Feedback Control for Manipulation

Russ Tedrake
Sept 11, 2018

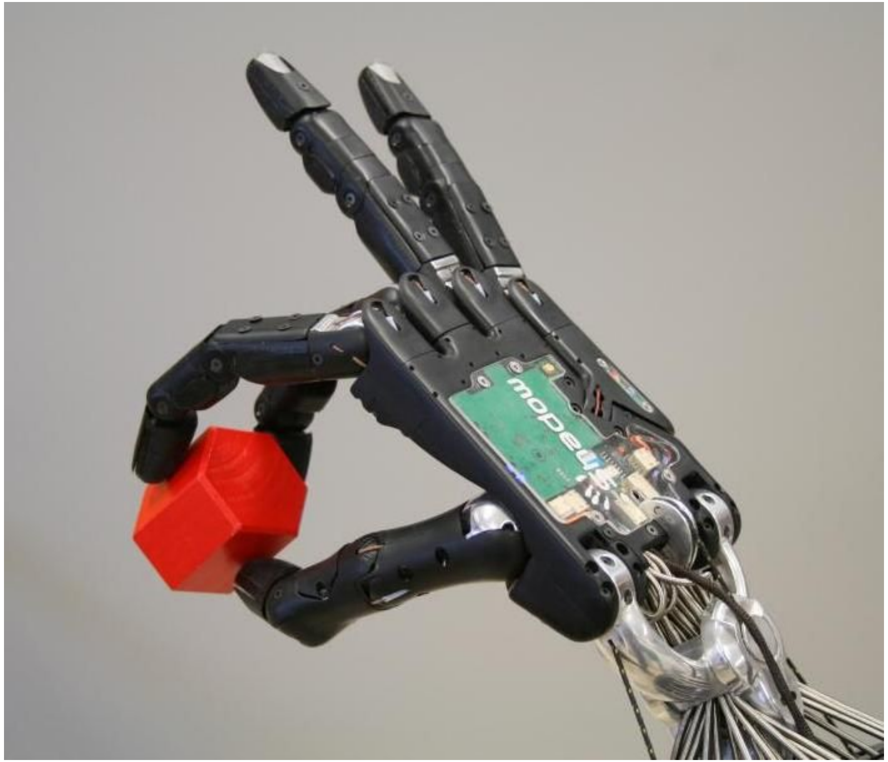
Massachusetts Institute of Technology

TOYOTA RESEARCH INSTITUTE

Aaron showed success stories.

I want to discuss where control theory has ~~fallen short.~~ Vistas.

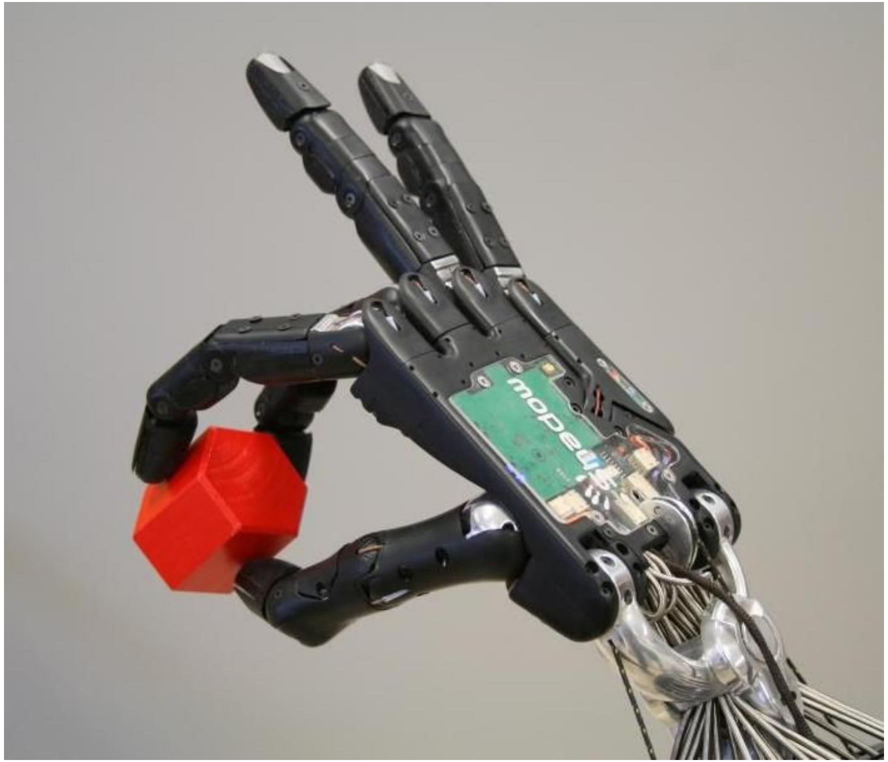
Nobody uses feedback control in state-of-the-art manipulation



...despite common agreement that robustness is a bottleneck.

“Most robots fail to pick up most objects most of the time”

-- Stefanie Tellex, 2016.



Let me be a bit more precise...

principled

Nobody uses feedback control in manipulation

Why no feedback?

- Don't need it?
  - Underactuated hands and enveloping grasps work well

Why no feedback?

- Don't need it?
  - Underactuated hands and enveloping grasps work well
  - … but there is much more to manipulation than enveloping grasps!
- Don't have the right sensors?
  - But we do have contact sensors (albeit expensive and not super robust)
  - and depth cameras are amazing
- Inaccurate models? Uncertainty?
  - But good control should accommodate these
  - … for most tasks we have sufficient control authority

- I think it's a failing of our algorithms

Three core challenges / vistas

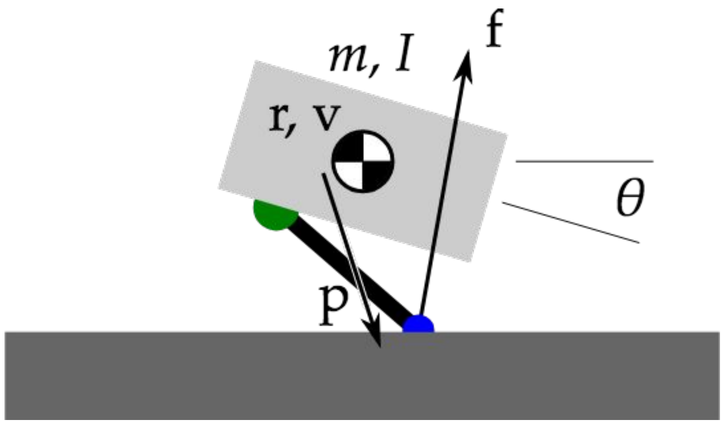
1. Combinatorics (of non-smooth mechanics in *contact-rich* interactions)

2. Severe partial observability + uncertainty
    - Full-state feedback often not viable/practical.
    - Central role of ***Perception.***
    - Solution? Principled approaches to ***Output Feedback***?

3. Wrong specification language
    - Mismatch between the way modern systems are being specified and the requirements we (typically) consume in control.

Combinatorics of Contact

Non-smooth mechanics of contact

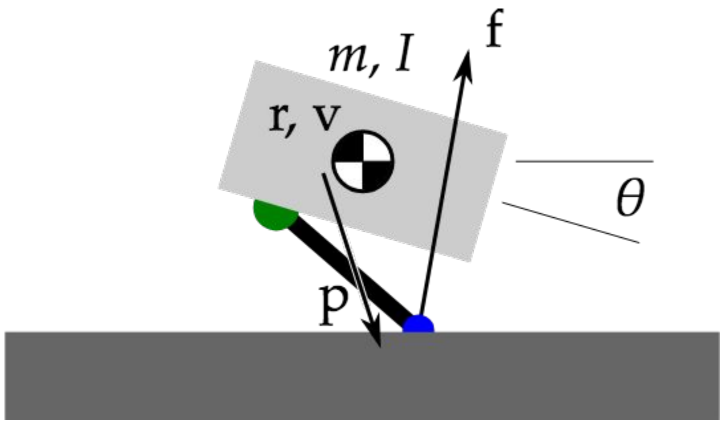


- Second-order differential equations (F=ma)
- but contact forces are
  - discontinuous (or stiff) in state -- no force unless we have contact.
  - set-valued (e.g. Coulomb friction)

⇒ (measure) differential inclusions / time-stepping linear complementarity problems

What does this imply for MPC?

MPC for contact mechanics



Linearization cannot capture even the local dynamics.

Locally valid approximation looks like a piecewise-affine system (PWA):

$$\mathcal{D}^i := \{(x, u) \mid F^i x + G^i u \le h^i\}$$

$$x^+ = A^i x + B^i u + c^i,$$
$$i \in \mathcal{I} \mid (x, u) \in \mathcal{D}^i,$$

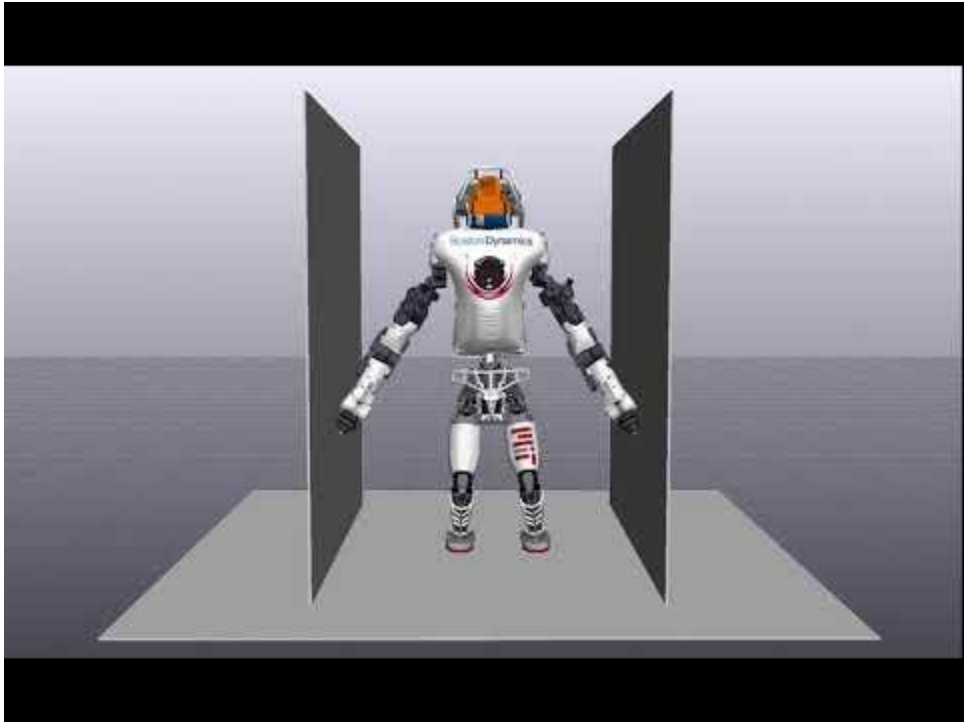
MPC for contact mechanics

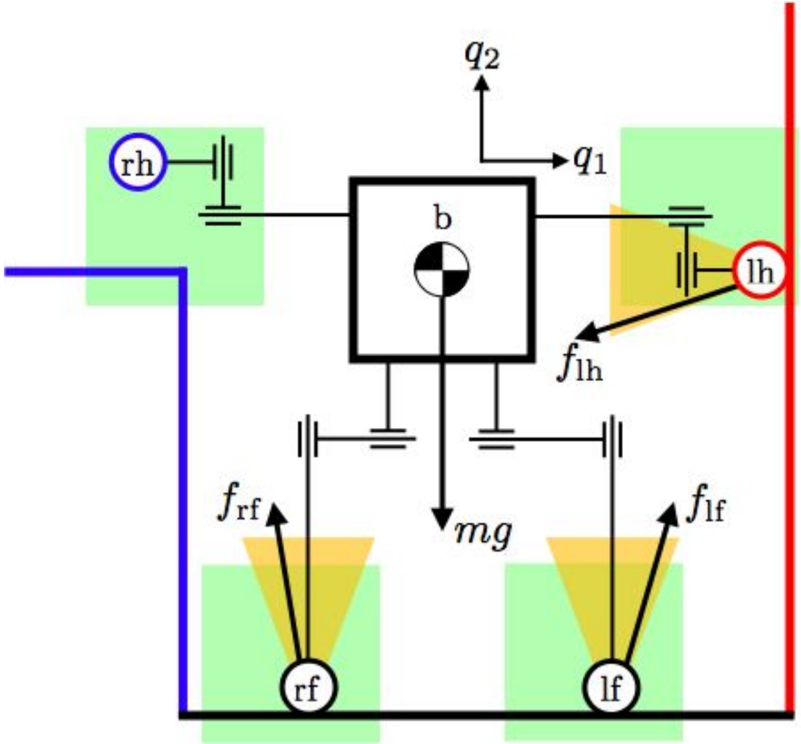
(Local) “contact MPC” problem naturally formulated as a ***mixed-integer convex optimization.***

$$
\begin{aligned}
\min_{u(t),x(t)} \quad & \|x(N)\|_P^2 + \sum_{t=0}^{N-1} \|x(t)\|_Q^2 + \|u(t)\|_R^2 \\
\text{s.t.} \quad & x(t+1) = A_i x(t) + B_i u(t) + c_i \quad \text{if} \quad (x(t), u(t)) \in \mathcal{D}_i \\
& x(0) = x_0 \\
& x(N) \in \mathcal{X}_N
\end{aligned}
$$

An important lesson from walking robots

Linearize in the "right" coordinates -- (here, centroidal dynamics)





A computational bottleneck

Mixed-integer problem has, at least,

2 x (number of potential contact ***pairs***) x (number of timesteps)

binary variables. [Some of this is real, some is a limitation of our transcription]

We are not yet close to solving this at real-time rates. Currently exploring:

- Tighter formulations (from disjunctive programming)
- Approximate explicit MPC
- Lyapunov-based (LMI/sums-of-squares) synthesis
- ...

Tight formulations for PWA MPC

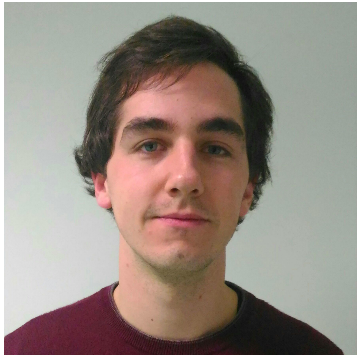
Obviously rich background in Hybrid MPC. (*Bemporad, Morari, ...*)

Performance of mixed-integer solvers depends on

- number of decision variables
- tightness of the convex relaxations during branch and bound
- complex (secret) heuristics in commercial solvers

Leverage (well-known) results from **disjunctive programming** to discuss the "strength" of our MI formulations.

Tight formulations for PWA MPC



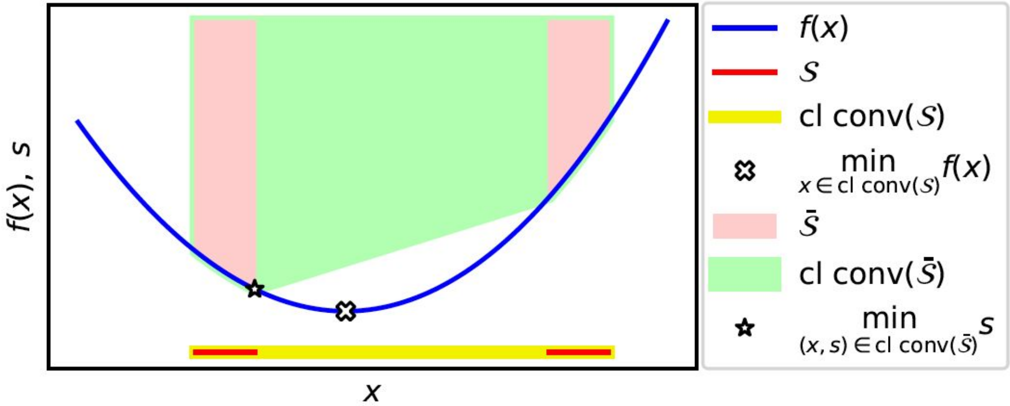
Key ideas:

- Convex hull formulation for subgroups of decision variables
  - balance tightness of relaxation with number of binary variables.
- Use the objective in the convex hull

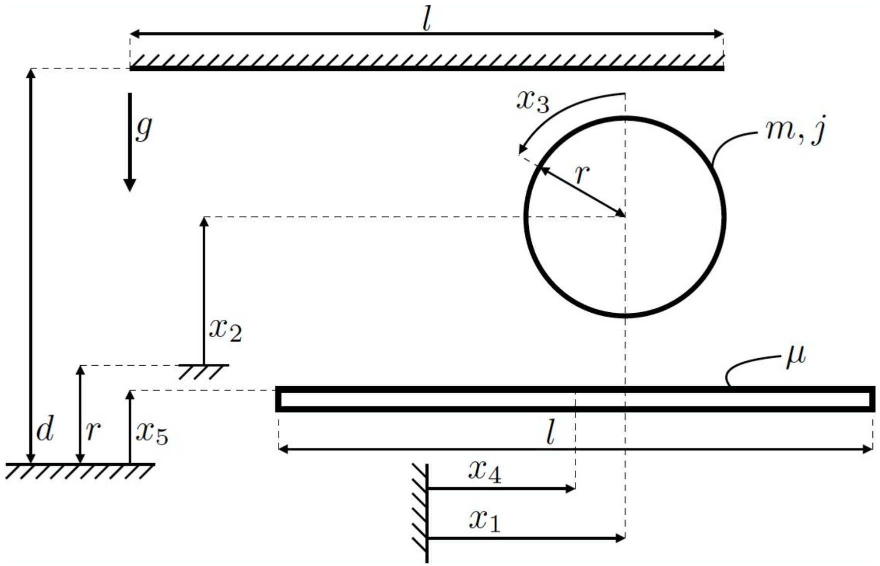
Mixed-Integer Formulations for Optimal Control of Piecewise-Affine Systems

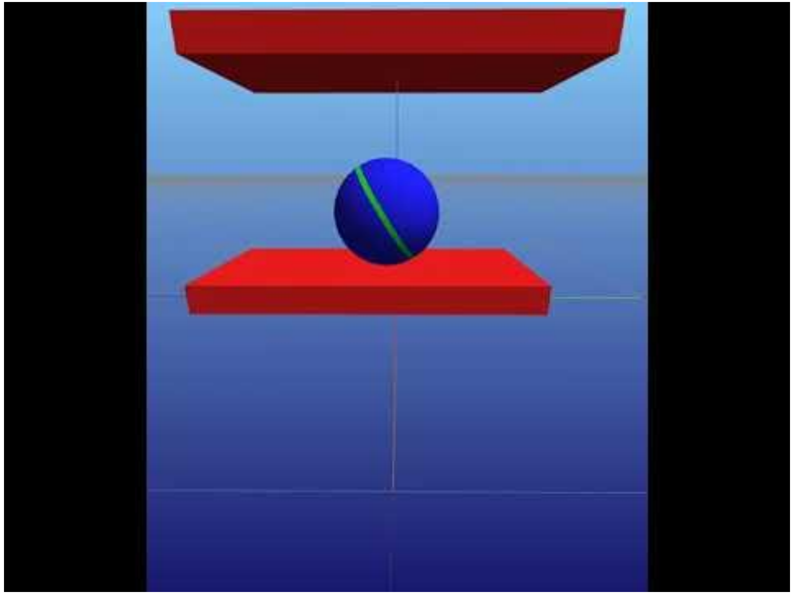
Tobia Marcucci and Russ Tedrake

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{tobiam,russt}@mit.edu



Example: 2D (frictional) ball reorientation

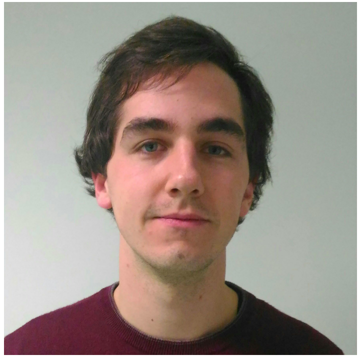




Traditional formulation does not find a feasible solution in 1 hour

Tight formulations solve to ***global*** optimality in ~ 320 seconds

Approximate Explicit MPC



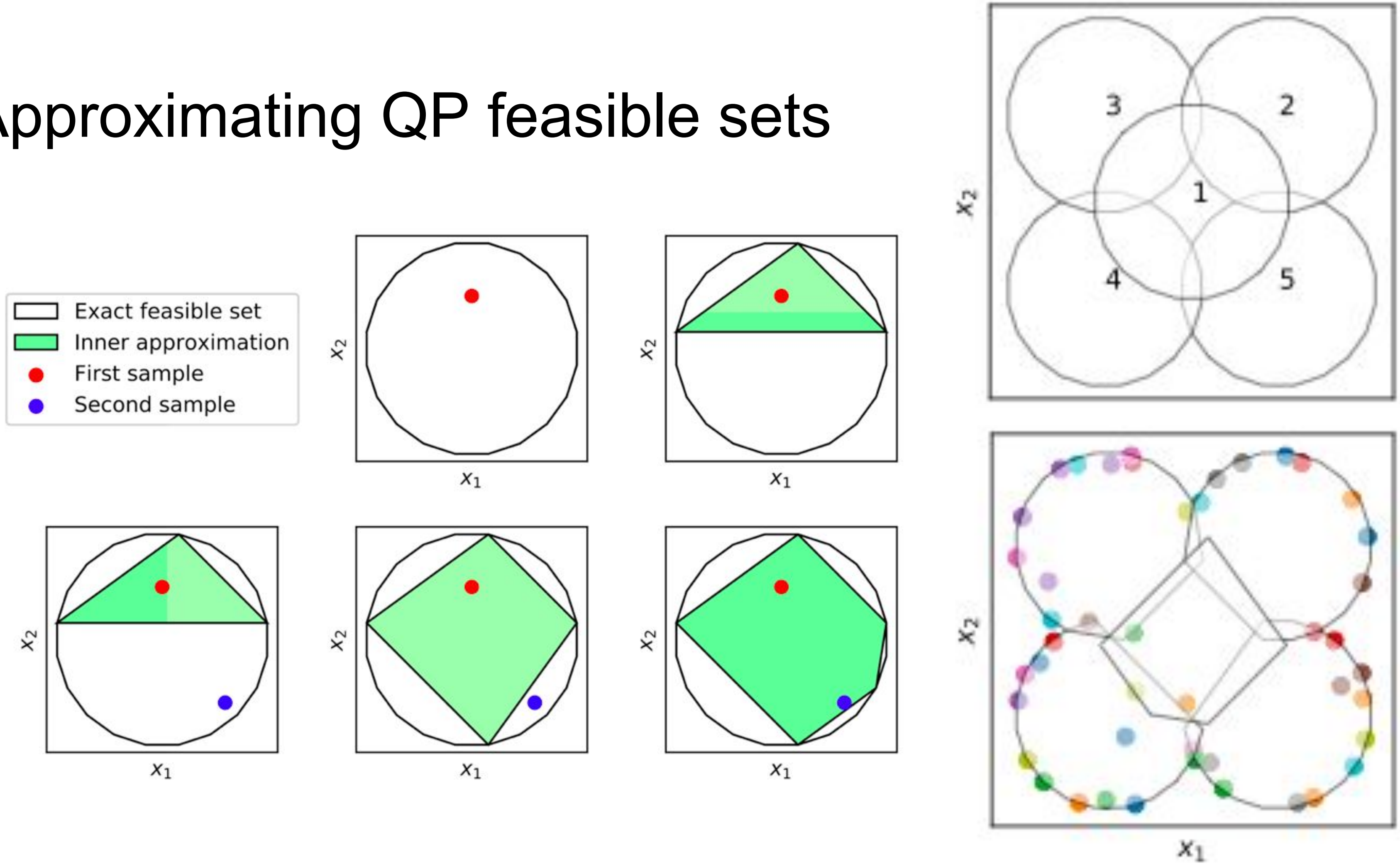
Still cannot achieve real-time rates (but still trying!)
What about Explicit MPC?

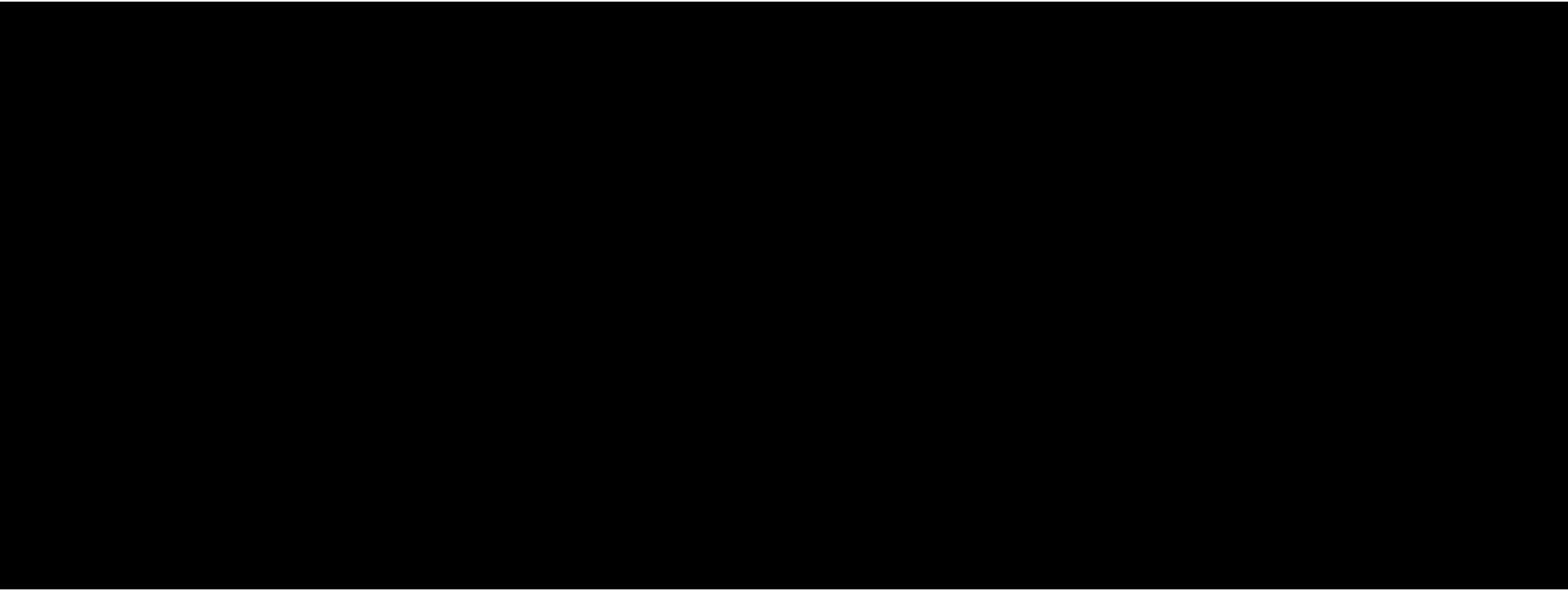
- Note that the hybrid case loses some of the nice properties (policy is still locally affine, but critical regions are no longer simple polytopes)
- Exact explicit MPC still intractable
- *Can we approximate this function (ideally guaranteeing strict feasibility) with simpler functions?*

One Approach:

- Sample in the state space, solve the MIQP.
- Approximate the feasible set of the QP with the integer solution fixed.
- Find new sample that is outside existing feasible sets (via rejection sampling)
- Repeat

Approximating QP feasible sets





System has 8 states, 8 inputs
593 selected mode sequences
(out of $5^{10} \approx 10^{7}$)
QPs are solved in ~ 25 ms

Still ***guarantee closed-loop stability***.
(but sacrificed global optimality)

Still working hard on it...

Limitations:

- Requires expensive precomputation phase. (maybe ok?)
- Depends heavily on state estimation.

Also exploring SDP relaxations, etc.

I believe good policies exist that take a much simpler form. They may also be more robust.

- Formal design of (simple) reactive controllers. Aka “output feedback”.

Output Feedback

What is the state space of this system?

Does (full) state estimation / feedback even make sense?

With my controls hat on:

- Model-order reduction + (reduced) state estimation + control?
  - Note: relevant subspace depends on the objective
  - "Subspace" identification may be more like "representation learning"
- ...

It was very interesting to hear stories last night about the birth of state-space methods / modern control.

But I feel that we are now reaching its limits.

Output Feedback

Simplest(?) case to describe:

$$x^+ = Ax + Bu$$
$$y = Cx + Du$$

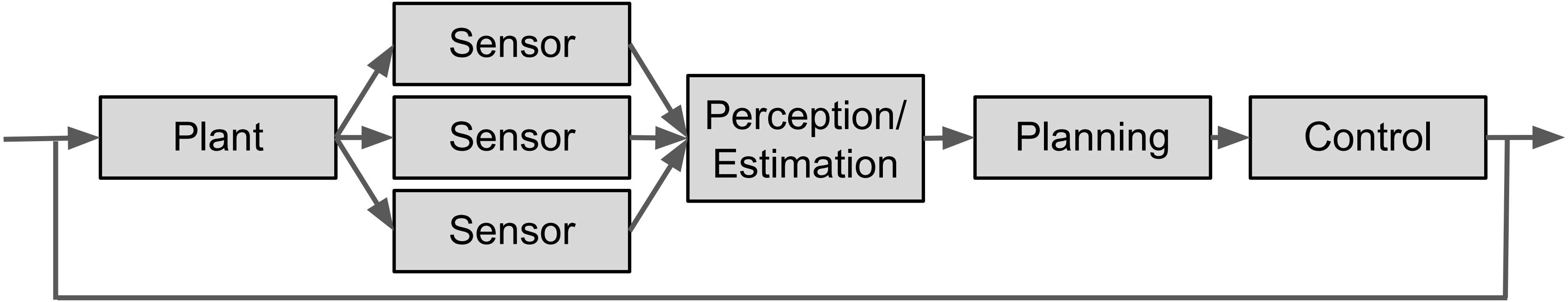
Want to find feedback gains *K* such that $u = -Ky$ stabilizes the system.

This “static” output feedback known to be NP hard [Blondel, ‘97]

Dynamic output feedback when the controller has internal state.

LQG is the special case we can solve.

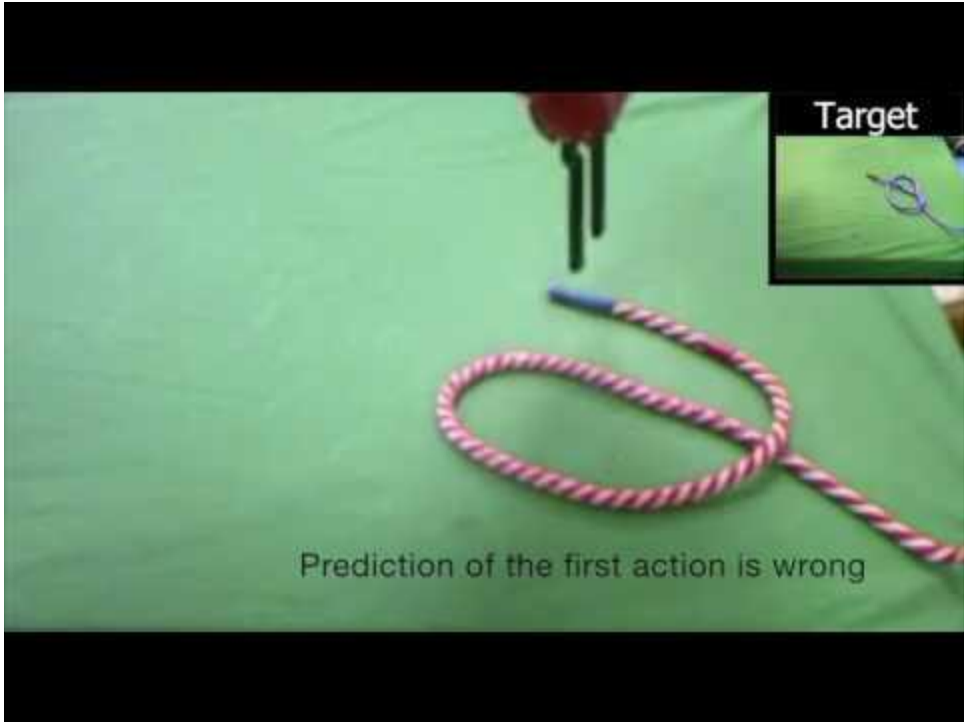
But the complexity of perception breaks our existing tools…



- Sensors include cameras ⇒ sensor model is a photo-realistic rendering engine
- Perception components (especially) include deep neural networks.
- Plant model has to capture distributions over natural scenes (lighting conditions)

Deep Learning for Control

Deep learning has another name for it: End-to-end learning. (aka “Pixels to torques”)



Pulkit Agrawal et al
2017

Deep Learning for Control

Many approaches:

- Reinforcement Learning
- Imitation Learning
- “Self-supervised” learning

Static Output Feedback w/ Convolutional Networks

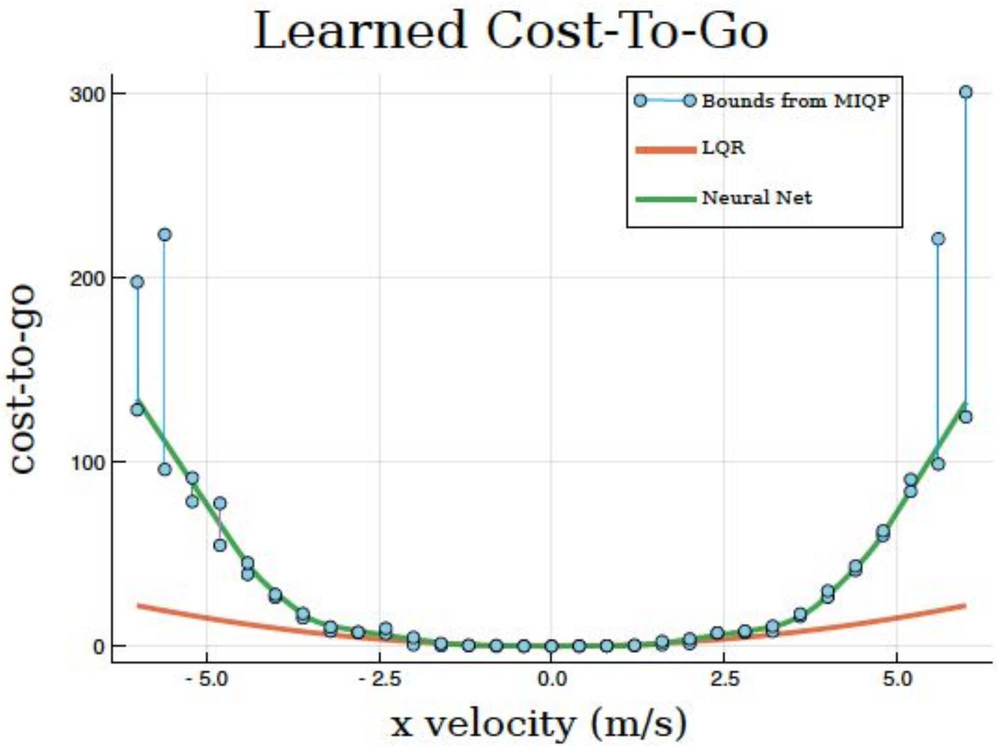
Dynamic Output Feedback w/ Recurrent Networks

Most applications to date use only stochastic gradient descent

Learned Value Interval Supervision

Can we use samples from MIQP to train a neural network controller?

- Structurally reasonable match to explicit MPC solutions.
- Expensive to solve MIQP to optimality
- Early termination of solver (or non-uniqueness of optimal soln) complicate policy learning
- But early termination of solver still gives bounds on cost-to-go.



work by Robin Deits

Systems theory applied to Deep Nets

Q: Can we derive meaningful input/output bounds on a deep neural network?

Evaluating Robustness of Neural Networks with Mixed Integer Programming

Vincent Tjeng **Kai Xiao** **Russ Tedrake**
Massachusetts Institute of Technology
Cambridge, MA 02139
{vtjeng, kaix, russt}@mit.edu

- For ReLU networks (with max-pooling, etc):
  - Can produce weak bounds on very large networks (using the LP relaxation)[1]
  - Branch-and-bound gives progressively tighter bounds; optimal bounds on modest architectures (MNIST)
- New work w/ Sasha Megretski on L2 gains for recurrent nets using IQC

Output Feedback for Manipulation (summary)

Simple, robust, output feedback controllers exist… and I don’t know how to find them (reliably)

Authoring Requirements

(perhaps my version of the “data-driven control” theme)

Machine learning is challenging the way that we perform systems engineering:

Rules of Machine Learning:

Best Practices for ML Engineering

Martin Zinkevich

This document is intended to help those with a basic knowledge of machine learning get the benefit of Google's best practices in machine learning. It presents a style for machine learning, similar to the Google C++ Style Guide and other popular guides to practical programming. If you have taken a class in machine learning, or built or worked on a machine-learned model, then you have the necessary background to read this document.

27 Feb 2018 | 16:48 GMT

Creating Driving Tests for Self-Driving Cars

Volvo-backed Zenuity wants to prove that autonomous vehicles can drive more safely than humans

By Erik Coelingh and Jonas Nilsson

Still a disconnect between **requirements** used in industry and problem formulations for robust control

Author distributions over environments/scenarios is hard; “corner cases” from large scale testing remain central

L2-gain-style computations are not enough[1]

Scenario-based verification and synthesis

Standard robust control formulation:

Find a controller $u = K(y)$ that minimizes some objective over many realizations of the plant (worst case, in expectation, etc).

But the realizations are drawn from distributions over tasks / environments

- which are very hard to author,
- typically sample-based,
- typically incredibly sparse (and expensive to obtain)

Need principled approaches to optimal experiment design, system ID, and “distributional robustness” that scale to this complexity.

- Mixing statistical methods and systems theory to address the complexity of distributional robustness

Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation

NIPS 2018

Search Algorithm	$\gamma = 0.14$	$\gamma = 0.16$	$\gamma = 0.18$	$\gamma = 0.40$	$\gamma = 0.42$
Naive	(2.0±2.0)e-5	(22.0± 6.6)e-5	(82.0±12.8)e-5	(334.4±8.0)e-4	(389.7±8.6)e-4
Cross-entropy	(3.2±2.6)e-6	(25.8 ± 4.5)e-5	(84.6± 9.3)e-5	(334.5 ± 8.0)e-4	(386.4 ± 8.6)e-4

Table 1: Estimate of rare-event probability p_γ (non-vision ego policy), with standard deviations

Search Algorithm	$\gamma = 0.26$	$\gamma = 0.28$	$\gamma = 0.30$	$\gamma = 0.50$	$\gamma = 0.52$
Naive	(8.0±4.0)e-3	(8.0±4.0)e-3	(12.0±4.9)e-3	(13.8±1.5)e-2	(15.6±1.6)e-2
Cross-entropy	(2.7±2.1)e-3	(5.4±2.7)e-3	(6.4±2.7)e-3	(7.6±1.0)e-2	(8.1±1.0)e-2

Table 2: Estimate of rare-event probability p_γ (vision-based ego policy), with standard deviations

My path forward

Scaling optimization-based synthesis to manipulation

I believe (to my core) in structured optimization and machine learning.

In ML: “whomever has the most data will win”.

For me: I covet parametric models (of mechanics, sensors, controllers, …).
Models should enable optimization-based design/analysis:

- Gradients (via autodiff)
- Introspection of sparsity, convexity
- Facilitate varying levels of fidelity

http://drake.mit.edu (on github)

- A modeling framework
  - Rigorous about declaring state, parameters, uncertainty, etc.
  - Physics engine, Rendering engine, Sensor models, ...
  - Gradients, Sparsity, Convexity, ...
- An optimization library
- Optimization algorithms for dynamical systems (planning, feedback design, perception/estimation, system identification…)

DRAKE

A PLANNING, CONTROL, AND ANALYSIS TOOLBOX FOR NONLINEAR DYNAMICAL SYSTEMS

Summary: Three core challenges / vistas

Nobody uses (principled) feedback control in manipulation.

1. Combinatorics (of non-smooth mechanics in *contact-rich* interactions)
2. Severe partial observability + uncertainty
   - Are we reaching the limits of state space methods?
   - Simple, robust, output feedback controllers exist and I don’t know how to find them reliably
3. Control should align w/ best practices for Machine Learning Engineering