Deployable Robotics (Part II)

Russ Tedrake

Pablo’s billion dollar question: “**What will be the epistemology of deployable ML?**”

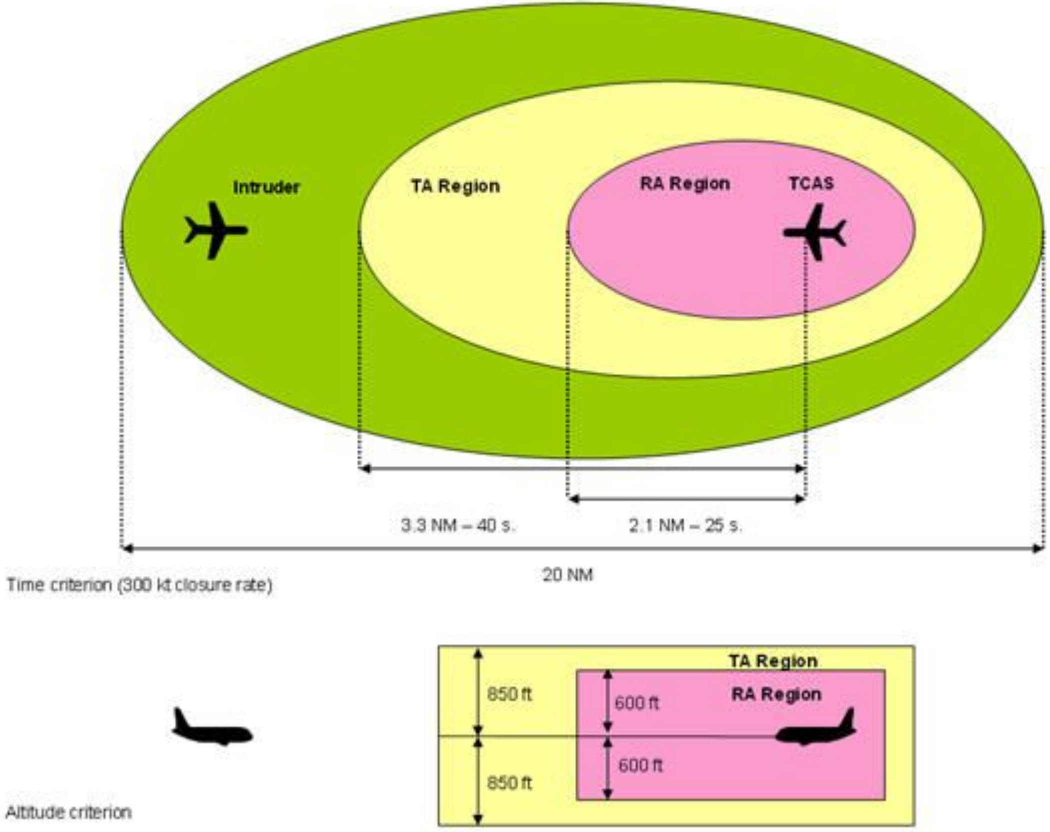
Verification & Validation

Some vocabulary:

- Desired outcomes described via **Requirements/Specifications**
- **Correctness** is the conformance of a system to its specification.
- **Verification** is the activity of establishing correctness.
- **Validation** is confirming that satisfying requirements achieved the intended results/performance.
- The whole bunch together makes can be used to form an "**assurance case**", which is the structured argument that we communicate about a system to convince a third party.

Example: Safety case for aircraft collision avoidance

ACAS II is an airborne avionics system designed to reduce the risk of mid-air collision. Its carriage by a majority of aircraft within Europe is mandatory.



Example of ACAS Protection Volume between 5000 and 10000 feet

The **[encounter] model**... from an analysis of encounters collected during 1998 and 2000 from European radar data.

The logic risk ratios reported here are computed ***assuming that all other aspects of the system operate as intended***: the surveillance of intruders is perfect, and pilots react to all resolution advisories (RAs) and with an ideal response.

... using an '**event tree**': a logical diagram that combines the relevant factors to calculate a risk of collision for the whole system. … probabilities for the base level events were estimated.

The logic risk ratios quoted above (and others calculated for various non-standard pilot responses) were ***combined with the probabilities of other system events***, using the event tree, to obtain risk ratios relevant to the operation of the total ACAS system.

5.2.1 *Monte Carlo approach*

5.2.1.1 As presented in Figure 5.1, the approach [WP-1 142] relies on using 'Monte Carlo' simulations. It consists in conducting a very large number of simulations and modifying for each simulation the initial encounters in order to take into account the uncertainties of surveillance data.

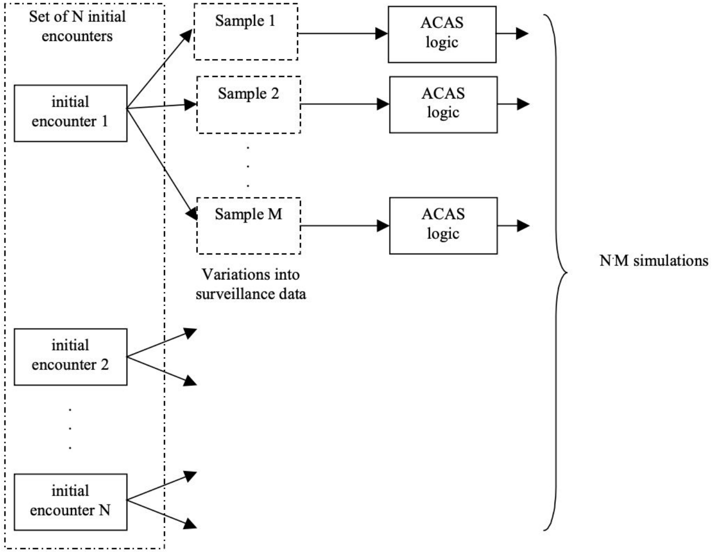


Figure 5.1: Monte Carlo simulations

5.2.1.2 Due to computer constraints, the number of initial encounters is restricted to $N = 10{,}732$. For each initial encounter, the number of variations into surveillance data is $M = 50$.

from *Final Report on Studies on the Safety of ACAS II in Europe* [ACAS/ACASA/02-014]

How do we build an **assurance case** for closed-loop systems with learning/perception/planning in the loop?

Last time

In controls (polytopic/ellipsoidal, etc)

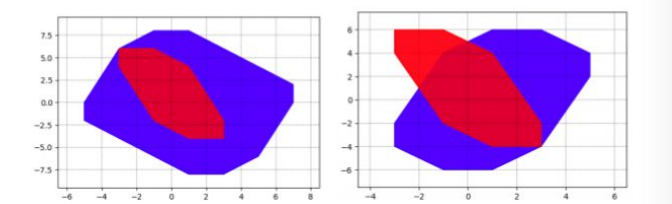
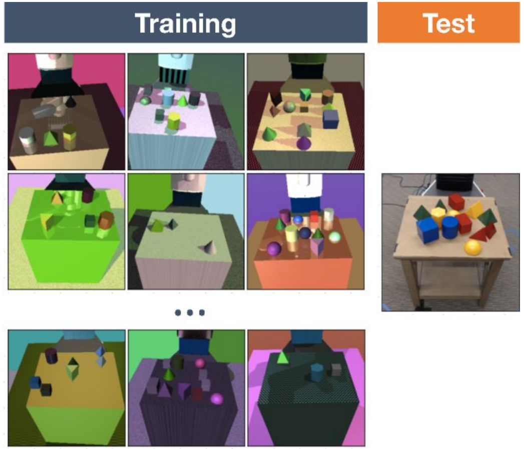


Fig. 1. Example 1: Zonotope Containment Problem: [left] $\mathbb{Z}_l \subseteq \mathbb{Z}_r$, [Right] $\mathbb{Z}_l \nsubseteq \mathbb{Z}_r^*$, where the last column of G_r is dropped.

Domain randomization in reinforcement learning



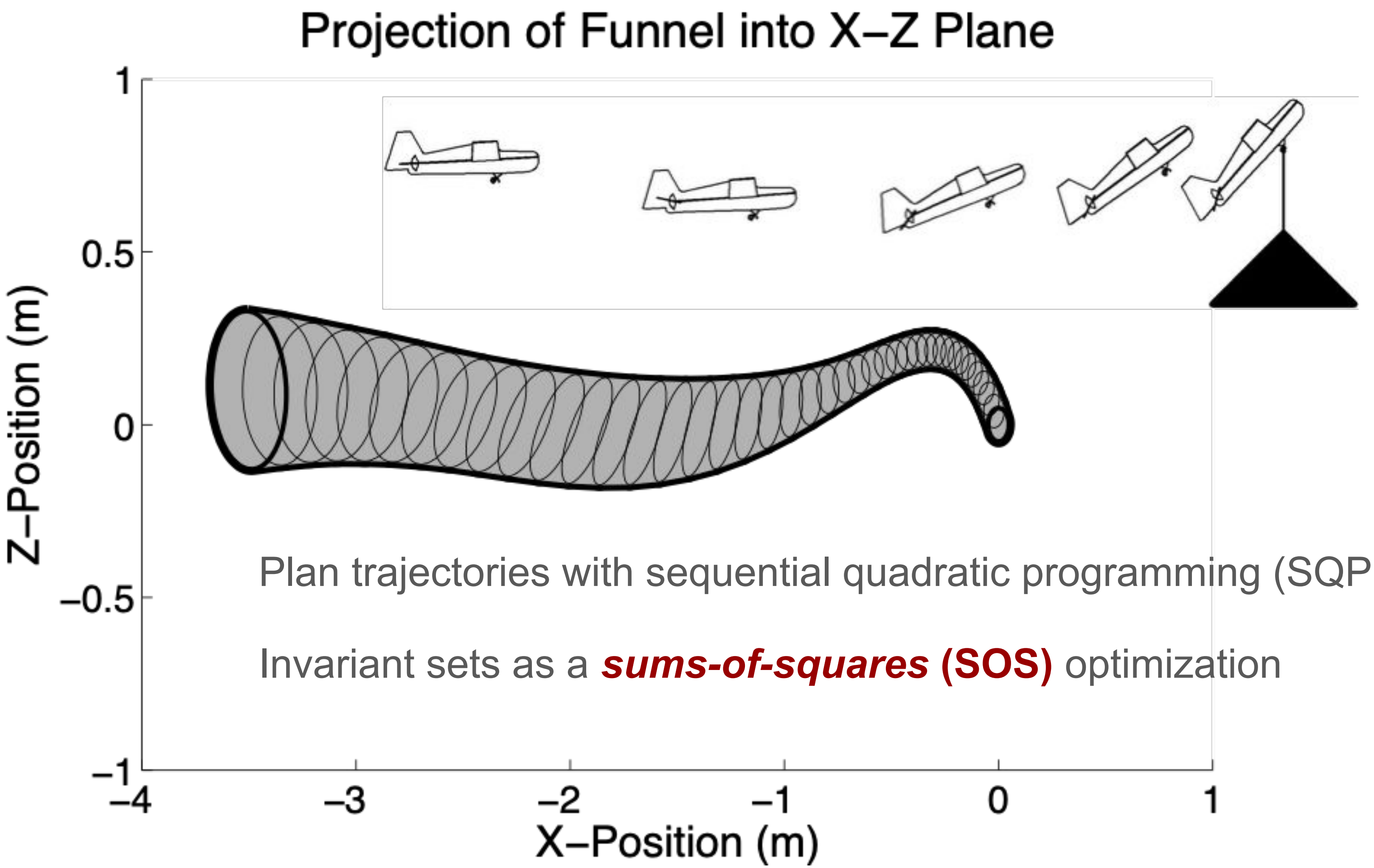
27 Feb 2018 | 16:48 GMT

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**

Developing autonomous systems in the real world.



Plan trajectories with sequential quadratic programming (SQP)

Invariant sets as a ***sums-of-squares* (SOS)** optimization

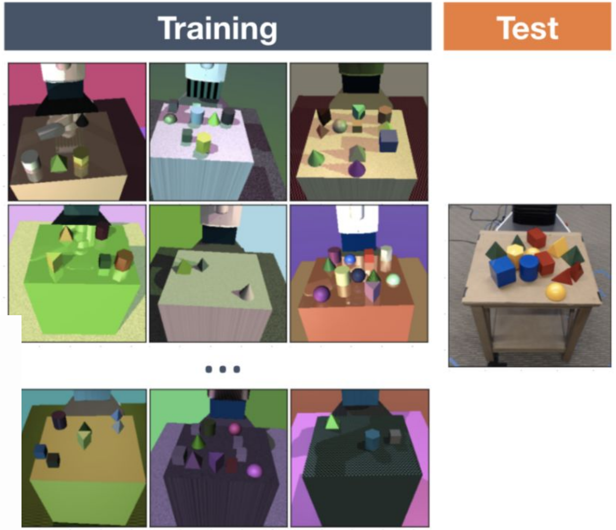
Lessons from Robust Control

Often criticized: sacrifice performance to guarantee robustness.

Domain Randomization in RL

In the original form of DR (Tobin et al, 2017; Sadeghi et al. 2016), each randomization parameter ξ_i is bounded by an interval, $\xi_i \in [\xi_i^{\text{low}}, \xi_i^{\text{high}}], i = 1, \ldots, N$ and each parameter is uniformly sampled within the range.

- Position, shape, and color of objects,
- Material texture,
- Lighting condition,
- Random noise added to images,
- Position, orientation, and field of view of the camera in the simulator.



https://lilianweng.github.io/lil-log/2019/05/05/domain-randomization.html

Physical dynamics in the simulator can also be randomized (Peng et al. 2018). Studies have showed that a *recurrent* policy can adapt to different physical dynamics including the partially observable reality. A set of physical dynamics features include but are not limited to:

- Mass and dimensions of objects,
- Mass and dimensions of robot bodies,
- Damping, kp, friction of the joints,
- Gains for the PID controller (P term),
- Joint limit,
- Action delay,
- Observation noise.

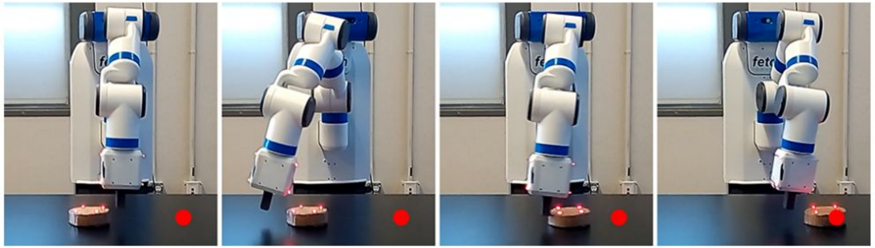
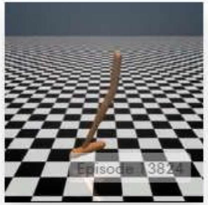


Fig. 1. A recurrent neural network policy trained for a pushing task in simulation is deployed directly on a Fetch Robotics arm. The red marker indicates the target location for the puck.

https://lilianweng.github.io/lil-log/2019/05/05/domain-randomization.html

HalfCheetah-v0
Make a 2D cheetah robot run.

Swimmer-v0
Make a 2D robot swim.

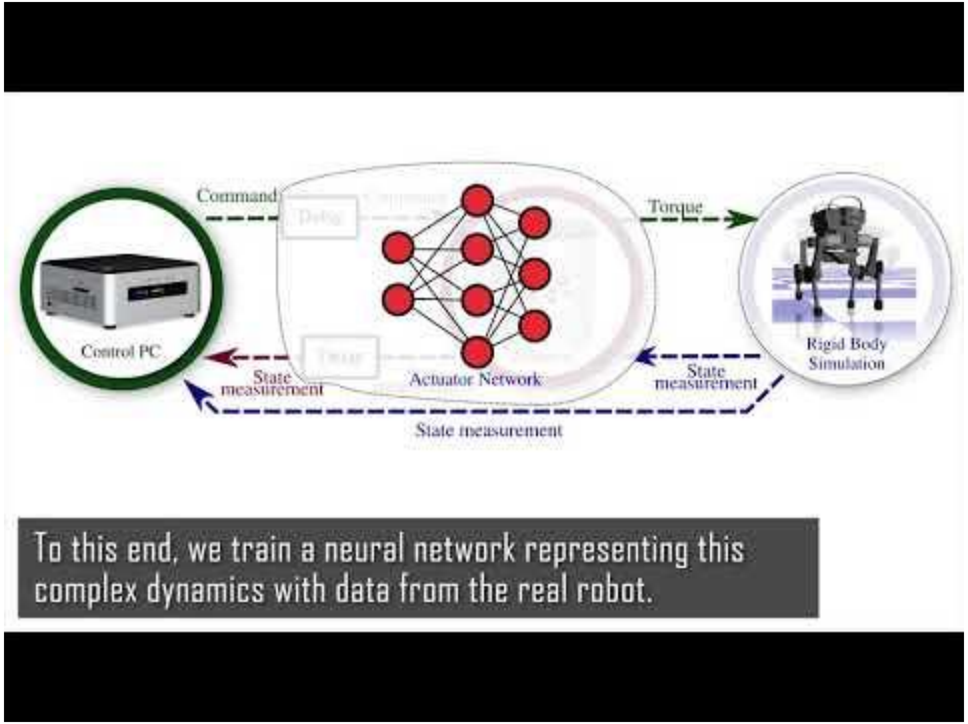


Hopper-v0
Make a 2D robot hop.

Walker2d-v0
Make a 2D robot walk.

Ant-v0
Make a 3D four-legged robot walk.

Humanoid-v0
Make a 3D two-legged robot walk.



OpenAI Gym vs. ETH ANYmal results

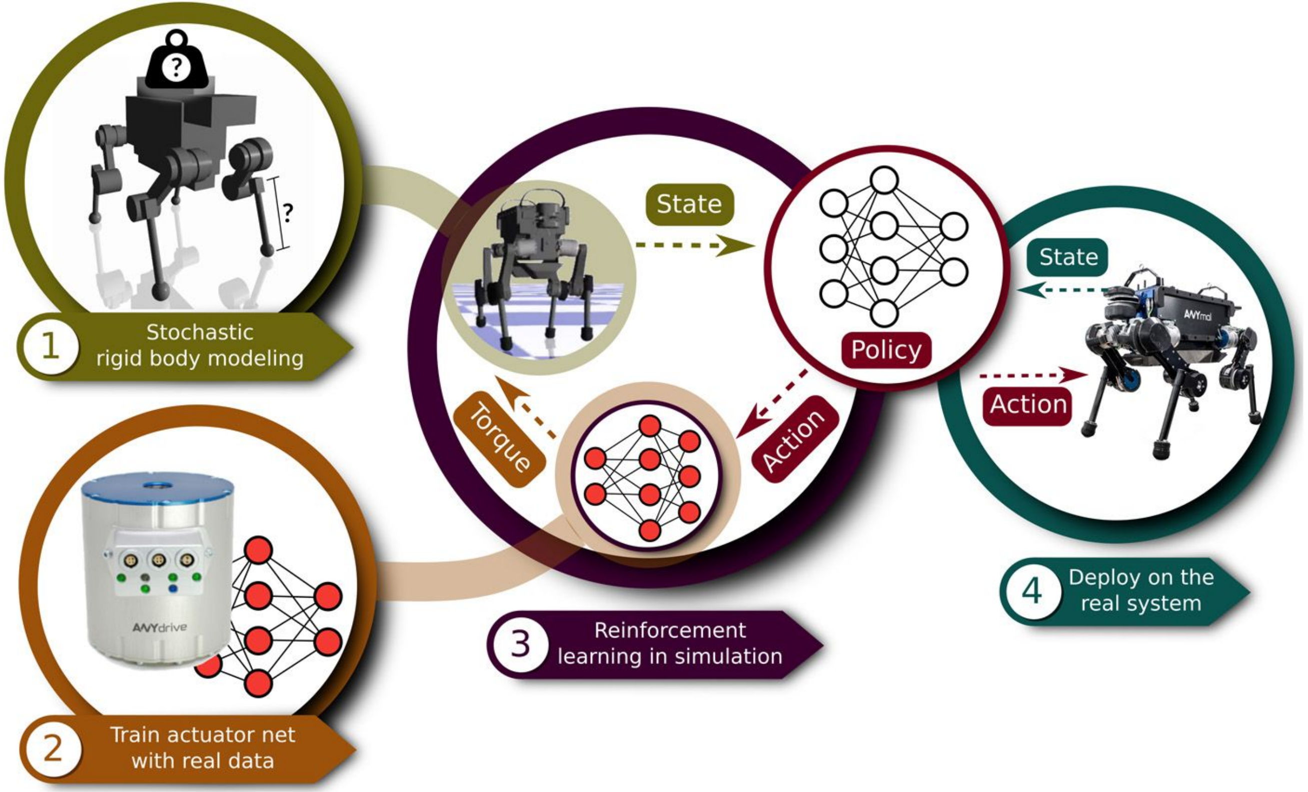
Learning agile and dynamic motor skills for legged robots

Jemin Hwangbo[1,*], Joonho Lee[1], Alexey Dosovitskiy[2], Dario Bellicoso[1], Vassilios Tsounis[1], Vladlen Koltun[3] and Marco Hutt...
+ See all authors and affiliations

Science Robotics 16 Jan 2019:
Vol. 4, Issue 26, eaau5872
DOI: 10.1126/scirobotics.aau5872

“learned actuator dynamics effectively reduce the reality gap, whereas stochastic modeling guides the policy to be sufficiently conservative.

The center of mass positions, the masses of links, and joint positions were randomized by adding a noise sampled from $U(-2, 2)$ cm, $U(-15, 15)$%, and $U(-2, 2)$ cm, respectively.”



I think the really interesting question are for systems with ***both*** **rich uncertainty** + **non-trivial tasks/dynamics**

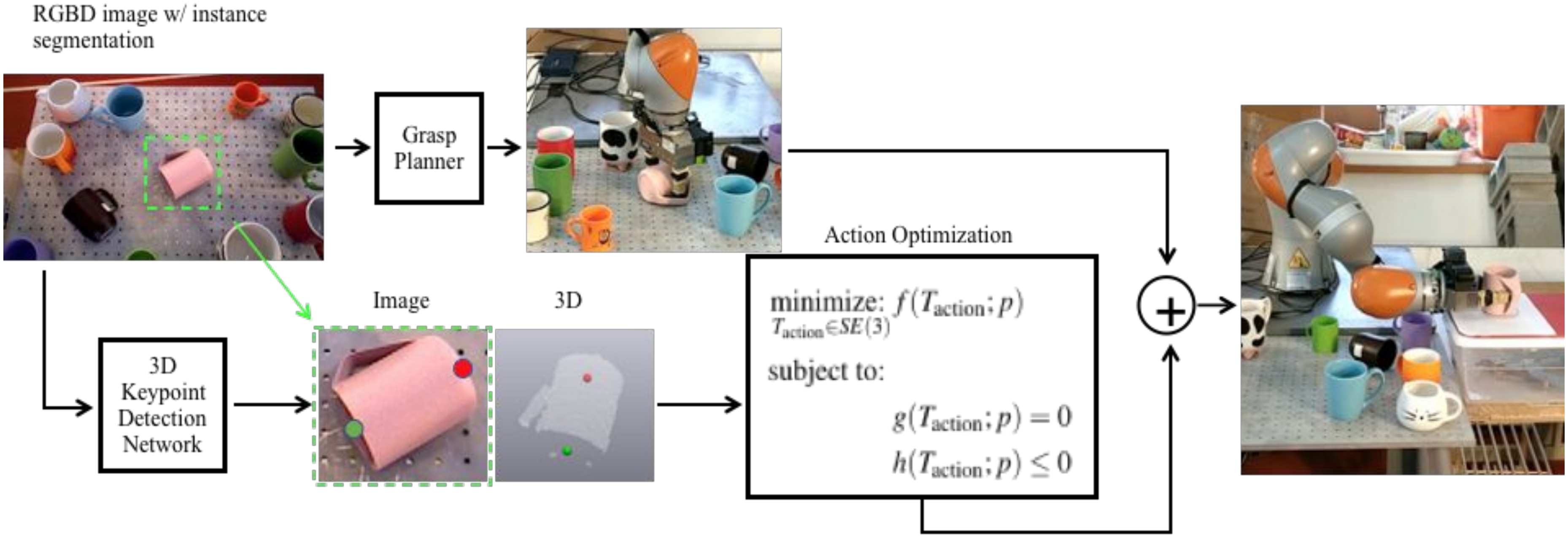

10

Proposed problem formulation:

“Class-general” manipulation.

kPAM pipeline

No template model or pose appears in this pipeline.




4x speed

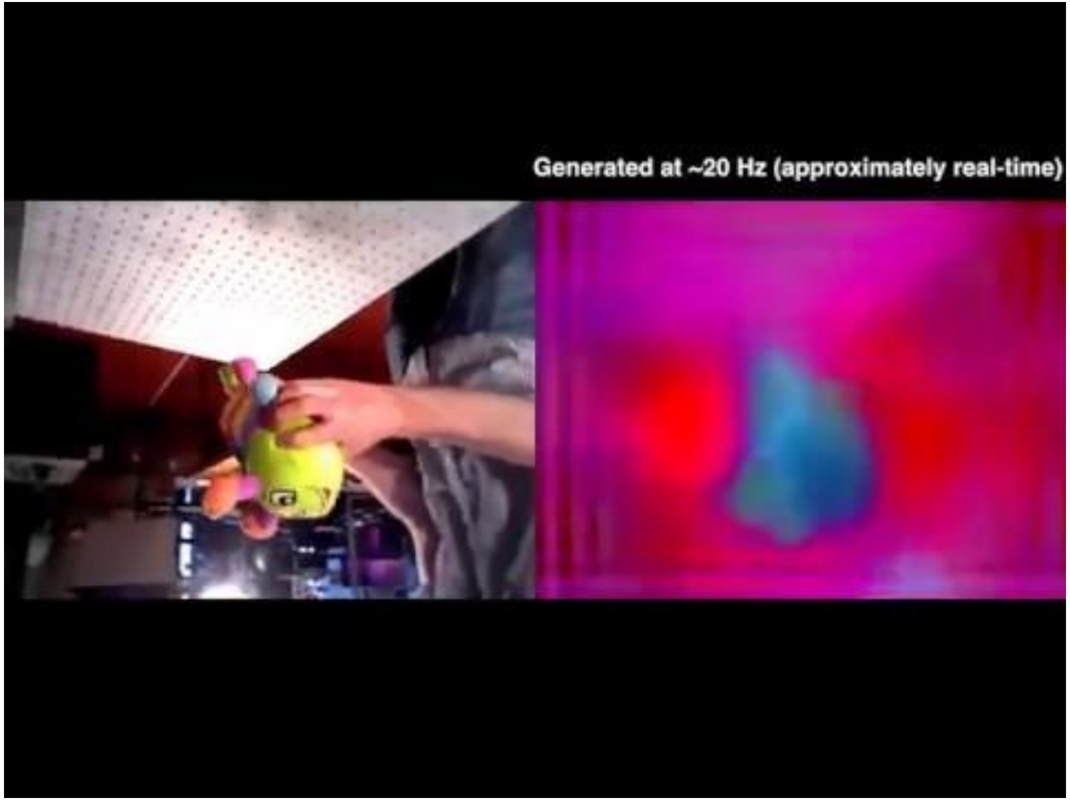

4x speed

Includes large neural net for perception (ResNet)

Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation

Peter R. Florence*, Lucas Manuelli*, Russ Tedrake
CSAIL, Massachusetts Institute of Technology
{peteflo,manuelli,russt}@csail.mit.edu
These authors contributed equally to this work.

And a recurrent network for control (LSTM)




KUKA
KUKA

Requirements/Specifications

In controls (polytopic/ellipsoidal, etc)

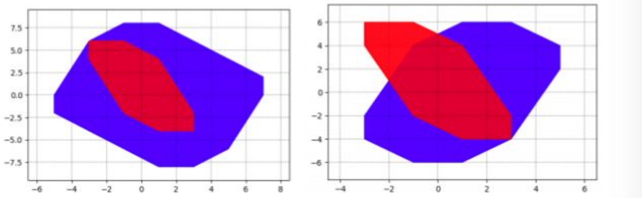
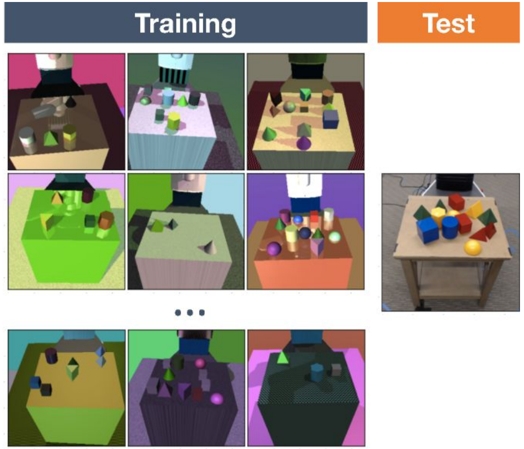


Fig. 1. Example 1: Zonotope Containment Problem: [left] $\mathbb{Z}_l \subseteq \mathbb{Z}_r$, [Right] $\mathbb{Z}_l \nsubseteq \mathbb{Z}_r^*$, where the last column of G_r is dropped.

Domain randomization in reinforcement learning



27 Feb 2018 | 16:48 GMT

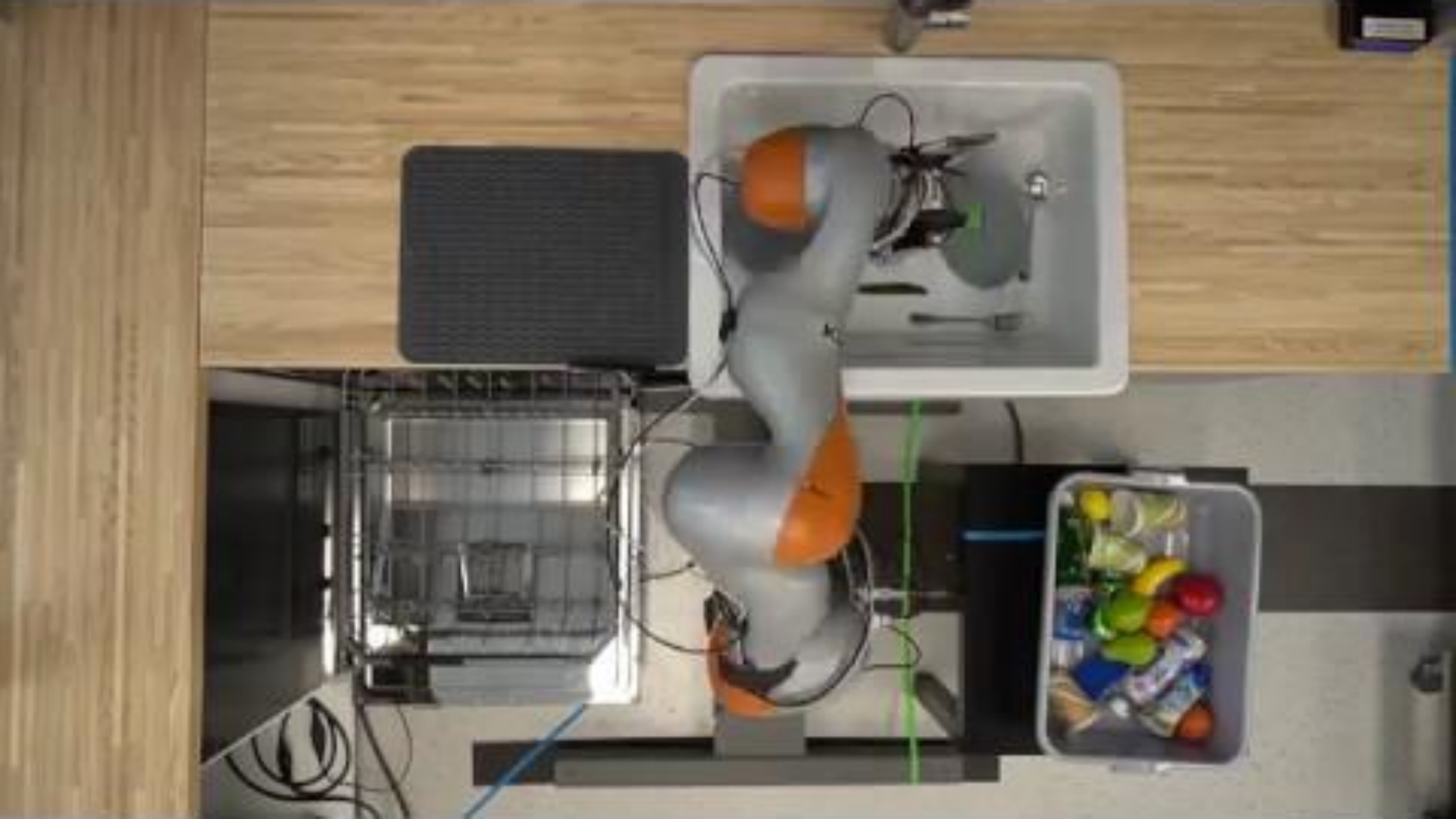
Creating Driving Tests for Self-Driving Cars

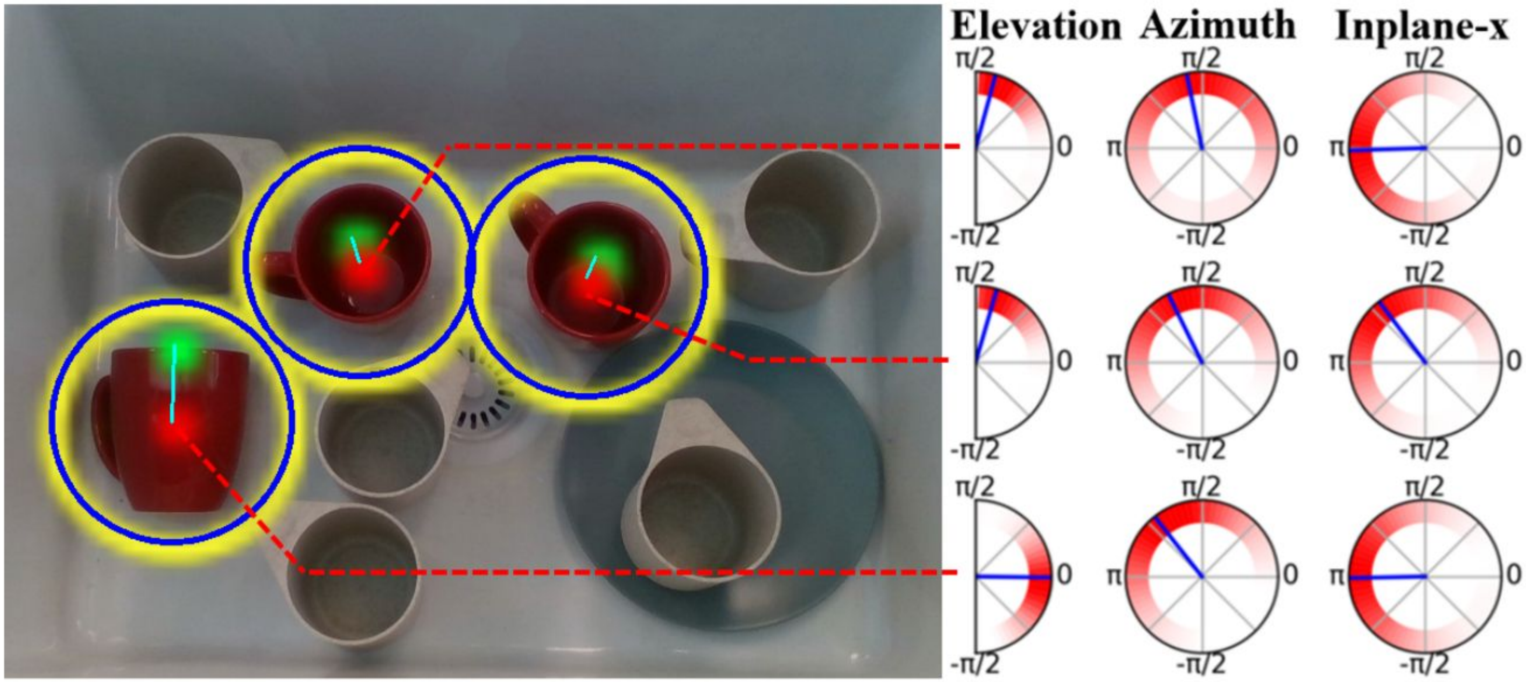
Volvo-backed Zenuity wants to prove that autonomous vehicles can drive more safely than humans

By Erik Coelingh and Jonas Nilsson

Developing autonomous systems in the real world.

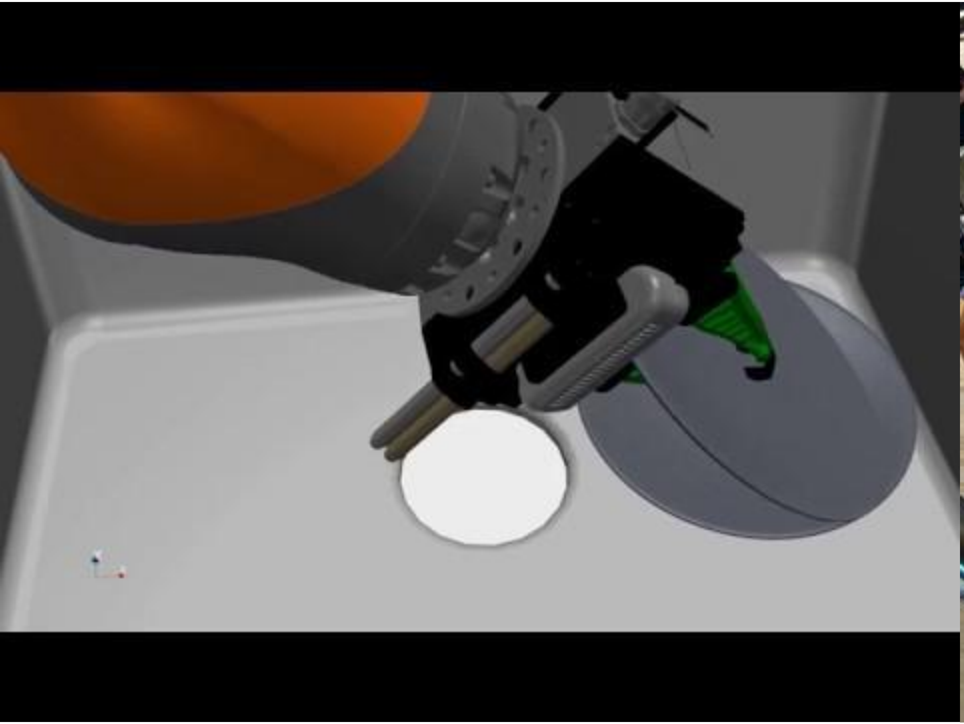
TOYOTA
RESEARCH INSTITUTE



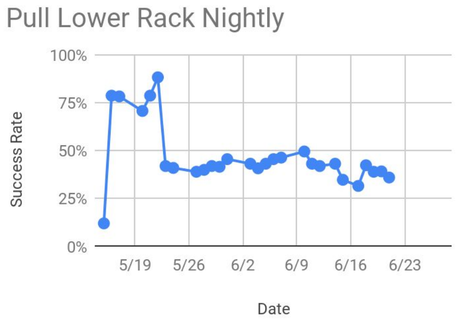


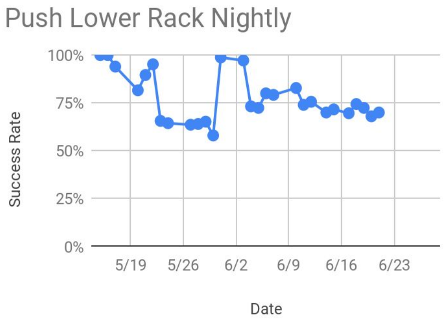
KOSNet: A Unified Keypoint, Orientation and Scale Network for Probabilistic 6D Pose Estimation

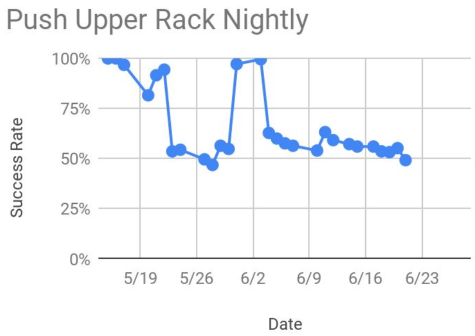
Kunimatsu Hashimoto*, Duy-Nguyen Ta*, Eric Cousineau and Russ Tedrake
*These authors contributed equally to this work.

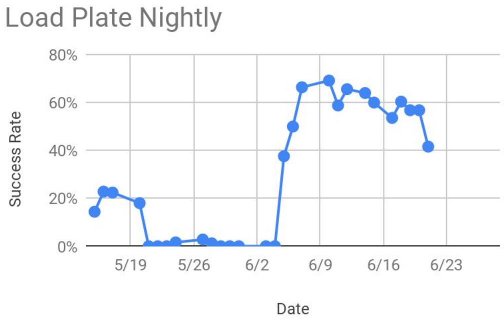
 

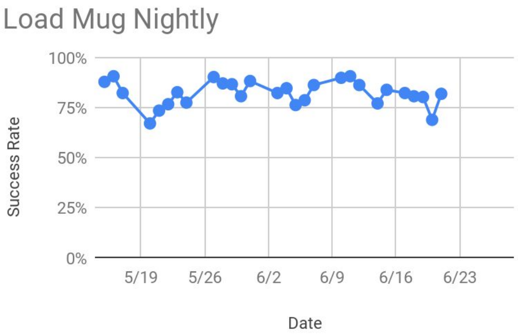
Monte Carlo falsification

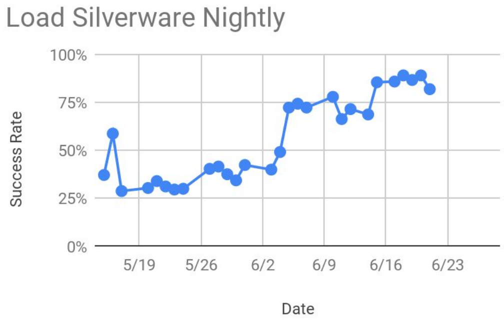

Pull Lower Rack Nightly
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date

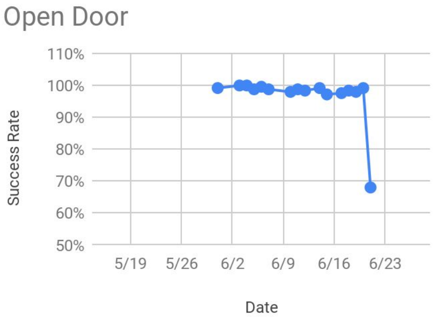

Push Lower Rack Nightly
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date

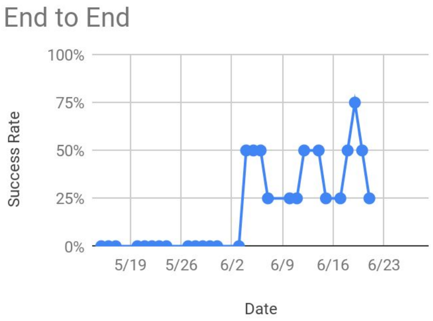

Push Upper Rack Nightly
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date


Load Plate Nightly
80%
60%
40%
20%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date


Load Mug Nightly
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date


Load Silverware Nightly
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date


Open Door
110%
100%
90%
80%
70%
60%
50%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date


End to End
100%
75%
50%
25%
0%
Success Rate
5/19
5/26
6/2
6/9
6/16
6/23
Date

Scenario description files

Parameters, initial conditions, and noise described as exact values or distributions

```
_DishwareConstants:
  - &dish_input sink
  - &mug_anywhere
      base_frame: *dish_input
      translation: !UniformVector
        min: [-0.10, -0.20, 0.10]
        max: [0.10, 0.20, 0.30]
      rotation_rpy_deg: !UniformRotation {}
  - &plate_anywhere
```

Scenario description files

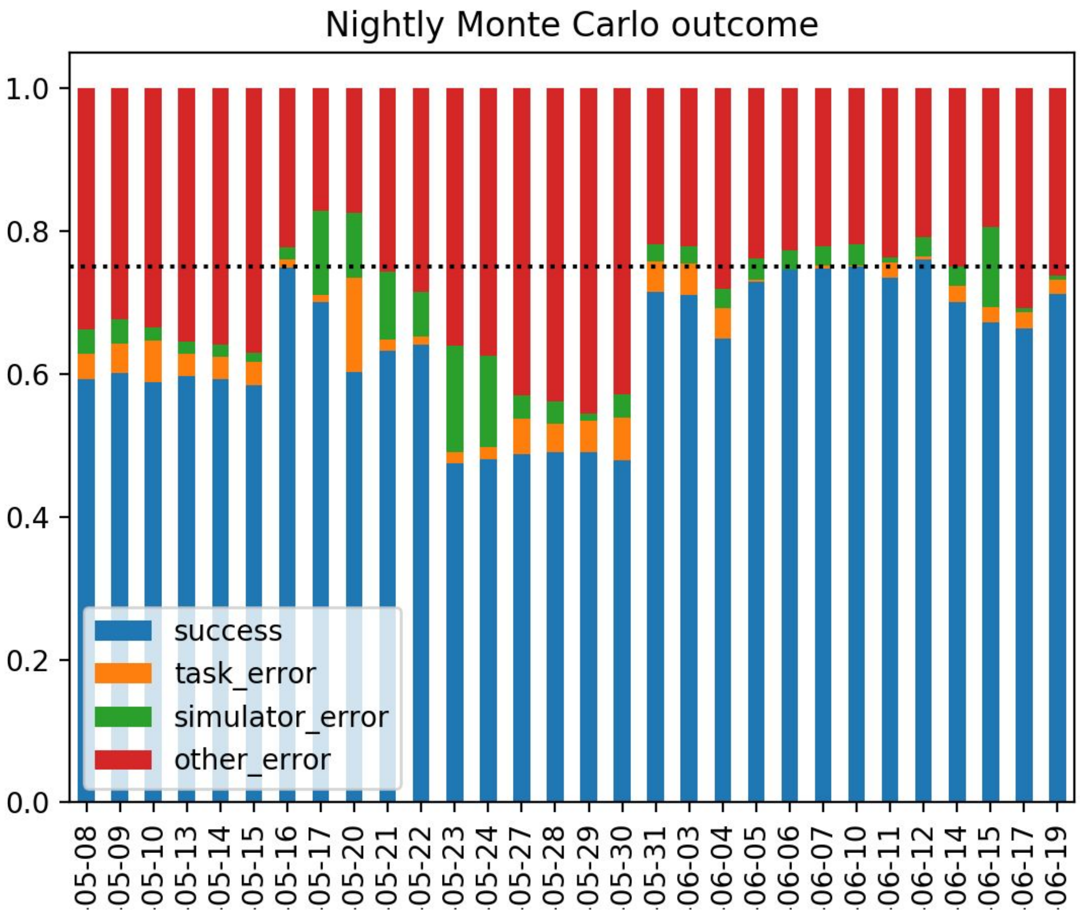
Success criteria specified as [constraints on systems]()

Can compose into complex diagrams, and be used for synthesis

```
&mug_placement_position
frame: *mug_link
base_frame: dishwasher_upper_rack
translation_lower: [-0.20, -0.20, 0.056]
translation_upper: [0.20, 0.20, 0.057]
```

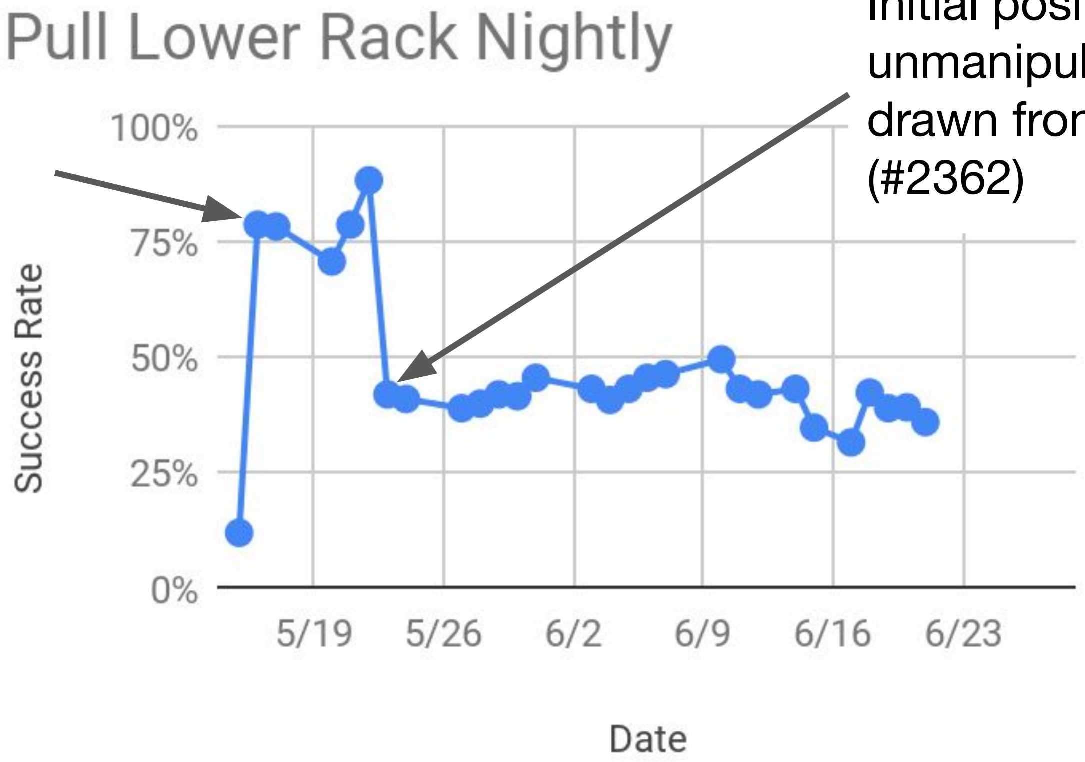
```yaml
TestMugLoadAcrossSink:
  station_name: central_square
  iiwa_q0: *iiwa_anywhere
  dishwashers:
    dishwasher:
      door_angle_deg: *door_open_deg
      silverware_rack_position: *silverware_rack_in
      upper_rack_position: *upper_rack_out
      lower_rack_position: *lower_rack_anywhere
      position_sensor_noise: *default_dishwasher_position_sensor_noise
  use_wrist_camera: True
  items:
  -
    kind: &mug
      role: corelle_livingware_11oz_mug_red
      model: &mug_model models/mug/corelle_livingware_11oz_mug_red.sdf
      link_name: &mug_link corelle_livingware_11oz_mug_red
    X_initial: *mug_anywhere
  dish_task: load_dish_test
  pose_constraints:
  -
    &mug_placement_position
    frame: *mug_link
    base_frame: dishwasher_upper_rack
    translation_lower: [-0.20, -0.20, 0.056]
    translation_upper: [0.20, 0.20, 0.057]
  vec_dir_constraints:
  -
    &mug_placement_orientation
    frame: *mug_link
    vectors_in_base_frame:
      - [0, 0, -1]
    vectors_in_frame:
      - [0, 0, 1]
    tolerance_deg_lower: [0]
    tolerance_deg_upper: [10]
```

This seems to be a theme in companies deploying AI…
requirements/specifications are authored as a set of objectives/constraints on a list of scenarios.



First you find bugs in your simulator!

Switched to a motion planning scheme that's less sensitive to rack initial position (#2304)

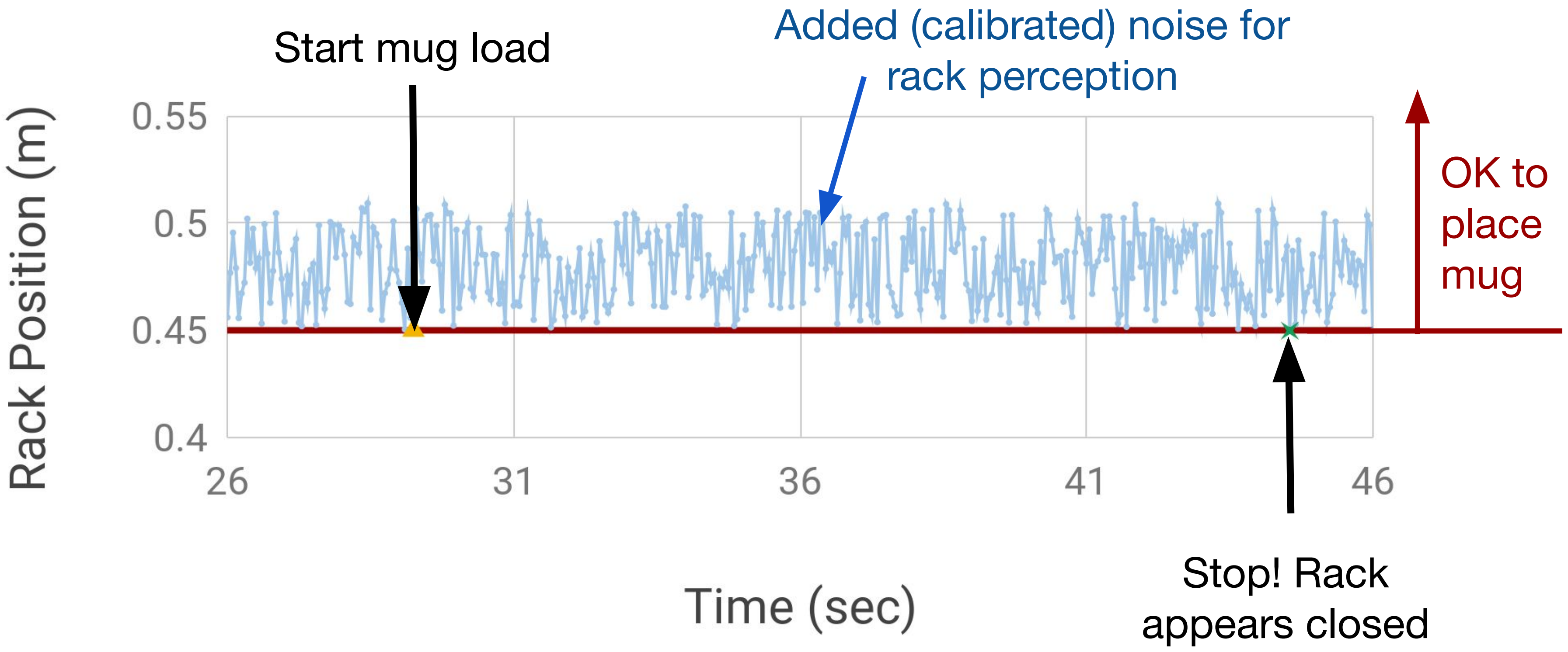


Initial positions of the unmanipulated racks are drawn from MC instead of 0 (#2362)

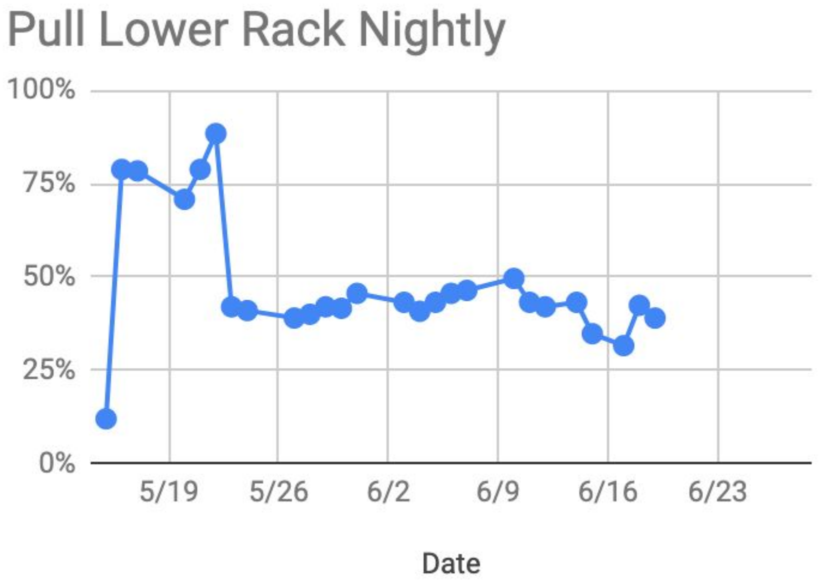
Finding subtle bugs

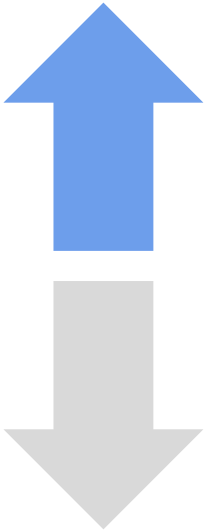
Finding subtle bugs

Finding subtle bugs



Falsification algorithms



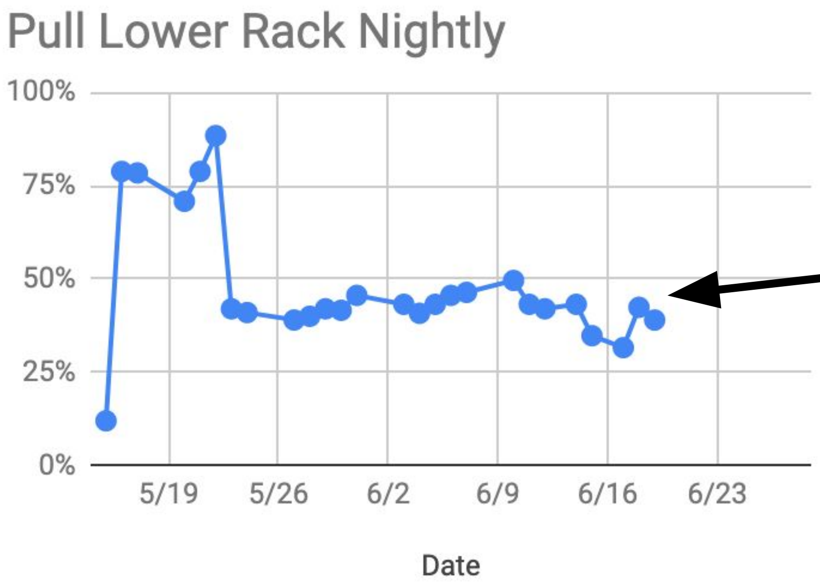


Improve robustness / fix bugs

Increase test randomness / scope

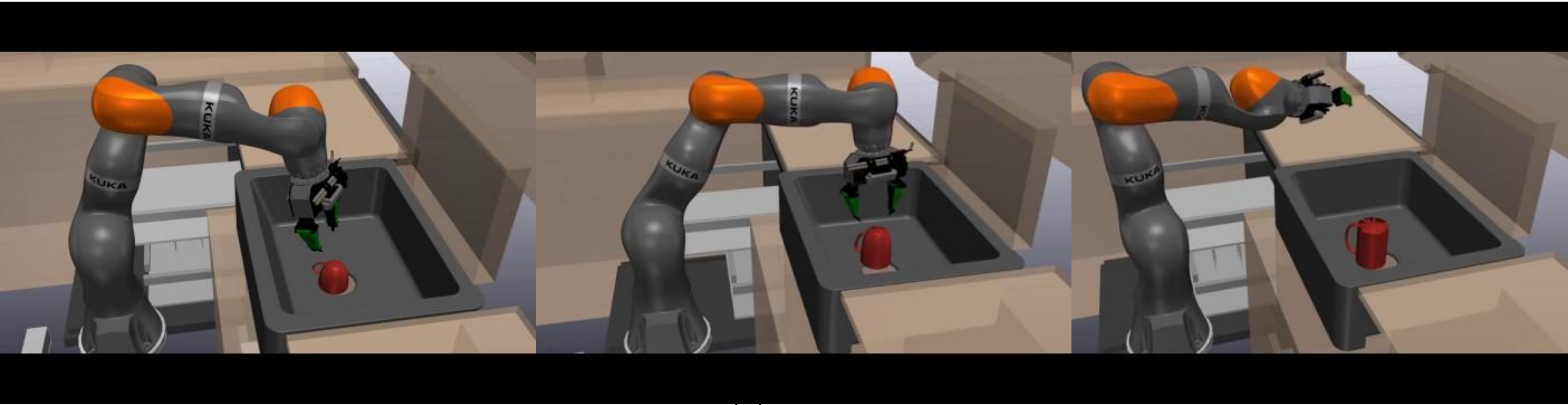
naive Monte Carlo has been sufficient (so far)

Sim vs Real

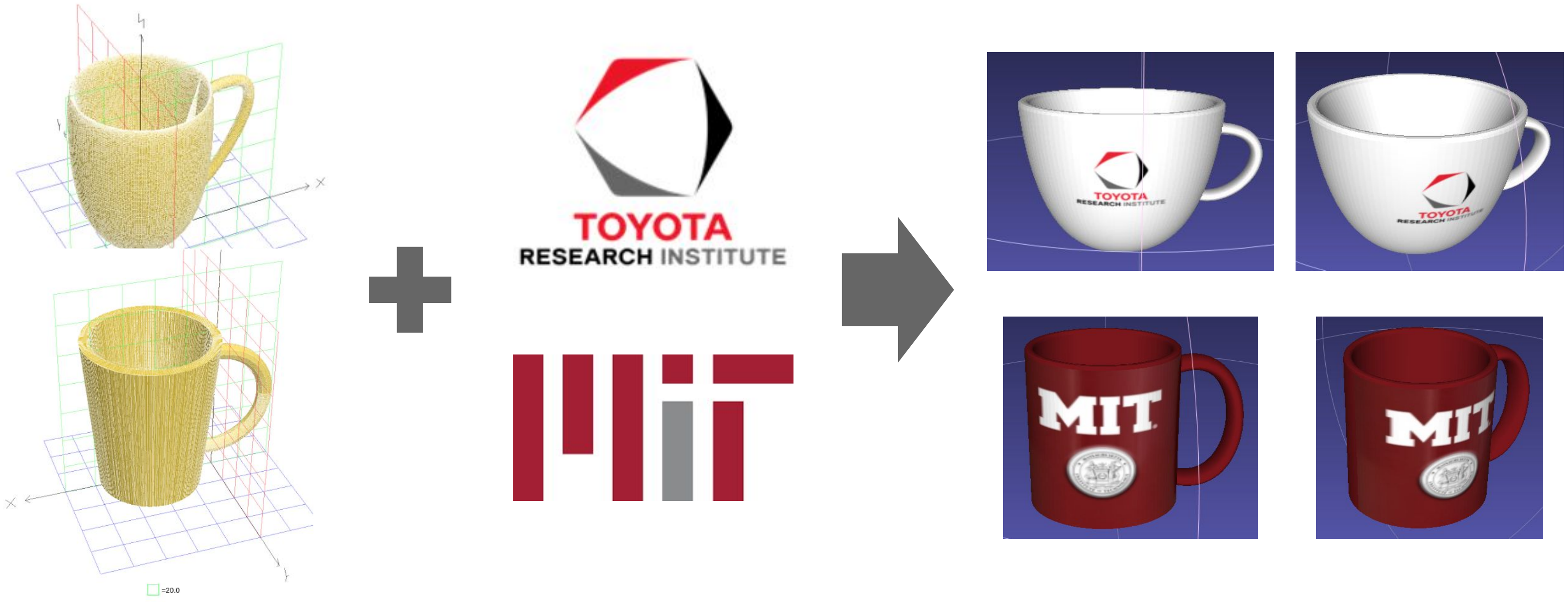


Made simulation tests ***more difficult*** than the real-world

Procedural dishes

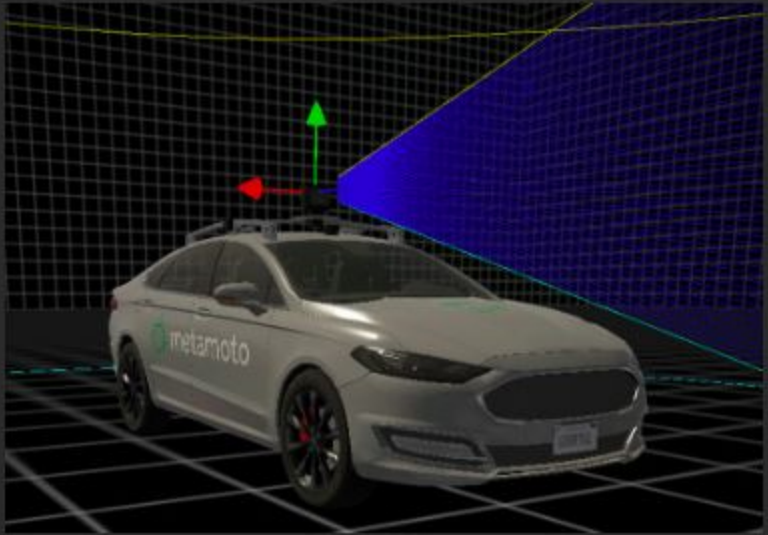


Procedural dishes



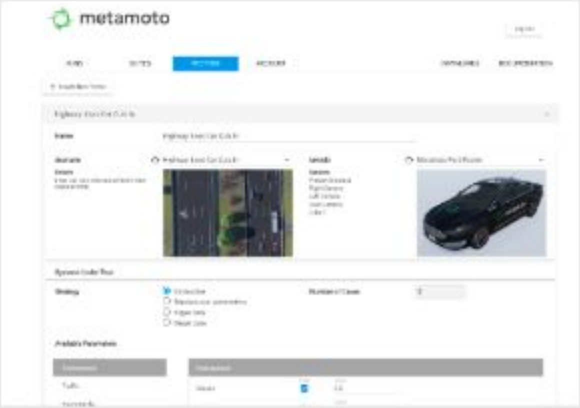
Very analogous to autonomous driving

Build your scenarios inside virtual scenes. Parameterize anything with the click of a button. Scenarios contain the behaviors of the ego vehicle, sensor configurations, traffic, pedestrians, and more.



http://metamoto.com

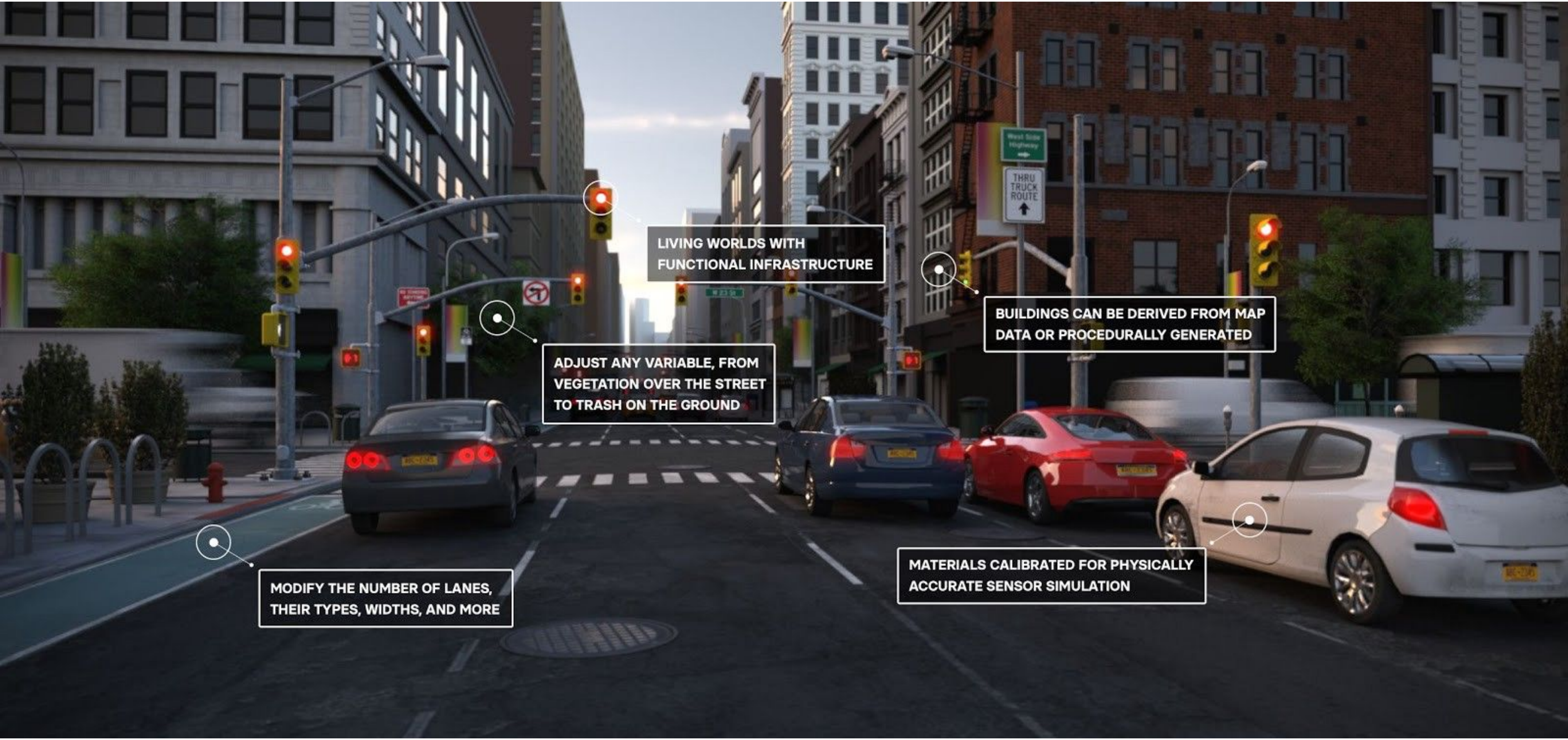
Schedule and run simulations across a spectrum of scenarios (e.g. unique edge cases), systems under test, and then ranges of environmental and hardware parameters. Run Monte Carlo training and regression testing exercises.



http://metamoto.com

Debug and replay simulations and comprehensively assess how your vehicle software performed. Dig into your software stack to get to the heart of the matter faster.

http://metamoto.com



http://paralleldomain.com

More advanced falsification via nonlinear black-box optimization and rare event simulation.

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

Nonlinear “black-box” optimization can be applied at scale; finds “rare” events

- E.g. CMA-ES (covariance matrix adaptation evolution strategy)

“Adversarial training” -- by engineers or fine-tuning

But performance bounds are only empirical -- and often weak.

Still requires a sufficiently rich simulation.

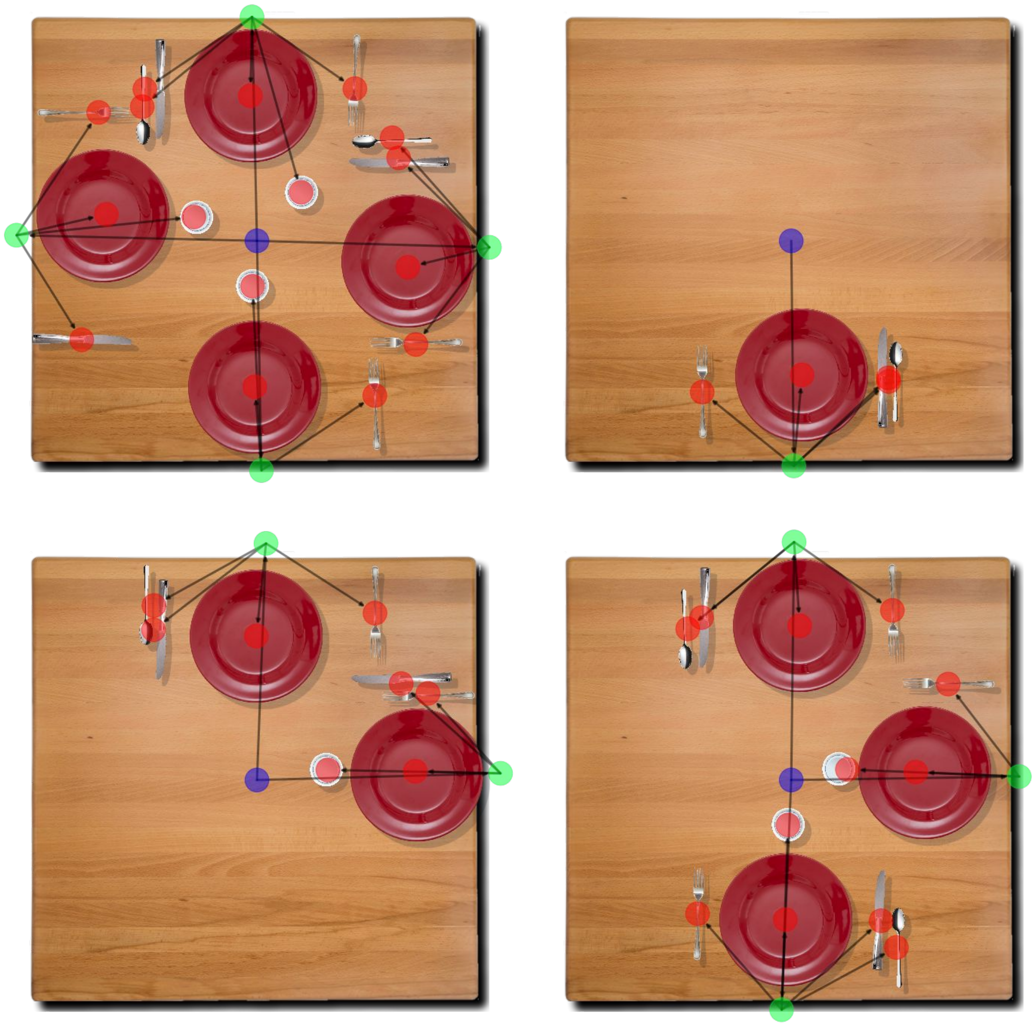
But how do I achieve robustness to
every mug? every shoe?
in every kitchen?

To achieve robustness, do I need to simulate the diversity of the world?

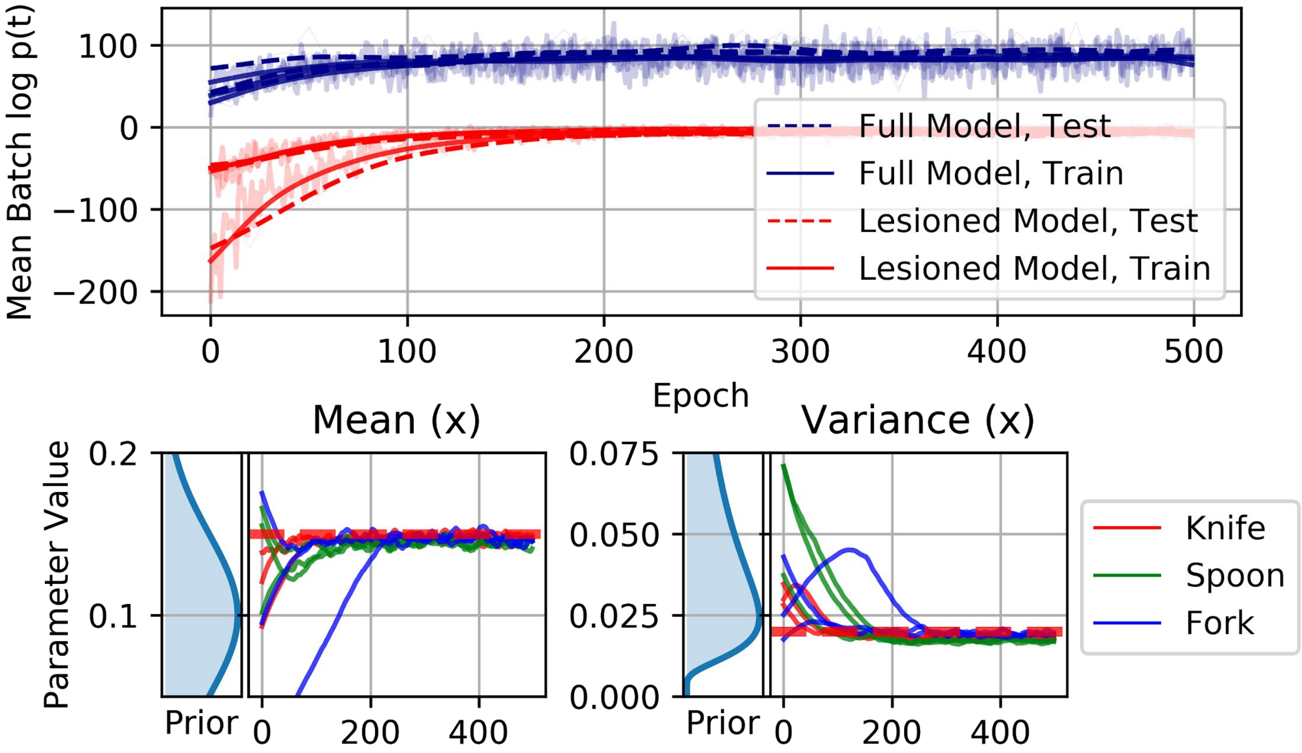
Generative Modeling of Environments with Scene Grammars and Variational Inference

Gregory Izatt and Russ Tedrake
$\{gizatt, russt\}$@csail.mit.edu

A 2D Example

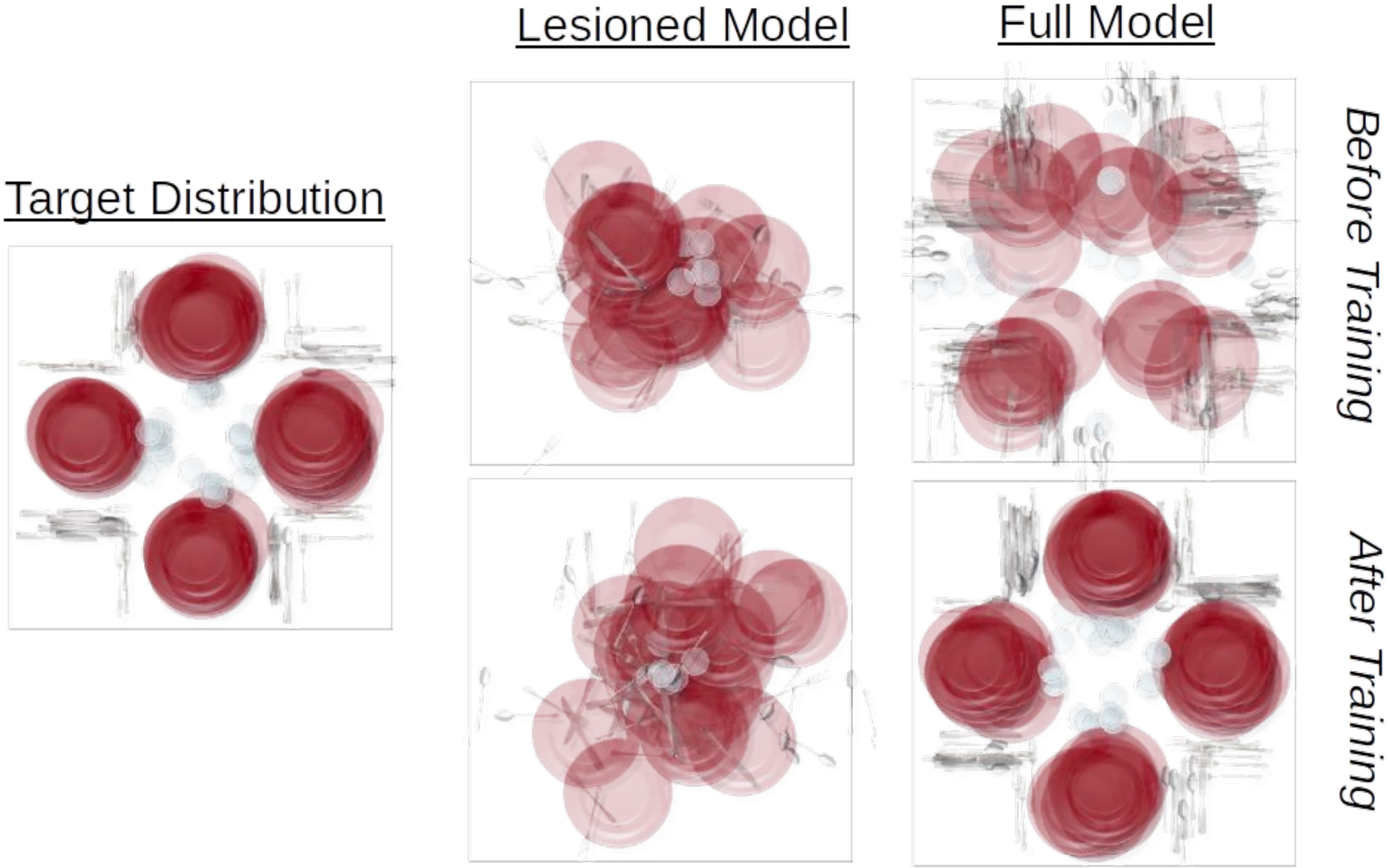


Training

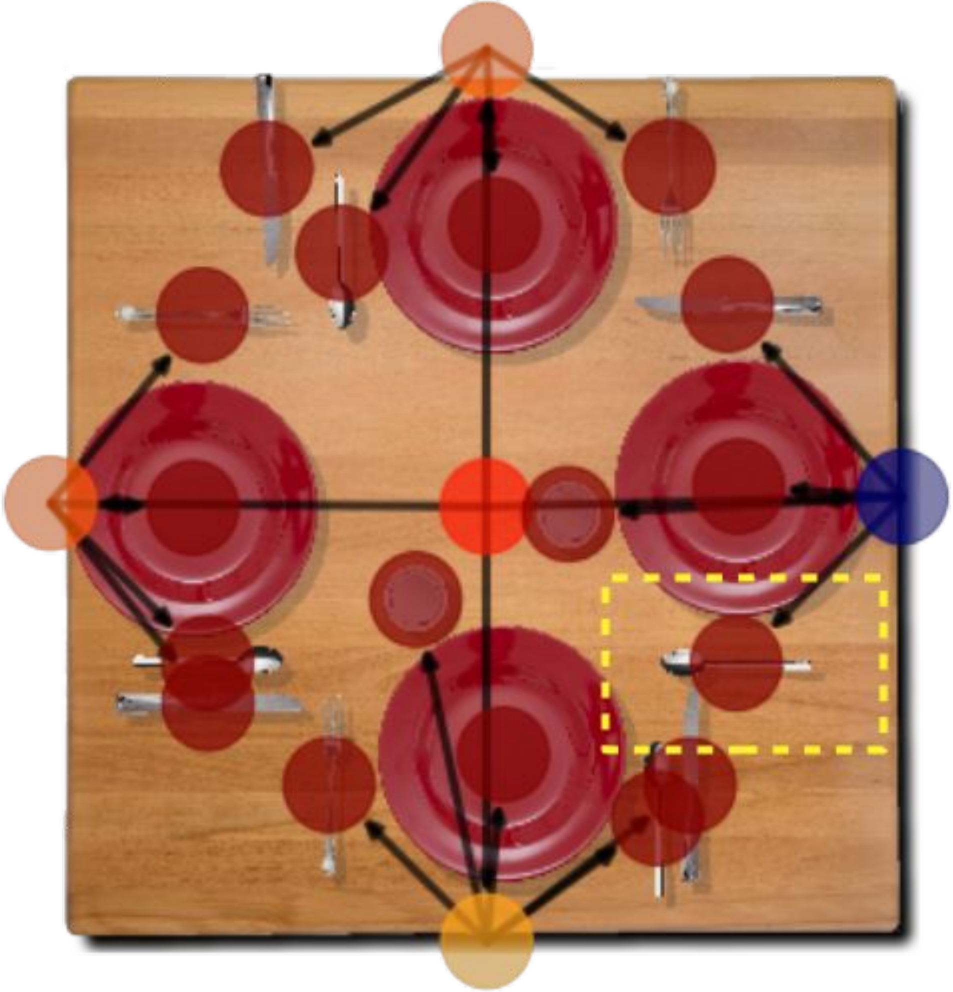


Optimize an ELBO loss with REINFORCE/ADAM in Pyro.

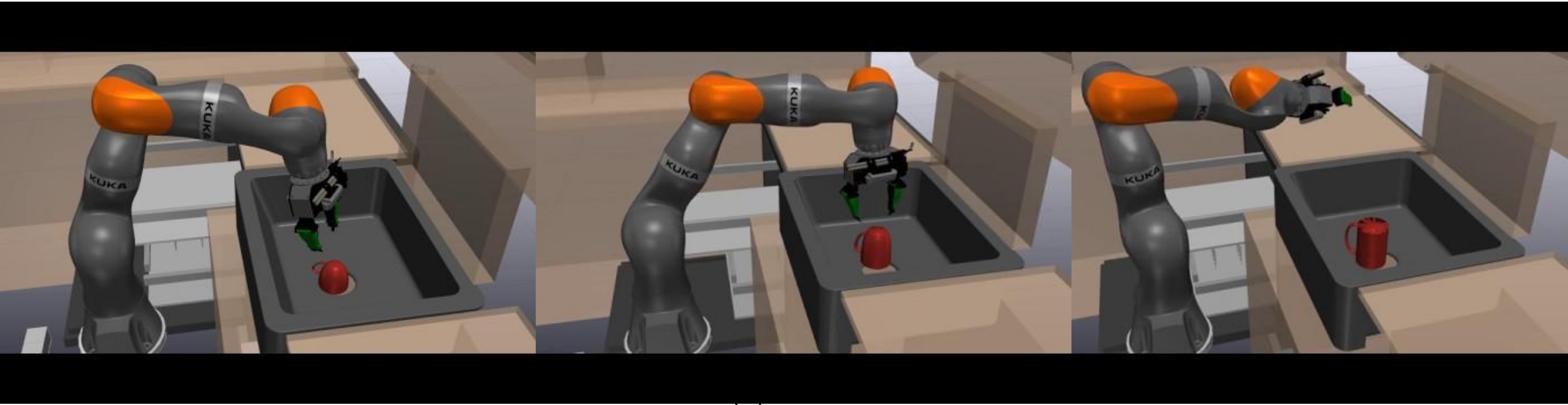
$$\arg\max_{\Theta,\Gamma} \sum_{\mathbf{o}_i \in D} \mathrm{E}_{t\sim q_\Gamma(t)} \left[\log p_\Theta(\mathbf{o}_i, t) - \log q_\Gamma(t) \right].$$


Target Distribution
Lesioned Model
Full Model
Before Training
After Training

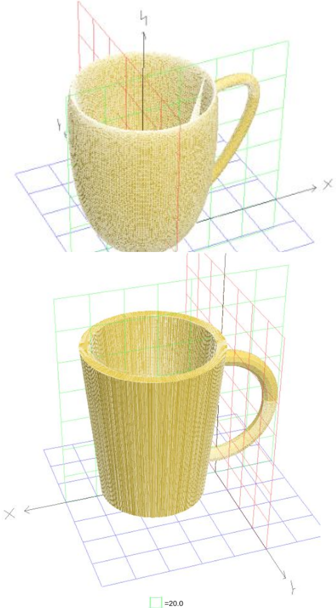
Outlier detection



Procedural dishes



Procedural dishes



TOYOTA
RESEARCH INSTITUTE

Uncertainty representations

In controls (polytopic/ellipsoidal, etc)

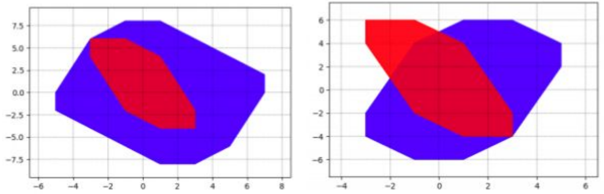
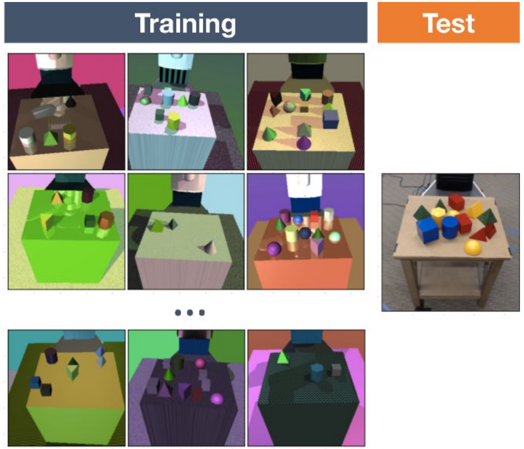


Fig. 1. Example 1: Zonotope Containment Problem: [left] $\mathbb{Z}_l \subseteq \mathbb{Z}_r$, [Right] $\mathbb{Z}_l \not\subseteq \mathbb{Z}_r^*$, where the last column of G_r is dropped.

Domain randomization in reinforcement learning



Abbeel et al.

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

Developing autonomous systems in the real world.

Uncertainty representations

Does simultaneous stabilization (or expected value) with richer randomization of the task/environment somehow make the problem fundamentally easier?

Can we bring robust control formulations closer to what the real world uses to develop autonomous systems?

More big questions

Can we simulate everything in the kitchen?

Napkins? Ketchup? Soba noodles?

How accurate do our models have to be?

Task requirements

Control authority
+ model fidelity

How do I provide test coverage for every possible kitchen?

Hypothesis: Only need a sufficiently rich sandbox to deploy

+ continual improvement (fleet learning)

Summary

Optimization brought us today's "**modern control**"...

..with strong results for relatively simple forms uncertainty.

Real world uncertainty and "domain randomization" in RL is much richer.
"Black-box" optimization in RL still works.

Dealing with perception and "open-worlds" may cause the next major shift in controls research; we need the maturity of control to help address fundamental problems in robustness and sample-complexity.