Experimental Protocol for real world testing of RF-DROPO

Andrea Protopapa[1], Gabriele Tiboni[1],
Tatiana Tommasi[1], Giuseppe Averta[1], Raffaello Camoriano[1*]

Thursday 2nd November, 2023

1 Purpose

The objective of this experimental study is to investigate the effectiveness of domain randomization in closed-loop control for soft robots. The study aims to assess whether the control strategies learned in a simulated environment with randomized parameters are robust to sim-to-real shift and able to accurately control a real manipulator. The experiments are based on the previous work presented in [1].

2 Materials

2.1 Requested

- **Trunk robot**
  - Silicone cable-driven continuum deformable manipulator, as described in [2].
    - Conical shape.
    - Actuated by eight cables passing through its silicone body. Four cables are attached to the middle of the trunk-robot and actuate the proximal part, while the other four go through its entire length, actuating the distal part.
      - Each action corresponds to a cable displacement of $\pm 5mm$.
    - Tracked using 21 points along each trunk cable: starting from the base, one point at the beginning and at the end of each of the 11 parts, without sensing the farthest tip (see Fig. 1).
      - Note: this state space configuration matches what was used in the simulation model of our previous work [1]. If required for real experiments, it can be adjusted to accommodate the feasibility of the physical prototype.

[*1]Politecnico di Torino, Turin, Italy `first.last@polito.it`

        - Details (indicatively):
            - Length: 200 mm
            - Mass: 0.42 kg
            - Poisson's ratio: 0.45
            - Young's Modulus: 4500 kg/(mm · s)

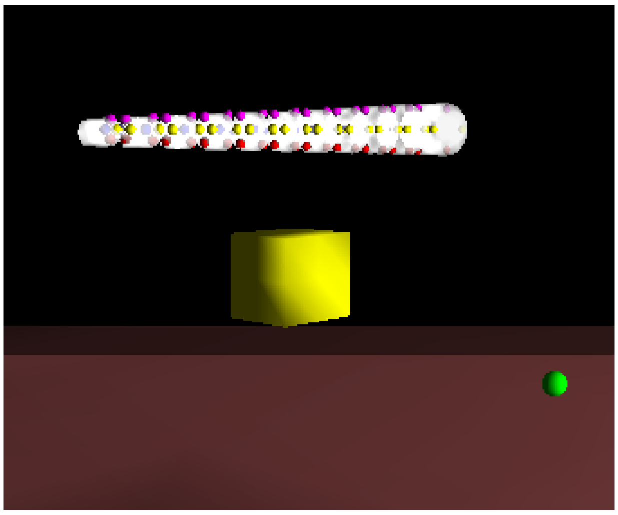

Figure 1: Initial configuration of the *TrunkPush* task during the simulation. The trunk, starting from an horizontal position, can be tracked by 21 points along each cable, here shown by different colors.

- **Cube**
    - Rigid cube, used for the manipulation task described in the *TrunkPush task* section.
    - Position and orientation can be tracked using an arbitrary set of markers asymmetrically placed on the upper face of the cube, assuming that the cube will not turn upside down (see Fig. 2 for an example representation).
    - Details (indicatively):
        - Shape (l × w × h): 50 mm × 50 mm × 50 mm
        - Mass: 0.05 kg
- **Motion capture system**
    - e.g., OptiTrack, Vicon
    - Needed to track the pose of the system objects (i.e., the robot, the cube, the goal point), as required by the state space of each task.

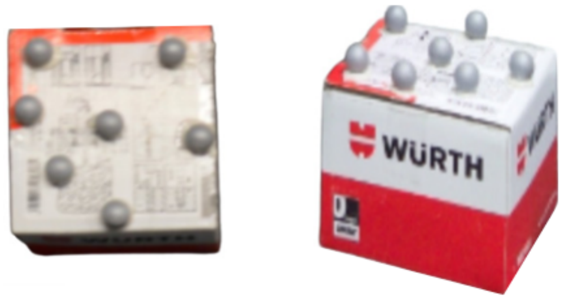

Figure 2: Exemplificative markers used to track the position and orientation of the cube with a motion capture system (e.g., OptiTrack).

2.2 Provided

- approx. 25 policies per task learned following the baselines presented in [1]:
  - approx. 5 policies for RF-DROPO (our method)
  - approx. 5 policies for BayesSim [3]
  - approx. 10 policies for UDR (Uniform Domain Randomization)
  - approx. 5 policies for NPDR [4] (added for the Journal Extension)

3 Experimental design

Regarding Adaptive Domain Randomization (ADR) methods - such as *RF-DROPO*, *BayesSim*, *NPDR* - each experimental pipeline consists in 4 different phases:

1. *Data collection*: A fixed dataset $\mathcal{D} = \{s_0, a_0, s_1, ..., s_{T-1}, a_{T-1}, s_T\}$ $\{(s_t, a_t)\}_{t=1}^{t=100}$ with real target domain state-action transitions shall be made available to run the inference phase. Such data may be collected offline (i.e., before the optimization phase) with any desirable strategy, such as kinesthetic teaching, human demonstrations, motor babbling or hard-coded policies. It is important to collect trajectories which well describe all the dynamics (e.g., for the *TrunkPush* task, it is needed to collect interactions between the trunk and the cube). **The same dataset is used for all seeds and all methods of the inference training.** Note that no reward computation in the real-world is needed, as the inference phase relies on state-action transition pairs only.

2. *Dynamics parameter inference*: estimation of the dynamics probability distribution $p_\phi(\xi)$. Given the previous collected dataset $\mathcal{D}$, the inference phase is done separately, without any real-world interaction.

3. *Policy Training*: The converged distribution $p^*_\phi(\xi)$ obtained through the inference phase is used to train a policy $\pi_\theta(a|s)$ entirely in simulation using Domain Randomization for the given task.

4. *Real world evaluation*: The optimal policy $\pi_\theta^*(a|s)$ learned in simulation is transferred and used in the real-world for the final evaluation on the real prototype, assessing the *Sim-to-Real transfer*.

Therefore, for these methods each experiment **requires two phases of real-world interaction** (step 1. and 4.), one for the data collection and another one for the policy evaluation. An overview of the experimental paradigm is illustrated in Fig. 3.

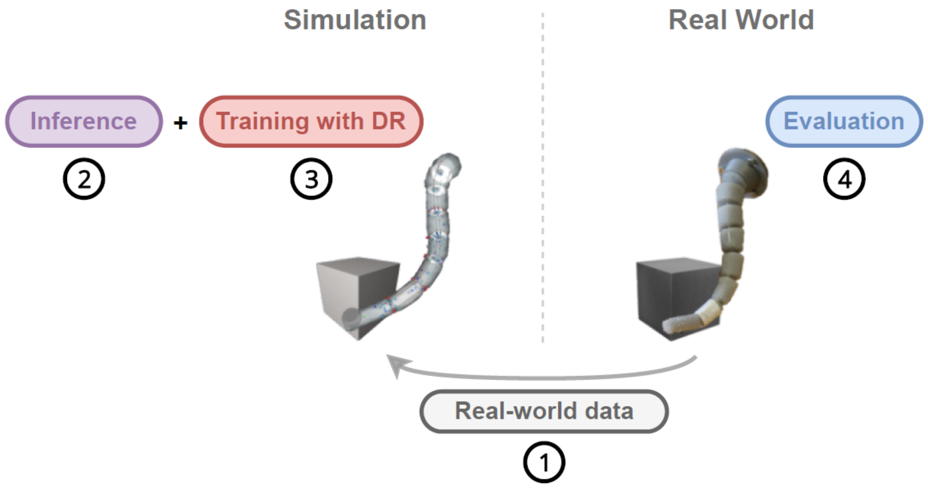

Figure 3: Experimental Paradigm

On the other hand, Static Domain Randomization methods, such as *UDR*, **only require the final evaluation in the real world** (step 4.), since the dynamics distributions are not inferred but statically fixed.

3.1 TrunkReach task

- **Note**: this task must be considered as a test case example, useful to assess the quality of the sim-to-real transfer in an easier scenario, if resources and time are enough. The most of the effort should be focused on the *TrunkPush* task – see Sec. 3.2.
- The robot should reach a point randomly located in space (i.e., a *goal* point) using the robot's endpoint, as depicted in Fig. 4-a.
- **Action space**: 16 discrete possible actions, consisting in contracting or extending each of the eight cables by one unit – see Sec. 4.1 for more details on the action unit.
- **State space**: 66 element array $S = [p_t, p_g]$
  - p_t: 63 element array representing 21 positional 3D key points along the robot. The points are computed taking $21 \cdot 4$ points along the four longest trunk cables, which are then averaged to compute the center point of 21 cross sections along the body.

    - p_g: 3D position of the goal point.

- **Experiments required**:
    - approx. 5 evaluation rollouts for each policy.
    - During the evaluation, the robot is asked to reach 27 different fixed goals, which are chosen by design as a three-dimensional grid of points contained inside the training box, specifically ranged in the space bounds $x_g^{eval} = [-20, 20]$ mm, $y_g^{eval} = [-35, 5]$ mm, $z_g^{eval} = [120, 145]$ mm.
    - Results must be reported in terms of l^2-norm distance [mm] between the final position of the trunk's tip and the target goal.

3.2 TrunkPush task

- The robot aims to push a rigid cube to a designated target location, as depicted in Fig. 4-b.
- **Action space**: 16 discrete possible actions, consisting in contracting or extending each of the eight cables by one unit – see Sec. 4.1 for more details on the action unit.
- **State space**: 73 element array $S = [p_t, p_g, p_c]$
    - p_t: 63 element array representing 21 positional 3D key points along the robot. The points are computed taking $21 \cdot 4$ points along the four longest trunk cables, which are then averaged to compute the center point of 21 cross sections along the body.
    - p_g: 3D position of the goal point.
    - p_c: 3D position and orientation of the cube (given by 7 DOFs: the translation of the rigid $[x_c, y_c, z_c]$ and the quaternion for the rotation $[r_{x_c}, r_{y_c}, r_{z_c}, r_{w_c}]$).
- **Experiments required**:
    - approx. 5 evaluation rollouts for each policy.
    - During the evaluation, a specific fixed target goal is set to $p_g^{eval} = [100.0, -100.0, 50.0]$ mm.
    - Results must be reported in terms of l^2-norm distance [mm] between the final position of the cube (i.e., of its center of mass) and the target goal considering xz-axis only.
- **Details**:
    - Floor position: $y_f = -100$ mm
    - Initial cube position (simulation): $[x_c, y_c, z_c] = [0.0, -60.0, 100.0] mm$
    - Initial cube position (real-world): $[x_c, y_c, z_c] = [0.0, -100.0, 100.0] mm$
    - Initial cube orientation: $[a_{x_c}, a_{y_c}, a_{z_c}] = [0, 0, 0]^\circ$
    - Initial trunk position: horizontal (see Fig. 1)

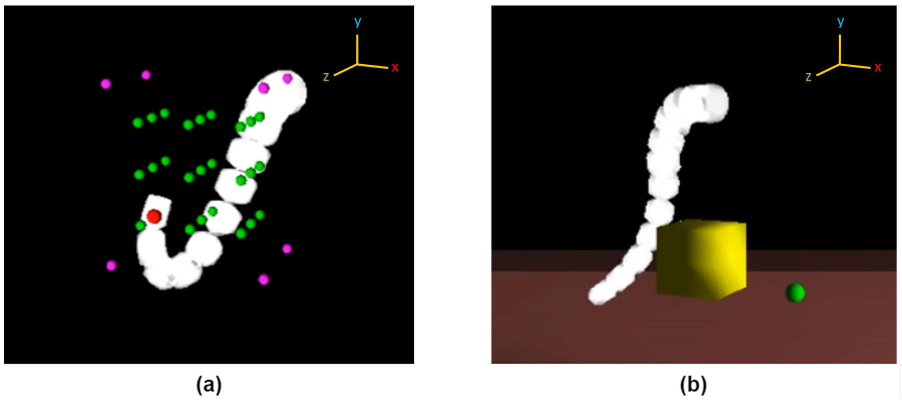

Figure 4: *TrunkReach* and *TrunkPush* setups. a) Purple dots define the box of possible goal locations sampled at training time, green dots are 27 fixed target locations for evaluation, with the red dot being the current goal. b) Green dot is the desired target location for the box center of mass.

4 Additional simulation details

- The origin of the Cartesian reference system $[x_o, y_o, z_o]$ is set to the center of the base of the trunk.
- Friction coefficient: 0.3 (used in simultation, not necessarily the same in the real world).
- Episode length: $T = 100$ policy timesteps (5 s).
- Policy query frequency: 20 Hz.
- Low-level control frequency: 100 Hz.
- Simulated sizes:
  - Trunk's length $= 195mm$
  - Shorter cables length $= 91mm$
  - Longer cables length $= 185mm$

4.1 Simulated actuation unit length and repetition

- Each cable is embedded in the elastic trunk. The cable is free to slide through the material. When the displacement is controlled (for example, with a servo motor), the simulation will solve for the force that is required to obtain that displacement.
  - For each action, a displacement ± 1 is imposed to a certain cable: the displacement $(\pm 1mm)$ is added to the actual value of the displacement.
- **Simulation step**: apply an action and execute $scale\_factor$ simulation steps of dt s.
  - The simulation step is equal to: $dt \cdot scale\_factor$

        - $dt = 0.01s$
        - $scale\_factor = 5$
    - Therefore, the same displacement value is applied throughout $scale\_factor$ simulation cycles.
        - i.e., **$\pm 5mm$ for each simulated action**

4.2 Further details

- Physics Engine used: SOFA [5].
- Software toolkit used for managing Reinforcement Learning algorithms: SofaGym [6].
- All the policies provided are learned using Proximal Policy Optimization (PPO) as a Reinforcement Learning algorithm. The implementation of the algorithms is based on Stable-Baselines3 (SB3) [7].

References

[1] G. Tiboni, A. Protopapa, T. Tommasi, G. Averta, "Domain Randomization for Robust, Affordable and Effective Closed-loop Control of Soft Robots," 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023), Detroit, Michigan, USA. `https://arxiv.org/abs/2303.04136`

[2] C. Agabiti, E. Ménager and E. Falotico, "Whole-arm Grasping Strategy for Soft Arms to Capture Space Debris" 2023 IEEE International Conference on Soft Robotics (RoboSoft), Singapore, Singapore, 2023. `https://ieeexplore.ieee.org/abstract/document/10122076`

[3] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators" (2019). `https://arxiv.org/abs/1906.01728`

[4] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, "Neural Posterior Domain Randomization" (2022). `https://proceedings.mlr.press/v164/muratore22a.html`

[5] E. Coevoet et al., "Software toolkit for modeling, simulation, and control of soft robots," Advanced Robotics 31.22 (2017), pp. 1208–1224. `https://inria.hal.science/hal-01649355/preview/AR_2017.pdf`

[6] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "SofaGym: An open platform for Reinforcement Learning based on Soft Robot simulations" (2022). `https://inria.hal.science/hal-03778189/preview/SofaGym.pdf`

[7] Stable Baselines3 Documentation. (accessed on Thursday 2nd November, 2023). Retrieved from `https://stable-baselines3.readthedocs.io/en/master/`