IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks

Youngwoon Lee[1], Edward S. Hu[2], and Joseph J. Lim[1]

***Abstract*— The *IKEA Furniture Assembly Environment* is one of the first benchmarks for testing and accelerating the automation of long-horizon and hierarchical manipulation tasks. The environment is designed to advance reinforcement learning and imitation learning from simple toy tasks to complex tasks requiring both long-term planning and sophisticated low-level control. Our environment features 60 furniture models, 6 robots, photorealistic rendering, and domain randomization. We evaluate reinforcement learning and imitation learning methods on the proposed environment. Our experiments show furniture assembly is a challenging task due to its long horizon and sophisticated manipulation requirements, which provides ample opportunities for future research. The environment is publicly available at `https://clvrai.com/furniture`.**

I. Introduction

The ability to perform complex manipulation of physical objects is necessary to use tools, build structures, and ultimately interact with the world in a meaningful way. Simulated benchmarks [1]–[5] have played a key role in the recent advances in reinforcement learning (RL) and imitation learning (IL) for robotic manipulation. However, these benchmarks are limited to simple, short-horizon tasks, such as picking, pushing, and peg inserting. This lack of a "standardized" simulated environment for long-term tasks is the main bottleneck of advancing RL and IL techniques toward solving long-horizon tasks. To this end, we introduce the *IKEA Furniture Assembly Environment* as a new benchmark for complex long-horizon robot manipulation. We believe our environment can play a key role in advancing RL and IL methods on long-horizon robotics tasks.

Even for humans, furniture assembly is not simple. Imagine that you are building IKEA furniture. First of all, it is not trivial to figure out how to assemble pieces into the final configuration. Specifically, it is not apparent from pieces on the floor which parts to choose for attachment and in what order. Hence, we need to dissect the final configuration and deduce the sequence of tasks necessary to build the furniture. Moreover, connecting two parts requires complicated manipulation skills, such as accurate alignment of two attaching points and sophisticated force control to firmly attach them. Therefore, furniture assembly is a comprehensive robotics task [6]–[8] requiring reliable 3D perception, high-level planning, and sophisticated control, making it a suitable benchmark for robot learning algorithms.

[1]Youngwoon Lee and Joseph J. Lim are with the Department of Computer Science, University of Southern California, Los Angeles, CA. {lee504,limjj}@usc.edu

[2]Edward S. Hu is with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA. hued@seas.upenn.edu

Fig. 1: The *IKEA Furniture Assembly Environment* is a furniture assembly simulator with 60 furniture models, 6 robots, and customizable background, lighting, and textures.

The *IKEA Furniture Assembly Environment* is a visually realistic environment that simulates the task of furniture assembly as a step toward autonomous long-horizon manipulation. The environment simulates 60 furniture models and supports various agents including Sawyer, Baxter, Jaco, Panda, and Fetch robots. To provide further diversity, the environment supports randomization in physics, lighting, textures, and more factors of variations. Lighting conditions, textures, and backgrounds can be customized or randomized using Unity [9] as illustrated in Fig. 1.

A variety of research problems could be investigated with this new environment that broadly span perception, planning, and control. For perception, the environment could be used to solve 3D object detection, pose estimation, instance segmentation, scene graph generation, and shape estimation problems. For robotic planning and control, the environment provides dense reward functions and automated demonstration generation, making it suitable for testing RL and IL on long-horizon complex manipulation tasks. Further, diverse shapes of furniture pieces can be used to learn and evaluate generalizable and transferable skills.

In this work, our aim is to provide a testbed and benchmark for complex long-horizon robotic manipulation tasks that allows researchers to study reinforcement learning and imitation learning. For benchmarking RL and IL algorithms, we provide a shaped dense reward and demonstrations for 8 selected furniture models. Our empirical evaluation of RL and IL methods on the 8 benchmark furniture models shows that, despite recent progress in RL and IL, current methods are not able to solve complex robotic manipulation tasks, providing ample opportunities for future research.

II. Related Work

Deep reinforcement learning (RL) has made rapid progress with the advent of standardized, simulated environments.

Most progress has been made in game environments, such as Atari [10], VizDoom [11], and StarCraft2 [12]. Recently, many simulated environments have been introduced in diverse applications, such as autonomous driving [13], [14], indoor navigation [15]–[18], continuous control [1], [4], [5], [19], and recommendation systems [20].

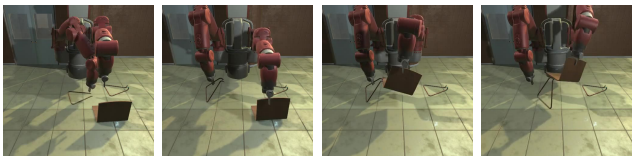
In robotic manipulation, most existing environments have been focused on short-term manipulation tasks, such as picking and placing [1], [21], in-hand dexterous manipulation [22], [23], door opening [24], and peg insertion [25], [26]. Recent advancements in simulators, such as Robosuite [4], RLBench [2], PyRoboLearn [27], and Meta-World [3], take a step towards a comprehensive manipulation simulator by offering a variety of manipulation tasks. However, these tasks, which consist of lifting, stacking, and picking and placing, are still limited to primitive skills.

Composite manipulation tasks, e.g., block stacking [28], ball serving [19], and kitchen tasks [29], [30], have been proposed but limited to little variation in shapes and physical properties of objects. In contrast, we simulate a complex manipulation task, *furniture assembly* to evaluate long-term planning and generalizable skills for various shapes and materials of objects. While robotic furniture assembly has been studied in instrumented and constrained settings [6]–[8], our simulated environment increases accessibility of the furniture assembly task to the community.

Moreover, recent progress in vision-based RL methods [31]–[33] shows comparable sample efficiency to state-based RL. To test the scalability of such vision-based RL approaches to real-world scenarios, our environment can serve as a visually realistic benchmark, which provides photorealistic rendering with diverse and configurable textures, backgrounds, and lighting.

III. IKEA Furniture Assembly Environment

To advance reinforcement learning and imitation learning from simple, videogame-esque tasks to complex and realistic tasks, the *IKEA furniture assembly environment* features long-horizon and hierarchical tasks, realistic rendering, and domain randomization. The furniture assembly can be accomplished by repeating (1) selecting two compatible parts, (2) grasping these part(s), (3) aligning the attachable connectors, and (4) firmly attaching them, as illustrated in Fig. 2, until all parts are assembled. Thus, furniture assembly has long horizon (200-1500 steps) compared to prior manipulation benchmarks, e.g., 280 for Franka Kitchen [29].



(a) Selecting (b) Grasping (c) Aligning (d) Attaching

Fig. 2: Our environment simulates robotic furniture assembly: a robot (a) decides which parts to assemble, (b) grasps the desired parts, and (c) aligns and (d) attaches the grasped parts. This procedure is repeated until all parts are assembled.

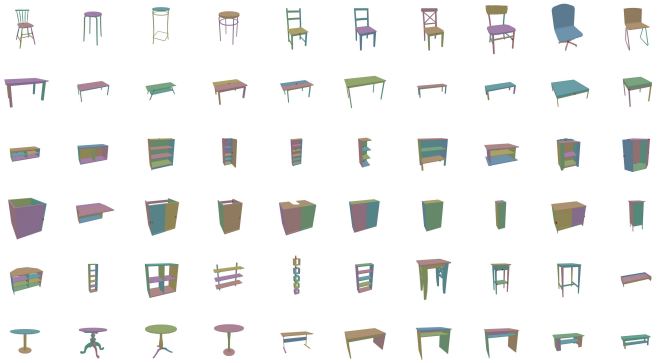


Fig. 3: Diverse furniture models included in our environment. Each furniture is modeled following the IKEA's user's manual. Different parts are colored differently for visualization.

A. Environment Development

As a challenging robotic manipulation testbed including long-horizon tasks and 3D alignment of various shapes of objects, we propose a novel 3D environment that supports assembling IKEA furniture. For fast and accurate physics simulation, we use MuJoCo [34] as the underlying physics engine. Specifically, our environment is built on top of Robosuite [4], which features modularized API design and diverse robot controllers. We use the Unity game engine [9] and the MuJoCo-Unity interface from DoorGym [24] for realistic and configurable 3D rendering. Our environment follows the OpenAI Gym interface [1] for easy integration with existing RL and IL libraries.

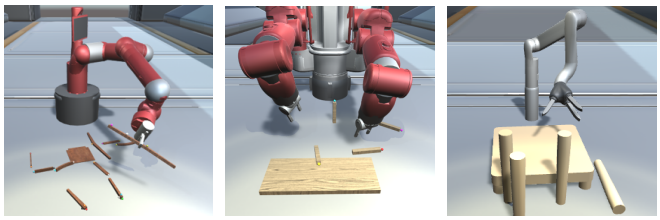
B. Furniture Models

Our environment provides simulation of furniture assembly of 60 different furniture models as shown in Fig. 3. Each furniture is modeled following IKEA's official user's manuals. Due to the limitation of physics simulation and difficulty in modeling, some complex structures are simplified and small details, such as carving and screws, are omitted.

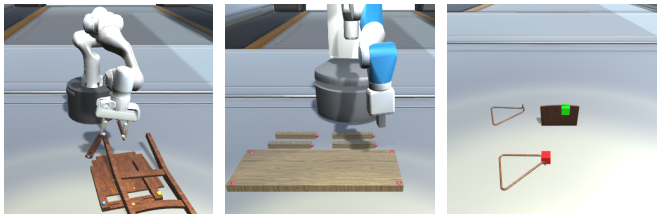
Although screwing is an important aspect of many robotic assembly tasks, conducting realistic screwing with available grippers is infeasible without developing end-effectors or robot arms dedicated to screwing. Moreover, accurate physical simulation of screws is not supported by the MuJoCo physics engine, and most physics engines in general. Currently, the environment contains an abstracted screwing phase where the robot must select the connect action to attach the parts. We plan to replace the abstracted connect action with a realistic screwing action as future work.

Parts: Based on IKEA's official user's manuals, the furniture models are created using the 3D modeling tool, Rhino, and each furniture part is converted to a separate 3D mesh file in a format of STL, which is used for both physics simulation in MuJoCo and rendering in Unity. For the robots with fixed bases (i.e., limited reach), our environment supports downscaling of furniture models to fit in the workspace.

Connectors: Physics simulators show limited accuracy for sophisticated screwing and peg inserting interactions between attaching points of furniture parts. Therefore, we abstract



(a) Sawyer (b) Baxter (c) Jaco



(d) Panda (e) Fetch (f) Cursor

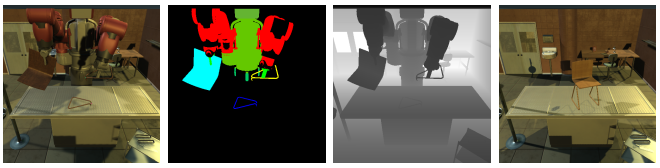
Fig. 4: The Sawyer, Baxter, Jaco, Panda, and Fetch mobile manipulator robots are supported. Various action spaces are provided including end-effector space, joint velocity, and joint torque controls.

connection information and attachment points between two furniture parts, such as screws and holes, with *connectors*. Connectors are located on the furniture parts and serve as areas of attachment by representing the correspondence between attaching points and relative 3D pose.

Specifically, we represent connectors with their ID, position, and orientation to the part. The IDs are used to verify compatibility between two connectors. For example, a pair of connectors on part `A` and part `B` have IDs `A-B` and `B-A`, respectively. Identical parts can be used interchangeably since they share the same connector IDs. For furniture pieces with symmetric shapes, acceptable attachment angles are also specified in connectors (e.g., $\{0°, 90°, 180°, 270°\}$ for rectangular table legs and $[0°, 360°]$ for bars). All furniture parts and connectors are manually annotated.

C. Furniture Assembly Simulation

In our environment, robotic arms can move around the environment and interact with furniture parts. While assembling two furniture parts requires screwing in the real world, we abstract it with a *connect* action. Therefore, in addition to actions for robot control (e.g., joint torque or 3D end-effector pose), the action space has one additional action dimension, *connect*, which attaches two *attachable* furniture parts (i.e., two corresponding connectors are compatible and aligned). Specifically, we examine the Euclidean distance between the connectors $d_{L2}(\mathbf{x}_A, \mathbf{x}_B)$, the cosine similarities between the connector up vectors $s_{cos}(\mathbf{u}_A, \mathbf{u}_B)$, forward vectors $s_{cos}(\mathbf{f}_A, \mathbf{f}_B)$, and projections of up vectors to a segment between two connectors $s_{cos}(\mathbf{u}_A, \mathbf{x}_B - \mathbf{x}_A)$ and $s_{cos}(\mathbf{u}_B, \mathbf{x}_A - \mathbf{x}_B)$, where $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{f}$ denote a 3D coordinate, up vector, and forward vector of a connector, respectively. By default, two connectors are attachable when they are within $2\,\mathrm{cm}$ and the cosine similarities are larger than 0.99.



(a) RGB (b) Seg. map (c) Depth (d) Goal

Fig. 5: An RGB image, pixel-wise part segmentation map, depth map, and goal image are available from cameras.

D. Agents

Our environment supports a variety of robots for interacting with furniture: Rethink Sawyer, Rethink Baxter, Kinova Jaco, Franka Emika Panda, Fetch mobile manipulator, and Cursor (abstract agent), as illustrated in Fig. 4.

Observation space: The observation space is fully configurable to fit a variety of problem settings. It can consist of agent state (e.g., joint positions, velocities, and contact forces), environmental state (e.g., 3D coordinates and orientations of furniture parts), and camera observations. Aside from third-person view RGB images, the environment also supports object segmentation masks (pixel-wise dense annotation) and depth camera observations as shown in Fig. 5. More cameras can be added to collect images from diverse views, such as egocentric view and wrist view.

Action space: The action space consists of arm movement, gripper control, and *connect* action but varies by control modes: 6D end-effector space control with inverse kinematics, joint velocity control, and joint torque control. The Fetch mobile manipulator has two additional actions for moving and turning its base.

For RL and IL benchmark, we use the Rethink Sawyer robot with joint velocity control as an agent. The observation consists of the robot state (joint angles and velocities), end-effector state (end-effector position, rotation, and velocity), and object state (3D coordinates and orientations of all parts).

E. Reward Function

For all furniture models, our environment provides a sparse reward for every successful pick, attachment, and full assembly. However, learning from such a sparse reward signal is not practical with existing RL methods. To ease the challenge of learning from a sparse reward, we provide a well-shaped dense reward function for 8 furniture models based on manually annotated way-points.

The dense reward function is a multi-phase reward defined with respect to a pair of furniture parts to attach (e.g., a table leg and a table top) and the corresponding manually annotated way-points, such as a target gripping point $\mathbf{g}$ for each part. The reward function for a pair of furniture parts consists of eight different phases as follows:

1) **Initial phase**: The robot has to reconfigure its arm pose to an appropriate pose $\mathbf{p}_{\text{init}}$ for grasping a new furniture part. The reward is proportional to the negative distance between the end-effector $\mathbf{p}_{\text{eff}}$ and $\mathbf{p}_{\text{init}}$.
2) **Reach phase**: The robot reaches above a target furniture part. The reward is proportional to the negative

distance between the end-effector $\mathbf{p}_{\text{eff}}$ and a point $\mathbf{p}_{\text{reach}}$ $5\,\text{cm}$ above the gripping point $\mathbf{g}$.

3) **Lower phase**: The gripper is lowered onto the target part. The phase reward is proportional to the negative distance between $\mathbf{p}_{\text{eff}}$ and the target gripping points $\mathbf{g}$.
4) **Grasp phase**: The robot learns to grasp the target part. The reward is given if the gripper contacts the part, and is proportional to the force exerted by the grippers.
5) **Lift phase**: The robot lifts the gripped part up to $\mathbf{p}_{\text{lift}}$. The reward is proportional to the negative distance between the gripped part $\mathbf{p}_{\text{part}}$ and the target point $\mathbf{p}_{\text{lift}}$.
6) **Align phase**: The robot roughly rotates the gripped part before moving it. The reward is proportional to the consine similarity between up vectors $\mathbf{u}_A, \mathbf{u}_B$ and forward vectors $\mathbf{f}_A, \mathbf{f}_B$ of the two connectors.
7) **Move phase**: The robot moves and aligns the gripped part to another part. The reward is proportional to the negative distance between the connector of the gripped part and a point $\mathbf{p}_{\text{move\_to}}$ $5\,\text{cm}$ above the connector of another part, and the cosine similarity between two connector up vectors, $\mathbf{u}_A$ and $\mathbf{u}_B$ and forward vectors $\mathbf{f}_A$ and $\mathbf{f}_B$. Note that all connectors are labeled with aligned up vectors and forward vectors.
8) **Fine-grained move phase**: The robot must finely align two connectors until attached. The same reward is used as the *move phase* with a higher coefficient, making the reward more sensitive to small changes. In addition, we provide reward based on the activation of the connect action $\mathbf{a}[connect]$ when the parts are attachable.

Upon every phase completion, a completion reward is given to encourage the agent to move onto the next phase. In addition to the phase-based reward, a control penalty $\|\mathbf{a}\|_2$, stable wrist pose reward $s_{\cos}(\mathbf{u}_{\text{wrist}}, (0,0,-1))$ and $s_{\cos}(\mathbf{f}_{\text{wrist}}, \mathbf{g}_1 - \mathbf{g}_2)$, and gripping reward (i.e., open the gripper only in *initial*, *reach*, and *lower* phases) are provided throughout the episode. If the robot releases the gripped object, the episode terminates early with a negative reward.

The phase completion is determined based on whether the robot and part configurations satisfy a distance and angle constraint with respect to a target configuration. When all phases are completed, the phase is reset to *initial phase*. This process is repeated until all parts are attached.

Our dense reward function is verified for full assembly on a simple three-block assembly task. Note that these phases are only used for computing rewards and not available to a policy. Please refer to our code for further details.[1]

F. Demonstration Collection and Generation

Our environment provides multiple ways to collect demonstrations. Human teleoperation can be used to collect demonstrations with the end-effector space control using keyboard, 3D space mouse, and HTC Vive VR controllers.

Scripted policies are provided to generate demonstrations for the benchmark furniture models with the Sawyer robot to assist in evaluating IL algorithms. The hard-coded policies operate by iterating through the predefined set of phases described in Section III-E and manually annotated way-points. Each phase consists of either minimizing the Euclidean distance between relevant points or maximizing cosine similarity between relevant directional vectors. The corresponding actions can be computed using positional or rotational differences, and then performed using the end-effector space control.

[1] https://github.com/clvrai/furniture

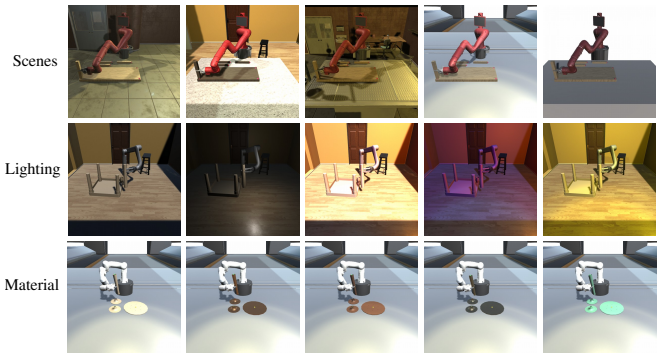


Fig. 6: Examples of diverse visual properties. The first row shows different scenes. The second row shows different lighting configurations, such as soft light, ambient light, and low-visibility. The final row shows variations in furniture textures, such as wood, aluminum, and glass.

G. Domain Randomization

To promote generalization of the learned skills, the environment should provide enough variability in furniture compositions, visual appearances, object shapes, and physical properties. Our environment provides variability in furniture compositions and object shapes by containing a diverse set of furniture including chairs, tables, cabinets, bookcases, desks, shelves, and TV units (see Fig. 3). For a given furniture, the environment can randomly initialize the pose of each furniture part in the scene to increase the diversity of the initial states. Additionally, the environment can randomize physical properties, such as gravity, scale, density, and friction, to add more variation in the task. The environment also supports diverse visual properties, such as lighting, backgrounds, and textures, as illustrated in Fig. 6.

H. Assembly Difficulty by Furniture

The difficulty of a furniture model largely depends on the shape of the furniture pieces. For example, `toy_table` is more difficult to grasp due to its cylindrical legs while rectangular legs of `bench_bjursta` are easier to grasp. Chairs are generally more difficult to assemble due to their irregular shapes (e.g., chair seat and chair back). Bookcases are generally more difficult than chairs due to the wide and thin pieces, which are difficult to grasp. We rank the furniture models by assembly difficulty with respect to the number and shape of the furniture parts.

IV. Experiments

To benchmark reinforcement learning (RL) and imitation learning (IL) methods on complex long-horizon ma-

nipulation tasks, we selected 8 furniture models as a benchmark and annotated way-points for the dense reward and demonstration generation. The first four benchmark furniture models require peg-insertion-like attachment: `three_blocks`, `toy_table`, `table_dockstra`, and `table_bjorkudden`, where a peg-like part must be precisely inserted into a recessed receptacle of another part. The remaining four furniture models (`bench_bjursta`, `chair_agne`, `chair_ingolf`, and `table_lack`) do not have recessed receptacles, which enables the parts to snap together like magnets.

The benchmark is conducted with the Sawyer robotic arm and joint velocity control, and the furniture models are sampled mostly from tables and chairs since other furniture models (e.g., bookcases and cabinets) often consist of multiple thin boards, which require two grippers to grasp.

With the goal of providing a challenging benchmark to compare performance in learning long-horizon manipulation tasks, we design an evaluation protocol for RL and IL methods in Section IV-A. Then, we evaluate RL (Section IV-B) and IL (Section IV-C) methods on the 8 benchmark furniture models.

A. Evaluation Protocol

Evaluation metric: The most basic metric for IL and RL algorithms is the successful assembly of a furniture. However, to provide more fine-grained progress measure than a simple success and failure signal, we record the number of successful phase completions in an episode (as defined in Section III-E). The trained models are evaluated on the hold-out episodes (first 50 episodes with the hold-out random seed 0). Following this evaluation metric, we benchmark RL and IL algorithms and report average episodic phase completions for each benchmark furniture over 3 different training seeds.

Experimental setup: For both RL and IL benchmarks, we use joint velocity control. During demonstration collection, training, and testing, each furniture part is initialized with the randomness of $[-2\,\mathrm{cm}, 2\,\mathrm{cm}]$ and $[-3°, 3°]$ in (x, y)-plane. We sequentially initialize furniture parts to avoid collisions between parts. After all furniture parts are initialized, the robot is initialized to a predetermined initial position with added random noise. We set the episode length 200 per attachment, e.g., 600 for 4 parts. Two parts are attachable if the corresponding connectors are within $2\,\mathrm{cm}$ and their forward and up vector cosine similarities are larger than 0.99.

Reinforcement learning: For RL benchmark, each algorithm is trained with the shaped dense reward described in Section III-E. To compare learning performance (i.e., sample efficiency), we compare the learning curves of evaluation results every 10K environment steps up to 2M for off-policy methods and 20M for on-policy methods. The SAC off-policy update is more expensive in terms of wall-clock time – SAC took 24 hours to reach 2M steps while PPO could reach 50M steps in 24 hours.

Imitation learning: For IL benchmark, we collect 100 demonstrations for each of 8 benchmark furniture models with a hard-coded assembly policy described in Section III-F. Each demonstration is around 200-1500 steps long due to the long-horizon nature of the task.

Furniture model	PPO	SAC	GAIL	GAIL+PPO	BC
`three_blocks`	4.96	3.88	1.06	4.89	1.00
`toy_table`	3.34	3.40	1.16	5.88	1.00
`table_bjorkudden`	5.66	3.66	1.01	5.54	1.00
`table_dockstra`	5.24	3.61	1.01	6.21	1.00
`bench_bjursta`	3.66	3.23	1.01	3.33	1.00
`chair_agne`	6.26	3.60	1.06	6.38	1.02
`chair_ingolf`	4.91	7.54	1.19	7.33	1.00
`table_lack`	6.66	4.10	1.00	5.88	1.00

TABLE I: Average number of phase completions of RL and IL algorithms. SAC, PPO, and GAIL+PPO learn to pick an object (corresponds to 4 phase completions) and succeed on attaching two parts in `chair_ingolf` (corresponds to 8 phase completions) while BC and GAIL rarely pass *reach phase*. The first four models require peg-insertion style attachments, which are more difficult to accomplish than the magnet style attachment of the remaining four.

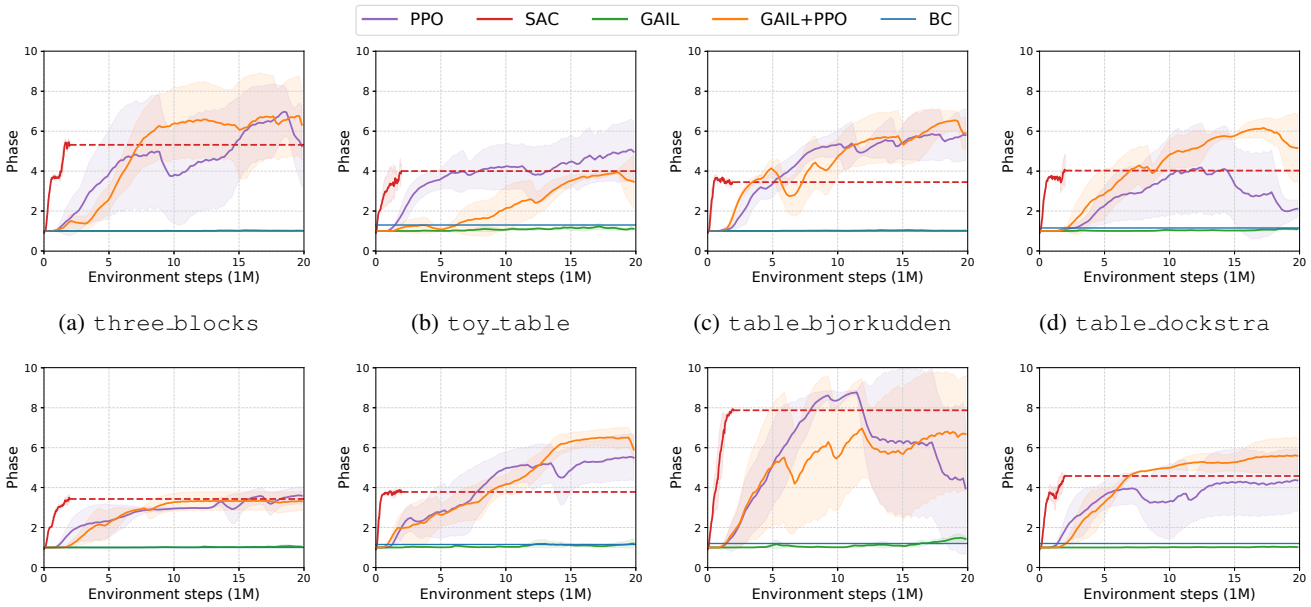
B. Reinforcement Learning Benchmark

We evaluate two state-of-the-art model-free RL algorithms, Soft-Actor Critic (SAC [35]) and Proximal Policy Optimization (PPO [36]), on the benchmark furniture models with the dense reward function. The number of completed phases is plotted over training environment steps in Fig. 7. Completing four phases means the agent has grasped the target part and accomplishing eight phases denotes the successful assembly of one pair of parts. Note that the y-axis can go up to $8 \times \#parts$, not 8.

Both SAC and PPO learn to pick the first furniture part in all tasks. In many furniture models, SAC struggles at completing the lift phase since the agent must firmly hold the gripped part and avoid collision while lifting. PPO successfully attaches the first pair of parts in `three_blocks`, `table_bjorkudden`, and `chair_ingolf`, in which the two parts to be attached are initialized close to each other. However, initializing the arm pose for the next step is challenging since the agent pushes the gripped part away while reconfigure the arm pose.

SAC is more sample efficient than PPO, with it reaching comparable performance levels millions of steps earlier than PPO. But in most cases, PPO learns to complete more phases with extensive exploration with diverse rollouts collected from 16 parallel workers, at the cost of more samples. This implies that learning to assemble even one pair of parts poses a challenging exploration problem.

While SAC and PPO can attach the first pair of parts in some furniture models, no policy is able to advance to the next pair of parts to attach. The best run is able to assemble the first pair of parts and pick the next part to be assembled. This highlights the difficulty of the long-horizon nature in furniture assembly, and shows ample room to improve RL algorithms for complex long-horizon manipulation tasks with our environment.



(a) `three_blocks` (b) `toy_table` (c) `table_bjorkudden` (d) `table_dockstra`

(e) `bench_bjursta` (f) `chair_agne` (g) `chair_ingolf` (h) `table_lack`

Fig. 7: Training curves of RL and IL algorithms on the benchmark furniture models. Successful grasping corresponds with completion of four phases, and successful assembly of one pair of parts corresponds with completion of eight phases. All algorithms were trained for about 24 hours. The dashed SAC line shows its final performance after 2M steps.

C. Imitation Learning Benchmark

For the IL benchmark, we evaluate Behavioral Cloning (BC [37]) and Generative Adversarial Imitation Learning (GAIL [38]) with joint velocity control. In addition, we also evaluate the demonstration-guided RL approach, GAIL+PPO [39], which learns from the weighted sum of GAIL and task rewards.

The quantitative results in Fig. 7 show that BC and GAIL fail to accomplish the phase of reaching to an object as the policy suffers from compounding error. On the other hand, GAIL+PPO successfully learns to pick an object and attaches the object in `three_blocks` and `chair_ingolf`. This result demonstrates the importance of having access to the shaped reward function to learn the assembly behavior.

We additionally compare with BC policies on end-effector space control, which is more robust to action errors than joint velocity control. The end-effector BC policies demonstrate some successful picking behaviors, while the joint velocity control BC policies failed completely. This suggests that BC on joint velocity control is more challenging, and therefore it is prone to failure and requires more demonstration data. Please refer to our website for further results and analysis.

D. Implementation Details

For all methods, we use 3-layer MLP with 256 hidden units for policy, value, and discriminator networks. The policy and value networks use ReLU nonlinearities while the discriminator networks use the $\tanh$ activation function. The output of the policy is squashed into $[-1, 1]$ using $\tanh$. The discount factor is 0.99. We use the Adam optimizer [40] with momentum $(0.9, 0.999)$. **PPO** is trained with the entropy coefficient $4\text{e}{-3}$, and rollout of 8192 transitions collected from 16 parallel workers. We normalize observations using the moving average and standard deviation. **SAC** first collects 1000 transitions to fill the replay buffer and the policy and critic networks are updated with the learning rate $3\text{e}{-4}$. **BC** is trained with batch size 128 for 500 epochs with learning rate $3\text{e}{-4}$. **GAIL** is trained using PPO without changing any hyperparameters. For reward, we use the vanilla GAIL reward $r_{\text{GAIL}} = -\log(1 - D(s, a))$. **GAIL+PPO** trains a policy using a combined reward $r = 0.8 \cdot r_{\text{GAIL}} + 0.2 \cdot r_{\text{ENV}}$.

V. Conclusion

We propose the *IKEA Furniture Assembly Environment* as a novel benchmark for testing complex long-horizon manipulation tasks. Furniture assembly is a challenging task requiring 3D perception, high-level planning, and sophisticated low-level control. Our experimental results show that current RL and IL methods cannot solve the task due to its long horizon and complex manipulation. Therefore, it is well suited as a benchmark for robot learning algorithms aiming to tackle complex long-horizon manipulation tasks. An important future direction is to enable sim-to-real transfer by improving realistic screwing interaction, accurate 3D modeling, and robot calibration. Moreover, supporting multi-arm or multi-robot collaboration can be another future work to overcome insufficient payloads of each robot. Finally, benchmarking visuomotor control and multi-task learning are the natural followups for the proposed environment.

Acknowledgment

The authors would like to thank Alex Yin and Zhengyu Yang for implementation, Taehoon Kim for initial discussion, and the USC CLVR lab members for constructive feedback.

REFERENCES

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[2] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *arXiv preprint arXiv:1909.12271*, 2019.

[3] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, 2019.

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

[5] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess, "dm_control: Software and tasks for continuous control," *arXiv preprint arXiv:2006.12983*, 2020.

[6] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration." in *Robotics: Science and Systems*, 2013.

[7] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 855–862.

[8] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham, "Can robots assemble an ikea chair?" *Science Robotics*, vol. 3, no. 17, p. eaat6385, 2018.

[9] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[10] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, jun 2013.

[11] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," in *IEEE Conference on Computational Intelligence and Games*, 2016, pp. 341–348.

[12] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on Robot Learning*, 2017, pp. 1–16.

[15] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[16] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[17] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[18] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," in *IEEE International Conference on Computer Vision*, 2019.

[19] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim, "Composing complex skills by learning transition policies," in *International Conference on Learning Representations*, 2019.

[20] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou, "Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising," *arXiv preprint arXiv:1808.00720*, 2018.

[21] Y. Lee, E. S. Hu, Z. Yang, and J. J. Lim, "To follow or not to follow: Selective imitation learning from observations," in *Conference on Robot Learning*, 2019.

[22] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[23] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Robotics: Science and Systems*, 2018.

[24] Y. Urakami, A. Hodgkinson, C. Carlin, R. Leu, L. Rigazio, and P. Abbeel, "Doorgym: A scalable door opening environment and baseline agent," *arXiv preprint arXiv:1908.01887*, 2019.

[25] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8973–8979.

[26] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. S. Sukhatme, J. J. Lim, and P. Englert, "Motion planner augmented reinforcement learning for obstructed environments," in *Conference on Robot Learning*, 2020.

[27] B. Delhaisse, L. Rozo, and D. G. Caldwell, "Pyrobolearn: A python framework for robot learning practitioners," in *Conference on Robot Learning*, 2019.

[28] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 1087–1098.

[29] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning," *Conference on Robot Learning*, 2019.

[30] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on Robot Learning*, 2020.

[31] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International Conference on Learning Representations*, 2021.

[32] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020.

[33] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," *arXiv preprint arXiv:2009.08319*, 2020.

[34] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, 2018, pp. 1856–1865.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[37] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[38] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[39] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International Conference on Machine Learning*, vol. 80, 2018, pp. 2469–2478.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.