CHALMERS
UNIVERSITY OF TECHNOLOGY

Open Source Dual-Purpose Acrobot and Pendubot Platform: Benchmarking Control Algorithms for Underactuated Robotics

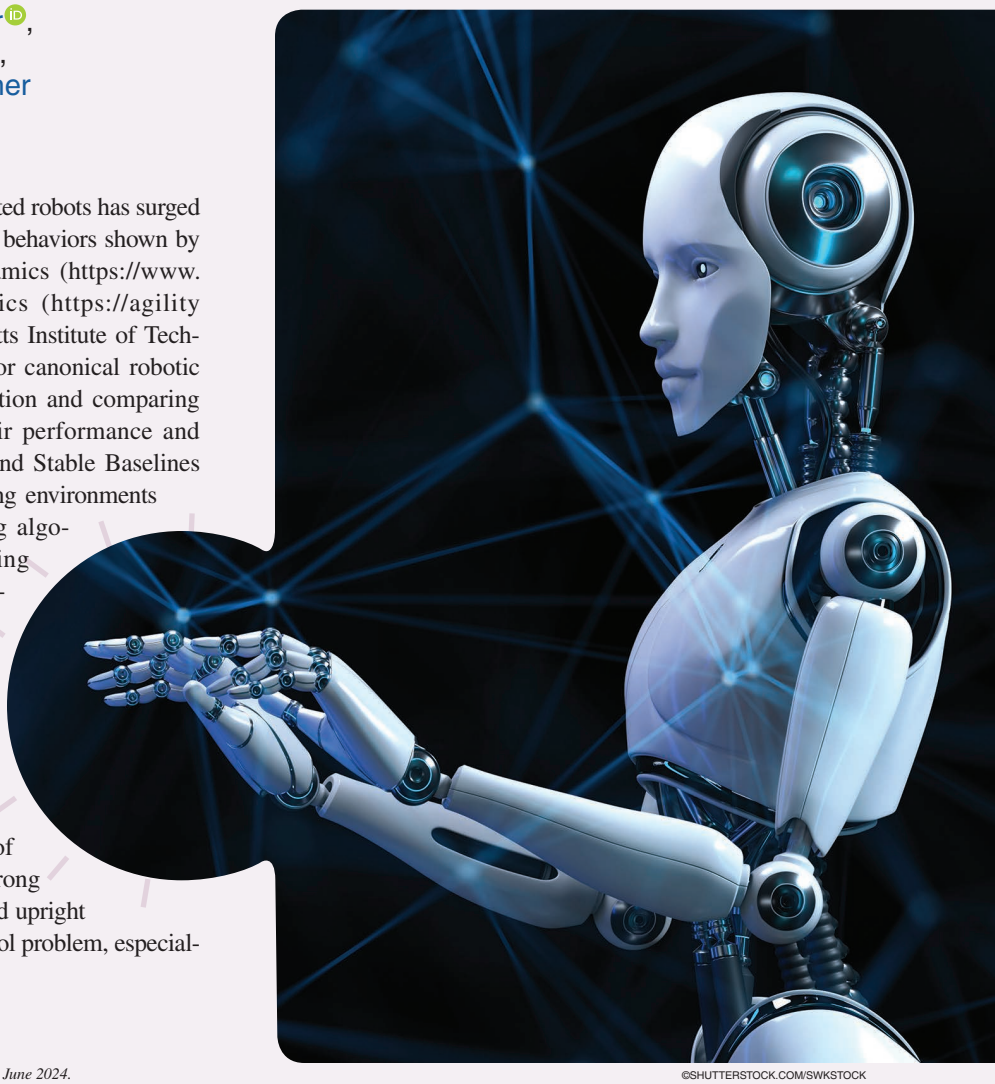
Citation for the original published paper (version of record):
Wiebe, F., Kumar, S., Shala, L. et al (2024). Open Source Dual-Purpose Acrobot and Pendubot Platform: Benchmarking Control Algorithms for
Underactuated Robotics. IEEE Robotics and Automation Magazine, 31(2): 113-124.
http://dx.doi.org/10.1109/MRA.2023.3341257

(article starts on next page)

Open Source Dual-Purpose Acrobot and Pendubot Platform

Benchmarking Control Algorithms for Underactuated Robotics

By Felix Wiebe, Shivesh Kumar, Lasse J. Shala, Shubham Vyas, Mahdi Javadi, and Frank Kirchner



Recent interest in the control of underactuated robots has surged significantly due to the impressive athletic behaviors shown by robots developed by, e.g., Boston Dynamics (https://www.bostondynamics.com), Agility Robotics (https://agilityrobotics.com/robots), and the Massachusetts Institute of Technology [1]. This gives rise to the need for canonical robotic hardware setups for studying underactuation and comparing learning and control algorithms for their performance and robustness. Similar to OpenAIGym [2] and Stable Baselines [3], which provide simulated benchmarking environments and baselines for reinforcement learning algorithms, there is a need for benchmarking learning and control methods on real canonical hardware setups. To encourage reproducibility in robotics and artificial intelligence research, these hardware setups should be affordable and easy to manufacture with off-the-shelf components, and the accompanying software should be open source. Acrobots and pendubots are classical textbook examples of canonical underactuated systems with strong nonlinear dynamics, and their swing-up and upright balancing is considered a challenging control problem, especially on real hardware.

Digital Object Identifier 10.1109/MRA.2023.3341257
Date of publication 28 December 2023; date of current version 14 June 2024.

INTRODUCTION

MOTIVATION

This article presents an open source and low-cost test bench for validating, comparing, and benchmarking the performance of control algorithms for underactuated robots with strong nonlinear dynamics. It introduces a double-pendulum platform built using two off-the-shelf quasi-direct drives (QDDs). Due to the low friction and high mechanical transparency offered by QDDs, one of the actuators can be kept passive and used as an encoder so that the system can be operated as a double pendulum, a pendubot, or an acrobot without changing the hardware. Using the proposed platform, trajectory optimization and control algorithms for the swing-up and upright stabilization of the acrobot and pendubot systems are compared and benchmarked. We show that, by considering simple variations of the design, the difficulty of the control problem can be varied, giving researchers an opportunity for showing the robustness of their control algorithms. We demonstrate the transfer of one exemplary controller from simulation to real hardware, with successful swing-ups on the pendubot and acrobot.

> “ACROBOTS AND PENDUBOTS ARE CLASSICAL TEXTBOOK EXAMPLES OF CANONICAL UNDERACTUATED SYSTEMS WITH STRONG NONLINEAR DYNAMICS.”

RELATED WORK

The acrobot is an underactuated robotic system inspired by a gymnastic acrobat and was first introduced in [4]. Following this, a large body of research on the dynamics and control of such systems was carried out. However, over the years, works carried out experimentally have been few and far between in comparison to theoretical work demonstrated only in simulation. The earliest work on balancing an experimental acrobot at the topmost position and other unstable equilibrium points can be seen in [5], where a (single-input) pseudolinearizing controller was used for the balancing task. Along with this, a region of attraction (ROA) analysis was performed for the given controller. The initial problem of swing-up and balance for the acrobot was formulated and demonstrated on a real system in [6], where partial feedback linearization (PFL) and energy-based control were used to perform the swing-up while a linear quadratic regulator (LQR) was used to balance at the top. A first comparative study on balancing controllers was carried out in [7] with both theoretical ROA analysis and experimental validation along with introducing an additional balancing controller. Following this, an energy-based swing-up designed using Lyapunov stability theory with LQR for top stabilization was showcased in experiments in [8]. In recent works, sums-of-squares (SOS)-based methods have been used to synthesize robust controllers for swing-up trajectory tracking and top balancing [9]. Furthermore, online trajectory planning for the swing-up and balance task using model predictive control (MPC) with particle swarm optimization was demonstrated on a physical system in [10]. Recently, for the first time, both swing-up and balancing were achieved with a single controller using the stable manifold approach [11]. It is interesting to note that all experimental implementations of the acrobot mount the actuator at the base link, and a transmission is used for actuating the second joint in order to minimize moving inertia. For the pendubot system, which, in contrast to the Acrobot, has the first joint actuated, a larger body of research can be found for experimental results (e.g., [12] and [13]). Due to mechanical design and control complexity, all reported platforms in the literature were constructed for a single purpose, either a pendubot or an acrobot. Only a recent parallel development showed a system that could potentially be used as a testbed for both the pendubot and acrobot [14]. The system is provided with open source MATLAB code. However, the system is complex to construct due to the use of belt transmissions, and the provided software requires a MATLAB license, which increases the accessibility barrier. Until now, a fully open source test platform for acrobot/pendubot-type systems with direct joint actuation was not available.

MECHATRONIC SYSTEM DESIGN

This section presents the mechatronic system design of the dual-purpose acrobot-pendubot hardware.

MECHANICAL DESIGN

The mechanical design (see Figure 1) consists of a shoulder motor mounting bracket made of folded aluminum and two lightweight links that are made of laser-cut 1-mm-thick sandwich aluminum plates with a laminate of 15-mm polyvinyl chloride rigid foam board (Airex) in between. By using sandwich materials, the weight of the pendulum arms can be kept very low in relation to the drives and the end-effector weight. The end of the first link contains the elbow motor housing, and the end of the second link mounts the weight of 0.5 kg. Two variations of both links (0.2 and 0.3 m long) are manufactured, which allows changing the complexity of the control problem. Since the used motors do not provide a hollow shaft, a cabling guide is mounted to the first link in the opposite direction to prevent the windup of the cables.

ELECTRONICS AND PROCESSING ARCHITECTURE

Both the shoulder and elbow actuators consist of off-the-shelf AK80-6 QDDs from T-Motor (https://store.tmotor.com/goods.php?id=981) with a gear ratio of 6:1, maximum speed

of 38.22 rad/s^{-1}, maximum continuous torque of 6 N·m, and peak torque of 12 N·m. The low friction offered by these motors enables chaotic dynamics in the system, which is interesting for control purposes [see Figure 1(b) for its free fall]. The two motors communicate with a standard Intel Core-i7 PC via a controller area network (CAN)–USB 2.0 interface from ESD. The setup allows real-time position, velocity, and torque control, with a control frequency of 1 kHz with Python on a standard PC. The power supply used is EA-PS 9032-40 from Elektro-Automatik, which can provide a maximum voltage of 36 V and 48 A of current. A capacitor bank of 10 single 2.7-V/400-F capacitor cells connected in series, resulting a total capacity of 40 F, is wired in parallel to the motor to protect the power supply from back electromotive force. An emergency stop button, which disconnects the actuator from the power supply and capacitor, is also integrated as an additional safety measure. A schematic of the setup can be found in Figure 2.

METHODOLOGY

This section gives an overview of the mathematical modeling of the double pendulum, the identification of the dynamic

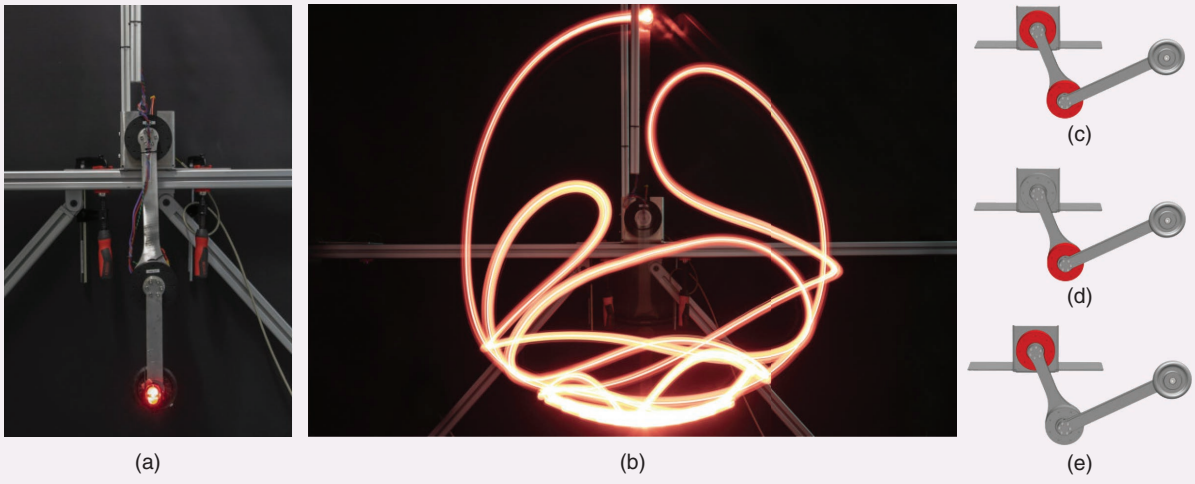


FIGURE 1. The double-pendulum test bench. (a) The hardware setup. (b) A long-exposure shot of the free-falling double pendulum. The use as (c) a fully actuated double pendulum, (d) an acrobot, and (e) a pendubot by selectively activating only the needed motor(s) (colored in red).

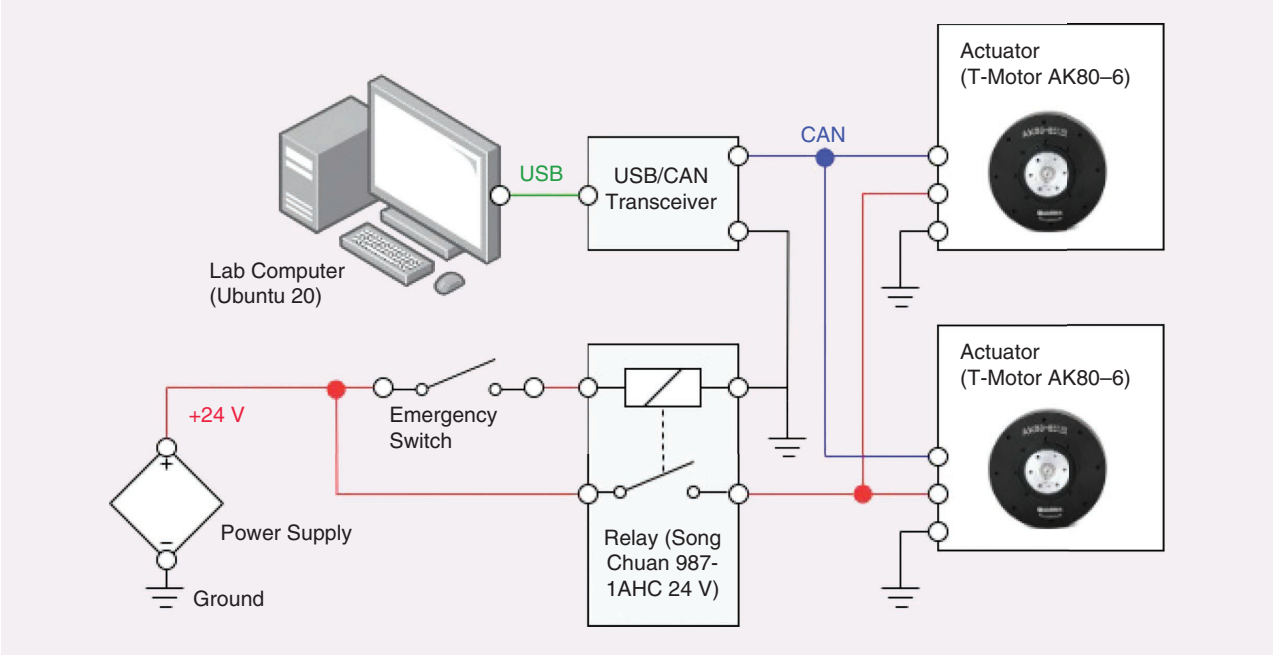


FIGURE 2. The mechatronic system design of the dual-purpose double pendulum.

parameters, and the methods used for controlling the double pendulum as an acrobot and a pendubot.

DYNAMICS

We model the dynamics of the double pendulum with 15 parameters, which include eight link parameters, namely, masses (m_1, m_2), lengths (l_1, l_2), centers of mass (r_1, r_2), and inertias (I_1, I_2), for the two links and six actuator parameters, namely, motor inertia (I_r), gear ratio (gr), coulomb friction (c_{f1}, c_{f2}), viscous friction (b_1, b_2) for the two joints (we found that the friction parameters for two actuators of the same type can be different due to manufacturing differences), and gravity (g). The generalized coordinates $\boldsymbol{q} = (q_1, q_2)^T$ are the joint angles measured from the free-hanging position. The state vector of the system contains the coordinates and their time derivatives: $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})^T$. The torques applied by the actuators are $\boldsymbol{u} = (u_1, u_2)$. The equations of motion of a dynamical system can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

$$= \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{D}\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{F}(\dot{\boldsymbol{q}})) \end{bmatrix}. \tag{2}$$

The dynamic matrices $\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{F}$, and $\boldsymbol{D}$ that we use to describe our double-pendulum test bench can be found in the supplementary material available at https://www.doi.org/10.1109/MRA.2023.3341257.

SYSTEM IDENTIFICATION

For the identification of the 15 model parameters, we fix the natural, provided, and easily measurable parameters g, g_r, l_1, and l_2. The equations of motion are then linear in the following (composed) model parameters:

$$m_1 r_1, m_2 r_2, m_2, I_1, I_2, I_r, b_1, b_2, c_{f1}, c_{f2}. \tag{3}$$

By executing excitation trajectories on real hardware, data tuples of the form $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{u})$ can be recorded. For finding the best system parameters, one can make use of the fact that the dynamic matrices $\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{G}$, and $\boldsymbol{F}$ are linear in the parameters in (3) and perform a least-squares optimization for the equations of motion on the recorded data.

BALANCING WITH LQR

The LQR controller is a well-established and widespread optimal controller that acts on a linear system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ and an objective that is specified by a quadratic instantaneous cost function $J = \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}$ with the symmetric and positive definite matrices $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0$ and $\boldsymbol{R} = \boldsymbol{R}^T \succ 0$. This allows for reducing the Hamilton–Jacobi–Bellman equation to the algebraic Riccati equation, for which good numerical solvers exist. Its solution is the optimal cost-to-go matrix $\boldsymbol{S}$, from which the optimal policy can be inferred:

$$\boldsymbol{u}(\boldsymbol{x}) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{S}\boldsymbol{x} = -\boldsymbol{K}\boldsymbol{x}. \tag{4}$$

In order to use an LQR controller for stabilizing the double pendulum on the top, the dynamics have to be linearized around the top position $\boldsymbol{x}^d = [\pi, 0, 0, 0]$ and $\boldsymbol{u}^d = [0, 0]$, and the state and actuation have to be expressed in relative coordinates $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}^d, \tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}^d$.

For the double pendulum with LQR control, x^d represents a stable fixed point, and the ROA $\mathcal{B}$ around that fixed point describes the set of initial states for which $\boldsymbol{x} \to \boldsymbol{x}^d$ as $t \to \infty$. Direct computation of this set is often not possible. However, it can be estimated by considering the sublevel set of a Lyapunov function $V(\boldsymbol{x})$. When using LQR to stabilize the system around $\boldsymbol{x}^*$, the cost to go can serve as a quadratic Lyapunov function. In this case, the estimated ROA can be written as $\mathcal{B}_{\text{est}} = \{\boldsymbol{x} \mid \boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x} \leq \rho\}$, where ρ is a scalar that can be estimated using either probabilistic or optimization-based methods.

TRAJECTORY OPTIMIZATION WITH ITERATIVE LQR

Iterative LQR (iLQR) [15] is an extension of LQR to nonlinear dynamics. The LQR uses fixed-point linearized dynamics for the entire state space and hence is useful only as long as the linearization error is small. In contrast to LQR, iLQR linearizes the dynamics for every given state at each time step and can deal with nonlinear dynamics at the cost of being able to optimize only over a finite time horizon.

As a trajectory optimization method, iLQR solves the following optimization problem:

$$\min_{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}} \boldsymbol{x}_N^T\boldsymbol{Q}_f\boldsymbol{x}_N + \sum_{i=0}^{N-1} (\boldsymbol{x}_i^T\boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{u}_i^T\boldsymbol{R}\boldsymbol{u}_i)$$
$$\text{subject to} \quad \boldsymbol{x}_{i+1} = f_{\text{discrete}}(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{5}$$

where a start state $\boldsymbol{x}_0$ is set beforehand. Here, $\boldsymbol{Q}_f$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are cost matrices penalizing the final state, intermediate states, and control input, respectively; f_{discrete} is the discretization of the system dynamics in (1). Again, $\boldsymbol{x}$ and $\boldsymbol{u}$ can also be expressed in relative coordinates $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}$.

TRAJECTORY STABILIZATION CONTROLLERS

TIME-VARYING LQR

Time-varying LQR (TVLQR) is another extension of the regular LQR algorithm and can be used to stabilize a nominal trajectory $[\boldsymbol{x}^d(t), \boldsymbol{u}^d(t)]$. For this, LQR formalization is used for time-varying linear dynamics,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)(\boldsymbol{x} - \boldsymbol{x}^d(t)) + \boldsymbol{B}(t)(\boldsymbol{u} - \boldsymbol{u}^d(t)) \tag{6}$$

which requires linearizing (1) at all steps around $[\boldsymbol{x}^d(t), \boldsymbol{u}^d(t)]$. This results in the optimal policy at time t:

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}^d - \boldsymbol{K}(t)(\boldsymbol{x} - \boldsymbol{x}^d(t)). \tag{7}$$

iLQR WITH RICCATI GAINS

During the iLQR optimization process, the trajectory is altered with Riccati gain matrices. During the execution, these Riccati gains can be used to stabilize the trajectory, as discussed in [16].

iLQR MPC STABILIZATION

iLQR is a shooting method and thus has the property that all trajectories during the optimization process are physically feasible, so, even when stopped before convergence, the solution is not inconsistent. This has the advantage that iLQR can be used in an MPC ansatz. For this, the optimization is performed online, and at every time step, the first control input $\boldsymbol{u}_0$ is executed. For the next time step, the previous solution is used to warm start the next optimization step. For stabilizing a nominal trajectory, the iLQR optimization problem (5) is solved with time-varying desired states $\boldsymbol{x}^d = \boldsymbol{x}^d(t)$ and inputs $\boldsymbol{u}^d = \boldsymbol{u}^d(t)$.

> “WHEN TRANSFERRING CONTROLLERS FROM SIMULATION TO REAL HARDWARE, MANY EFFECTS THAT ARE NOT PRESENT IN SIMULATION MAY INFLUENCE THE BEHAVIOR.”

POLICY-BASED CONTROLLERS

iLQR MPC (FREE)

The iLQR MPC method can also solve the full optimization problem online without a nominal trajectory. For this, the optimization problem is solved with a fixed goal state $\boldsymbol{x}^d$.

PFL

PFL [6] is a classical method from control theory. With PFL, it is possible to provoke a linear response in both joints of the double pendulum even if operated as a pendubot or an acrobot. For an intuition of its functionality, consider the lower part of (2) for the acrobot ($u_1 \equiv 0$). The unactuated upper part of the vector equation can be solved for the acceleration $\ddot{q}_1$ and then plugged into the lower part of the equation. The control input u_2 can now be designed as proportional derivative control with an energy term,

$$u_2(\boldsymbol{x}) = -k_p(q_2 - q_2^d) - k_d\dot{q}_2 + k_e(E - E^d)\dot{q}_1 \tag{8}$$

with the desired configuration q_2^d of the second link, the total energy E, the desired total energy E^d, and the gain parameters k_p, k_d, and k_e. This method is called *collocated PFL*. Similarly, it is also possible to eliminate $\ddot{q}_2$ instead of $\ddot{q}_1$ from the equations, which is called *noncollocated PFL*. PFL for the pendubot can be done in the same way. The collocated control law in this case reads

$$u_1(\boldsymbol{x}) = -k_p(q_1 - q_1^d) - k_d\dot{q}_1 + k_e(E - E^d)\dot{q}_2. \tag{9}$$

CONTROLLER COMPARISON

This section explains how the design parameters were chosen, states the results from the dynamic system identification, introduces the controller robustness criteria, and presents the results of the controller comparison.

SYSTEM DESIGN

Our double-pendulum setup allows the use of different link lengths l_1 and l_2. Under the assumption that the acrobot and pendubot each benefit from different ratios of link lengths, we aimed to find two designs, one tailored to the acrobot and the other to the pendubot configuration. In the following, we consider the closed-loop dynamics of the system under LQR control and focus on the volume of the ROA associated to the fixed point of the upright pose. Link lengths l_1 and l_2 have been determined by employing a design optimization similar to the one introduced in [17], with the only difference being that the ROA estimation was carried out using SOS optimization.

During the optimization, the masses were kept constant ($m_1 = m_2 = 0.6$ m), and we assumed point masses ($r_1 = l_1, r_2 = l_2, I_1 = m_1 l_1^2, I_2 = m_2 l_2^2$). The LQR control weights, the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, were set to unit matrices. We searched for lengths between 0.2 and 0.3 m to ensure that we could actually construct and operate the hardware.

The optimizations resulted in two designs, $\mathcal{D}_1$ and $\mathcal{D}_2$. Design $\mathcal{D}_1$ was optimized for the pendubot and features a longer first link ($l_1 = 0.3, l_2 = 0.2$ m), while $\mathcal{D}_2$ was optimized for the acrobot and has switched link lengths ($l_1 = 0.2, l_2 = 0.3$ m). The left-hand side of Figure 3 shows the volume of $\mathcal{B}_{\text{est}}$ for the acrobot [Figure 3(a)] and pendubot [Figure 3(c)] as a function of the design variables l_1 and l_2. The right-hand side shows slices of the estimated ROA of each model in the q_1-versus-q_2 plane (for $\dot{q}_1 = \dot{q}_2 = 0$) for the acrobot [Figure 3(b)] and pendubot [Figure 3(d)]. Recall that every initial state that lies inside the ellipse belongs to the estimated ROA, for which the closed-loop dynamics will bring the system back to the upright pose. Hence, a larger (projected) ROA is associated with greater robustness with respect to off-nominal initial states.

One can see on the right-hand side of Figure 3 that the volume of the estimated ROA of the closed-loop dynamics of the top LQR $\mathcal{B}_{\text{est}}$ of $\mathcal{D}_2$ is larger than that of $\mathcal{D}_1$ for the acrobot configuration (1.2 versus $2.3 \cdot 10^{-4}$), while the converse holds for the pendubot (0.014 versus 0.037). Note that even though the ROA of the $\mathcal{D}_2$ acrobot is larger than that of the $\mathcal{D}_1$ pendubot, the estimated ROA of the latter has a larger minor axis (0.014 versus 0.021), which allows more evenly distributed perturbations around the fixed point.

SYSTEM IDENTIFICATION

For the system identification, we recorded the data of multiple excitation trajectories with a combined length of about

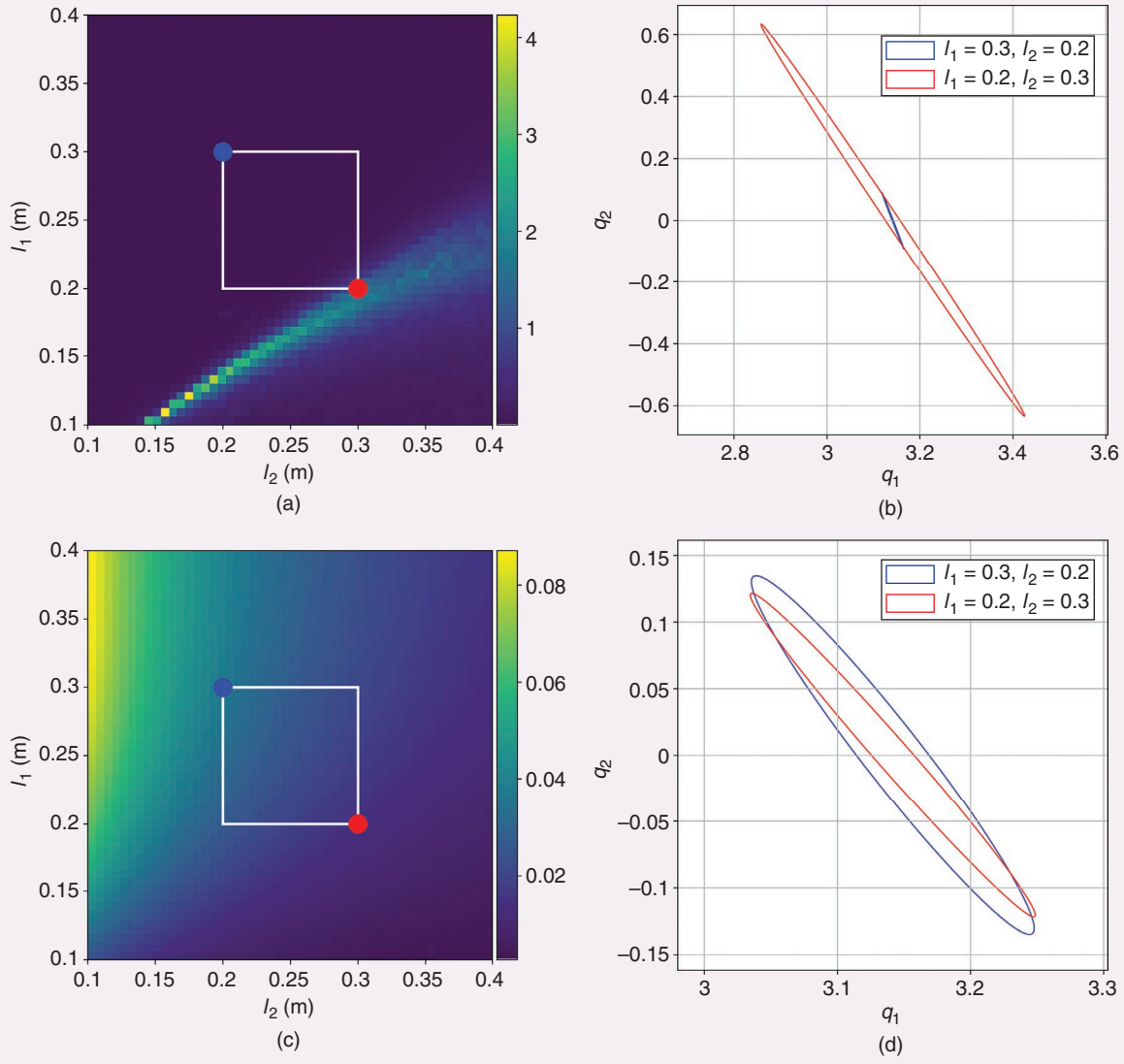


FIGURE 3. The ROA volume as a function of design parameters l_1 and l_2 and associated ROA projections of two design variations, $\mathcal{D}_1$ (blue) and $\mathcal{D}_2$ (red). The (a) acrobot ROA volume, (b) acrobot ROA ellipse, (c) pendubot ROA volume, and (d) pendubot ROA ellipse.

TABLE 1. The model parameters.

PARAMETER	MODEL $\mathcal{M}_1$	MODEL $\mathcal{M}_2$
Mass m (kg)	(0.55, 0.6)	(0.64, 0.56)
Length l (m)	(0.3, 0.2)	(0.2, 0.3)
Center of mass r (m)	(0.3, 0.183)	(0.2, 0.32)
Inertia I (kg/m^2)	(0.053, 0.024)	(0.027, 0.054)
Motor inertia I_r (kg/m^2)	$6.29{\cdot}10^{-5}$	$9.94{\cdot}10^{-5}$
Gear ratio g_r	6	6
Gravity g (m/s^{-2})	9.81	9.81
Viscous friction b (kg/m/s)	(0.001, 0.001)	(0.001, 0.001)
Coulomb friction c_f (N·m)	(0.093, 0.077)	(0.093, 0.077)

4 min on both designs of the real double-pendulum hardware. We identified the parameters in Table 1 with a least-squares optimization, with a root-mean-square error (RMSE) of $\Delta\tau_{\text{RMSE}} \approx 0.2$ N·m in (2) over all data points for both models. The viscous and coulomb friction parameters of the motors were determined by separate measurements with the individual motors. We set $r_1 = l_1$ after identifying the parameters to list separate values for m_1 and r_1.

ROBUSTNESS CRITERIA

When transferring controllers from simulation to real hardware, many effects that are not present in simulation may influence the behavior. Often, it is the case that controllers are tuned in simulation and capable of high-quality performances, while in real system experiments, they fail to

achieve the desired results. This phenomenon is commonly referred to as the *simulation-reality gap*. In order to study the transferability of controllers, we conduct robustness tests in simulation. The robustness tests quantify how well the controllers perform under the following conditions:

1) *Model inaccuracies*: The model parameters that have been determined with system identification, as described in the "System Identification" section, will never be perfectly accurate. To assess inaccuracies in these parameters, we vary the independent model parameters from (3) one at the time in the simulator while using the original model parameters in the controller. The parameters $m_1 r_1$, $m_2 r_2$, m_2, I_1, and I_2 are varied between 75% and 125% of their identified values; viscous frictions are varied between –0.1 and 0.1 kg/m/s, the coulomb friction between –0.2 and 0.2 N·m, and the motor inertia between 0 and 10^{-4} kg/m^2.
2) *Measurement noise*: The controllers' outputs depend on the measured system state. In the case of the QDDs, the online velocity measurements are noisy. Hence, it is important for the transferability that a controller can handle at least this amount of noise in the measured data. For testing the robustness, Gaussian noise with standard deviations between 0 and 0.5 m/s is added to the velocity measurements. The controllers are tested with and without a low-pass noise filter.
3) *Torque noise*: Not only are the measurements noisy but the torque that the controller outputs is not always exactly the desired value. During this test, Gaussian noise with standard deviations in the range from 0 to 2 N·m is added to the applied motor torque.
4) *Torque response*: The requested torque of the controller will in general not be constant but change during the execution. The motor, however, is sometimes not able to react immediately to large torque changes and will instead overshoot or undershoot the desired value. This behavior is modeled by applying the torque $\tau = \tau_{t-1} + k_{\text{resp}}(\tau_{\text{des}} - \tau_{t-1})$ instead of the desired torque τ_{des}. Here, τ_{t-1} is the applied motor torque from the last time step, and k_{resp} is the factor that scales the responsiveness. For the tests, k_{resp} is varied between 0.1 and 2.
5) *Time delay*: When operating on a real system, there will always be time delays due to communication and reaction times. For the evaluation, the measurement results are artificially delayed for 0 to 0.04 s.

For all the above comparisons, the parameters are varied in $N = 21$ steps, and for each case, it is tested whether the controller is still able to perform a swing-up and reach the final state with an accuracy of $\varepsilon = (0.1, 0.1, 0.5, 0.5)$ in the four state dimensions. For the nondeterministic noise tests, 10 simulations were conducted for each parameter, and the controller had to have at least a 50% success rate to be considered successful. The ranges of the friction parameters b_1, b_2, c_{f1}, and c_{f2} extend to negative values because during real system experiments, we use friction compensation on both motors. A negative value tests the situation where the friction is overcompensated.

CONTROLLER SETUP

We tested the following controllers for the pendubot and acrobot: 1) TVLQR, 2) iLQR MPC (trajectory stabilization), 3) iLQR with Riccati gains, 4) iLQR MPC (free), and 5) collocated PFL.

For the trajectory stabilization methods, the nominal trajectory was computed with iLQR. The trajectory consists of $N = 6{,}000$ time steps, with a step size of $\delta t = 0.001$. As we intend to compensate the friction in the motors during the experiments, the friction coefficients were set to zero for the trajectory optimization. The cost parameters for the trajectory optimization as well as all parameters of the controllers can be found in the supplementary material. The parameters for the LQR and PFL controllers as well as the iLQR trajectory optimization for the acrobot have been obtained with a covariance matrix adaptation evolution strategy [18] parameter search, with the objective to reach the upright position as close as possible. The swing-up trajectory with iLQR computed for the acrobot is visualized in Figure 4. As the PFL controller is not able to stabilize the double pendulum, it is combined with an LQR controller, which takes over when the cost to go falls below a specified value of 15.

ROBUSTNESS RESULTS

We conducted robustness tests for all the controllers on both system designs for the acrobot and pendubot. All the results are listed in Table 2, where the listed numbers are the number of successful swing-up attempts out of 21 variations in each category. From the swing-up attempts in each category, we

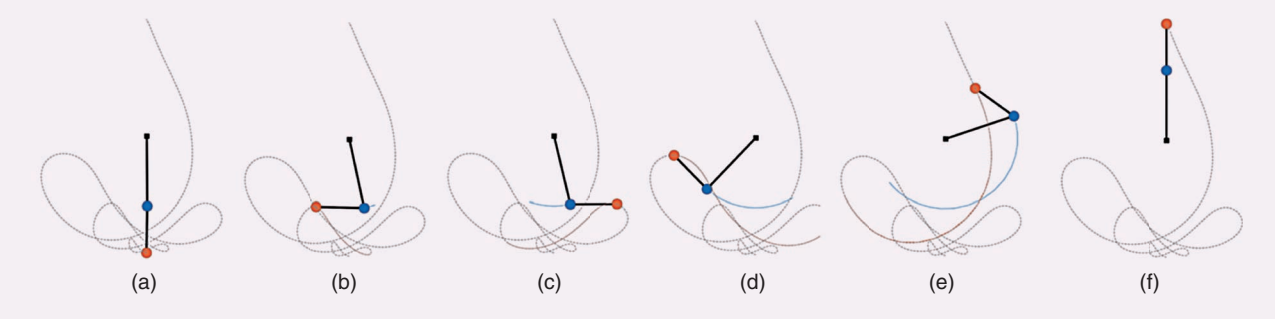


FIGURE 4. The swing-up trajectory for the acrobot: (a) $t = 0.02$ s, (b) $t = 1.76$ s, (c) $t = 2.54$ s, (d) $t = 3.24$ s, (e) $t = 3.92$ s, and (f) $t = 5.3$ s.

TABLE 2. The results of the robustness tests conducted in simulation.

	m_1r_1	m_2r_2	m_2	I_1	I_2	I_R	b_1	b_2	c_{f1}	c_{f2}	$\dot{q}_{\text{noise}}$	$\dot{q}_{\text{noise (FILTERED)}}$	τ_{noise}	τ_{resp}	DELAY	TOTAL
$\mathcal{M}_1$ acrobot																382
TVLQR	6	6	2	2	1	5	1	2	1	2	6	5	**18**	19	8	84
iLQR (stab)	**13**	**8**	**14**	**17**	**19**	**21**	**9**	**19**	**1**	**14**	**8**	**8**	7	**20**	**13**	**191**
iLQR (Riccati)	1	3	1	1	1	4	1	2	1	3	6	5	10	19	3	61
iLQR (free)	1	1	1	1	1	1	2	1	3	1	1	1	1	1	3	20
PFL	3	1	1	2	2	2	1	1	1	2	1	1	1	2	5	26
$\mathcal{M}_2$ acrobot																634
TVLQR	11	1	1	12	1	4	1	8	3	13	8	8	**21**	**21**	7	120
iLQR (stab)	**15**	**10**	**16**	**21**	16	**21**	**14**	**21**	**9**	**18**	12	11	14	19	**16**	**233**
iLQR (Riccati)	1	1	1	1	1	1	1	2	1	3	8	8	13	17	2	61
iLQR (free)	1	1	1	1	1	2	1	1	1	1	1	1	1	1	3	18
PFL	14	9	14	19	**17**	19	10	11	1	9	**16**	**15**	**21**	20	7	202
$\mathcal{M}_1$ pendubot																861
TVLQR	**21**	**18**	**18**	**21**	**21**	18	**21**	7	**21**	5	**21**	**21**	**21**	**21**	**17**	**272**
iLQR (stab)	**21**	15	16	**21**	17	14	**21**	7	21	6	8	8	15	20	15	225
iLQR (Riccati)	4	2	2	5	2	2	1	1	2	1	**21**	**21**	14	21	7	106
iLQR (free)	1	1	1	1	2	1	1	1	1	2	1	1	1	1	3	19
PFL	19	16	16	18	16	**20**	19	3	**21**	**11**	16	15	19	19	11	239
$\mathcal{M}_2$ pendubot																827
TVLQR	19	**18**	**17**	**21**	**17**	**17**	17	3	16	1	**21**	**21**	**21**	20	**15**	244
iLQR (stab)	**21**	15	16	**21**	**17**	16	**21**	**10**	**21**	**12**	4	4	5	15	11	209
iLQR (Riccati)	16	3	2	14	3	3	3	1	5	3	**21**	**21**	21	21	8	145
iLQR (free)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	17
PFL	18	12	12	**21**	14	15	14	2	15	8	13	14	21	19	14	212

Listed is the number of successful swing-ups out of 21 variations of the quantity in the header.

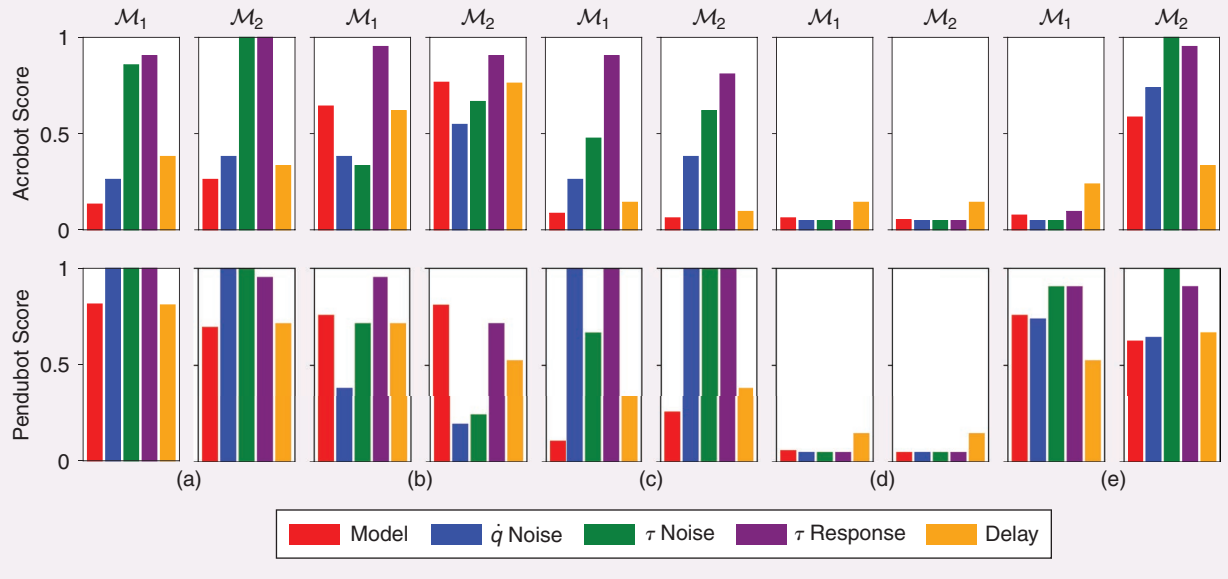


FIGURE 5. The success scores of the robustness tests for all the controllers applied to the acrobot and pendubot: (a) TVLQR, (b) iLQR (stab), (c) iLQR (Riccati), (d) iLQR (free), and (e) PFL.

computed a success score, which is the percentage of successful swing-up motions of all tested error variations. Figure 5 describes the success scores for both the acrobot and pendubot results and for both models in histograms.

As an example, the results for the robustness of the iLQR MPC (trajectory stabilization) controller for the acrobot swing-up with model $\mathcal{M}_1$ are presented in Figure 6. In addition to the Boolean success criterion, the figure shows the relative cost increase in the controller's cost function. The first two rows show modeling errors, and as expected, the cost increases when the controller acts on model parameters that do not match the parameters used for the simulation. It can be observed that the iLQR (stab) controller can deal with changes in the motor inertia I_r, the inertias I_1 and I_2, and the viscous friction b_2 in the second joint. The robustness to changes in $m_1 r_1, m_2 r_2, m_2, b_1$, and c_{f2} is a little worse. Most critical are changes in c_{f1}, where small deviations from the correct value prevent a successful swing-up. This is not surprising, as the acrobot cannot directly compensate for the friction in the first joint, and that friction makes dynamic motions at low velocities more difficult. The same controller on the pendubot can deal well with changes in c_{f1} but not so well with changes in c_{f2} (all 21 variations of c_{f1} were successful; see Table 2).

When comparing the performance of all the controllers, it can be observed that the iLQR (stab) controller achieved the most successful swing-ups on the acrobot configurations ($\mathcal{M}_1$: 191, $\mathcal{M}_2$: 233 out of 315 total attempts), while on the pendubot configurations, the TVLQR controller was most successful

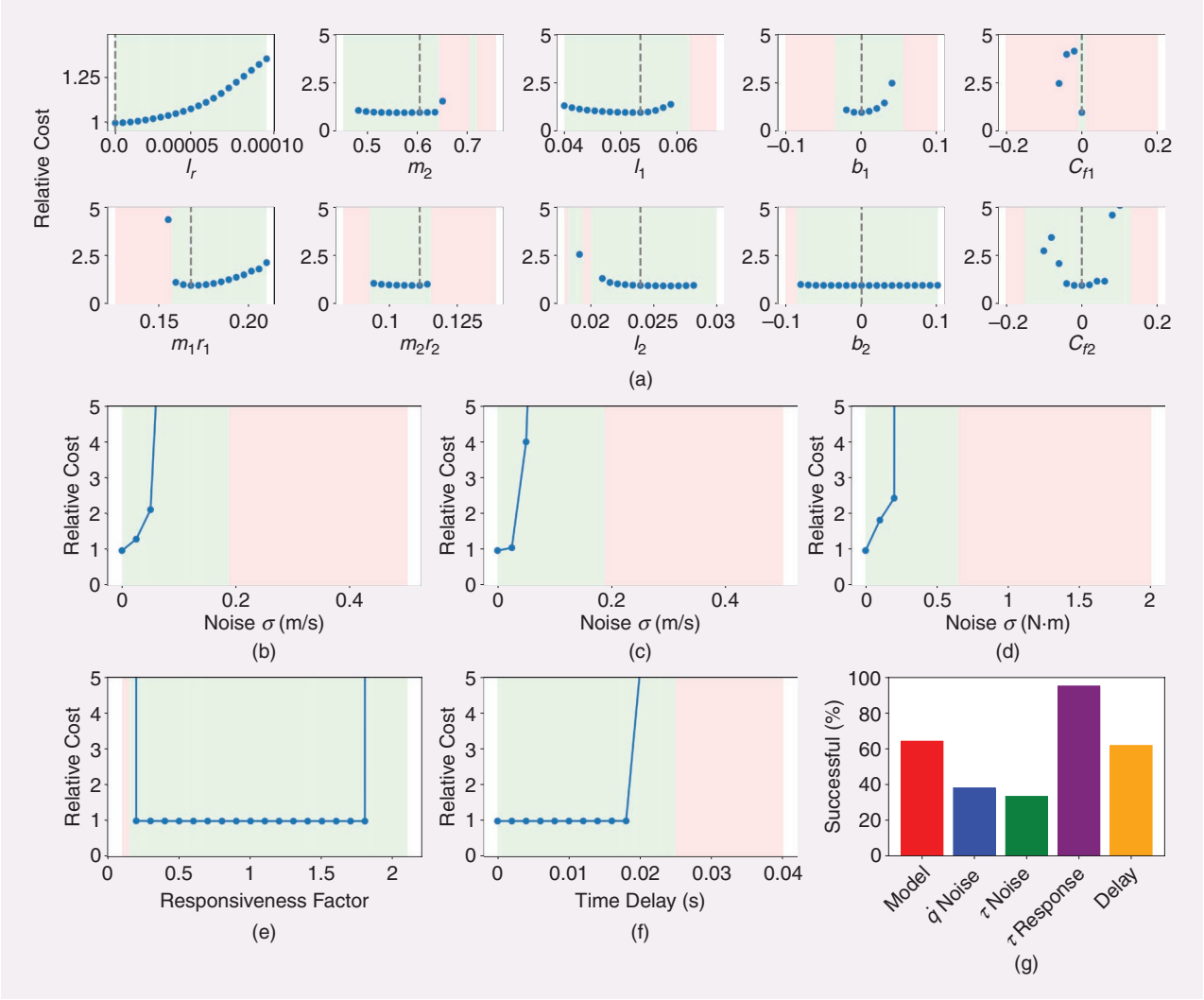


FIGURE 6. The robustness to model errors, noise, responsiveness, and delay for the iLQR MPC (trajectory stabilization) controller on the acrobot with model $\mathcal{M}_1$ and the trajectory from Figure 4. The dashed gray lines in the model parameter variations (the top two lines) indicate the parameters that the controller uses in its internal model. The model parameters are varied on the *x*-axis, and the *y*-axis shows the increase in cost relative to the cost with unperturbed model parameters. A green background means that the swing-up was successful, while a red background means the swing-up failed. Similarly, the noise responsiveness and delay plots show the cost increase and success for the varying noise, responsiveness factor, and delay time. The bottom right is a histogram of the percentages of successful swing-up simulations for each category. The (a) model parameter variations, (b) $\dot{q}$ noise, (c) $\dot{q}$ noise (low pass), (d) σ noise, (e) σ response, (f) delay, and (g) robustness criteria.

($\mathcal{M}_1$: 272, $\mathcal{M}_2$: 244). iLQR (Riccati) receives decent results on the pendubot configurations ($\mathcal{M}_1$: 106, $\mathcal{M}_2$: 145), with the good robustness to noise and responsiveness but less robustness to modeling errors and delay. On the acrobot configuration, iLQR (Riccati) scores only 61 successful swing-ups in both configurations. With at most 20 successful swing-ups across all categories in each configuration, iLQR (free) showed to be very sensitive to all kinds of errors, perturbations, or noise. The PFL controller showed little robustness on the $\mathcal{M}_1$ acrobot configuration but performed well in all other configurations (Acrobot—$\mathcal{M}_1$: 26, $\mathcal{M}_2$: 202; Pendubot—$\mathcal{M}_1$: 239, $\mathcal{M}_2$: 212).

The robustness results can also be used to quantify the difficulty of the swing-up task on the different configurations. Summing up all the scores on the acrobot configurations yields 382 successful swing-ups on model $\mathcal{M}_1$ and 634 on model $\mathcal{M}_2$, clearly indicating that the swing-up task on the acrobot with the longer second link ($\mathcal{M}_2$) is easier than when the link lengths are reversed ($\mathcal{M}_1$). This is consistent with the ROA analysis that was used during the system design in the "System Design" section. In the case of the pendubot, there are 861 successful swing-ups on the $\mathcal{M}_1$ model and 827 on the $\mathcal{M}_2$ model. This confirms that for the pendubot, the $\mathcal{M}_1$ model poses the easier task and that the difference in difficulty is less significant, as was the case when comparing the ROAs of the pendubot LQR controller.

EXPERIMENTS

SETUP

The QDDs from T-Motor pose some challenges to the control algorithms, due to the presence of noise in velocity measurements with standard deviations of $\sigma_{\dot{q}} = 0.05 \frac{\text{rad}}{\text{s}}$ and torque noise with $\sigma_\tau = 0.05$ N·m. The torque response factor k_{resp} ranges between 0.4 and 0.9. The operating delay lies between 0.005 and 0.01 s. To achieve more dynamical motions, we used friction compensation in both motors for the acrobot and in the passive motor for the pendubot.

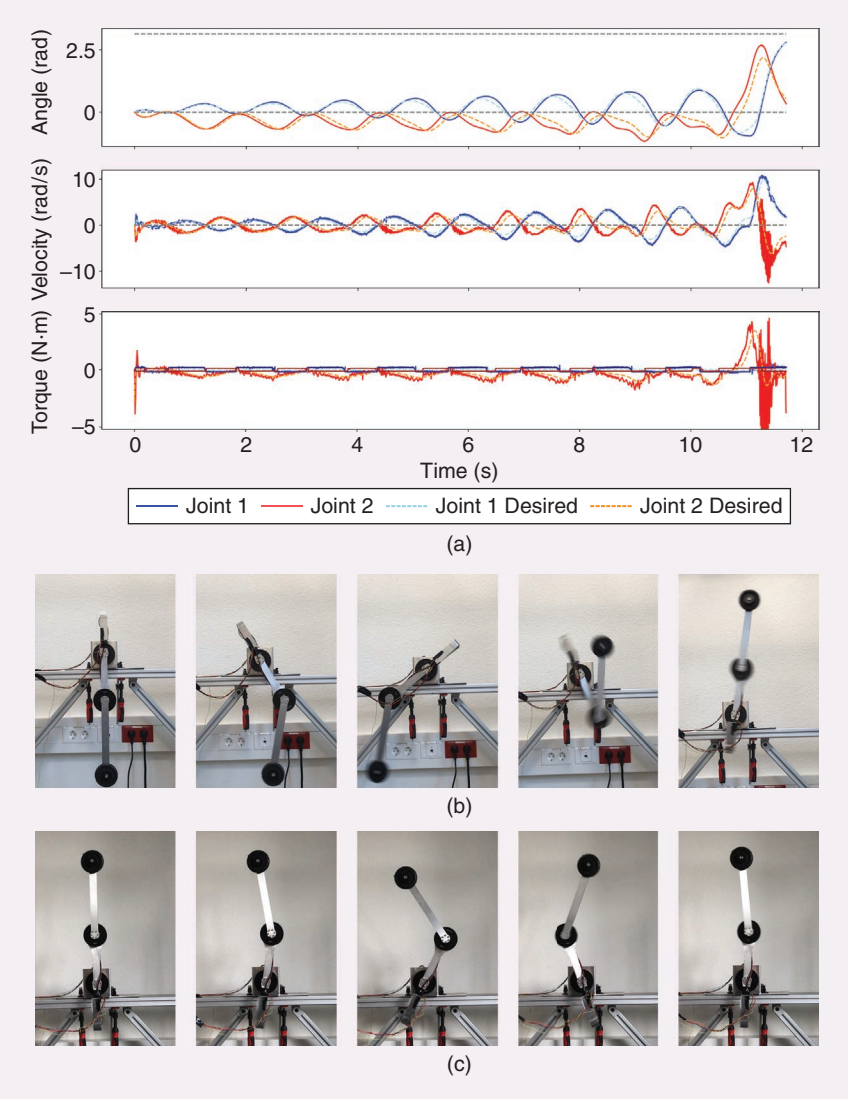


FIGURE 7. The acrobot swing-up and stabilization on the real system with TVLQR and LQR (separate experiments). (a) Recorded data of the acrobot swing-up with TVLQR. (b) The acrobot swing-up with TVLQR. (c) The acrobot stabilization with LQR.

ACROBOT RESULTS

For the acrobot swing-up, we tested the TVLQR controller to stabilize the iLQR trajectory on model $\mathcal{M}_2$. The results are displayed in Figure 7(a), where the colored dashed lines show the nominal trajectory that the TVLQR controller is supposed to stabilize and the solid lines show the actual measured positions, velocities, and torques. The controller tracks the trajectory well except for the final part, which is the most challenging, as the high velocities in this phase cause vibrations in the system. Due to the imperfect final phase, we tested the stabilization with LQR in a separate experiment. The LQR controller is able to stabilize the acrobot, and the controller is able to recover from relatively large state deviations, as pictured in Figure 7(c).

PENDUBOT RESULTS

Based on the robustness tests in the "Controller Comparison" section, the most robust controller for the pendubot with model $\mathcal{M}_1$ is the TVLQR controller. For the experiment, we combined the TVLQR controller with an LQR controller for the stabilization after the swing-up. The TVLQR controller is indeed able to perform a

successful swing-up on the real system, and the LQR controller is able to stabilize the unstable fixed point afterward. The data recorded during the experiment can be seen in Figure 8(a). At $t \approx 4.5$ s, the control switches from the TVLQR trajectory stabilization to the LQR fixed-point stabilization. The pendulum follows the position and velocity of the desired trajectory closely. The torque that is necessary for the tracking deviates noticeably from the nominal torque, especially at the peaks and during the final phase of the swing-up. During the LQR stabilization phase, the control output switches between minimum and maximum torque with a high frequency. Attempts to tune the LQR parameters by hand to avoid this behavior were not successful. However, the LQR stabilization is stable and can also be perturbed with a stick. The pendulum drops only once the controller is switched off.

> “THE ROBUSTNESS CRITERIA QUANTIFY THE SENSITIVITY OF THE CONTROLLERS AND ALLOW FOR A COMPARISON OF THE DIFFICULTY OF THE SWING-UP TASK ON THE ACROBOT AND PENDUBOT PLATFORMS.”

CONCLUSION

We introduced a canonical hardware platform that allows comparison of the performance of different control algorithms. The double pendulum can be operated either as a pendubot or as an acrobot without changes in the hardware. The double-pendulum design can be changed with different link lengths and a different attached mass at the tip, which creates systems with different difficulty for the controller. In this article, we evaluated the performance of LQR, TVLQR, and PFL controllers and three versions based on iLQR. We tested their robustness to model inaccuracies, noise, motor response, and delay and demonstrated a successful pendubot swing-up with TVLQR on real hardware. The necessary hardware components are inexpensive, and all software, from the drivers to the operating software and controllers, is open source. The double pendulum is integrated in RealAIGym [19] along with other canonical systems. The transparency of this project has two major advantages for the robotics research community. First, it enables reproducibility of experimental results, which is of major importance for sustainable scientific research. Second, the availability and openness allow students and newcomers in the field to study dynamic control at any level, without boundaries set by high costs, licenses, or closed software.

REPRODUCIBILITY

The entire platform is designed to be replicated for reproducing and improving on existing results. Two key aspects are that the hardware is inexpensive and that all the software is openly available (https://github.com/dfki-ric-underactuated-lab/double_pendulum), including all scripts and data that were used to obtain the results in this article. Additionally, the software is archived at Zenodo (https://zenodo.org/record/7529896), and the simulations from this article are uploaded to Code Ocean (https://codeocean.com/capsule/2798174/tree) to be run online without the need for a local installation. The GitHub repository also contains all necessary files for rebuilding the double-pendulum test bench, such as the mechanical design, CAD models, bill of materials, and so on. Instructions for the assembly of the hardware setup, with step-by-step pictures, can be found in the same repository as well as in the supplementary material for this article. The supplementary

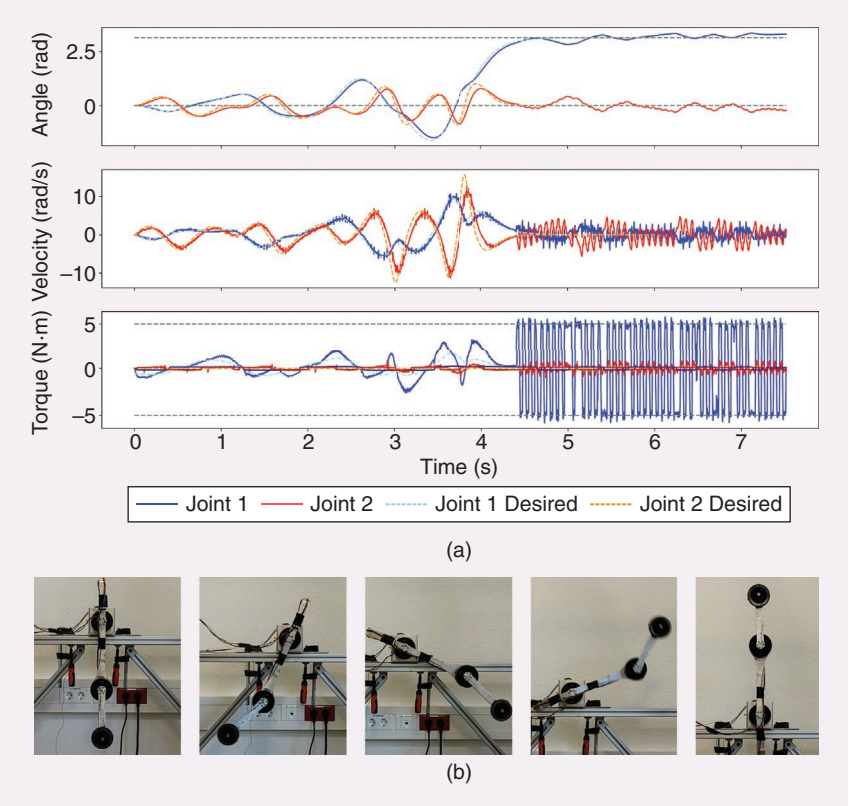


FIGURE 8. The pendubot swing-up on the real system. The switch from TVLQR to LQR stabilization happens at $t \approx 4.5$ s. (a) Recorded data of the pendubot swing-up (TVLQR) and stabilization (LQR). (b) The pendubot swing-up.

material also contains more detailed descriptions of the controllers, including pseudocode, more details for operating the hardware, and more benchmark results.

This work complies with the good experimental methodology (GEM) guidelines [20]. This article introduced an open source and low-cost test bench and contains experiments conducted in simulation and on real hardware (Q1). We laid out the assumptions and research questions in the "Methodology" and "Controller Comparison" sections (Q2). In the "Controller Comparison" section, we also explained the evaluation criteria (Q3) and how they were measured (Q4). The robustness criteria quantify the sensitivity of the controllers and allow for a comparison of the difficulty of the swing-up task on the acrobot and pendubot platforms (Q5). We published all relevant information and data for reproducing our work (Q6); thoroughly reported methods, parameters, and results to give a realistic picture of our results (Q7); and drew conclusions in the "Conclusion" section (Q8). A summarizing table of the GEM guidelines and the compliance of this article can be found in the supplementary material.

ACKNOWLEDGMENT

This work has been performed as part of the M-RoCK project, funded by the German Aerospace Center, with federal funds (Grant FKZ 01IW21002) from the Federal Ministry of Education and Research, and is additionally supported with project funds from the federal state of Bremen for setting up the Underactuated Robotics Lab (Grant 201-342-04-2/2021-4-1). Shubham Vyas acknowledges support from the Stardust Reloaded project, which has received funding from the European Union's Horizon 2020 research and innovation program under Marie Skłodowska-Curie Grant 813644. This article has supplementary downloadable material available at https://www.doi.org/10.1109/MRA.2023.3341257, provided by the authors.

AUTHORS

Felix Wiebe, Robotics Innovation Center, German Research Center for Artificial Intelligence, D-28359 Bremen, Germany. E-mail: felix.wiebe@dfki.de.

Shivesh Kumar, Robotics Innovation Center, German Research Center for Artificial Intelligence, Bremen, Germany, and Division of Dynamics, Department of Mechanics and Maritime Sciences, Chalmers University of Technology, 412 96 Gothenburg, Sweden. E-mail: shivesh.kumar@dfki.de.

Lasse J. Shala, Robotics Innovation Center, German Research Center for Artificial Intelligence, D-28359 Bremen, Germany. E-mail: lasse.jenn.may@gmail.com.

Shubham Vyas, Robotics Innovation Center, German Research Center for Artificial Intelligence, D-28359 Bremen, Germany. E-mail: shubham.vyas@dfki.de.

Mahdi Javadi, Robotics Innovation Center, German Research Center for Artificial Intelligence, D-28359 Bremen, Germany. E-mail: mahdi.javadi@dfki.de.

Frank Kirchner, Robotics Innovation Center, German Research Center for Artificial Intelligence and AG Robotics, Department of Mathematics and Computer Science, University of Bremen, D-28359 Bremen, Germany. E-mail: frank.kirchner@dfki.de.

REFERENCES

[1] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 6295–6301, doi: 10.1109/ICRA.2019.8793865.

[2] G. Brockman et al., "OpenAi Gym," 2016, *arXiv:1606.01540.*

[3] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[4] J. Hauser and R. M. Murray, "Nonlinear controllers for non-integrable systems: The Acrobot example," in *Proc. Amer. Control Conf.*, 1990, pp. 669–671, doi: 10.23919/ACC.1990.4790817.

[5] S. Bortoff, "Advanced nonlinear robotic control using digital signal processing," *IEEE Trans. Ind. Electron.*, vol. 41, no. 1, pp. 32–39, Feb. 1994, doi: 10.1109/41.281605.

[6] M. W. Spong, "The swing up control problem for the Acrobot," *IEEE Control Syst.*, vol. 15, no. 1, pp. 49–55, Feb. 1995, doi: 10.1109/37.341864.

[7] N. A. Andersen, L. Skovgaard, and O. Ravn, "Control of an under actuated unstable nonlinear object," in *Experimental Robotics VII*. Berlin, Germany: Springer-Verlag, 2007, pp. 481–490.

[8] X. Xin and T. Yamasaki, "Energy-based swing-up control for a remotely driven Acrobot: Theoretical and experimental results," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 4, pp. 1048–1056, Jul. 2012, doi: 10.1109/TCST.2011.2159220.

[9] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, Piscataway, NJ, USA: IEEE Press, May 2013, pp. 4054–4061, doi: 10.1109/ICRA.2013.6631149.

[10] K. Van Heerden, Y. Fujimoto, and A. Kawamura, "A combination of particle swarm optimization and model predictive control on graphics hardware for real-time trajectory planning of the under-actuated nonlinear Acrobot," in *Proc. IEEE 13th Int. Workshop Adv. Motion Control (AMC)*, Piscataway, NJ, USA: IEEE Press, Mar. 2014, pp. 464–469, doi: 10.1109/AMC.2014.6823326.

[11] T. Horibe and N. Sakamoto, "Nonlinear optimal control for swing up and stabilization of the Acrobot via stable manifold approach: Theory and experiment," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 6, pp. 2374–2387, Nov. 2019, doi: 10.1109/TCST.2018.2865762.

[12] H. G. González-Hernández, J. Alvarez, and J. Alvarez-Gallegos, "Experimental analysis and control of a chaotic Pendubot," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 891–901, 2004, doi: 10.1177/0278364904044407.

[13] F. B. Mathis, R. Jafari, and R. Mukherjee, "Impulsive actuation in robot manipulators: Experimental verification of Pendubot swing-up," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 4, pp. 1469–1474, Aug. 2014, doi: 10.1109/TMECH.2013.2293474.

[14] R. Ng, Y. Tang, H. Lin, X. Chu, and K. W. S. Au, "Development of a portable Hybrid Pendubot-Acrobot Robotic platform for on and off-campus teaching and learning," in *Proc. Amer. Control Conf. (ACC)*, Piscataway, NJ, USA: IEEE Press, May 2021, pp. 98–105, doi: 10.23919/ACC50511.2021.9482939.

[15] L. Weiwei and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. Int. Conf. Inform. Control, Automat. Robot.*, 2004, pp. 222–229.

[16] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 7330–7336, doi: 10.1109/ICRA48506.2021.9560990.

[17] L. Maywald, F. Wiebe, S. Kumar, M. Javadi, and F. Kirchner, "Co-optimization of Acrobot design and controller for increased certifiable stability," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 2636–2641, doi: 10.1109/IROS47612.2022.9981825.

[18] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation. Studies in Fuzziness and Soft Computing*, vol. 192, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., Berlin, Germany: Springer-Verlag, 2006, pp. 75–102.

[19] F. Wiebe, S. Vyas, L. J. Maywald, S. Kumar, and F. Kirchner, "RealAIGym: Education and research platform for studying athletic intelligence," in *Proc. Robot. Sci. Syst. Workshop Mind Gap, Opportunities Challenges Transition Res. Ind.*, New York, NY, USA, Jul. 2022, pp. 7–9.

[20] F. Bonsignoriom, J. Hallam, and A. P. del Pobil. "Special interest group: Good experimental methodology." Heron Robots. Accessed: Nov. 4, 2023. [Online]. Available: http://www.heronrobots.com/EuronGEMSig/downloads/GemSigGuidelinesBeta.pdf