MOTION PLANNING FOR STRUCTURED EXPLORATION IN ROBOTIC REINFORCEMENT LEARNING

CANNON LEWIS

WHO AM I?

Senior at Rice University

Majoring in Computer Science and Mathematics

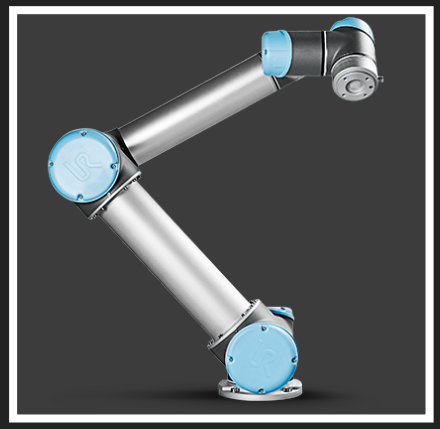
Currently applying to artificial intelligence PhD programs

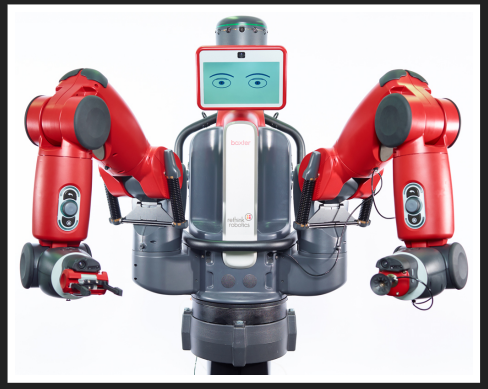
WHY AM I HERE?

Took COMP 311: Functional Programming from Dr. Eric Allen

Interned at Two Sigma during Summer 2016

Currently doing research with Dr. Lydia Kavraki





SO MANY ROBOTS!

Excitement due to recent advances in artificial intelligence

New use cases in medicine, search and rescue

New kinds of robots (e.g. drones, "safe" robotics, nanorobots)

We need to be able to plan motions for these robots in the presence of obstacles

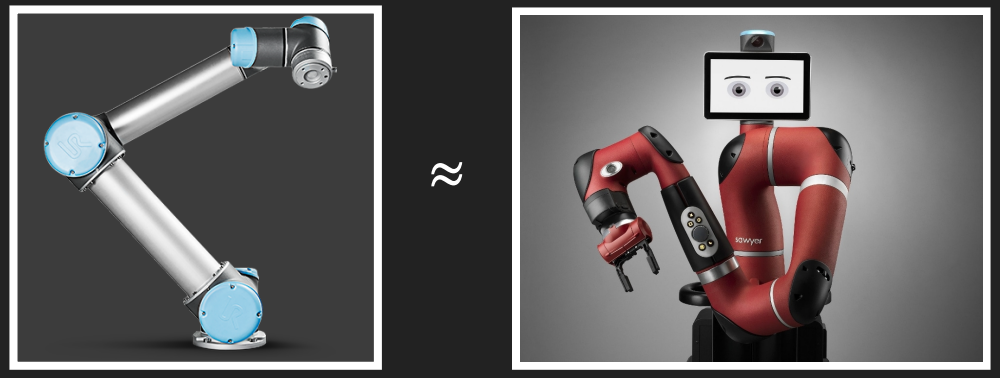
But how do we work with these very different robots?

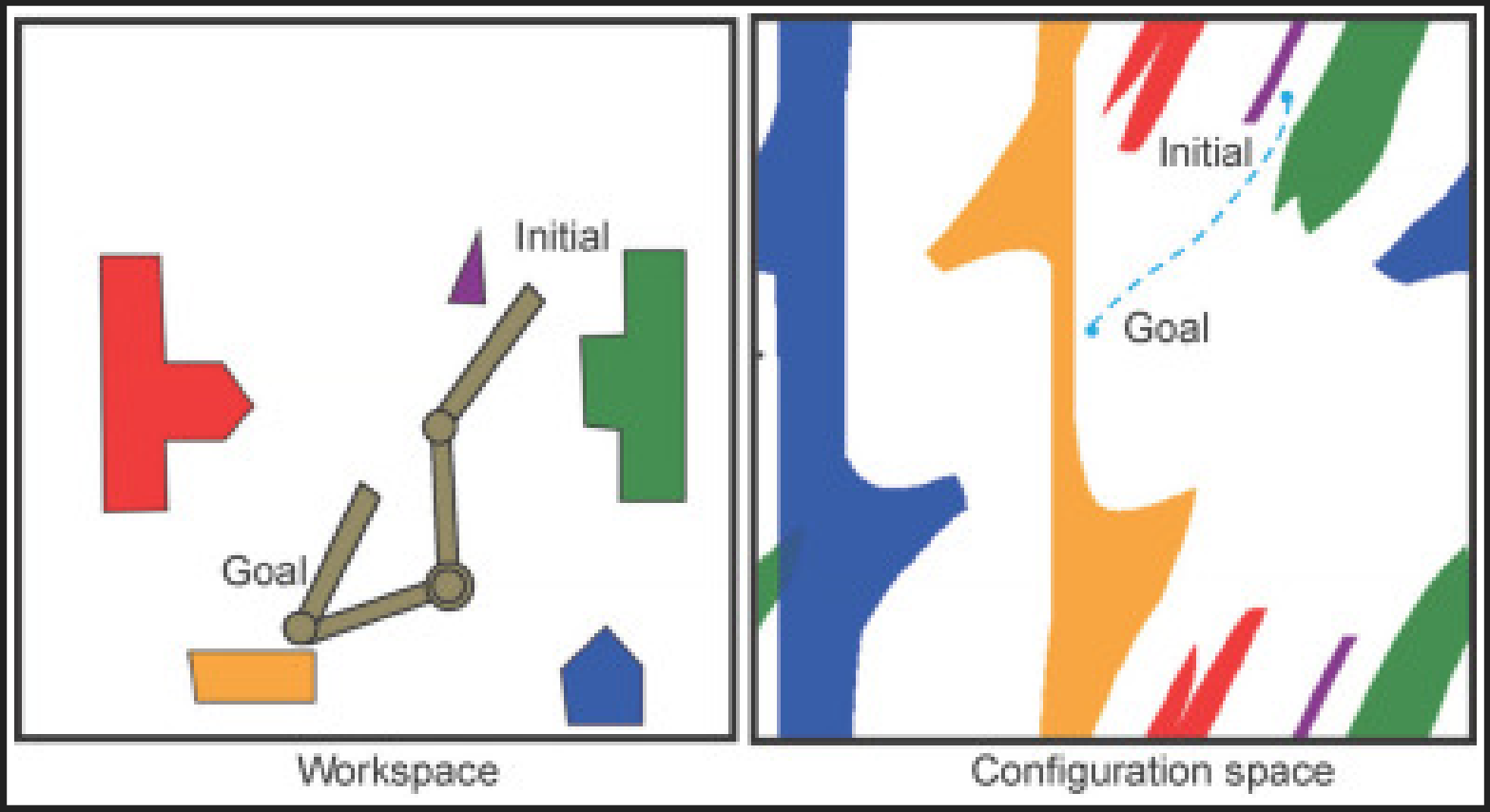
CONFIGURATION SPACES

In order to plan for the wide array of robots that exist, we need an abstraction, which we call the "configuration space"

Even though our robots operate in $\mathbb{R}^3$, the space of movements for a robot is more accurately represented by $\mathbb{R}^6$, or even T^6

Under this abstraction, we can consider certain robots as being equivalent




Initial
Goal
Workspace
Initial
Goal
Configuration space

ROBOT MOTION IS DIFFICULT

As we have seen, the configuration space can look very different from the 3D world that we are used to

Search in this space is difficult due to

1. High dimensionality
2. Continuity
3. Limited computational resources

These issues prevent us from using traditional search and planning methods

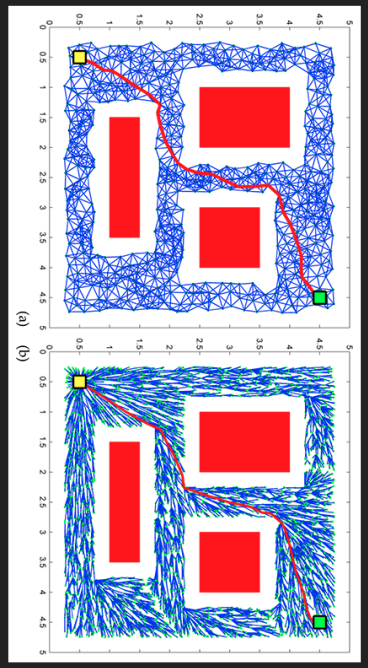
SAMPLING-BASED MOTION PLANNING

Instead of searching through the configuration space directly or discretizing the space, we build an implicit roadmap by sampling

Algorithms in this vein have revolutionized robot motion planning, and are now the dominant method

- These algorithms are fully general - they can plan for any robot whose configuration space is connected

Algorithms for SBMP such as PRM, RRT, KPIECE, and more are available out-of-the-box in my lab's OMPL library



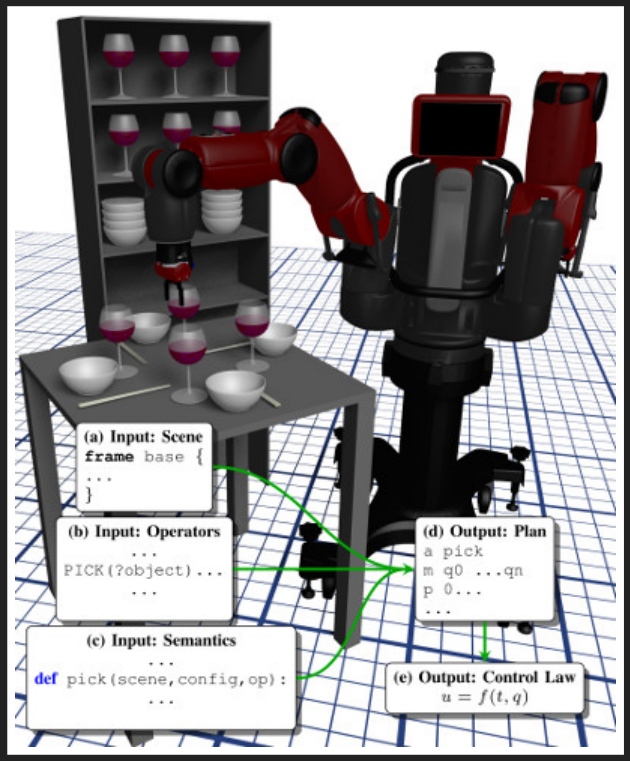
HIGH-LEVEL TASK PLANNING

In addition to making low-level motion plans, we would like to solve high-level tasks

For example, in tabletop manipulation we would like to simply tell the robot where to place dishes, instead of specifying how to move its arm for each motion

TMKit is a project from our lab that performs this planning incrementally

Unfortunately, even abstract problem solving is PSPACE-complete, so exact Task-and-Motion Planning is intractable in many cases



THEORETICAL BASICS

RL is a form of machine learning in which a learning agent is provided with a real-valued reward function in response to its actions

RL stems from research into Markov Decision Processes (MDPs)

Goal is to build an agent which can learn a maximizing policy $\pi(s, a)$ for an arbitrary MDP that the agent is interacting with

WHAT IS AN MDP?

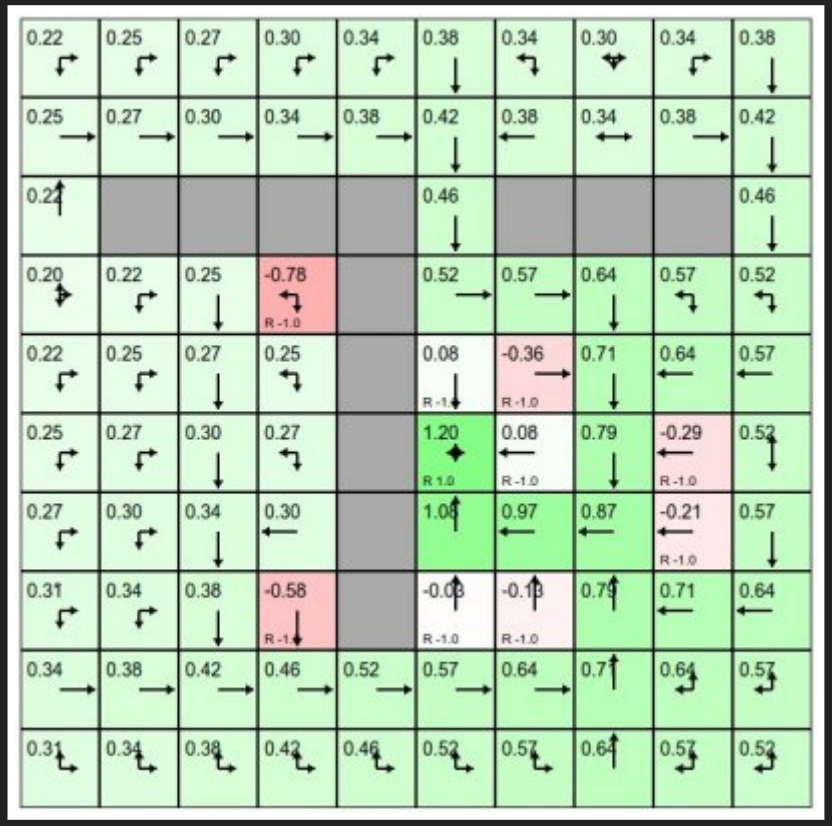
- A set of states S, set of actions A.
- Stochastic transition function $P_a(s, s')$
- Reward function $R(s, a, s')$

VALUE FUNCTIONS

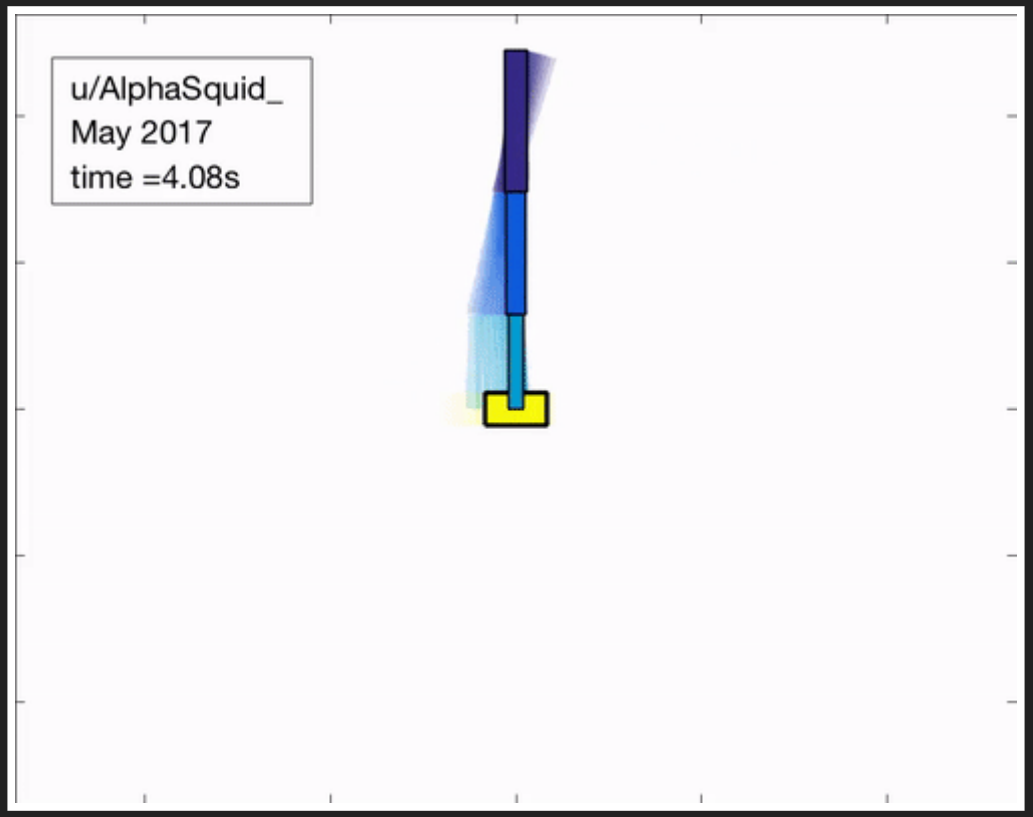
For elementary solution methods we care about:

- $V^\pi(s) = \mathbf{E}_\pi\{R_t | s_t = s\} = \mathbf{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right.$
- $Q^\pi(s, a) = \mathbf{E}_\pi\{R_t | s_t = s, a_t = a\}$
  $= \mathbf{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t =\right.$

GRIDWORLD


0.22 0.25 0.27 0.30 0.34 0.38 0.34 0.30 0.34 0.38
0.25 0.27 0.30 0.34 0.38 0.42 0.38 0.34 0.38 0.42
0.22 0.46 0.46
0.20 0.22 0.25 -0.78 R -1.0 0.52 0.57 0.64 0.57 0.52
0.22 0.25 0.27 0.25 0.08 R -1.0 -0.36 R -1.0 0.71 0.64 0.57
0.25 0.27 0.30 0.27 1.20 R 1.0 0.08 R -1.0 0.79 -0.29 R -1.0 0.52
0.27 0.30 0.34 0.30 0.97 0.87 -0.21 R -1.0 0.57
0.31 0.34 0.38 -0.58 R -1.0 R -1.0 R -1.0 0.71 0.64
0.34 0.38 0.42 0.46 0.52 0.57 0.64 0.57
0.31 0.34 0.42 0.46 0.52 0.57 0.64 0.57 0.52

SIMPLE CONTROL



RL TOOLCHAIN

Due to recent high-profile reinforcement learning results, many open-source RL libraries are available

Prominent among these is OpenAI Gym. Inexplicably, this project seems to have been abandoned recently

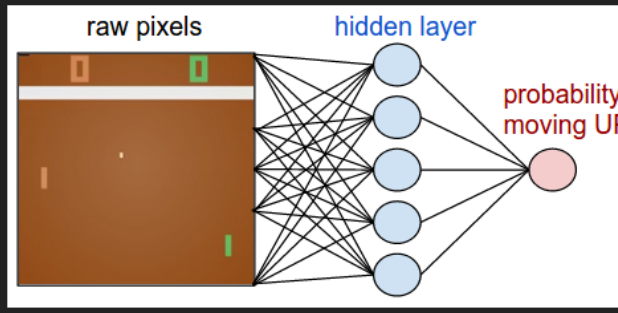
OpenAI also created Universe and RoboSchool, but these are similarly poorly maintained

Serpent AI is a high-quality recent addition to this landscape, originating from frustration with OpenAI Universe

LIMITATIONS OF NAIVE RL

Exact solution is prohibitively expensive in both time and space for general MDPs

Furthermore, an enumerative approach to value function calculation is incompatible with continuous state or action spaces



DEEP LEARNING

To overcome these difficulties, we introduce function approximation, typically through neural networks

We can parameterize either the value functions or the policy as a neural network, then use typical gradient descent techniques to train our RL agent

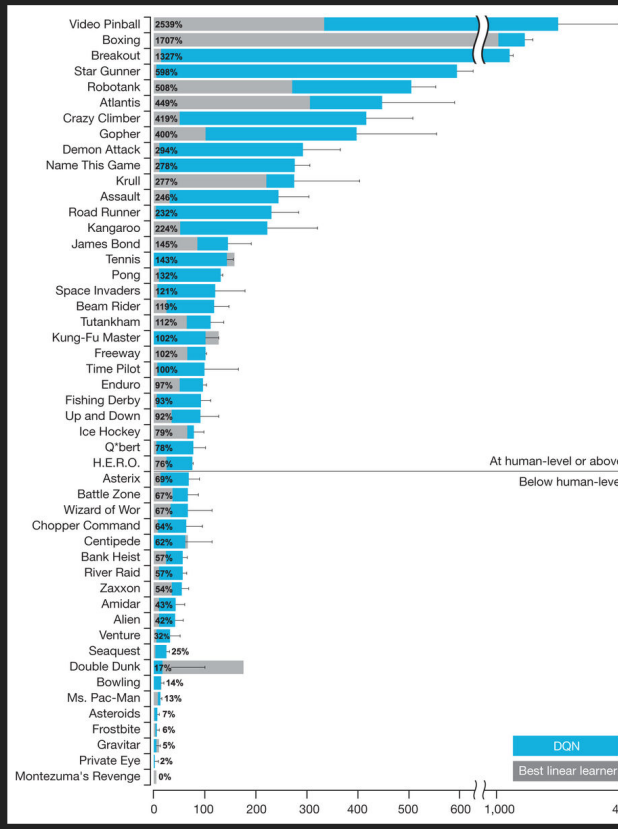
Q-FUNCTION LEARNING: DQN

The Q-function encodes how valuable actions in each state are

If we use a neural network to represent our $Q^{\pi}(s, a)$, we can do supervised learning using rewards we get from the MDP

This leads naturally to the Deep Q-Network algorithm, which was published by Google Deepmind in 2015

However, DQN is limited to discrete action spaces



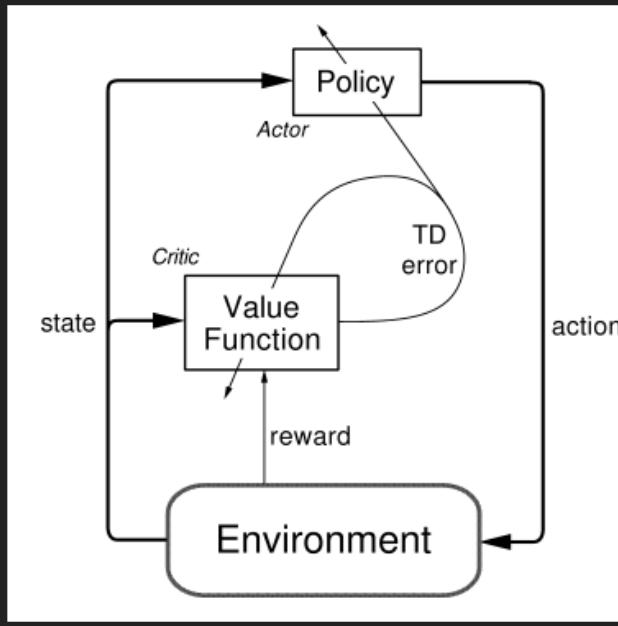
CONTINUOUS ACTION SPACES: DDPG

To deal with continuous action spaces, we need to represent our policy by a neural network

The Deep Deterministic Policy Gradients algorithms stems from this observation and naive "actor-critic" methods

DDPG is also a recent (2016) result out of Deepmind

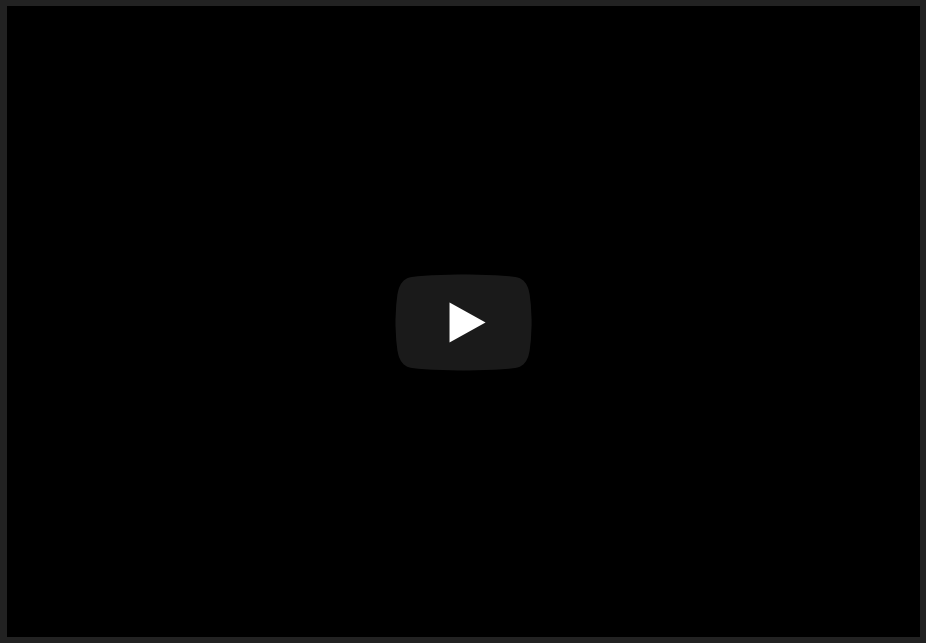
Now, we are finally (theoretically) capable of tackling robotic control tasks



TRACTABILITY THROUGH SIMPLIFICATION

Most research previous to 2015 in robotic reinforcement learning (RRL) simplified the task by modifying the action space

- Jens Kober's work into dynamical system motor primitives
- Physics model-based approaches such as CMU's inverted helicopter research
- Myriad discretizations of the state and action spaces

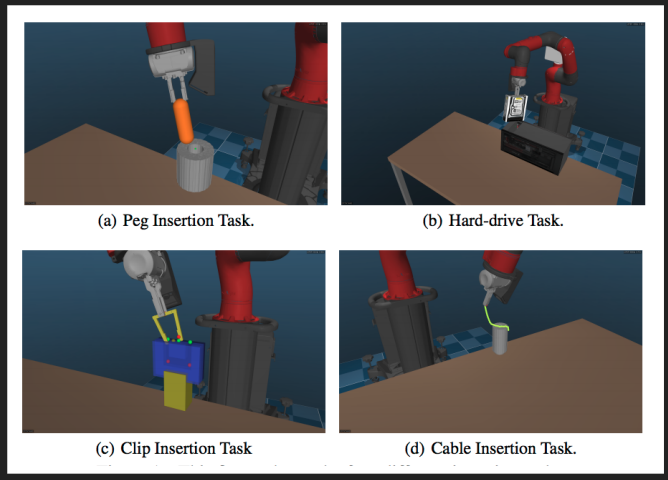


RECENT UC BERKELEY, DEEPMIND WORK

Groups at UC Berkeley and Google Deepmind have demonstrated success in using Deep RL for robotic tasks

However, this research still tends to simplify the state and action space involved by using a subset of the robot's joints

Even taking the reduced space, RRL can be intractable without human demonstration for initialization


(a) Peg Insertion Task. (b) Hard-drive Task.

(c) Clip Insertion Task (d) Cable Insertion Task.

CURRENT PROBLEMS

1. For more complex tasks, we need to be able to learn robot policies involving all joints in the robot
2. For difficult or dangerous tasks, it may not be feasible to provide human expert demonstration
3. Furthermore, the precise difficulty of RRL is not fully known right now
4. Experimental setups tend to be ad hoc, involve unreported engineering effort, and unreproducible

EXPLORATION

So far, we have avoided discussing exploration in RL

The "exploration-exploitation" tradeoff is a big deal in theoretical results for RL

In practice, most algorithms simply use some form of probabilistic noise as exploration

For high-dimensional spaces, such as RRL, this is likely not sufficient

MOTION PLANNING AS EXPLORATION

Recall that sampling-based motion planning allows us to generate motion plans for arbitrary robots

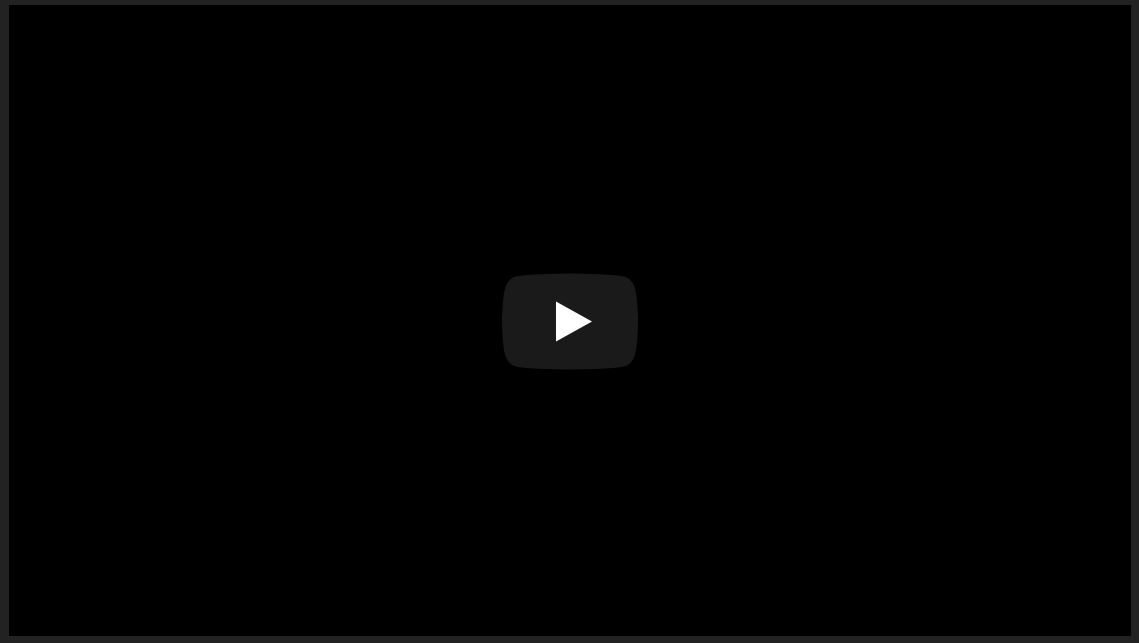
These motion plans are effectively structured, sustained exploration from the perspective of an RRL learner

If we can correctly hook motion planning for exploration into existing deep RL methods, it could replace human demonstration

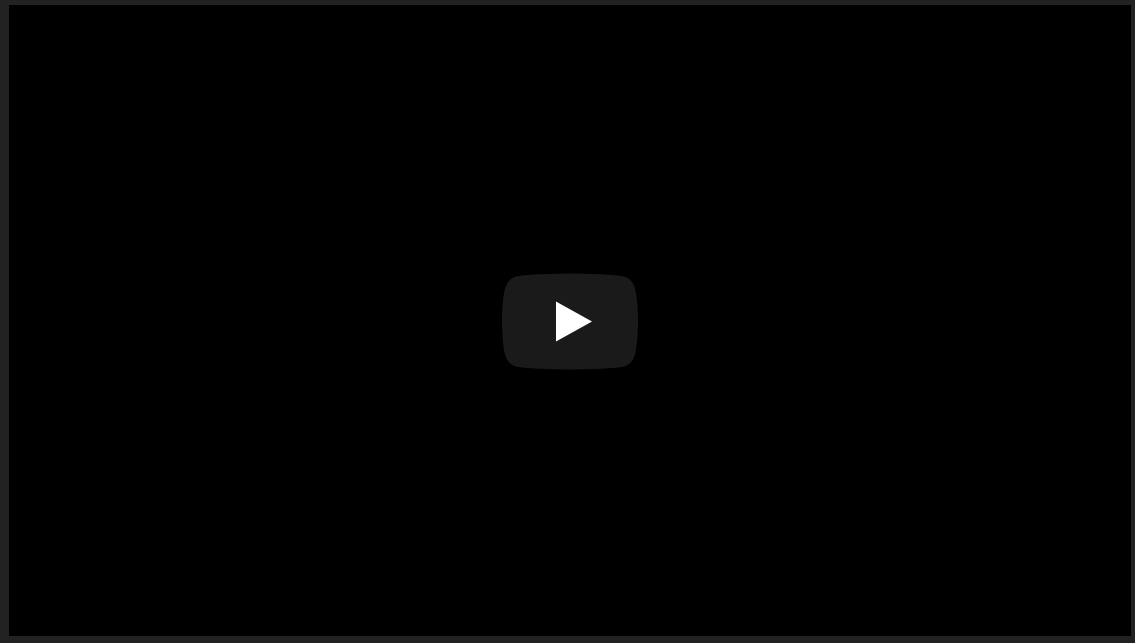
Furthermore, this allows RRL to be environment-agnostic, a goal of reinforcement learning

+

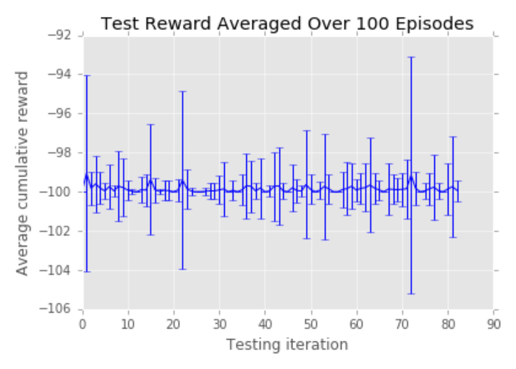
LEARNING ON A UR5 ROBOT

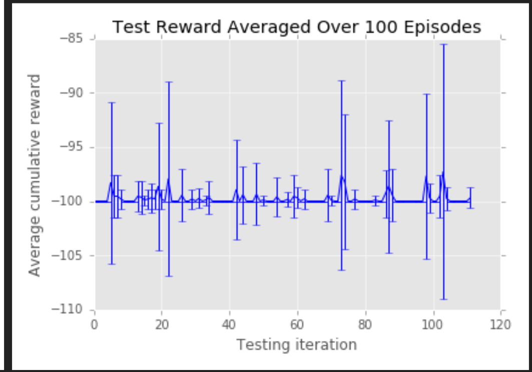


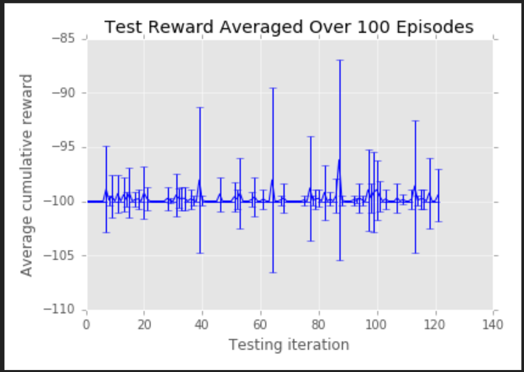
LEARNING ON A FETCH ROBOT

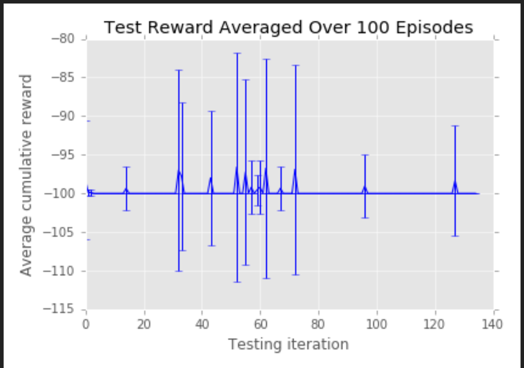


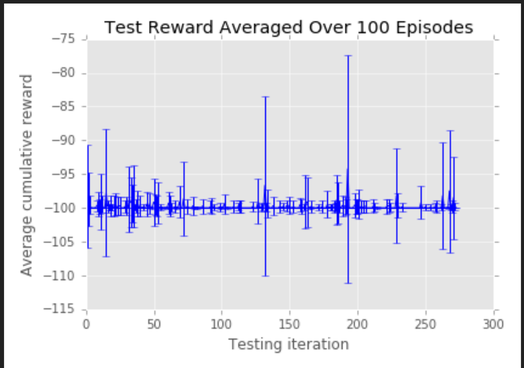
HYPERPARAMETER SEARCH IS DIFFICULT

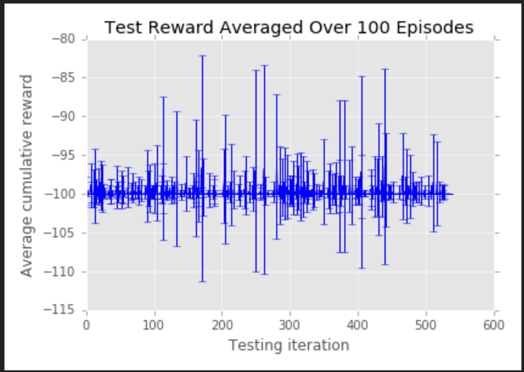












RRL EXPERIMENTATION IS DIFFICULT

As we have seen, many technical hurdles exist for the budding RRL researcher

- Many tools are not open-source or not well-maintained
- Myriad hyperparameters must be tuned
- It is not well known where the difficult problems lie
- Most robotics researchers do not have Google-scale compute resources

I intend to publish a paper soon dealing with the complexity of RRL for different numbers of joints

Much, much more research is left in this area

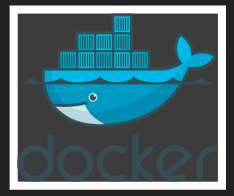
BUILDING AN OPEN-SOURCE STACK

I have used entirely open source tools

Built my own versions of proprietary tools for experiment running

Once I have published my research, I intend to open source my experiment code as well

Continued open-source collaboration is necessary to progress in CS research



A CASE AGAINST PROPRIETARY RESEARCH SYSTEMS

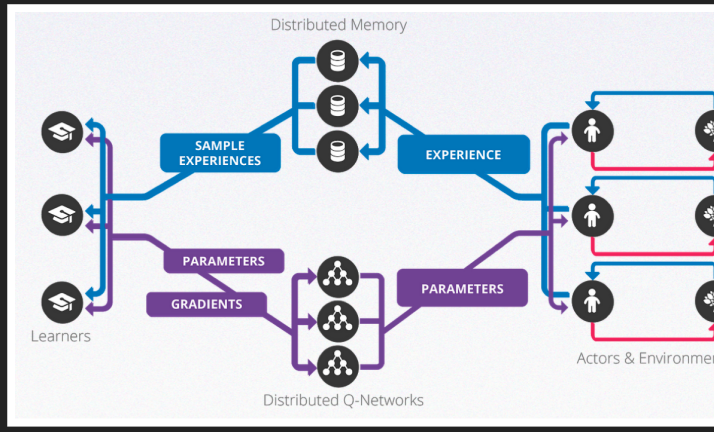
In this talk, I have mentioned Google Deepmind several times

One of their main research advantages is a distributed RL system that they call "Gorila"

Deepmind has not released this system design to the public

Proprietary advantages like this inhibit the progress of research in RL

I built a similar small distributed system using Docker and Amazon Web Services for my research



THANK YOU