Proceedings of Machine Learning Research 176:190–204, 2022 NeurIPS 2021 Competition and Demonstration Track

Real Robot Challenge: A Robotics Competition in the Cloud *

Stefan Bauer[*1] baue@kth.se
Manuel Wüthrich[*2] manuel.wuthrich@pm.me
Felix Widmaier[*2] felix.widmaier@tuebingen.mpg.de
Annika Buchholz[2] annika.buchholz@tuebingen.mpg.de
Sebastian Stark[2] stark@tuebingen.mpg.de
Anirudh Goyal[3] anirudhgoyal9119@gmail.com
Thomas Steinbrenner[2] thomas.steinbrenner@tuebingen.mpg.de
Joel Akpo[2] joel.akpo@is.mpg.de
Shruti Joshi[3] shruti.joshi@tuebingen.mpg.de
Vincent Berenz[2] vberenz@tuebingen.mpg.de
Vaibhav Agrawal[2] vaibhav.agrawal@tuebingen.mpg.de
Bernhard Schölkopf[2] bs@tuebingen.mpg.de

[*] *Equal contribution*
[1] *KTH Stockholm*
[2] *MPI for Intelligent Systems*
[3] *MILA*

Niklas Funk[†3], **Julen Urain De Jesus**[†3], **Jan Peters**[†3], **Joe Watson**[†3], **Claire Chen**[†4], **Krishnan Srinivasan**[†4], **Junwu Zhang**[†4], **Jeffrey Zhang**[†4], **Matthew R. Walter**[†5], **Rishabh Madan**[†9], **Charles Schaff**[†5], **Takuma Yoneda**[†5], **Denis Yarats**[†6], **Arthur Allshire**[†7], **Ethan K. Gordon** [†8], **Tapomayukh Bhattacharjee**[†9], **Siddhartha S. Srinivasa**[†8], **Animesh Garg**[†7], **Takahiro Maeda**[†10], **Harshit Sikchi**[†11], **Jilong Wang**[†12], **Qingfeng Yao**[†12], **Shuyu Yang**[†12], **Robert McCarthy**[†13], **Francisco Roldan Sanchez**[†14], **Qiang Wang**[†13], **David Cordova Bulens**[†13], **Kevin McGuinness**[†14], **Noel O'Connor**[†14], **Stephen J. Redmond**[†13]

[†] *Challenge participant,* [3]*TU Darmstadt,* [4]*Stanford University,* [5]*TTI Chicago,* [6]*New York University,* [7]*University of Toronto,* [8]*University of Washington,* [9]*Cornell University,* [10] *TTI,* [11] *The University of Texas,* [12] *Westlake University,* [13] *University College Dublin,* [14] *Dublin City University*

Editor: Douwe Kiela, Marco Ciccone, Barbara Caputo

[*] https://real-robot-challenge.com

Abstract

Dexterous manipulation remains an open problem in robotics. To coordinate efforts of the research community towards tackling this problem, we propose a shared benchmark. We designed and built robotic platforms that are hosted at the MPI-IS[1] and can be accessed remotely. Each platform consists of three robotic fingers that are capable of dexterous object manipulation. Users are able to control the platforms remotely by submitting code that is executed automatically, akin to a computational cluster. Using this setup, i) we host robotics competitions, where teams from anywhere in the world access our platforms to tackle challenging tasks ii) we publish the datasets collected during these competitions (consisting of hundreds of robot hours), and iii) we give researchers access to these platforms for their own projects.

Keywords: Reinforcement Learning, Robotics, Representation Learning, Optimal Control, Dexterous Manipulation

1. Introduction

Dexterous manipulation is humans' interface to the physical world. Our ability to manipulate objects around us in a creative and precise manner is one of the most apparent distinctions between human and animal intelligence. The impact robots with a similar level of dexterity would have on our society cannot be overstated. They would likely replace humans in most tasks that are primarily physical, such as working at production lines, packaging, constructing houses, agriculture, cooking, and cleaning. Yet, robotic manipulation is still far from the level of dexterity attained by humans, as witnessed by the fact that these are still mostly carried out by humans. This problem has been remarkably resistant to the rapid progress of machine learning over the past years. A factor that has been crucial for progress in machine learning, but nonexistent in real-world robotic manipulation, is a shared benchmark. Benchmarks allow for different labs to coordinate efforts, reproduce results and measure progress. Most notably, in the area of image processing, such benchmarks were crucial for the rapid progress of deep learning. More recently, simulation benchmarks have been proposed in reinforcement learning (RL) Brockman et al. (2016); Tassa et al. (2018). However, methods that are successful in simulators transfer only to a limited degree to real robots. Therefore, the robotics community has recently proposed a number of open-source platforms for robotic manipulation Yang et al. (2019); Ahn et al. (2019); Wüthrich et al. (2020). These robots can be built by any lab to reproduce results of other labs. While this is a large step towards a shared benchmark, it requires effort by the researchers to set up and maintain the system, and it is nontrivial to ensure a fully standardized setup.

Therefore, we provide **remote access to dexterous manipulation platforms** hosted at MPI-IS, see Figure 2 in the appendix and Figure 1 (*interested researchers can contact us to request access, see our website*[1]). This allows for an objective evaluation of robot-learning algorithms on real-world platforms with minimal effort for the researchers. In addition, we publish a **large dataset of these platforms interacting with objects**, which we collected during a competition we hosted in 2020 and 2021 as part of the Neural Information Processing Systems (NeurIPS) Conference[2]. During this competition, teams from across the

1. `https://people.tuebingen.mpg.de/felixwidmaier/trifinger`
2. `https://real-robot-challenge.com/2020`

world developed algorithms for challenging object manipulations tasks, which yielded a very diverse dataset containing meaningful interactions.

To facilitate research into sim-to-real transfer, we also provide a **simulation of the robotic setup** (see website[3]). All the code, for both simulation and control of the real robots, is open-source.

In the rest of the paper, we describe the robotic hardware and the software interface which allows easy robot access, similarly to a computational cluster. We also describe the robot competitions we hosted and the data we collected in the process.

2. Related Work

In the past years, a large part of the RL community has focused on simulation benchmarks, such as the deepmind control suite Tassa et al. (2018) or OpenAI gym Brockman et al. (2016) and extensions thereof Zamora et al. (2016). These benchmarks internally use physics simulators, typically Mujoco Todorov et al. (2012) or PyBullet Coumans and Bai (2016).

These commonly accepted benchmarks allowed researchers from different labs to compare their methods, reproduce results, and hence build on each other's work. Very impressive results have been obtained through this coordinated effort Haarnoja et al. (2018); Fujimoto et al. (2018); Popov et al. (2017); Mnih et al. (2016); Heess et al. (2017); Duan et al. (2016); Henderson et al. (2018).

In contrast, no such coordinated effort has been possible on real robotic systems, since there is no shared benchmark. This lack of standardized real-world benchmarks has been recognized by the robotics and RL community Behnke (2006); Bonsignorio and del Pobil (2015); Calli et al. (2015a,b); Amigoni et al. (2015); Murali et al. (2019). Recently, there have been renewed efforts to alleviate this problem:

Affordable Open-Source Platforms: The robotics community recently proposed affordable open-source robotic platforms that can be built by users. For instance, Yang et al. (2019) propose Replab, a simple, low-cost manipulation platform that is suitable for benchmarking RL algorithms. Similarly, Ahn et al. (2019) propose a simple robotic hand and quadruped that can be built from commercially available modules. CMU designed LoCoBot, a low-cost open-source platform for mobile manipulation. Grimminger et al. (2020) propose an open-source quadruped consisting of off-the-shelf parts and 3D-printed shells. Based on this design, Wüthrich et al. (2020) developed an open-source manipulation platform consisting of three fingers capable of complex dexterous manipulation (here, we use an industrial-grade adaptation of this design).

Such platforms are beneficial for collaboration and reproducibility across labs. However, setting up and maintaining such platforms often requires hardware experts and is time-intensive. Furthermore, there are necessarily small variations across labs that may harm reproducibility. To overcome these limitations, the robotics community has proposed a number of remote robotics benchmarks.

Remote Benchmarks: For mobile robotics, Pickem et al. (2017) propose the Robotarium, a remotely accessible swarm robotics research platform. Similarly, Duckietown Paull et al. (2017) hosts the AI Driving Olympics AI-DO: AI Driving Olympics – Duckietown twice

3. same as footnote 1

per year. However, a remote benchmark for robotic manipulation accessible to researchers around the world is still missing. Therefore we propose such a system herein.

3. Robotic Platforms

We host 8 robotic platforms at MPI-IS (see Figures 1 and 2), remote users can submit code which is then assigned to a platform and executed automatically, akin to a computational cluster. Users have access to the data collected during execution of their code. Submission and data retrieval can be automated to allow for RL methods that alternate between policy evaluation and policy improvement.

The platforms we use here are based on an open-source design that was published recently Wüthrich et al. (2020) (see also website[4]). The benefits of this design are

- **Dexterity:** The robot design consists of three fingers and has the mechanical and sensorial capabilities necessary for complex object manipulation beyond grasping.
- **Safe Unsupervised Operation:** The combination of robust hardware and safety checks in the software allows users to run even unpredictable algorithms without supervision. This enables, for instance, training of deep neural networks directly on the real robot.
- **Ease of Use:** The C++ and Python interfaces are simple and well-suited for RL as well as optimal control at rates up to 1 kHz. For convenience, we also provide a simulation (PyBullet) environment of the robot.

Here, we use an industrial-grade adaptation of this hardware (see Figures 1 and 2) to guarantee an even longer lifetime and higher reproducibility. In addition, we developed a submission system to allow researchers from anywhere in the world to submit code with ease.

3.1. Observations and Actions

Actions: This platform consists of 3 fingers, each with 3 joints, yielding a total of 9 degrees of freedom (and 9 corresponding motors). There are two ways of controlling the robot: One can send 9-dimensional **torque-actions** which are directly executed by the motors. Alternatively, we provide the option of using **position-actions** (9-dimensional as well), which are then translated to torques by an internal controller. The native control rate of the system is 1kHz, but one can control at a lower rate, if so desired.

Observations: An observation consists of proprioceptive measurements, images and the object pose inferred using an object tracker. The proprioceptive measurements are provided at a rate of 1 kHz and contain the **joint angles, joint velocities, joint torques** (each 9 dimensional) and **finger-tip forces** (3 dimensional). There are three cameras placed around the robot. The **camera images** are provided at 10 Hz and have a resolution of 270x270 pixels. In addition, our system also contains an object tracker, which provides the **pose of the object** being manipulated along with the images.

4. https://sites.google.com/view/trifinger

3.2. Submitting Code

We use HTCondor[5] to provide a cluster-like system where users can submit jobs which are then automatically executed on a randomly-selected robot. During execution, a backend process automatically starts the robot, monitors execution and records all actions and observations. The users use the simple front-end interface to send actions to and retrieve observations from the robot (see Wüthrich et al. (2020) for more details on the design of the interface, see the website[6] for links to the code repositories).

Below is a minimal example of actual user code in Python. It creates a front-end interface to the robot and uses it to send torque commands that are computed by a user-specified control policy based on the observations:

```python
# Initialise front end to interact with the robot.
robot = robot_fingers.TriFingerPlatformFrontend()

# Create a zero-torque action to start with
action = robot_interfaces.trifinger.Action()

while True:
    # Append action to the "action queue". Returns the time step at which the given action will be executed.
    t = robot.append_desired_action(action)

    # Get observations of time step t. Will wait if t is in the future.
    robot_obs = robot.get_robot_observation(t)
    camera_obs = robot.get_camera_observation(t)

    # Compute next action using some control policy. The different observation values are listed separately for illustration.
    torque = control_policy(
        robot_obs.position,
        robot_obs.velocity,
        robot_obs.torque,
        robot_obs.tip_force,
        camera_obs.cameras[0].image,
        camera_obs.cameras[1].image,
        camera_obs.cameras[2].image,
        camera_obs.object_pose,
    )
    action = robot_interfaces.trifinger.Action(torque=torque)
```

At the end of each job, the recorded data is stored and provided to the user, who can then analyse it and use it, for example, to train a better policy. Users can automate submissions and data retrieval to run RL algorithms directly on the robots. See our website[7] to learn how to get started and submit code to our robots.

5. https://research.cs.wisc.edu/htcondor
6. https://people.tuebingen.mpg.de/felixwidmaier/trifinger
7. https://people.tuebingen.mpg.de/felixwidmaier/trifinger

3.3. Simulation

To facilitate testing code and sim-to-real transfer, we provide a simulation of the platform with same software interface as the real robot[8]. In addition, we developed a more advanced version of the simulator where parameters of the robot and the environment (such as masses, colors of objects, size of objects etc.) can be changed easily (see paper Ahmed et al. (2021) and website[9]). This allows for learning the causal structure of the control problem and facilitates transfer learning, in particular transferring a policy to the real world.

4. The Real Robot Challenge

Using these TriFinger robots, we organized two competitions, called "Real Robot Challenge" (RRC), in 2020 and in 2021 as part of NeurIPS. In this section we describe the general structure of the challenges and how user submissions were evaluated. The specific tasks and results of each challenge will be presented in Sections 5 and 6.

4.1. Challenge Phases

Each challenge was split into three phases. The first phase consisted of a task in our simulator and served as a qualification round before granting access to the real robots. Teams who achieved promising results in the first phase could then move on to phases 2 and 3 on the real robots. They were given remote access through the submission system described in Section 3.2. Phase 2 consisted of solving the task from phase 1 on the real robots, so participants could build on what they already achieved in simulation. In the last phase the object (and in RRC 2021 also the task itself) was changed.

4.2. Evaluation

Each task had to be solved within a fixed time frame (usually an episode of two minutes, corresponding to 120000 steps at 1 kHz). For the evaluation, a task-specific reward r_t was computed for every time step t. We defined the performance of an episode as the cumulative reward $R = \sum_t r_t$. At the end of each real-world phase the code of each team was executed for multiple goals on different robots (using the same set of goals for all users). The average cumulative reward of these runs was then used to rank the submissions.

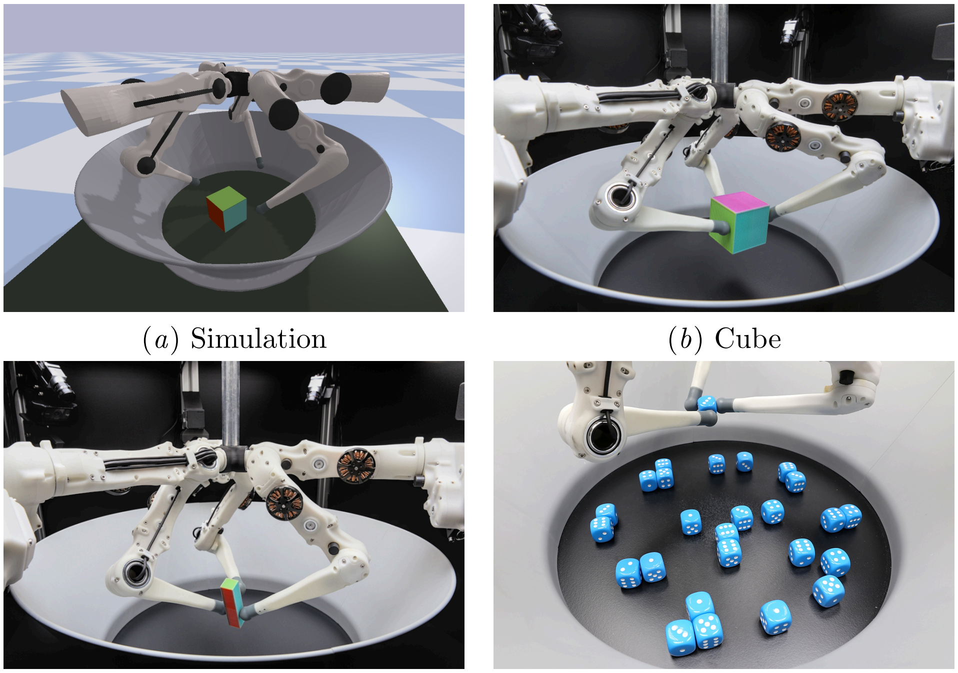
5. Tasks and Results of RRC 2020

The first challenge lasted from August to December 2020. During the challenge, the data of all runs made by the participants was recorded, resulting in a large dataset that is rich in contact interactions. This dataset is publicly available and also describe in the following.

5.1. Tasks

In all three phases, the task was to move an object from its initial position at the center of the workspace to a randomly sampled goal. In phases 1 (simulation) and 2 this object was a

8. https://open-dynamic-robot-initiative.github.io/trifinger_simulation
9. https://sites.google.com/view/causal-world/home



(a) Simulation
(b) Cube
(c) Elongate Cuboid (only 2020)
(d) Dice (only 2021)

Figure 1: Illustration of the different challenge tasks. In RRC 2020 the cube and the cuboid were used and in RRC 2021 the cube and the dice.

65 mm cube, in phase 3 it was replaced by a smaller, elongate cuboid (20×20×80 mm) that is more difficult to grasp and manipulate. Figure 1 shows pictures of the different objects. In each phase, there were four levels of difficulty corresponding to different goal distributions:

- **Level 1:** The goal is randomly sampled on the table, so it can be attained by pushing the object. The orientation is not considered for the reward computation.
- **Level 2:** The object has to be lifted to a fixed goal position 8 cm above the table center. The orientation is not considered for the reward computation.
- **Level 3:** The goal is randomly sampled somewhere within the arena with an height of up to 10 cm. The orientation is not considered for the reward computation.
- **Level 4:** As level 3, but in addition to the position, a goal orientation is sampled uniformly.

5.1.1. Reward Function for Levels 1-3

For difficulty levels 1-3, we only considered position error (i.e. orientation is ignored). We used a weighted sum of the Euclidean distance on the x/y-plane and the absolute distance along the z-axis. Both components are scaled based on their expected range. The sum is again rescaled so that the total error is in the interval $[0, 1]$. Given goal position $p_g = (x_g, y_g, z_g)$, actual position $p_a = (x_a, y_a, z_a)$, arena diameter d and maximum expected height h, the position error e_{pos} is computed as

$$e_{\text{pos}} = \frac{1}{2}\left(\frac{\sqrt{(x_g - x_a)^2 + (y_g - y_a)^2}}{d} + \frac{|z_g - z_a|}{h}\right) \tag{1}$$

We set $d = 0.39\,\text{m}$ (matching the inner diameter of the arena boundary), and $h = 0.1\,\text{m}$. The reward r is then simply the negative error $r = -e_{\text{pos}}$.

5.1.2. Reward Function for Level 4

Phases 1 and 2 (Cube) For level 4, we considered both position and orientation. The position error e_{pos} is computed as in the previous level, according to (1). We define the rotation error as the normalized angle of the rotation (represented as quaternion q) that would have to be applied to the orientation of the object to match the goal orientation:

$$e_{\text{rot}} = \frac{2 \cdot \text{atan2}(\|(q_x, q_y, q_z)\|, |q_w|)}{\pi}. \tag{2}$$

As reward we use the negative average error $r = -\frac{e_{\text{pos}} + e_{\text{rot}}}{2}$, which lies in the interval $[0, 1]$.

Phase 3 (Cuboid) We found that for the narrow cuboid used in phase 3, our object tracking method was unreliable with respect to the rotation around the long axis of the cuboid. To prevent this from affecting the reward computation, we changed the computation of the rotation error to use only the normalized absolute angle between the long axes of the cuboid in goal and actual pose.

5.1.3. Evaluating Submissions

As described in Section 4.2, we used the average score over multiple runs with different goals for evaluating the submissions. Since there were multiple difficulty levels for the goals, an equal number of goals of each level i was used and the average cumulative Reward R_i was computed separately for each level. The total score for the ranking was then computed as a weighted sum $\sum_{i=1}^{4} i \cdot R_i$. This gives a higher weight to higher (more difficult) levels, encouraging teams to solve them.

5.2. Challenge Results

After the simulation stage, seven teams with excellent performance qualified for the real-robot phases. These teams made thousands of submissions to the robots, corresponding to approximately 250 hours of robot-run-time. See Appendix B for the final evaluation results and submission statistics of phases 2 and 3. The top teams in both phases found solutions that successfully grasp the object and move it to the goal position. Videos published by the winning teams are available on YouTube[10]. They also published reports describing their methods (Yoneda et al. (2021); Anonymous (2020); Chen et al. (2021)) and open-sourced

10. ardentstork: https://youtube.com/playlist?list=PLBUWL2_ywUvE_czrinTTRqqzNu86mYuOV
troubledhare: https://youtube.com/playlist?list=PLEYg4qhK8iUaXVb1ij18pVwnzeowMCpRc
sombertortoise: https://youtu.be/I65Kwu9PGmg

their code[11]. We collected all the data produced during the challenge and aggregated it into a dataset, which is described in Section 5.3.

The Winning Policies The winning teams used similar methods for solving the task: They made use of motion primitives that are sequenced using state machines. For difficulty level 4, where orientation is important, they typically first perform a sequence of motions to rotate the object to be roughly in the right orientation before lifting it to the goal position. For details regarding their implementations, please refer to Yoneda et al. (2021); Anonymous (2020); Chen et al. (2021); Allshire et al. (2021). Further Funk et al. (2021) contains a more detailed description of the solutions of some of the teams.

5.3. The Dataset

The dataset contains the recorded data of all jobs that were executed during phases 2 and 3 of the challenge. Combined with the runs from the weekly evaluation rounds, this results in 2856 episodes of phase 2 and 7422 episodes of phase 3. The data of each episode can be downloaded individually. It contains all robot and camera observations (including object pose information) as well as all actions that were sent by the user (see Section 3.1), the goal pose that was used in this episode, camera calibration parameters and metadata such as timestamp, challenge phase, etc. Further some metrics like cumulative reward, initial distance to goal, maximum height of the object throughout the episode and some more. We expect that users of the dataset will typically not use all episodes (which would be a large amount of data), but select those that are interesting for their project. To help with this, we provide a database containing the metadata and metrics of all episodes. This allows to filter the data before downloading. The dataset itself, the tools for filtering the episodes as well as a more technical description of the data format can be found on the dataset website[12].

6. Tasks and Results of RRC 2021

After the successful RRC 2020, we organised a second challenge from May to September 2021. The challenge was structured in the same way as the previous one but with new tasks.

Note on nomenclature ("phase" vs "stage"): While in 2020 the phases were called *phase 1,2,3*, the naming changed to *pre-stage* and *stage 1,2* in 2021. We keep the respective wording for each challenge, to be consistent with documentation and other publications.

6.1. Tasks

In the simulation-stage ("pre-stage") and in stage 1 the *move cube on trajectory* task had to be solved, in stage 2 the *rearrange dice* task. Figure 1 shows pictures of the different objects.

11. ardentstork: `https://github.com/ripl-ttic/real-robot-challenge`
troubledhare: `https://github.com/madan96/rrc_example_package`
sombertortoise: `https://github.com/stanford-iprl-lab/rrc_package`
12. `https://people.tuebingen.mpg.de/mpi-is-software/data/rrc2020`

6.1.1. Move Cube on Trajectory

The *move cube on trajectory* task is an extension of the task of RRC 2020 using the 65 mm cube (see Section 5.1). Instead of a single goal position, a list of sub-goals is given with the active sub-goal changing over time. So the cube has to be moved on a trajectory from goal to goal. The active sub-goal changes every 10 seconds, with an exception on the first one, which is active for 30 seconds to account for the additional time needed to pick up the cube. Each sub-goal is sampled independently somewhere within the arena. Only the position is considered, so the reward is computed as in Section 5.1.1.

6.1.2. Rearrange Dice

For the *rearrange dice* task the arena is filled with 25 dice (regular six-sided dice with a width of 22 mm), which are initially shuffled around randomly. They have to be arranged in a given 2d pattern, consisting of 25 goal positions on the ground of the arena.

To evaluate a given state, a ”goal mask” is created by projecting bounding cubes at the goal positions into the camera images. The error e is computed as the number of pixels in the segmentation mask (i.e. the pixels in which dice are visible, determined from color segmentation) that are not in the goal mask. Let G_c the set of pixels of the goal mask and S_c the pixels of the segmentation mask of camera $c \in \{1, 2, 3\}$.

$$e = \sum_{c \in \{1,2,3\}} |S_c \setminus G_c| \tag{3}$$

As for the other tasks, the reward is then simply the negated error: $r = -e$.

6.2. Challenge Results

After the pre-stage in simulation, six teams qualified for the real-robot stages. Four of the six teams submitted a solution in the end of stage 1 (see Appendix B for evaluation results and submission statistics).

It appears that the task for stage 2 was too difficult, since none of the teams managed to solve it, therefore we only discuss the results of stage 1 in the following.

The Winning Policies of Stage 1 Compared to the RRC 2020, the approaches were much more diverse in 2021:

- The winning team **thriftysnipe** McCarthy et al. (2021) used pure reinforcement learning (DDPG + HER) with minimal domain-specific knowledge.
- Team **decimalswift** Anonymous (2021) extended the CPC-TG approach from RRC 2020 Funk et al. (2021) (using position-based motion primitives for the finger tips) by interpolating trajectories between the goal positions for smoother motions.
- Team **grumpyzebra** Yao et al. (2021) used a mixed approach: They employ reinforcement learning to find good contact points on the cube and then use classic position control to move the finger tips.

Most teams published videos on YouTube[13] and open-sourced their code[14].

7. Takeaways from the Challenges

The following are a few interesting insights we gained from organizing these two challenges.

Reinforcement Learning Versus Classical Control: Interestingly, the most successful teams in the challenge of 2020 relied on traditional approaches to control rather than machine learning. In particular, hand-crafted state machines and motion primitives proved to perform well. However, the picture was more diverse in the challenge of 2021, where in stage 1 the winning team (thriftysnipe) solved the task using reinforcement learning and the runner-up team (decimalswift) employed an extension of a classical, control-based approach from the previous year. It is instructive to compare how these two completely different approaches perform: The classic approach from decimalswift achieves a stable grip of the cube and drops it rarely, but moves slowly. In contrast, the RL-based solution drops the cube more often, but it is able to recover very quickly and moves very fast. It learned that it pays off to move faster at the expense of making more mistakes, a strategy that would likely not have been found by an engineer hand-designing a policy.

Task Design: It can be hard to predict how engaging and difficult a task will be for participants and what solutions it will motivate. If the task is too easy, not much is gained by solving it and participants will not be engaged for long. On the other hand, if it is too hard, participants will also lose motivation quickly. Therefore, we are particularly excited about opening the platforms up to researchers for their own research projects. This may organically give rise to tasks that become benchmarks that are shared across the community.

Platform Design: The platform design proved to work very well. Participants learned quickly how to use the software interface and were able to create submissions with ease. During the hundreds of hours of operation, without any intervention from our side, there were almost no software and hardware failures. We addressed the rare issues that came up to make the system even more robust. Therefore, we will continue to use this setup to organize further challenges and to provide users with robot-access for their own research.

8. Conclusion

We designed and built a robot cluster to facilitate reproducible research into dexterous robotic manipulation. We believe that this cluster can greatly enhance coordination and collaboration between researchers all across the world. We hosted competitions on these robots to advance the state-of-the-art and to produce a publicly-available dataset that contains contact-rich interactions between the robots and external objects. These competitions validated our platform design, as participants were able to use the robots with ease and there were almost no hardware and software failures during the hundreds of hours of unsupervised operation. We now open these platforms to scientists around the world for their own research project.

13. thriftysnipe: https://www.youtube.com/playlist?list=PLLJoWXUn8XplFszi16-VZMTDBhMQFuc5o
decimalswift: https://youtu.be/dl0ueoaRWrM, grumpyzebra: https://youtu.be/Jr176xsn9wg
14. thriftysnipe: https://github.com/RobertMcCarthy97/rrc_phase1
grumpyzebra: https://github.com/42jaylonw/RRC2021ThreeWolves

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *The Ninth International Conference on Learning Representations (ICLR)*. arxiv.org, 2021. URL https://arxiv.org/abs/2010.04296.

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: Robotics benchmarks for learning with low-cost robots. *arXiv preprint arXiv:1909.11639*, 2019.

AI-DO: AI Driving Olympics – Duckietown. Website. https://www.duckietown.org/research/ai-driving-olympics. Accessed: 2021-8-27.

Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. *arXiv preprint arXiv:2108.09779*, 2021.

F Amigoni, E Bastianelli, J Berghofer, A Bonarini, G Fontana, N Hochgeschwender, L Iocchi, G Kraetzschmar, P Lima, M Matteucci, P Miraldo, D Nardi, and V Schiaffonati. Competitions for Benchmarking: Task and Functionality Scoring Complete Performance Assessment. *IEEE robotics & automation magazine / IEEE Robotics & Automation Society*, 22(3):53–61, September 2015.

Anonymous. Real robot challenge phase 2: Manipulating objects using high-level coordination of motion primitives. In *Submitted to Real Robot Challenge 2020*, 2020. URL https://openreview.net/forum?id=9tYX-lukeq. under review.

Anonymous. Real robot challenge stage 1: Cartesian position control with triangular grasp and trajectory interpolation. In *Submitted to Real Robot Challenge 2021*, 2021. URL https://openreview.net/forum?id=rERpgihmFq6. under review.

Sven 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.

F Bonsignorio and A P del Pobil. Toward Replicable and Measurable Robotics Research [From the Guest Editors]. *IEEE robotics & automation magazine / IEEE Robotics & Automation Society*, 22(3):32–35, September 2015.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

B Calli, A Walsman, A Singh, S Srinivasa, P Abbeel, and A M Dollar. Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set. *IEEE robotics & automation magazine / IEEE Robotics & Automation Society*, 22(3):36–52, September 2015a.

Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols. February 2015b.

Claire Chen, Krishnan Srinivasan, Jeffrey Zhang, Junwu Zhang, Lin Shao, Shenli Yuan, Preston Culbertson, Hongkai Dai, Mac Schwager, and Jeannette Bohg. Dexterous manipulation primitives for the real robot challenge, 2021.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. February 2018.

Niklas Funk, Charles Schaff, Rishabh Madan, Takuma Yoneda, Julen Urain De Jesus, Joe Watson, Ethan K. Gordon, Felix Widmaier, Stefan Bauer, Siddhartha S. Srinivasa, Tapomayukh Bhattacharjee, Matthew R. Walter, and Jan Peters. Benchmarking structured policies and policy optimization for real-world dexterous object manipulation, 2021.

Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Jonathan Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An Open Torque-Controlled Modular Robot Architecture for Legged Locomotion Research. In *International Conference on Robotics and Automation (ICRA)*, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. January 2018.

Nicolas Heess, T B Dhruva, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S M Ali Eslami, Martin Riedmiller, and David Silver. Emergence of Locomotion Behaviours in Rich Environments. July 2017.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Robert McCarthy, Francisco Roldan Sanchez, Qiang Wang, David Cordova Bulens, Kevin McGuinness, Noel O'Connor, and Stephen J. Redmond. Solving the real robot challenge using deep reinforcement learning, 2021.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In Maria Florina Balcan and Kilian Q Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of

Proceedings of Machine Learning Research, pages 1928–1937, New York, New York, USA, 2016. PMLR.

Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. PyRobot: An Open-source Robotics Framework for Research and Benchmarking. June 2019.

Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, Daniel Hoehener, Shih-Yuan Liu, Michael Novitzky, Igor Franzoni Okuyama, Jason Pazis, Guy Rosman, Valerio Varricchio, Hsueh-Cheng Wang, Dmitry Yershov, Hang Zhao, Michael Benjamin, Christopher Carr, Maria Zuber, Sertac Karaman, Emilio Frazzoli, Domitilla Del Vecchio, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504, May 2017.

Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus 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.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. April 2017.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Manuel Wüthrich, Felix Widmaier, Felix Grimminger, Joel Akpo, Shruti Joshi, Vaibhav Agrawal, Bilal Hammoud, Majid Khadiv, Miroslav Bogdanovic, Vincent Berenz, Julian Viereck, Maximilien Naveau, Ludovic Righetti, Bernhard Schölkopf, and Stefan Bauer. TriFinger: An Open-Source Robot for Learning Dexterity. In *Proceedings of the Conference on Robot Learning (CoRL)*, August 2020.

Brian Yang, Jesse Zhang, Vitchyr Pong, Sergey Levine, and Dinesh Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019.

Qingfeng Yao, Jilong Wang, and Shuyu Yang. Real-world dexterous object manipulation based deep reinforcement learning, 2021.

Takuma Yoneda, Charles Schaff, Takahiro Maeda, and Matthew Walter. Grasp and motion planning for dexterous manipulation for the real robot challenge, 2021.

Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.