Manual2Skill: Learning to Read Manuals and Acquire Robotic Skills for Furniture Assembly Using Vision-Language Models

Chenrui Tie*[1] Shengxiang Sun*[2] Jinxuan Zhu[1] Yiwei Liu[4] Jingxiang Guo[1]
Yue Hu[5] Haonan Chen[1] Junting Chen[1] Ruihai Wu[3] Lin Shao[1]
[1]National University of Singapore
[2]University of Toronto [3]Peking University [4]Sichuan University [5]Zhejiang University

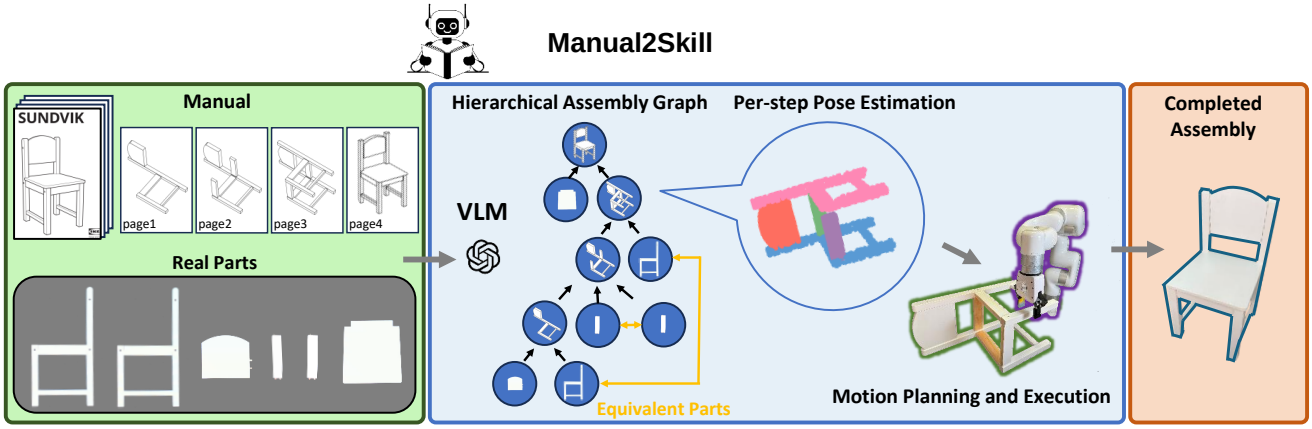


Figure 1. **Overview of Manual2Skill Framework.** We propose Manual2Skill, which learns manipulation skills from manuals, enabling robots to understand and execute complex manipulation tasks in a manner akin to humans. The green region showcases the input of our pipeline: the pictures of the assembly manual and real parts. The blue region depicts our pipeline: 1) a Vision-Language Model (VLM) generates a Hierarchical Assembly Graph, 2) a per-step pose estimation module predicts the 6D-poses of components, and 3) a motion planning and execution module controls the robot arms to assemble the furniture autonomously.

Abstract

Humans excel at interpreting abstract instruction manuals to perform complex manipulation tasks, which is a capability that remains challenging for robots. We present Manual2Skill, a framework that enables robots to execute complex assembly tasks using high-level manual instructions. Our approach leverages a Vision-Language Model (VLM) to extract structured information from instructional images and constructs hierarchical assembly graphs capturing furniture parts and their relationships. For execution, a pose estimation model predicts 6D poses of parts, and a motion planner generates executable actions. We demonstrate the effectiveness of Manual2Skill by successfully assembling several real-world IKEA furniture items, highlighting its ability to manage long-horizon manipulation tasks with both efficiency and precision. This work advances robot learning from manuals and brings robots closer to human-level understanding and execution. *Project Website*

1. Introduction

Humans can learn manipulation skills from instructional images or texts, such as assembling IKEA furniture by following manuals. In contrast, robots typically rely on data-intensive approaches like imitation or reinforcement learning [43, 59]. Bridging this gap remains challenging, as manuals use abstract diagrams designed for human understanding [32, 48, 49].

Manuals encode structural information, breaking down high-level goals into ordered subgoals. This is essential for understanding task flow, dependencies, and spatial relationships [19, 33, 38]. Prior works have leveraged sketches or trajectories [15, 42], but are limited to simple tasks.

We propose Manual2Skill, a novel framework that enables robots to learn manipulation skills directly from visual manuals. As shown in Figure 1, our system parses manuals into hierarchical graphs using a vision-language model (VLM), estimates 6D assembly poses, and generates executable motion plans for furniture assembly.

In summary, our main contributions are as follows:

- We propose Manual2Skill, a generalizable framework for robotic complex skill learning from manuals, applied to IKEA furniture assembly.
- We introduce a hierarchical graph generation pipeline that utilizes a VLM to extract structured information for assembly tasks. Our pipeline facilitates real-world assembly and extends to other assembly applications.
- We define a novel assembly pose estimation task within the learning-from-manual framework. We predict the 6D poses of all involved parts at each assembly step to meet real-world assembly requirements.
- We perform extensive experiments to validate the effectiveness of our proposed system and modules.
- We conduct extensive experiments validating Manual2Skill on four real IKEA furniture items.

2. Problem Formulation

Given a complete set of 3D parts and its assembly manual, the goal is to generate a physically feasible sequence of robotic actions for autonomous furniture assembly. Manuals typically use schematic diagrams and symbols to convey step-by-step instructions in a universally understandable format. The manual is defined as a set of N images. $\mathcal{I} = \{I_1, I_2, \cdots, I_N\}$, where each image I_i illustrates a specific assembly step (e.g., merging parts).

The furniture consists of M parts $\mathcal{P} = \{P_1, P_2, \cdots, P_M\}$, where each *part* is initially disconnected. A *subassembly* is any proper subset of $\mathcal{P}$ containing connected parts (for example, $\{P_1, P_2\}$). A *component* refers to either parts or subassemblies.

Using the manual and 3D parts, the system generates an assembly plan: each step aligns with a manual image and specifies the involved components, their 6D poses, and required assembly actions or trajectories.

3. Technical Approach

We give an outline of our approach, for a more detailed version, please refer to Appendix .2. We automate furniture assembly using a Vision-Language Model (VLM) to interpret IKEA-style manuals and guide robot actions. Given a manual and a pre-assembly scene, the VLM generates a hierarchical assembly graph showing part dependencies. A pose estimation model then predicts 6D poses for each part using manual images and point clouds. Finally, the system plans and executes motions based on predicted poses and the graph structure (Fig. 2).

3.1. VLM Guided Hierarchical Assembly Graph Generation

The VLM takes multi-image inputs (manual pages and pre-assembly scene) and outputs a hierarchical graph. Leaf nodes represent parts, and parent nodes are subassemblies, enabling structured, step-wise assembly. We give the formal definition of the hierarchical graph in Appendix .13. To build the graph, we perform two stages:

- Stage I: Part Association. The VLM labels physical parts in the scene using sketches from the manual. Outputs are structured as triplets {name, label, role}, aiding interpretability and identifying equivalent parts
- Stage II: Step-wise Part Identification. Segmented manual images help the VLM identify parts/subassemblies in each step. Outputs are lists of (step, parts), later merged into the hierarchical graph.

3.2. Per-step Assembly Pose Estimation

Given an assembly order, we train a cross-modal transformer to predict 6D poses $T \in SE(3)$ for each component represented in the camera frame of I_i using manual images and part point clouds. The model encodes images and point clouds with their respective encoders, fuses features via a graph transformer [54], and regresses poses using geodesic and Chamfer distances as loss function. The loss function jointly considers pose prediction accuracy and point cloud alignment, following [30, 60]. To support the training for the pose estimation task, we create a dataset, detailed in Appendix .4.

Note that the number of components at each step is not fixed, depending on the subassembly division of the furniture. Canonical alignment via PCA ensures orientation consistency for each component. For equivalent parts, loss permutations resolve pose ambiguities (see Appendix .5)

3.3. Robot Assembly Action Generation

- **Frame Alignment:** At each assembly step, the previous stage predicts each component's pose in the camera frame of the manual image. However, real-world robotic systems operate in their world frame, requiring a 6D transformation between these coordinates.
- **Grasping & Motion Planning:** After scanning each real-world part, we obtain the corresponding 3D meshes for each part. We employ FoundationPose [52], and the Segment Anything Model (SAM) [24] to obtain the initial poses of all parts in the scene. Geometry-specific strategies handle sticks (centroid grasp) and flat parts (edge grasp). We provide further details on these grasping methods in Appendix .10. RRT-Connect [26] plans collision-free trajectories using scene point clouds.
- **Assembly Insertion Policy:** The robot arm positions a component near its target pose. Assembly insertions require force sensing and closed-loop control, but automated insertion is out of scope for this work. Currently, human experts handle insertion.

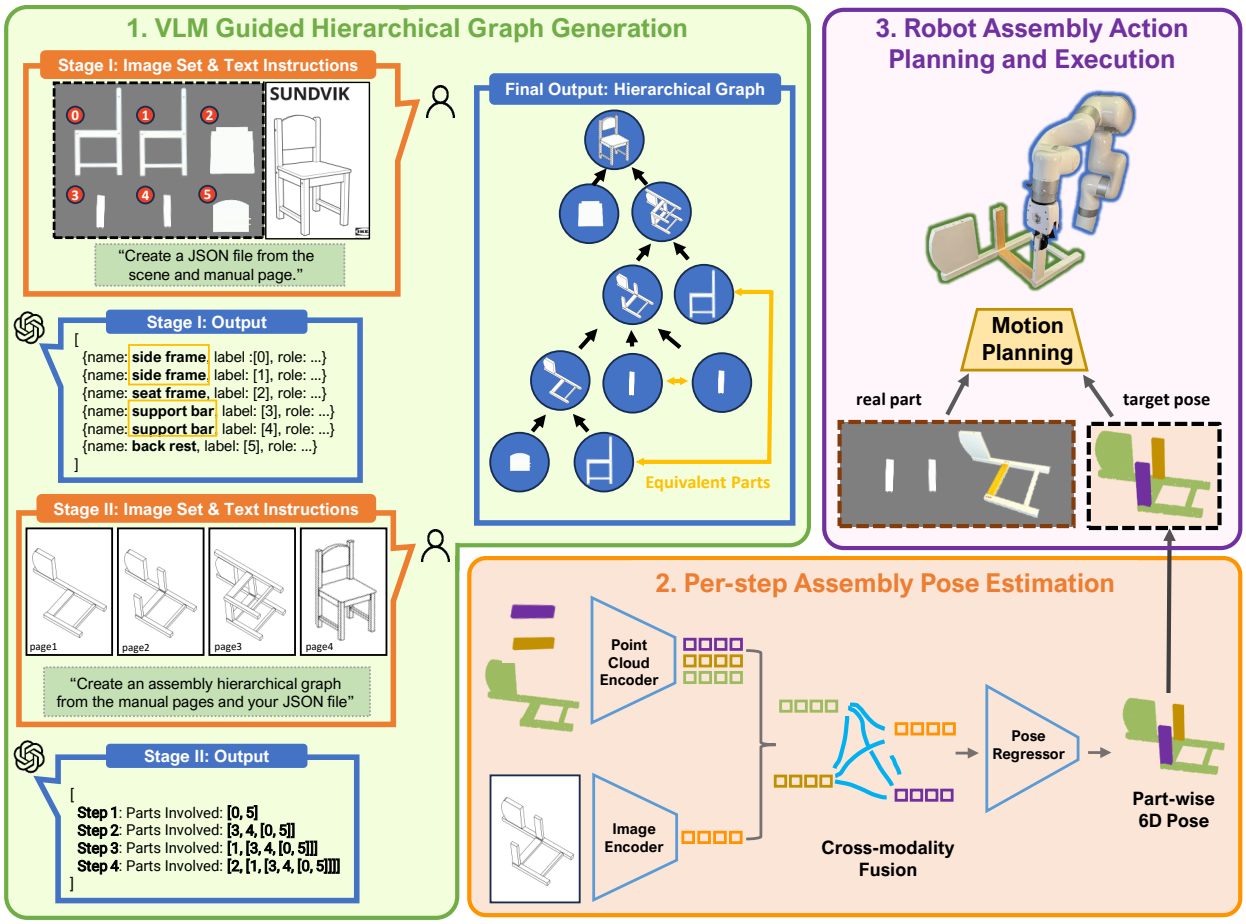


Figure 2. **Framework Overview.** (1) GPT-4o [1] is queried with manual pages to generate a sequential assembly plan, represented as a hierarchical assembly graph. (2) The furniture components' point clouds and corresponding manual images are processed by a pose estimation module to predict target poses for each component. (3) The system sequentially executes the assembly by planning and performing robotic actions based on the hierarchical assembly graph and estimated poses.

4. Experiments

4.1. Hierarchical Assembly Graph Generation

Using the IKEA-Manuals dataset [49], our VLM-based method achieves **62% success rate** in generating correct assembly graphs (vs. 22% for SingleStep baseline as seen in Table 1). Figure 5 shows qualitative improvements over geometric clustering approaches.

4.2. Per-step Assembly Pose Estimation

Our cross-modal model reduces pose error by 60% compared to prior work [29], achieving 0.20m Chamfer distance (Table 2). The graph transformer architecture improves part relationship modeling, with 86.8% part alignment accuracy. We also provide qualitative results for each furniture category in Figure 6.

4.3. Overall Performance Evaluation

In PyBullet simulations, our pipeline achieves 58% success rate across 50 furniture items (vs. 30% for baseline). Failures primarily stem from VLM graph errors propagating to pose estimation. See Table 3 for more details.

4.4. Real-world Assembly Experiments

We successfully assemble 4 IKEA items with 60-85% completion rates (Table 4). Figure 3 shows the manual images, per-step pose estimation results, and real-world assembly process of the 4 items: Flisat (Wooden Stool), Varier a (Iron Shelf), Sundvik (Chair), and Knagglig (Box). Failures occur due to collision-avoidance challenges in cluttered scenes. For detailed implementation of our real-world experiments, please check Appendix .10.

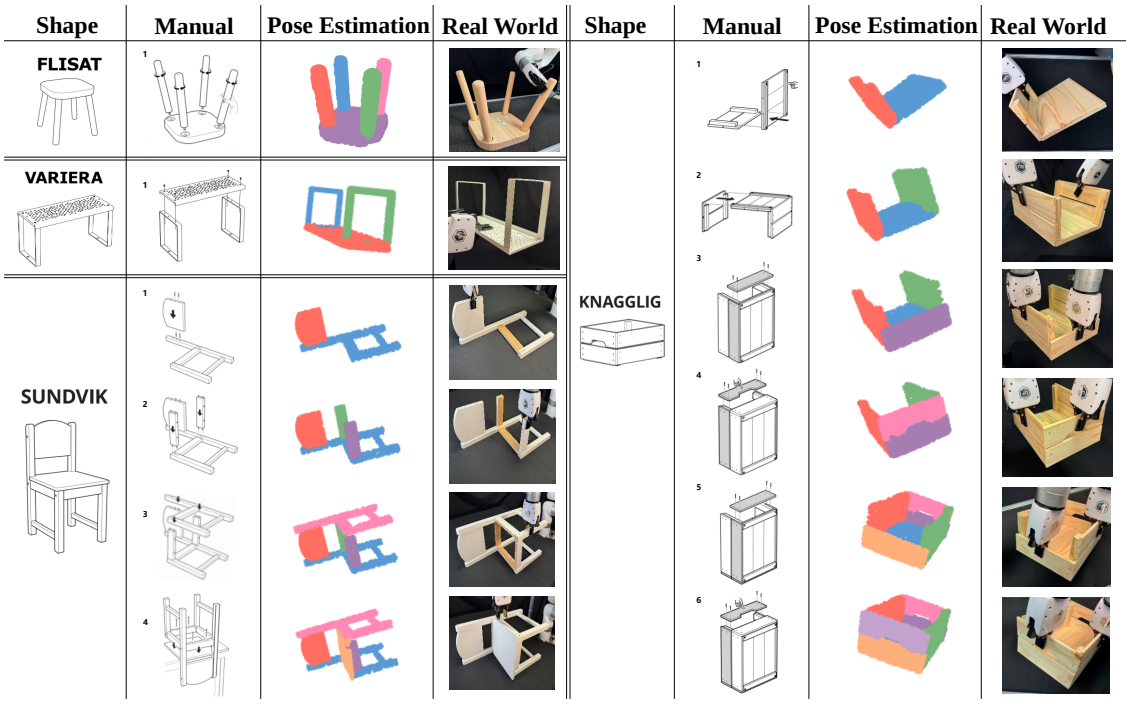


Figure 3. **Qualitative Evaluation on real IKEA furniture items.** This figure illustrates the assembly process of various IKEA furniture items, including FLISAT, VARIERA, SUNDVIK, and KNAGGLIG, with our approach. For each item, we display the manual images, per-step 3D parts pose estimation results, and real-world assembly outcomes.

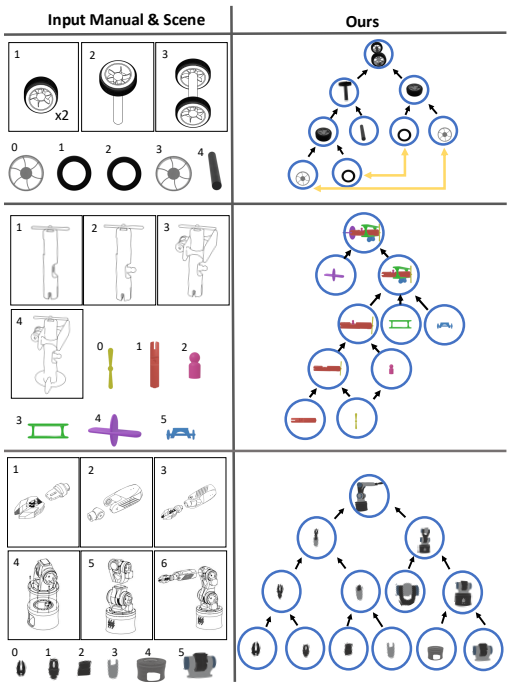


Figure 4. **Pipeline Extension Beyond Furniture Assembly.**

4.5. Generalization to Other Assembly Tasks

Zero-shot tests on toy cars, aircraft models, and robotic arms in Figure 4 show 100% graph generation accuracy, demonstrating broader applicability. Manual2Skill adapts to varied manual styles without retraining.

5. Conclusion

We present Manual2Skill, a novel framework that uses VLMs to learn robotic manipulation skills from human manuals. We validate our approach on IKEA furniture assembly and show its applicability to broader tasks. This work advances robots' ability to acquire complex skills from human instructions, opening new avenues for learning diverse manipulations.

6. Limitation

This paper investigates acquiring complex manipulation skills from manuals. However, limitations remain: it primarily identifies assembly objects but misses details like grasp points and connector locations (e.g., screws). Incorporating a vision-language model to extract such information could improve robotic insertion. Additionally, the method does not address automated fastening actions reliant on force and tactile sensing, which we leave for future work.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 2, 10

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 1

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 1

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 5

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 5

[6] Yun-Chun Chen, Haoda Li, Dylan Turpin, Alec Jacobson, and Animesh Garg. Neural shape mating: Self-supervised object assembly with adversarial shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12724–12733, 2022. 1

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023. 1

[8] Allan Costa, Manvitha Ponnapati, Joseph M. Jacobson, and Pranam Chatterjee. Distillation of msa embeddings to folded protein structures with graph transformers. *bioRxiv*, 2021. doi: 10.1101/2021.06.02.446809. URL https://www.biorxiv.org/content/early/2021/06/02/2021.06.02.446809. 5

[9] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. ACM, 2015. 6

[10] Bi'an Du, Xiang Gao, Wei Hu, and Renjie Liao. Generative 3d part assembly via part-whole-hierarchy message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20850–20859, 2024. 1

[11] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics (T-RO)*, 2023. 4, 12

[12] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 1

[13] Thomas Funkhouser, Hijung Shin, Corey Toler-Franklin, Antonio García Castañeda, Benedict Brown, David Dobkin, Szymon Rusinkiewicz, and Tim Weyrich. Learning how to match fresco fragments. *Journal on Computing and Cultural Heritage (JOCCH)*, 4(2):1–13, 2011. 1

[14] Andrew Goldberg, Kavish Kondap, Tianshuang Qiu, Zehan Ma, Letian Fu, Justin Kerr, Huang Huang, Kaiyuan Chen, Kuan Fang, and Ken Goldberg. Bloxnet: Generative design-for-robot-assembly using vlm supervision, physics simulation, and a robot with reset. *arXiv preprint arXiv:2409.17126*, 2024. 1

[15] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023. 1

[16] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *arXiv preprint arXiv:2305.12821*, 2023. 1

[17] Haoxu Huang, Fanqi Lin, Yingdong Hu, Shengjie Wang, and Yang Gao. Copa: General robotic manipulation through spatial constraints of parts with foundation models. *arXiv preprint arXiv:2403.08248*, 2024. 1

[18] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. 1, 2

[19] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. Roboexp: Action-conditioned scene graph via interactive exploration for robotic manipulation. *arXiv preprint arXiv:2402.15487*, 2024. 1

[20] Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G Kim, and Adriana Schulz. Au-

tomate: A dataset and learning approach for automatic mating of cad assemblies. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 1

[21] Ananth Jonnavittula, Sagar Parekh, and Dylan P Losey. View: Visual imitation learning with waypoints. *arXiv preprint arXiv:2404.17906*, 2024. 1

[22] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video. *arXiv preprint arXiv:2410.24221*, 2024. 1

[23] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1, 2

[24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 2, 4

[25] Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *2013 IEEE International conference on robotics and automation*, pages 855–862. IEEE, 2013. 1

[26] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 2, 4, 6

[27] Youngwoon Lee, Edward S Hu, and Joseph 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. 1

[28] Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18061–18070, 2024. 1

[29] Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas Guibas. Learning 3d part assembly from a single image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 664–682. Springer, 2020. 3, 1, 5, 6, 7, 8

[30] Yichen Li, Kaichun Mo, Yueqi Duan, He Wang, Jiequan Zhang, and Lin Shao. Category-level multi-part multi-joint 3d shape assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3281–3291, 2024. 2, 1, 3

[31] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2025. 2, 14

[32] Yunong Liu, Cristobal Eyzaguirre, Manling Li, Shubh Khanna, Juan Carlos Niebles, Vineeth Ravi, Saumitra Mishra, Weiyu Liu, and Jiajun Wu. Ikea manuals at work: 4d grounding of assembly instructions on internet videos. *arXiv preprint arXiv:2411.11409*, 2024. 1

[33] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019. 1, 13

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

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 5

[36] Gianluca Scarpellini, Stefano Fiorini, Francesco Giuliari, Pietro Moreiro, and Alessio Del Bue. Diffassemble: A unified graph-diffusion model for 2d and 3d reassembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28098–28108, 2024. 1

[37] Silvia Sellán, Yun-Chun Chen, Ziyi Wu, Animesh Garg, and Alec Jacobson. Breaking bad: A dataset for geometric fracture and reassembly. *Advances in Neural Information Processing Systems*, 35:38885–38898, 2022. 1

[38] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023. 1

[39] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv preprint arXiv:2403.12910*, 2024. 1

[40] Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[41] Francisco Suárez-Ruiz, Xian Zhou, and Quang-Cuong Pham. Can robots assemble an ikea chair? *Science Robotics*, 3(17):eaat6385, 2018. 1

[42] Priya Sundaresan, Quan Vuong, Jiayuan Gu, Peng Xu, Ted Xiao, Sean Kirmani, Tianhe Yu, Michael Stark, Ajinkya Jain, Karol Hausman, Dorsa Sadigh, Jeannette Bohg, and Stefan Schaal. Rt-sketch: Goal-conditioned imitation learning from hand-drawn sketches, 2024. URL https://arxiv.org/abs/2403.02709. 1

[43] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8, 2024. 1

[44] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. 1

[45] Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl DD Willis, and Wojciech Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Transactions on Graphics (TOG)*, 41(6):1–11, 2022. 1

[46] Yunsheng Tian, Karl DD Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro 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. 1, 7

[47] Sai H Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, 2024. 1

[48] Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Chin-Yi Cheng, and Jiajun Wu. Translating a visual lego manual to a machine-executable plan. In *European Conference on Computer Vision*, pages 677–694. Springer, 2022. 1

[49] Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Ran Zhang, Chin-Yi Cheng, and Jiajun Wu. Ikea-manual: Seeing shape assembly step by step. *Advances in Neural Information Processing Systems*, 35:28428–28440, 2022. 1, 3, 2, 4, 5, 6, 7, 10

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

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 2

[52] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024. 2, 4, 11

[53] Ruihai Wu, Chenrui Tie, Yushi Du, Yan Zhao, and Hao Dong. Leveraging se (3) equivariance for learning 3d geometric shape assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14311–14320, 2023. 1, 8

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32 (1):4–24, 2020. 2, 3

[55] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023. 2

[56] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. 1

[57] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023. 1

[58] Mingxin Yu, Lin Shao, Zhehuan Chen, Tianhao Wu, Qingnan Fan, Kaichun Mo, and Hao Dong. Roboassembly: Learning generalizable furniture assembly policy in a novel multi-robot contact-rich simulation environment. *arXiv preprint arXiv:2112.10143*, 2021. 1

[59] Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54 (12):7173–7186, 2024. doi: 10.1109/TCYB.2024.3395626. 1

[60] Jiahao Zhang, Anoop Cherian, Cristian Rodriguez, Weijian Deng, and Stephen Gould. Manual-pa: Learning 3d part assembly from instruction diagrams. *arXiv preprint arXiv:2411.18011*, 2024. 2, 3, 8

[61] Minghua Zhang, Yunfang Wu, Weikang Li, and Wei Li. Learning universal sentence representations with

mean-max attention autoencoder. *arXiv preprint arXiv:1809.06590*, 2018. 5

[62] Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[63] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. 2

[64] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pages 1199–1210. PMLR, 2023. 1

[65] Zuyuan Zhu and Huosheng Hu. Robot learning from demonstration in robotic assembly: A survey. *Robotics*, 7(2):17, 2018. 1

[66] Zortrax Library. Zortrax robotic arm, n.d. URL https://library.zortrax.com/project/zortrax-robotic-arm/. Accessed: 2025-02-01. 7

Manual2Skill: Learning to Read Manuals and Acquire Robotic Skills for Furniture Assembly Using Vision-Language Models

Supplementary Material

.1. Related Works

.1.1. Furniture Assembly

Part assembly is a long-standing challenge with extensive research exploring how to construct a complete shape from individual components or parts [6, 13, 20, 27, 29, 36, 45, 46, 53]. Broadly, we can categorize part assembly into *geometric assembly* and *semantic assembly*. *Geometric assembly* relies solely on geometric cues, such as surface shapes or edge features, to determine how parts mate together [6, 10, 37, 53]. In contrast, *semantic assembly* primarily leverages high-level semantic information about the parts to guide assembly process [13, 20, 27, 29, 45].

Furniture assembly is a representative *semantic assembly* task, where each part has a predefined semantic role (e.g., a chair leg or a tabletop), and the assembly process follows intuitive, common-sense relationships (e.g., a chair leg must be attached to the chair seat). Previous studies on furniture assembly have tackled different aspects of the problem, including the motion planning [41], multi-robot collaboration [25], and assembly pose estimation [29, 30, 58]. Researchers have developed several datasets and simulation environments to facilitate research in this domain. For example, Liu et al. [32], Wang et al. [49] introduced IKEA furniture assembly datasets containing 3D models of furniture and structured assembly procedures derived from instruction manuals. Additionally, Lee et al. [27] and Yu et al. [58] developed simulation environments for IKEA furniture assembly, while Heo et al. [16] provides a reproducible benchmark for real-world furniture assembly. However, existing works typically focus on specific subproblems rather than addressing the entire assembly pipeline. In this work, we aim to develop a comprehensive framework that learns the sequential process of furniture assembly from manuals and deploys it in real-world experiments.

.1.2. VLM Guided Robot Learning

Vision Language Models (VLMs) [57] have been widely used in robotics to understand the environment [17] and interact with humans [39]. Recent advancements highlight VLMs' potential to enhance robot learning by integrating vision and language information, enabling robots to perform complex tasks with greater adaptability and efficiency [18]. A potential direction is the development of the Vision Language Action Model (VLA Model) that can generate actions based on the vision and language inputs [2, 3, 23, 44]. However, training such models requires vast amounts of data, and they struggle with long-horizon or complex manipulation tasks. Another direction is to leverage VLMs to guide robot learning by providing high-level instructions and perceptual understanding. VLMs can assist with task descriptions [17, 18], environment comprehension [19], task planning [47, 56, 62], and even direct robot control [28]. Additionally, Goldberg et al. [14] demonstrates how VLMs can assist in designing robot assembly tasks. Building on these insights, we explore how VLMs can interpret abstract manuals and extract structured information to guide robotic skill learning for long-horizon manipulation tasks.

.1.3. Learning from Demonstrations

Learning from demonstration (LfD) has achieved promising results in acquiring robot manipulation skills [7, 12, 64]. For a broader review of LfD in robotic assembly, we refer to Zhu and Hu [65]. The key idea is to learn a policy that imitates the expert's behavior. However, previous learning methods often require fine-grained demonstrations, like robot trajectories [7] or videos [21, 22, 40]. Collecting these demonstrations is often labor-intensive and may not always be feasible. Some works propose to learn from coarse-grained demonstrations, like the hand-drawn sketches of desired scenes [42] or rough trajectory sketches [15]. These approaches reduce dependence on expert demonstrations and improve the practicality of LfD. However, they are mostly limited to tabletop manipulation tasks and do not generalize well to more complex, long-horizon assembly problems. In this work, we aim to extend LfD beyond these constraints by tackling a more challenging assembly task using abstract instruction manuals.

.2. Detailed Technical Approach

Our approach automates furniture assembly by leveraging the VLM to interpret IKEA-style manuals and guide robotic execution. Given a visual manual and physical parts in a pre-assembly scene, a VLM generates a hierarchical assembly graph, defining which parts and subassemblies are involved in each step. Next, a per-step pose estimation model predicts 6D poses for each component using a manual image and the point clouds of involved components. Finally, for assembly execution, the estimated poses are transformed into the robot's world frame, and a motion planner generates a collision-free trajectory for part mating.

This paper shows an overview of our framework in Fig. 2. We describe the VLM-guided assembly hierarchical graph generation in Appendix .2.1, followed by per-step assembly pose estimation in Appendix .2.5 and assembly

action generation based on component relationships in Appendix .2.8.

.2.1. VLM Guided Hierarchical Assembly Graph Generation

This section demonstrates how VLMs can interpret IKEA-styled manuals to generate high-level assembly plans. Given a manual and a real-world image of furniture parts (*pre-assembly scene image*), a VLM predicts *a hierarchical assembly graph*. We show one example in Fig. 2. In this graph, leaf nodes represent atomic parts, while non-leaf nodes denote subassemblies. We structure the graph in multiple layers, where each layer contains nodes representing parts or subassemblies involved in a single assembly step (corresponding to one manual image). The directed edges from the children to a parent node indicate that the system assembles the parent node from all its children nodes. Additionally, we add edges between equivalent parts, denoting these parts are identical($e.g.$ four legs of a chair). Representing the assembly process as a hierarchical graph can decomposes the assembly into sequential steps while specifying necessary parts and subassemblies. We give the formal definition of the hierarchical graph in Appendix .13. We achieve this in two stages: *Associating Manuals with Real Parts* and *Identifying Parts needed in Each Image*.

.2.2. VLM Capabilities and General Prompt Structure

The task is inherently complex due to the diverse nature of input images. Manuals are typically abstract sketches, whereas *pre-assembly scene images* are high-resolution real-world images. Such diversity requires advanced visual recognition and spatial reasoning across varied image domains, which are strengths of VLMs due to their training on extensive, internet-scale datasets. We demonstrate the effectiveness of VLMs for this task in Appendix .3.1 and Appendix .7.

Every VLM prompt consists of two components:

- **Image Set:** This includes all manual pages and the real-world *pre-assembly scene image*. Unlike traditional VLM applications in robotics [18, 23], which process a single image, our method requires multi-image reasoning.
- **Text Instructions:** These instructions provide a task-specific context, guiding the model in interpreting the image set. The instructions range from simple directives to Chain-of-Thought reasoning [51]. All instructions incorporate in-context learning examples, specifying the required output format—be it JSON, Python code, or natural language. This structure is essential to our multi-stage pipeline, ensuring well-structured, interpretable outputs that seamlessly integrate into subsequent stages.

.2.3. Stage I: Associating Real Parts with Manuals

Given the manual's cover sketch of the assembled furniture and the *pre-assembly scene image*, the VLM aims to associate physical parts with the manual. The VLM achieves this by predicting the roles of each physical part through semantically interpreting the manual's illustrations. This process involves analyzing spatial, contextual, and functional cues in the manual illustrations to enable a comprehensive understanding of each physical part. This design mimics human assembly cognition—people first map abstract manual images to physical parts before assembling. Our method follows CoT [51] and Least-to-Most [63] prompting, reducing cognitive load and improving accuracy. We considered pairwise matching of parts from manuals and scene images, but we found it impractical because the manuals do not depict each part independently.

To enhance part identification, we employ Set of Marks [55] and GroundingDINO [31] to automatically label parts on the *pre-assembly scene image* with numerical indices. The labeled scene image and manual sketch form the **Image Set**. **Text instructions** consist of a brief context explanation for the association task of predicting the roles of each physical part, accompanied by in-context examples of the output structure:

$$\{\textit{name, label, role}\}$$

For example, in Figure 2 In Stage I Output, we describe the chair's seat as *name: seat frame, label: [2], role: for people sitting on a chair, the seat offers essential support and comfort and is positioned centrally within the chair's frame.*. Here, *[2]* indicates that this triplet corresponds to the physical part labeled with index 2 in the pre-assembly scene image. This triplet format enhances interpretability and ensures consistency by structuring all outputs into the same data format. We use the Image Set and Text Instructions as the input prompt for the VLM (specifically GPT-4o [1]) and query it once to generate real assignments for all physical parts. We then use these labels as leaf nodes in the hierarchical assembly graph.

We can obtain equivalent parts through these triplets. When two physical parts share the same geometric shapes, their triplets only differ by label. For example, in Figure 2 Stage I Output, {*name: side frame, label: [0], role:...*} and {*name: side frame, label: [1], role:...*}—these two parts are considered equivalent. Understanding equivalent part relationships is crucial for downstream modules, as demonstrated by our ablation experiments(see Appendix .6).

.2.4. Stage II: Identify Involved Parts in Each Step

This stage focuses on identifying the particular parts and subassemblies involved in each manual page. The VLM achieves this by reasoning through the illustrated assembly steps, using the triplets and the labeled pre-assembly scene from the previous stage as supporting hints.

In practice, we observe that irrelevant elements in the manual (e.g., nails, human figures) can distract the VLM. Following [49], we manually crop the illustrated parts and subassemblies in each manual step to focus the VLM's at-

tention (Figure 2 Stage II Image Set), significantly improving performance (see Ablation Study for details). Automating Region-of-Interest (ROI) detection remains an open problem beyond the scope of this work and is left for future research.

The manual pages, combined with the labeled pre-assembly scene from the previous stage, form the **Image Set**. The **Text Instructions** use a Chain-of-Thought prompt to guide the VLM in identifying parts and subassemblies step by step and includes in-context examples that clarify the structured output format: a pair consisting of (Step N, Labeled Parts Involved). The bottom left output of Figure 2 provides an example of this format. Together, the **Image Set** and **Text Instructions** compose the input prompt for GPT-4o, which generates pairs for all assembly steps using a single query.

As shown in Fig. 2, the system outputs nested lists. We then transform these lists, along with the equivalent parts, into a hierarchical graph. Using this assembly graph, we traverse all non-leaf nodes and explore various assembly orders. Formally, a feasible assembly order is an ordered set of non-leaf nodes, ensuring that a parent node appears only after all its child nodes. A key advantage of the hierarchical graph representation is its flexibility—since the assembly sequence is not unique, it allows for parallel assembly or strategic sequencing.

.2.5. Per-step Assembly Pose Estimation

Given an assembly order, we train a model to estimate the poses of components (parts or subassemblies) at each step of the assembly process. At each step, the model inputs the manual image and the point clouds of the involved components, predicting their target poses to ensure proper alignment. To support this task, we construct a dataset for sequential pose estimation. For a detailed description, see Appendix .4.

Given each component's point cloud (obtained from real-world scans or our dataset), we first center it by translating its centroid to the origin. Next, we apply Principal Component Analysis (PCA) to identify the dominant object axes, which define a canonical coordinate frame. The most dominant axes serve as the reference frame, ensuring a shape-driven and consistent orientation that remains independent of arbitrary coordinate systems.

The dataset we create provides manual images, point clouds, and target poses for each component in the camera frame of the corresponding manual image(following [29]). For an assembly step depicted in the manual image $\mathcal{I}_i$, the inputs to our model include: (1) the manual image $\mathcal{I}_i$; (2) the point clouds of all involved components. The output is the target pose $T \in SE(3)$ for each component represented in the camera frame of I_i.

.2.6. Model Architecture

Note that the number of components at each step is not fixed, depending on the subassembly division of the furniture. Our pose estimation model consists of four parts: an image encoder $\mathcal{E}_I$, a point cloud encoder $\mathcal{E}_P$, a cross-modality fusion module $\mathcal{E}_G$, and a pose regressor $\mathcal{R}$.

We first feed the manual image I into the image encoder to get an image feature map $\mathbf{F}_I$.

$$\mathbf{F}_I = \mathcal{E}_I(I) \tag{1}$$

Then, we feed the point clouds into the point cloud encoder to get the point cloud feature for each component.

$$\{\mathbf{F}_j\} = \mathcal{E}_P(\{P\}_j) \tag{2}$$

In order to fuse the multi-modality information from the manual image and the point cloud features, we leverage a GNN [54] to update the information for each component. We consider the manual image feature and component-wise point cloud features as nodes in a complete graph, employing a GNN to update the information for each node.

$$\mathbf{F}'_I, \{\mathbf{F}'_j\} = \mathcal{E}_G(\mathbf{F}_I, \{\mathbf{F}_j\}) \tag{3}$$

where $\mathbf{F}'_I, \{\mathbf{F}'_j\}$ are updated image and point cloud features.

Finally, we feed the updated point cloud features as input into the pose regressor to get the target pose for each component.

$$T_j = \mathcal{R}(\mathbf{F}'_j) \tag{4}$$

.2.7. Loss Function

t jointly considers pose prediction accuracy and point cloud alignment, following [30, 60]. The first term penalizes errors in the predicted SE(3) transformation, while the second measures the distance between predicted and ground truth point clouds. To account for interchangeable components, we compute the loss across all possible permutations of equivalent parts and select the minimum loss as the final training objective. We provide further details on the loss formulation and training strategy in Appendix .5.

.2.8. Robot Assembly Action Generation

.2.9. Align Predicted Poses with the World Frame

At each assembly step, the previous stage predicts each component's pose in the camera frame of the manual image. However, real-world robotic systems operate in their world frame, requiring a 6D transformation between these coordinates. Consider two components, A and B. The predicted target poses in the camera frame are denoted as ${}^{I_i}\hat{\mathcal{T}}_a$ and ${}^{I_i}\hat{\mathcal{T}}_b$. Meanwhile, our system can collect the current 6D pose of part A in the world frame, represented as ${}^{W}\mathcal{T}_a$. To align ${}^{I_i}\hat{\mathcal{T}}_a$ to ${}^{W}\mathcal{T}_a$, we compute the 6D transformation matrix ${}^{W}_{I_i}\mathcal{T}$, which maps the camera frame to the world frame.

$${}^{W}\mathcal{T}_a = {}^{W}_{I_i}\mathcal{T}\,{}^{I_i}\hat{\mathcal{T}}_a \tag{5}$$

Using the same transformation ${}^{W}_{I_i}\mathcal{T}$, we compute the assembled target pose of part B (and all remaining components) in the world frame.

$$ {}^{W}\mathcal{T}_b = {}^{W}_{I_i}\mathcal{T}\, {}^{I_i}\hat{\mathcal{T}}_b \tag{6} $$

This transformation accurately maps predicted poses from the manual image frame to the robot's world frame, ensuring precise assembly execution.

.2.10. Assembly Execution

Once our system determines the target poses of each component in the world frame for the current assembly step, it grasps each component and generates the required action sequences for assembly.

Part Grasping After scanning each real-world part, we obtain the corresponding 3D meshes for each part. We employ FoundationPose [52], and the Segment Anything Model (SAM) [24] to obtain the initial poses of all parts in the scene.

Given the pose and shape of each part, we design heuristic grasping methods tailored to the geometry of individual components. While general grasping algorithms such as GraspNet [11] are viable, grasping is beyond the scope of this work. Instead, we employ heuristic grasping strategies specifically designed for structured components in assembly tasks. For stick-shaped components, we grasp the centroid of the object after identifying its longest axis for stability. For flat and thin-shaped components, we use fixtures or staging platforms to securely position the object, allowing the robot to grasp along the thin boundary for improved stability. We provide further details on these grasping methods in Appendix .10.

Part Assembly Trajectory Once the robot arm grasps a component, it finds a feasible, collision-free path to predefined robot poses (anchor poses). At these poses, the 6D pose of the grasped component is recalculated in the world frame, leveraging the FoundationPose [52] and the Segment Anything Model (SAM)[24]. The system then plans a collision-free trajectory to the component's target pose. We use RRT-Connect [26] as our motion planning algorithm. All collision objects in the scene are represented as point clouds and fed into the planner. Once the planner finds a collision-free path, the robot moves along the planned trajectory.

Assembly Insertion Policy Once the robot arm moves a component near its target pose, the assembly insertion process begins. Assembly insertions are contact-rich tasks that require multi-modal sensing (e.g., force sensors and closed-loop control) to ensure precise alignment and secure connections. However, developing closed-loop assembly insertion skills is beyond the scope of this work and will be addressed in future research. In our current approach, human experts manually perform the insertion action.

.3. Detailed Experiments Description

In this section, we perform a series of experiments aimed at addressing the following questions.

- Q1: Can our proposed hierarchical assembly graph generation module effectively extract structured information from manuals? (see Appendix .3.1)
- Q2: Can the per-step pose estimation be applicable to different categories of furniture and outperform previous settings? (see Appendix .3.2)
- Q3: How effective is the proposed framework in the assembly of furniture with manual guidance? (see Appendix .3.3)
- Q4: Can this pipeline be applied to real-world scenarios?(see Appendix .3.4)
- Q5: Can this pipeline be extended to other assembly tasks? (see Section 4.5)
- Q6: How should we determine and evaluate the key design choices of each module? (ablation experiments, see Appendices .6 and .8)

In addition, we have included a comprehensive set of prompts utilized in the VLM-guided hierarchical graph generation process in Appendix .14

.3.1. Hierarchical Assembly Graph Generation

In this section, we evaluate the performance of our VLM-guided hierarchical assembly graph generation approach. Specifically, we assess Stage II: Identifying Parts in Each Image using the IKEA-Manuals dataset [49]. We provide the rationale for excluding Stage I evaluation in Appendix .11.

Table 1. **Assembly Plan Generation Results**.

Method	Precision	Recall	F1 Score	Success Rate
SingleStep	0.220	0.220	0.220	0.220
GeoCluster	0.197	0.201	0.196	0.080
Ours	**0.690**	**0.680**	**0.684**	**0.620**

Experiment Setup. The IKEA-Manuals dataset [49] includes 102 furniture items, each with IKEA manuals, 3D parts, and assembly plans represented as trees in nested lists. We load each item's 3D parts into Blender and render an image of the pre-assembly scene. Moreover, we split the 102 furniture items into two sets. The first set consists of 50 furniture items with six or fewer parts, and the second set contains 52 furniture items with seven or more parts. We observe that current VLMs can effectively deal with the first set, and a significant portion of real-world furniture

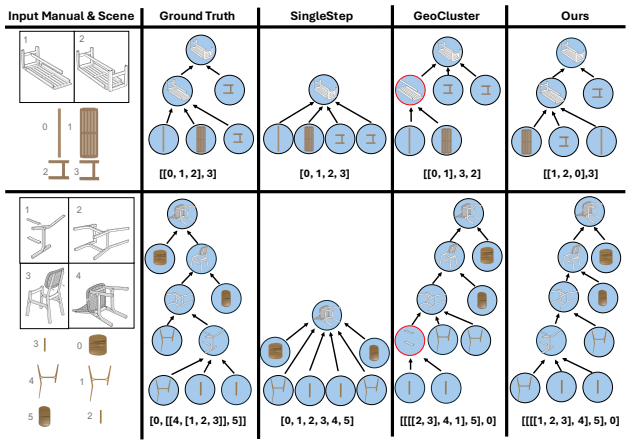


Figure 5. **Qualitative results.** Our method significantly outperforms the baselines. SingleStep fails on moderately complex furniture, while GeoCluster generates physically impossible subassemblies (highlighted in red). In contrast, our approach closely aligns with the ground truth.

also contains fewer than seven parts (as seen in real-world experiments). Here, we report the results of the first set. Please refer to Appendices .7 and .14 for complete results and prompts. This rendered image, along with the manual, is processed by the VLM through the stages outlined in Appendix .2.1 to generate a hierarchical assembly graph. Since we represent our graph as a nested list, we align our notation with the assembly tree notation used in IKEA-Manuals [49]. In this subsection, we refer to our generated assembly graph as the *predicted tree*.

Evaluation Metrics. We use the same metrics as IKEA-Manuals [49], which include precision, recall, and F1 score to compare predicted and ground-truth nodes of the assembly tree. For detailed descriptions of these metrics, we refer readers to [49].

The *Matching* criterion for each node is defined as follows: We consider a predicted non-leaf node correct only if its set of leaf and non-leaf child nodes exactly matches that of the corresponding ground-truth node(With consideration of equivalent parts). In other words, the predicted node must have the same children as its ground-truth counterpart. We compute precision, recall, and F1 scores based on this criterion.

The *Success Rate* criterion measures the proportion of the predicted tree that exactly matches the ground-truth tree. We consider a predicted tree exactly matched if all its non-leaf nodes satisfy the *Matching* criterion.

Baselines. We compare our VLM-based method against two heuristic approaches introduced in IKEA-Manuals [49].

- SingleStep predicts a flat, one-level tree with a single parent node and n leaf nodes.
- GeoCluster employs a pre-trained DGCNN [50] to iteratively group furniture parts with similar geometric features into a single assembly step. Compared to SingleStep, it generates deeper trees with more parent nodes and multiple hierarchical levels.

Results. As shown in Table 1, quantitative results demonstrate that both baseline methods face challenges in generating accurate assembly trees under the Matching and Assembly criterion. In contrast, our VLM-guided method achieves significantly superior performance, with a success rate of **62%**. These findings underscore the robust generalization capabilities when guided by well-structured prompts. Figure 5 provides qualitative results for two furniture items, illustrating the advantages of our approach in greater detail. With the ongoing development of more advanced VLMs, we expect further enhancements in assembly planning accuracy. Please refer to Appendix .8 for ablation results.

.3.2. Per-step Assembly Pose Estimation

Data Preparation. We select three categories of furniture items from PartNet [34]: chair, table, and lamp. For each category, we select 100 furniture items and generate 10 parts selection and subassembly division for each piece of furniture. To generate the assembly manual images, we render diagrammatic images of parts at 20 random camera poses using Blender's Freestyle functionality. We provide more details about it in Appendix .4. In general, we generate 12,000 training and 5,200 testing data pieces for each category.

Training Details. For the Image Encoder $\mathcal{E}_I$, we selected the encoder component of DeepLabV3+, which includes MobileNet V2 as the backbone and the atrous spatial pyramid pooling (ASPP) module. We made this choice because DeepLabV3+ leverages atrous convolutions on the basis of Auto Encoder, enabling the model to capture multi-scale structures and spatial information effectively [4, 5]. It generates a multi-channel feature map from the image I, and we use mean-max pool [61] to derive a global vector $\mathbf{F}_I \in \mathbb{R}^{256}$ from the feature map. For the Point Clouds Encoder $\mathcal{E}_P$, we use the encoder part of PointNet++ [35]. For each part and subassembly, we extract a part-wise feature $\mathbf{F}_j \in \mathbb{R}^{256}$. For the GNN $\mathcal{E}_G$, we use a three-layer graph transformer [8]. The pose regressor $\mathcal{R}$ is a three-layer MLP. We provide more details of the mean-max pool for the image feature and our training hyperparameter setting in Appendix .5.

Baselines. We evaluate the performance of our method on our proposed per-step assembly pose estimation dataset. We compare our method with two baselines:

- Li et al. [29] proposed a pipeline for single image guided 3D object pose estimation.
- Mean-Max Pool is a variant of our method, replacing GNN with a mean-max pool trick, similar to our ap-

Table 2. **Qualitative Results of Pose Estimation.**

	GD↓			RMSE↓			CD ↓			PA↑		
Method	Chair	Lamp	Table	Chair	Lamp	Table	Chair	Lamp	Table	Chair	Lamp	Table
Li et al. [29]	1.847	1.865	1.894	0.247	0.278	0.318	0.243	0.396	0.519	0.268	0.121	0.055
Mean-Max Pool	0.434	1.118	1.059	0.087	0.187	0.200	0.046	0.229	0.280	0.457	0.199	0.107
Ours	**0.202**	**0.826**	**0.953**	**0.042**	**0.153**	**0.172**	**0.027**	**0.189**	**0.276**	**0.868**	**0.240**	**0.184**

proach of obtaining a one-dimensional vector from a multi-channel feature map, with details in Appendix .5.

Evaluation Metrics. We adopt comprehensive evaluation metrics to assess the performance of our method and baselines.

- Geodesic Distance (GD), which measures the shortest path distance on the unit sphere between the predicted and ground-truth rotations.
- Root Mean Squared Error (RMSE), which measures the Euclidean distance between the predicted and ground-truth poses.
- Chamfer Distance (CD), which calculates the holistic distance between the predicted and the ground-truth point clouds.
- Part Accuracy (PA), which computes the Chamfer Distance between the predicted and the ground truth point clouds; if the distance is smaller than 0.01m, we count this part as "correctly placed".

Results. As shown in Table 2, our method outperforms Li et al. [29] and the mean-max pool variant in all evaluation metrics and on three furniture categories. We attribute this to the effectiveness of our multi-modal feature fusion and GNN in capturing the spatial relationships between parts. We also provide qualitative results for each furniture category in Figure 6.

Ablation. To assess the impact of equivalent parts, guided image, and per-step data about subassemblies, we perform ablation studies on these components. We present the details and results in Appendix .6.

.3.3. Overall Performance Evaluation

We evaluate the overall performance of our method by assembling furniture models in a simulation environment. We implement the evaluation process in the PyBullet [9] simulation environment and test the entire pipeline. We source all test furniture models from the IKEA-Manuals dataset [49]. Given these manuals along with 3D parts, we generate the pre-assembly scene images as described in .2.8, and our pipeline generates the hierarchical graphs. Then, we traverse the hierarchical graph to determine the assembly order. Following this sequence and the predicted 6D poses of each component, we implement RRT-Connect [26] in simulation to plan feasible motion paths for the 3D parts

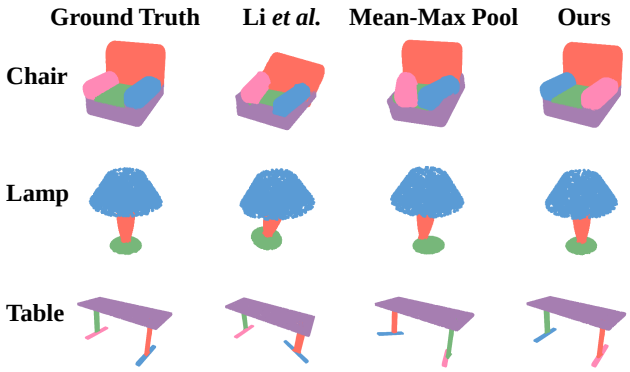


Figure 6. **Qualitative results on three furniture categories.** We observe better pose predictions than baselines.

and subassemblies, ensuring they move towards their target poses. Note that, in this experiment, we focus on object-centric motion planning and omit robotic execution in our framework.

Baselines. As the first to propose a comprehensive pipeline for furniture assembly, there is no direct baseline for comparison. So we design a baseline method that uses previous work [29] to estimate the poses of all parts, with the guidance of an image of the fully assembled furniture, and adopt a heuristic order to assemble all parts. Specifically, given the predicted poses of all parts, we can calculate the distance between each pair of parts. The heuristic order is defined as follows: starting from a random part, we find the nearest part to it and assemble it, then successively find the nearest part to the assembled parts until we assemble all parts.

Evaluation Metrics. We adopt the assembly success rate as the evaluation metric and define the following situations as a failure: 1) A part is placed at a pose that is too far from the ground truth pose. 2) A part collides with other parts when moving to the estimated pose. In other words, the RRT-Connect algorithm [26] finds no feasible path when mating it with other parts. 3) We place a part that is not near any other components, causing it to suspend in midair after each assembly step.

Results. We evaluate the overall performance on 50 fur-

Table 3. **Success Rate on 4 Furniture Categories**(↑)

Method	Bench	Chair	Table	Misc	Average
Li et al. [29]+Heuristic	0.00	0.39	0.11	0.00	0.30
Ours	**0.67**	**0.61**	**0.44**	**0.50**	**0.58**

niture items from the IKEA-Manual dataset [49], each consisting of fewer than seven parts. These items fall into four categories (Bench, Chair, Table, Misc), and we report the success rate for each in Table 3.

Our system successfully assembles 29 out of 50 furniture pieces, whereas the baseline method assembles only 15. Our framework achieves a success rate of **58%**, demonstrating the effectiveness of our proposed framework. The most common failure occurs when the VLM fails to generate a fully accurate assembly graph, leading to misalignment between the point cloud and the instruction manual images used for pose estimation.

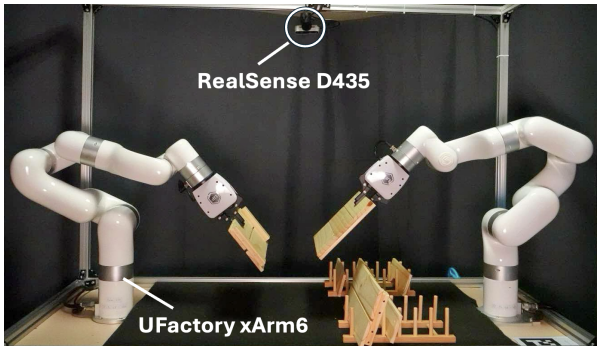


Figure 7. **Real-World Setup.** We use two UFactory xArm6 for assembly and a RealSense D435 camera for pose estimation.

.3.4. Real-world Assembly Experiments

To evaluate the feasibility and performance of our pipeline, we conducted experiments in the real world using four IKEA furniture items: Flisat (Wooden Stool), Variera (Iron Shelf), Sundvik (Chair), and Knagglig (Box). Figure 7 illustrates our real-world experiment setup. We show the manual images, per-step pose estimation results, and real-world assembly process in Figure 3. We also attach videos of the real-world assembly process in the supplementary material. For detailed implementation of our real-world experiments, please check Appendix .10. We evaluated all the assembly tasks with target poses provided by three different methods: Ground truth Pose, Mean-Max Pool (see Appendix .3.2), and our proposed approach. The Ground truth Pose method uses the ground truth poses for each part to assemble the furniture. We use the Average Completion Rate (ACR) as the evaluation criterion and calculate it as follows:

$$ACR = \frac{1}{N}\sum_{j=1}^{N}\frac{S_j}{S_{\text{total}}} \tag{7}$$

where N is the total number of trials, S_j is the number of steps completed in trial j, and S_{total} denotes the total number of steps in the task.

We perform each task over 10 trials with varying initial 3D part poses. We present the results in Table 4, showing that our method outperforms the baseline and achieves a high success rate in real-world assembly tasks.

These findings underscore the practicality and effectiveness of our approach for real-world implementation. The primary failure mode arises from planning limitations, particularly in handling complex obstacles. Failures occur when the RRT-Connect algorithm cannot find a feasible trajectory when the planned path results in collisions with the robotic arm or surrounding objects or due to suboptimal grasping poses. To improve robustness in real-world scenarios, we plan to develop a low-level policy for adaptive motion refinements—a topic we leave for future work.

Table 4. **Real World Success Rate** (↑) **over 10 trials.**

Method	FLISAT	VARIERA	SUNDVIK	KNAGGLIG
Oracle Pose	72.5	85.0	80.0	90.0
Mean-Max Pool	52.5	61.7	40.0	70.0
Ours	60.0	80.0	68.0	85.0

We design Manual2Skill as a generalizable framework capable of handling diverse assembly tasks with manual instructions. To assess its versatility, we evaluate the VLM-guided hierarchical graph generation method across three distinct assembly tasks, each varying in complexity and application domain. These include: **(1) Assembling a Toy Car Axle** (a low-complexity task with standardized components, representing consumer product assembly), **(2) Assembling an Aircraft Model** (a medium-complexity task, representing consumer product assembly), and **(3) Assembling a Robotic Arm** (a high-complexity task involving non-standardized components, representing research & prototyping assembly).

For the toy car axle and aircraft model, we sourced 3D parts from [46] and reconstructed pre-assembly scene images using Blender. We manually crafted the manuals in their signature style, with each page depicting a single assembly step through abstract illustrations. For the robotic arm assembly, we used the Zortrax robotic arm [66], which includes pre-existing 3D parts and a structured manual. These inputs were then processed through the VLM-guided hierarchical graph generation pipeline (described in Sec. .3.1), yielding assembly graphs as shown in Figure 4. This **zero-shot** generalization achieves a success rate of

100% over five trials per task. The generated graphs align with ground-truth assembly sequences, confirming the generalization of our VLM-guided hierarchical graph generation across diverse manual-based assembly tasks and highlighting its potential for broader applications.

.4. Per-step Assembly Pose Estimation Dataset

We build a dataset for our proposed manual guided per-step assembly pose estimation task. Each data piece is a tuple $(I_i, \{P\}_j, \{T\}_j, \mathbf{R}_i)$, where I_i is the manual image, $\{P\}_j$ is the point clouds of all the components involved in the assembly step, $\{T\}_j$ is the target poses for each component, and $\mathbf{R}_i$ is the spatial and geometric relationship between components.

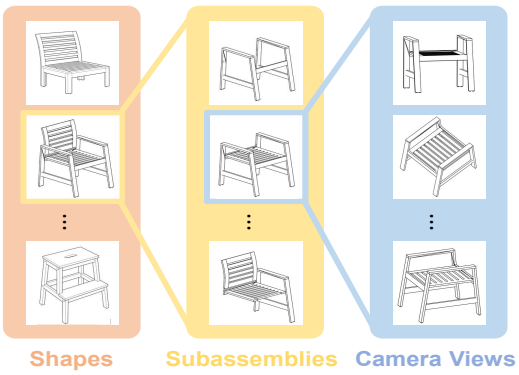


Figure 8. Manual images of our proposed dataset. There are variations in furniture shapes, subassemblies, and camera views.

Instruction manuals in the real world come in a wide variety. To cover as many scenarios as we might encounter in real-life situations, we considered three possible variations of instruction manuals when constructing the dataset, as shown in Figure 8. Our dataset encompasses a variety of furniture shapes. For each piece of furniture, we randomly selected some connected parts to form different subassemblies. Meanwhile, for each subassembly, there are multiple possible camera perspectives for taking manual photos. This definition enables our dataset to cover various manuals that we might encounter in real-world scenarios.

Formally, for furniture consisting of M parts, we randomly select m connected parts to form a subassembly. Denoted as $P_{\text{sub}} = \{P_1, P_2, \cdots, P_m\}$, here each P_i is a atomic part. Then, we randomly group the m atomic parts into n components while keeping all parts within the same group are connected, denoted as $P_{\text{sub}} = \{\{P_{11}, \cdots P_{1\alpha_1}\}, \cdots \{P_{n1}, \cdots P_{n\alpha_n}\}\}$, where each α_i represents the number of atomic parts in i-th component, and thus $\sum_i \alpha_i = m$. We sample the point cloud for each component to consist of the point cloud of the data piece. We can also take photos of the subassembly from different perspectives.

We also provide annotations for equivalent parts in the auxiliary information. In this paper, we propose new techniques to leverage the auxiliary information for each assembly step, which significantly enhances the precision and robustness of our pose estimation model.

.5. Pose Estimation Implementation

.5.1. Loss Functions for Pose Estimation

Rotation Geodesic Loss: In 3D pose prediction tasks, we commonly use the rotation geodesic loss to measure the distance between two rotations [53]. Formally, given the ground truth rotation matrix $R \in SO(3)$ and the predicted rotation $\hat{R} \in SO(3)$, the rotation geodesic loss is defined as:

$$\mathcal{L}_{\text{rot}} = \arccos\left(\frac{\text{tr}(R^T\hat{R}) - 1}{2}\right) \tag{8}$$

where $\text{tr}(\cdot)$ denotes the trace of a matrix and R^T is the transpose of R.

Translation MSE Loss: Following [29], we use the mean squared error (MSE) loss to measure the distance between the ground truth translation t and the predicted translation $\hat{t}$:

$$\mathcal{L}_{\text{trans}} = ||t - \hat{t}||_2 \tag{9}$$

Chamfer Distance Loss: This loss function minimizes the holistic distance between each point in the predicted and ground truth point clouds. Given the ground truth point cloud $S_1 = RP + t$ and the predicted point cloud $S_2 = \hat{R}P + \hat{t}$, it is defined as:

$$\mathcal{L}_{\text{cham}} = \frac{1}{|S_1|}\sum_{x \in S_1} \min_{y \in S_2} ||x-y||_2^2 + \frac{1}{|S_2|}\sum_{x \in S_2} \min_{y \in S_1} ||y-x||_2^2 \tag{10}$$

where S_1 is the point cloud after applying the ground truth 6D pose transformation, and S_2 is the point cloud after applying the predicted 6D pose transformation.

Pointcloud MSE Loss: We supervise the predicted rotation by applying it to the point of the component and use the MSE loss to measure the distance between the rotated point and the ground truth point:

$$\mathcal{L}_{\text{pc}} = ||RP - \hat{R}P||_2 \tag{11}$$

Equivalent Parts: Given a set of components, we might encounter geometrically equivalent parts that we must assemble in different locations. Inspired by [60], we group these geometrically equivalent components and add an extra loss term to ensure we assemble them in different locations. For each group of equivalent components, we apply the predicted transformation to the point cloud of each component and then compute the Chamfer distance (CD) between the transformed point clouds. For all pairs (j_1, j_2) within the

same group, we compute the Chamfer distance between the transformed point clouds $\hat{P}_{j_1}$ and $\hat{P}_{j_2}$, encouraging the distance to be large:

$$\mathcal{L}_{\text{equiv}} = -\sum_{\text{group}} \sum_{(j_1,j_2)} \text{CD}(\hat{P}_{j_1}, \hat{P}_{j_2}) \tag{12}$$

Finally, we define the overall loss function as a weighted sum of the above loss terms:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{rot}} + \lambda_2 \mathcal{L}_{\text{trans}} + \lambda_3 \mathcal{L}_{\text{cham}} + \lambda_4 \mathcal{L}_{\text{pc}} + \lambda_5 \mathcal{L}_{\text{equiv}} \tag{13}$$

where $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 1$, $\lambda_4 = 20$, $\lambda_5 = 0.1$.

.5.2. Mean-Max Pool

The core mechanic of the mean-max pool is to obtain the mean and maximum values along one dimension $\mathbb{R}^C$ of a set of vectors or matrices with the same dimensions and concatenate them into a one-dimensional vector in $\mathbb{R}^{2C}$ to obtain a global feature. For one-dimensional vectors, we take the mean and maximum values along the sequence length dimension. For two-dimensional matrices, we take the mean and maximum values along the height $\times$ width dimensions:

$$\mathbf{F}_{global} = [\mathbf{avg}; \mathbf{max}] \in \mathbb{R}^{2F} \tag{14}$$

In the setting of our work, we set F to 128.

We use this trick twice in this work. One instance is when we obtain a one-dimensional vector with a channel dimension from a multi-channel feature map, thus obtaining a one-dimensional feature vector for the image. In this case, we can express the mean-max pool as follows:

$$\begin{cases} \mathbf{X} = (\mathbf{X}_{c,h,w})_{c=1,h=1,w=1}^{C,H,W} \\ \mathbf{avg} = (\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{X}_{c,h,w})_{c=1}^{C} \in \mathbb{R}^C \\ \mathbf{max} = (\max_{h,w} \mathbf{X}_{c,h,w})_{c=1}^{C} \in \mathbb{R}^C \end{cases} \tag{15}$$

Where $\mathbf{X}$ is the multi-channel feature map of image I_i with dimensions channels(C) $\times$ height(H) $\times$ width(W), $\mathbf{avg}$ and $\mathbf{max}$ denote one-dimensional vectors of length channels. Thus, $\mathbf{F}_{global}$ of the multi-channel feature map is a C-dimensional vector.

The other instance is when we compare the baseline. To aggregate point cloud features on a per-part basis and obtain a one-dimensional global feature for the shape, we express the mean-max pool in the following form:

$$\begin{cases} \mathbf{avg} = \frac{1}{M} \sum_{j=1}^{M} \mathbf{F}_j \in \mathbb{R}^F \\ \mathbf{max} = \max_{F} \{\mathbf{F}_j\} \in \mathbb{R}^F \end{cases} \tag{16}$$

Here, we let M denote the number of parts in a shape. For each part in this baseline, we concatenate the one-dimensional image feature $\mathbf{F}_I$, the global point cloud feature $\mathbf{F}_{global}$ (both obtained by mean-max pool), and the part-wise point cloud feature $\mathbf{F}_j$ to form a one-dimensional cross-modality feature. We then use this feature as input for the pose regressor MLP.

.5.3. Hyperparameters in Training of Pose Estimation

We train our pose estimation model on a single NVIDIA A100 40GB GPU with a batch size of 32. Each experiment runs for 800 epochs (approximately 46 hours). We set the learning rate to $1e-5$ and employ a 10-epoch linear warm-up phase. Afterward, we use a cosine annealing schedule to decay the learning rate. We also set the weight decay to $1e-7$. The optimizer configuration for each component of the model is as shown in Table 5.

Table 5. Optimizer Corresponding to Each Component

Component	Optimizer
Image Encoder	RMSprop
Pointcloud Encoder	AdamW
GNN	AdamW
Pose Regressor	RMSprop

.6. Pose Estimation Ablation Studies

To evaluate the effectiveness of each component in our pipeline, we conduct an ablation study on the chair category. We show the quantitative results in Table 6 and the qualitative results in Figure 9. First, we remove the image input and only use the point cloud input to predict the pose. The performance drops significantly, indicating that the image input is crucial for pose estimation. Second, we remove the permutation mechanism for equivalent parts(Equation (12)). As shown in the visualizations, the model fails to distinguish between equivalent parts, placing two legs in similar positions.

Table 6. **Pose Estimation Ablations.**

Method	GD↓	RMSE↓	CD↓	PA↑
w/o Image	1.797	0.234	0.227	0.138
w/o Permutations	0.252	0.051	0.029	0.783
Ours	**0.202**	**0.042**	**0.027**	**0.868**

Previous works usually train and predict only fully assembled shapes. In contrast, our pose estimation dataset includes per-step data ($i.e.$, subassemblies). We conduct an ablation study comparing two settings:

- *w/o Per-step*: Training and testing on a dataset of fully assembled shapes.

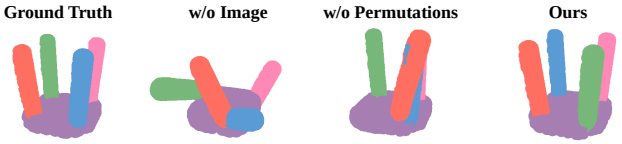


Figure 9. **Qualitative Results of Ablations.** We observe salient performance drops in ablated settings.

- *Per-step*: Training on a dataset with per-step data and testing on fully assembled shapes.

Table 7. **w/o Per-step vs. Per-step**

Method	GD↓	RMSE↓	CD↓	PA↑
w/o Per-step	0.233	0.046	0.015	0.753
Per-step **(Ours)**	**0.064**	**0.016**	**0.004**	**0.983**

As shown in Table 7, adding per-step data improves assembly prediction accuracy, demonstrating that per-step inference enhances robot assembly performance.

.7. Complete VLM Plan Generation Results

We provide the complete analysis for VLM plan generation. In addition to the results for all 50 furniture items with six or fewer parts, shown in the main paper, we include results for all 52 furniture items with seven or more parts (denoted as ≤ 7 Parts) and the complete dataset of 102 furniture items spanning all part counts (denoted as All Parts) in Table 8. Furthermore, we categorized the full set of 102 furniture items in greater detail, with Hard Matching results for individual part counts ranging from 2 to 16 parts, as shown in Table 9. For detailed descriptions of Simple Matching and Hard Matching, we refer readers to [49].

For the GeoCluster baseline, we could not replicate the exact results shown in the IKEA-Manuals dataset [49]. Thus, we used the scores from our experiments for the ≤ 6 Parts and ≥ 7 Parts categories while retaining the original scores from the dataset [49] for the All Parts category.

To obtain our scores, we repeatedly ran the experiment 5 times using the same input and a temperature of 0. We repeated sampling to account for slight variations in GPT-4o's [1] outputs, even when we set the temperature to 0, and to capture the range of possible outcomes. This approach provides a better estimate of the model's true performance. When taking the maximum between precision, recall, and F1, the average score for ≤ 6 parts on Hard Matching is 63.7%, the worst score is 57.2%, and the best score is 69.0%. Since the average and best scores are similar, we choose to report the best score in all of our tables related to Assembly Plan Generation.

To compare the trees generated by GPT-4o [1] with the ground truth trees in the dataset, we accounted for equivalence relationships among parts, which can result in multiple valid ground truth trees. For instance, if parts 1 and 2 are equivalent and [[1, 3], 2] is a valid tree, then so is [[2, 3], 1]. Since the dataset does not account for this isomorphism of trees, we manually defined all equivalent parts for each of the 102 furniture items. We then permuted the predicted tree using the equivalent parts, comparing each permutation to the ground truth and selecting the highest score. For furniture with 13 or more parts (6 items), we performed manual verification due to the computational cost of permutations. Overall, by employing this permutation method to evaluate predicted trees, we managed to increase our scores overall metrics by around 5%. To ensure fairness, we also applied this permutation over the two baselines but saw no effects.

As shown in Table 8, tasks with ≥ 7 parts experience a significant drop in performance—Hard Matching achieves a maximum of 13.36%, compared to 69.0% for tasks with ≤ 6 parts—indicating that the model's performance declines as the number of parts increases. This decrease is likely driven by increased task complexity and occlusion in manual drawings as the number of furniture parts grows, causing GPT-4o [1] to misinterpret out-of-distribution images and fail in the plan generation stage. As noted in [49], SingleStep always outputs the root node and selects all other nodes as its children, achieving perfect precision in Simple Matching for all cases. Beyond this, our GPT-4o-based method outperforms both baselines across all categories in Table 8, which highlights the effectiveness of VLMs in interpreting manuals and designing reliable hierarchical assembly graphs.

Similarly, in Table 9, our method has a significant advantage over the two baselines in all numbers of parts. Mask Seg is an additional method we evaluated, which overlays segmentation masks from the IKEA-Manuals dataset [49] onto manual pages (prompt 3.a Appendix .14), improving part identification, image clarity, and comprehension of assembly steps. Although Mask Seg slightly outperforms the original version without mask segmentations, we chose the latter for all reported tables. Otherwise, such masks are costly in real-world scenarios. Overall, the trend observed in Table 8 persists here, with higher scores for furniture with fewer parts and lower scores as the number of parts increases.

.8. Assembly Graph Generation Ablation Studies

We present the effectiveness of our VLM plan generation pipeline, emphasizing the critical role of cropped manual pages as input. The manual pages' visuals, detailing parts and subassemblies for each step, directly influence GPT-4o's output. Thus, we prioritize this content and ablate the strategy of inputting cropped pages. For furniture requiring N assembly steps, instead of providing N cropped manual pages corresponding to each step, we input the entire man-

Table 8. VLM Assembly Plan Generation Results

Method	Simple Matching (All Parts)			Hard Matching (All Parts)			Simple Matching ($\geq$ 7 Parts)			Hard Matching ($\geq$ 7 Parts)		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SingleStep	100.00	35.77	48.64	10.78	10.78	10.78	100.00	21.96	35.09	0.00	0.00	0.00
GeoCluster	44.90	48.46	43.53	16.54	16.50	16.30	31.99	28.88	29.66	7.31	6.91	6.92
Ours	**58.11**	**55.98**	**56.84**	**40.63**	**39.94**	**40.22**	**33.72**	**31.95**	**32.65**	**13.36**	**12.96**	**13.11**

Table 9. Performance Across Different Numbers of Parts

Number of Parts	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
SingleStep	100	50	12.50	31.58	0	0	0	0	0	0	0	0	0	0	0
GeoCluster	100	25	10.42	14.04	21.76	14.40	6.99	15.00	4.17	2.22	0	16.67	0	0	0
Ours (Mask Seg)	100	100	75.00	72.81	56.08	29.64	24.17	19.05	16.67	9.63	3.33	37.50	0.00	0.00	0.00
Ours	**100**	**100**	**72.92**	**78.51**	**45.59**	**25.24**	**13.05**	**16.67**	**27.78**	**0**	**9.33**	**6.25**	**0.00**	**0.00**	**0.00**
Furniture Count	2	4	8	19	17	14	10	3	4	9	5	2	1	2	1

ual consisting of $M \geq N$ pages. As shown in Table 10, this "no-crop" method leads to 7% accuracy drops in the Simple Matching category and 25% in the more important Hard Matching category. The decrease is likely due to irrelevant details in full manual pages, such as the nails, people, and speech bubbles in prompt 2.a), which divert GPT-4o's focus from the critical furniture parts for each step. Overall, Table 10 underscores the importance of cropping manual pages to simplify the input and direct GPT-4o's attention to the most relevant details.

Table 10. Assembly Plan Generation Ablation Results on Furniture with $\leq$ 6 Parts

Method	Simple Matching			Hard Matching		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Ours (no crop)	69.13	81.13	73.05	42.37	45.50	43.45
Ours	**83.47**	**80.97**	**81.99**	**69.00**	**68.00**	**68.41**

.9. Failure Cases Analysis

We highlight failure cases of VLMs using GPT-4o in Figure 10 for plan generation of complex furniture. Figure 10 demonstrates that while GPT-4o surpasses previous baselines in assembly planning, it struggles with complex structures, often producing entirely incorrect results.

.10. Real-World Experiment Details

This section provides the details of the real-world experiment.

.10.1. Pose Estimation in the Real World

We utilize FoundationPose [52] to evaluate the 6D pose and point cloud of components in the real-world scene. First, a mobile app, ARCode, is used to scan the mesh of all atomic parts of the furniture. During each step of the assembly process, the mesh—along with the RGB and depth images and an object mask—is input into the FoundationPose model, which then generates the precise 6D pose and point cloud of the component within the scene. This information is crucial for subsequent tasks, including camera pose alignment, grasping, and collision-free planning.

.10.2. Camera Frame Alignment

After we get the estimated target pose, we first use the PCA mentioned before to canonize them. To accurately map these target poses to the real world, we need to align the camera frame in the manual page image, denoted as P_{m_i}, with the real-world camera frame, denoted as P_{w_i}, for each step i. This section will introduce how we calculate the 6D transformation matrix T_{mw} between these two frames.

To achieve this, we designate a stable part of the scene as a base in the world frame using the VLM and utilize FoundationPose to extract the point cloud of this part. We then canonicalize the point cloud using the same PCA algorithm, ensuring that the relative 6D pose of the same component remains consistent. We denote the canonical base pose in the real world as P_{B_w}, which remains static during this step. From the model's predictions, we can also determine the pose of the same part used as the base in the manual, denoted as P_{B_m}. We denote the transformation matrix between these two frames as T_{mw}. Using this transformation

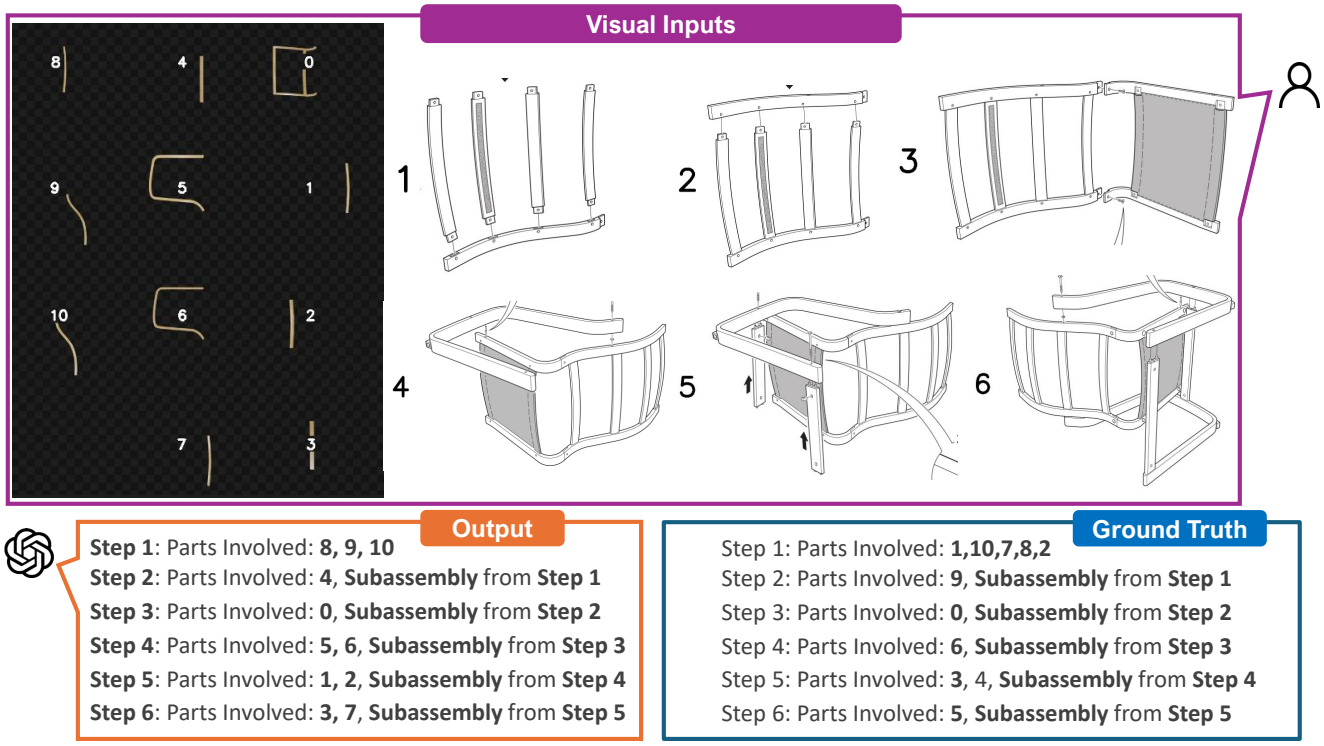


Figure 10. The input consists of the scene image, the corresponding assembly steps from the manual, and the text instruction from prompt 3.b). Clearly, GPT-4o's response is wrong and unreliable.

matrix, we map the target pose in the manual frame, P_{T_m}, to the corresponding target pose in the real-world frame, P_{T_w}, for subsequent motion planning. We compute the transformation as follows:

$$T_{mw} = P_{B_w} P_{B_m}^{-1}$$

We then calculate the target pose in the real-world frame using:

$$P_{T_w} = T_{mw} P_{T_m}$$

As illustrated in Figure 11, the stool example clearly demonstrates the process of aligning poses between the manual and real-world frames, ensuring a consistent and reliable foundation for motion planning.

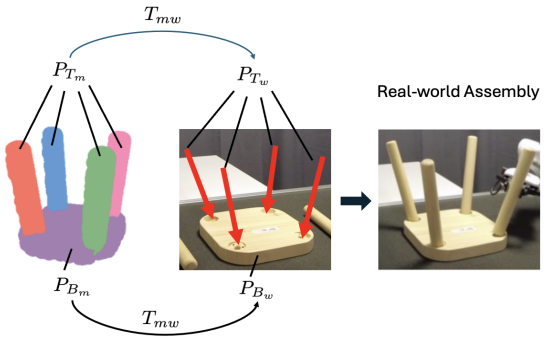


Figure 11. This figure shows the transformation between the estimated pose and the real-world frame; we designate the board of the stool as a base and map the four legs of the stool to the real world

.10.3. Heuristic Grasping Policy

For general grasping tasks, pre-trained models such as GraspNet[11] are commonly used to generate grasping poses. However, in the case of furniture assembly, where components are often large and flat, we need to grasp specific parts of the object that are suitable for subsequent assembly. This requirement poses challenges for GraspNet, as it does not always estimate the best pose for the subsequent action. To address this, in addition to GraspNet, we utilize the poses generated by FoundationPose and consider the shapes of the furniture components in corner cases. These shapes are categorized into two types, as shown in Figure 12:

Stick-Shaped Components: For stick-shaped furniture parts, such as stool legs, we select the center of the point cloud as the grasping position. We define the grasping pose as a top-down approach.

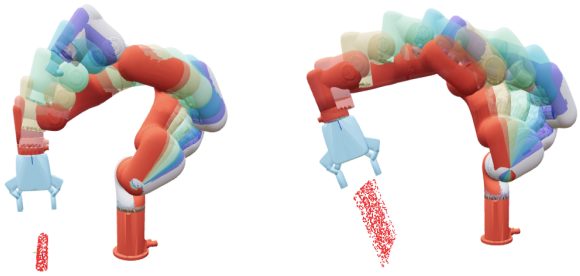


Figure 12. This figure shows the grasping policies for different shapes in our setting; the left one is for stick-shaped, and the right one is for flat, thin-shaped.

Flat and thin-Shaped Components: We first estimate the pose of flat and thin, board-shaped furniture parts using a bounding box. Based on this estimate, we determine the grasping pose by aligning it with the bounding box's orientation. The grasping position is set approximately 3 cm below the top surface.

.11. Rationale for Excluding Performance Evaluation of Stage I in Hierarchical Assembly Graph Generation

Stage I, Associating Real Parts with Manuals, focuses on associating real parts with the manual. Still, since the IKEA manual lacks isolated images of individual parts, direct quantitative evaluation is challenging. Instead, Stage II implicitly reflects the quality of these associations by outputting the indices of identified real parts. Therefore, we report Stage II results as an intermediate measure of how effectively our approach aligns manual images with real components.

.12. Justification for Hierarchical Assembly Graph

Using a hierarchical structure to represent assembly steps provides several advantages over simple linear data structures or unstructured step-by-step plans in plain text.

- Hierarchical structures align naturally with the assembly process where multiple parts and subassemblies combine into larger subassemblies.
- Lists or text plans struggle to store geometric and spatial relationships between each part or subassembly of the step, which is crucial in real assembly tasks.
- The hierarchical graph clearly shows the dependencies between steps, revealing which steps you can perform in parallel and which ones you must complete before proceeding to others. So, it provides flexibility for parallel construction or strategic sequencing.

.13. Formal Definition of Hierachial Assembly Graph

Inspired by Mo et al. [33], we represent the assembly process as a hierarchical graph $S = (\mathbf{P}, \mathbf{H}, \mathbf{R})$. A set of nodes $\mathbf{P}$ represents the parts or subassemblies in the assembly process. A structure $(\mathbf{H}, \mathbf{R})$ describes how these nodes are assembled and related to each other. The structure consists of two edge sets: $\mathbf{H}$ describes the *assembly relationship* between nodes, and $\mathbf{R}$ represents the *geometric* and *spatial* relationship between nodes.

Node. Each node $v \in \mathbf{P}$ is an atomic part or a subassembly, consisting of a non-empty subset of parts $p(v) \subset \mathcal{P}$. The root node v_N represents the fully assembled furniture, with $p(v_N) = \mathcal{P}$. A non-root, non-leaf node v_i represents a subassembly with $p(v_i)$ as a non-empty and proper subset of $\mathcal{P}$. All leaf nodes v_l represent atomic parts, containing exactly one element from $\mathcal{P}$. Additionally, each non-leaf node corresponds to a manual image I that describes how to merge smaller parts and subassemblies to form the node.

Assembly relationship. We formulate the assembly process as a tree, with all atomic parts serving as leaf nodes. The atomic parts are then recursively combined into subassemblies, forming non-leaf nodes until they reach the root node, which represents the fully assembled furniture. The directed edges from a child node to its parent node indicate the assembly relationship. The edge set $\mathbf{H}$ includes directed edges from a child node to its parent node, indicating the assembly relationship. For a non-leaf node v_i, denote its child nodes as C_i, the following properties hold:

(a) $\forall v_j \in C_i, p(v_j)$ is a non-empty subset of $\mathcal{P}$

(b) All children nodes contain distinct elements

$$p(v_j) \cap p(v_k) = \emptyset, \forall v_j, v_k \in C_i, j \neq k \tag{17}$$

(c) The union of all child subsets equals $p(v_i)$:

$$\bigcup_{v_j \in C_i} p(v_j) = p(v_i) \tag{18}$$

Equivalence relationship. In addition to the assembly process's hierarchical decomposition, we also consider the equivalence relationship between nodes. We label two parts *equivalent* if they share a similar shape and can be used interchangeably in the assembly process. We represent this relationship with undirected edges $\mathbf{R_i}$ in child nodes C_i of node v_i. An edge $\{v_a, v_b\} \in R_i$ appears between two nodes $v_a \in C_i$, $v_b \in \mathcal{P}$, if the shape represented by v_a and v_b are geometric equivalent and thus can be changed during assembly. Note that v_b is not constrained as a child of v_i since any two nodes could be equivalent, regardless of their hierarchical positions.

The assembly structure is a hierarchical graph, where the nodes represent parts or subassemblies, and the edges represent the assembly and equivalence relationships. We con-

sider this structured representation to be a more informative and interpretable way to formulate the assembly process than a flat list of parts.

.14. Prompts

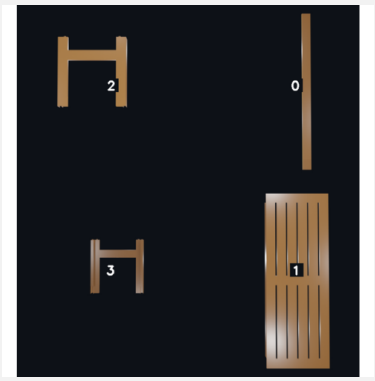
We offer a comprehensive set of prompts utilized in the VLM-guided hierarchical graph generation process. The process involves four distinct prompts, divided into two stages. The first two prompts, which are slight variations of each other, are used in *Stage I: Associating Real Parts with Manuals*. The remaining two prompts, also slight variations of each other, are employed in *Stage II: Identifying Involved Parts in Each Step*.

1. The first prompt is part of Stage I, and it initializes the JSON file's structure and consists of two sections:
   - 1.a): **Image Set:** An image of the scene with furniture parts labeled using GroundingDINO [31], alongside an image of the corresponding manual's front page.
   - 1.b) **Text Instructions:** A few sentences explaining the JSON file generation, supported by an example of the desired structure via in-context learning.

   This prompt is passed into GPT-4o to generate a JSON file with the name and label for each part.
2. The second prompt belongs in Stage I as well, and it populates the JSON file with detailed descriptions of roles. It includes:
   - 2.a): **Image Set:** Images of all manual pages (replacing the front page) to provide context about the function of each part and the scene image from the first prompt.
   - 2.b): **Text Instructions:** a simple text instruction explaining the context and output.

   We combine the JSON output from the first prompt with the second prompt, then query GPT-4o to generate the populated JSON file.
3. The third prompt is a part of Section II, and it generates a step-by-step assembly plan using:
   - 3.a): **Image Set:** The scene image and cropped manual pages highlight relevant parts and subassemblies, helping GPT-4o focus on key details. The cropped images also have a highlighted black number on the left, indicating the current assembly step. Our ablation studies demonstrate the effectiveness of these cropped images.
   - 3.b): **Text Instructions:** A text instruction combining chain-of-thought and in-context learning to describe the assembly plan generation process and guide the VLM. The JSON file from Step 2 is concatenated with the third prompt as input, guiding GPT-4o to produce the final text-based assembly plan.
4. Section II includes the fourth prompt, which converts the text-based plan into a traversable tree structure for action sequencing in robotic assembly. We achieve this conversion using a simple text input with in-context learning examples.

1.a) **Image Set** for JSON File Generation



1.b) **Text Instructions** for JSON File Generation

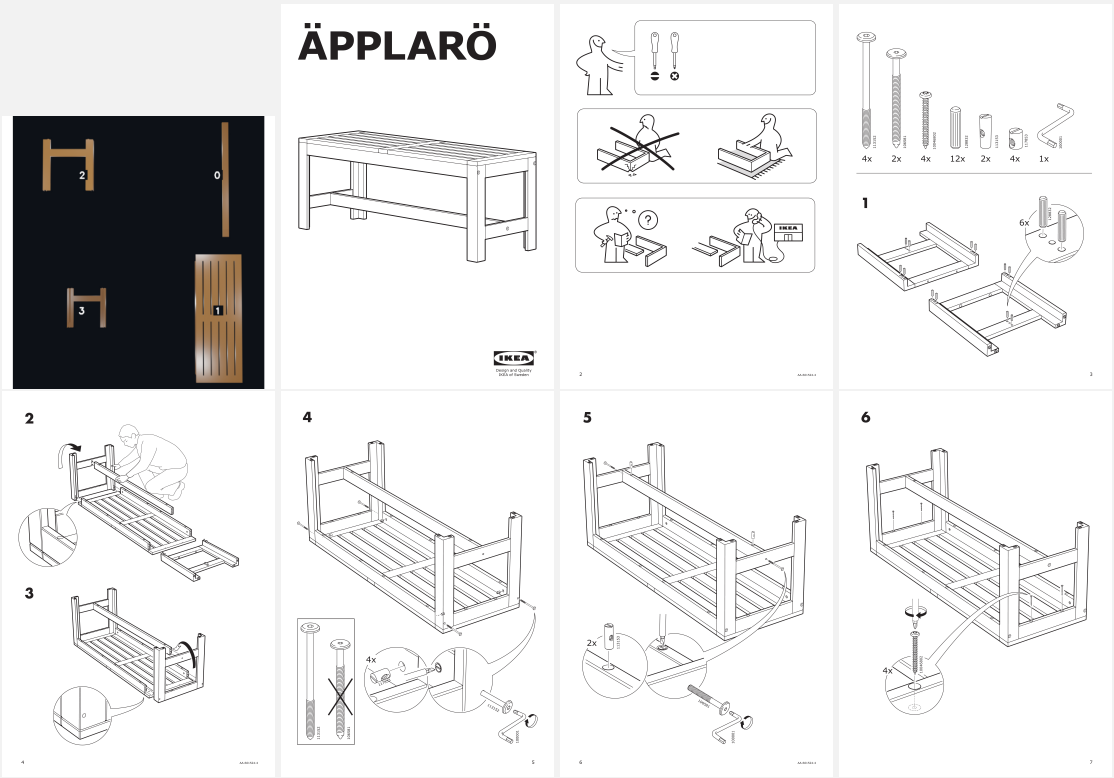
Input is one image, which is a top view of all the parts of one piece of furniture, each has a number, and another image, which is the first page of the setup manual

You should list all the parts in the image, determine their number and name (short description of the part), and show your result in JSON format.

Following is an example. Note that your output should only contain the JSON code without any explanation.

########## example start ##########
[
{
”name”: ”seat frame,”
”number”: [0]
},
{
”name”: ”side leg,”
”number”: [1]
},
{
”name”: ”side le,”
”number”: [2]
},
{
”name”: ”support b,” ”
”number”: [3]
}
]
########## example end ##########

2.a) **Image Set** for JSON File Refinement



2.b) **Text Instructions** for JSON File Refinement

You are a robot assistant responsible for assembling IKEA furniture.

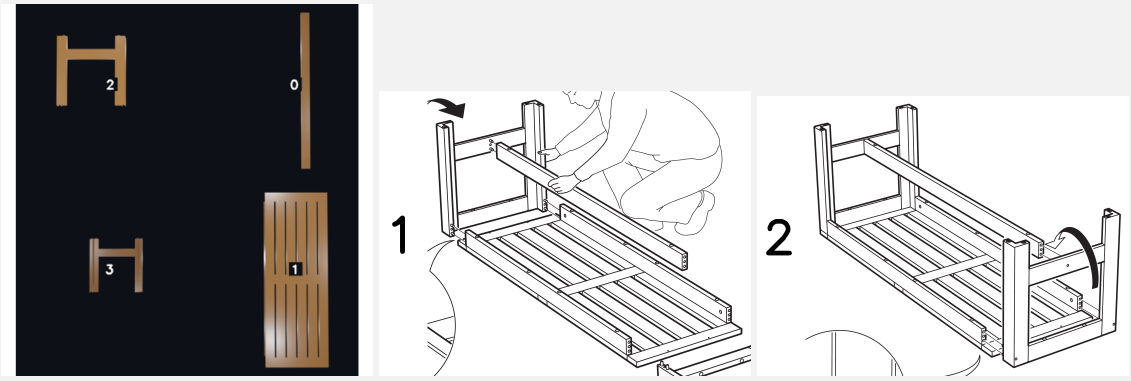
Your inputs include {A}: an rbg image of the scene consisting of furniture parts labeled with white numbers on a black background, {B}: a JSON file that describes the image's objects and labels, and {C}: a set of IKEA setup manual pages.

Please note that you will only construct the piece of furniture that the manual describes.

You can ignore nails and other tools in the manual and only focus on the furniture parts that exist in {A}: the rbg scene image.

First, you are ONLY responsible for identifying the relevant materials that will be required to assemble the furniture in the image. Output a table of selected materials, with their labeled numbers and a brief explanation of why they are selected and how they are related to items on the setup manual. The table format should be JSON, and it should be really similar to {B}, but with an additional explanation section for each selected material and its labeled number. Hint: Usually, in 99.999% of cases, the number of selected materials equals the number of labeled furniture parts.

3.a) **Image Set** for Step-By-Step Plan Generation



3.b) **Text Instructions** for Step-By-Step Plan Generation

You are a robot assistant responsible for assembling IKEA furniture. You will be responsible for creating a detailed step-by-step plan for assembling the furniture.

For your input, you will receive a set of images, which represent a few pages of the setup manual containing the setup instructions for the furniture. On the left of each page, there is a rectangular section with a white background and a big, black, bolded number. This number indicates the current assembly step. On each page, we segment the furniture with different colors (the three most common are red, green, and purple, though sometimes other colors are used). The purpose of using these colors is to help you clearly identify which furniture parts are involved in each assembly step.

You will also receive an rbg image of the scene consisting of furniture parts labeled with white numbers on a black background and a JSON-formatted table that describes the RGB image's objects and labels.

Your new task is to carefully describe every step according to the manual. Each colored segmented furniture part should correspond to one step. Your planned steps should only describe what and how segmented furniture parts are involved; don't worry about nails and other minor tools for now. Your focus should only be on the colored segmented furniture parts. Be as specific as possible in your description.

Let's think step by step: (1) count the total number of colored, segmented furniture parts. (Hint: This equals the total number of pages in the manual, with each page identified by a big, bold black number on the left.) The total number of colored, segmented furniture parts will be your total number of steps. (2) for each step, focus on one colored, segmented furniture part at a time. Describe only the furniture parts involved in that step. (3) We repeat step 2 for each remaining step until we have described all the steps. So, if there is only one page of the setup manual overlayed with mask segmentations, then there is only one step. If there are ten pages of the setup manual overlayed with mask segmentations, then there are ten steps.

Here is an example of a fully constructed plan for your reference only. It has nothing to do with the current plan:

########## assistant example start ##########
We have five input images, but one image shows furniture parts lying on a floor that we label with marks (white numbers on a black square background). Therefore, we have only four pages of the setup manual overlaid with mask segmentations. Thus, there are four total steps.

Step 1:
- **Parts Needed:** Backrest Frame (1), Seat Cushion (5)
- **Instructions:**
- **Align Frame and Seat:** Connect the backrest frame (1) next to the seat cushion (5) as shown in the segmented manual.

Step 2:
- **Parts Needed:** Subassembly from Step 1, Side Leg Frame (2)
- **Instructions:**
- **Position Leg Frame:** Link the first side leg frame (2) with the assembled seat and backrest combo from Step 1.

Step 3:
- **Parts Needed:** Subassembly from Step 2, Support Beam (3), Support Beam (4), Side Leg Frame (6)
- **Instructions:**
- **Connect Support Beams:** Attach support beams (3), (4), and the second side leg frame (6) between the assembled frame and leg structure from Step 2.

########## assistant example end ##########

Now it is your turn to generate a detailed step-by-step plan; here is the JSON formatted table:

4) Prompt for Converting Text-Formatted Plan to Tree

You are a robot assistant responsible for assembling IKEA furnitures.

Your new task is to convert a step-by-step furniture assembly instruction plan from text format into a tree format.

The tree represents the stage of the furniture assembly, with lower-level nodes representing the initial and beginning stages and the upper level representing the concluding and finished stages of the furniture assembly.

We treat each end node (leaf) of the tree as an atomic furniture part that we cannot further decompose. As you move up the tree, each parent node will represent two or more child nodes combined. Finally, the root node will be the completed furniture.

You should clearly describe how every node is connected.

We output the tree strictly as a nested list of integers without any additional comments or text.

4) (Continued) In-Context Learning Examples for Text-Formatted Plan to Tree Prompt

EXAMPLE INPUT 1:
Here's a step-by-step assembly plan for the furniture using the provided parts:

Step 1: Assemble Backrest and Seat
- **Parts Needed:** Backrest Frame (1), Seat Cushion (5)
- **Instructions:**
- Place the Backrest Frame (1) and Seat Cushion (5) adjacent as shown in their respective colors (red and green).
- Ensure the backrest is upright and securely attached to the seat.

Step 2: Attach Side Leg Frame
- **Parts Needed:** Side Leg Frame (2) and subassembly from Step 1
- **Instructions:**
- Position the Side Leg Frame (2) on one side of the assembled backrest and seat structure.

Step 3: Attach Side Leg Frame Again
- **Parts Needed:** Side Leg Frame (7) and subassembly from Step 2
- **Instructions:**
- Position the Side Leg Frame (7) on the other side of the assembled backrest and seat structure.

Step 4: Connect Support Beams
- **Parts Needed:** Support Beams (3, 4) and subassembly from Step 3
- **Instructions:**
- Attach Support Beams (3, 4) to the inside of the Side Leg Frame, as depicted.

Check the entire assembly for any loose parts and re-tighten as necessary. The chair should now be fully assembled and ready for use.

EXAMPLE OUTPUT 1:
```python
[ [ [ [ 1, 5 ], 2 ], 7 ], 3, 4 ]
```

4) (Continued) In-Context Learning Examples for Text-Formatted Plan to Tree Prompt

EXAMPLE INPUT 2:

Step 1: Connect Support Beams and Leg Frame
Parts Involved: Support Beams (0 and 3), Leg Frame (4)
- **Instructions:** Position the leg frame (4) horizontally on the floor. Align the support beams (0 and 1) vertically to connect with the leg frame. Ensure that each beam is fitted securely into the designated slots on the frame.

Step 2: Attach Backrest Slats
Parts Involved: Backrest Slats (2) and subassembly from Step 1
- **Instructions:** Insert the backrest slats (2) into the slots on the leg frame. Ensure that the slats are facing outward and securely fitted to provide back support.

Step 3: Connect Seat Cushion
Parts Involved: Seat Cushion (1) and subassembly from Step 2
- **Instructions:** Place the seat cushion (1) on top of the assembled frame. Align the cushion with the edges of the frame for balance and comfort.

EXAMPLE OUTPUT 2:
```python
[ [ [ 0, 3, 4 ], 2 ], 1 ]
```
EXAMPLE INPUT 3:

Step 1: Connect Support Beams and Leg Frame
Parts Involved: Support Beams (7, 11, 6), Leg Frame (5)
- **Instructions:** Position the leg frame (5) horizontally on the floor. Align the support beams (7, 11, 6) vertically to connect with the leg frame. Ensure that each beam is fitted securely into the designated slots on the frame.

Step 2: Attach Backrest Slats
Parts Involved: Backrest Slats (1, 10) and subassembly from Step 1
- **Instructions:** Insert the backrest slats (1, 10) into the slots on the leg frame. Ensure that the slats are facing outward and securely fitted to provide back support.

Step 3: Connect Seat Cushion
Parts Involved: Seat Cushion (3) and subassembly from Step 2
- **Instructions:** Place the seat cushion (3) on top of the assembled frame. Align the cushion with the edges of the frame for balance and comfort.

Step 4: Connect Support Beams and Leg Frames
Parts Involved: Support Beams (8, 4), Leg Frames (2, 9)
- **Instructions:** Position the leg frame (2, 9) horizontally on the floor. Align the support beams (8, 4) vertically to connect with the leg frame.

Step 5: Connect Support Beams and Leg Frames
Parts Involved: Subassembly from Step 4 and subassembly from Step 3
- **Instructions:** Connect the two subassemblies together

Step 6: Connect Support Beams and Leg Frames
Parts Involved: Leg frame (0) and subassembly from Step 5
- **Instructions:** Connect the final leg frame with the previous subassembly

EXAMPLE OUTPUT 3:
```python
[ [ [ 8, 4, 2, 9 ], [ [ [ 7, 11, 6, 5 ], 1, 10 ], 3 ] ] 0]
```

YOUR REAL INPUT: