ROBEL: Robotics Benchmarks for Learning with Low-Cost Robots

Michael Ahn[†] **Henry Zhu**[δ] **Kristian Hartikainen**[δ] **Hugo Ponte**[†]

Abhishek Gupta[δ] **Sergey Levine**[δ†] **Vikash Kumar**[†]

[δ]**UC Berkeley, USA** [†]**Google Research, USA**

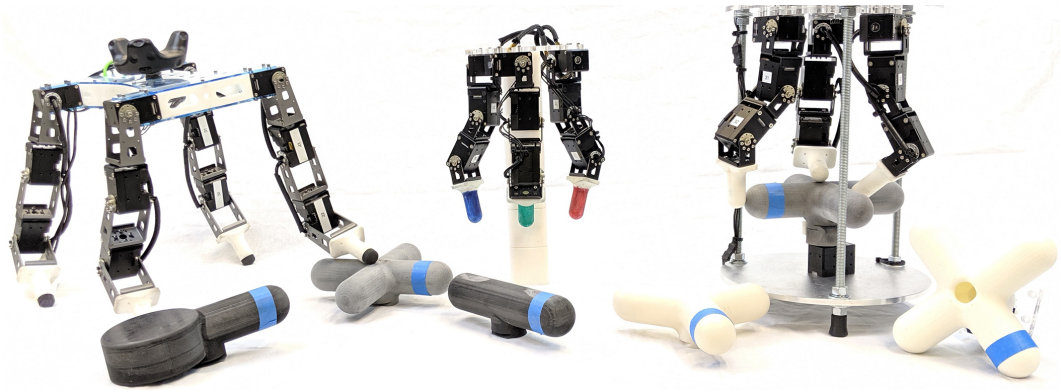


Figure 1: *ROBEL* robots: *D'Kitty* (left) and *D'Claw* (middle and right)

Abstract: ROBEL is an open-source platform of cost-effective robots designed for reinforcement learning in the real world. ROBEL introduces two robots, each aimed to accelerate reinforcement learning research in different task domains: D'Claw is a three-fingered hand robot that facilitates learning dexterous manipulation tasks, and D'Kitty is a four-legged robot that facilitates learning agile legged locomotion tasks. These low-cost, modular robots are easy to maintain and are robust enough to sustain on-hardware reinforcement learning from scratch with over 14000 training hours registered on them to date. To leverage this platform, we propose an extensible set of continuous control benchmark tasks for each robot. These tasks feature dense and sparse task objectives, and additionally introduce score metrics for hardware-safety. We provide benchmark scores on an initial set of tasks using a variety of learning-based methods. Furthermore, we show that these results can be replicated across copies of the robots located in different institutions. Code, documentation, design files, detailed assembly instructions, trained policies, baseline details, task videos, and all supplementary materials required to reproduce the results are available at www.roboticsbenchmarks.org

Keywords: benchmarks, reinforcement learning, low cost robots

1 Introduction

Learning-based methods for solving robotic control problems have recently seen significant momentum, driven by the widening availability of simulated benchmarks [1, 2, 3] and advancements in flexible and scalable reinforcement learning [4, 5, 6, 7]. While learning through simulation is relatively inexpensive and scalable, developments on these simulated environments often encounter difficulty in deploying to real-world robots due to factors such as inaccurate modeling of physical phenomena and domain shift. This motivates the need to develop robotic control solutions directly in the real world on physical hardware.

Modern advancements in reinforcement learning have shown some success in the real world [8, 9, 10]. However, learning on real robots generally does not take into account physical limitations

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

– aggressive exploration can induce wear and permanent damage to the robot due to collisions with itself and the surrounding physical environment. A significant portion of current robotics research is conducted on high-cost, industrial-quality robots that are intended for precise, human-monitored operation in controlled environments. Furthermore, these robots are designed around traditional control methods that focus on precision, repeatability, and ease of characterization. This stands in sharp contrast with learning-based methods that are robust to imperfect sensing and actuation, but demand (a) a high degree of resilience to allow real-world trial-and-error learning over a long duration, (b) low cost and ease of maintenance to enable scalability through replication, and (c) reliable mechanisms to allow large scale data collection without strict human monitoring requirements for providing rewards and episodic resets.

To address these emerging requirements, we introduce *ROBEL* – an open-source platform for cost-effective, modular robots that are designed around the needs of reinforcement learning in the real world. This release of *ROBEL* consists of two robots that are each intended to accelerate research in a distinct task domain: a nine degree of freedom (DOF) manipulation robot ***D'Claw*** and a twelve DOF locomotion robot ***D'Kitty***. In addition, *ROBEL* includes a wide variety of benchmark tasks that run in the real world and support a simulated back-end to facilitate rapid prototyping. We present performance metrics on these benchmark tasks over a diverse collection of learning-based methods. Finally, we show that these robots are replicable and are able to reproduce desired behavior from a control policy that was trained on a different copy of the robot.

2 Related Work

Before delving into the specifics of *ROBEL*, we first review current work looking into simulated benchmarks, the disconnect between the challenges in simulation and reality, hardware benchmarks, and factors influencing real world progress in relevance to continuous control problems in robotics.

Recent advancements in continuous control problems in robotics via learning based methods are fueled in part by access to fast compute at affordable rates, and in part by algorithmic developments [4, 11] that generalize and scale well with the complexities of high dimensional problems. Access to easy to use simulated benchmarks [12, 3, 1] has significantly catalyzed these developments by facilitating fast prototyping, and by providing standard metrics for analysis and comparisons between various methods.

Various algorithms have been shown to be effective on a large set of simulated environments [6, 7], but these developments have not precipitated down in equal proportions to real world systems, owing to the large divide between the challenges presented by the real world and their simulated counterparts. Precise and programmable resets, noise-free instantaneous observations, high data bandwidth, and lack of concern for environmental safety are a few of the many privileges that the prevalent simulated environments [12, 3, 1] enjoy, which are impractical in the real world. In contrast, *ROBEL* exposes many of these challenges on physical hardware and provides the tools to study them, encouraging future development to directly address these issues.

Algorithms for applying RL to real world robotics have either (a) resorted to solving problems in simulation and transferring to the real world, relying on algorithms like domain randomization to deal with domain shift by solving a more challenging robust control problem [13, 14, 15, 16], or (b) have resorted to completely pose and solve the problem in the real world [9, 17]. While the former does not scale well as the task complexity grows [18, 19], the latter has traditionally required a significant time and cost investment that is task-specific, and is not commonly accessible more broadly to the field. Shared investments in terms of competitive challenges [20, 21, 22] have also been investigated to boost research and developments on physical robots in the real world. These challenges have failed to stay relevant to the scientific community owing to significant costs and reproducibility issues. *ROBEL* alleviates these investments by providing a low-cost, easily extensible platform to facilitate real world results by the broader community under reproducible settings.

Roboticists have long been fascinated by the idea of building low cost manipulation [23, 24, 25] as well as locomotion platforms [26, 27]. Many of these platforms can be limited to few DoFs [23], aggressively under-actuated [24] for simplicity and cost gains, or are difficult to independently assemble and replicate [25, 24]. *ROBEL* leverages its modular design to provide high DOF, easy-to-assemble robots, while retaining easy control and reasonable precision: *D'Claw* has nine actuated DOFs while remaining low cost. On the other end, [27] is perhaps the closest, but more expensive, counterpart of our *D'Kitty* platform.

Although not posed as benchmarks, the idea of comparing progress in the real world via shared datasets [28], testbeds [29], and hardware designs [24, 30, 31] has been around for a while. Recently, benchmarking in the real world using commercially available platforms has also been proposed [32, 33]. These benchmarks include robot-centric tasks such end-effector reaching, joint angle tracking, and grasping via parallel jaws grippers. To further diversify the benchmarking scene, *ROBEL* presents a wide variety of high DoF tasks spanning dexterous manipulation as well as quadruped locomotion.

With learning-based methods [4], it is common to measure the average episodic return to evaluate the performance of an agent. These returns are task-specific and often ignore the challenges of the real world, such as unsafe exploration, movement quality, hardware risks, energy expenditure, etc. These challenges are highlighted by the DARPA Robotics Challenge [20, 21, 22] where many robots failed to achieve their task objective due to undervaluing safety objectives, thereby indicating that real world challenges (such as safety) are important objectives to prioritize. Hardware safety considerations have been posing as explicit constraints (position, velocity, acceleration, jerk limits) as well as regularization (energy, control cost) before, but have not found appropriate emphasis in existing [3, 1, 12] learning related benchmarks. Addressing this, *ROBEL* provides three signals (dense-reward, sparse-score and hardware-safety) to facilitate the study of these challenges.

3 ROBEL

Hardware Platforms

As the number of actuated DOFs of a system grows, we tend to see a proportional increase in cost and decrease in reliability. The modularity of *ROBEL* allows us to build reasonably high DOF robots while remaining low-cost and easily maintainable. The robots only use off-the-shelf components, commonly-available prototyping tools (3D printers, laser cutters), and require only a few hours to build (Table 1). *ROBEL* robots are actuated at joint level (i.e. no transmission between joint and actuators) via *Dynamixel* smart actuators [34] that feature fully integrated motors with an embedded controller, reduction drive, and high-baudrate communication. Multiple actuators can be daisy-chained together to increase the number of DOFs in the system, which allows *ROBEL* robots to be easy to build (Table 1) and extend. For the context of this work we use a USB-serial bus [35] for communication to the robots. An 12V power supply is used to power the platforms. *ROBEL* platforms also support a wide variety of choices in sensing and actuation modes, which are summarized in Table 2.

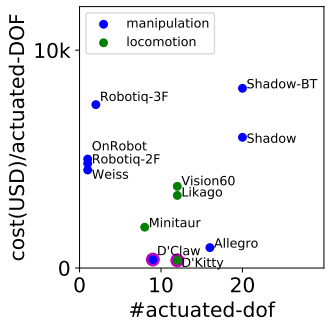


Figure 2: Cost comparison of *ROBEL* with other commonly used platforms. We note that (a) *ROBEL* platforms have the most economical price point, thereby facilitating experiment's scalability and (b) Prices scale linearly with # of DOFs, thanks to modular design, thereby facilitating experiments' complexity

Table 1: *ROBEL* platform initial cost and time investments

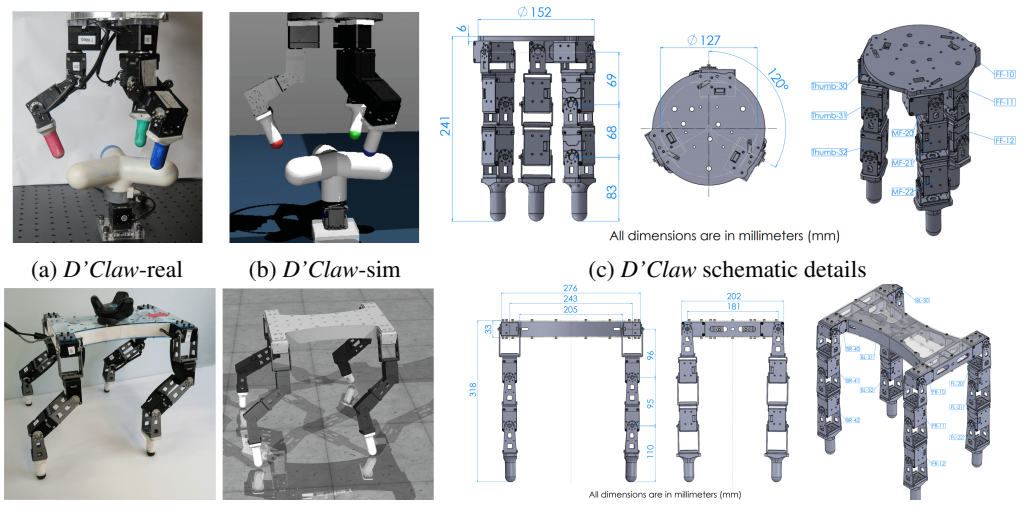
platform	D'Claw	D'Kitty
# DOF	9	12
Price ($)	3500	4200
Build(hr)	4	6

Table 2: *ROBEL* platform features a variety of sensing options, control modes, limits and communication speeds

Property	Options
Control	Torque, Velocity, Position, Extended Position, Current, PWM
Sensing	Position, Velocity, Current, Realtime tick, Trajectory, Temperature, Input Voltage
Limits	Position, Velocity, PWM, Current
Bandwidth	9600 bps $\sim$ 4.5 Mbps

The schematic details of *ROBEL* platforms are summarized in Figure 3. Detailed CAD models and bill of materials (BOM) with step-by-step assembly instructions are included in the supplementary materials package. *ROBEL* platforms have also been independently replicated and tested for reliability (subsection 5.3) at a geographically remote location which demonstrates the reproducibility (details in subsection 5.2) of the *ROBEL* platforms and associated results.

The combination of reproducibility and scalibility exhibited by *ROBEL* platforms presents to the field of robotics a lucrative preposition of a standard set of benchmarks (proposed in section 4) to facilitate sharing and collaborative comparison of results. *ROBEL* consists of the following two platforms:



(a) *D'Claw*-real (b) *D'Claw*-sim (c) *D'Claw* schematic details

(d) *D'Kitty*-real (e) *D'Kitty*-sim (f) *D'Kitty* schematic details

Figure 3: *ROBEL* features two low-cost, robust, modular platforms – *D'Claw* (9 DOF manipulation platform) and *D'Kitty* (12 DOF locomotion platform)

1. ***D'Claw*** (Figure 3a-3c) is a nine-DOF manipulation platform capable of performing dexterous manipulation. It consists of three identical fingers mounted symmetrically on a circular laser cut base. The finger tips are 3D printed parts. The base can be fixed to a stationary position, or mounted to a portable frame. *D'Claw* robots have been featured in wide stream of prior research [10, 7, 36] and have registered 1000s of hours of real world training on them.
2. ***D'Kitty*** (Figure 3d-3f) is a twelve-DOF quadruped capable of agile locomotion. It consists of four identical legs mounted symmetrically on a square base. The feet are simple 3d printed parts with rubber ends. *D'Kitty* is symmetric along all three axes and can also walk normally when upside down.

4 Benchmark Tasks

ROBEL proposes a collection of tasks for *D'Claw* and *D'Kitty* to serve as a foundation for real-world benchmarking for continuous control problems in robotics. We first outline the formulations of these benchmark tasks, and then provide details of the tasks grouped into manipulation and locomotion.

ROBEL tasks are formulated in a standard Markov decision process (MDP) setting [37], in which each step, corresponding to a time t in the environment, consists of a state *observation* s, an input *action* a, a resulting *reward* r_d, and a resulting *next state* s'. In addition to the reward r_d, which is usually dense, *ROBEL* also provides a sparse signal called *score* r_s, which can be interpreted as a sparse task objective without any shaping. To standardize quantification a *policy*'s π performance, *ROBEL* provides *success evaluator* $\phi_{se}(\pi)$ metrics and *hardware safety* $\phi_{hs}(\pi)$ metrics.

To implement the MDP setting, we employ the commonly-adopted OpenAI Gym [2] API. *ROBEL* is presented as an open-source Python library consisting of modular, reusable software components that enable a common interface to interact with hardware and simulation. Figure 4 provides architectural outline of *ROBEL*. The implementation of the environments are largely agnostic to whether they are running on real *hardware robot* or *simulation robot*. The simulated robot is a modular component of the system and can be exchanged for any

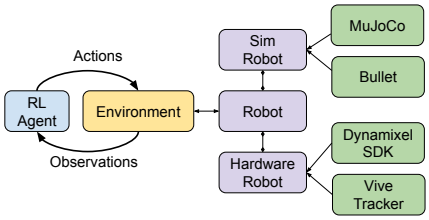


Figure 4: *ROBEL* architecture

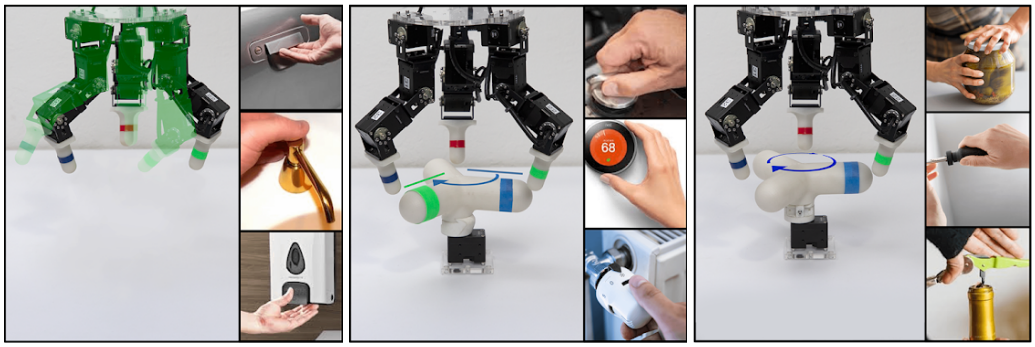
physics simulation engines. Figure 3b, and Figure 3e show the simulated robot modelled in MuJoCo [38]. We encourage the usage of simulation primarily as a rapid prototyping tool and promote purely real-world hardware results as *ROBEL* benchmarks.

The *reward* r_d is the most commonly used signal in reinforcement learning that the agents directly optimize. Since the reward often consists of multiple sub-goals and regularization terms, *score* r_s provides a more direct task-specific sparse objective. *Success evaluator* $\phi_{se}(\pi)$ is defined to be reward (or score) agnostic. It evaluates success (task-specific) percentage of policy over multiple runs. Unlike rewards and score, which are provided at each step, *hardware safety* $\phi_{hs}(\pi)$ is an array of counters that evaluates a policy over the specified horizon to measure the number of safety violations. We include the following violations in our safety measure: joint limits, velocity limits, and current limits.

We propose an initial set of *ROBEL* benchmark tasks to tackle a variety of challenges involving manipulation and locomotion. We summaries the tasks below and encourage readers to refer to Appendix A and supplementary material[1] for task details.

4.1 Manipulation Benchmarks on D'Claw

D'Claw is 9-DoF dexterous manipulator capable of contact rich diverse behaviors. We structure our first group of the benchmark tasks around fundamental manipulation behaviors.



(a) Pose: conform to a shape (b) Turn: rotate to a fixed target (c) Screw: rotate to a moving target

Figure 5: *D'Claw* manipulation benchmarks: Pose, Turn and Screw are motivated by commonly observed manipulation behaviors in daily life

a) **Pose** (conform to the shape of the environment): This task is motivated by the primary objective of a manipulator to conform to its surrounding in order to prepare for the upcoming maneuvers – commonly observed as various pre-grasp and latching maneuvers (Figure 5a). This set of tasks is posed as trying to match randomly selected joint angle targets. Successful completion of this task demonstrates the capability of a manipulator to have controlled access to all its joints. This set of tasks are comparatively easier to train, thereby facilitates fast iteration cycles and a gradual transition to the rest of the tasks. Two variants of this task are provided: a static variant *DClawPoseFixed* where the desired joint angles remain constant, and a dynamic variant *DClawPoseRandom* where the desired joint angle is time-dependent and oscillates between two goal positions that are sampled at the beginning of the episode.

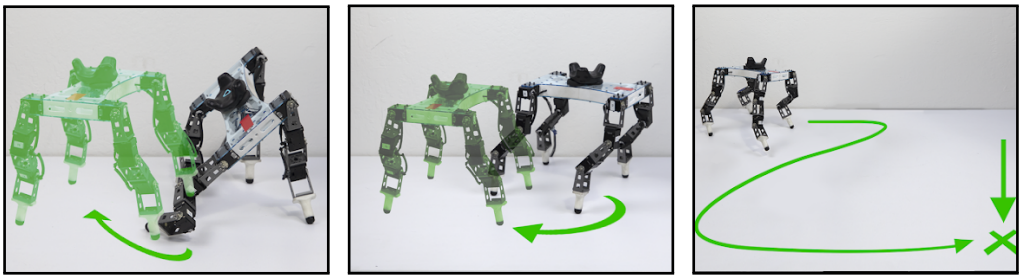
b) **Turn** (rotate to a fixed target angle): This task encapsulates the ability of a manipulator to reposition unactuated DoFs present in the environments to target configurations – commonly observed as turning various knobs, latches and handles. This set of tasks is posed as trying to match randomly selected joint angle targets for the unactuated object(s). Successful completion of this task demonstrates the ability of a manipulator to bring desired changes on external targets. In order to succeed, the manipulator requires not only co-ordination between the internal DoFs, but also an understanding of environment dynamics perceived through contact interactions. Three variants of this task are provided: *DClawTurnFixed* where initial and target angles are constant, *DClawTurnRandom* where both initial and target angles are randomly selected, *DClawTurnRandomDynamics* where initial and target angles are randomly selected as well as the environment (object size, surface, and dynamics properties) is randomized.

[1] code repository, detailed documentation, and task videos are available at www.roboticsbenchmarks.org

c) **Screw** (rotate to a moving target angle): This task focuses on the ability of a manipulator to continuously rotate an unactuated object at a constant velocity. This set of tasks is posed as trying to match joint angle targets that are themselves moving. Although very similar to turn tasks but the nuances of moving target challenge the manipulator's strategy to constantly evolve as the target drifts. Fingers often enter singular positions as the rotation progresses. A successful strategy needs to learn finger co-ordinated gating to simultaneous progress as well as stay out of local minima. Three variants of this task are provided: *DClawScrewFixed* where target velocity is constant, *DClawScrewRandom* where the initial angle and target velocity is randomly selected, *DClawScrewRandomDynamics* where the initial angle and target velocity is randomly selected as well as the environment (object size, surface, and dynamics properties) is randomized.

4.2 Locomotion Benchmark on D'Kitty

The twelve DoF locomotion platform *D'Kitty* is capable of exhibiting diverse behaviors. We structure this group of the benchmark tasks on the platform around simple locomotion behaviors exhibited by quadrupeds.



(a) Stand: getting upright (b) Orient: align heading (c) Walk: get to target

Figure 6: *D'Kitty* locomotion benchmarks

a) **Stand**: Standing upright is one of the most fundamental behavior exhibited by the animals. This task involves reaching a pose while being upright. A successful strategy requires maintaining the stability of the torso via the ground reaction forces. Three variants of this task are provided: *DKittyStandFixed* standing up from a fixed initial configuration, *DKittyStandRandom* standing up from a random initial configuration, *DKittyStandRandomDynamics* standing up from random initial configuration where the environment (surface, dynamics properties of *D'Kitty* and ground height map) is randomized. See supplementary materials[1] for full details

b) **Orient**: This task involves *D'Kitty* changing its orientation from an initial facing direction to a desired facing direction. This set of tasks is posed as matching the target configuration of the torso. A successful strategy requires maneuvering the torso via the ground reaction forces while maintaining balance. Three variants of this task are provided: *DKittyOrientFixed* maneuvers to a fixed target orientation, *DKittyOrientRandom* maneuvers to a random target orientation, *DKittyOrientRandomDynamics* maneuver to a random target orientation where the environment (surface, dynamics properties of *D'Kitty* and ground height map) is randomized. See supplementary materials[1] for full details

c) **Walk**: This task involves the *D'Kitty* moving its world position from an initial cartesian position to desired cartesian position while maintaining a desired facing direction. This task is posed as matching the cartesian position of the torso with a distant target. Successful strategy needs to exhibit locomotion gaits while maintaining heading. Three variants of this task are provided: *DKittyWalkFixed* walk to a fixed target location, *DKittyWalkRandom* walk to a randomly selected target location, *DKittyWalkRandomDynamics* walk to a selected target location where the environment (surface, dynamics properties of *D'Kitty* and ground height map) is randomized. See supplementary materials[1] for full details

ROBEL tasks variants are carefully designed to represent a wide task spectrum. The *fixed* variants (task-name suffix "Fixed") are fast to iterate and are helpful for getting started. The *random* variants (task-name suffix "Random") present a wide initial and goal distribution to study task generalization. In addition to the wider distribution, the *random dynamics* variants (task-name suffix "RandomDynamics") presents variability in the various environment properties. This variant is hardest to solve and is well suited for sim2real line of research.

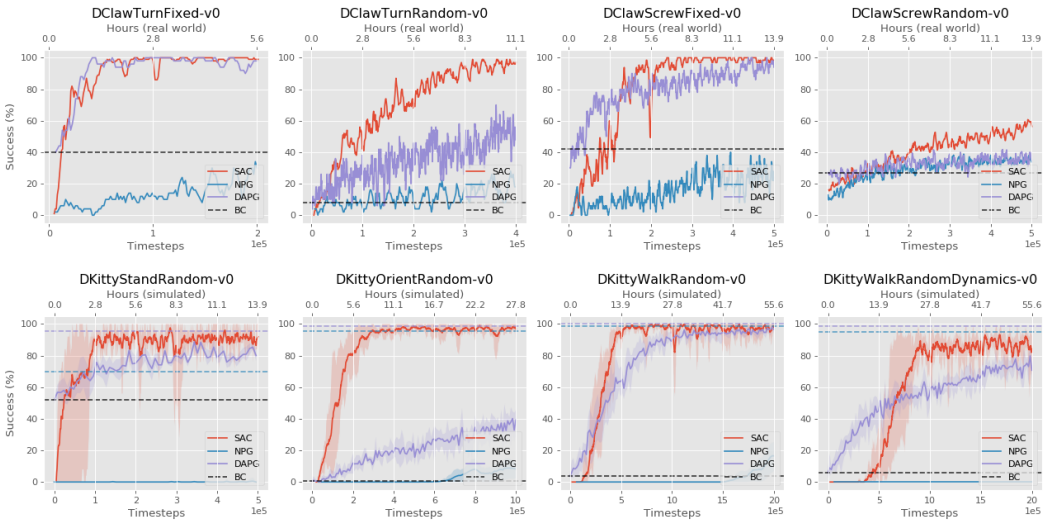


Figure 7: Success percentage for *D'Claw* and *D'Kitty* tasks trained on a physical *D'Claw* robot and a simulated *D'Kitty* robot using several agents: Soft Actor Critic (SAC)[7], Natural Policy Gradient (NPG)[39], Demo-augmented Policy Gradient (DAPG)[40], and Behavior Cloning (BC) over 20 trajectories. Success is measured via the success evaluator $\phi_{se}(\pi)$ of the task (See Appendix A for details). Each timestep corresponds to 0.1 real-world seconds

5 Experiments

We first summarize on-hardware training runs of various reinforcement learning algorithms that are included as *ROBEL* baselines. Later we evaluate *ROBEL* for its reproducibility with-in the same as well as at a geographically separated location, and reliability over extended usage. We conclude by presenting performance of our baselines over the proposed safety metrics.

5.1 Baselines

ROBEL has been tested to meet the rigor of a wide variety of learning algorithms. One candidate from each algorithmic class was added to the spectrum of baselines (Figure 7). We include Natural Policy Gradient [39] for on-policy, Soft Actor Critic [7] for off-policy, Demo Augmented Policy Gradients [40] for demonstration accelerated methods, and behavior cloning as supervised learning baseline. Using the sim-robot, dynamics randomized variant of all the tasks (referred as randomDynamics) are also included in the package to facilitate sim2real research direction. We also invite the open source community to add to our family of baselines via our open source repository.

5.2 Reproducibility

Figure 8: Left-3: Training reproducibility between two real *D'Claw* robots, developed at different laboratory locations, over the benchmark tasks. Right: Effectiveness of a policy on different hardware it wasn't trained on. Score denotes closeness to to the goal.

We test *ROBEL* reproducibility on multiple platforms independently developed at different locations (60 miles apart) via different groups (no in-person visits) using only *ROBEL* documentation[2]. We

[2]Occasional minor clarification over emails were later adopted into the documentation

evaluate *ROBEL*'s reproducibility by studying the effectiveness of a policies on different hardware. Figure 8 outlines the effectiveness of a policy on multiple hardware across two different sites.

5.3 Reliability

We provide a qualitative measure of the reliability of the system in Table 3. It should be noted that these metrics include data gathered while the system was under development. The matured system reported in this paper is much more reliable. Figure 9 provides a qualitative depiction of the robustness of the system using side by side comparison of a new and used *D'Claw* assembly. The system is fairly robust in facilitating multiple day real-world experimentation on the hardware. Occasional maintenance needs primarily include screws becoming loose. We attribute this to the vibrations caused by recurring collision impacts during manipulation and locomotion. We also observe occasional motor failures (Table 3). Owing to the modularity of *ROBEL*, this is easy to replace[3].

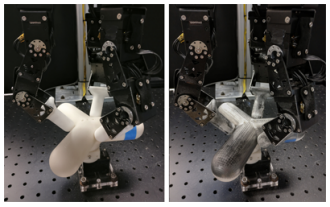


Figure 9: Change in physical appearance depicting *D'Claw* resilience to extreme usage. (left: new *D'Claw*) (right: operational for $\tilde{6}$ months)

Figure 10: Safety violations observed during the training of the *DClawScrewFixed* task

site	A	B
training hours	9000	5000
motors bought	150	40
motors broken	19	10

Table 3: (approximate) Usage statistics of *ROBEL* over 12 months. Note that statistics include data from when *ROBEL* was still under development

5.4 Safety

Smooth, elegant behavior has been a desirable but hard-to-define trait for all continuous control problems. Various forms of regularization on control, velocity, acceleration, jerks, and energy are often used to induce such properties. While there is not a universally accepted definition for smoothness, few metrics for safe behaviors can be defined in terms of hardware safety limits. In addition to dense and sparse objective, *ROBEL* also provides hardware safety objectives, which has been largely ignored in available benchmarks [3][12][1]. *ROBEL* defines safety objects over position, velocity, and torque violations calculated over a finite horizon trajectory. The *success evaluator*, provided with all benchmarks, not only reports the average task success metric, but also reports the average number of safety violations. A benchmarks challenge is considered successful when there are no safety violations. Figure 10 shows the average number of joint per episode under safety violations for two RL agents. We observe that these policies, while successful in solving the task, exhibit significant safety violations. While safety is desirable, it has largely been ignored in existing RL benchmarks resulting in limited progress. We hope that safety-metric included in *ROBEL* will sprout research in this direction.

6 Conclusion

This work proposes *ROBEL*– an open source platform of cost-effective robots designed for on-device reinforcement learning experimentation needs. *ROBEL* platforms are robust and have sustained over 14000 hours of real world training on them till date. *ROBEL* feature a 9-DOF manipulation platform *D'Claw*, and a 12-DOF locomotion platform *D'Kitty* , with a set of prepackaged benchmark tasks around them. We show the performance of these benchmarks on a variety of learning-based agents – on-policy (NPG), off-policy (SAC), demo-accelerated method (DAPG), and supervised method (BC). We provide these results as baselines for ease of comparison and extensibility. We show reproducibility of the *ROBEL*'s benchmarks by independently reproducing results at a remote site. We are excited to bring *ROBEL* to the larger robotics community and look forward to the possibilities it presents towards the evolving experimentation needs of learning-based methods, and robotics in general.

[3]Broken motors are repairable via manufacturers RMA. Motor sub-assemblies are available online as well.

Acknowledgments

We thank Aravind Rajeswaran, Emo Todorov, Vincent Vanhoucke, Matt Neiss, Chad Richards, Thinh Nguyen, Byron David, Garrett Peake, Krista Reymann, and the rest of Robotics at Google for their contributions and discussions all along the way.

References

[1] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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

[3] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

[4] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[6] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics*, 2017.

[7] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv:1812.05905*, 2018.

[8] 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*, 37(4-5):421–436, 2018.

[9] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, 2016.

[10] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *preprint arXiv:1810.06045*, 2018.

[11] F. Allgöwer and A. Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 2016.

[13] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[14] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[15] 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.

[16] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

[17] D. Kalashnikov, A. Irpan, P. P. Sampedro, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. 2018.

[18] F. Ramos, R. C. Possas, and D. Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[19] M. Bhairav, D. Manfred, G. Florian, J. P. Christopher, and P. Liam. Active domain randomization. *arXiv preprint arXiv:1904.04762*, 20189.

[20] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, et al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[21] S. Behnke. Robot competitions-ideal benchmarks for robotics research. In *Proc. of IROS-2006 Workshop on Benchmarks in Robotics Research*. Institute of Electrical and Electronics Engineers (IEEE), 2006.

[22] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[23] A. M. Dollar and R. D. Howe. The highly adaptive sdm hand: Design and performance evaluation. *The international journal of robotics research*, 29(5):585–597, 2010.

[24] Y. She, C. Li, J. Cleary, and H.-J. Su. Design and fabrication of a soft robotic hand with embedded actuators and sensors. *Journal of Mechanisms and Robotics*, 7(2):021007, 2015.

[25] Z. Xu and E. Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3485–3492. IEEE, 2016.

[26] S. H. Collins, M. Wisse, and A. Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 2001.

[27] W. Bosworth, S. Kim, and N. Hogan. The mit super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–8. IEEE, 2015.

[28] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[29] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.

[30] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2165–2170. IEEE, 2008.

[31] M. J. Lum, D. C. Friedman, G. Sankaranarayanan, H. King, K. Fodero, R. Leuschke, B. Hannaford, J. Rosen, and M. N. Sinanan. The raven: Design and validation of a telesurgery system. *The International Journal of Robotics Research*, 28(9):1183–1197, 2009.

[32] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. *arXiv preprint arXiv:1809.07731*, 2018.

[33] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019.

[34] Dynamixel smart actuator. `http://www.robotis.us/dynamixel/`. Accessed: 2019-07-02.

[35] Usb-serial bus. `http://www.robotis.us/u2d2/`. Accessed: 2019-07-02.

[36] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[37] M. L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

[38] 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.

[39] S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[40] 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. *arXiv preprint arXiv:1709.10087*, 2017.

Appendix

A *ROBEL* task details

In this section, we outline details of the benchmark task presented in section 4.

A.1 *D'Claw* tasks

The action space of all *D'Claw* tasks (subsection 4.1) is a 1D vector of 9 *D'Claw* joint positions.

(i) **Pose**: This task involves posing *D'Claw* by driving its joints $\boldsymbol{\theta}_t$ to a desired joint angles $\boldsymbol{\theta}_{goal}$ sampled randomly from the feasible joint angle space at the beginning of the episode. The observation space s_t is a 36-size 1D vector that consists of the current joint angles $\boldsymbol{\theta}_t$, the joint velocities $\dot{\boldsymbol{\theta}}_t$, the error between the goal and current joint angles, and the last action. The reward function is defined as:

$$r_t = -\|\boldsymbol{\theta}_{goal} - \boldsymbol{\theta}_t\| - 0.1\left\|\dot{\boldsymbol{\theta}}_t * \mathbb{1}(|\dot{\boldsymbol{\theta}}| > 0.5)\right\|$$

Three variants of this task are provided:

(a) *DClawPoseFixed*: a static variant where the desired joint angles remain constant for the episode

(b) *DClawPoseRandom*: a dynamic variant where the desired joint angle is time-dependent and oscillates between two goal positions that are sampled at the beginning of the episode.

(c) *DClawPoseRandomDynamic*: same as previous. The joint damping, and the joint friction loss are randomized at the beginning of every episode.

Success evaluator metric $\phi_{se}(\pi)$ of policy π is defined using the mean absolute tracking error being within the threshold $\beta = 10^\circ$

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}\Big[\frac{1}{T}\sum_{t=0}^{T} mean\Big\|\theta^{(\tau)}_{goal} - \theta^{(\tau)}_{t}\Big\| < \beta\Big]$$

(ii) **Turn**: This task involves rotating an object from an initial angle $\theta_{0,obj}$ to a goal angle $\theta_{goal,obj}$. The observation space is a 21-size 1D vector of the current joint angles $\boldsymbol{\theta}_t$, the joint velocities $\dot{\boldsymbol{\theta}}_t$, the sine and cosine values of the object's angle $\theta_{t,obj}$, the last action, and the error between the goal and the current object angle $\Delta\theta_{t,obj} = \theta_{t,obj} - \theta_{goal,obj}$. The reward function is defined as

$$r_t = -5|\Delta\theta_{t,obj}| - \|\boldsymbol{\theta}_{nominal} - \boldsymbol{\theta}_t\| - \left\|\dot{\boldsymbol{\theta}}_t\right\| + 10\mathbb{1}(|\Delta\theta_{t,obj}| < 0.25) + 50\mathbb{1}(|\Delta\theta_{t,obj}| < 0.1)$$

Three variants of this task are provided:

(a) *DClawTurnFixed*: constant initial angle (0°) and constant goal angle (180°).

(b) *DClawTurnRandom*: random initial and goal angle.

(c) *DClawTurnRandomDynamics*: same as previous. The position of the *D'Claw* relative to the object, the object's size, the joint damping, and the joint friction loss are randomized at the beginning of every episode.

Success evaluator metric $\phi_{se}(\pi)$ of policy π is defined using the error in last step of the episode $(t = T)$ being within the goal threshold $\beta = 0.1$ as:

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}[\Delta\theta^{(\tau)}_{T,obj} < \beta]$$

(iii) **Screw**: This task involves rotating an object at a desired velocity $\dot{\theta}_{desired}$ from an initial angle. This is represented by a $\theta_{t,goal}$ that is updated every step as $\theta_{t,goal} = \theta_{t-1,goal} + \dot{\theta}_{desired} * dt$. Screw tasks have the same observation space and reward definitions as the Turn tasks. Three variants of this task are provided:

(a) *DClawScrewFixed*: constant initial angle (0°) and velocity ($0.5\frac{\text{rad}}{\text{sec}}$)

(b) *DClawScrewRandom*: random initial angle ($[-180^\circ, 180^\circ]$) and desired velocity ($[-0.75\frac{\text{rad}}{\text{sec}}, 0.75\frac{\text{rad}}{\text{sec}}]$)

(c) *DClawScrewRandomDynamics*: same as previous. The position of the *D'Claw* relative to the object, the object's size, the joint damping, and the joint friction loss are randomized at the beginning of every episode.

Success evaluator metric $\phi_{se}(\pi)$ of policy π is defined using the mean absolute tracking error being within the threshold $\beta = 0.1$

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}\Big[\frac{1}{T}\sum_{t=0}^{T}|\Delta\theta^{(\tau)}_{t,obj}| < \beta\Big]$$

A.2 *D'Kitty* tasks

The action space of all of the *D'Kitty* tasks is a 1D vector of 12 joint positions. The observation space shares 49 common entries: the Cartesian position (3), Euler orientation (3), velocity (3), and angular velocity (3) of the *D'Kitty* torso, the joint positions $\boldsymbol{\theta}$ (12) and velocities $\dot{\boldsymbol{\theta}}$ (12) of the 12 joints, the previous action (12), and 'uprightness' $u_{t,kitty}$ (1). The uprightness $u_{t,kitty}$ of the *D'Kitty* is measured as it's orientation projected over the global vertical axis:

$$u_{t,kitty} = \boldsymbol{R}_{\hat{z},t,kitty} \cdot \hat{\boldsymbol{Z}}$$

The *D'Kitty* tasks share a common term in the reward function $r_{t,upright}$ regarding uprightness defined as:

$$r_{t,upright} = \alpha_{upright}\frac{u_{t,kitty} - \beta}{1-\beta} + \alpha_{falling}(u_{t,kitty} < \beta)$$

where β is the cosine similarity threshold with the global z-axis beyond which we consider the *D'Kitty* to have fallen. When perfectly upright $\alpha_{t,upright}$ reward is collected, when alignment ($u_{t,kitty}$) falls below the threshold β, the episode terminates early and $\alpha_{falling}$ is collected.

(i) **Stand**: This task involves *D'Kitty* coordinating its 12 joints $\boldsymbol{\theta}_t$ to stand upright maintaining a pose specified by $\boldsymbol{\theta}_{goal}$. The observation space is a 61-size 1D vector of the shared observation space entries and pose error $\boldsymbol{e}_{t,pose} = (\boldsymbol{\theta}_{goal} - \boldsymbol{\theta}_t)$. The reward function is defined as:

$$r_t = r_{t,upright} - 4\bar{e}_{t,pose} - 2||\boldsymbol{p}_{t,kitty}||_2 + 5u_{t,kitty}\mathbb{1}(\bar{e}_{t,pose} < \frac{\pi}{6}) + 10u_{t,kitty}\mathbb{1}(\bar{e}_{t,pose} < \frac{\pi}{12})$$

where $\bar{e}_{t,pose}$ is mean absolute pose error, $\boldsymbol{p}_{t,kitty}$ is the cartesian position of *D'Kitty* on the horizontal plane and the shared reward function constants are $\alpha_{upright} = 2$, $\alpha_{falling} = -100$, $\beta = \cos(90^\circ)$.

Three variants of this task are provided:

(a) *DKittyStandFixed*: constant initial pose.

(b) *DKittyStandRandom*: random initial pose.

(c) *DKittyStandRandomDynamics*: same as previous. The joint gains, damping, friction loss, geometry friction coefficients, and masses are randomized. In addition, a randomized height field is generated with heights up to $0.05\mathrm{m}$

The successor evaluator indicates success if the mean pose error is within the goal threshold $\beta = \frac{\pi}{12}$ and the *D'Kitty* is sufficiently upright at the last step ($t = T$) of the episode:

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}[\mathbb{1}(\bar{e}^{(\tau)}_{T,pose} < \beta) * \mathbb{1}(u^{(\tau)}_{T,kitty} > 0.9)]$$

(ii) **Orient**: This task involves *D'Kitty* matching its current facing direction $\boldsymbol{\omega}_t$ with a goal facing direction $\boldsymbol{\omega}_{goal}$, thus minimizing the facing angle error $e_{t,facing}$ between $\boldsymbol{\omega}_{desired}$ and $\boldsymbol{\omega}_t$. The observation space is a 53-size 1D vector of the shared observation space entries, $\boldsymbol{\omega}_t$ and $\boldsymbol{\omega}_{goal}$ represented as unit vectors on the (X,Y) plane, and angle error $e_{t,facing}$. The reward function is defined as:

$$\begin{aligned} r_t &= r_{t,upright} - 4e_{t,facing} - 4||\boldsymbol{p}_{t,kitty}||_2 + r_{bonus\_small} + r_{bonus\_big} \\ r_{bonus\_small} &= 5(e_{t,facing} < 15^\circ \text{ or } u_{t,kitty} > \cos(15^\circ)) \\ r_{bonus\_big} &= 10(e_{t,facing} < 5^\circ \text{ and } u_{t,kitty} > \cos(15^\circ)) \end{aligned}$$

where the shared reward function constants are $\alpha_{upright} = 2$, $\alpha_{falling} = -500$, $\beta = \cos(25^\circ)$.

Three variants of this task are provided:

(a) *DKittyOrientFixed*: constant initial facing ($0°$) and goal facing ($180°$).
(b) *DKittyOrientRandom*: random initial facing ($[-60°, 60°]$) and goal facing ($[120°, 240°]$)
(c) *DKittyOrientRandomDynamics*: same as previous. The joint gains, damping, friction loss, geometry friction coefficients, and masses are randomized. In addition, a randomized height field is generated with heights up to 0.05m

The successor evaluator indicates success if the facing angle error is within the goal threshold and the *D'Kitty* is sufficiently upright at the last step ($t = T$) of the episode:

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}[\mathbb{1}(e^{(\tau)}_{T,facing} < 5°) * \mathbb{1}(u^{(\tau)}_{T,kitty} > \cos(15°)]$$

(iii) **Walk**: This task has the *D'Kitty* move its current Cartesian position $\boldsymbol{p}_{t,kitty}$ to a desired Cartesian position $\boldsymbol{p}_{goal}$, minimizing the distance $d_{t,goal} = ||\boldsymbol{p}_{goal} - \boldsymbol{p}_{t,kitty}||_2$. Additionally, the *D'Kitty* is incentivized to face towards the goal. The heading alignment is calculated as $h_{t,goal} = \boldsymbol{R}_{\hat{y},t,kitty} \cdot \frac{\boldsymbol{p}_{goal} - \boldsymbol{p}_{t,kitty}}{d_{t,goal}}$. The observation space is a 52-size 1D vector of the shared observation space entries, $h_{t,goal}$ and $\boldsymbol{p}_{goal} - \boldsymbol{p}_{t,kitty}$.
The reward function is defined as:

$$\begin{aligned} r_t &= r_{t,upright} - 4d_{t,goal} + 2h_{t,goal} + r_{bonus\_small} + r_{bonus\_big} \\ r_{bonus\_small} &= 5(d_{t,goal} < 0.5 \text{ or } h_{t,goal} > \cos(25°)) \\ r_{bonus\_big} &= 10(d_{t,goal} < 0.5 \text{ and } h_{t,goal} > \cos(25°)) \end{aligned}$$

and the shared reward function constants are $\alpha_{upright} = 1$, $\alpha_{falling} = -500$, $\beta = \cos(25°)$.
Three variants of this task are provided:

(a) *DKittyWalkFixed*: constant distance (2m) towards $0°$.
(b) *DKittyWalkRandom*: random distance ($[1, 2]$) towards random angle ($[-60°, 60°]$)
(c) *DKittyWalkRandomDynamics*: same as previous. The joint gains, damping, friction loss, geometry friction coefficients, and masses are randomized. In addition, a randomized height field is generated with heights up to 0.05m

The successor evaluator indicates success if the goal distance is within a threshold and the *D'Kitty* is sufficient upright at the last step of the episode:

$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}[\mathbb{1}(d^{(\tau)}_{T,goal} < 0.5) * \mathbb{1}(u^{(\tau)}_{T,kitty} > \cos(25°))]$$

A.3 Safety metrics

The following safety scores are shared between all tasks.

(i) **Position violations**: This score indicates that the joint positions are near their operating bounds. For the N joints of the robot, this is defined as:

$$s_{position} = \sum_{i=1}^{N} \Big(\mathbb{1}(|\theta_i - \beta_{i,lower}| < \epsilon) + \mathbb{1}(|\theta_i - \beta_{i,upper}| < \epsilon)\Big)$$

where $\beta_{i,lower}$ and $\beta_{i,upper}$ is the respective lower and upper joint position bound for the ith joint, and ϵ is the threshold within which the joint position is considered to be near the bound.

(ii) **Velocity violations**: This score indicates that the joint velocities are exceeding a safety limit For the N joints of the robot, this is defined as:

$$s_{velocity} = \sum_{i=1}^{N} \mathbb{1}(|\dot{\theta}_i| > \alpha_i)$$

where α_i is the speed limit for the ith joint.

(iii) **Current violations**: This score indicates that the joints are exerting forces that exceed a safety limit. For the N joints of the robot, this is defined as:

$$s_{current} = \sum_{i=1}^{N} \mathbb{1}(|k_i| > \gamma_i)$$

where γ_i is the current limit for the ith joint.

B Locomotion benchmark performance on *D'Kitty*

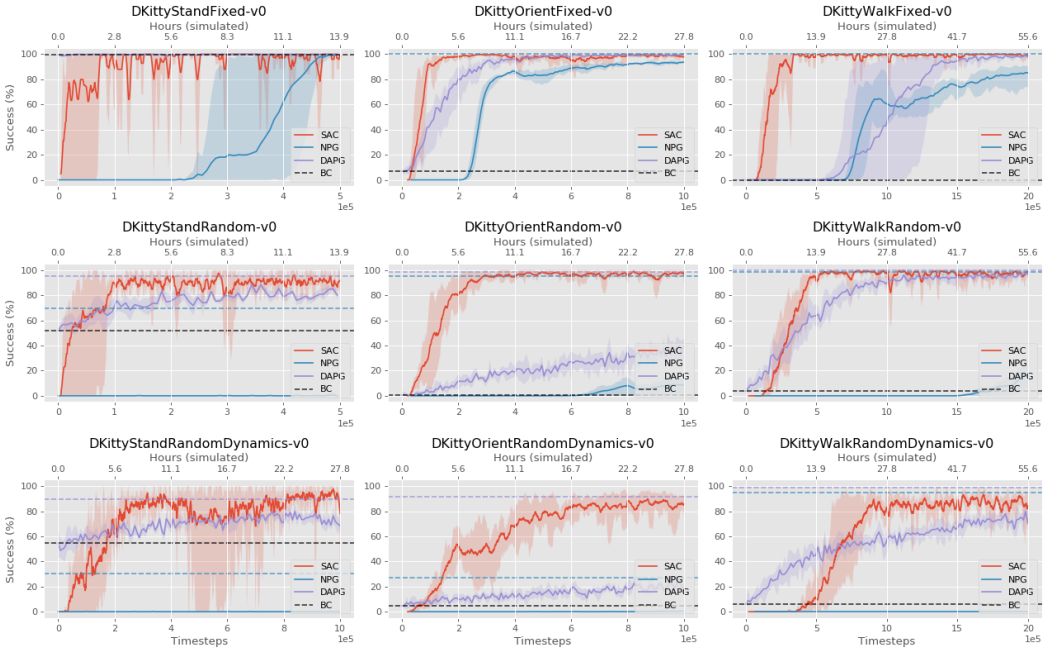


Figure 11: Success percentage (3 seeds) for all *D'Kitty* tasks trained on a simulated *D'Kitty* robot using Soft Actor Critic (SAC), Natural Policy Gradient (NPG), Demo-Augmented Policy Gradient (DAPG), and Behavior Cloning (BC) over 20 trajectories. Each timestep corresponds to 0.1 simulated seconds.

C *ROBEL* reproducibility

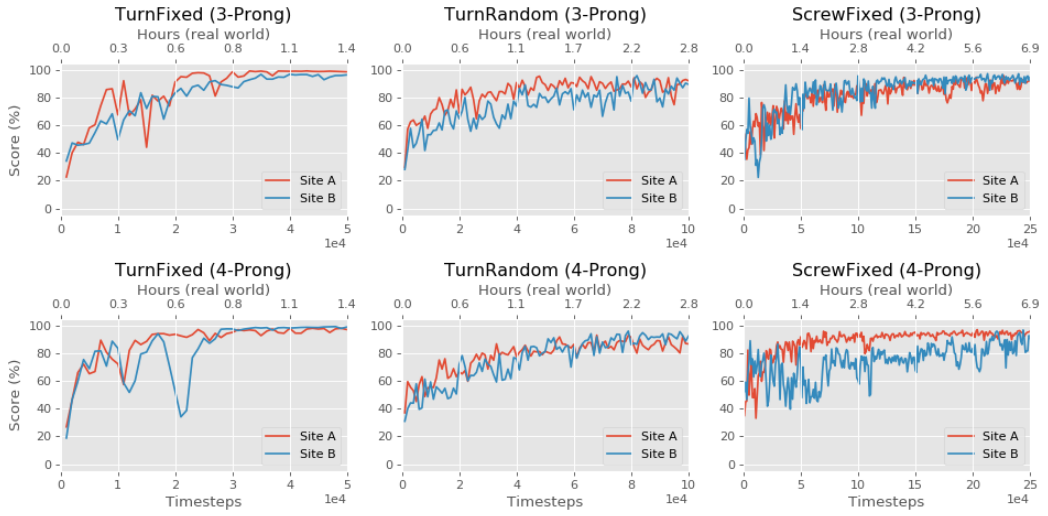


Figure 12: SAC training performance of *D'Claw* tasks on two real *D'Claw* robots each at different laboratory locations. Score denotes the closeness to the goal. Each timestep corresponds to 0.1 simulated seconds. Each task is trained over two different task objects: a 3-prong valve and a 4-prong valve.