TOYOTA
RESEARCH INSTITUTE

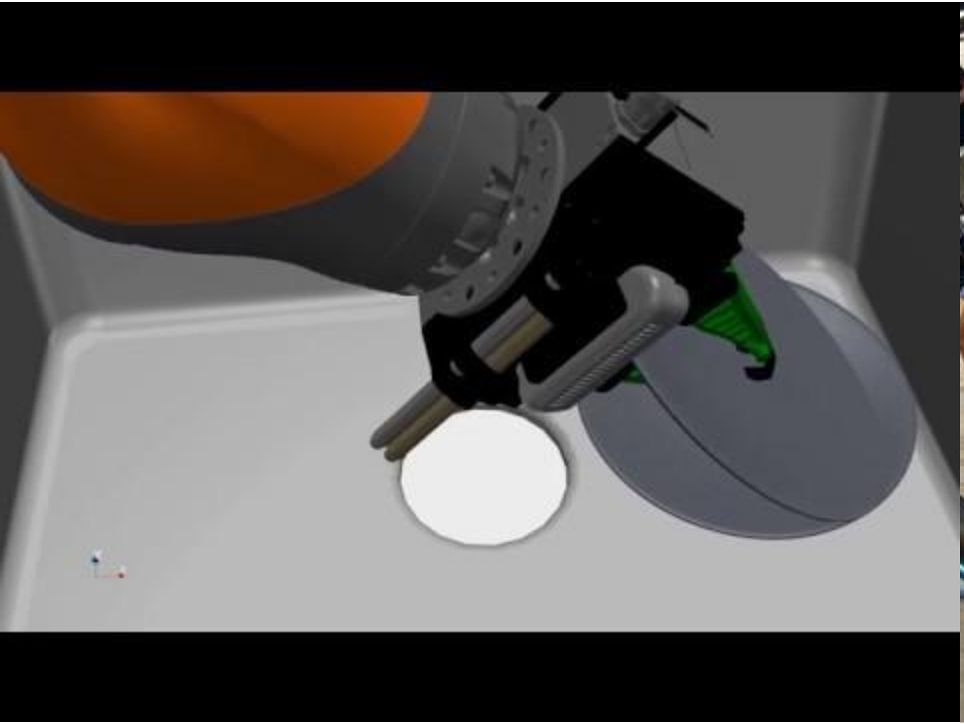
Sim2Real for the corner cases?
Getting to robust manipulation

Russ Tedrake, MIT and TRI

TRI's robotics mission is to develop breakthrough capabilities that dramatically improve the quality of life

Today: One experiment in *robust manipulation* -- **loading the dishwasher**

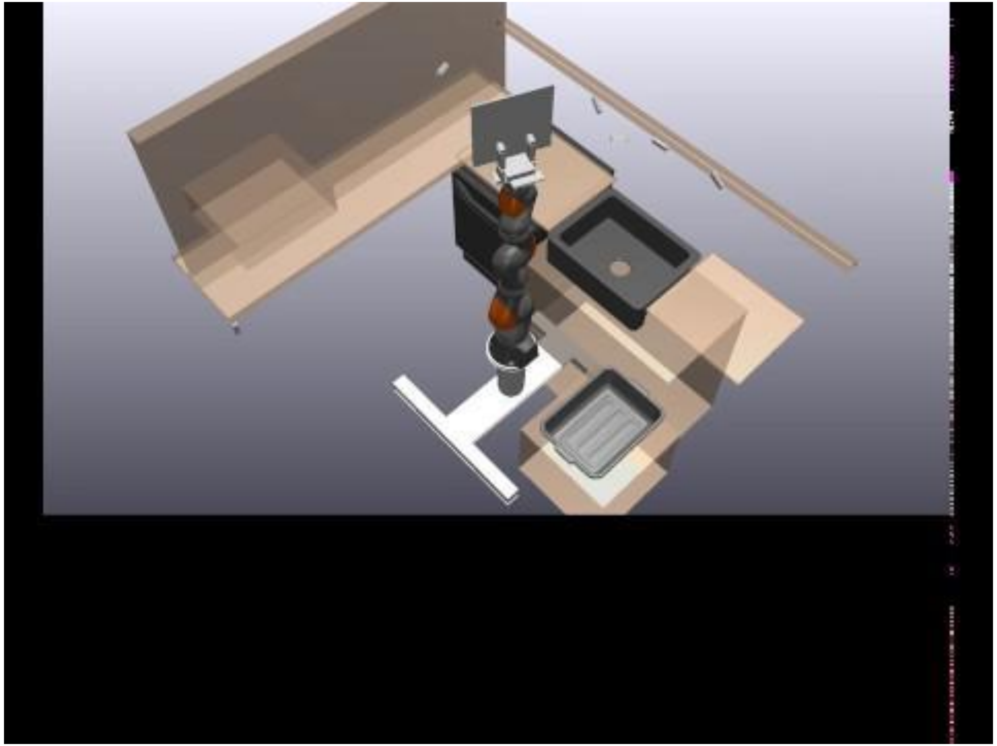
Simulation



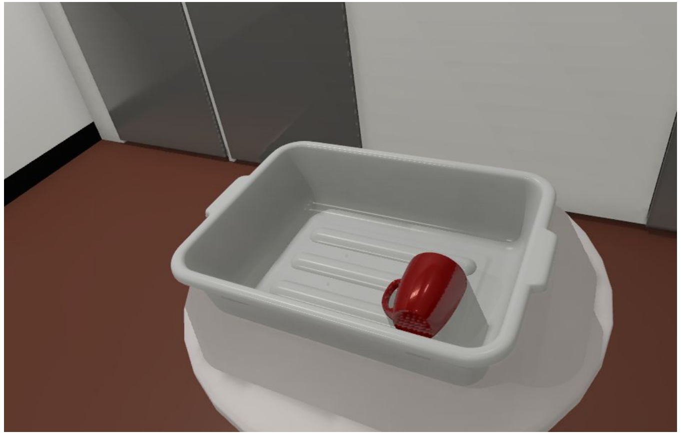
“Simulation-first” development

Early example:

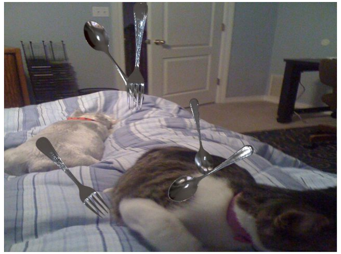
Camera calibration refactor

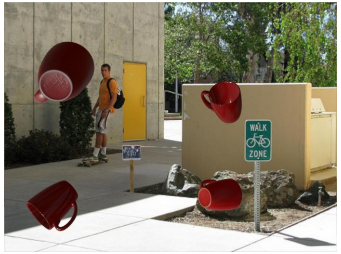


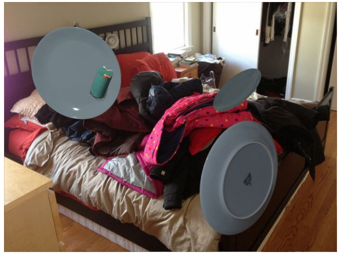
Ground-truth labels for training perception



Godot PBR + OSPray ray-tracing

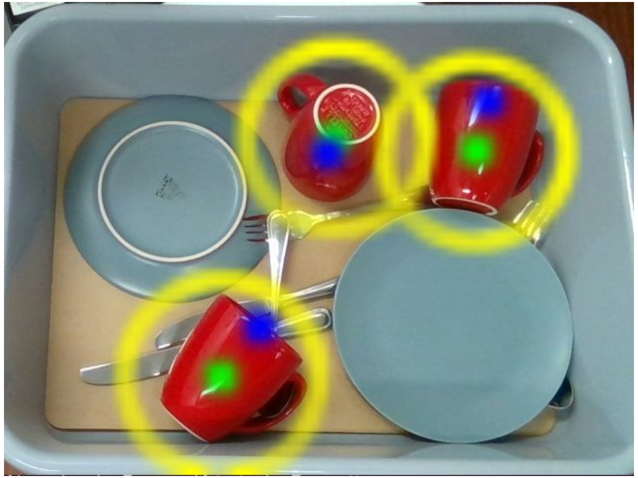




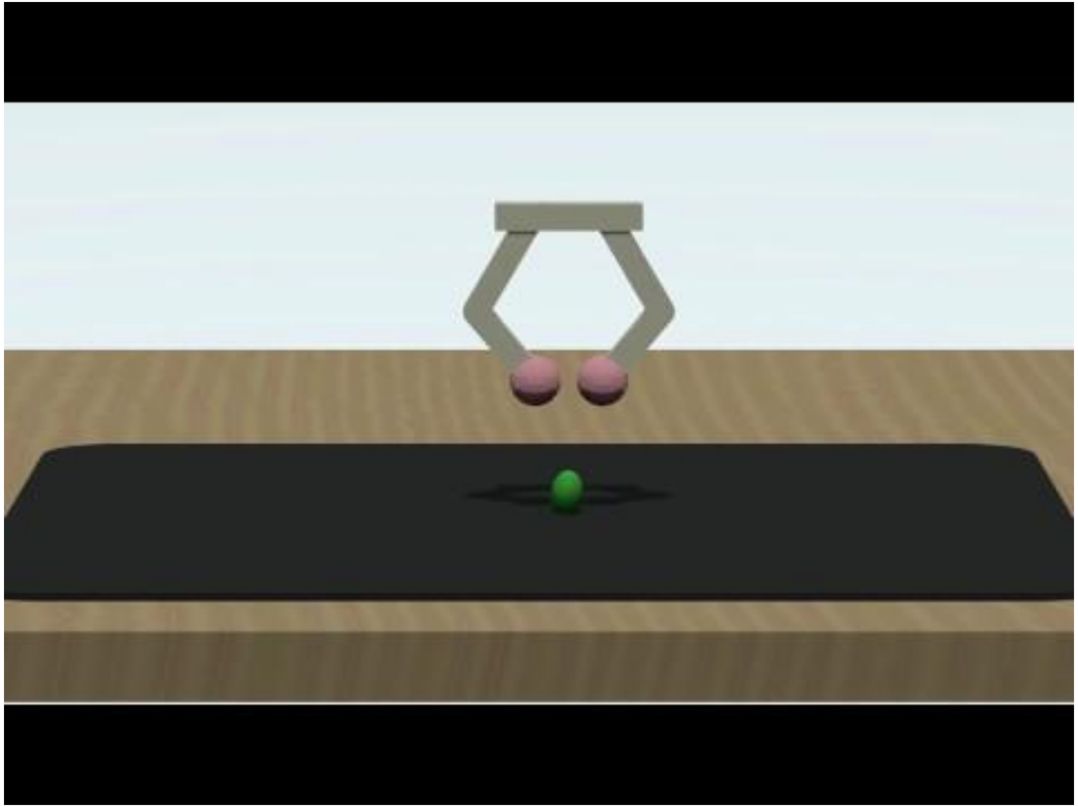


Rendered objects on COCO backgrounds

+ (sometimes) fine-tuning on labeled real data



Robust contact simulation

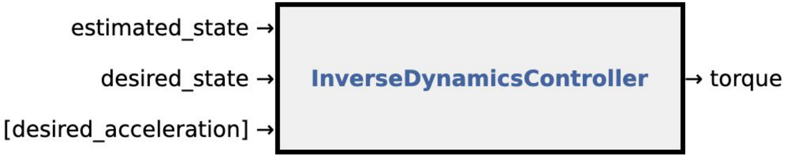


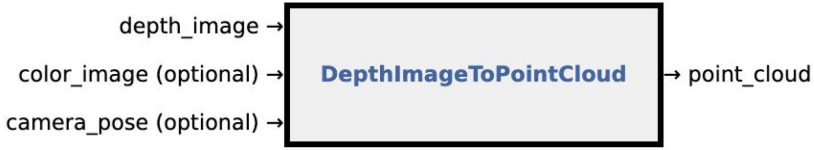
“Hydro-elastic” contact model from TRI + Cornell: Ryan Elandt, Evan Drumwright, Michael Sherman, and Andy Ruina

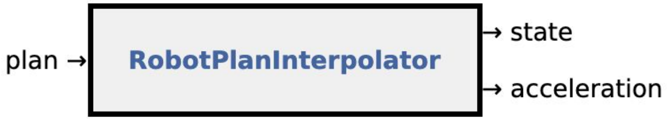
Physics and rendering are not sufficient...

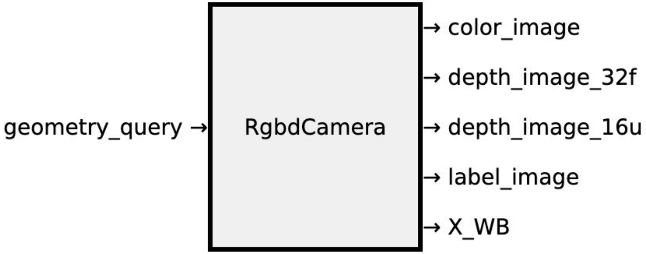
Simulation also requires modeling

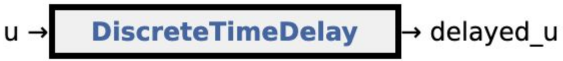
- Robot controllers/firmware
- Sensors
- Sensor noise
- Perception components
- Planning components..
- Time delays
- ...





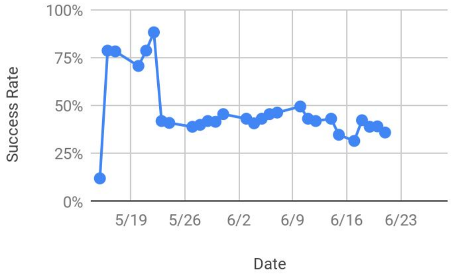




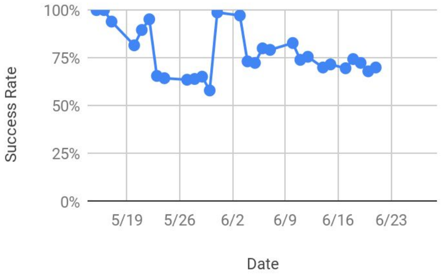


Monte Carlo falsification

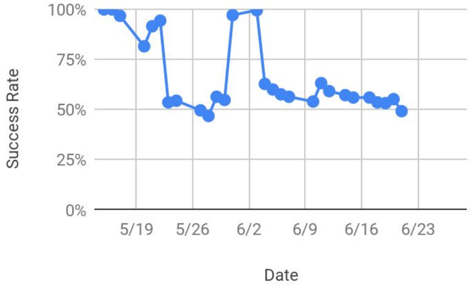
Pull Lower Rack Nightly



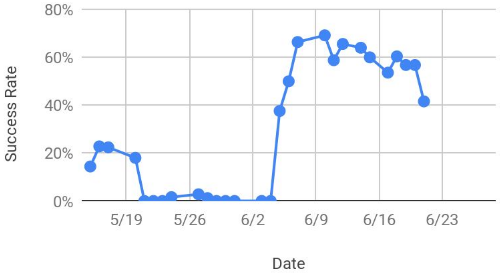
Push Lower Rack Nightly



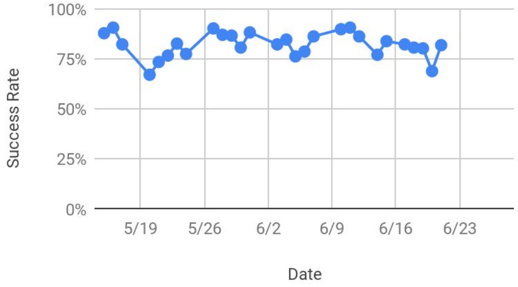
Push Upper Rack Nightly



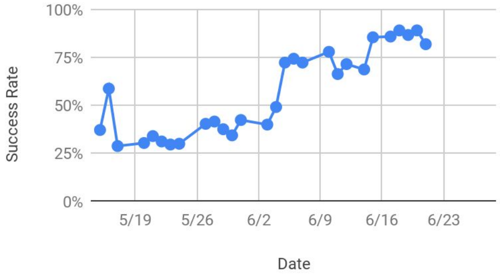
Load Plate Nightly



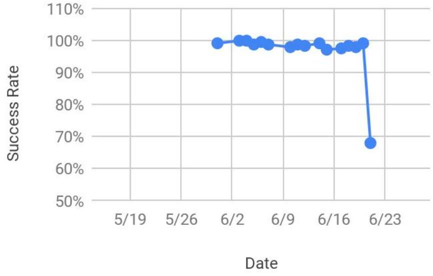
Load Mug Nightly



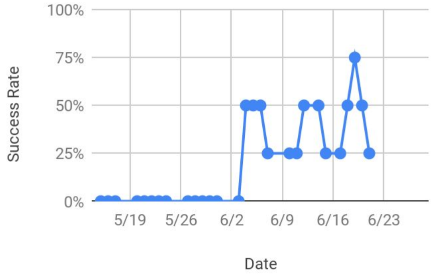
Load Silverware Nightly



Open Door

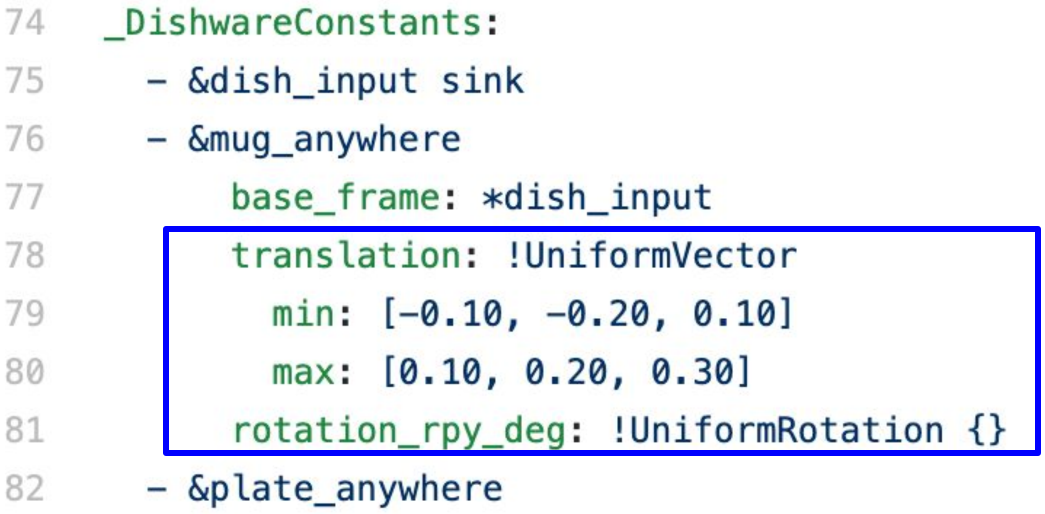


End to End



Scenario description files

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

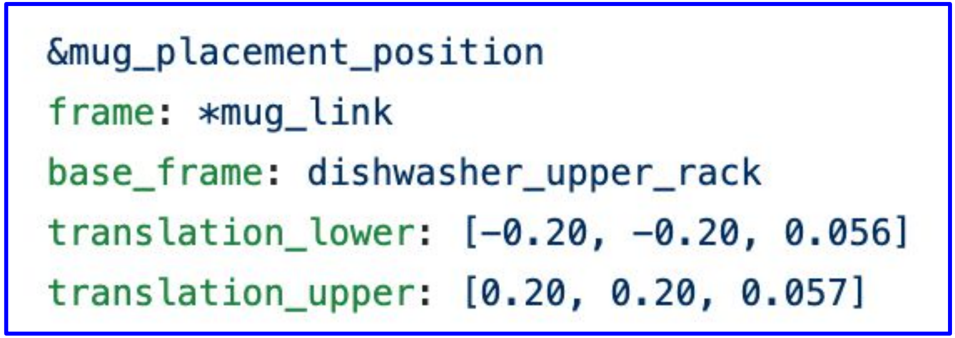


```
_DishwareConstants:
  - &dish_input sink
  - &mug_anywhere
      base_frame: *dish_input
      translation: !UniformVector
        min: [-0.10, -0.20, 0.10]
        max: [0.10, 0.20, 0.30]
      rotation_rpy_deg: !UniformRotation {}
  - &plate_anywhere
```

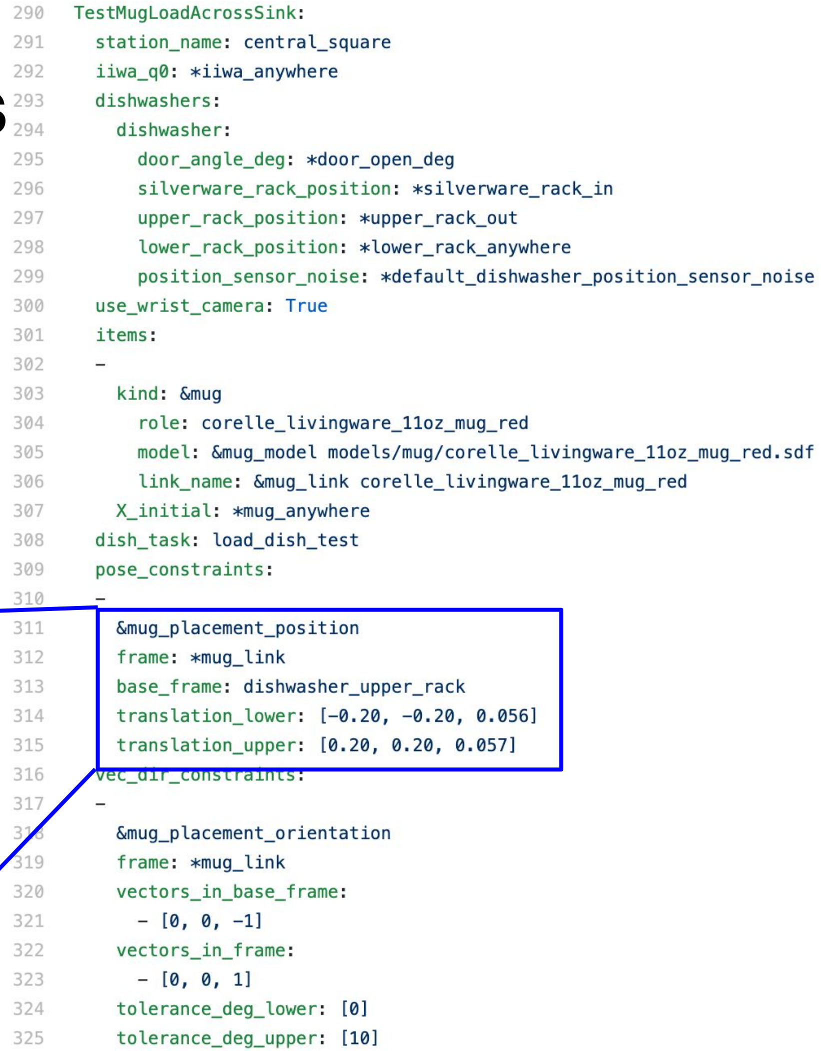
Scenario description files

Success criteria specified as constraints on systems

Can compose into complex diagrams, and be used for synthesis

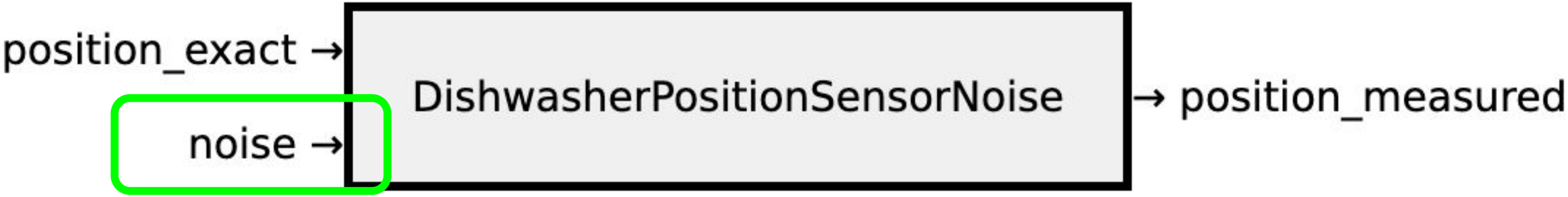


```
&mug_placement_position
frame: *mug_link
base_frame: dishwasher_upper_rack
translation_lower: [-0.20, -0.20, 0.056]
translation_upper: [0.20, 0.20, 0.057]
```



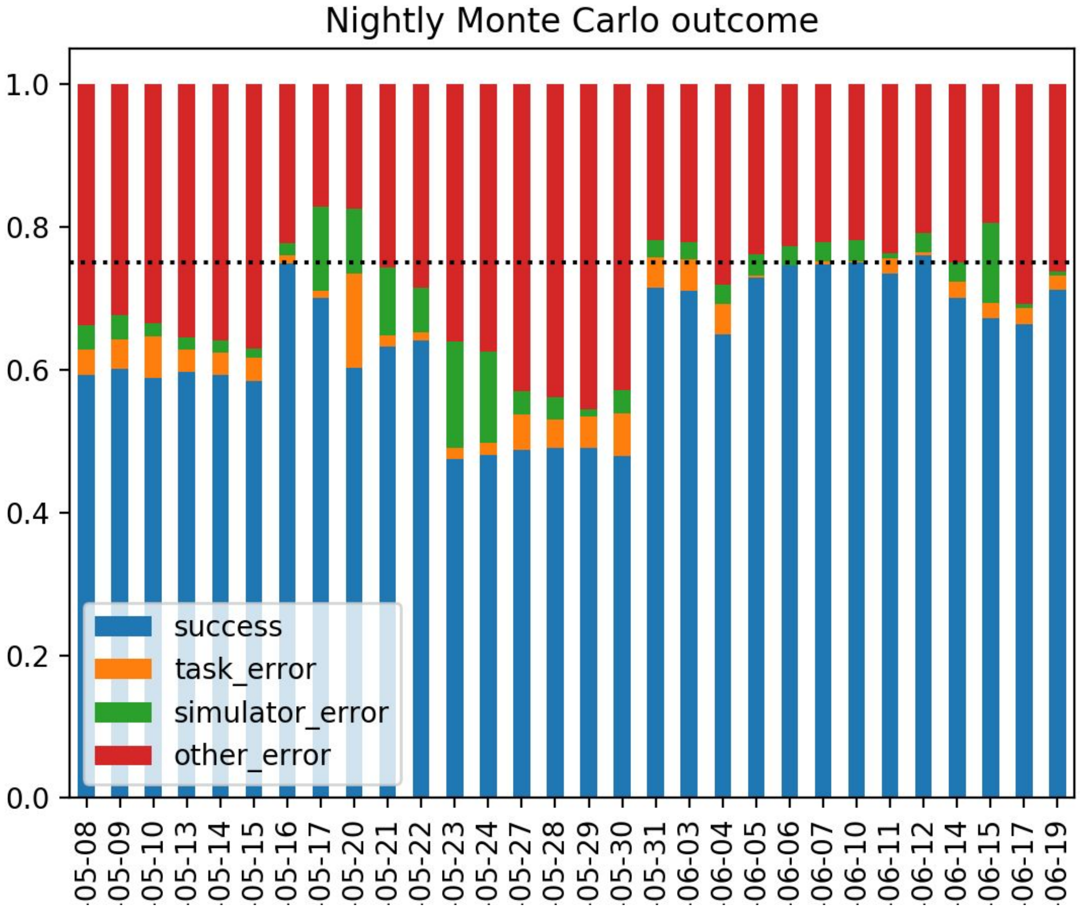
```
TestMugLoadAcrossSink:
  station_name: central_square
  iiwa_q0: *iiwa_anywhere
  dishwashers:
    dishwasher:
      door_angle_deg: *door_open_deg
      silverware_rack_position: *silverware_rack_in
      upper_rack_position: *upper_rack_out
      lower_rack_position: *lower_rack_anywhere
      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]
```

Rigorous about randomness



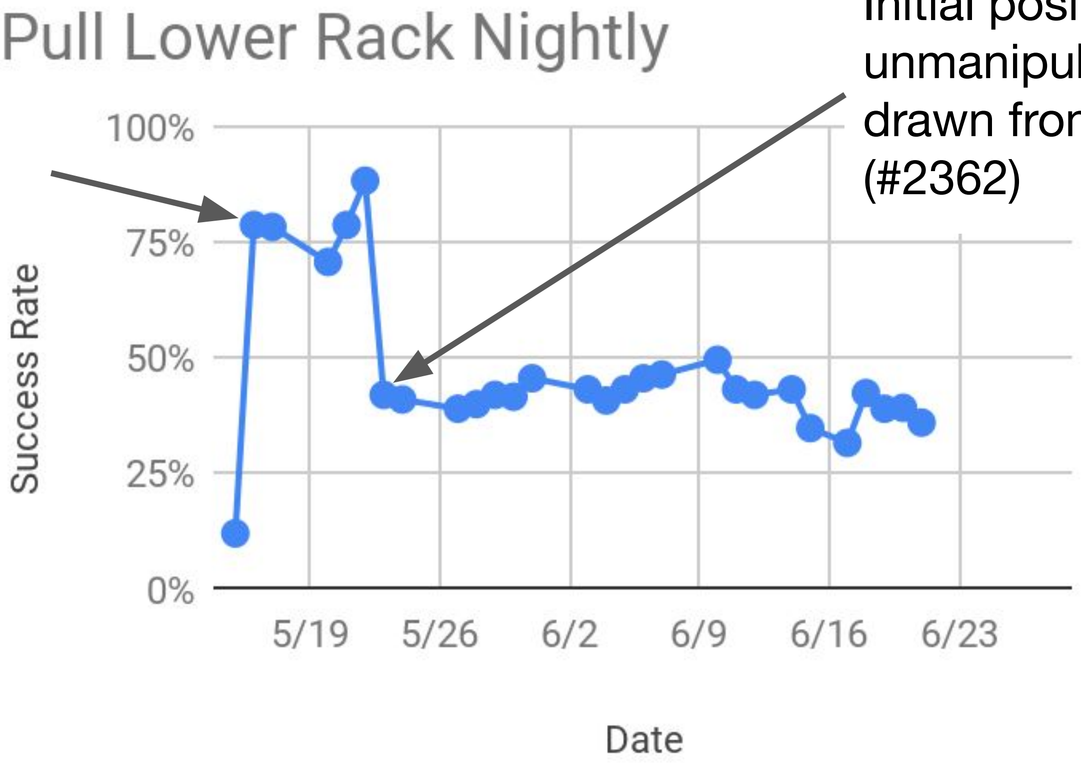
Every source of randomness is declared explicitly (using elementary distributions)

- Scene (#/type of objects)
- Parameters/initial conditions
- Time-varying noise



First you find bugs in your simulator!

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

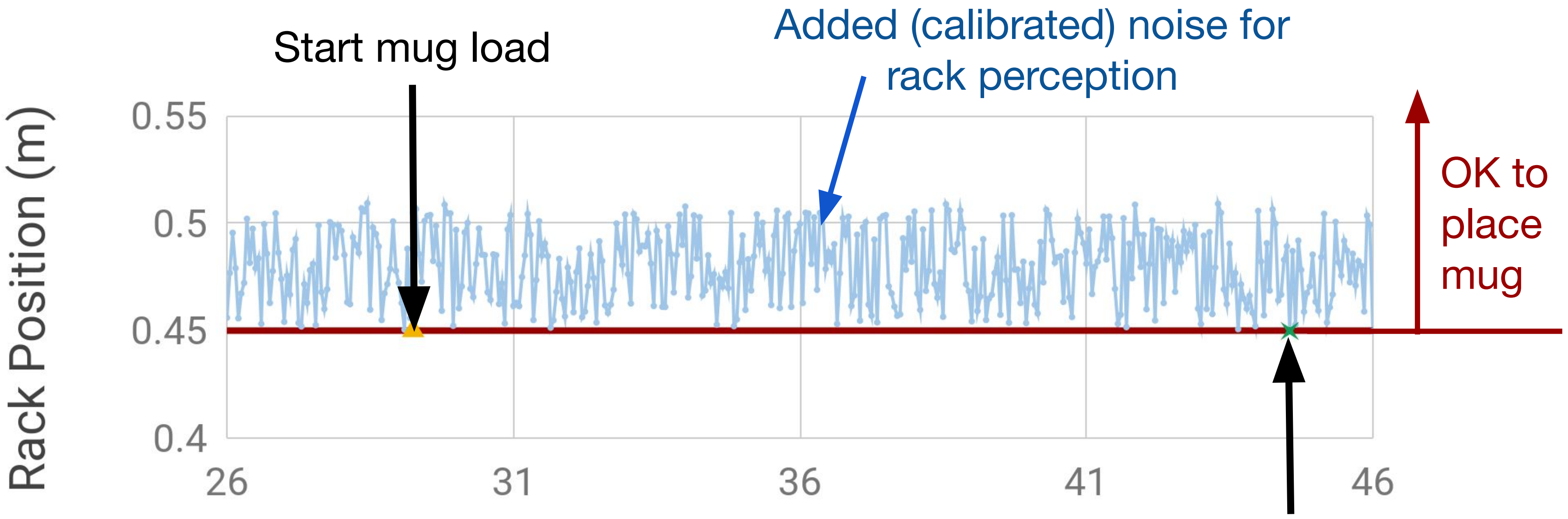


Initial positions of the unmanipulated racks are drawn from MC instead of 0 (#2362)

Finding subtle bugs

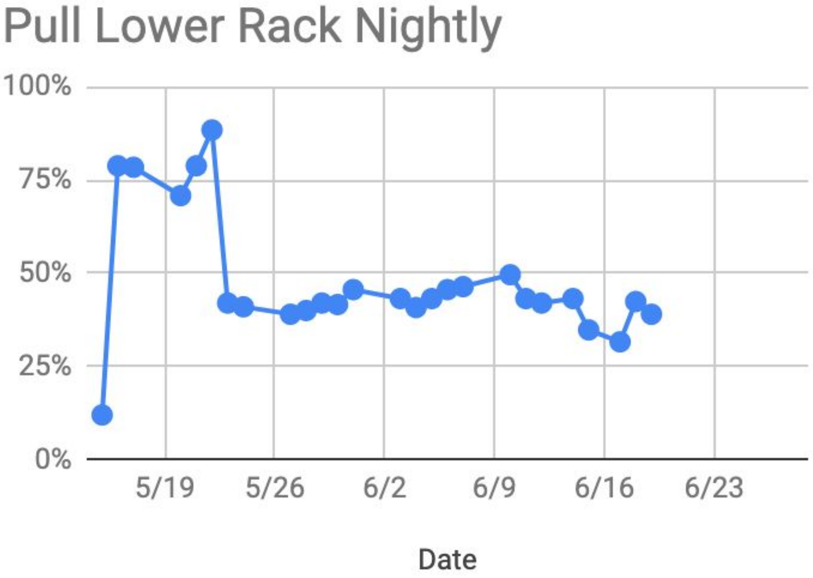
Finding subtle bugs

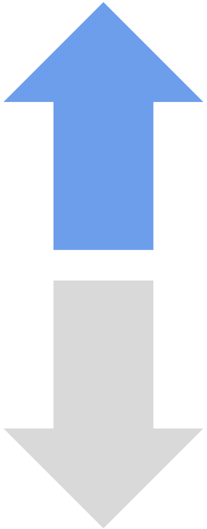
Finding subtle bugs



Stop! Rack appears closed

Falsification algorithms



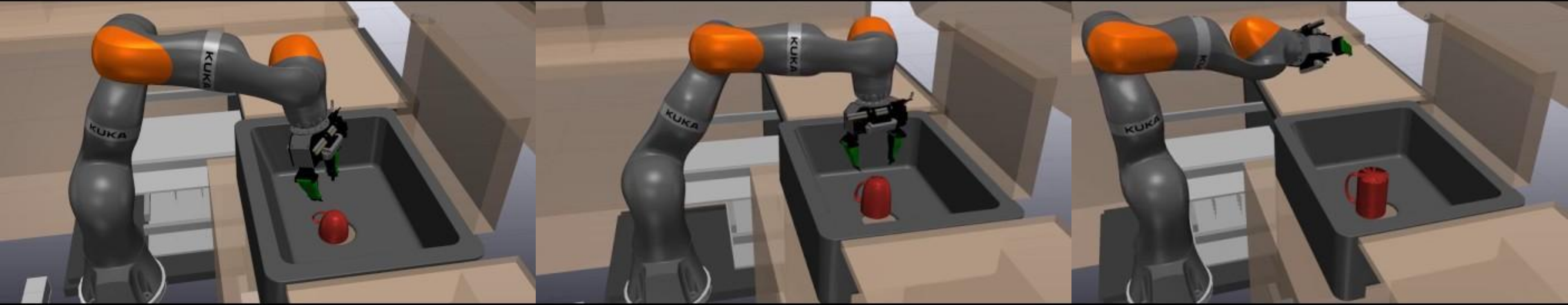


Improve robustness / fix bugs

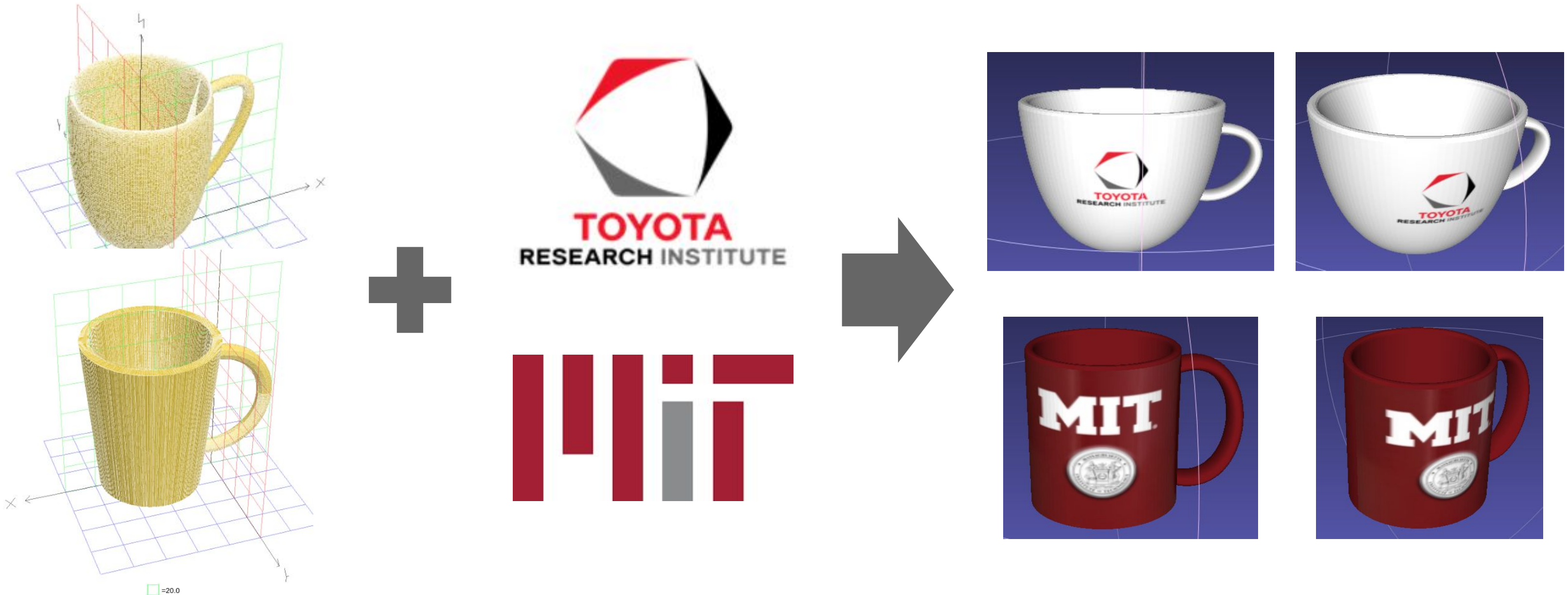
Increase test randomness / scope

naive Monte Carlo has been sufficient (so far)

Procedural dishes



Procedural dishes



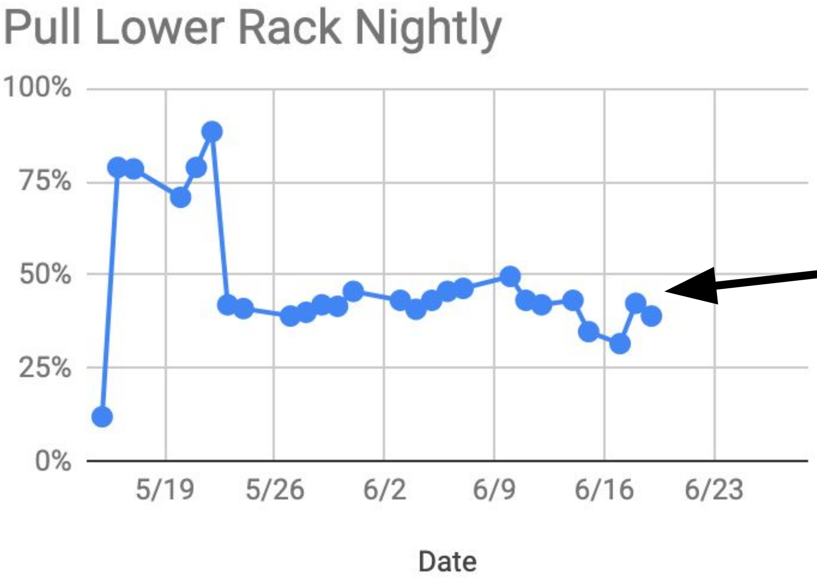
Built in our framework for
optimization-based control/analysis...

DRAKE

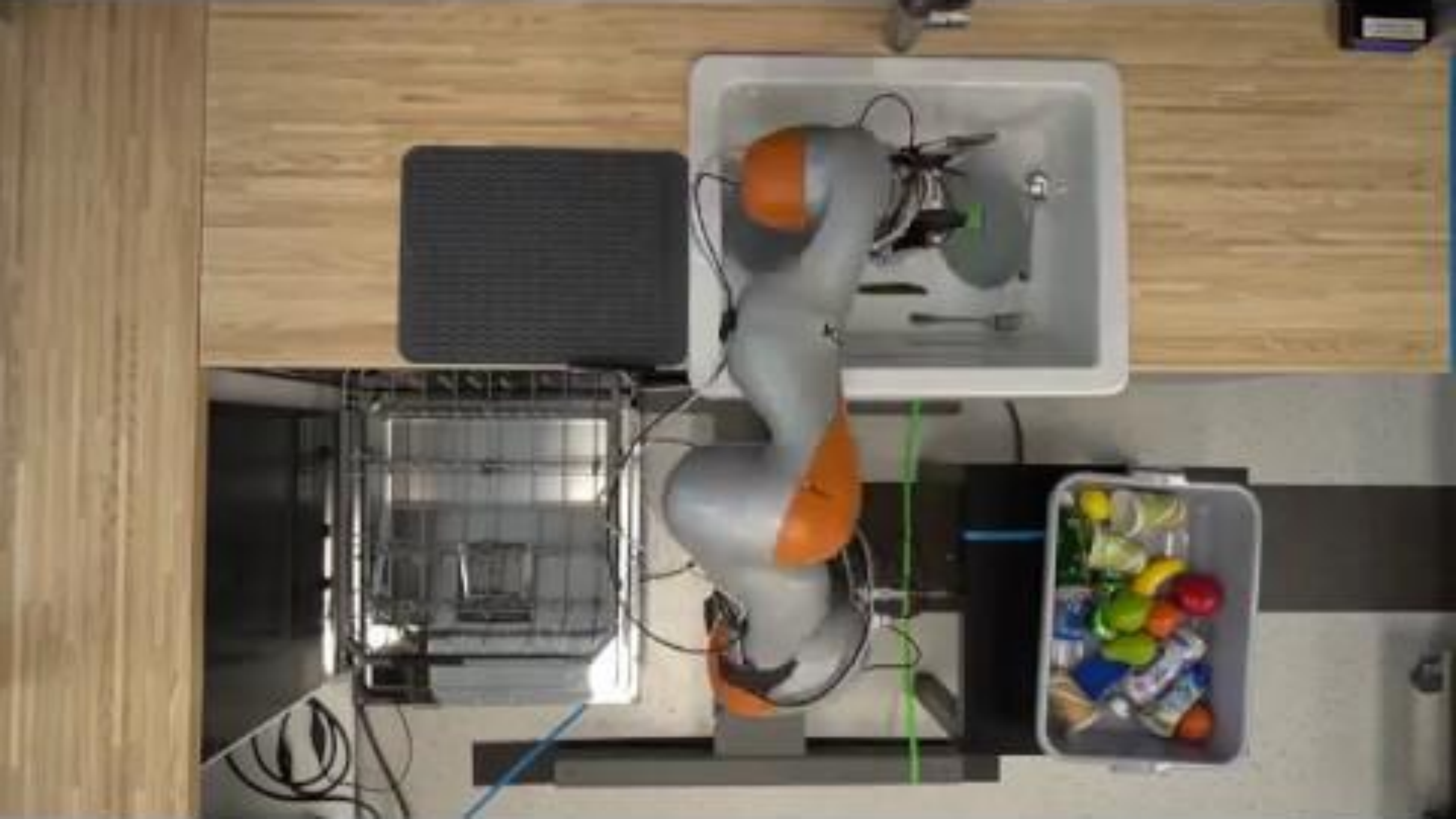
MODEL-BASED DESIGN AND
VERIFICATION FOR ROBOTICS

So how well does it work?

Sim vs Real



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



The big questions

Can we simulate everything in the kitchen?

Napkins? Ketchup? Soba noodles?

How accurate do our simulations have to be?

Control authority

+ model fidelity

Task requirements

How do I provide test coverage for every possible kitchen?

Hypothesis: Only need a sufficiently rich sandbox to deploy

+ continual improvement (fleet learning)

Summary

- Investigating “Sim2Real” for manipulation; even for the corner cases
- Develop novel algorithms faster, with quantifiable robustness metrics and real-world gains
- Manipulation stack with rigorous system + uncertainty modeling (+ gradients, etc)
- Core elements are available at http://drake.mit.edu