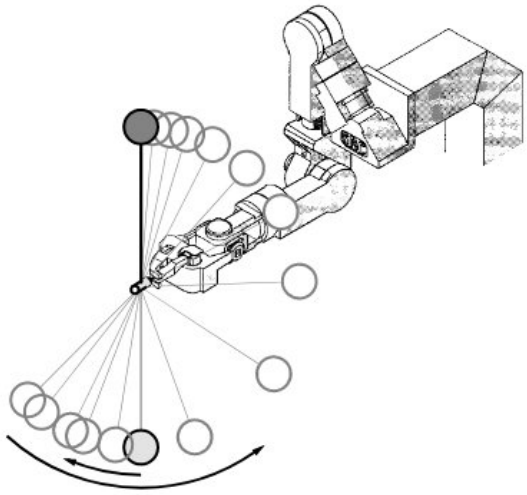
Robotics MVA 2025–26 Final exam topics

Learning pendulum swing ups from demonstrations

Context: In their 1997 work *Robot Learning from Demonstration* [1], Atkeson and Schaal learned both reward and model from human demonstrations. They showed pendulum swing ups performed by a real robotic arm. Jump forward 25 years, and some of our best manipulation systems [2] are based on behavior cloning [3].



(Figure adapted from [1].)

Goals: In this topic, we will reproduce [1] with concepts seen in class and modern robotics software. We would like to compare it to behavior cloning [3].

Project plan:

1. Read reference [1].
2. Adapt section 5, *Learning to balance*, using the Cart Pole environment of Gymnasium. What are the main steps to take?
    a. Note that the cart-pole environment will implement its own dynamics, replacing equation (3).
    b. You can find a ready-to-use Python implementation of *dlqr* in control.dlqr of the Python Control Systems library.
    c. Alternatively, you can teach yourself on the Linear-Quadratic regulator and implement your own `lqr` function using for instance SciPy's solve_continuous_are.
3. Pick a robot arm description and load it in PyBullet.
4. Apply *pybullet.TORQUE_CONTROL* with PD gains to make the robot go to a desired configuration.
    a. Note that in PyBullet one needs to disable velocity control `setJointMotorControl2(..., controlMode=p.VELOCITY_CONTROL, force=0)` before sending torque-control commands
    b. That is, we have a configuration vector q (of size model.nq) and a tangent velocity vector v (of size model.nv, for your tests you can start with v=0), and we want each joint i to reach an angle q[i] and angular velocity v[i]. To achieve this, PD control sends a joint torque computed as:
        i. *tau = Kp * (q[i] – q_measured[i]) + Kd * (v[i] – v_measured[i])*.
    c. You will have to find values of the two parameters *Kp* and *Kd* that work well for the robot arm model you have selected.

5. Use closed-loop inverse kinematics to command the robot's end effector position along a horizontal line of your choosing. You can implement your own, as in the inverse kinematics assignment, or use Pink.
6. Modify the URDF file of the robot description to add a pendulum at the end effector of the arm. The pendulum will consist in:
    a. A revolute joint that will be passive (i.e. always command it with zero torque).
    b. A mass at the end of the child link of the revolute joint.
7. Can you adapt your solution from step 2 to this more realistic simulation?
8. Propose a way to adapt section 6, *Learning the swing up task*, using this simulation and any input device connected to your computer.
    a. You can pick either a parametric or nonparametric model; no need to implement both.
    b. Your solution should be feasible on a real robot as well, using some of the systems you have seen in class or in the lab.
    c. Explain how you would perform the experiment on a real robot.
9. Implement and test it, report how many samples you need to realize a successful swing up. You can use CasADi for nonlinear trajectory optimization of pendulum trajectories.
10. Explain similarities and differences between this approach and behavior cloning [3].
11. *Bonus (optional) step:* Implement behavior cloning in your simulation. How many samples do you need to realize a successful swing up?

References:

[1] Atkeson, C. G., & Schaal, S. (1997, July). Robot learning from demonstration. In ICML (Vol. 97, pp. 12-20).

[2] Vanhoucke, V., (2023). Shifting Winds in Robot Learning Research.

[3] Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. arXiv preprint arXiv:1805.01954.