthetic point clouds. Every demonstration is not only initialized with a *Medium* randomness [5], but also has a randomly sampled execution order of the furniture parts, representing different sequences generated by the high-level sequence generator. We constrained the order of the insertion holes, since a different order may result in a collision with the previously inserted parts. We collect a total of 300 full-task demonstrations of the *square table* with randomly generated execution sequences in simulation. Given the high step count involved in the *screw* sub-task, we limit each demonstration of *screwing* to 50 steps, combining these with other sub-tasks to train the student estimator.

A.3.2 Baseline Implementation Details

For a fair comparison, we implement the same PointNet backbone for all the baselines and our QDP in the experiment and evaluate per-skill performance among them. To ensure a consistent comparison, we focus on executing a fixed leg, as the baseline methods tend to underperform in a multi-component assembly setting. We use the same number of demonstrations for the fixed leg as in the dataset for each approach. Since QDP has two learning stages, in order to fairly compare the convergence, we pair intermediate checkpoints of pose estimation with intermediate checkpoints of teacher policy.

A.3.3 Ablation and Breakdown Study

Besides the aforementioned baselines, we also evaluate the performance of QDP's low-level ablation variants: QDP without the rotation invariance assumption (**w/o rot-inv**), where instead of using two euler angles representing rotation, we use full quaternions. QDP without query-centric (**w/o query**), where instead of using the queried states, we estimate the states of all furniture parts jointly and choose the queried part heuristically.

We also perform three breakdowns of the success rate on low-level policies. 1) We introduce a shape transformation by twisting the furniture leg (**v/ shape**: shape variation, as shown in Figure 6), which challenges the generalization of the state estimation module with point cloud input. 2) To ensure numerically valid results, we evaluate the *One Leg* task on the final leg to be assembled, randomly varying the assembly order for the last hole (**v/ order**: order variation). For example, we randomly arrange the hole closest to the camera to be the last one assembled, while the other holes have already been completed. 3) Building on the *One Leg* setup, we introduce an additional variation where we perturb the tabletop randomly when executing each sub-task (**v/ tabletop**: tabletop variation). This setup is conducted both in simulation and real-world, designed to validate the robustness of low-level skills.

For the baselines, ablation study, and shape variation in the breakdown study, each evaluation is conducted 5 times, each 50 roll-outs in the simulation environment, using 10 parallel environments and randomly generating an executable order during the reset process. Other evaluations are conducted 5 times, each 10 roll-outs in single simulation environments, with the setup tailored to their specific configuration.

A.4 High-level Evaluation Protocol

The QDP high-level policy was evaluated through a comprehensive analysis comprising two parts: ablation studies and robustness tests. In the ablation studies, we evaluated the policy's performance across various variants. For the robustness test, we evaluated the policy's ability to handle and recover from perturbations.

A.4.1 High-level Ablation Study Details

The ablation study focused on evaluating the performance of QDP's high-level variants: QDP without VLM-PC (**w/o VLM-PC**), where the installation sequence is generated solely based on the initial frame, without utilizing VLM Predictive Control. QDP without marker (**w/o Marker**), this variant excludes the markers generated by SoM in the input image. QDP without interaction history (**w/o History**), this variant excludes interaction history input to the VLM. Additionally, we analyzed the impact of history length on the policy's performance.

A.4.2 Evaluation Procedure

To evaluate the performance of the high-level policy independently from the low-level policy since a correctly chosen skill might fail due to the inherent randomness of the low-level policy. In cases where the chosen skill failed during execution, we reset the simulated environment to its state before execution and attempted to re-execute the skill. If the skill could not be successfully executed within

a predefined retry limit, it was considered a failure. Otherwise, we proceeded to evaluate the next chosen skill. We utilized GPT-4o as the VLM for the evaluation process.

A.5 Additional assumptions

We assume smooth transitions between consecutive sub-tasks within the skill chaining process. We make our best efforts to ensure the terminal state of a given sub-task is a subset of the initial state of the subsequent sub-task.