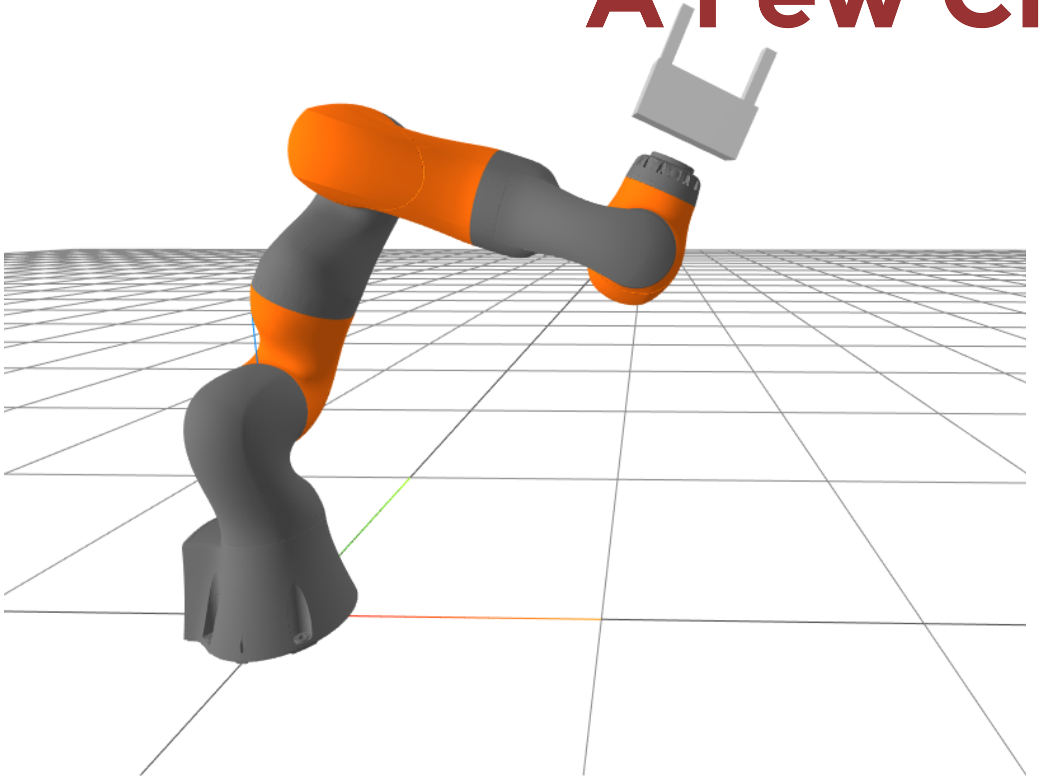
Follow along at https://slides.com/russtedrake/rl-2020-bc/live
or view later at https://slides.com/russtedrake/rl-2020-bc/



A Few Challenge Problems from Robotics

Russ Tedrake

Massachusetts Institute of Technology

TOYOTA RESEARCH INSTITUTE

My goals for today

- How well is control working in robotics today?

- A few core questions/challenges
  - What role can simulation play?
  - Why/when does gradient-based policy search work?
  - Parameterizations/algorithms from control.
  - RL in the rare-event regime.
- Challenge problem instances
  - Raibert's hopper, planar gripper, plates, onions, shoe laces.

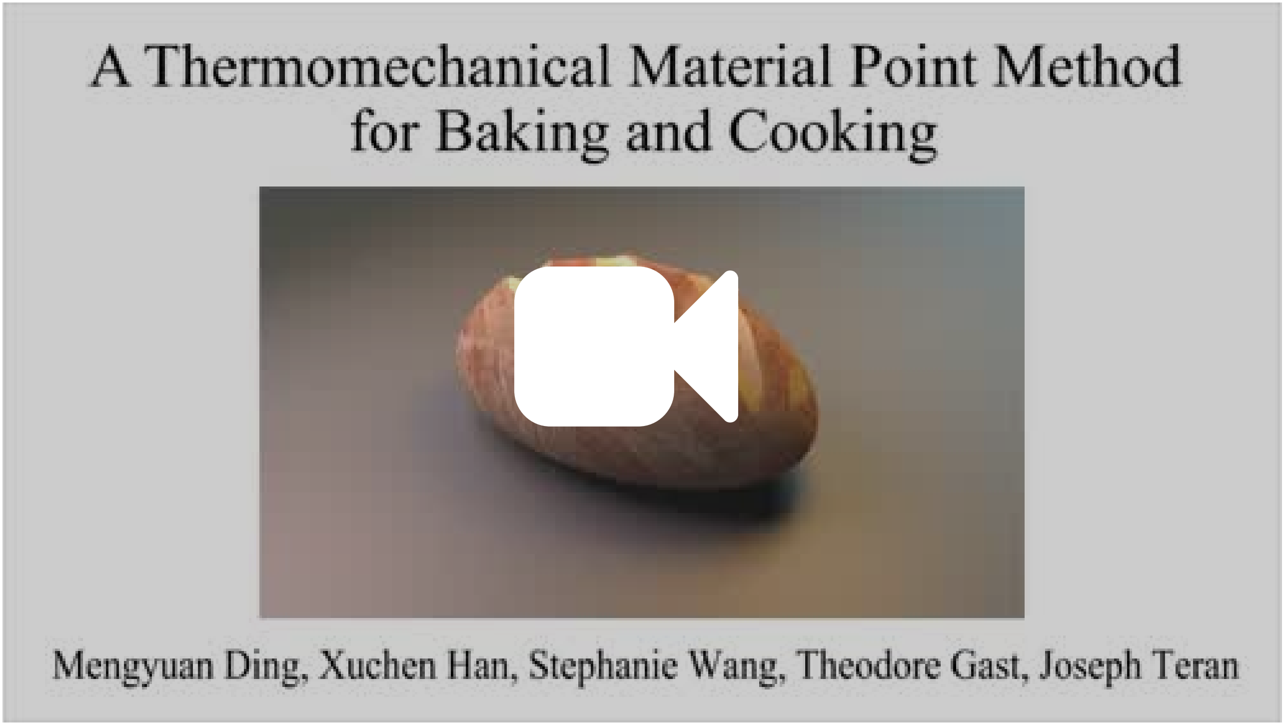
How well is control working in robotics?

https://www.youtube.com/embed/fRj34o4hN4I?enablejsapi=1&mute=1

https://www.youtube.com/embed/UtqWlHxfs8E?enablejsapi=1&mute=1&start=200

Feels like an opportunity for RL?

Is the task difficult because we don't have a model?



What role can simulation play?

for spreading peanut butter, buttoning my shirt, etc.

As more people started applying RL to robots, we saw a distinct shift from "model-free" RL to "model-based" RL.

(most striking at the 2018 Conference on Robot Learning)

As more people started applying RL to robots, we saw a distinct shift from "model-free" RL to "model-based" RL.

(most striking at the 2018 Conference on Robot Learning)

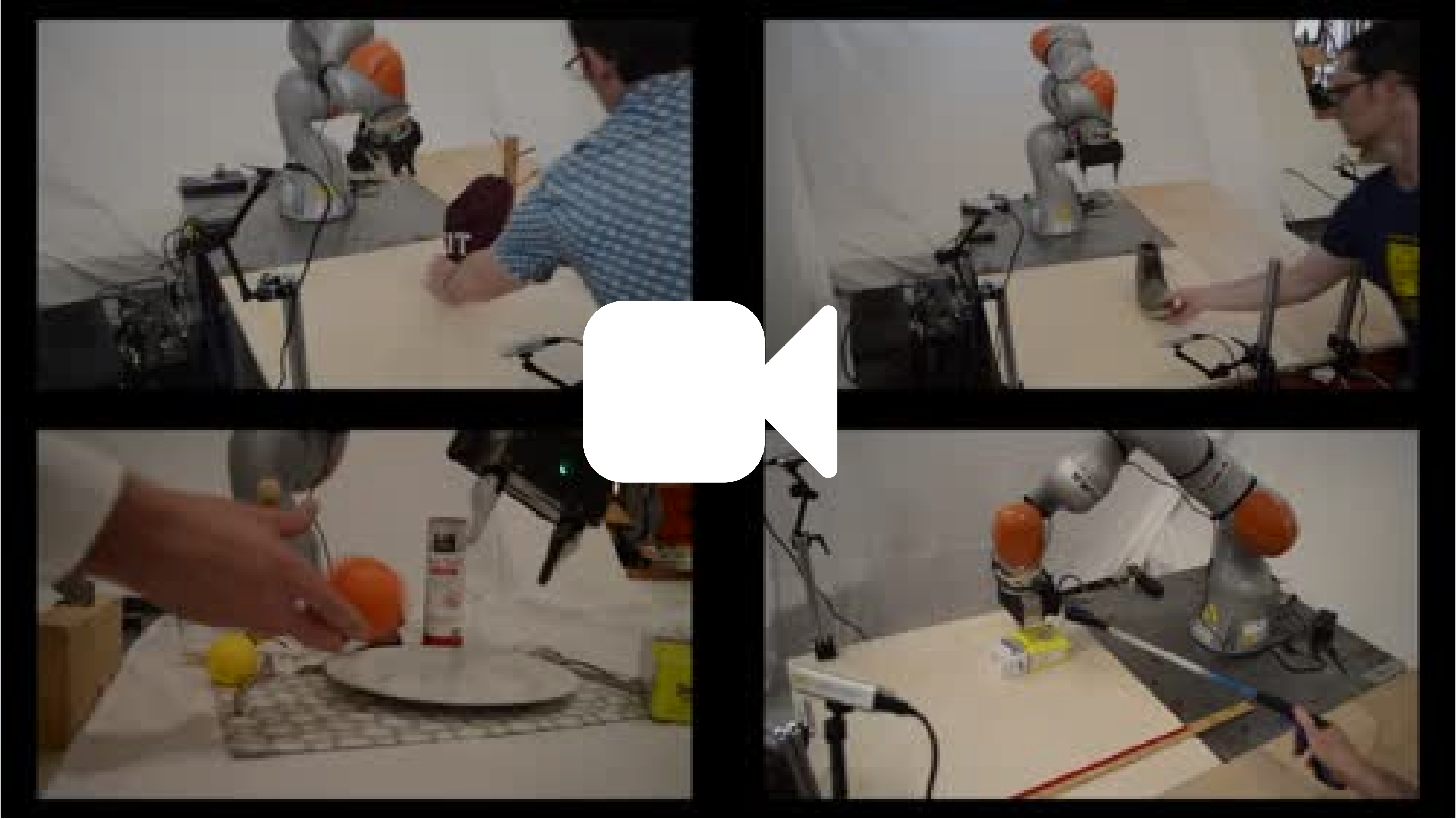
Recent IFRR global panel on "data-driven vs physics-based models"

As more people started applying RL to robots, we saw a distinct shift from "model-free" RL to "model-based" RL.

(most striking at the 2018 Conference on Robot Learning)

Recent IFRR global panel on "data-driven vs physics-based models"

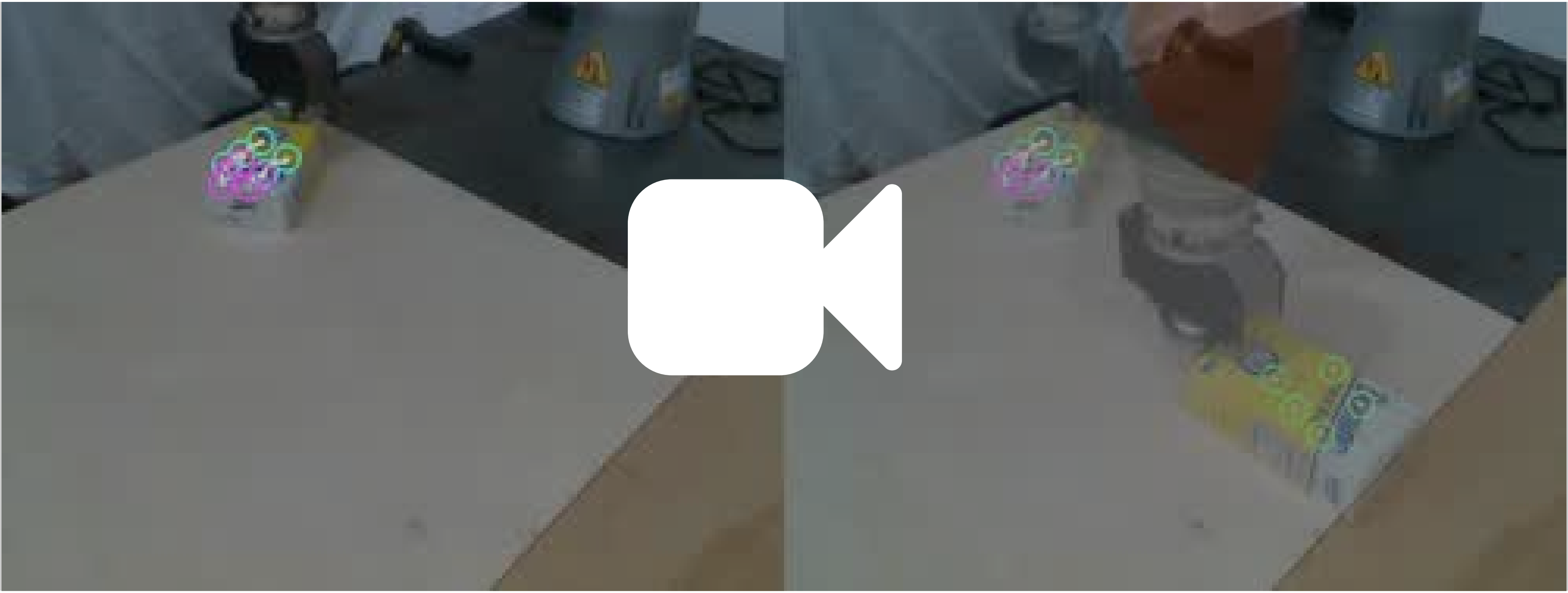
(caveat: I didn't choose the panel name)



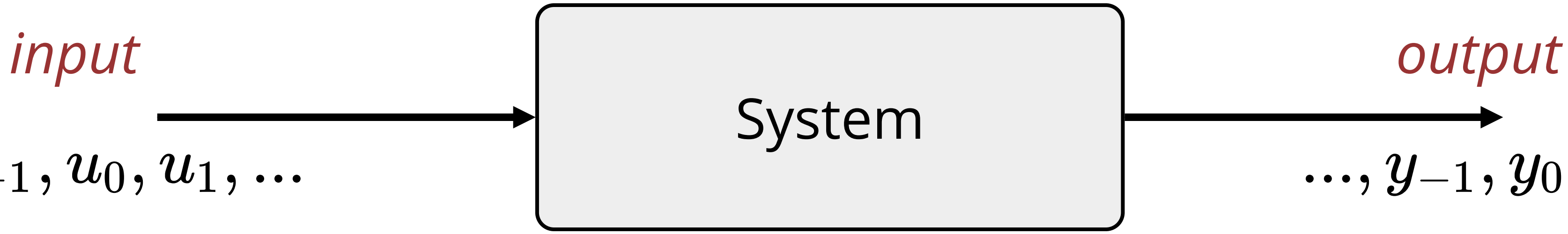
Core technology: Deep learning perception module that learns "dense correspondences"

Learn a ***deep*** dynamic model of "keypoint" dynamics.

Online: use model-predictive control (MPC)



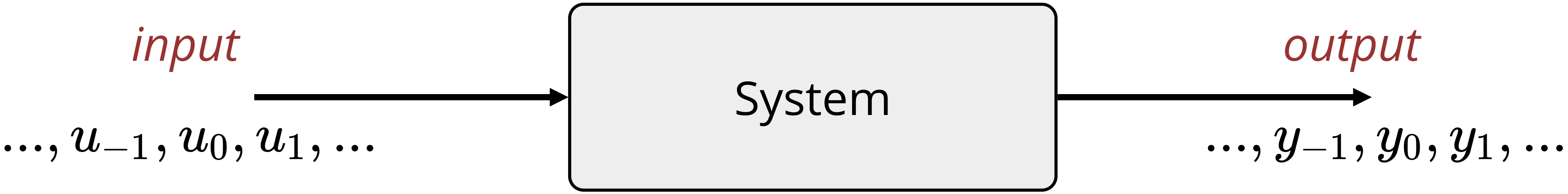
What is a (dynamic) model?



$\ldots, u_{-1}, u_0, u_1, \ldots$

$\ldots, y_{-1}, y_0, y_1, \ldots$

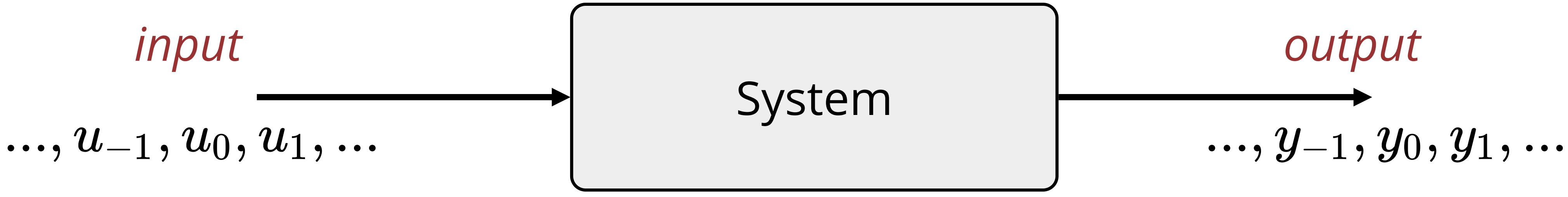
What is a (dynamic) model?



State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

What is a (dynamic) model?

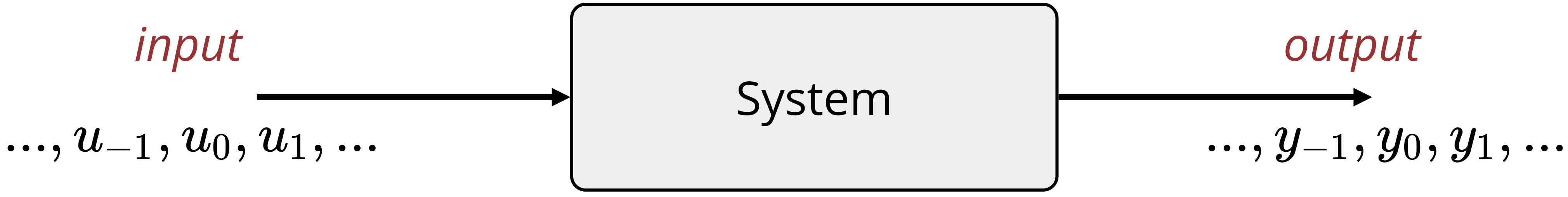


State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

state

What is a (dynamic) model?



State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

state

noise/disturbances

What is a (dynamic) model?

input

$$\ldots, u_{-1}, u_0, u_1, \ldots$$

System

output

$$\ldots, y_{-1}, y_0, y_1, \ldots$$

State-space

$$x_{n+1} = f(n, x_n, u_n, w_n, \theta)$$
$$y_n = g(n, x_n, u_n, w_n, \theta)$$

state

noise/disturbances

parameters

What is a (dynamic) model?

input

$\ldots, u_{-1}, u_0, u_1, \ldots$

System

output

$\ldots, y_{-1}, y_0, y_1, \ldots$

State-space

$$x_{n+1} = f(n, x_n, u_n, w_n, \theta)$$
$$y_n = g(n, x_n, u_n, w_n, \theta)$$

state

noise/disturbances

parameters

$p(\theta, x_0, w_0, w_1, \ldots)$

What is a (dynamic) model?

input System *output*

$$\dots, u_{-1}, u_0, u_1, \dots \qquad \dots, y_{-1}, y_0, y_1, \dots$$

State-space

$$x_{n+1} = f(n, x_n, u_n, w_n, \theta)$$
$$y_n = g(n, x_n, u_n, w_n, \theta)$$

state

noise/disturbances

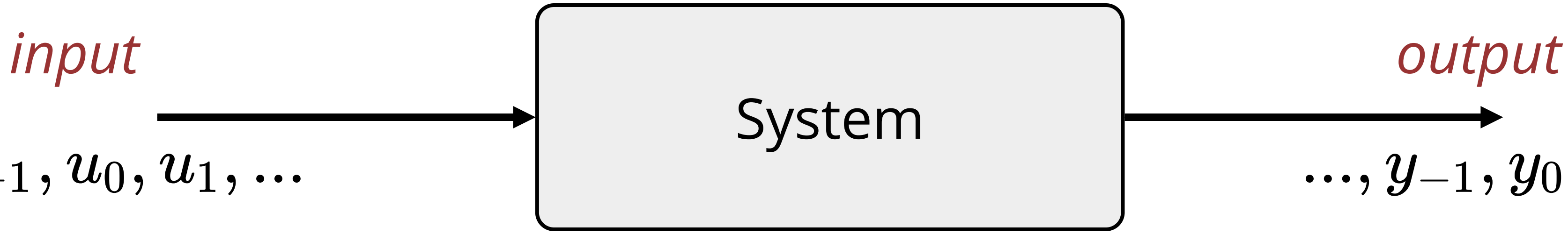
parameters

$$p(\theta, x_0, w_0, w_1, \dots)$$

Auto-regressive (eg. ARMAX)

$$y_{n+1} = f(n, u_n, u_{n-1}, \dots, y_n, y_{n-1}, \dots, w_n, w_{n-1}, \dots, \theta)$$

What is a (dynamic) model?



$\ldots, u_{-1}, u_0, u_1, \ldots$

$\ldots, y_{-1}, y_0, y_1, \ldots$

What is a (dynamic) model?

input → System → *output*

$\ldots, u_{-1}, u_0, u_1, \ldots$ → System → $\ldots, y_{-1}, y_0, y_1, \ldots$

State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

Lagrangian mechanics,

Recurrent neural networks (e.g. LSTM), ...

What is a (dynamic) model?

input

$\ldots, u_{-1}, u_0, u_1, \ldots$

System

output

$\ldots, y_{-1}, y_0, y_1, \ldots$

State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

Lagrangian mechanics,

Recurrent neural networks (e.g. LSTM), ...

Feed-forward networks

(e.g. y_n*= image)*

What is a (dynamic) model?

input → System → *output*

$\ldots, u_{-1}, u_0, u_1, \ldots$ → System → $\ldots, y_{-1}, y_0, y_1, \ldots$

State-space

$$\begin{aligned} x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\ y_n &= g(n, x_n, u_n, w_n, \theta) \end{aligned}$$

Lagrangian mechanics,

Recurrent neural networks (e.g. LSTM), …

Auto-regressive (eg. ARMAX)

$$\begin{aligned} y_{n+1} = f(n, &u_n, u_{n-1}, \ldots, \\ &y_n, y_{n-1}, \ldots, \\ &w_n, w_{n-1}, \ldots, \theta) \end{aligned}$$

Feed-forward networks

(e.g. y_n= image)

What is a (dynamic) model?

input → System → *output*

$\dots, u_{-1}, u_0, u_1, \dots$ $\dots, y_{-1}, y_0, y_1, \dots$

State-space

$$\begin{aligned} x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\ y_n &= g(n, x_n, u_n, w_n, \theta) \end{aligned}$$

Lagrangian mechanics,

Recurrent neural networks (e.g. LSTM), …

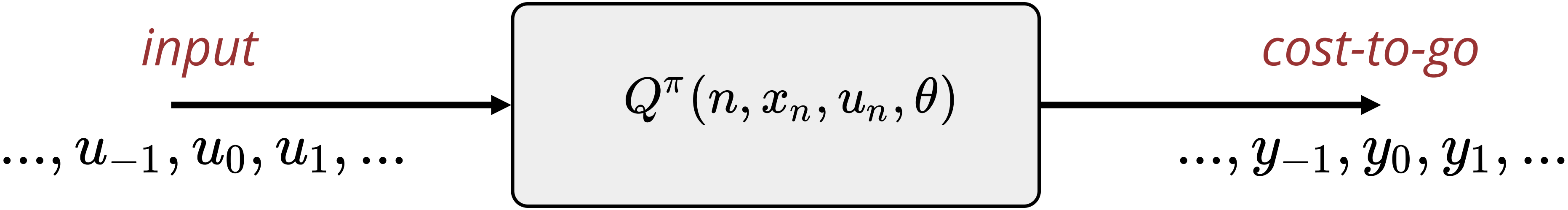
Auto-regressive (eg. ARMAX)

$$\begin{aligned} y_{n+1} = f(n, &u_n, u_{n-1}, \dots, \\ &y_n, y_{n-1}, \dots, \\ &w_n, w_{n-1}, \dots, \theta) \end{aligned}$$

Feed-forward networks

(e.g. y_n = image)

Models come in many forms



Q-functions are models, too. They try to predict only one output (the cost-to-go).

As you know, people are using *Q*-functions in practice on non-Markovian state representations.

Model "class" vs model "instance"

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

- f and g describe the model class.

Model "class" vs model "instance"

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

- f and g describe the model class.
- with θ describes a model instance.

Model "class" vs model "instance"

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

- f and g describe the model class.
- with θ describes a model instance.

"Deep models vs Physics-based models?" is about model ***class****:*

Should we prefer writing f and g using physics or deep networks?

Model "class" vs model "instance"

$$\boxed{\begin{aligned} x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\ y_n &= g(n, x_n, u_n, w_n, \theta) \end{aligned}}$$

- f and g describe the model class.
- with θ describes a model instance.

*"Deep models vs Physics-based models?" is about model **class**:*

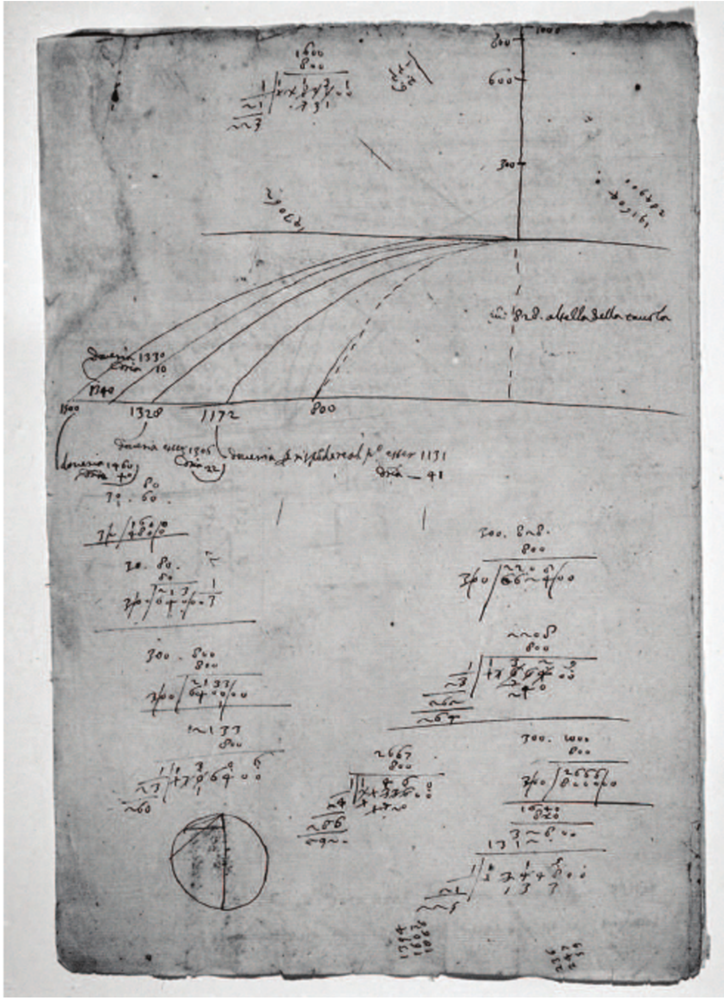
Should we prefer writing f and g using physics or deep networks?

Maybe not so different from

- should we use ReLU or $\tanh$?
- should we use LSTMs or Transformers?

Galileo, Kepler, Newton, Coulomb, Hooke...

were ***data scientists.***

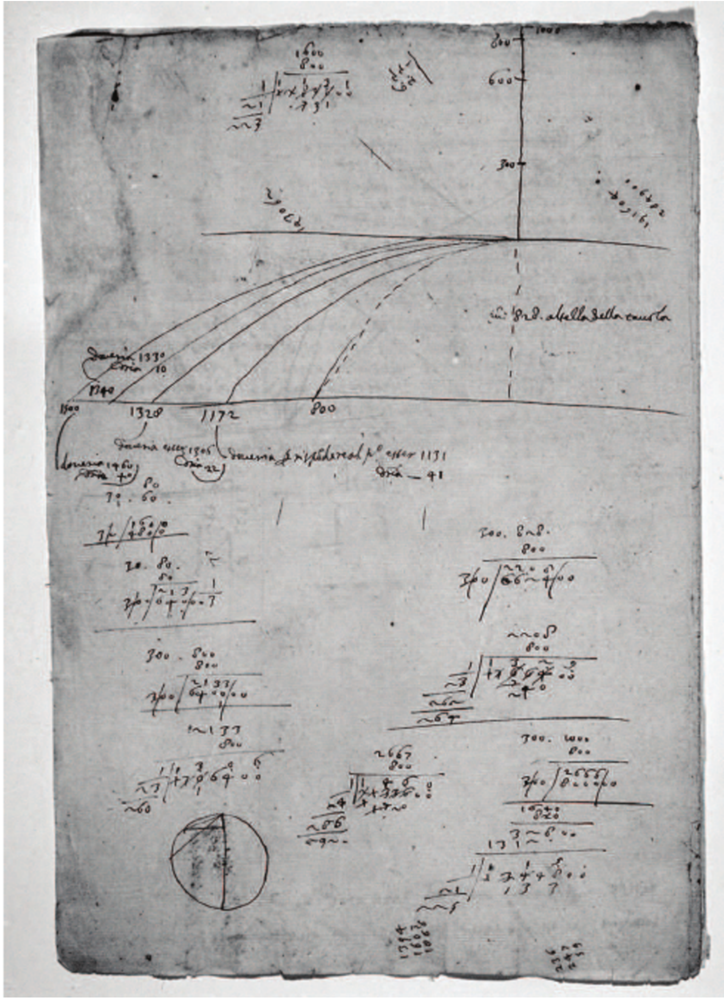


Galileo's notes on projectile motion

Galileo, Kepler, Newton, Coulomb, Hooke...

were ***data scientists.***

They fit very simple
models to very noisy data.



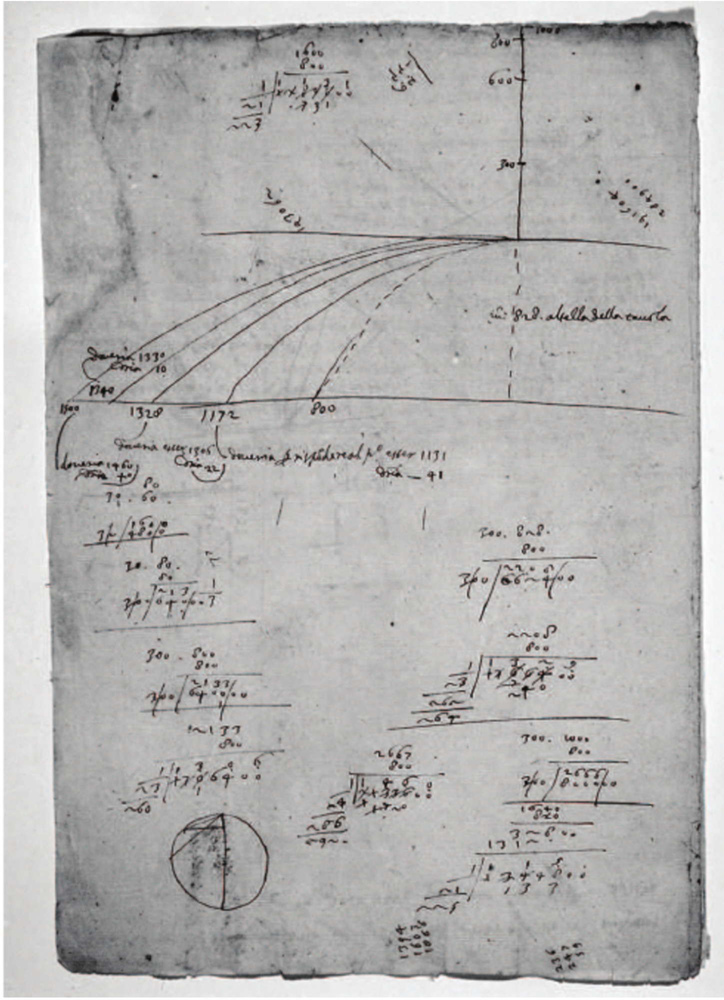
Galileo's notes on projectile motion

Galileo, Kepler, Newton, Coulomb, Hooke...

were ***data scientists.***

They fit very simple
models to very noisy data.

Gave us a rich ***class*** of
parametric models that
we could fit to new data.



Galileo's notes on projectile motion

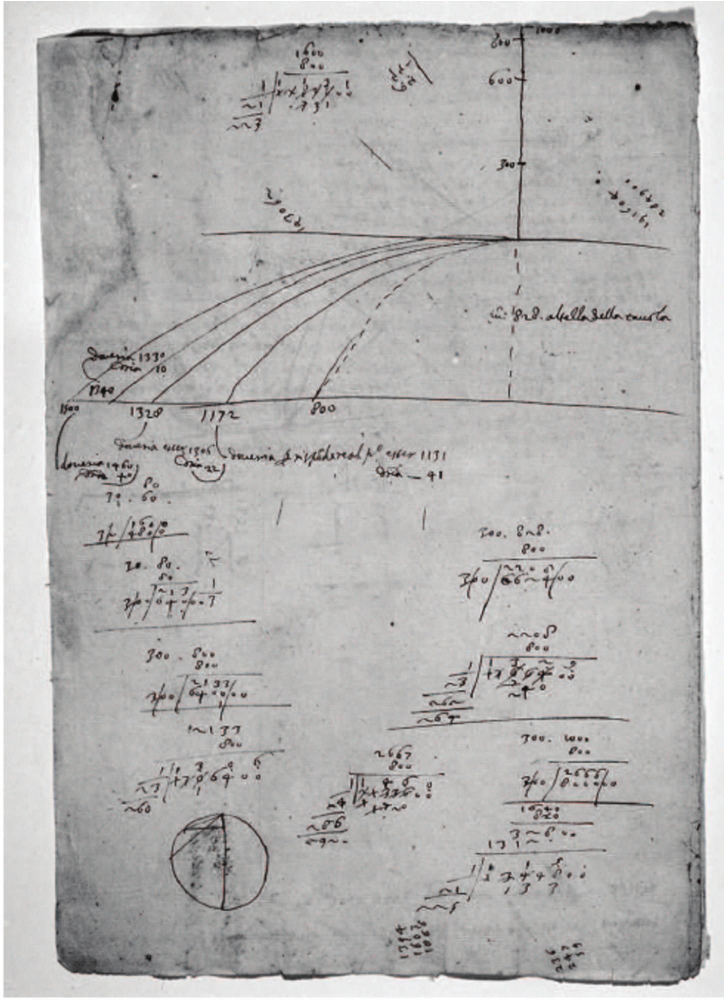
Galileo, Kepler, Newton, Coulomb, Hooke...

were ***data scientists.***

They fit very simple models to very noisy data.

Gave us a rich ***class*** of parametric models that ***we*** could fit to new data.

What if Newton had deep learning...?



Galileo's notes on projectile motion

https://www.youtube.com/embed/0LJb66aYtG8?
start=111&mute=1&enablejsapi=1&controls=0&rel=0&modestbranding=1

"All models are wrong, but some are useful" -- George Box

What makes a model useful?

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

What makes a model useful?

Unreal 5 engine trailer

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

What makes a model useful?

- Reasonable: $y_{sim}(u) \in \{y_{real}(u)\}$

Unreal 5 engine trailer

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

What makes a model useful?

- Reasonable: $y_{sim}(u) \in \{y_{real}(u)\}$
- Coverage: $\forall y_{real}(u), \exists y_{sim}(u)$

Unreal 5 engine trailer

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

What makes a model useful?

- Reasonable: $y_{sim}(u) \in \{y_{real}(u)\}$
- Coverage: $\forall y_{real}(u), \exists y_{sim}(u)$
- Accuracy: $p(y_{real}|u) \approx p(y_{sim}|u)$

Unreal 5 engine trailer

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

Unreal 5 engine trailer

What makes a model useful?

- Reasonable: $y_{sim}(u) \in \{y_{real}(u)\}$
- Coverage: $\forall y_{real}(u), \exists y_{sim}(u)$
- Accuracy: $p(y_{real}|u) \approx p(y_{sim}|u)$
- *Corollary:* Reliable system identification (data $\Rightarrow \theta$)

Use case: Simulation

e.g., for

- generating synthetic training data
- Monte-Carlo policy evaluation
- ***offline policy optimization***

Unreal 5 engine trailer

What makes a model useful?

- Reasonable: $y_{sim}(u) \in \{y_{real}(u)\}$
- Coverage: $\forall y_{real}(u), \exists y_{sim}(u)$
- Accuracy: $p(y_{real}|u) \approx p(y_{sim}|u)$
- *Corollary:* Reliable system identification (data $\Rightarrow \theta$)
- Generalizable, efficient, repeatable, **interpretable**/debuggable, ...

Use case: Online Decision Making (Planning/Control)

What makes a model useful?

- Reasonable, Accurate, Generalizable, ...
- ***Efficient / compact***
- ***Observable***
- ***Task-relevant***

Use case: Online Decision Making (Planning/Control)

What makes a model useful?

- Reasonable, Accurate, Generalizable, ...
- ***Efficient / compact***
- ***Observable***
- ***Task-relevant***

State-space models tend to be more efficient/compact, but require *state estimation.*

State-space

$$
\begin{aligned}
x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\
y_n &= g(n, x_n, u_n, w_n, \theta)
\end{aligned}
$$

vs.

$$
\begin{aligned}
y_{n+1} = f(n, & u_n, u_{n-1}, \dots, \\
& y_n, y_{n-1}, \dots, \\
& w_n, w_{n-1}, \dots, \theta)
\end{aligned}
$$

Auto-regressive

Use case: Online Decision Making (Planning/Control)

What makes a model useful?

- Reasonable, Accurate, Generalizable, ...
- ***Efficient / compact***
- ***Observable***
- ***Task-relevant***

State-space models tend to be more efficient/compact, but require *state estimation.*

- Ex: chopping onions.
  - Lagrangian state not observable.
  - Task relevant?

State-space

$$\begin{aligned} x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\ y_n &= g(n, x_n, u_n, w_n, \theta) \end{aligned}$$

vs.

$$\begin{aligned} y_{n+1} = f(n, u_n, u_{n-1}, \ldots, \\ y_n, y_{n-1}, \ldots, \\ w_n, w_{n-1}, \ldots, \theta) \end{aligned}$$

Auto-regressive

Use case: Online Decision Making (Planning/Control)

What makes a model useful?

- Reasonable, Accurate, Generalizable, ...
- ***Efficient / compact***
- ***Observable***
- ***Task-relevant***

State-space models tend to be more efficient/compact, but require *state estimation.*

- Ex: chopping onions.
  - Lagrangian state not observable.
  - Task relevant?
- Doesn't imply "no mechanics"

State-space

$$\begin{aligned} x_{n+1} &= f(n, x_n, u_n, w_n, \theta) \\ y_n &= g(n, x_n, u_n, w_n, \theta) \end{aligned}$$

vs.

$$\begin{aligned} y_{n+1} = f(n, u_n, u_{n-1}, \ldots, \\ y_n, y_{n-1}, \ldots, \\ w_n, w_{n-1}, \ldots, \theta) \end{aligned}$$

Auto-regressive

Occam's razor and predicting what ***won't*** happen

Perhaps the biggest philosophical difference between traditional physics models and "universal approximators".

Occam's razor and predicting what ***won't*** happen

Perhaps the biggest philosophical difference between traditional physics models and "universal approximators".

- f, g are *not* arbitrary. Mechanics gives us *constraints*:
  - Conservation of mass
  - Conservation of energy
  - Maximum dissipation
  - …

Occam's razor and predicting what ***won't*** happen

Perhaps the biggest philosophical difference between traditional physics models and "universal approximators".

- f, g are *not* arbitrary. Mechanics gives us *constraints*:
  - Conservation of mass
  - Conservation of energy
  - Maximum dissipation
  - ...
- Arguably these constraints give our models their *structure*
  - Control affine
  - Inertial matrix is positive definite
  - Inverse dynamics have "branch-induced sparsity"

Occam's razor and predicting what ***won't*** happen

Perhaps the biggest philosophical difference between traditional physics models and "universal approximators".

- f, g are *not* arbitrary. Mechanics gives us *constraints*:
  - Conservation of mass
  - Conservation of energy
  - Maximum dissipation
  - ...
- Arguably these constraints give our models their *structure*
  - Control affine
  - Inertial matrix is positive definite
  - Inverse dynamics have "branch-induced sparsity"
- *Without structure, maybe we can only ever do stochastic gradient descent...?*

For e.g. onions, laundry, peanut butter, ...

The failings of our physics-based models are mostly due to *the **unreasonable** burden of estimating the "Lagrangian state"* and parameters.

For e.g. onions, laundry, peanut butter, ...

The failings of our physics-based models are mostly due to *the **unreasonable** burden of estimating the "Lagrangian state"* and parameters.

The failings of our deep models are mostly due to our inability to due *efficient/reliable* planning, control design and analysis.

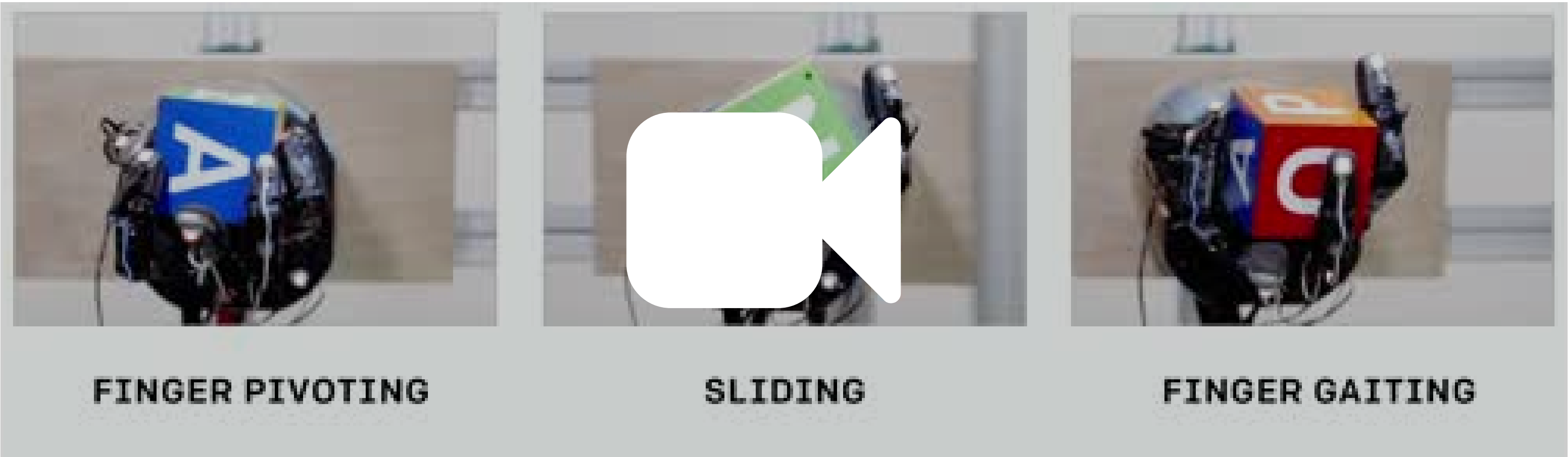
For e.g. onions, laundry, peanut butter, ...

The failings of our physics-based models are mostly due to *the **unreasonable** burden of estimating the "Lagrangian state"* and parameters.

The failings of our deep models are mostly due to our inability to due *efficient/reliable* planning, control design and analysis.

I want the next Newton to come around and to work on onions, laundry, peanut butter...

Why/when does gradient-based policy search work?



OpenAI - Learning Dexterity

"PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance."

https://openai.com/blog/openai-baselines-ppo/

Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator

Maryam Fazel[1], Rong Ge[2], Sham M. Kakade[1], and Mehran Mesbahi[1]

Proceedings of Machine Learning Research vol 120:1–9, 2020 2nd Annual Conference on Learning for Dynamics and Control

Learning the model-free linear quadratic regulator via random search

Hesameddin Mohammadi HESAMEDM@USC.COM
Mahdi Soltanolkotabi SOLTANOL@USC.COM
Mihailo R. Jovanović MIHAILO@USC.COM
Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089.

But there are also cases where it will not work...

A simple counter-example from static output feedback:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u, \quad y = \mathbf{C}x,$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 1 & 3 \end{bmatrix},$$

$$u = -ky.$$

The set of stabilizing k is a disconnected set.

http://underactuated.mit.edu/policy_search.html

But there are also cases where it will not work...

A simple counter-example from static output feedback:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u, \quad y = \mathbf{C}x,$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

$$u = -ky.$$

k	**Maximum real closed-loop eigenvalue**
0.9	-0.035
1.5	0.032
2.1	-0.009

The set of stabilizing k is a disconnected set.

http://underactuated.mit.edu/policy_search.html

What characterizes the good cases?

- Maybe it is *over-parameterization* of deep policies?
  - Is there a comparable story to interpolating solutions in high-dimensional policy space?

- Maybe it is the *distribution over tasks*?
  - Control has traditionally studied algorithms that must work for all A,B,C. Maybe the world never gives us the hard ones?
  - Optimizing simultaneously over diverse tasks might be easier than optimizing over one task.

Lessons from Control

(for instance: better controller parameterizations)

- Just because you *can* search over $u = -Kx$ directly, does not mean that you should!
  - Slow convergence
  - Set of stabilizing K is nontrivial
- If model is known, searching Q and R is better.

Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

ADPRL, 2012

paper link

For LQG, H_∞, etc., we know convex parameterizations.

Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

ADPRL, 2012

paper link

For LQG, H_∞, etc., we know convex parameterizations.

- Youla parameterization (disturbance-based feedback)

Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

ADPRL, 2012

paper link

For LQG, H_∞, etc., we know convex parameterizations.

- Youla parameterization (disturbance-based feedback)
- LMI formulations

Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

ADPRL, 2012

paper link

For LQG, H_∞, etc., we know convex parameterizations.

- Youla parameterization (disturbance-based feedback)
- LMI formulations
- Result in convex formulations only for linear systems, but the benefits are likely more general.

Feedback Controller Parameterizations for Reinforcement Learning

John W. Roberts
CSAIL, MIT
Cambridge, MA 02139
Email: jwr@mit.edu

Ian R. Manchester
CSAIL, MIT
Cambridge, MA 02139
Email: irm@mit.edu

Russ Tedrake
CSAIL, MIT
Cambridge, MA 02139
Email: russt@mit.edu

ADPRL, 2012

paper link

Youla parameters (time-domain, no-noise)

http://underactuated.csail.mit.edu/lqr.html

$$\min \sum_{n=0}^{N-1} \mathbf{x}^T[n]\mathbf{Q}\mathbf{x}[n] + \mathbf{u}^T[n]\mathbf{R}\mathbf{u}[n], \qquad \mathbf{Q} = \mathbf{Q}^T \succeq 0, \mathbf{R} = \mathbf{R}^T \succ 0$$

$$\text{subject to} \quad \mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n],$$
$$\mathbf{x}[0] = \mathbf{x}_0$$

$$\mathbf{u}[n] = \mathbf{K}_n\mathbf{x}[n],$$

$$\mathbf{x}[1] = \mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{K}_0\mathbf{x}_0,$$
$$\mathbf{x}[2] = \mathbf{A}(\mathbf{A} + \mathbf{B}\mathbf{K}_0)\mathbf{x}_0 + \mathbf{B}\mathbf{K}_1(\mathbf{A} + \mathbf{B}\mathbf{K}_0)\mathbf{x}_0$$
$$\mathbf{x}[n] = \left(\prod_{i=0}^{n-1}(\mathbf{A} + \mathbf{B}\mathbf{K}_i)\right)\mathbf{x}_0$$

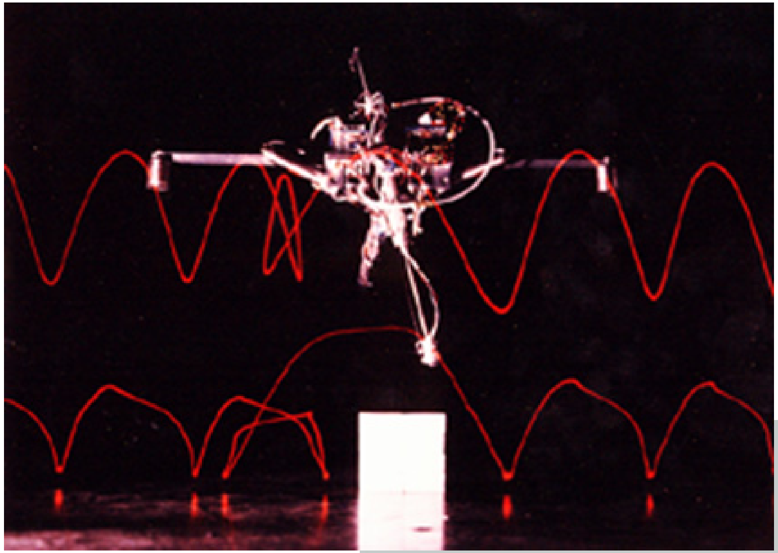
$$\mathbf{u}[n] = \tilde{\mathbf{K}}_n\mathbf{x}_0,$$

$$\mathbf{x}[1] = \mathbf{A}\mathbf{x}_0 + \mathbf{B}\tilde{\mathbf{K}}_0\mathbf{x}_0,$$
$$\mathbf{x}[2] = \mathbf{A}(\mathbf{A} + \mathbf{B}\tilde{\mathbf{K}}_0)\mathbf{x}_0 + \mathbf{B}\tilde{\mathbf{K}}_1\mathbf{x}_0$$
$$\mathbf{x}[n] = \left(\mathbf{A}^n + \sum_{i=0}^{n-1}\mathbf{A}^{n-i-1}\mathbf{B}\tilde{\mathbf{K}}_i\right)\mathbf{x}_0$$

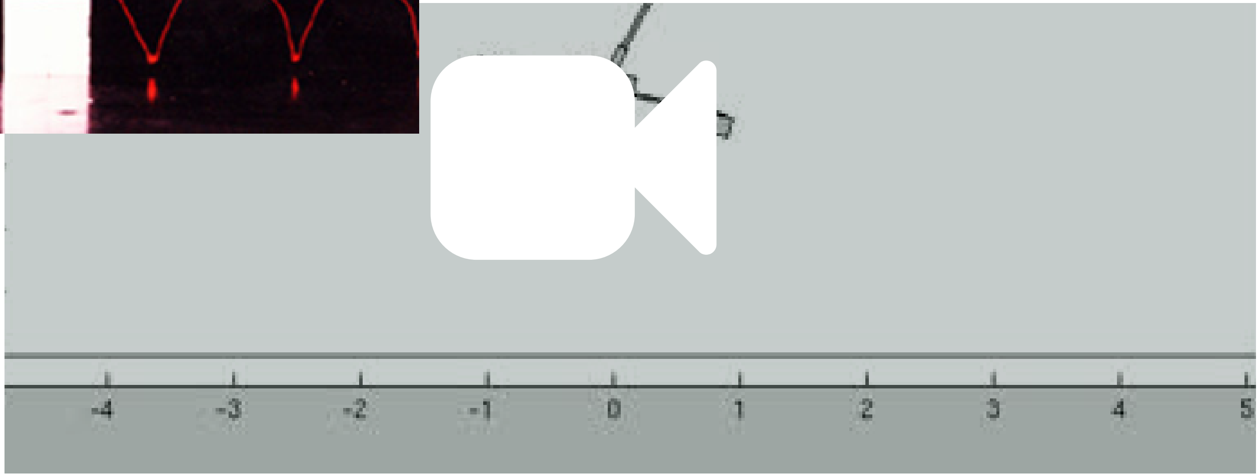
Control parameterizations from the "robot whisperers"

https://www.youtube.com/embed/aFuA50H9uek?enablejsapi=1&mute=1

The MIT Leg Lab Hopping Robots

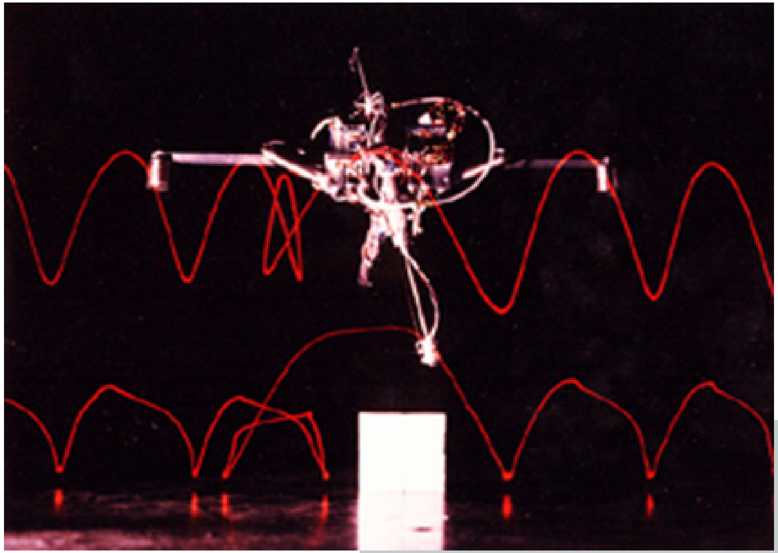


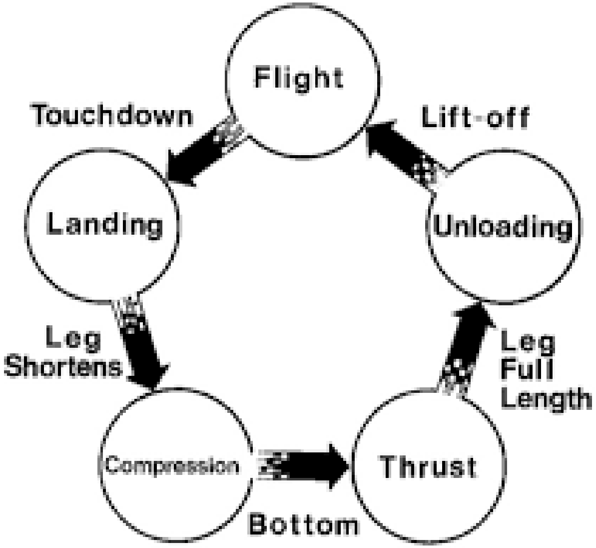
http://www.ai.mit.edu/projects/leglab/robots/robots.html

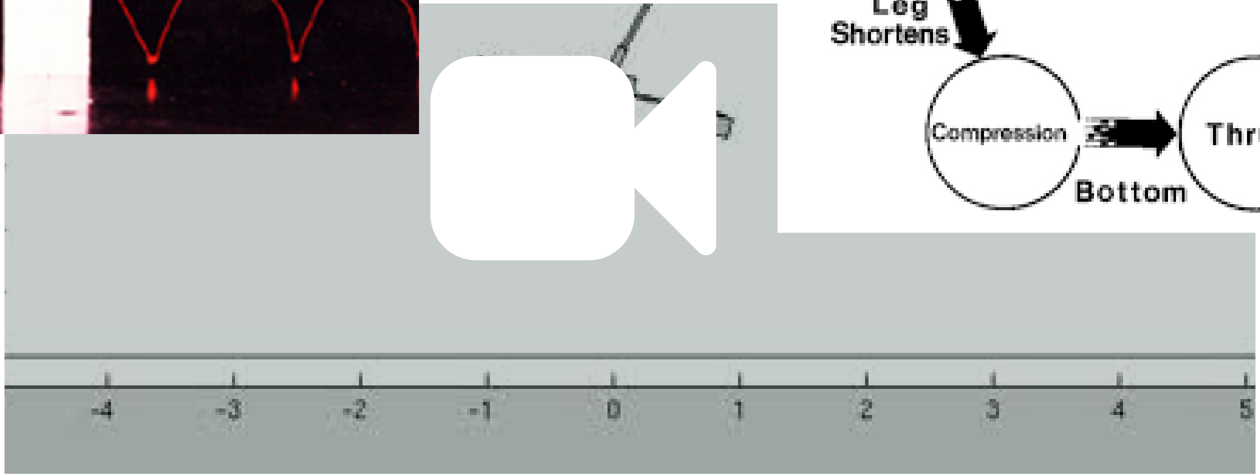


The MIT Leg Lab Hopping Robots

http://www.ai.mit.edu/projects/leglab/robots/robots.html





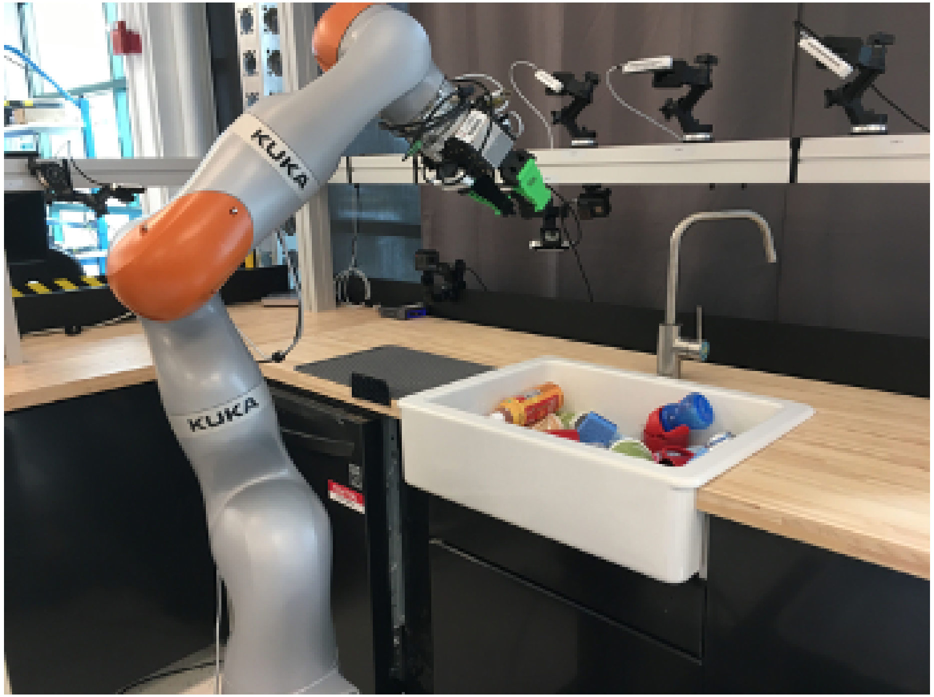


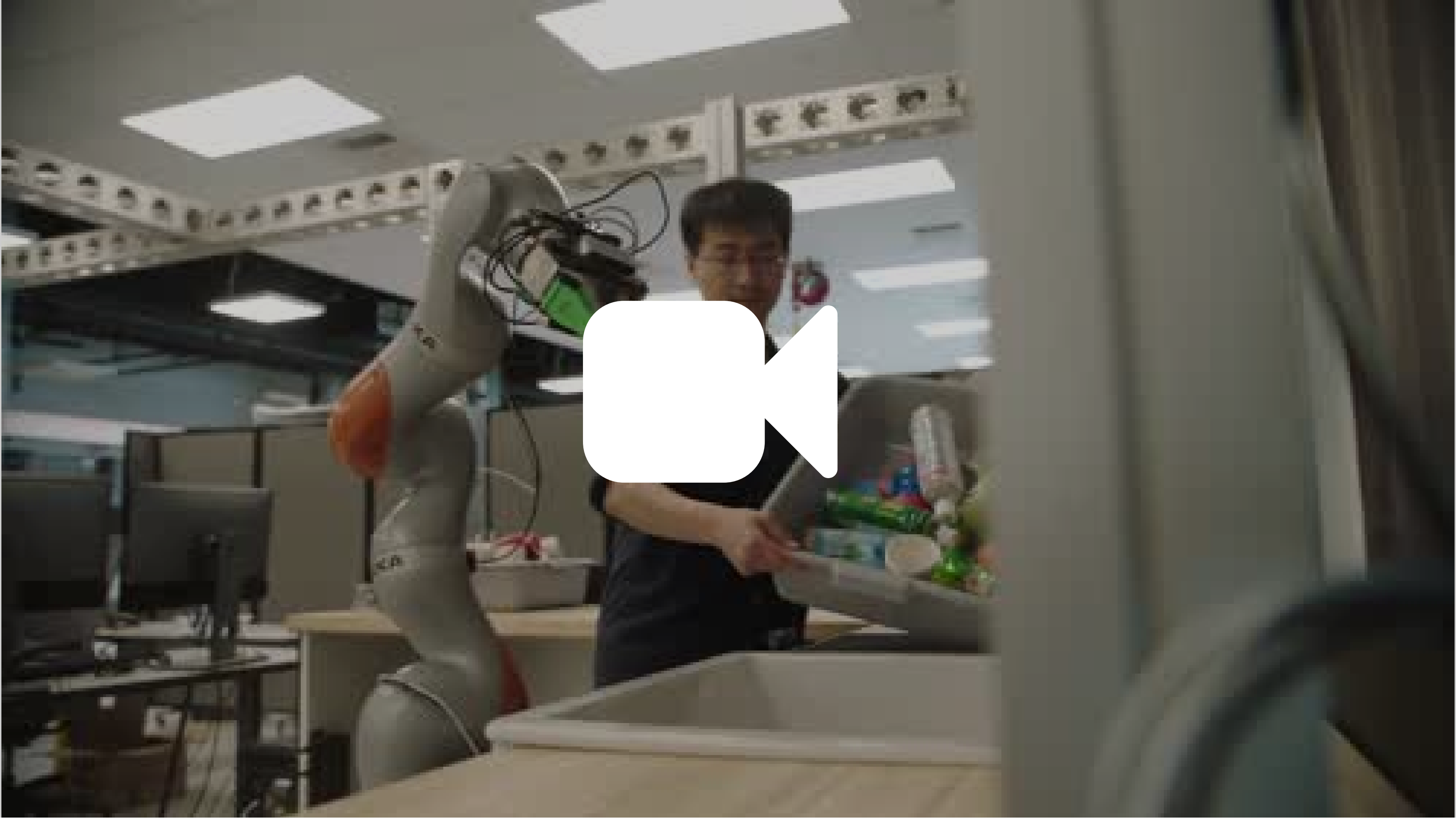
RL in the rare-event regime (for robotics)

Motivation

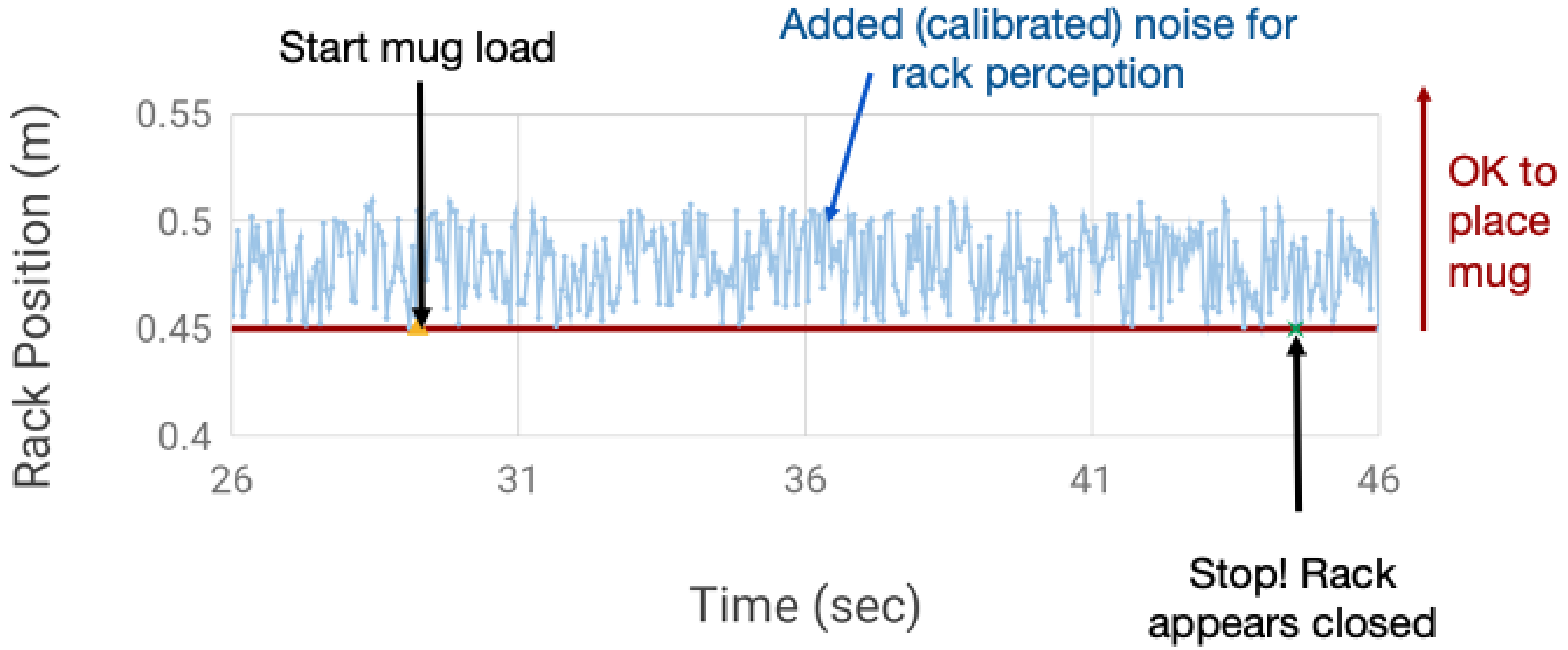
Consider the task of loading a dishwasher...

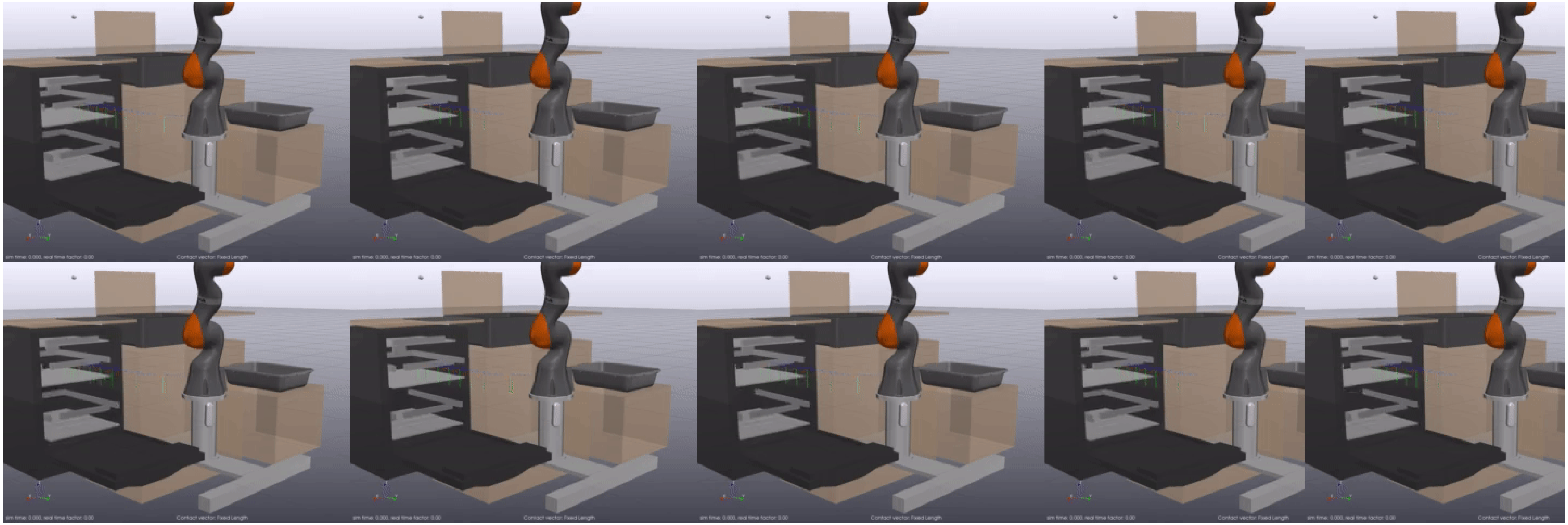
(One project I've been working on at TRI Robotics)





Finding subtle bugs





Estimating failure probability

for black-box, very high-dimensional, complex simulators

- adaptive multi-level splitting
- Hamiltonian MC
- parametric warping distributions via normalizing flows

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

Matthew O'Kelly[*1] Aman Sinha[*2] Hongseok Namkoong[*2]
John Duchi[2] Russ Tedrake[3]

[1]University of Pennsylvania
[2]Stanford University
[3]Massachusetts Institute of Technology

mokelly@seas.upenn.edu {amans,hnamk,jduchi}@stanford.edu russt@mit.edu

paper link

Neural Bridge Sampling for Evaluating Safety-Critical Autonomous Systems

Aman Sinha[*1] Matthew O'Kelly[*2] John Duchi[1] Russ Tedrake[3]

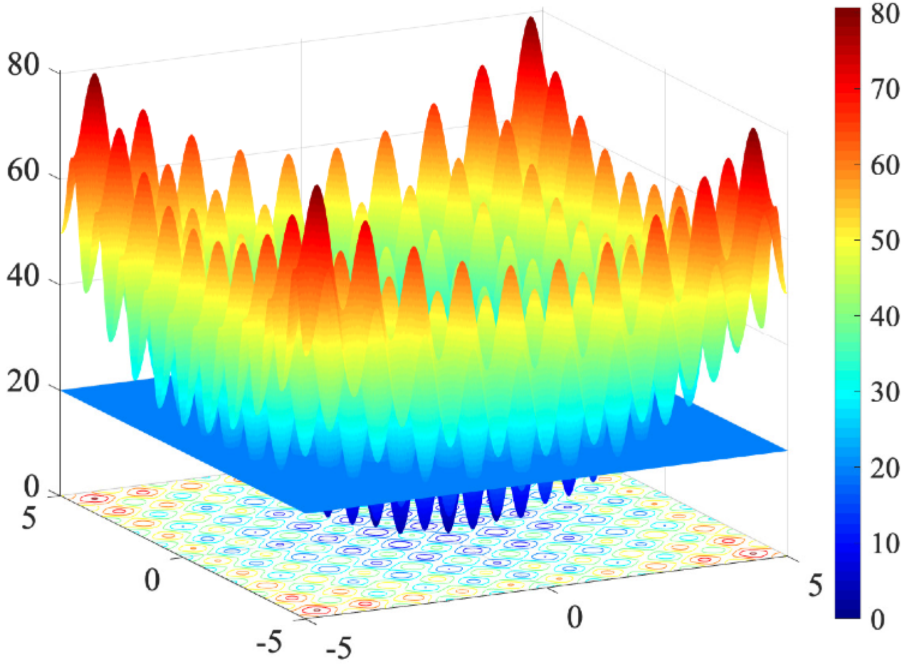
[1]Stanford University
[2]University of Pennsylvania
[3]Massachusetts Institute of Technology

amans@stanford.edu, mokelly@seas.upenn.edu, jduchi@stanford.edu, russt@mit.edu

paper link

Failure probability vs "falsification"

Falsification algorithms are not designed for *coverage*.



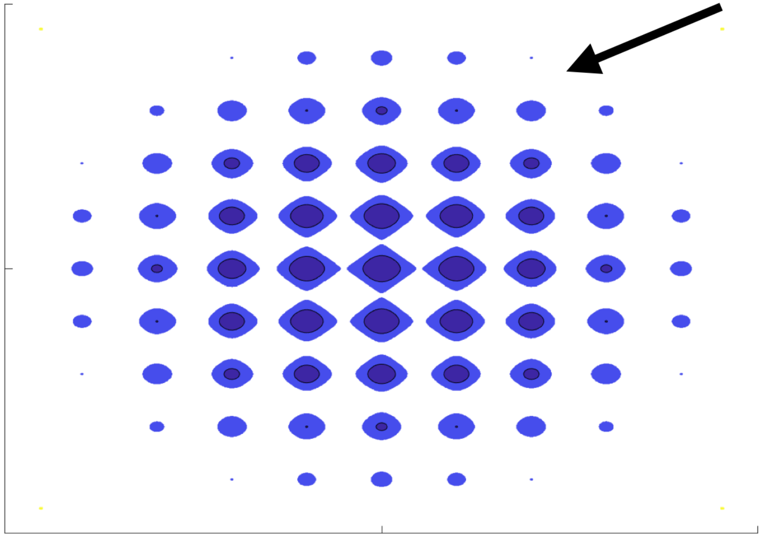
find $x < 20$

vs

estimate $p(x < 20)$

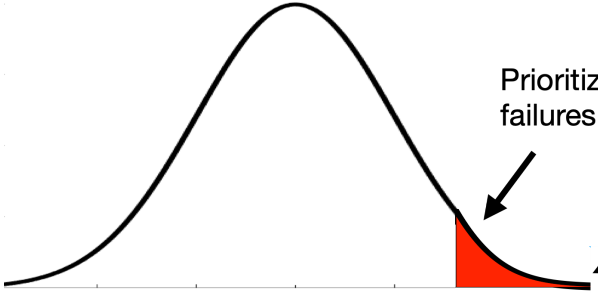
The risk-based framework

Don't just find one spot

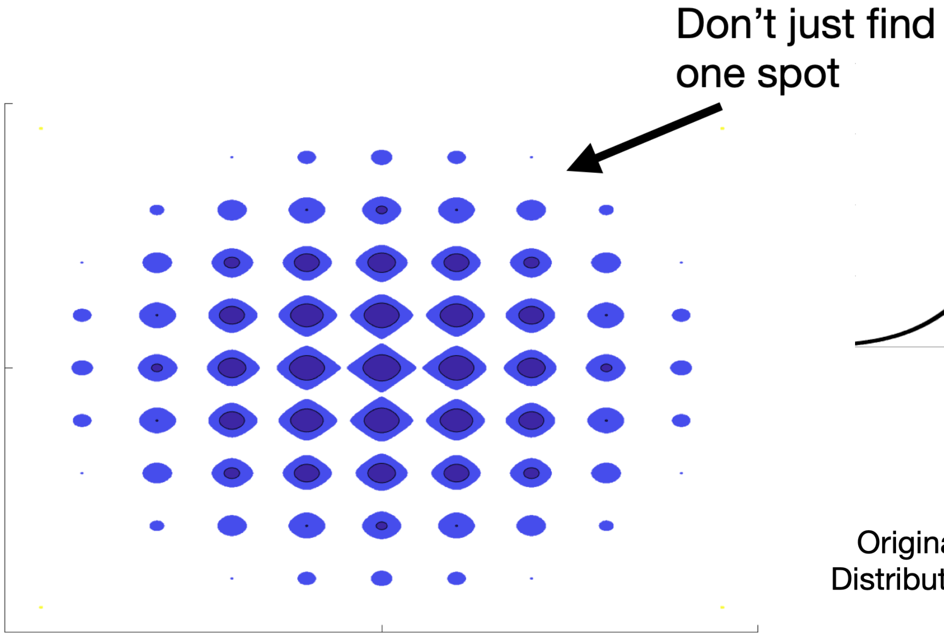


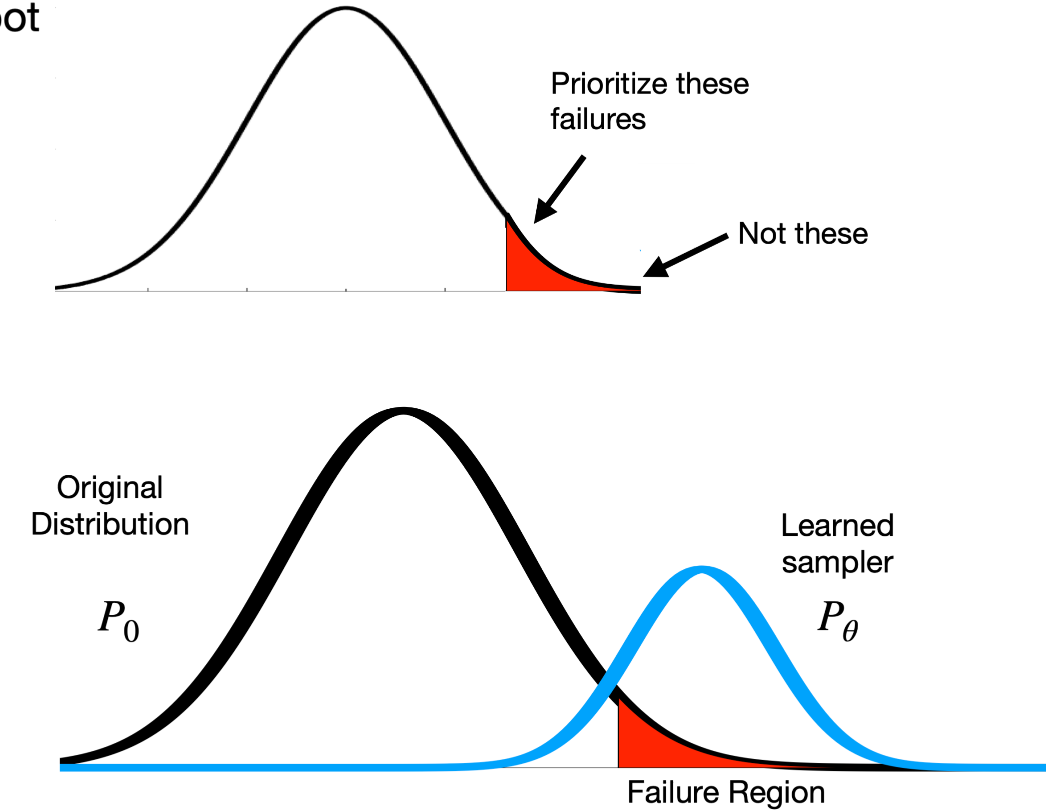
Prioritize these failures

Not these



The risk-based framework



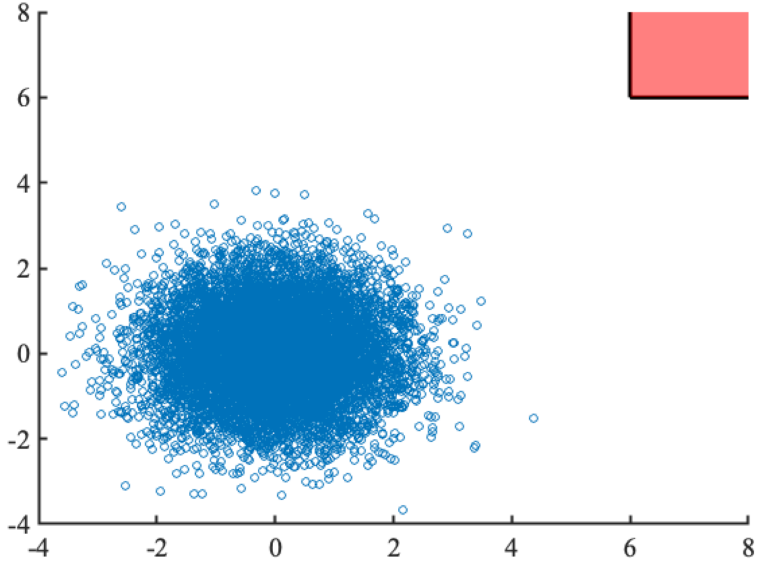


The risk-based framework

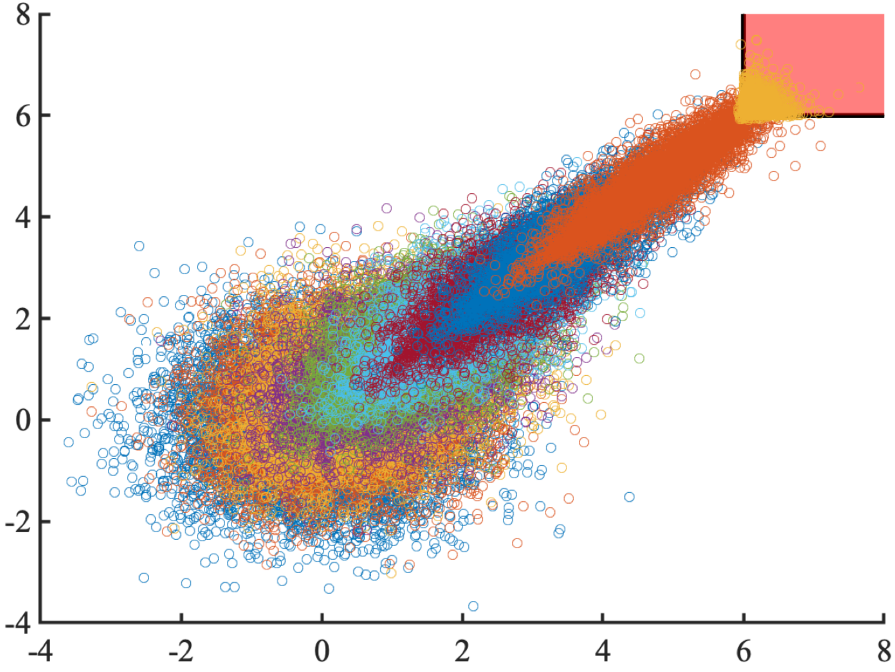
$$P_0 = \mathcal{N}(0, I)$$

$$p(\max(x_i) > 6)$$

Region of interest

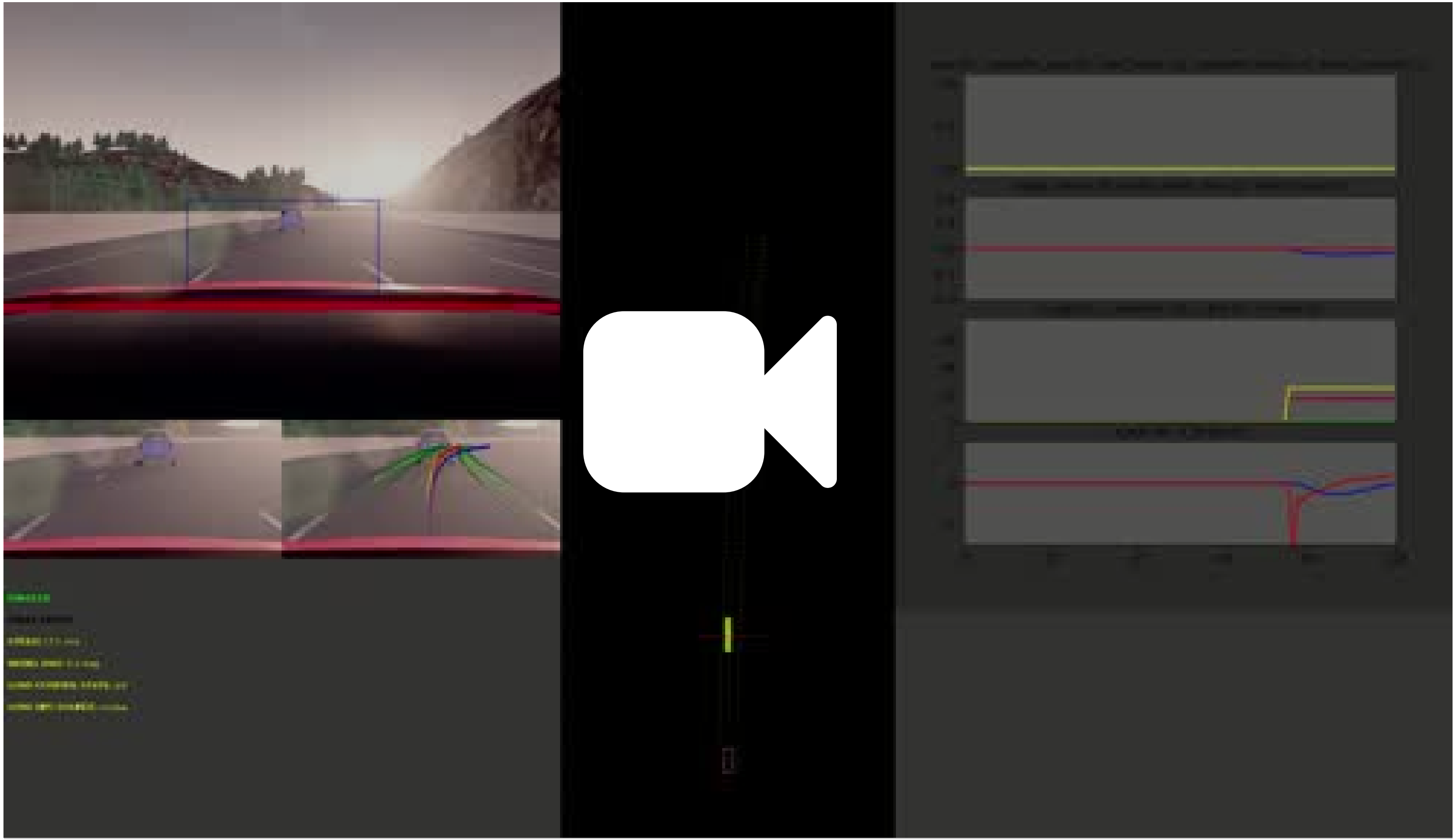


a smooth ladder of samplers



Finds surprisingly rare and diverse failures of the full comma.ai openpilot in the Carla simulator.

The primary authors have now created a startup: trustworthy ai



trustworthy ai

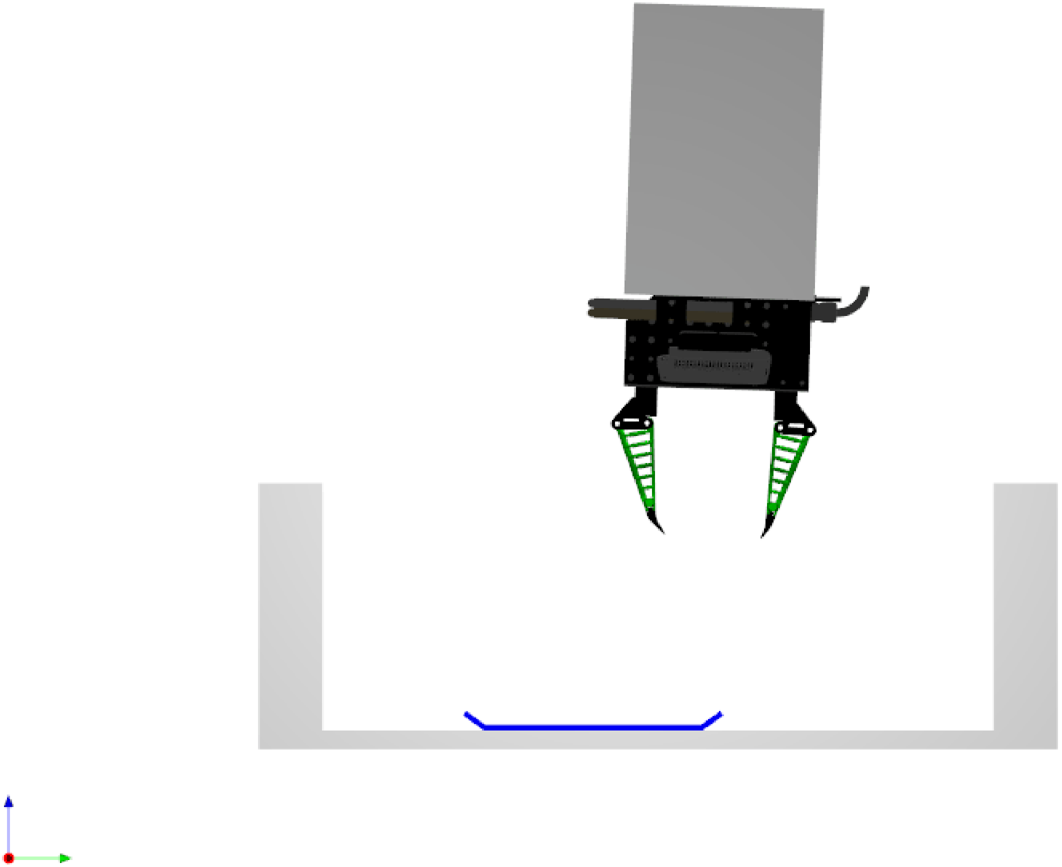
RL in the rare-event regime

Invites super interesting questions for RL / control.

- Convinced me that empirical risk estimation *might* actually to scale to high dimensional data.
- Plausible formulation of "safety" in difficult domains.

- Do the parameterized warping distributions enable more efficient importance-sampling RL?

Some specific robotics problem instances

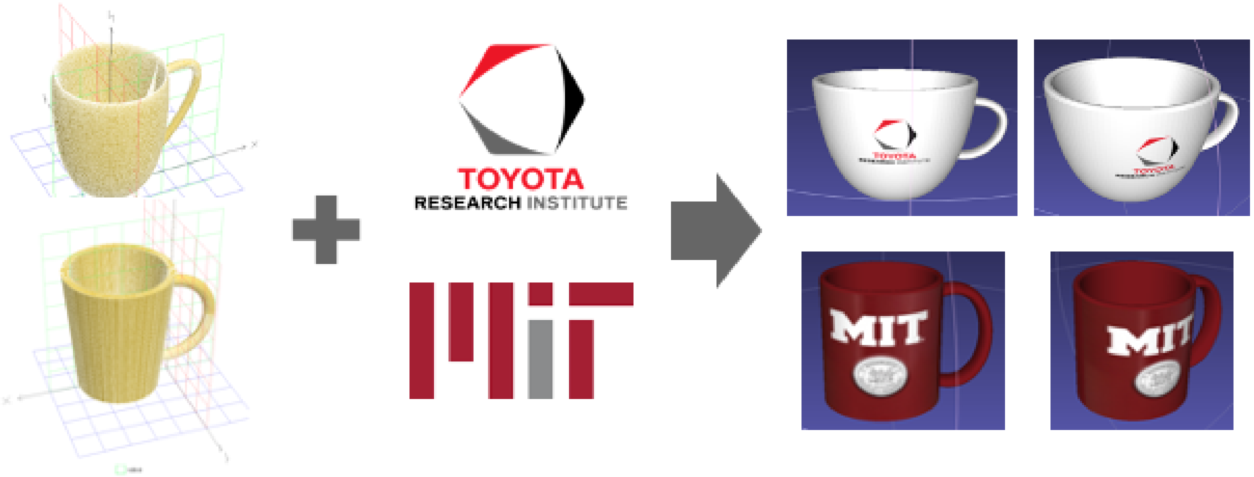


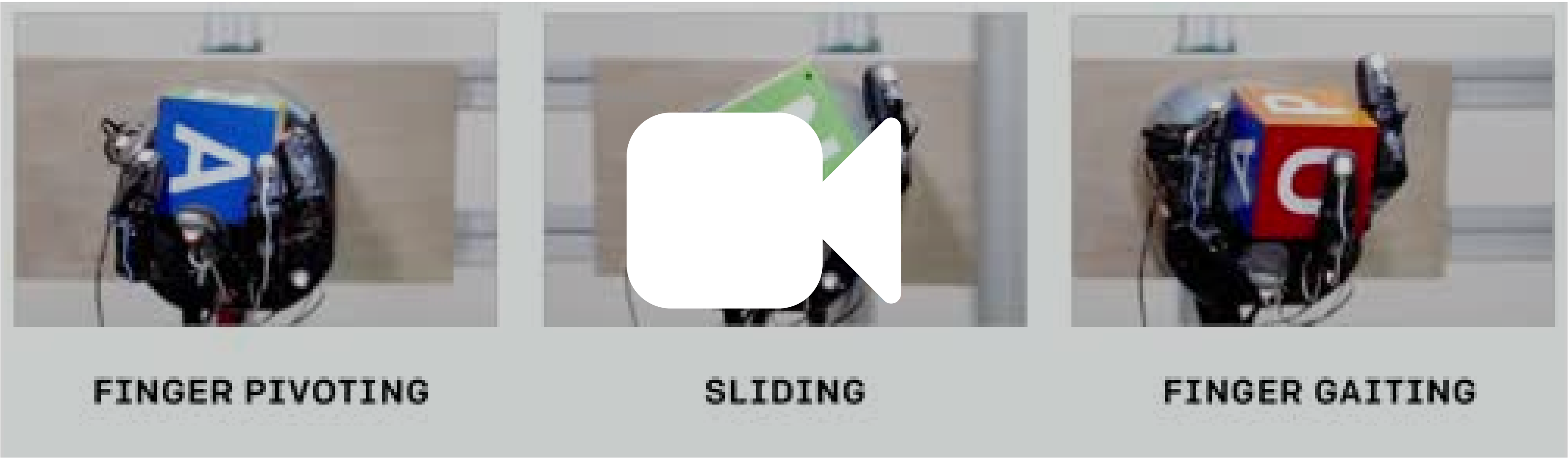
TOYOTA
RESEARCH INSTITUTE

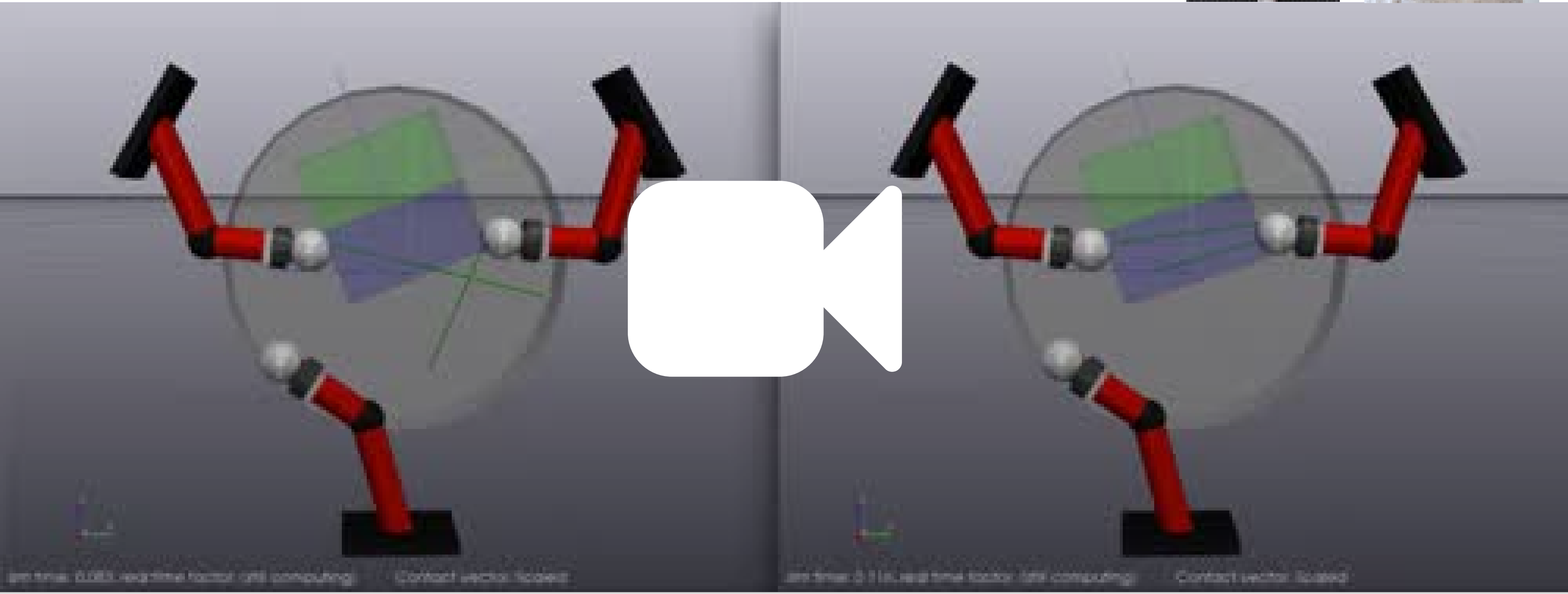
Distributional Robustness

We have parameterized simulations to study distributional robust / distribution shift.

- Parameterized environments (lighting conditions, etc)
- Carefully parameterized procedural mugs!
- ...




FINGER PIVOTING
SLIDING
FINGER GAITING



TOYOTA RESEARCH INSTITUTE

Releasing all of these as a part of my fall manipulation class at MIT (which is open and online).

https://people.csail.mit.edu/russt/uploads/iiwa.html

Summary

- How well is control working in robotics today?

- A few core quesitons/challenges
  - What role can simulation play?
  - Why/when does gradient-based policy search work?
  - Parameterizations/algorithms from control.
  - RL in the rare-event regime.
- Challenge problem instances
  - Plates, planar gripper, Raibert's hopper, onions, shoe laces...

https://groups.csail.mit.edu/locomotion/6-881-website/data/intro.html

Click here
(on this slide)

or from Ch.1 at
http://manipulation.mit.edu

https://groups.csail.mit.edu/locomotion/6-881-website/data/intro.html

Click here
(on this slide)

or from Ch.1 at
http://manipulation.mit.edu

You will need to rerun this cell if you restart the kernel, but it should be fast because the machine will already have drake installed.

```
import importlib
import sys
from urllib.request import urlretrieve

# Install drake.
if 'google.colab' in sys.modules and importlib.util.find_spec('pydrake') is None:
  version='20200901'
```

Then here
(in the colab window)