`Sapience`: The ability to know things and reason with that knowledge

Vikash Kumar* Aravind Rajeshwaran*

University of Washington

Abstract

We introduce `Sapience`, a new software package and ecosystem for research in *Embodied Artificial Intelligence.* `Sapience` consists of a set of environments that span prior work of the authors, as well as novel environments. These include environments like dexterous hand manipulation with a Shadow Hand, whole arm manipulation tasks with robots like Franka and Fetch, locomotion tasks with a variety of quadrupeds, as well as environments representative of challenges in multi-task, hierarchical, and multi-agent RL. `Sapience` also comes with a number of algorithms developed and optimized by the authors including model-free Natural Policy Gradient (NPG), demonstration augmented policy gradient (DAPG), PAL and MAL – versions of model-based NPG based on a *game theoretic formulation* of MBRL, and MOReL – a model-based offline RL algorithm. By unifying a number of algorithms into a common API, and in providing more advanced environments that are representative of real-world challenges in robotic control, we hope to make adoption easier and advance the field of embodied artificial intelligence.

1 Introduction

While life is unique to the planet earth, the variability of its existence within our ecosystem is overwhelmingly diverse. Different life forms have evolved to perceive their environment in different ways. Similar diversity is found in the way they interact with the environment as well. Such a large variance is a testimony to the fact that no sentient being is a universal winner. Different forms have different advantages based on the ecosystem they exist in. Additionally, changes in the ecosystem test their adaptability and resistance – some thrive while the other perishes. The path to Artificial Embodied intelligence is no different. Much like biological intelligence, Artificial Embodied intelligence is a complex puzzle with multiple design principles in play. These principles that can't be understood in isolation. Studying these principles and their interdependence will be key to realize the promise of Robotics. To facilitate this investigation we present `Sapience` with the goal of providing a flexible and comprehensive ecosystem for the study of Embodied Intelligence.

Sapience (Noun): - Quality of being wise, or wisdom. - Ability to apply knowledge or experience or understanding or common sense and insight - The ability to know things and reason with that knowledge.

Sapient (Adjective) - wise, or attempting to appear wise. - being capable of experiencing things through its senses.

[Sentience vs Sapience] [1] is the distinction between being aware in the sense of being merely awake (which we share with nondiscursive animals - those that do not grasp concepts), on the one hand, and, on the other hand, being aware in a sense that involves knowledge either by being a kind of knowledge, or as potentially serving to justify judgments that so qualify.

**Equal contributions

2 Philosophy and Ecosystem

Talk about our principles/approach, and the project organization

3 Algorithms

`Sapience` comes with an extensive set of algorithms that span multiple families (e.g. model-free, model-based, MPC). We plan to include additional algorithms in the future. All algorithms are based on the interaction protocol (inspired by OpenAI) followed in the `Sapience` environments. All algorithms share the following components:

- **Policy:** The policy class parameterizes the conditional distribution $P(\boldsymbol{a}|\boldsymbol{s}) = \pi_\theta(\boldsymbol{a}|\boldsymbol{s})$. This class supports basic functions to sample actions from the conditional distribution, compute the likelihood, and other distributional information like entropy.
- **Critic:** This class parameterizes either the value function $V_\phi(\boldsymbol{s})$ or the action-value function $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$. This class is utilized for Q-learning based algorithms, policy gradient algorithms (as a baseline), as well as trajectory optimization algorithms (e.g. POLO [2]).
- **Model:** The model, broadly parameterized as $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = T_\psi(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, is utilized in a number of algorithms like model-based NPG [3], trajectory optimization based algoirithms like POLO [2] and PDDM [4], as well as offline RL algorithms like MOReL [5].
- **Sampler:** The sampler module supports fast sampling from: (a) simulators running in the CPU like MuJoCo [6] through `multiprocessing`; (b) fast rendering in GPU for pixel based tasks; (c) learned neural network dynamics models typically running in GPU. We utilize a common API for sampling in all these cases to ensure algorithms are general purpose and separated from the data collection process.
- **Logger:** We use a custom logger class that logs data as key-value pairs in an ordered dictionary. The logs are periodically saved as both `.pickle` and `.csv` files. This enables the use of either our custom visualization functions, or external packages like `tensorboard`.

Based on the above abstractions, we provide the following algorithms initially, with the plan of including more algorithms in the future.

- **Model-free policy gradient:** In the family of model-free policy gradients, we provide an efficient implementation of NPG [7, 8], TRPO [9], and PPO [10].
- **Model-based RL:** In the family of model-based RL algorithms, we provide extensions of the above algorithms (most emphasis on NPG), to the model-based RL seting. Here, a dynamics model is learned using data of interaction with the environment. Subsequently, policy learning happens through synthetic trajectories from the learned model. In our prior work, through a game theoretic formulation of MBRL [3], we point out that for such model-based RL algorithms to be successful, a two-timescale separation between model learning and policy learning is required. The resulting algorithms, called PAL and MAL, are provided with `Sapience`, which to our knowledge are currently the most sample efficient algorithms for the OpenAI gym [11] tasks. The model-based NPG algorithm can also be augmented with a pessimistic truncation function resulting in MOReL [5], a state of the art offline RL algorithm.
- **Trajectory optimization (MPC):** We also provide the MPC algorithm presented in POLO [2], which is closely inspired by MPPI [12] (but has subtle differences). This MPC algorithm can be used either directly with a simulator model (useful for sim2real experiments) or with a learned dynamics model like in PDDM [4]. The MPC algorithm can also be augmented with a learned terminal value function resulting in the POLO algorithm [2].

4 Robot

Sapience supports a collection of robots from different domains.

- Hands: Adroit Hand, Shadow Hand, Allegro Hand, MPL hand, D'Hand
- Arms: Fraka Arm, Sawyer Arm, Fetch
- Bi-Manual: Sally, Bi-Franka
- Quadrupeds: D'Kitty, D'Neko, Spot Mini, MIT-Cheetah
- Bipeds: Darwin

These robots are exposed via an abstract *robot* class that homogenizes the simulation and hardware differences. The *robot* class has two backends *robot-simulation* and *robot-hardware*. The *robot-simulation* class uses MuJoCo backend and exposes all details with real world constrains (delays, noise, etc). The *robot-hardware* class exposes robots in the real world via respective drivers. The *robot* class supports multiple control modalities – position, velocity, torque control, etc, that is easily configured via a config file.

4.1 Simulation

Developed using MuJoCo engine with physical realism. Modular. Building Blocks. Physically realistic, Multiple sensing modalities

4.2 Hardware

Provides base class for users to extend as per the hardware configurations they have access to.

5 Environments

Sapienceprovides a variety of physically realistic environment carefully designed to cater to various challenges associated with embodied intelligence. The environments are exposed via the OpenAI Gym API [13]. Environments are designed with keen attention for flexibility and extension. For example, tasks can be spawned with different robots, rewards and observation can be flexibly chosen. Additionally all environments support physic and visual randomization. While rewards are usually taken as a measure of performance, its scale can be arbitrary. Additionally, we realize that reward-design and reward-learning is also an integral component of behavior systhesis. To facilitate such endeavors and to introduce additional rigor in measuring performance, all environments are additionally equipped with functionality to evaluate if the task has – failed/ solved. Success percentages reported by the environments are used as performance measures. Unlike rewards, which can take arbitrary values, success percentages are human interpretible and lies within the range 0-100%. Task success as the performance measure is more aptly aligned with the true objectives of successful behaviors, and accommodates reward-learning efforts within the scope of Sapience.
Sapience environments are organized into various suites.

5.1 Classical Suite

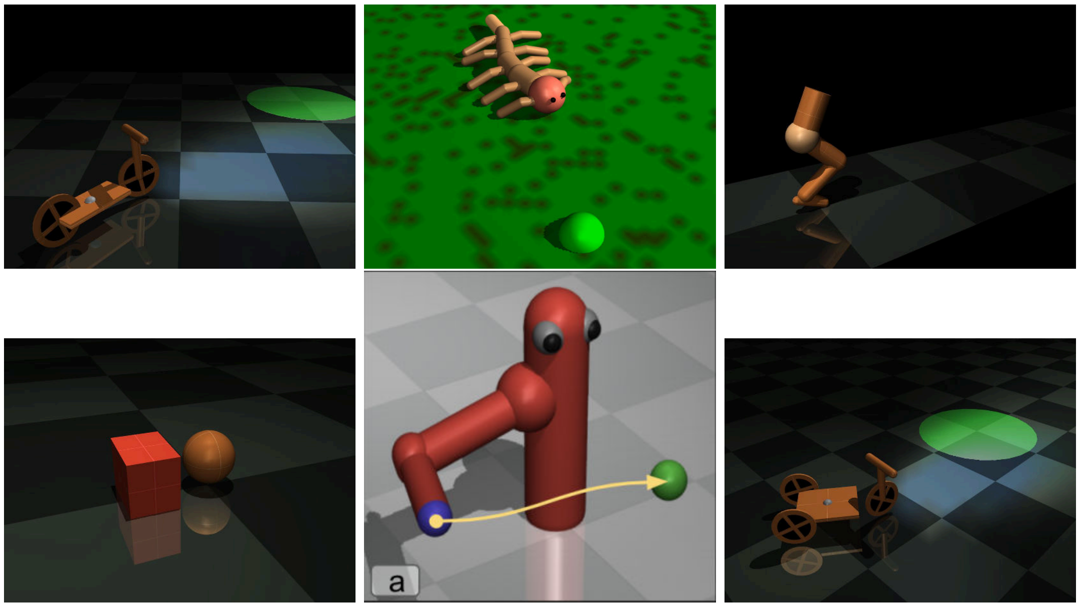

Figure 1: Task suite of classical control problems

5.2 Hand Manipulation Suite

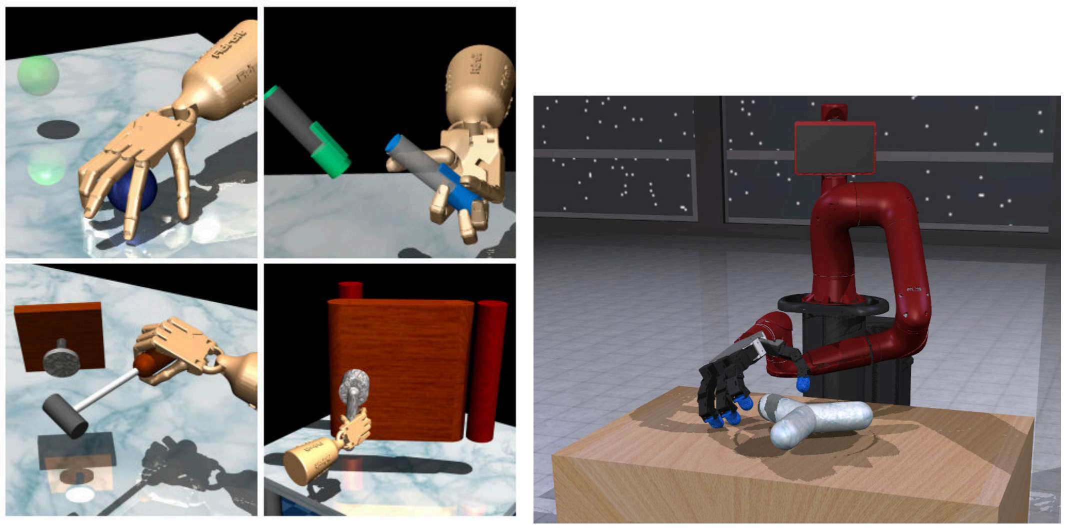

Figure 2: Task suite of dexterous manipulation skills such as object relocation, in-hand manipulation, tool use, and opening doors [14]

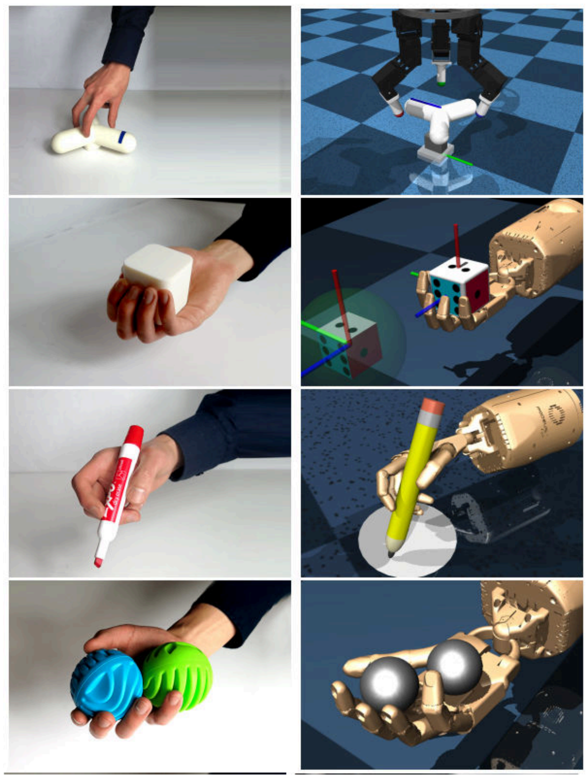

Figure 3: Task suite of simulated and realworld dexterous manipulation: valve rotation, in-hand reorientation, handwriting, and manipulating Baoding balls as introduced in [4]

5.3 Arm Manipulation Suite

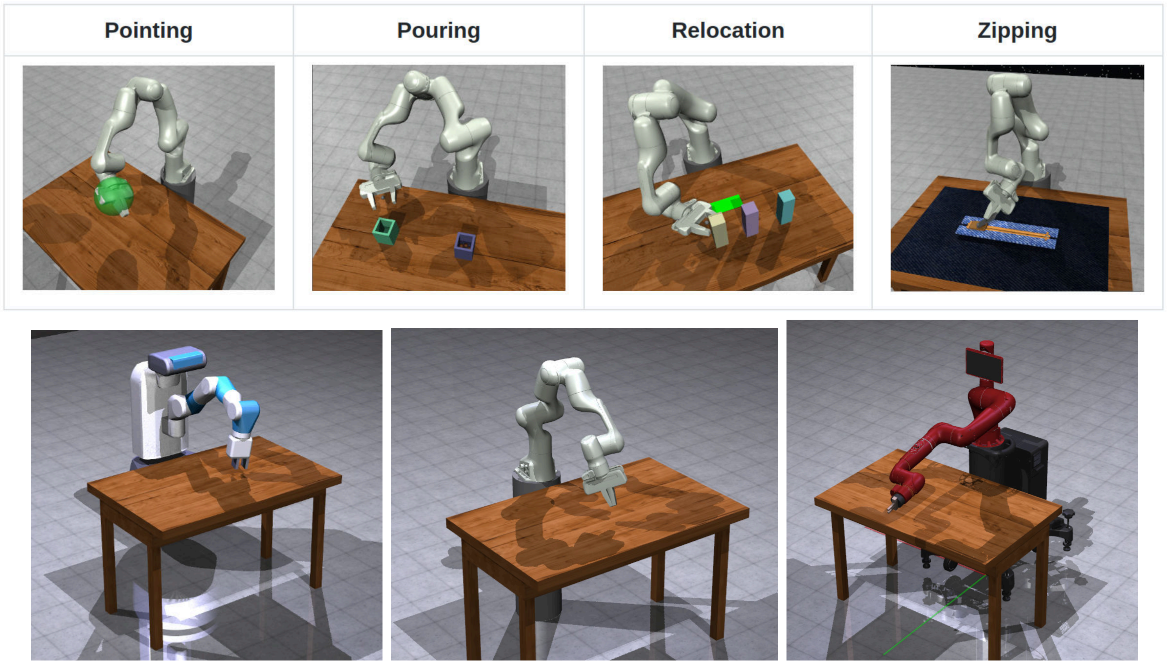

Figure 4: A task suite for arm manipulation skills

5.4 Bi-manual Manipulation Suite

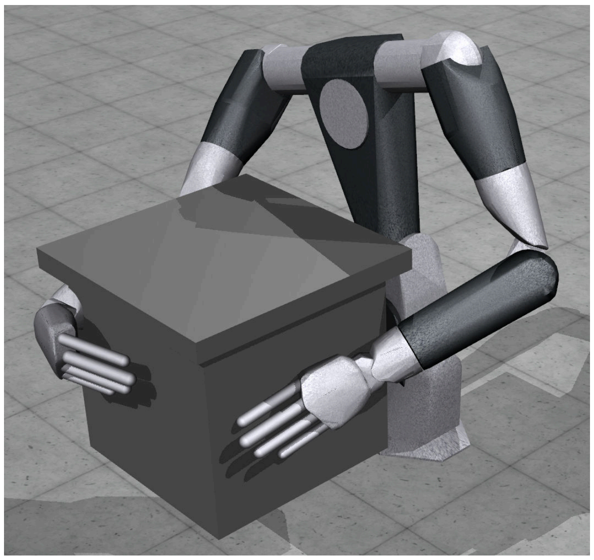

Figure 5: Task suite for bi-manual manipulation skills

5.5 Deformable Manipulation Suite

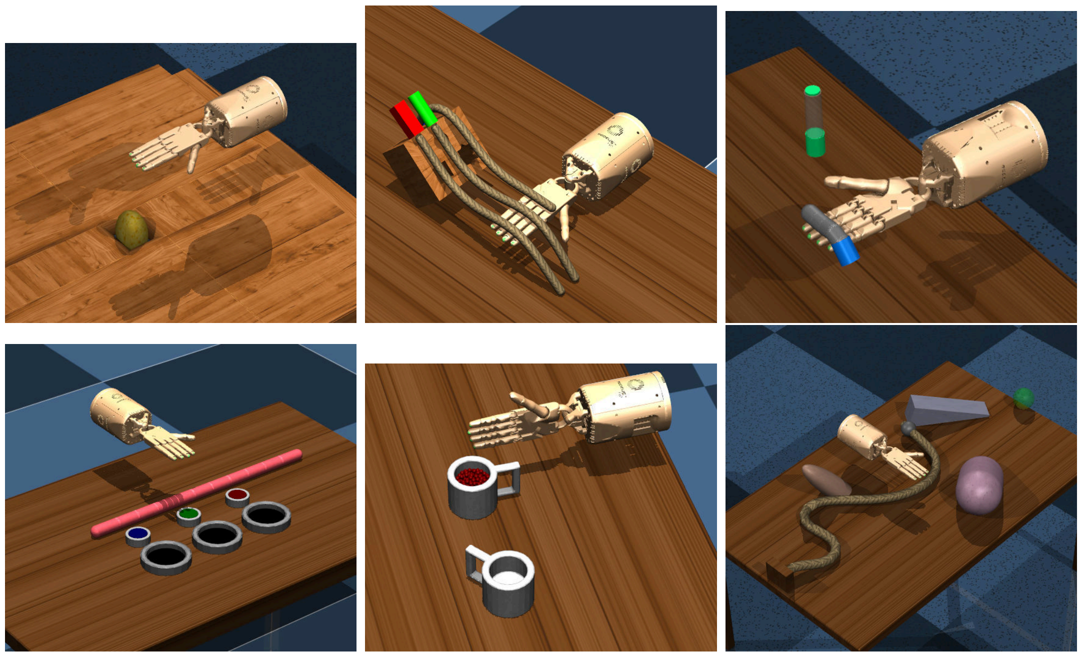

Figure 6: Task suite for learning manipulation skills with flexible objects

5.6 Locomotion Suite

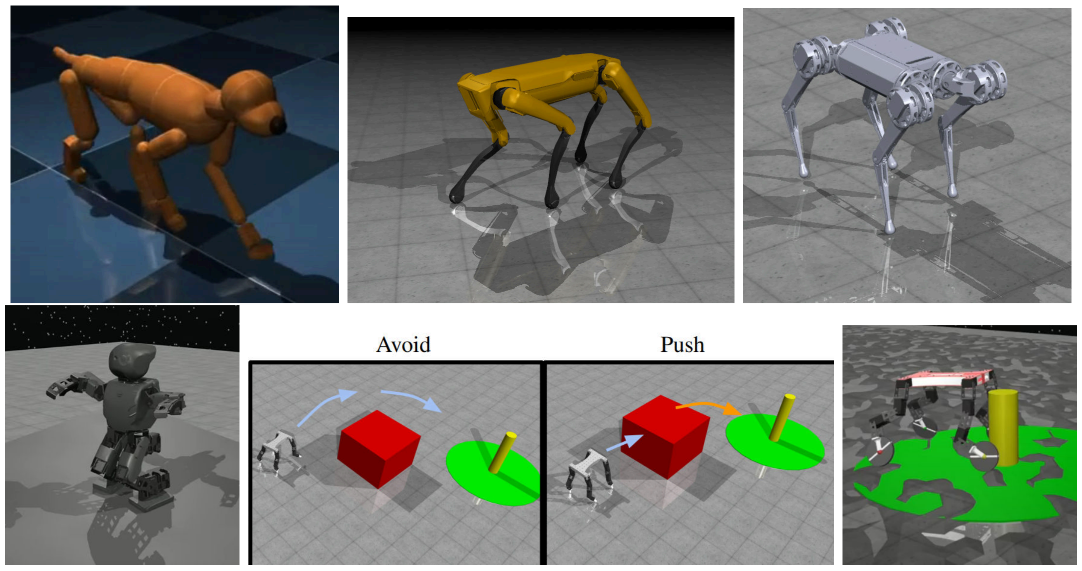

Figure 7: Task suite for agile locomotion skills

5.7 Multi-Task Suite

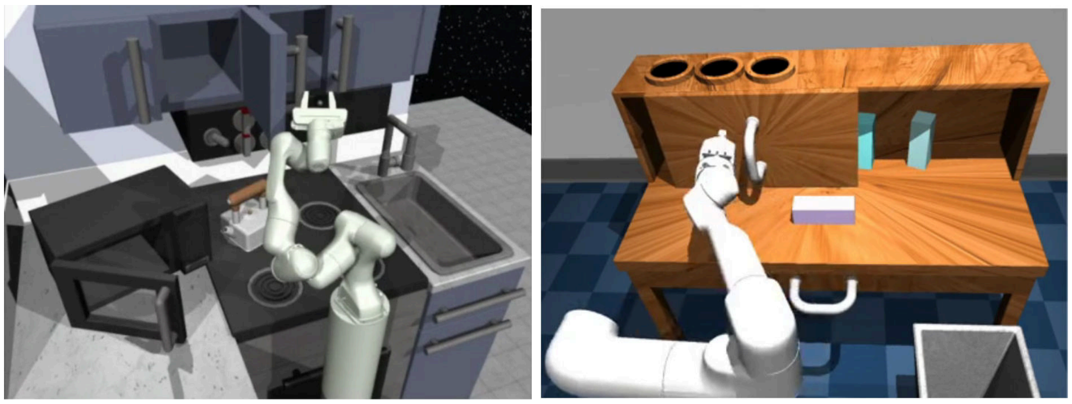
Figure 8: A simulated kitchen scene (left)(as introduced in [15]) and table scene (right)(as introduced in [16]) well suited for multi-task learning

5.8 Multi-Agent Suite

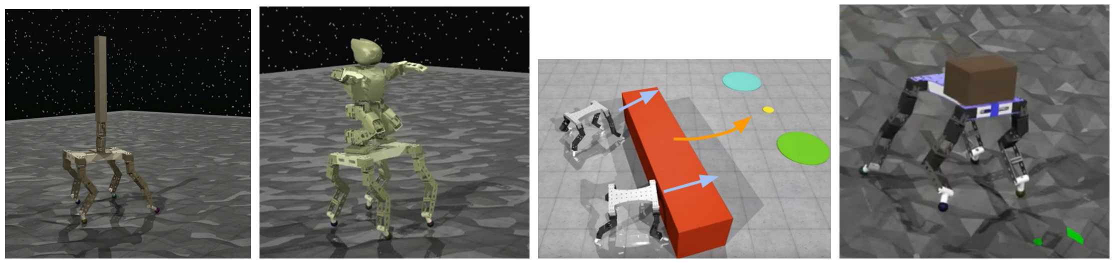
Figure 9: Task suite for collaborative skills

5.9 Robel Suite

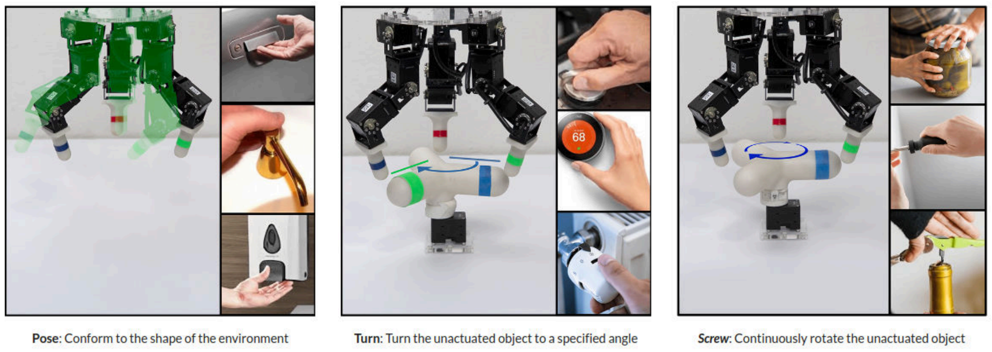
68
Pose: Conform to the shape of the environment
Turn: Turn the unactuated object to a specified angle
Screw: Continuously rotate the unactuated object

Figure 10: Task suite of dexterous manipulation skills such as posing, turning, rotating etc as introduced in [17]

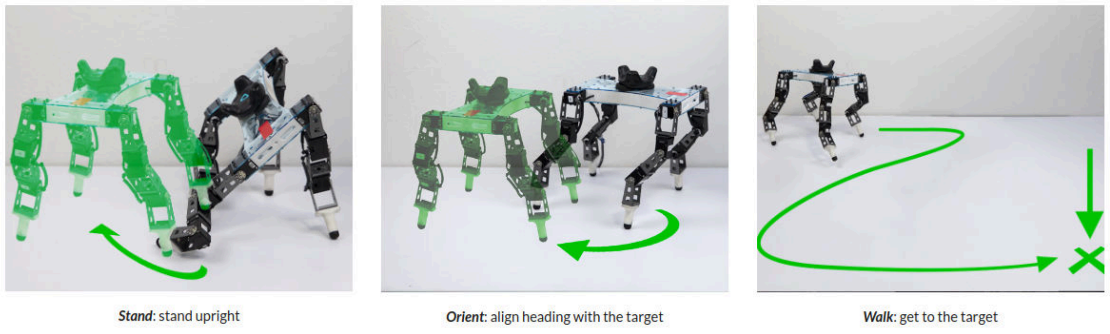

Figure 11: Task suite of agile locomotion skills such as posing, turning, rotating etc as introduced in [17]

6 Evaluations and Benchmarks

Talk about our evaluations are done – success percentages, not rewards. Users are free to pick rewards however they like.

7 Conclusion

Hope the ecosystem provides a firm footing for those who are getting started with the field as well as those who are at the forefront of research and development.

References

[1] Wilfrid Sellars, Richard Rorty, Robert Brandom, et al. *Empiricism and the Philosophy of Mind*, volume 1. Harvard University Press, 1997.

[2] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

[3] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A Game Theoretic Framework for Model Based Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[4] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

[5] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-Based OfflineReinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[6] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[7] Sham M Kakade. A natural policy gradient. In *NIPS*, 2002.

[8] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards Generalization and Simplicity in Continuous Control. In *NIPS*, 2017.

[9] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *ICML*, 2015.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[12] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos Theodorou. Information theoretic mpc for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[14] Aravind Rajeswaran*, Vikash Kumar*, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[15] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.

[16] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019.

[17] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: RObotics BEnchmarks for Learning with low-cost robots. In *Conference on Robot Learning (CoRL)*, 2019.