Synthesis of contact-rich behaviors with optimal control

Emo Todorov

Applied Mathematics, Computer Science & Engineering
University of Washington

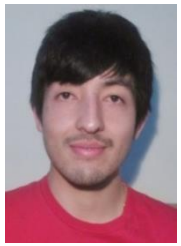
Contributions from:


Igor Mordatch


Vikash Kumar


Yuval Tassa


Tom Erez


Kendall Lowrey


Galen Andrew


Paul Kulchenko


Svet Kolev

Funding: NSF, NIH, DARPA

Unification

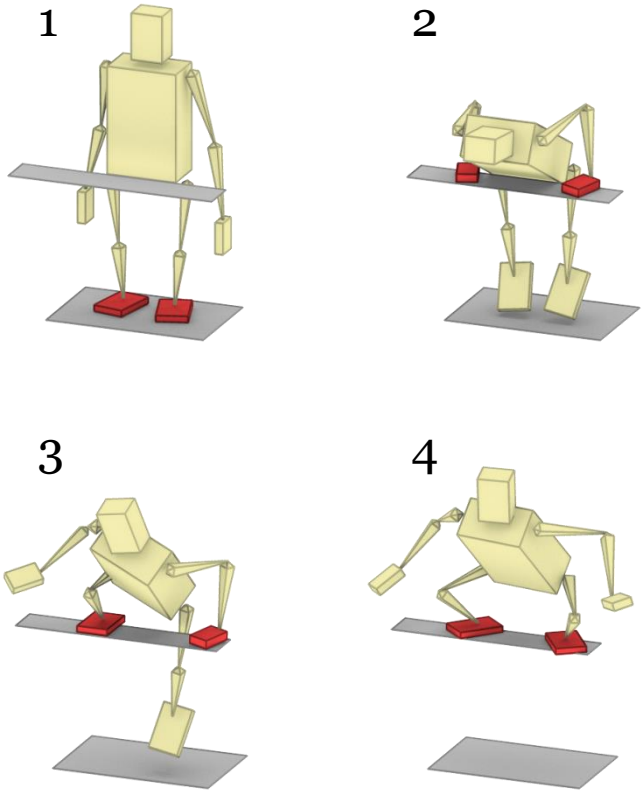
If you start with domain-specific methods (ZMP, SLIP, synergies, closures) unification is a problem.

If you start with optimal control, everything is unified by definition,
but you have to figure out how to make it feasible.

Contact-invariant optimization (CIO)

Contacts make the dynamics discontinuous, and may seem to require combinatorial search.

CIO is a domain-specific relaxation method designed to avoid such combinatorial search.



Configuration: q
Contact force: f
Applied force: $u = M(q)\ddot{q} + c(q,\dot{q}) - J(q)^T f$

Minimize over trajectories $q(t), f(t)$:

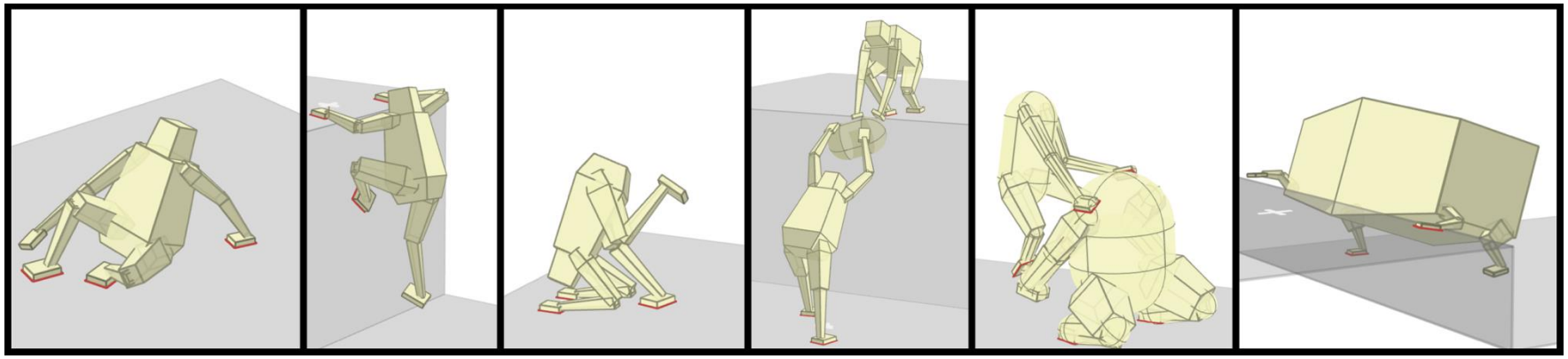
TaskCost(q, f) + ControlCost($u(q, f)$) +

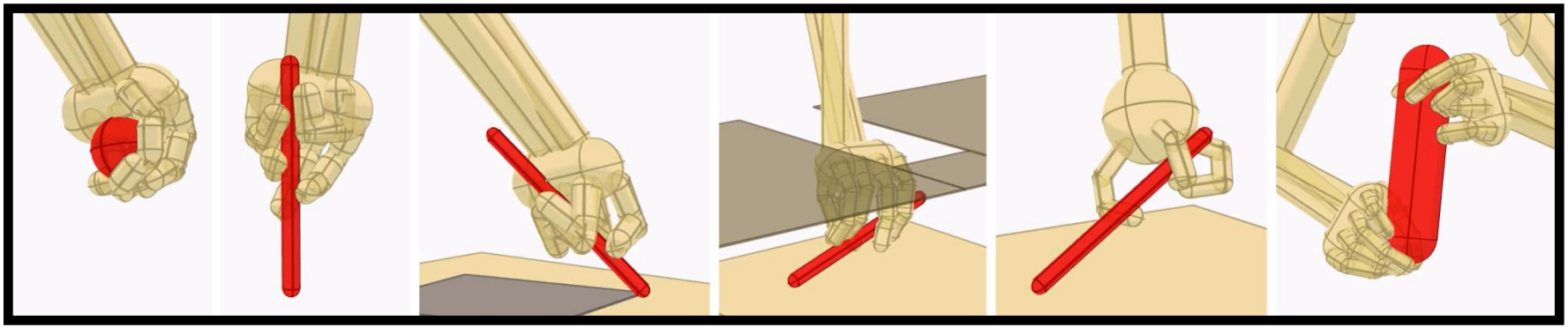
Penetration(q) + FrictionCone(f) +

f_{normal} * ContactDistance(q) * *continuation*

Mordatch, Todorov and Popovic, *SIGGRAPH* 2012
Mordatch, Popovic and Todorov, *SCA* 2012

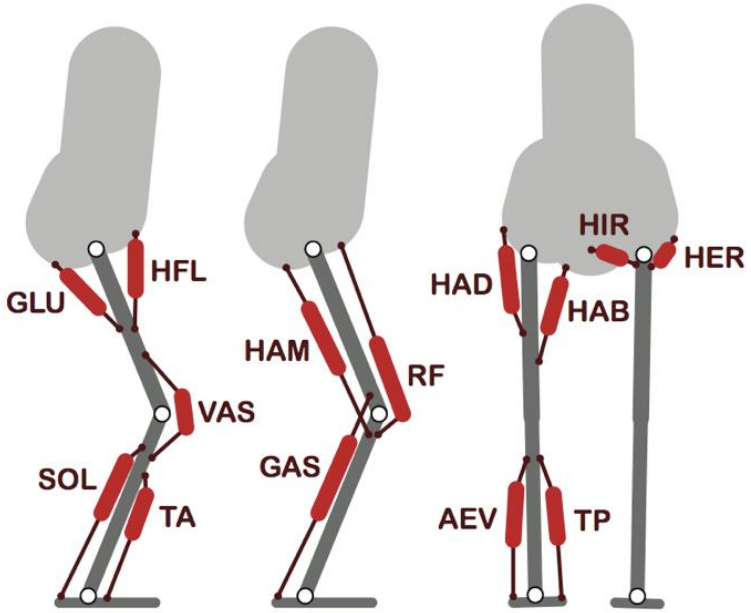
Results

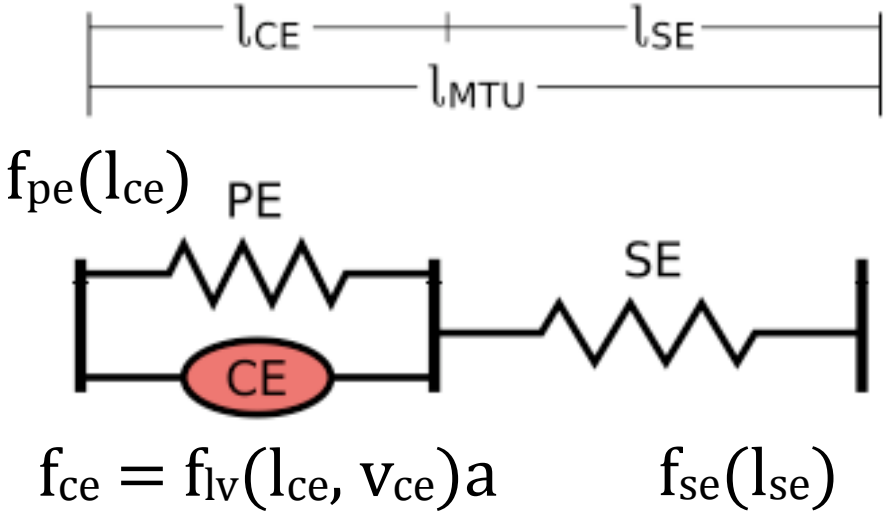




Extension to musculo-skeletal dynamics

- skeletal dynamics simulated in MuJoCo
- muscle model based on Wang et al 2012
- metabolic energy model based on Anderson and Pandy 1999
- modifications to the CIO method





Mordatch, Wang, Todorov and Koltun, *SIGGRAPH ASIA* 2013

Summary so far

Trajectory optimization can generate surprisingly rich and complex movements, without help from motion capture, manual scripting or careful initialization.

Generic optimization methods can be used (L-BFGS, Gauss-Newton), but contacts must be handled carefully so as to provide sufficient smoothing.

Optimizing contact-related variables (as in CIO) together with the trajectory produces the richest behaviors we have obtained.

We need ~1000 iterations to discover these trajectories from scratch.

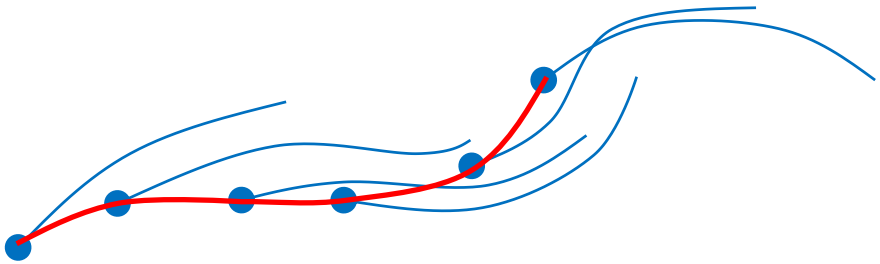
This takes ~5 minutes of CPU time.

How do we get to real-time?

Model-predictive control (MPC)

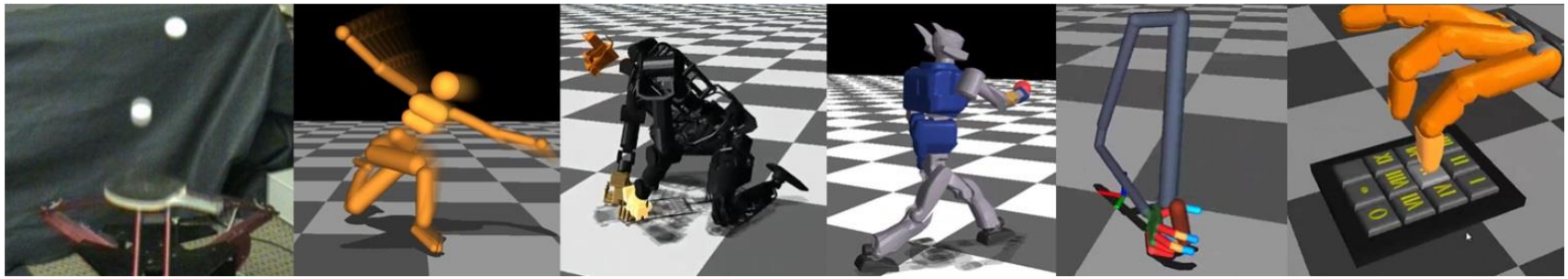
At each time step t, do trajectory optimization with warm-start from the previous time step:

$$\min h(x_{t+N}) + \sum_{k=t}^{t+N-1} \ell(x_k, u_k)$$



We have been able to apply MPC to the full robot dynamics thanks to:

- improved models of contact dynamics;
- efficient physics simulator (MuJoCo);
- efficient optimization algorithm (iLQG);
- selection of cost functions that are realistic yet easier to optimize.



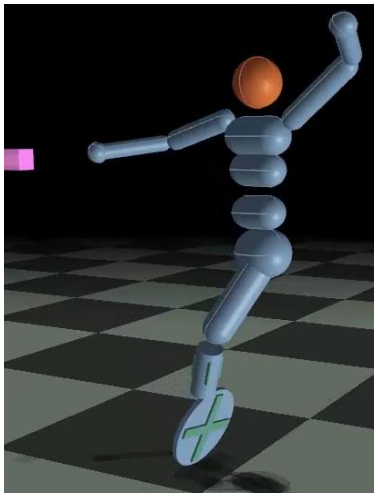
Tassa, Erez and Todorov, *IROS* 2012
Erez et al, *Humanoids* 2013
Tassa et al; Kumar et al; Erez et al; *ICRA* 2014

Cost function automation

$$\text{cost}(t) = \sum_i \text{coef}_i * \text{norm}_i\left(\text{feature}_i\left(\text{state}(t), \text{control}(t)\right) - \text{reference}_i(t)\right)$$

XML file format for specifying cost functions:

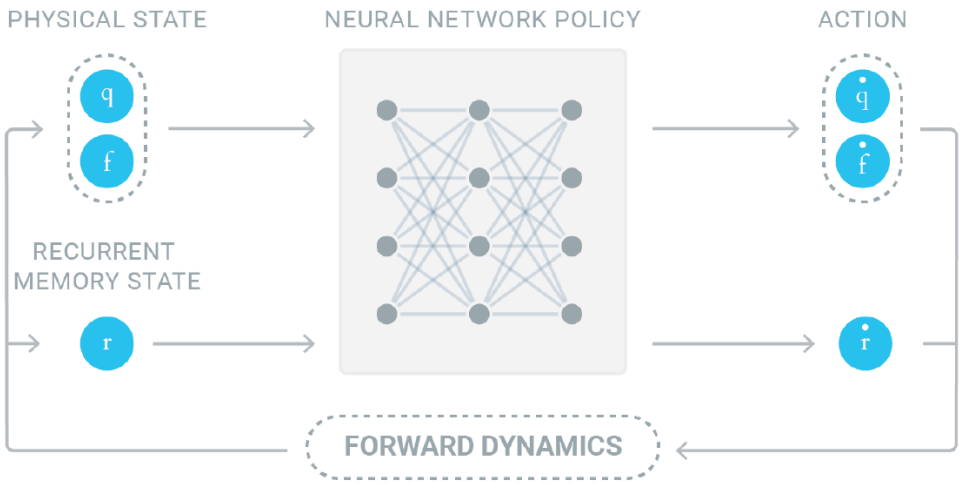
```
<feature name='torques'>
    <data field='ctrl' item='all'/>
</feature>

<feature name='height'>
    <op type='sum' ref='.7'>
        <data field='xipos' item='pelvis' entry='z' coef='1'/>
        <data field='xipos' item='wheel' entry='z' coef='-1'/>
    </op>
</feature>

<feature name='upright'>
    <data field='xmat' item='torso' entry='zz zx zy' ref='1 0 0'/>
</feature>

<cost>
    <term feature='torques'  norm='quadratic'   coef='.1'   />
    <term feature='balance'  norm='L2'          coef='.03'  />
    <term feature='height'   norm='smooth_abs2' coef='1'    />
    <term feature='upright'  norm='quadratic'   coef='1'    />
    <term feature='reach'    norm='L2'          coef='.07'  />
    <term feature='com vel'  norm='power'       coef='.0002'/>
    <term feature='ang mom'  norm='quadratic'   coef='.0001'/>
</cost>
```



torques	0.1
balance	0.03
height	1
upright	1
reach	0.07
com vel	0.0002
ang mom	0.0001

Tassa, Erez and Todorov, 2014, unpublished

Training neural networks with trajectory optimization



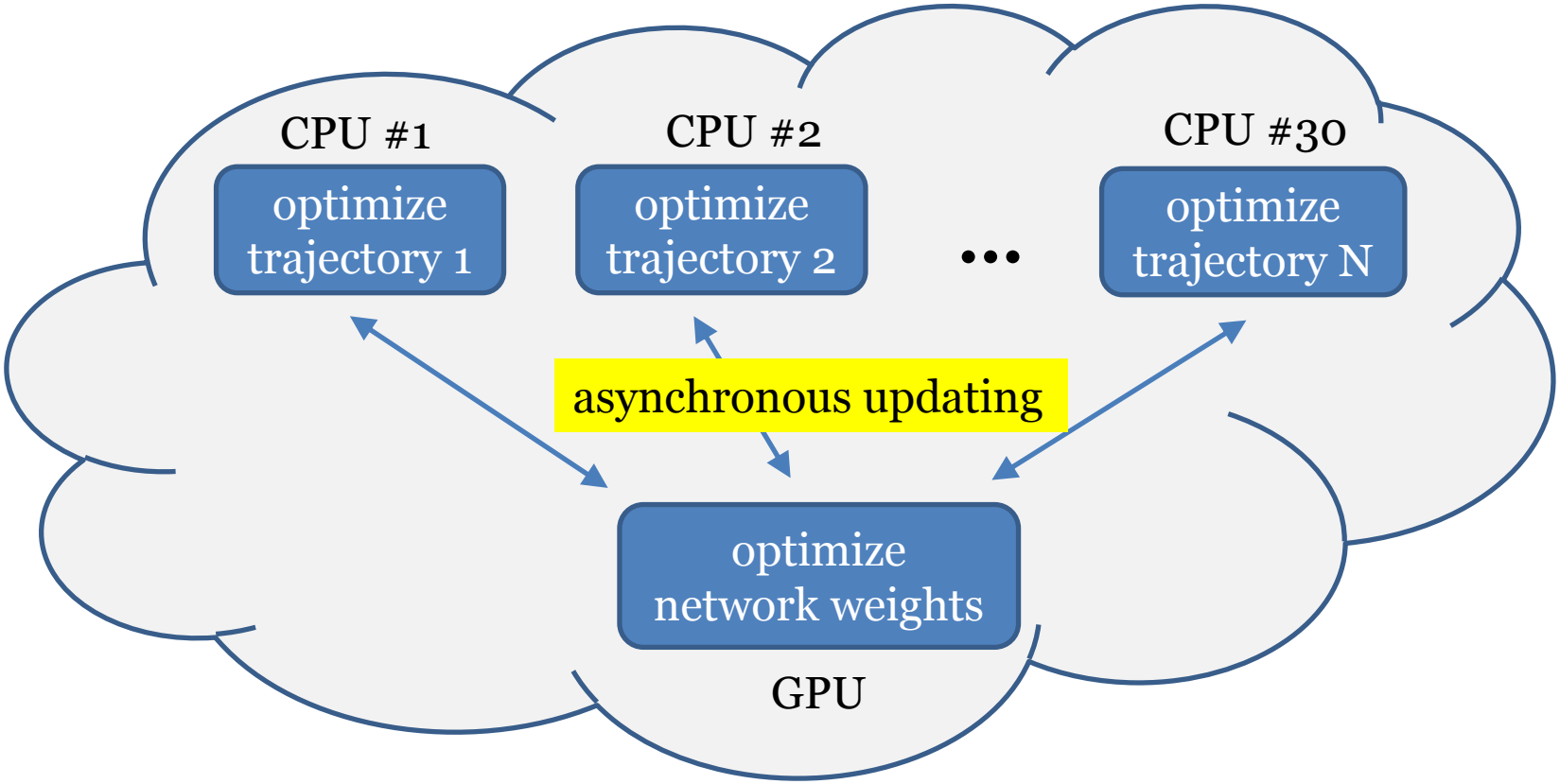
Trajectory optimization:
optimize the sequence of **body and brain** states, under a mixed cost that includes movement costs and differences from the network output (so as to keep the trajectories and the network close throughout training)

Network training:
supervised learning from the optimized trajectories

The "action" is the change in state. Thus the network is not learning a control law, but rather the dynamics of the system under an (implicit) control law.

Mordatch, Andrew, Lowrey, Popovic and Todorov, 2015, under review

Training in the cloud

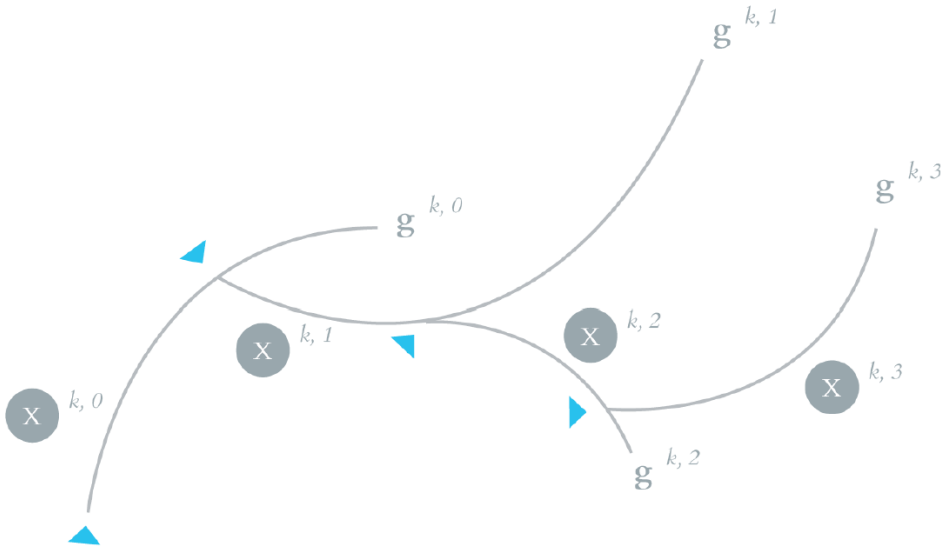


Training a network with 5 layers of 250 units (250,000 weights) takes 2.5 hours, 200 steps of (re) optimization for 1,000 trajectory pieces.

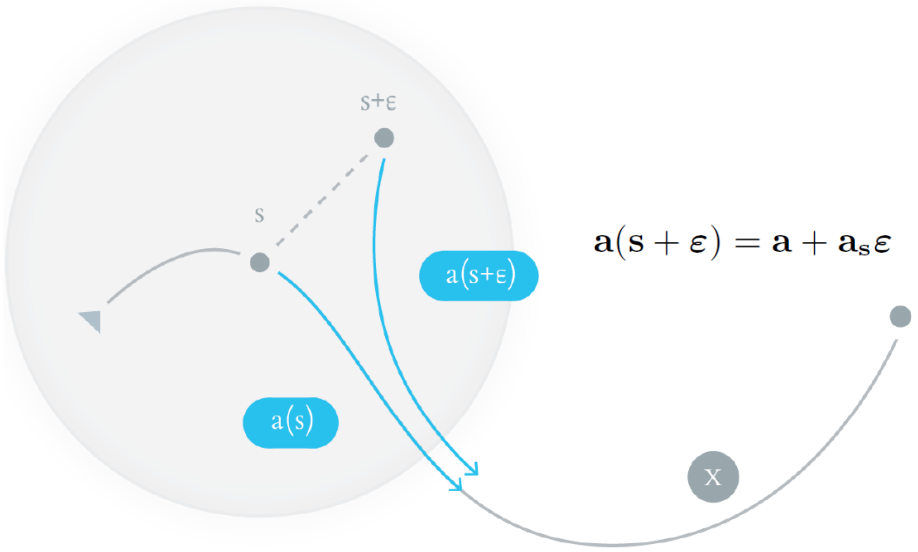
Each CPU has 16 cores. The GPU has 1536 cores.

Making the training set diverse

The movement cost (i.e. the spatial goal) change often, so as to generate a tree of trajectories that span a large portion of the relevant behavioral space.



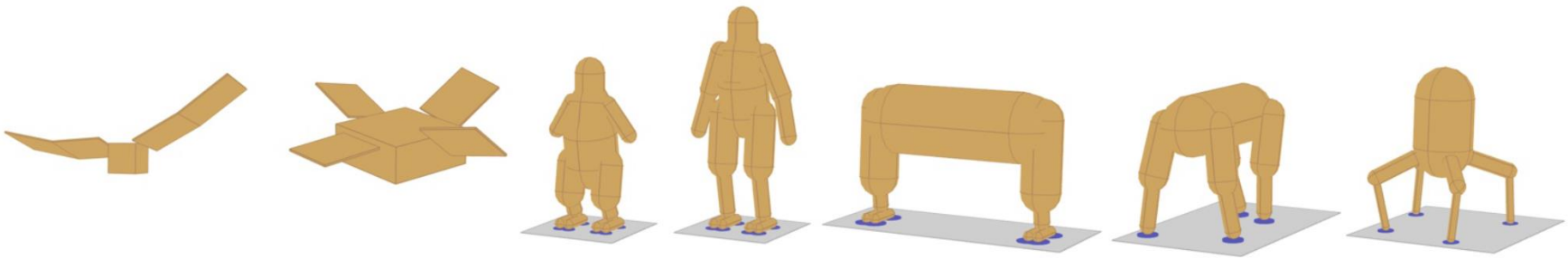
Noise is injected during training, forcing the network to learn not only the nominal response but also the corrections around it. This resembles Tangent Propagation.



Results

The same training method works for a variety of motor tasks and body morphologies:

- flying
- swimming
- bipedal walking
- quadrupedal walking



Separate vs. joint optimization

Can we create a fixed dataset of optimized trajectories, and then train the network?

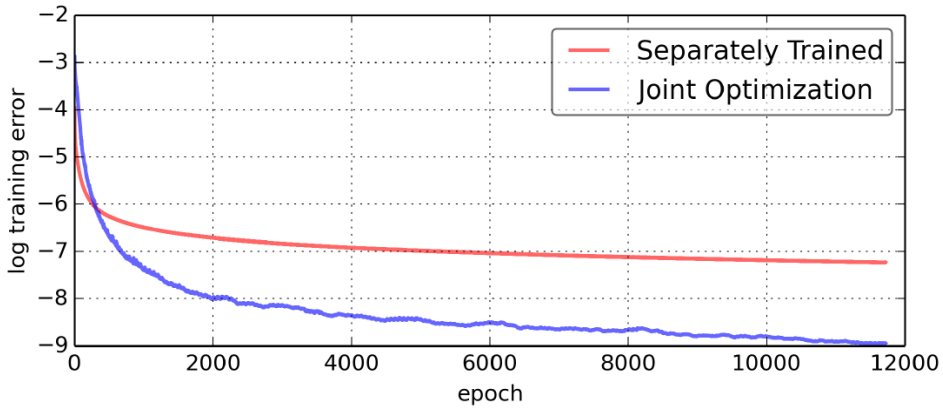
Or alternatively, use imitation learning from motion capture data?

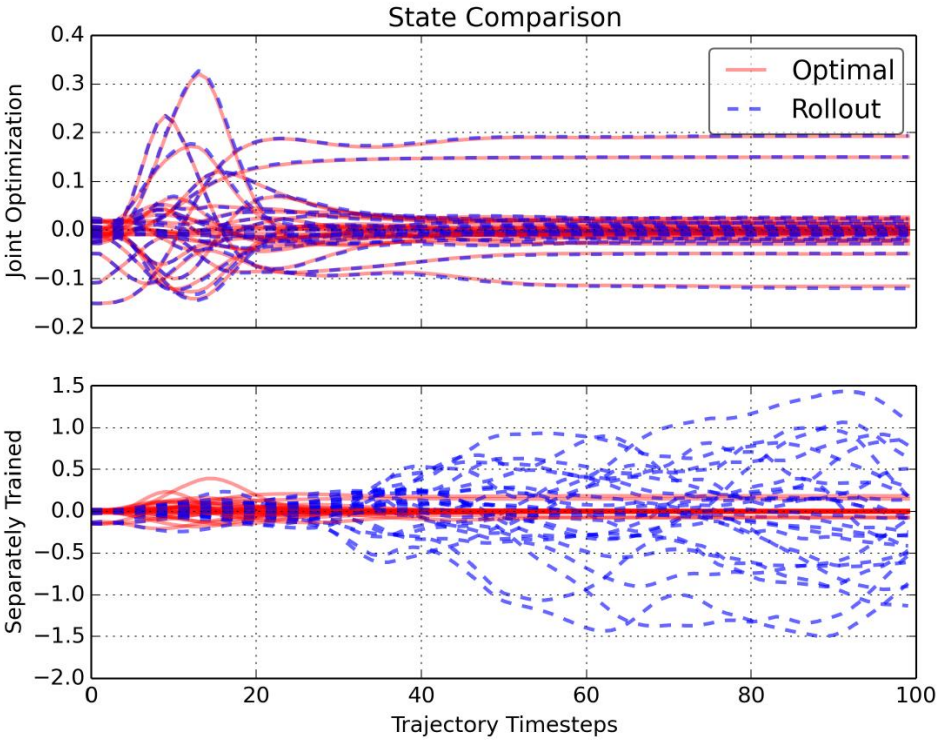
This does **not** work well:

- the cost is higher at the end of training;

- rollouts from the network are unstable.

Joint optimization of the network and the trajectories has proven to be essential.



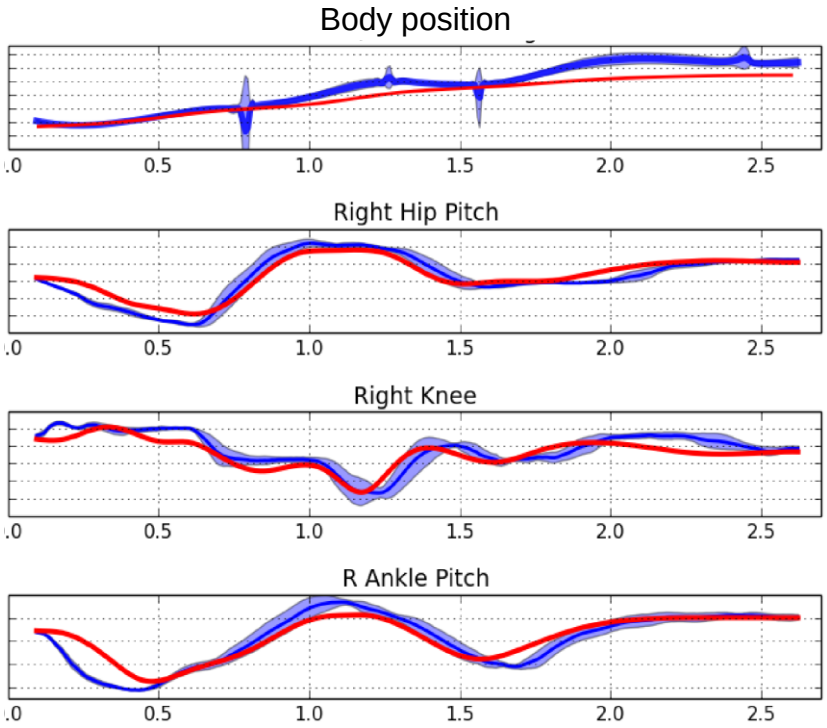


Robust control with ensemble-CIO

Robots tend to be different from our models, even after system identification.
Thus a movement optimized with respect to the model may not work on the robot.

To improve robustness, we generate ~10 perturbed models around the nominal one, and optimize a single trajectory that must be feasible for all perturbed models.

We then execute this trajectory on the robot, using the locally-optimal feedback gains obtained from the trajectory optimizer.



Mordatch, Lowrey and Todorov, IROS 2015

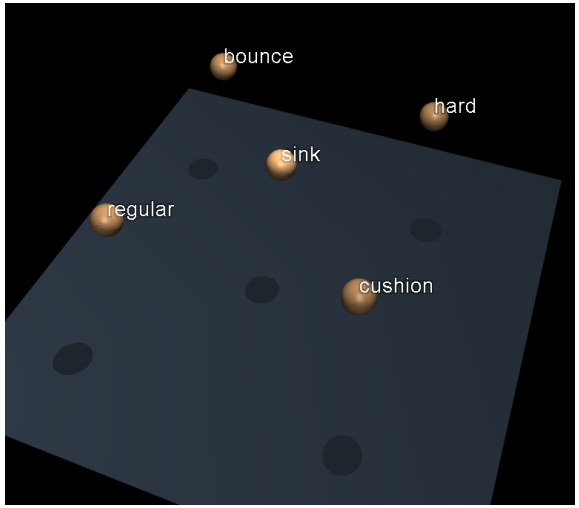
Contact dynamics with adjustable softness

We either have to start with a soft model and use continuation to make it harder, or in the case of MPC, optimize through a soft model all the time and apply the controls to the nominal hard model.

This requires a contact model that has adjustable softness, is sufficiently smooth to be used within an optimization loop, and can be evaluated quickly.

We have developed such a contact model and implemented it in MuJoCo.
Forward dynamics become a convex optimization problem.
Inverse dynamics are uniquely-defined and can be computed analytically.

NP-hard complementarity constraints are avoided.
Continuous-time formulation, amenable to RK4 etc.
Rich parameterization that facilitates system id.



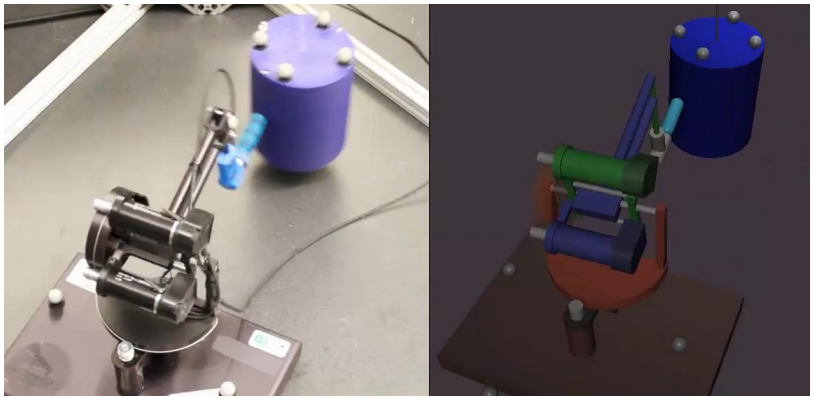
Todorov, *ICRA* 2010, 2011, 2014

Physically-consistent estimation

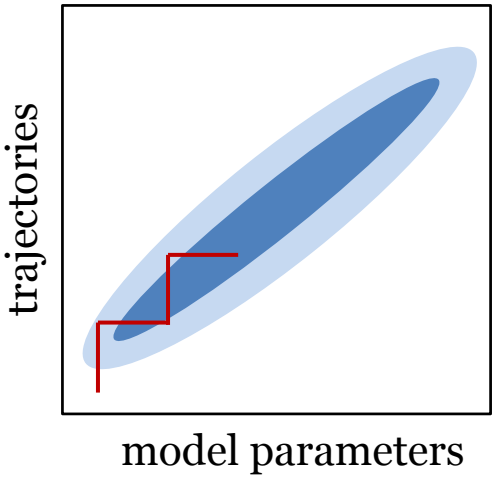
Given (noisy and incomplete) sensor data:
- actuator controls
- movement kinematics
- contact forces

estimate:
- movement kinematics
- contact forces
- model parameters

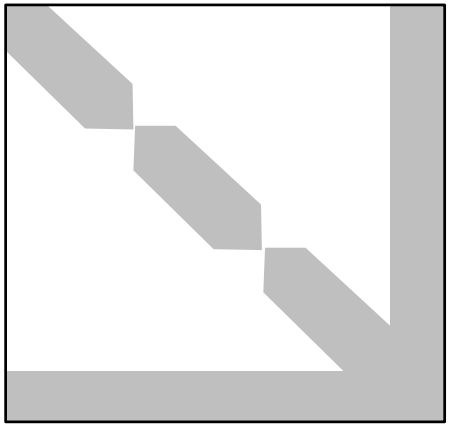
by minimizing the difference between measured and predicted sensor data, plus deviations from parameter priors.



coupled optimization

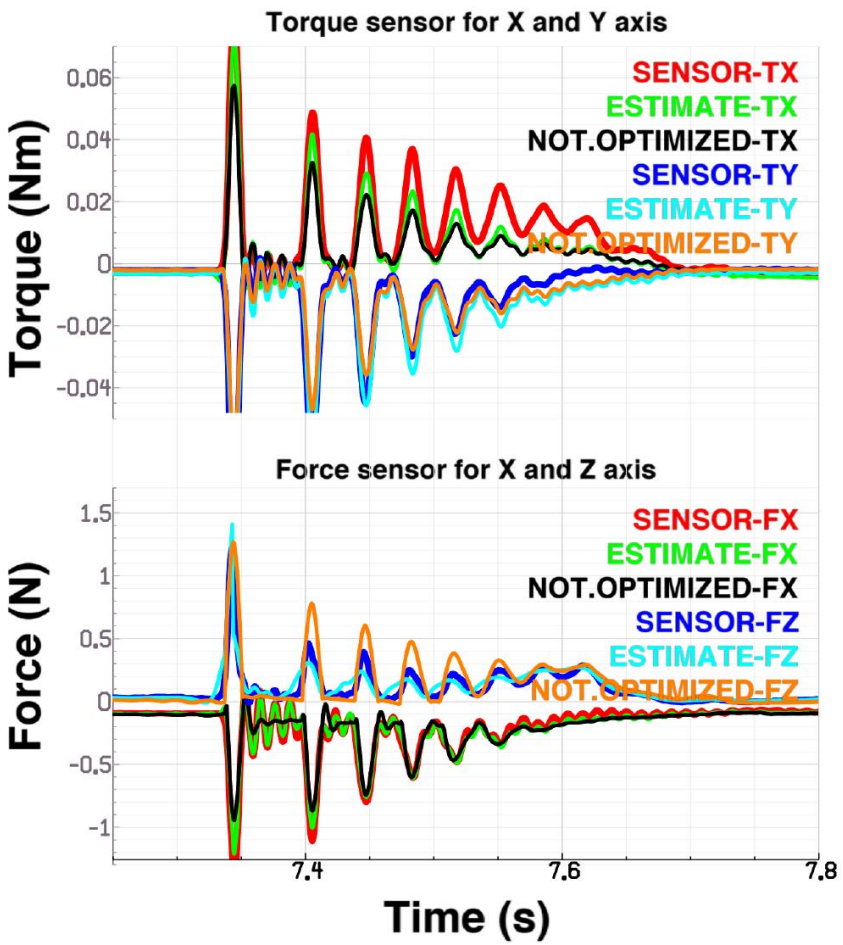


sparse Hessian factorization



Kolev and Todorov, 2015, under review

Results



Model parameters from different datasets

	Parameter			
Dataset #	*Coefficient of friction*	*Contact stiffness* $\frac{N}{m}$	*Spring stiffness* $\frac{Nm}{rad}$	*Spring damping* $\frac{mNm}{\frac{rad}{s}}$
Nominal	1.000	100	50.0	10.0
$ID\#1$	0.247	221	48.4	19.9
$ID\#2$	0.288	94.8	38.5	3.2
$ID\#3$	0.268	196	41.9	16.6
$ID\#4$	0.213	292	46.9	12.4
$ID\#5$	0.250	338	41.4	20.2

Model parameters from multiple restarts

Parameter	Units	Mean	Variance
Final Cost		126.7	18.0
Friction coefficient		0.2770	0.0079
Contact softeness	$\frac{N}{m}$	272.3	102.9
Spring stiffness	$\frac{Nm}{rad}$	41.3	0.93
Spring damping	$\frac{mNm}{\frac{rad}{s}}$	15.86	1.51

MuJoCo: www.mujoco.org

MuJoCo Pro
SDK for expert users;
invitation-only preview.

MuJoCo Sim
simulator built with the SDK;
coming soon.

MuJoCo HAPTIX
Sim + motion capture and VR;
developed for DARPA;
available now.

MuJoCo is just the simulator.

The next step is to implement a universal optimizer on top of it.