Robustness of model-based control

Emo Todorov

Roboti LLC
University of Washington

Model-based control already works on complex dynamical systems

model
predictive
control

nominal
physics model

Abbeel et al, *IJRR* 2010

Williams et al, *ICRA* 2016

offline
trajectory
optimization

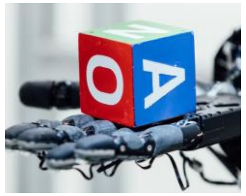
adaptive
local model

Kumar et al, *ICRA* 2016

Mordatch et al, *IROS* 2015

randomized
physics model

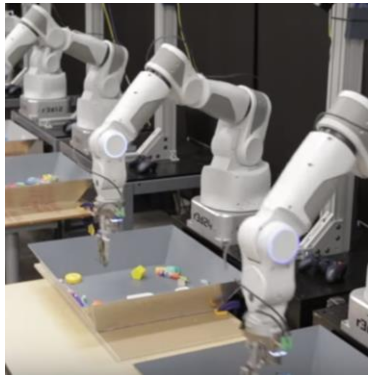
policy
gradient



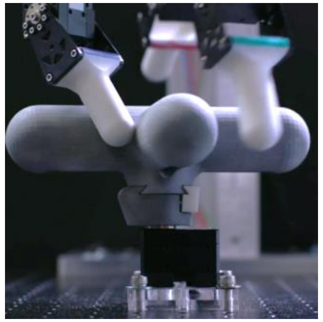
OpenAI, 2018

+ quadrupeds

Model-free RL sounds great, but …



Existing results are impressive mostly because of computer vision.
Works well in quasi-static tasks where sampling is safe/automated and suboptimal solutions are feasible.

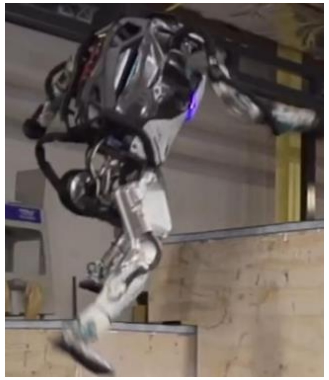


Mechanical contraptions enable safe/automated sampling, but they limit real-world applications …

… unless reality = publishing ☺

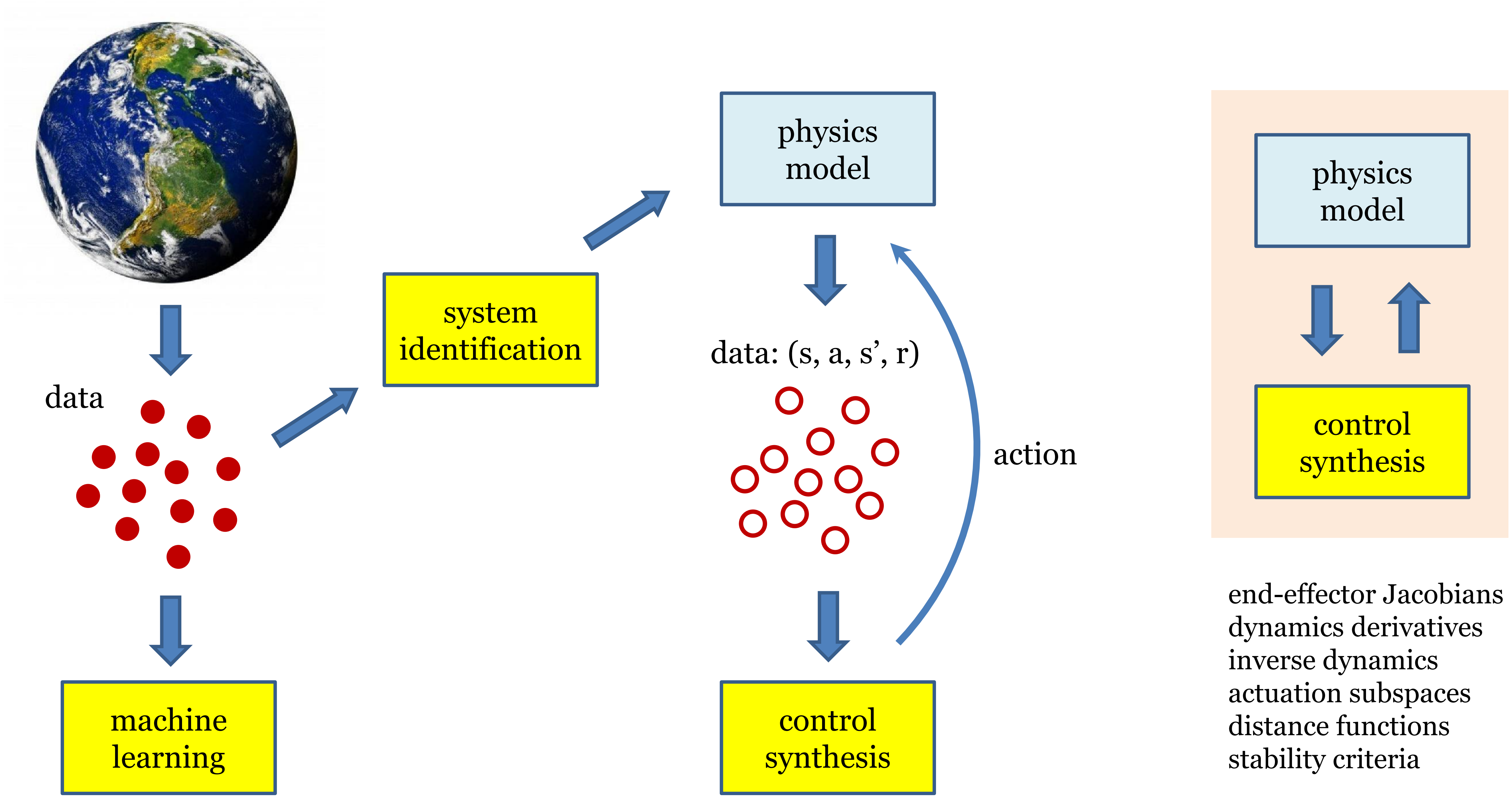
There are situations where control is easier than modeling, but that alone does not make model-free RL a good idea.

Alternative to learning/optimization: design a controller manually, then tune a small number of control parameters on the real system.



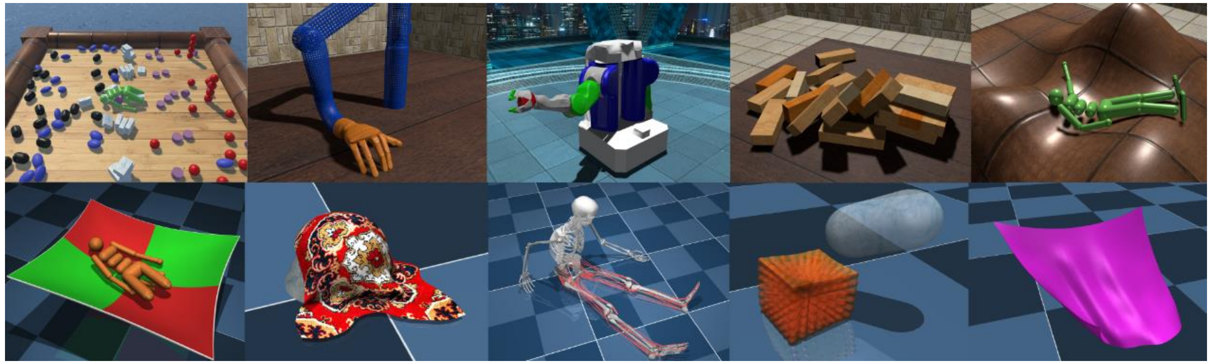
Expert manual design + parameter tuning can still outperform any form of learning.

Models can do more than sample data



MuJoCo (2009-2019)

q	configuration
v	velocity
τ	applied force
$c(q, v)$	internal force
$M(q)$	inertia matrix
$J(q)$	constraint Jacobian
$\lambda(q, v, \tau/\dot{v})$	constraint force



~ 10,000 active licenses

Forward dynamics: **numerical** solution (convex optimization)

$$\dot{v} = \arg\min_a \left\| a + M^{-1}(c - \tau) \right\|_M^2 + s(Ja - r)$$

Inverse dynamics: **analytical** solution

$$\tau = M\dot{v} + c + J^T \nabla s(J\dot{v} - r), \quad \lambda = -\nabla s$$

Now has analytical derivatives!

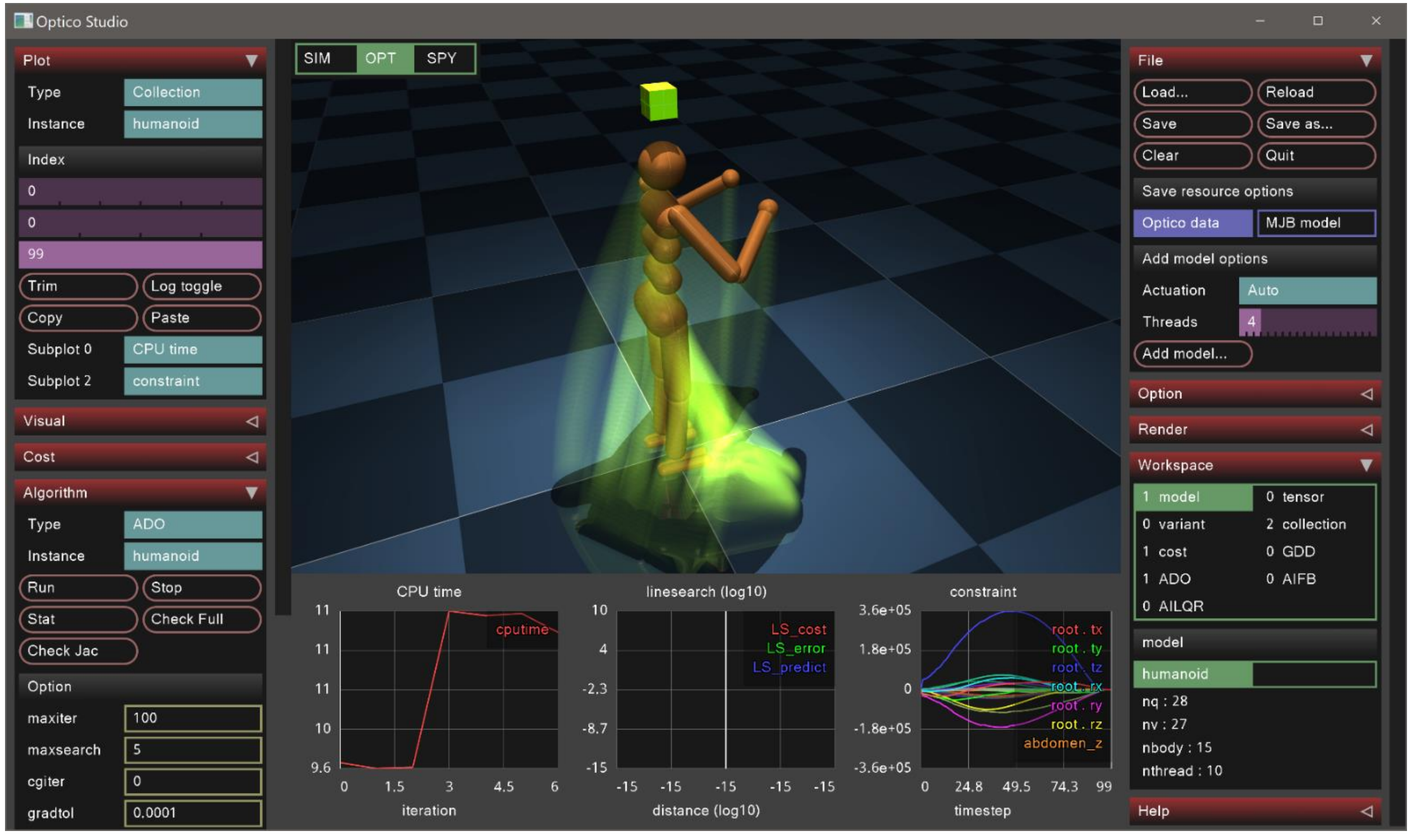
model	**evals / s 20 threads**
humanoid	300,800
humanoid100	17,100
hammock	40,400
particle	7,150
grid2	19,350
ellipsoid	31,600

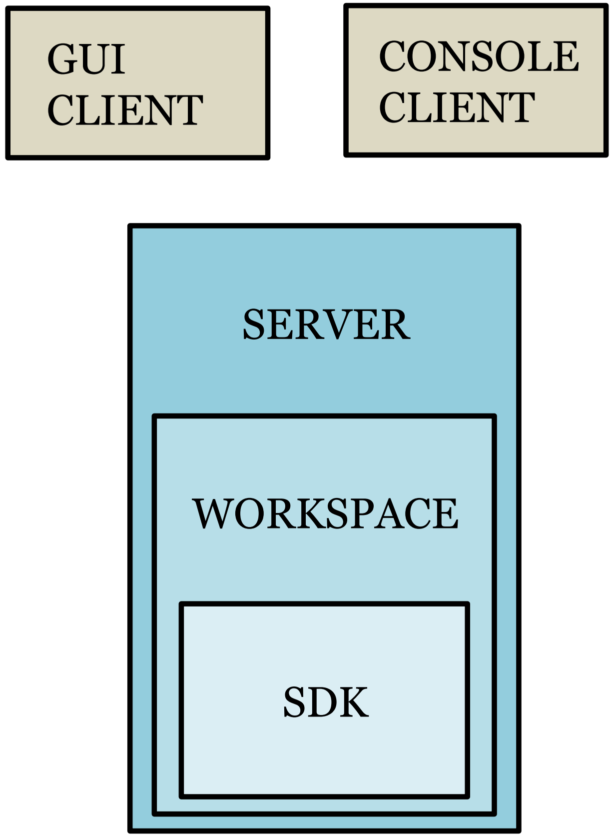
10-core processor

Optico (2016-2019)

Unified environment for physics modeling, cost function specification and model-based optimization: control, estimation, system id, mechanism design

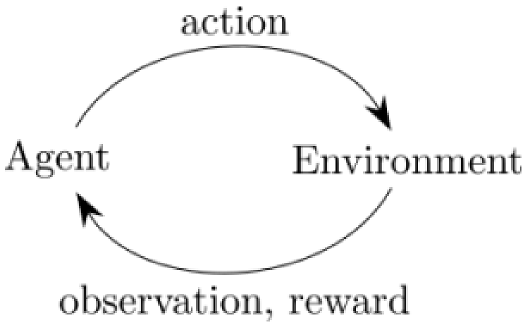
Speed goals: (on desktop)	ensemble MPC in real-time long trajectory optimization in seconds model/policy/value parameter learning in minutes





Deterministic dynamics and initial states

MDP/RL:
stochastic

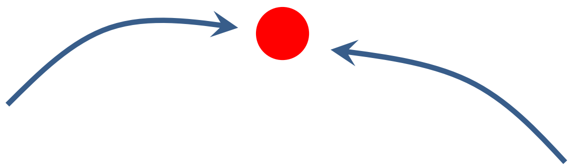


Control:
deterministic

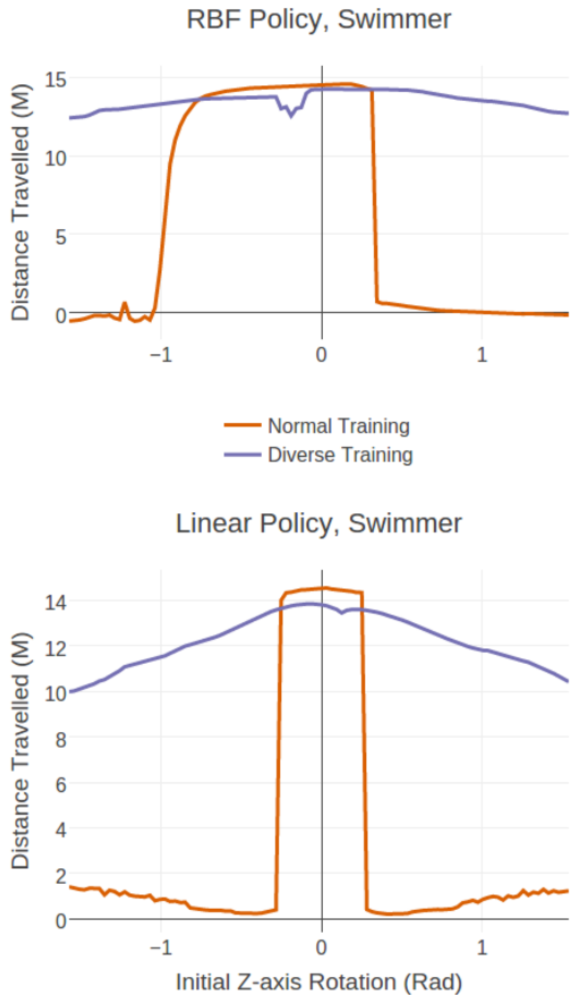
Dynamics	$\dot{x} = f(x, u)$
Cost	$\ell(x, u)$

In a deterministic system moving towards some goal, the initial state determines what other states are visited.

Different initial states may require different control strategies.



Training policies with diverse initial states avoids overfitting and increases robustness.



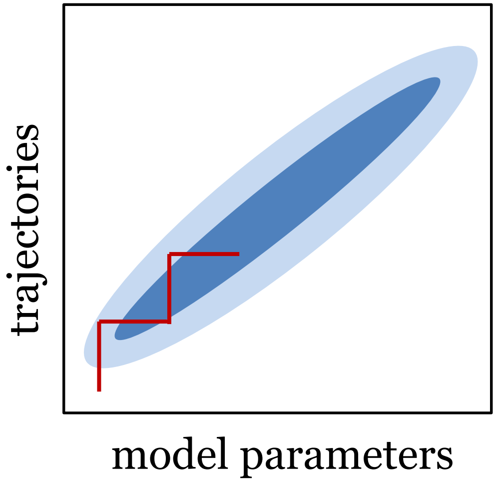
Rajeswaran et al, *NIPS* 2017

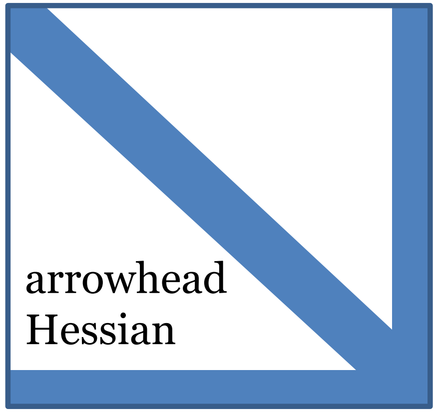
Physically-consistent state estimation and system identification

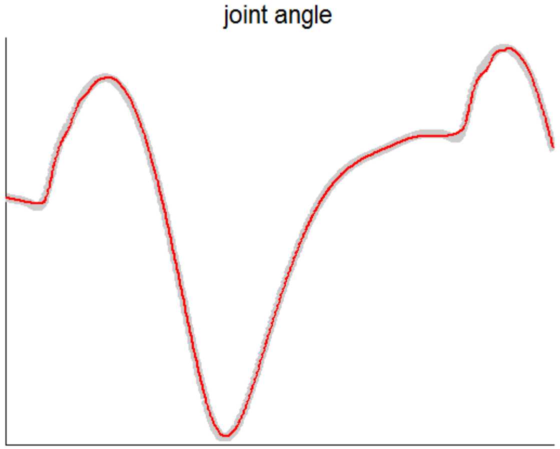
given noisy sensor data:
- movement kinematics
- contact forces
- actuator forces

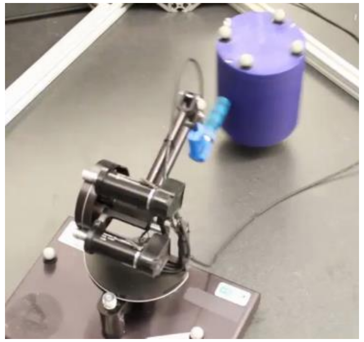
estimate **jointly**:
- kinematics
- forces
- model parameters

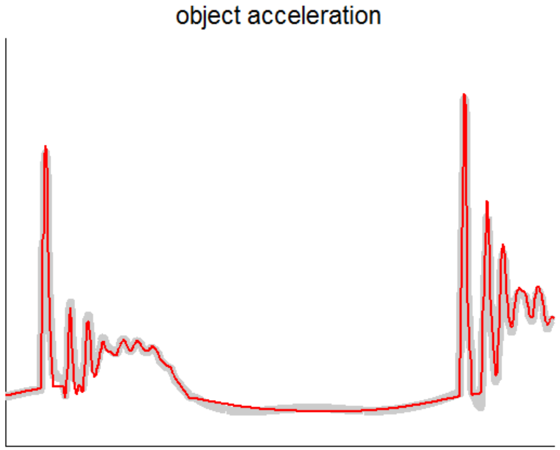
contacts introduce strong coupling between state estimation and system identification:





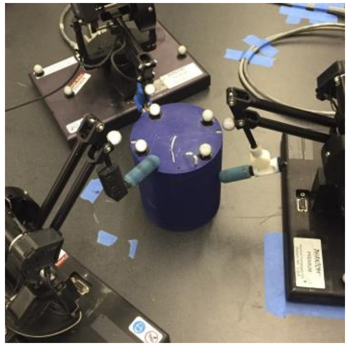






linear policy

2 min NPG training on 24 CPU cores



Kolev and Todorov, *Humanoids* 2015
Lowrey et al, *SIMPAR* 2018

Learning to act like a model

If we cannot make the model behave like the robot, make the robot behave like the model.

let the true (but hard to model) dynamics be $x' = f(x, u)$

specify reference model $x' = r(x, v)$ where v is some abstract control

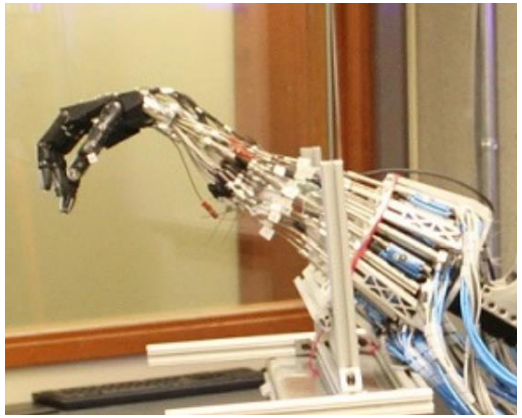
learn **feedback transformation** $u = g(x, v)$ such that $f(x, g(x, v)) = r(x, v)$

do model-based control with respect to $r(x, v)$

Examples: high-gain PID control (r : identity), feedback linearization (r : linear).

Specific motivation:

we built an amazing robot that we never controlled properly, even though it has very fast and strong actuation.



Sim-to-real transfer

Collect real data and do the best system identification possible.

Build a model-based controller (and a state estimator).

Test on the real system **as early as possible**. In many cases it will just work.

If it fails, options are:

- make controller less aggressive (gain reduction, larger control cost, smoothness)
- ensemble optimization / domain randomization / diverse initial states / min-max
- adaptive control: extend system id with data collected while running controller
- augment physics-based model with non-parametric models trained on residuals
- learn feedback transformation making the real system behave like the reference model

There are multiple good options for sim-to-real transfer, and they are relatively easy to try.

Building the model-based controller (and estimator) in the first place is the more difficult part.