Contact-Rich Object Insertion: Lessons for Zero-Shot Policy Transfer

Samarth Brahmbhatt[1], Ankur Deka[1], Andrew Spielberg[2], and Matthias Müller[1]

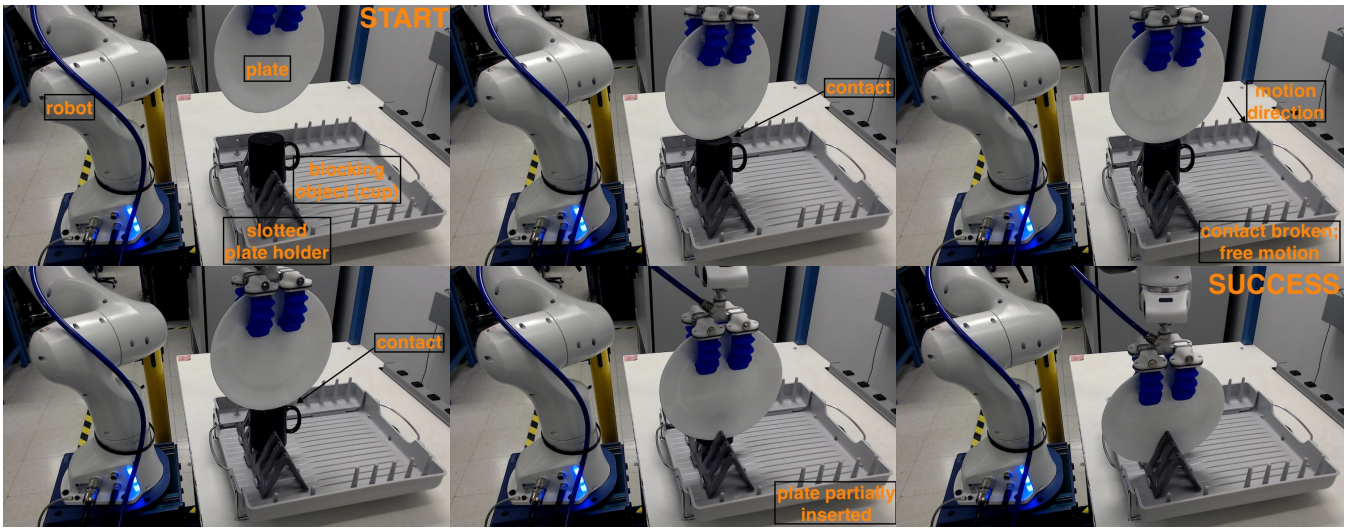

Fig. 1: Top-left to bottom-right: A plate insertion episode. The robot is controlled by our plate insertion policy trained entirely in simulation. The policy successfully deals with multiple collisions caused by noise in the target slot location, and an unseen object (a cup) blocking it, and inserts the plate into the neighbouring free slot. Zoom in to see collision details.

***Abstract*— Robotics simulators have opened up the possibility for contact-rich manipulation policies to be trained entirely or mostly in simulation. Training in simulation can be safer, cheaper, and faster than training with real robots. However, policies for contact-intensive tasks often suffer a large performance drop when transferred to a real robot. In this work, we examine this problem through the task of inserting a plate into a narrow slot. We train a policy with reinforcement learning, propose various steps to make the simulation training more realistic, and report their impact on real robot performance. Our policy not only outperforms baselines and transfers with a negligible sim-to-real performance drop, it also generalizes with a minor modification to inserting a cup and plates of different sizes and weights. Demo videos are available at `https://youtube.com/playlist?list=PLdMOXIlbRGoVL0XezrRgk4-LsqXqtICmi`.**

I. Introduction and Related Work

Common household tasks like loading a dishwasher, stocking a bookshelf, or inserting a pot into a coffee maker are contact-intensive. As robots increasingly collaborate with humans in dynamic, unseen, and un-instrumented environments, they will need to perform similar contact-intensive object placement tasks. Simulators [1], [2] are an attractive option for training robot policies for such tasks because they provide safety and scalability. However, simulation-trained policies for contact-rich tasks often show a significant drop in performance when executed on a real robot, primarily because of inaccuracies in geometry and contact modeling.

In this paper, we focus on inserting plates in a holder with narrow slots (see Figure 1). Previous works mostly address this from the planning perspective [3], [4], while this paper is about "last centimeter" contact-rich insertion. Peg-in-hole insertion, a similar task, has been an active research topic since decades. However, to the best of our knowledge, ours is the first non-hardcoded insertion policy trained completely in simulation i.e. without pre-training, training, or finetuning with real-world data or demonstrations. Please see Table I for a detailed comparison to previous insertion works.

II. Method

We assume that the object has already been grasped before the policy execution begins. In the first stage, an open-loop planner [16], [17] is used to move the end-effector to the vicinity of its approximate target pose. The policy then switches to the second stage *i.e.* a learnt controller that handles the contact-rich segment of the task. This is the main contribution of this paper. We describe its details below.

[1] S. Brahmbhatt, A. Deka, and M. Müller are with Intel Labs. E-mail: {samarth.manoj.brahmbhatt, ankur.deka, matthias.mueller}@intel.com.

[2] A. Spielberg is with the Massachusetts Institute of Technology and contributed mostly during an internship at Intel Labs. E-mail: aespielberg@csail.mit.edu.

	[5], [6]	[7]	[8]	[9]	[10], [11]	[12]	[13]	[14]	[15]	ours
Insertion task	peg	peg, USB	peg, clip	plug, USB	plug, gear	pins	peg	boxes	pegs	plates
Learnable policy	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Generalizes to different geometries	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓
Avoids need for unoccluded vision	✓	✓	✗	✗	✓	✓	✓	✓*	✗	✓
Avoids need for expert demonstrations	✓	✗	✗	✓	✓	✗	✗	✓	✓	✓
Avoids need for real world pre-training	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
Avoids need for real world training / finetuning	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓

TABLE I: Comparison of insertion tasks, policy formulations, and training requirements of the proposed algorithm with related work. *: [22] requires a GelSlim planar tactile sensor.

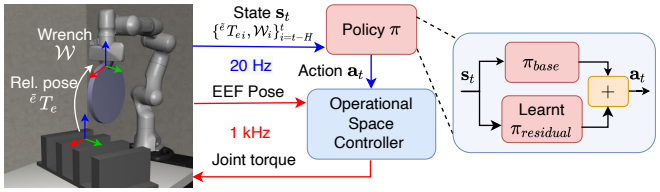

Fig. 3: Policy execution architecture.

A. RL Problem Formulation

We model the task as a Markov Decision Process (MDP) with a state space $\mathcal{S}$, continuous action space $\mathcal{A}$, transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, discount factor $\gamma \in [0, 1)$, and reward function $R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{a}_{t-1})$. Given a horizon T, we aim to learn a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ mapping observations to action probabilities $P \in \mathcal{P}$ that maximize the discounted return $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{a}_{t-1}\right)\right]$. $\mathcal{P}$ is the family of unimodal Gaussian PDFs. π is parameterized by a 2-layer (256, 256) MLP and learnt using the Soft Actor-Critic RL algorithm [18] containing a 2-layer (256, 256) MLP critic.

Observation: The policy observation contains (1) ${}^{\tilde{e}}T_e$, the current 6-DoF pose of the end-effector (e) w.r.t. its ideal target pose when the plate is successfully inserted ($\tilde{e}$), and (2) the wrench $\mathcal{W}$ exerted by the end-effector, expressed in the base frame. Using the end-effector instead of the object pose avoids the need for object tracking during execution. The ideal target end-effector pose is known accurately in simulation but not in reality, so we add random noise to it as detailed in Section II-B. Gravity, coriolis, and acceleration components are removed from $\mathcal{W}$, enabling it to exclusively capture only the object weight and contact forces.

Action: The policy outputs an action vector $\mathbf{a}_t \in [-1, 1]^6$. The first 3 elements are scaled by 45 cm to get a translation vector, while the last 3 elements are scaled by 45° to get an axis-angle rotation vector. The resulting 6-DoF pose is the residual motion target for the end-effector w.r.t. its current pose. Like [19], it is composed with a similarly scaled base motion target that moves the end-effector straight to its target pose ${}^{b}T_{\tilde{e}}$ (b is the robot base coordinate frame).

Low-level controller: As shown in Figure 3, the composite motion target is passed to a low-level operational space controller [20] running at 1 kHz. A new state vector is captured from the robot after every 50 low-level control steps. The operational space formulation and the end-effector wrench formulation described above decouple the simulated robot dynamics from the real robot dynamics. Because robot manufacturers often have closed-source dynamics APIs and simulators only approximate the dynamics parameters, this is important to reduce the simulation-reality gap.

Reward: The reward function $R\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{a}_{t-1}\right)$ sums the following components: per-step time penalty $R_{time} = -1/T$, object drop penalty (also ends the episode) $R_{drop} = -1.1$, success reward (also ends the episode) $R_{success} = +0.5$, distance to true target penalty $R_{dist} = -\sum_{m \in \{trans, rot\}} K_{dist}^{(m)} min\left(\lambda_{dist}^{(m)}, \|{}^{\tilde{e}}T_e^{(m)}\|\right)$, and action change penalty for smoothness $R_{\Delta a} = -\sum_{m \in \{trans, rot\}} K_{\Delta a}^{(m)} min\left(\lambda_{\Delta a}^{(m)}, \|{}^{a_{t-1}}T_{a_t}^{(m)}\|\right)$. $K_{dist}^{(m)}$ and $K_{\Delta a}^{(m)}$ are scale factors, while $\lambda_{dist}^{(m)}$ and $\lambda_{\Delta a}^{(m)}$ are cutoff values. In contrast to some other works like [9], [15], we do not need a staged reward function for different phases.

B. Encouraging sim-to-real transfer

${}^{\tilde{e}}T_e$ noise: In initial experiments without noise added to ${}^{\tilde{e}}T_e$ in the observation, the policy overfits by remembering the obstacle geometry and learning a trajectory that avoids all collisions. However, it fails on the real robot because simulated object and slot geometries are just approximations of the real geometries, and target slot pose is also not known perfectly in the real world. This makes unexpected collisions inevitable. Adding 6-DOF noise $\mathcal{U}\left[-\epsilon, \epsilon\right]$ forces collisions, because the true target pose is no longer observable during training. This allows us to learn collision-friendly policies and generalize to different geometries. We increase the noise magnitude ϵ in a piecewise linear curriculum.

Partial insertion initialization: But this also significantly reduces the probability of the object reaching its intended target pose with purely random exploration. Hence, inspired from [21], we initialize the robot in 50% of the episodes in poses such that the object is already partially inserted.

Policy inference delay: In a simulator, time stops during policy inference and round-trip communication of state and action from the environment to the policy. This is not true for the real world, in which the low-level controller continues moving the robot and changing its state. Adding a delay randomly sampled in the range of $[7, 13]$ low-level control steps between capturing the observation and executing the

action computed from it, was significant for sim-to-real transfer (see Table II).

Observation history: Several relevant quantities like end-effector velocity, contact history, current contact duration etc. require memory. Hence, we stack the observation for $H = 8$ steps in the policy observation to increase observability of the true state [22]. We evaluate other memory representations like recurrent policies [23] in Section III.

Wrench observation scaling: To resolve contact force spikes arising from simulation inaccuracy, we scaled the end-effector wrench observation in simulation such that the maximum force is 20 N.

C. Implementation details

We learn the policy with the Soft Actor Critic (SAC) [18] algorithm, seeding the replay buffer with 500 episodes of transitions collected with a random policy to encourage exploration. We use the simulation environment and operational space controller implementations in Robosuite [24], which uses the MuJoCo physics engine [1]. Training is performed using tf-agents [25] and Reverb [26]. Training one policy took 14 hours. While the Franka Emika Panda robot [27] is used for both simulation and real-world experiments, the former use the Panda rigid gripper and the latter use a Soft Robotics Inc. mGrip gripper [28] (see Fig. 4).

III. Experiments

Real robot experiment protocol: The episode starts with the end-effector at a random pose above the plate holder, aligned perpendicular to the table plane. The operator holds an ArUco [29], [30] marker above the target slot, estimating the end effector location for the plate to be fully inserted. The marker pose detected by a shoulder-level camera is used to calculate the noisy target end effector pose for observations (see Section II). The policy trained in simulation is deployed directly on the real robot without adaptation. As seen in Figure 4, the plate holder has 6 slots. We perform 4 trials for each slot and compute the success rate separately across slots for the first, second, third, and fourth trial to get 4 success rate values. We report the mean and standard deviation of these for each method.

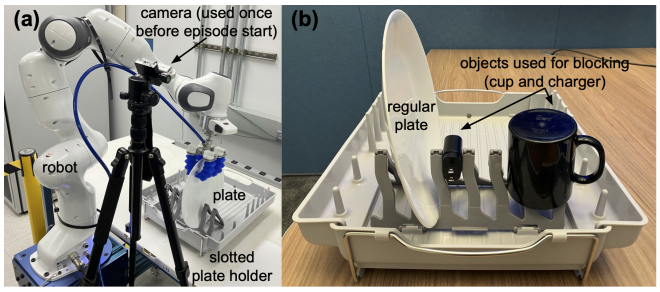

Fig. 4: **(a)**: Real robot policy execution setup. **(b)**: Detailed view of the main plate, plate holder slots, and the unseen blocking objects - cup and charger.

Blocking objects: Eventhough the plate and slot width clearance is small (2.5 mm) in the real setup, to rigorously test the learnt policies' collision behaviour we place a blocking object e.g. a cup or a phone charger in the target slot. This also better reflects real-world applications like a mobile manipulator attempting to load a partially loaded dishwasher.

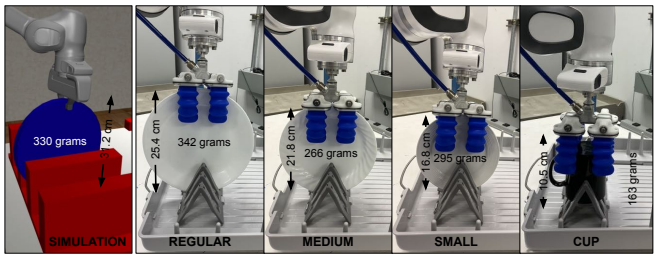

Fig. 5: Characteristics of the simulated plate used for training, and plates and cup used for real robot experiments, showing the highly approximate nature of simulation geometry. The ablation study (Figure 8) is done with the "regular" plate. Note also the gripper differences.

Differences from simulation: Figure 5 shows how the simulated plate and slot geometries are very rough approximations of their real counterparts. Other approximations include the in-hand object pose, the soft gripper simulated as a rigid gripper with spring-loaded joints, and blocking the target slot with a cup or charger. Our proposed algorithm yields a high-performance policy despite these differences.

Comparison to baselines: We compare our policy to the following baselines also implemented at the low-level through the same operational space controller as ours: (1) `straight-down`: The end-effector moves downwards blindly, (2) `random-search` [31]: `straight-down` from a point randomly sampled from a horizontal 5 cm square till a 3 N contact force. If the episode does not succeed in this pose, it moves upwards, samples another point from the square, and repeats. (3) `no vision` variant of the algorithm from [15]. The original architecture has a large number of learnable parameters and does not perform well in simulation ($17.0 \pm 13.1\%$ success rate). Hence we implement smaller 16-d wrench and proprioception feature vectors. Table II compares our policy with the three baselines. It significantly outperforms them and is able to deal with multiple collisions and finish the task (see Figures 1, 6).

Generalization: All policies are trained in simulation with a perfectly cylindrical plate, while the real robot experiments are done with a cup and three plates of varying sizes, shapes, and weights, all different from the simulated plate (see Figure 5). As shown in Figure 7, our policy consistently outperforms baselines across objects. Even though the size and geometry of the "small" plate and the cup differs significantly from the training simulated plate, our policy performs reasonably well after adding a constant offset to the Z translation component of ${}^{\tilde{e}}T_e$ observations to compensate for the height difference. Figure 6 shows a multiple object insertion application with our policy, where the cup is inserted first and the plate, targeted to the slot covered by the cup, is then inserted in the next free slot.

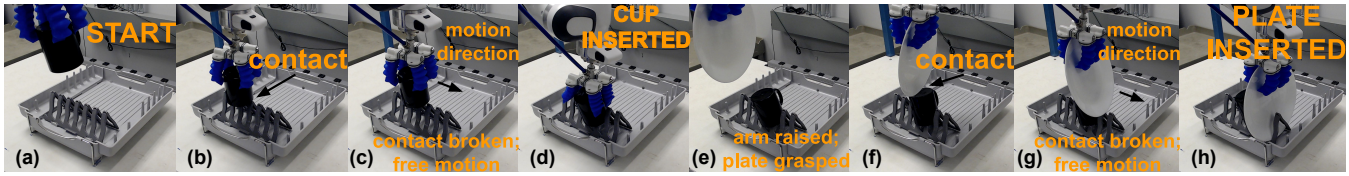

Fig. 6: Left to right: Sequential video frames showing multiple object insertion and the generalization ability of our policy.

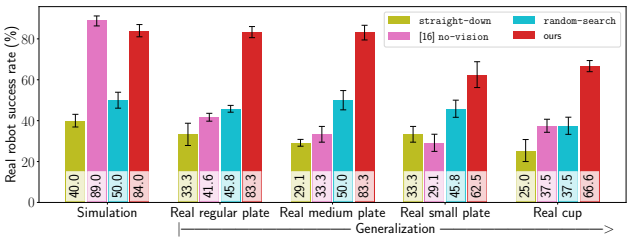

Fig. 7: **Generalization**: Real robot performance for different plates shown in Figure 5 (error bars show 20% std. dev.). Simulation performance is also shown for reference. Our algorithm consistently outperforms baselines.

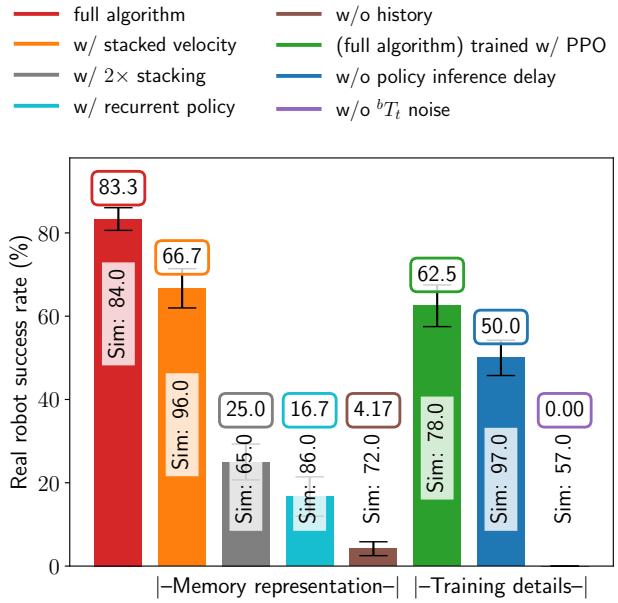

Fig. 8: **Ablation study**: Impact of various design choices. Bar height indicates real robot success rate, while simulation success rates are mentioned inside the bars (error bars show 20% std. dev.). Our full algorithm, which represents memory with $H = 8$ stacked observations, performs best. The importance of training with SAC, policy inference delay, and ${}^{\tilde{e}}T_e$ noise in observations is also shown.

Method	**Sim. success rate (%)** mean over 4 trials ± std. dev.	**Real success rate (%)**
`straight-down`	40.0 ± 15.5	33.3 ± 27.2
[15] `no-vision`	**89.0** ± 13.4	41.7 ± 9.62
`random-search`	50.0 ± 19.5	45.8 ± 8.33
ours	84.0 ± 15.0	**83.3** ± 13.6

TABLE II: Plate insertion performance in simulation and reality. Our algorithm significantly outperforms others in real-world performance, and has the smallest sim-to-real gap.

What enables sim-to-real transfer? Figure 8 shows the simulation and real robot performance of various ablated versions of our full policies. All policies are evaluated under the same conditions with the "regular" plate i.e. full-scale observation noise and non-zero policy inference delay. History representation: Observation stacking, incorporating recurrent units in the policy network, and including gripper velocity w.r.t. goal in observations all improve true state observability compared to no history, because they result in higher success rate. Observation stacking performs best. However, the policies with $H = 16$ and explicit gripper velocity significantly underperform our main policy ($H = 8$), probably because a larger observation vector linearly increases the number of learnable parameters. Adding noise in ${}^{\tilde{e}}T_e$ observations, and a non-zero policy inference delay are also crucial for ensuring the policy's transferability to the real robot. The policy had 0 real robot success rate without wrench observation scaling.

What enables successful training? The policy completely failed to train without partial insertion initialization and the residual action formulation.

Limitations: Currently the policy does not perform well when the base motor is rotated far from its center, even though it is trained in simulation under these conditions. This may be caused by imperfectly identified inertial parameters in the operational space controller.

IV. Conclusion

Through the task of inserting plates and cups into narrow slots, we presented an algorithm for learning contact-rich manipulation in simulation and transferring it directly to the real robot. We quantitatively evaluated the impact of various design choices and found that training a successful policy requires thinking carefully about observation memory, proprioception noise, time delays in the system, and contact force scale mismatch.

REFERENCES

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

[2] NVIDIA Corporation, "NVIDIA PhysX," https://github.com/NVIDIA-Omniverse/PhysX, 2023, [Online; accessed 2023-05-05].

[3] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, "Efficient and interpretable robot manipulation with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2740–2747, 2022.

[4] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "Knowrob-map-knowledge-linked semantic object maps," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2010, pp. 430–435.

[5] J. F. Broenink and M. L. Tiernego, "Peg-in-hole assembly using impedance control with a 6 dof robot," in *Proceedings of the 8th European Simulation Symposium*. Citeseer, 1996, pp. 504–508.

[6] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5844–5850.

[7] W. Gao and R. Tedrake, "kpam 2.0: Feedback control for category-level robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2962–2969, 2021.

[8] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz, "A practical approach to insertion with variable socket position using deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 754–760.

[9] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5548–5555.

[10] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[11] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9728–9735.

[12] Y. Ma, D. Xu, and F. Qin, "Efficient insertion control for precision assembly based on demonstration learning and reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4492–4502, 2020.

[13] T. Tang, H.-C. Lin, and M. Tomizuka, "A learning-based framework for robot peg-hole-insertion," in *Dynamic Systems and Control Conference*, vol. 57250. American Society of Mechanical Engineers, 2015, p. V002T27A002.

[14] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-rl for insertion: Generalization to objects of unknown geometry," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6437–6443.

[15] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 582–596, 2020.

[16] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.

[17] I. A. Sucan and S. Chitta, "MoveIt," https://moveit.ros.org, 2014, [Online; accessed 2022-06-11].

[18] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. [Online]. Available: http://arxiv.org/abs/1812.05905

[19] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.

[20] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[21] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on robot learning*. PMLR, 2017, pp. 482–495.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *2015 AAAI fall symposium series*, 2015.

[24] 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.

[25] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo, "TF-Agents: A library for reinforcement learning in tensorflow," https://github.com/tensorflow/agents, 2018, [Online; accessed 2022-06-12].

[26] A. Cassirer, G. Barth-Maron, E. Brevdo, S. Ramos, T. Boyd, T. Sottiaux, and M. Kroiss, "Reverb: A framework for experience replay," 2021.

[27] Franka Emika, "Panda research v2," https://www.franka.de/research, 2022, [Online; accessed 2022-06-11].

[28] Soft Robotics Inc., "mGrip soft gripper and controller system," https://www.softroboticsinc.com/products/mgrip-modular-gripping-solution-for-food-automation, 2022, [Online; accessed 2022-06-11].

[29] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, vol. 76, pp. 38–47, 2018.

[30] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern recognition*, vol. 51, pp. 481–491, 2016.

[31] J. A. Marvel, R. Bostelman, and J. Falco, "Multi-robot assembly strategies and metrics," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–32, 2018.