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.
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.

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)

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 $\xi_i$ 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.
OpenAI Gym vs. ETH ANYmal results

To this end, we train a neural network representing this complex dynamics with data from the real robot.
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
Proposed problem formulation:

“Class-general” manipulation.
kPAM pipeline

No template model or pose appears in this pipeline.
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)
Requirements/Specifications

In controls (polytopic/ellipsoidal, etc)

![Graphs showing containment problems](image1)

Fig. 1. Example 1: Zonotope Containment Problem: [left] $Z_d \subseteq Z_r$, [Right] $Z_d \not\subseteq Z_r$, where the last column of $G_r$ is dropped.

Creating Driving Tests for Self-Driving Cars

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

Developing autonomous systems in the real world.
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
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
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)

---

**Pull Lower Rack Nightly**

![Graph showing success rate over time]

Success Rate

Date

Finding subtle bugs
Finding subtle bugs
Finding subtle bugs

Start mug load

Added (calibrated) noise for rack perception

OK to place mug

Stop! Rack appears closed
Falsification algorithms

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

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

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

Table 1: Estimate of rare-event probability $p_\gamma$ (non-vision ego policy), with standard deviations

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

Table 2: Estimate of rare-event probability $p_\gamma$ (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
Optimize an ELBO loss with REINFORCE/ADAM in Pyro.

\[
\arg\max_{\Theta, \Gamma} \sum_{o_i \in D} \mathbb{E}_{t \sim q_{\Gamma}(t)} \left[ \log p_{\Theta}(o_i, t) - \log q_{\Gamma}(t) \right].
\]
Outlier detection
Procedural dishes
Procedural dishes
Uncertainty representations

In controls (polytopic/ellipsoidal, etc)

Developing autonomous systems in the real world.

Domain randomization in reinforcement learning

Abbeel et al.
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?
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.