2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Scalable Platform for Robot Learning and Physical Skill Data Collection

Samuel Schneider[1], Yansong Wu[1], Lars Johannsmeier[2], Fan Wu[1] and Sami Haddadin[1]

***Abstract*— The intersection of robotics and artificial intelligence led to a profound paradigm shift in Robot Learning. Robots have the capacity to replicate human actions and also dynamically adapt, innovate, and excel across a spectrum of tasks. However, the heterogeneity in the deployment of robot platforms and software frameworks poses considerable challenges in terms of systematic testing and comparative analyses. Additionally, the data scarcity of especially force controlled robot manipulation is still restraining the development of advanced foundation models. A reference platform with default software stack can help to increase comparability, reducing development time and collect a large amount of tactile robot manipulation data. To address on this problem, we developed a Parallel and Distributed Robot AI (PD.RAI) framework, comprising a scalable ensemble of Robot Learning Units (RLUs), a global database, and the Robot Cluster Intelligence (RoCI). Each RLU is endowed with robot arms, cameras, and local computational units to autonomously engage in planning, control, and local machine learning of tactile manipulation skills. The RoCI system oversees the learning process and schedules the RLUs tasks. To show the functionality of the system, two black-box optimization algorithms are compared within the robot skill learning domain. An experiment with 24 different optimization tasks is conducted in parallel. The algorithms are incorporated into the same existing default modules acting as a reference environment. This allows for a realistic comparison without sacrificing diversity of possible configurations and testing environments.**

I. Introduction

Artificial General Intelligence (AGI) has been the "holy grail" pursued by many researchers in AI for decades. Tremendous progress has been made towards this ultimate goal in mastering purely intellectual capabilities. The first famous attempt, chosen by early AI pioneers such as Turing [1] and Shannon [2] to make AI compete with humans, was playing chess. 40 years after Shannon's proposal, IBM Deep Blue defeated the human champion Kasparov. The explosive advancement in Deep Reinforcement Learning fostered the first AI outperformed the best human player in Go [3], the most intricate board game. Recent progress in foundation models has showcased significant improvement of generalization ability thus tackling a wide-variety of tasks in the domain of language, vision, or a combination of both, is no longer impossible.

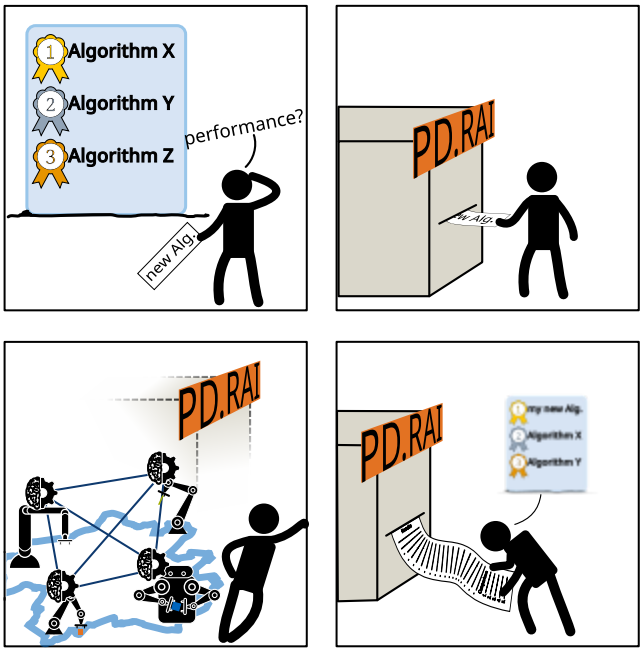

Fig. 1. 1. (upper left): How does a new algorithm compare to others? 2.(upper right) Deploy new algorithm on PD.RAI. 3.(lower left): PD.RAI is a distributed data collecting experiment platform. 4. (lower right) Collect and compare data.

On the other hand, the avenue of research in *embodied intelligence*, follows its north star — making physical learning machines — in a different way in which the physical presence, cognition, and actions of the intelligent agents play essential roles [4]. In this paradigm, the ability of embodied agents to learn from their interactions with the physical world, including manipulating their own morphology [5], is of pivotal importance. In contrast to the rapid growth of web-scale data leveraged to train Large Language Models (LLMs) [6], [7] and Visual-Language Models (VLMs) [8], exploiting scaling for training embodied AI by physical interaction and learning in the real world remains challenging due to the fact that scaling physical experiments is expensive and that there is a huge variety among robot hardware and software.

With the recent breakthroughs in vision and language foundation models [8], [7], robotics community has started to explore integrating these models, pre-trained on large-scale internet data, for various purposes, see a recent survey in [9]. By leveraging the semantic understanding capability of LLMs and VLMs in open-ended context, new doors have been opened to train agents via language-conditioned imitation learning [10] or reinforcement learning [11], for solving problems on high-level decision making and planning

[1]The authors are with the Chair of Robotics and Systems Intelligence, MIRMI - Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, Germany. Address correspondence to {samuel.schneider, f.wu}@tum.de

[1]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University Munich

[2]Funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden

[12], [13], or end-to-end control [14], [15].

A. Data Scarcity

The new paradigm shift towards embodied AI and general-purpose robots accentuates the data scarcity issue in robotics. One main source of data in robot learning, particularly imitation learning approach, is demonstrations from humans. To train a single general large model for different robot tasks, the Open-X Embodiment dataset [16] is assembled by merging 60 existing datasets from 34 research labs across the world. Those datasets were collected on different robots, via various ways, such kinesthetic teaching, teleoperation, or from expert policies. A ineluctable limitation is that samples of non-prehensile and contact-rich manipulation occupy only a small portion of the existing datasets. Another type of data source is robotic simulations, which the Deep Reinforcement Learning (DRL) approach heavily relies on. The recent decade in robotics has seen a myriad of works implementing DRL [17], [18] to solve robotic tasks from legged locomotion [19] to manipulation [20], [21]. Increasing efforts have been invested into improving the fidelity of physics simulation which is critical to the performance of sim-to-real transfer [22]. Particularly, in the realm of contact-rich simulation, a notable example is Nvidia's Factory environment [23]. However, generating new high-quality assets and simulation environments involving contact-rich and long-horizon complex tasks requires expert knowledge in modeling and significant manual efforts. Therefore, before the modeling and asset generation becomes easy, ideally automated, for general users, accurate and efficient high-fidelity contact simulation can hardly be used to generate large synthetic dataset reflecting realistic tactile manipulation.

In summary, the aforementioned trends and their limitations highlight the imperative need for acquisition and accumulation of extensive real-world data, especially on contact-rich tactile manipulation. This motivates us to focus on the third approach: *scaling up physical experiments* in robot manipulation. Rare examples can be found in this line and they are mainly limited within major corporations. In [24], Levine et al. used up to 14 robot manipulators and collected grasp attempts over two months. Google's fleet of Everyday Robots has been used in [14], [15], [25], [26]. To provide access to large-scale experiment platforms for more researchers in robotics community, CloudGripper project [27] designed an open source manipulation cell that can be stacked to form a large testbed. The CloudGripper cell is designed to be modular and low-cost, however sacrificing the potential of tackling complex tasks requiring torque-controlled redundant manipulators.

B. Benchmarking

The key to the rapid growth in the field of robot learning lies in the continuous development and improvement of simulation-enabled benchmarking frameworks, such as RoboTHOR [28], VirtualHome [29], Robosuite [30], RL-bench [31], to name a few. Other benchmarks for real world manipulation usually offer standardized objects [32] or encapsulate sequential tasks [33]. In [34], we designed an electronic task board to benchmark various industry-relevant manipulation tasks. Data collection is facilitated on this task board by automating performance measurement and logging data to a cloud server. To date, the community still lacks a benchmark framework for robot manipulation providing standardized tasks and reference robot platform as a whole. Our work aims to bridge this gap by presenting not only a physical platform, but also a complete pipeline with a standardized experimental protocol to foster benchmarking and scientific discovery in robotics.

C. PD.RAI - An AI Platform for Robot Learning

To address the data bottleneck in robotics community, in this paper we present our work on designing and building a Parallel and Distributed Robot AI Systems Engine (PD.RAI-Platform). The PD.RAI platform is designed to serve as a data collection and large-scale experiment platform to facilitate robot learning research aligned with the trend of paradigm shift in embodied AI. Its composition makes it possible to supplement a tested algorithm with a default reference software stack and therefore enable a realistic but comparable testing environment. Benchmarking tools help to evaluate new algorithms according to their performance within the PD.RAI.

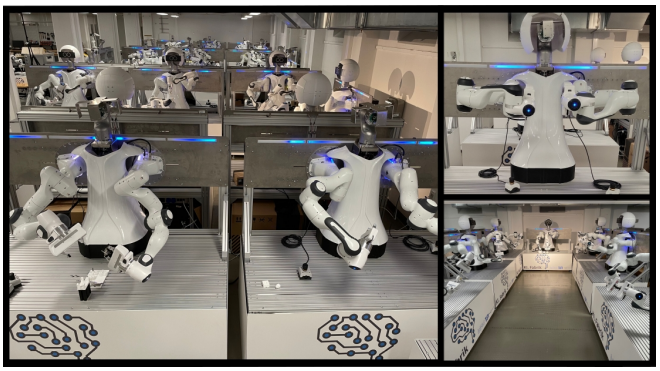

Fig. 2. Picture of the RLUs connected in the PD.RAI platform. Each RLU is a dualarm robot consisting of two Franka Emika Robot arms.

The remainder of the paper is organized as follows. Section II presents the architecture design, followed by section III where the default system configuration is introduced. Section IV demonstrates a case study in which an experiment with 24 distinct tasks is conducted to compare two black-box optimization algorithms. Finally, the paper concludes in Sec. V.

II. Architecture

PD.RAI features a parallel and distributed architecture, as illustrated in Fig. 3. A central unit named as Robot Cluster Intelligence (RoCI) module communicates with all Robot Learning Units (RLUs) to schedule and monitor their tasks. A global database is hosted alongside RoCI. The following subsections will introduce the hardware and software architecture and components. An exemplar workflow going through the experiment protocol is presented afterwards to explain how to execute an experiment on the system. The physical setup is shown in Fig. 2.

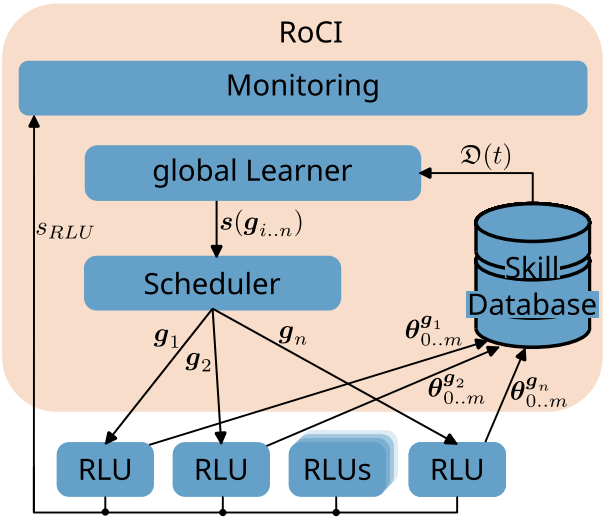

Fig. 3. The Robot Cluster Intelligence module serves as a central unit allocating goals to the RLUs and monitoring the status of PD.RAI. The investigations of the RLUs on the given goals are stored in the Skill Database to be used for optimizing the shedule of the remaining goals.

A. Hardware

A larger number of robots working in parallel does not just increase the throughput, but also provides the capability of robot-to-robot communication while acting. This leads to a broader field of algorithms that can be run on PD.RAI including transfer learning or multi-agent manipulation. Currently, there are 32 RLUs within the PD.RAI system.

Even though, the RLUs are capable to communicate over distributed locations via a VPN-like software they are all located at our lab right now. This distributed, but parallel operation provides the capability to connect idle robots of other laboratories to the PD.RAI platform. This enables more contribution to the skill models database and provides more hardware power to the system.

To enable vision-based manipulation and learning in the PD.RAI system, both hardware and software considerations are taken into account. The vision pipeline can be used (according to the configuration) in the feedback loop of the policy layer or for task planning. Each RLU has a 2 DoF head module. On the multi-purpose mounting flange, the camera (*eg.*, Intel Realsense d435i) is attached. With a kinematic range similar to that of a human head, the camera can oversee, review or guide the manipulation process and provides adaptability during operations. The head module has space reserved for future implementation of acoustic sensors, as well. The multi-modal head perception module can potentially be used for automatic task evaluation or human intention detection.

The RLU is designed to be a dual-use robot platform, which can be configured either as a bimanual robot or single arms. In order to enable dexterous manipulation, Franka Emika Robot manipulator [35] was selected for the RLUs.[1] They are equipped with more joint torque and motion sensors than usual industry robots, which enables proprioception-based collision detection and external wrench estimation and therefore tactile manipulation. The robot arms are securely mounted on a vertical back plate of an aluminum profile table, on which objects, tools and object/tool changing mechanism can be mounted to set up and reset experiments.

[1]PD.RAI will be extended to include other tactile robot arms.

B. Software components

On the software side, the PD.RAI is structured in a modular way to ensure easy exchange of algorithms is possible. The platform comes with a set of default modules. These default modules are used to supplement the tested algorithm in order to archive a full stack deployment. Together with the hardware, they serve as a realistic work environment for the tested algorithm and build a reference platform. This increases the comparability of different algorithms and approaches.

Various learning algorithms can be wrapped as a set of standardized modules with interfaces of the same paradigm, depicted in Figure 4. Through these interfaces, modules can be evoked remotely via websockets, UDP or ROS protocols, by transmitting a Json format payload. As a result, we can easily deploy an arbitrary learning algorithm on any RLU as the **local Learner**.

The **Scheduler** assigns a goal $\boldsymbol{g}$ to one of the RLUs. It decides according to the datasets $\mathfrak{D}$ available at the **Skill Database** when to assign which goal to a RLU in order to increase learning efficiency.

On each RLU, the **Policy** module creates a plan or a strategy to achieve the goal $\boldsymbol{g}$. It reacts to changes in the environment $\boldsymbol{y}(t)$ or is directly guided by a human via teleoperation. The output defines the desired action and can take different forms, dependent on the current policy. Therefore, an appropriate controller able to handle the desired action has to be loaded into the **Controller** module.

The **Controller** itself sends the realtime torque commands $\boldsymbol{\tau}(t)$ to hardware connected to RLU. Currently, we mainly use two Franka Emika Research as robot arms. The realtime sensory data from the robot arms, external wrench estimation, as well as the camera input are fed back to the software system and form the RLU's perception $\boldsymbol{y}(t)$. The **Controller** uses it as part of its feedback control loop. The **Policy** module relies on it to make decisions.

The **Quality Evaluation** module evaluates the robot's action according to a quality metric given by the goal $\boldsymbol{g}$. The quality metric defines how and with what kind of information the robots actions are evaluated. The optimization goal could be for example time efficientcy, engery consumption or utilised forces on the environment or a combination of these metrics. The applied quality metric $\mathcal{Q}$ is then forwarded to the **local Learner**.

The **local Learner** module implements different optimization algorithms. It uses the applied quality metric $\mathcal{Q}$ of the RLU's actions to find a optimal configuration of the **Policy** and the **Controller** modules. In addition to that the **local Learner** can access the **Skill Database** which serves as a connection point between all RLUs. The **Skill Database** can provide suggestions $\boldsymbol{\theta}^*(\boldsymbol{g})$ for a given goal $\boldsymbol{g}$. The results of the local Learner $\boldsymbol{\theta}_{\pi,c}(\boldsymbol{g})$ are stored in the **Skill Database** in order to be accessed by other RLUs.

The **Skill Database** hosts the dataset $\mathfrak{D}$ containing knowledge about manipulation skills. This dataset is compiled by

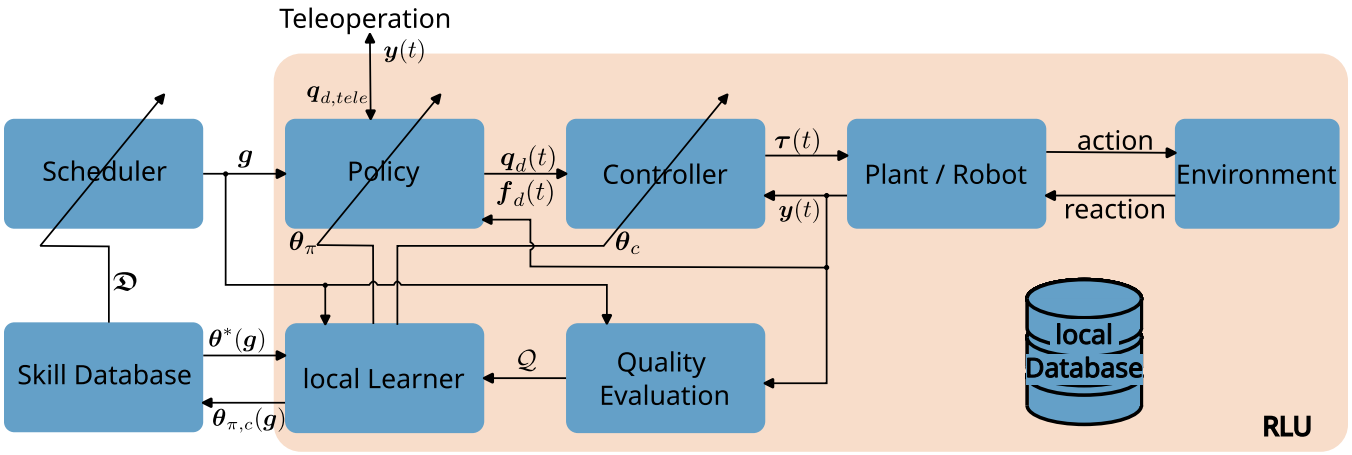

Fig. 4. Default full stack structure of a Robot Learning Unit (RLU). It depicts the dividing steps from the given goal over the policy module to the controller which sends torque commands to the robot. The feedback originating from the sensory input of the robot is evaluated to be used by the local learner to modulate the policy and controller module.

the RLUs in order to enable knowledge exchange and model training. To store the data permanently the 3rd party software *MongoDB* is used.

A *local Database* module keeps track of all RLU related information. It stores taught positions, environment information and logs all data transferred between the modules. After the RLU is done with its current task, this information is collected to investigate the performance of the deployed algorithms.

C. Execution Protocol

This section will explain the intended use and general idea behind the PD.RAI system on the example of a new black-box optimization algorithm. Figure 5 serves as a illustration of this section.

In order to test and compare a new black-box optimization algorithm in the domain of robotics, additional parts like robot hardware or a simulation environment, manipulation policy or motion generator and controller as well as some performance evaluation are needed. With this full stack deployment, the optimization algorithm can be put to work and optimizing parameters of the other modules.

With the help of PD.RAI, a full stack software environment can be put together as shown in the Fig. 5. The first step is to configure the problem statement: Which modules should to test the algorithms? What kind of movement policies do we include with what kind of controllers?

In the second step, the learning configuration is defined. A number of target goals can be chosen, that the robot should solve. For resetting the environment to the initial state, a set of motions has to be defined. A quality metric specifies how the feedback is evaluated and for which value, like time, energy consumption or forces, we want to optimize for.Dependent on the optimization algorithm it is possible to start with initial knowledge or a pre-trained model. If such a model is available on the **Skill Database**, it can be used as initial starting point. Additionally, it is possible to configure the communication between the agents $\boldsymbol{I}_m^n$, meaning what kind of knowledge they share over the **Skill Database**. Also the the strategy for task scheduling and whether the whole system should run in a optimization loop.

With these configurations, the experiment can be started as a third step. To deploy the desired goals $\boldsymbol{g}_n$ to the robots, a setup protocol has to followed:

1) Setup the RLU's evironment. This includes placing objects and tools at the RLUs table and log their position into RLU's local database. 2) teach relevant position including safe retraction poses. 3) check the available knowledge access of the RLU to avoid contamination with extraneous knowledge 4) check the state of all RLUs participating to the experiment 5) deploying the configurations and goals $\boldsymbol{g}$ to the RLUs. Starting and monitoring the software modules is done with container orchestration tool kubernetes [36]. The configurations are sent via one of the supported protocols. For now, these steps are done manually. When the system will be scaled up, those steps have to be automated.

The RoCI module makes sure that the no RLU runs idle and the individual goals $\boldsymbol{g}_n$ are distributed according to the scheduling strategy, see Fig. 3. Therefore, it monitors the state of each RLU, denoted by $\boldsymbol{s}_{RLU}$. If additional human teaching or intervention is part of the experiment, the RoCI relays the teleoperation input to the dedicated RLU.

While the RLUs are learning to reach they individual goals, the data is collected first on a local Database and later merged with the global accessible **Skill Dataset**. Information that is collected are dependent on the applied modules, but in general, it consists of the trajectories and velocities in join and cartesian form, the force and torque profiles, performance evaluations, learner trials as well as the information entering the RLU. A trial of a learner refers to a single change in policy and controller parametrization $\boldsymbol{\theta_\pi}$ and $\boldsymbol{\theta_c}$. The trials that lead to success in reaching the goal $\boldsymbol{g}$ form solution goal dependent solution distribution.

On the **Skill Dataset** the collected informations are merged in order to create skill models or knowledge that can be used as pre-training or initial start for new goals. With all the data in this dataset in place it is possible to draw conclusions on the performance of the tested modules. How did the new black-box optimization perform against

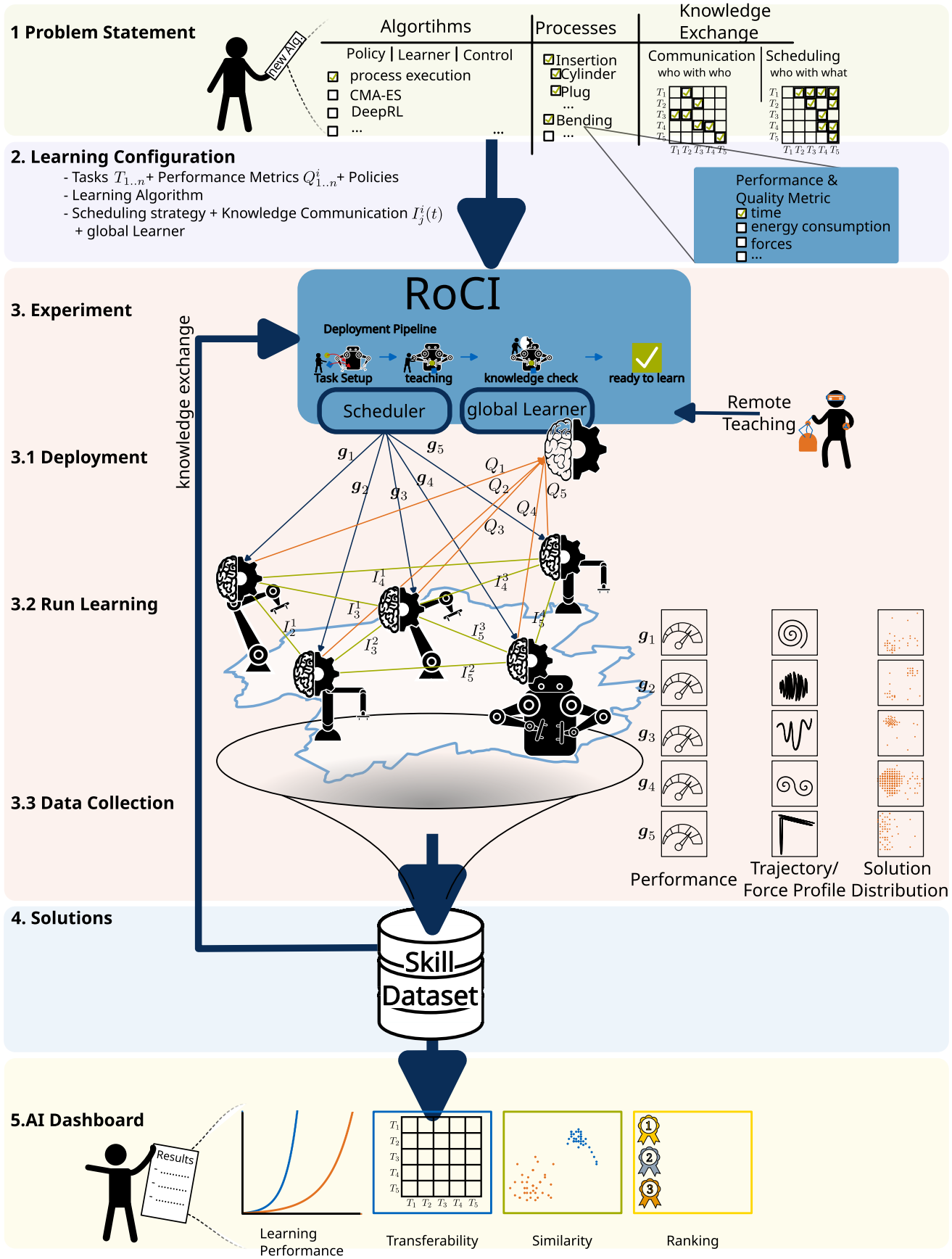

Fig. 5. Intended use and general concept of the PD.RAI platform. The system is intended for large scale experiments where constellation of system components can be easily modified and configured. Together with a large set of default modules complementing an algorithm to a full stack robot deployment this results in a data collection and benchmarking platform.

the default ones? With this benchmarking as fifth step, not just the performance of the algorithm can be evaluated, but also the found solutions can be compared for similarities. With the reusing of knowledge from the **Skill Database** on new goals, even a transferability coefficient can be estimated. Those benchmarking tools will be discussed in section IV.

III. Reference Platform

In this section, the default modules are described that build a reference platform for the testing of new algorithms. It is subdivided into single modules that come with default implementations.

For the control of a single robot of the RLU, the graph-base skill formalism [37] is deployed to cover **Policy** and the **Controller** module.

As the **Controller** modules can be loaded with different kinds of controllers, it also accepts different desired action formats. Dependent on the controller the input formats can take desired joint angels $\boldsymbol{q}_d(t)$, desired cartesian pose $\boldsymbol{T}_d$ or desired twist $\boldsymbol{x}_d$, desired joint velocities $\dot{\boldsymbol{q}}_d(t)$ or desired cartesian velocity $\dot{\boldsymbol{x}}_d$. All controllers also available in a dualarm version. The **Controller** module takes care that the output in the form of torques $\tau(t)$ is valid and comply with safety constraints defined in the task space of the goal $\boldsymbol{g}$ or the joint space.

For the **Policy** module, the used formalism comes with a set of different robot skills that are based on physical manipulation processes. It dissects the robot's action space into tailored sub-policies, each designed for specific processes and governed by a set of parameters $\boldsymbol{\theta}_\pi$. As input, the **Policy** module takes the current percept $\boldsymbol{y}(t)$ and outputs a desired position, velocity or wrench that is fed directly into the **Controller**. In this framework, a skill represents a graph-based amalgamation of elementary movement or manipulation strategies. Each strategy encompasses three critical conditions: $\mathcal{C}_{suc}$, $\mathcal{C}_{err}$ and $\mathcal{C}_{pre}$. The strategy terminates when either $\mathcal{C}_{suc}$ or $\mathcal{C}_{err}$ is met, and a strategy is loaded based on the fulfillment of $\mathcal{C}_{pre}$. Based on these condition, the strategies can be connected to form an action graph that is capable of complex manipulation and reaction to the environment.

With this setup, the graph-based formalism fits perfectly into the hierarchical architecture of PD.RAI and provides the default policies in the form of process graph manipulation skills [37]. These skills can be seen as possible general purpose solutions to reinforcement learning for small tactile problem statements defining actions for states. Currently, there are 16 different pre-built and validated skills available for PD.RAI, see Fig. 6. For other problems, a reinforcement learning approach could be used to find to find a feasible solution.

Even though, camera input is available for the RLUs, the default policies are not dependent on camera input. A robust general purpose computer vision module that can be used as default for this framework does not yet exist.

Policy and controller parameters $\boldsymbol{\theta_\pi}$ and $\boldsymbol{\theta}_c$ can be set in the goal configuration $\boldsymbol{g}$ or can be optimized by the **local Learner**.

The **local Learner** module can deploy different learning and optimization algorithms. It serves as a parent class,

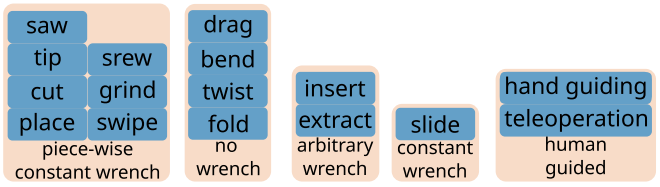

Fig. 6. Process based robot manipulation skill [37] used as default policies in PD.RAI. The skills are ordered according to a process-centric manipulation taxonomy

providing pre-trained models or initial believes, normalizing, de-normalizing functions and transforming functions to bring the data into a learner friendly format. For initializing the learning process, the **local Learner** can request knowledge from the globally reachable **Skill Dataset**. This knowledge has the form of $\boldsymbol{\theta^*} = [\boldsymbol{\theta}_\pi^{*\top}, \boldsymbol{\theta}_c^{*\top}]^\top$ with the parametrization of the policy and controller. If other then the default policies or controllers are used, this knowledge vector has to be adopted according to the configurations. If initial knowledge is available, it is applied to the optimization algorithm. As default, the PSP algorithm [38] is used, but other algorithms, like the CMA-ES [39] are available. The PSP algorithm was modified to accept external suggestions during the learning phase in order to redirect the learning direction.

The **local Learner** bases its algorithms on the quality metrics $\mathcal{Q}$ for a given sensory input $\boldsymbol{y}(t)$ resulting form the policy and controller parametrization $\boldsymbol{\theta} = [\boldsymbol{\theta}_\pi^\top, \boldsymbol{\theta}_c^\top]^\top$ If no other specification is present in the configuration, the **Quality Evaluation** will judge the manipulation performance according to

$$\mathcal{Q}_{total} = \frac{t_{\text{exe}}}{t_{\max}} + \phi \cdot \mathrm{e}^d$$

with t_{exe} and $t_{\max}$ represent the execution time and the maximal time limitation (by default $t_{\max} = 5$ seconds) to reach a goal. The Boolean value ϕ indicates the success in reaching the goal $\boldsymbol{g}$ according to the percept $\boldsymbol{y}(t)$.

$$\phi = \begin{cases} 1 & \text{if the trial is not successful} \\ 0 & \text{if the trial is successful} \end{cases}$$

The distance d between the goal pose of the manipulated object and the current pose of the object is used to push the **local Learner** towards the desired goal. The manipulated object is for example a key and the goal pose is the key inside the lock.

IV. Case Study

To demonstrate the PD.RAI platform's feasibility to generate large-scale real-world data and benchmark robot learning algorithms, in this section, we compare the performance of two optimization algorithms. The goals are to solve 24 tight-clearance insertion tasks as they are still seen as challenge in the automated manufacturing industry. The algorithms are deployed to RLUs together with the default modules from section III.

The two black-box optimizers, PSP [38] and CMA-ES [39], are leveraged to search for the optimal parameters during the learning process. For the rigor of the results, the learning processes for both optimizers are performed 10

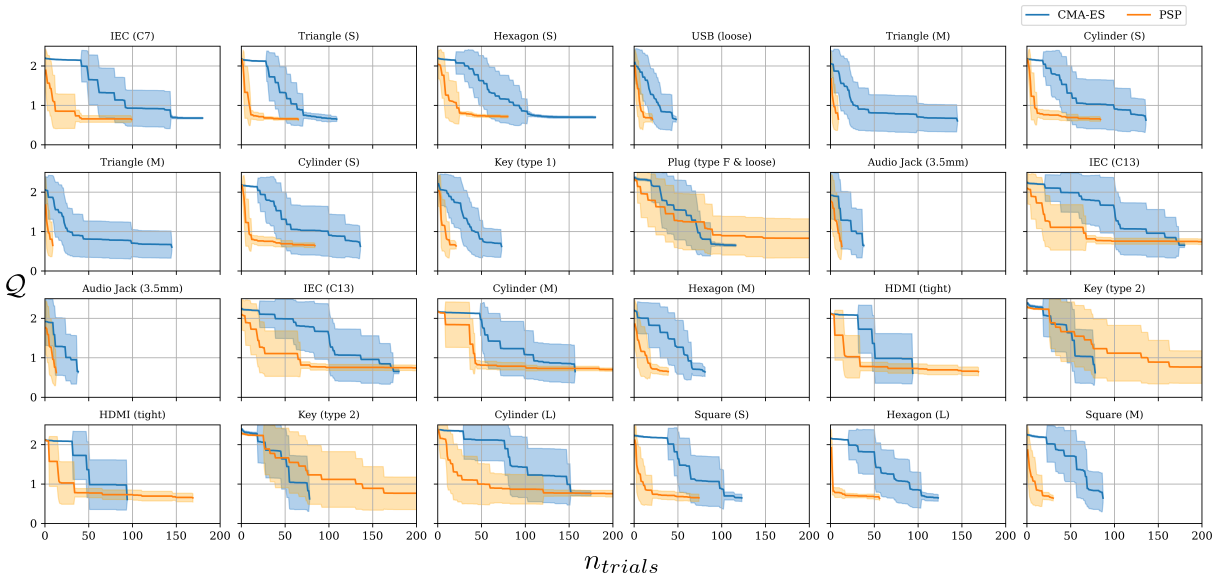

Fig. 7. Monotonically decreasing costs of 24 different insertion goals using CMA-ES (blue) and PSP (orange). The transparent areas mark the standard deviation.

times on 24 distinct tasks.For every goal, a quality threshold $\in \mathcal{Q}$ is set according to expert knowledge. During this process, the PD.RAI platform dynamically assigns goals to RLUs based on the system status and the RLUs' availabilities.

The results are shown in Fig. 7, where the curves illustrate the monotonically decreasing cost throughout the learning process. This means the resulting time series always show the lowest so far seen cost value as it progresses. The blue curve represents the learning process of CMA-ES and the orange line the one of PSP.

For most of the goals, we can see that the PSP is faster in finding solutions but for some tasks reluctant to further optimize the found solution. The CMA-ES on the other hand is robustly finding solutions. Even though the CMA-ES is often slower than the PSP is showing the ability to steadily reducing cost $\mathcal{Q}$. This result is only visible by comparing optimizations on several different goals, what the PD.RAI supports.

This can be explained by the explore-exploit tradeoff. The PSP has an inherent focus on successful trials which leads to a fast convergence, but a inability for further exploration. For some goals, this lead to a sup-optimal solution.

To conclude, the PSP can be used to find sufficient solutions fast, while the strength of CMA-ES lies in optimizing to lower thresholds. This case study is just using a small subset of the capabilities of the PD.RAI platform. Next step is to investigate the impact of the task execution sequence with respect to transfer abilities as well as the relations of number of goals and number of agents used. Therefore the number of goals will be drastically increased. The datasets of this and other experiments are going to be released.

V. CONCLUSION

The developed framework, PD.RAI, opens the door to a wide array of training, testing, and deploying robot learning algorithms on a state-of-the-art platform. This ranges from high quality data collection including tactile information to training models and transferring knowledge onto new goals. Additionally, PD.RAI offers a variaty of default modules in order to provide a full stack deployment for robot software development and a realistic environment for new algorithms to be tested in. This leads to a better ability to compare new approaches.

The platform is envisioned to become a benchmarking system that can be extended by other labs in a cooperative fashion and is used as basis for future robot software development. Therefore, PD.RAI is designed to be a dynamic AI testing and exploration space for robotic manipulation, where various algorithms and learning strategies converge and interact, fostering innovation and collaboration within the field.

In future works, the PD.RAI will first be used for large scale experiments to evaluate different transfer learning strategies and see what works best for a big number of tasks. Currently, a growing tactile skill database is build with a wide range of skills for robots to learn from that are covering wide applications of industrial manufacturing. Next steps involve extensive experimental testing of the collective learning paradigm, aiming to show how robots acquire, share and synthesize new skills in a network of robots. Investigating the role of task scheduling and goal assignment for the learning system, as well as determining optimal communication strategies, are central questions that will be addressed.

ACKNOWLEDGMENT

The authors acknowledge the financial support by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK, (Phase 1: Infrastructure as well as the research and development program under grant no. DIK0249).

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany

(BMBF) in the programme of "Souverän. Digital. Vernetzt." Joint project 6G-life, project identification number 16KISK002.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test*, R. Epstein, G. Roberts, and G. Beber, Eds. Springer Netherlands, 2009, pp. 23–65.

[2] C. E. Shannon, "Programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] T. Gupta, W. Gong, C. Ma, N. Pawlowski, A. Hilmkil, M. Scetbon, *et al.*, "The Essential Role of Causality in Foundation World Models for Embodied AI," 2024, arXiv:2402.06665.

[5] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, "Embodied intelligence via learning and evolution," *Nature Communications*, vol. 12, no. 1, p. 5721, 2021.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[7] OpenAI, "GPT-4 technical report," 2024.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763.

[9] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, *et al.*, "Foundation models in robotics: Applications, challenges, and the future," 2023.

[10] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 894–906.

[11] N. Di Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller, "Towards a unified agent with foundation models," in *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, version: 1.

[12] b. ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Proceedings of The 6th Conference on Robot Learning*, vol. 205. PMLR, 2023, pp. 287–318.

[13] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, and J. e. a. Tremblay, "ProgPrompt: program generation for situated robot task planning using large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 999–1012, 2023.

[14] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[16] E. Collaboration, A. Padalkar, A. Pooley, A. Mandlekar, A. Jain, and A. T. et al., "Open x-embodiment: Robotic learning datasets and rt-x models," 2023.

[17] S. Dankwa and W. Zheng, "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ser. ICVISP 2019. New York, NY, USA: Association for Computing Machinery, 2020.

[18] A. A. Shahid, L. Roveda, D. Piga, and F. Braghin, "Learning continuous control actions for robotic grasping with reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 4066–4072.

[19] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[20] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6284–6291.

[21] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396.

[22] 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[23] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, *et al.*, "Factory: Fast contact for robotic assembly," *arXiv preprint arXiv:2205.03532*, 2022.

[24] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, Apr. 2018.

[25] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, *et al.*, "Autort: Embodied foundation models for large scale orchestration of robotic agents," *arXiv preprint arXiv:2401.12963*, 2024.

[26] A. Herzog, K. Rao, K. Hausman, Y. Lu, P. Wohlhart, *et al.*, "Deep RL at Scale: Sorting Waste in Office Buildings with a Fleet of Mobile Manipulators," in *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023.

[27] M. Zahid and F. T. Pokorny, "Cloudgripper: An open source cloud robotics testbed for robotic manipulation research, benchmarking and data collection at scale," *arXiv preprint arXiv:2309.12786*, 2023.

[28] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, *et al.*, "Robothor: An open simulation-to-real embodied ai platform," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3164–3174.

[29] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8494–8502.

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

[31] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[32] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.

[33] National Institute of Standards and Technology, "Robotic Grasping and Manipulation Assembly," https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly, 2024, [Accessed: 03-13-2024].

[34] P. So, A. Sarabakha, F. Wu, U. Culha, F. J. Abu-Dakka, and S. Haddadin, "Digital Robot Judge: Building a Task-centric Performance Database of Real-World Manipulation With Electronic Task Boards," *IEEE Robotics & Automation Magazine*, pp. 2–14, 2024.

[35] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, *et al.*, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.

[36] D. K. Rensin, *Kubernetes - Scheduling the Future at Cloud Scale*. 1005 Gravenstein Highway North Sebastopol, CA 95472: O'Reilly Media (May, 2015); eBook (Compliments of Red Hat), 2015.

[37] 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)*, 2019, pp. 5844–5850.

[38] F. Voigt, L. Johannsmeier, and S. Haddadin, "Multi-level structure vs. end-to-end-learning in high-performance tactile robotic manipulation," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 2306–2316.

[39] N. Hansen and A. Auger, "Cma-es: Evolution strategies and covariance matrix adaptation," in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 991–1010.