ECE / CS 598 SG

Special Topics in Learning-based Robotics

Saurabh Gupta
Assistant Prof. (ECE, CS, CSL)
Today, we will…

- Course outline
- Course logistics
- Get to know each other
Understand how we can build intelligent machines

Does that mean game playing agents?

How Google's AlphaGo Beat a Go World Champion
Inside a man-versus-machine showdown

CHRISTOPHER MOYER  MARCH 28, 2016

The South Korean professional Go player Lee Sedol reviews the match after finishing against Google's artificial-intelligence program, AlphaGo. (LEE JIN-MAN / AP)
Understand how we can build intelligent machines

… that can favorably change the state of the *physical* world around them.

Does that mean factory robots?
Understand how we can build intelligent machines

... that can favorably change the state of the physical world around them.

Or these fun Boston Dynamics robots?

Video credit: Boston Dynamics, CNN
Understand how we can build intelligent machines

... that can favorably change the state of cluttered real world environments to solve a variety of tasks.

Household Robots

Understand how far are we from making this PRI showcase a reality.

Video credit: Pieter Abbeel
What can or can’t robots do today?

| Dexterous robot hands generally available. | NET 2030                                |
|                                         | BY 2040 (I hope!)                        |

A robot that can navigate around just about any US home, with its steps, its clutter, its narrow pathways between furniture, etc.  

| Lab demo: NET 2026          | Expensive product: NET 2030 | Affordable product: NET 2035 |

A robot that can carry out the last 10 yards of delivery, getting from a vehicle into a house and putting the package inside the front door.  

| Lab demo: NET 2025          | Deployed systems: NET 2028   |

A robot that seems as intelligent, as attentive, and as faithful, as a dog.  

| NET 2048                  |

A robot that has any real idea about its own existence, or the existence of humans in the way that a six year old understands humans.  

| NIML                     |

Goals of the Course

• Understand state-of-the-art in robotics and robot learning
Successes in Computer Vision “in the Wild”

Image Labeling Tasks

person, motorcycle, car, chair

K. He et al. Mask R-CNN ICCV 2017
Successes in Computer Vision “in the Wild”

Shape and Pose Estimation for Objects and Humans

S. Goel et al. Shape and Viewpoint without Keypoints. ECCV 2020
A. Kanawaza et al. End-to-end Recovery of Human Shape and Pose. CVPR 2018
Successes in Computer Vision “in the Wild”

Image Generation

<table>
<thead>
<tr>
<th>TEXT PROMPT</th>
<th>an armchair in the shape of an avocado, an armchair imitating an avocado.</th>
</tr>
</thead>
</table>

Factors Leading to Success in Computer Vision

Big models trained on big datasets

Big datasets

Big models

Hand-designed models → End-to-end learned models

Even bigger datasets and models

A. Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012
J. Deng et al. ImageNet: A Large-Scale Hierarchical Image Database. CVPR 2009
A. Dosovitskiy et al. An Image is worth 16x16 words: Transformers for Image Recognition at Scale. ICLR 2021
Factors Leading to Success in Computer Vision

Big models trained on big datasets

Factors Leading to Success in NLP

Big models (transformers) trained on big datasets (Internet text)

Can large-scale learning enable robots to execute a variety of tasks in cluttered real-world environments?

T. Brown et al. Language Models are Few-shot Learners. NeurIPS 2020
Robotic Tasks

Navigation

Robot with a first person camera
Dropped into a novel environment
Navigate around

"Go 300 feet North, 400 feet East"
"Go Find a Chair"
Robotic Tasks

Manipulation
Typical Classical Robotics Pipeline

- Observations
- State Estimation
- Planning
- Low-level Controller
- Control

Slide adapted from S. Levine.
Typical Classical Robotics Pipeline

Slide adapted from S. Levine.
Typical Classical Robotics Pipeline

Observations → State Estimation → Planning → Low-level Controller → Control

*But why would learning be useful at all?*
Robot Navigation

Robot with a first person camera
Dropped into a novel environment
Navigate around

“Go 300 feet North, 400 feet East”
“Go Find a Chair”
Observations → State Estimation → Planning → Low-level Controller → Control

Hartley and Zisserman. 2000. Multiple View Geometry in Computer Vision


Video Credits: Mur-Artal et al., Palmieri et al.
Geometric 3D Reconstruction of the World

Unnecessary

Do we need to tediously reconstruct everything on this table?

Can’t speculate about space not directly observed.
Can’t exploit patterns in layout of indoor spaces.
Geometric 3D Reconstruction of the World

Can’t exploit patterns in layout of indoor spaces.
Learn to make good decisions from partial information

Simulator based on scans of Real World Environments

Simulate robot views and motion

Compute ground truth traversability

Armeni et al. CVPR 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces
Learn to make good decisions from partial information

![Diagram of differentiable planner and mapper](image)

- **Egomotion**
- **Differentiable Mapper**
- **Differentiable Planner**
- **Optimal Action**
- **Predicted Action**

Train with back-propagation

And, making such speculations helps!

<table>
<thead>
<tr>
<th></th>
<th>Goal reaching efficiency</th>
<th>% Area explored in 500 steps</th>
<th>Area explored alongside other adversarial agents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object Goal</td>
<td>Image Goal</td>
<td></td>
</tr>
<tr>
<td>Without Speculation</td>
<td>0.46</td>
<td>0.33</td>
<td>78.2</td>
</tr>
<tr>
<td>With Speculation</td>
<td>0.53</td>
<td>0.48</td>
<td>86.2</td>
</tr>
</tbody>
</table>
Agent can make predictions about its surroundings
Agent can make predictions about its surroundings
Agent can make predictions about its surroundings

Free Space
Agent can make predictions about its surroundings
Agent can make predictions about its surroundings
Legged Locomotion

Hard to analytically model the system
Learn a simulator and learn a policy within it

1. Stochastic rigid body modeling
2. Train actuator net with real data
3. Reinforcement learning in simulation
4. Deploy on the real system

Fig. 1. Creating a control policy. In the first step, we identify the physical parameters of the robot and estimate uncertainties in the identification. In the second step, we train an actuator net that models complex actuator/software dynamics. In the third step, we train a control policy using the models produced in the first two steps. In the fourth step, we deploy the trained policy directly on the physical system.

We use the hybrid simulator for training controllers via reinforcement learning (Fig. 1, step 3). The controller is represented by a multi-layer perceptron that takes as input the history of the robot's states and produces as output the joint position target. Specifying different reward functions for RL yields controllers for different tasks of interest.

The trained controller is then directly deployed on the physical system (Fig. 1, step 4). Unlike the existing model-based control approaches, our proposed method is computationally efficient at runtime. Inference of the simple network used in this work takes 25 µs on a single CPU thread, which corresponds to about 0.1% of the available onboard computational resources on the robot used in the experiments. This is in contrast to model-based control approaches that often require an external computer to operate at sufficient frequency.

We apply the presented methodology to learning several complex motor skills that are deployed on the physical quadruped. First, the controller enables the ANYmal robot to follow base velocity commands more accurately and energy-efficiently than the best previously existing controller running on the same hardware. Second, the controller makes the robot run faster than ever before, breaking the previous speed record of ANYmal by 25%. The controller can operate at the limits of the hardware and push performance to the maximum. Third, we learn a controller that manifests vastly different behaviors. Although these behaviors are trained separately, they share the same code base: only the high-level task description changes depending on the behavior. In contrast, most of the existing controllers are task-specific and have to be developed nearly from scratch for every new maneuver.

Control Performance at Test-time

Efficient test-time motion

Fig. 2. Quantitative evaluation of the learned locomotion controller.
(A) The discovered gait pattern for 1.0 m/s forward velocity command. The abbreviations stand for Left Front (LF) leg, Right Front (RF) leg, Left Hind (LH) leg, and Right Hind (RH) leg, respectively. (B) The accuracy of the base velocity tracking with our approach. (C)-(E) Comparison of the learned controller against the best existing controller, in terms of power efficiency, velocity error, and torque magnitude, given forward velocity commands of 0.25, 0.5, 0.75, and 1.0 m/s.

High-speed locomotion

In the previous section, we evaluated the generality and robustness of the learned controller. Now we focus on operating close to the limits of the hardware to reach the highest possible speed. The notion of high speed is in general hardware-dependent. There are some legged robots that are exceptional in this regard. Park et al. [44] demonstrated full 3D legged locomotion at over 5.0 m/s with the MIT Cheetah. The Boston Dynamics WildCat has been reported to reach 8.5 m/s [45]. These robots are designed to run as fast as possible whereas ANYmal is designed to be robust, reliable, and versatile. The current speed record on ANYmal is 1.2 m/s and has been set using the flying trot gait [12]. Although this may not seem high, it is 50% faster than the previous speed record on the platform [39]. Such velocities are challenging to reach via conventional controller design while respecting all limits of the hardware.

We have used the presented methodology to train a high-speed locomotion controller. This controller was tested on the physical system by slowly increasing the commanded velocity to 1.6 m/s and lowering it to zero after 10 meters. The forward speed and joint velocities/torques are shown in Fig. 3. ANYmal reached 1.58 m/s in simulation and 1.5 m/s on the physical system when the command was set to 1.6 m/s. All speed values were computed by averaging over at least 3 gait cycles. The controller used both the maximum torque (40 Nm) and the maximum joint velocities (12 rad/s) on the physical system as shown in Fig. 3B and 3C. This shows that the learned policy can exploit the full capacity of the hardware to achieve the goal. For most existing methods, planning while accounting for the limitations of the hardware is very challenging, and executing the plan on the real system reliably is harder still. Even state-of-the-art methods [12, 46] cannot limit the actuation during planning due to limitations of their planning module. Modules in their controllers are not aware of the constraints in the later stages and, consequently, their outputs may not be realizable on the physical system.

The gait pattern produced by our learned high-speed controller, shown in Fig. 3D, is distinct from the one exhibited by the command-conditioned locomotion controller. It is close to a flying trot but with significantly longer flight phase and asymmetric flight phase duration. This is not a commonly observed gait pattern in nature and we suspect that it is among multiple near-optimal solution modes for this task. The behavior of the policy is illustrated in movie S6.

Recovery from a fall

Legged systems change contact points as they move and are thus prone to falling. If a legged robot falls and cannot autonomously restore itself to an upright configuration, a human operator must intervene. Autonomous recovery after a fall is thus highly desirable. One possibility is to represent recovery behaviors...
Control Performance at Test-time

Fig. S1. Base velocity tracking performance of the learned controller while following random commands. (A) Forward velocity, (B) Lateral velocity, (C) Yaw rate. For all graphs, the dotted lines represent the commanded velocity and the solid lines represent the measured velocity. All commands are followed with a reasonable accuracy even when the commands are given in a random fashion.

Fig. S2. Base velocity tracking performance of the best existing method while following random commands. (A) Forward velocity, (B) Lateral velocity, (C) Yaw rate. For all graphs, the dotted lines represent the commanded velocity and the solid lines represent the measured velocity. The tracking performance is significantly worse than the learned policy.

ANYmal Baseline
To this end, we train a neural network representing this complex dynamics with data from the real robot.
Grasping

Partial observation of environment, hard to model agent-object interaction
and execute grasp at the corresponding grasp location and select the maximum score across all angles and all patches, into the CNN. For each patch, the output is 18 values which be thought of an 18-way binary classification problem. Is graspable at angle of the gripper, and therefore an image patch can be not and then selects the grasp angle for positive patches.

There are multiple grasp locations for each object; (b) CNNs are on the image, to include context as well. The patch size around the grasp point. For our experiments, we use patches Input:

models from Dex-Net 1.0: 1,371 synthetic models from

variant of the pose error robust metric [56] that includes model grasp success as:

TABLE I: Details of the distributions used in the Dex-Net 2.0 graphical model

(1) Rendered Depth Images. (2) Robust Parallel-Jaw Grasps. (3) Grasp Image Dataset (6.7 Million)

Positive

Negative

L. Pinto et al. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. ICRA 2016
Large-scale dataset of grasp outcomes

<table>
<thead>
<tr>
<th></th>
<th>IGQ</th>
<th>REG</th>
<th>GQ-Adv-Phys</th>
<th>GQ-Adv</th>
<th>GQ-S</th>
<th>GQ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Success Rate (%)</strong></td>
<td>60±13</td>
<td>52±14</td>
<td>68±13</td>
<td>74±12</td>
<td>72±12</td>
<td>80±11</td>
</tr>
<tr>
<td><strong>Precision (%)</strong></td>
<td>N/A</td>
<td>N/A</td>
<td>68</td>
<td>87</td>
<td>92</td>
<td>100</td>
</tr>
<tr>
<td><strong>Robust Grasp Rate (%)</strong></td>
<td>N/A</td>
<td>N/A</td>
<td>100</td>
<td>30</td>
<td>48</td>
<td>58</td>
</tr>
<tr>
<td><strong>Planning Time (sec)</strong></td>
<td>1.8</td>
<td>3.4</td>
<td>0.7</td>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Heuristic</th>
<th>Learning based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min eigenvalue</td>
<td>kNN</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.534</td>
<td>0.694</td>
</tr>
</tbody>
</table>

Why Learning?

- Environments are only partially observed
- Complex systems that are hard to model
- Environments or environment-agent interactions can be hard to model
- …

How can we use learning in robotic systems?
Learning in Computer Vision
Robot Learning vs Visual Learning

• Supervision?
• More than one answer
• Delayed
• Sustainable source of supervision
• Non-stationarity
• Exploration vs exploitation
Formalism for Modeling Behavior

Reinforcement Learning
Markov Decision Process

Transition Function
\[ p(s_{t+1} \mid s_t, a_t) \]

Reward Function
\[ r_t = R(s_{t+1}, s_t, a_t) \]

Goal
\[ \text{argmax}_{a_0, \ldots, a_T} \sum_t \gamma^t r_t \]
Goals of the Course

• Understand state-of-the-art in robotics and robot learning
• Formulate robot learning problems as MDPs
Challenges with Markov Decision Process

$O_t$  $a_t$  $O_{t+1}$

Step Back

3D Relative Pose

Transition Function
How you move, how the tiger moves?

Reward Function
Survived?

Need to live many, many lives to learn how to live.
Credit assignment problem in RL

Yann LeCun’s Cake
Alternatives to Solving MDPs

Pieter Abbeel’s Cake

Solve a Related but Supervision-rich Problem

S. Levine et al. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. ISER 2017.
Build Models and Plan with Them

PILCO - Inverting a pendulum
Build Models and Plan with Them

PILCO - Inverting a pendulum

Learn by Imitating Experts

Learn by Observing Experts

Think about going to the airport.

<table>
<thead>
<tr>
<th>Request Uber</th>
<th>Wait for Uber</th>
<th>Take Uber to airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>FB</td>
<td>Check</td>
</tr>
<tr>
<td>Dest.</td>
<td>Get Into Car</td>
<td>Talk to the Uber driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Get Off Car</td>
</tr>
</tbody>
</table>

- Request Uber
- Wait for Uber
- Take Uber to airport
- App
- Dest.
- FB
- Check
- Get Into Car
- Talk to the Uber driver
- Get Off Car

**tension in various muscles**

**time**
Course Outline

• Understand state-of-the-art in robotics and robot learning
• Formulate robot learning problems as MDPs
• Investigate alternative ways of solving MDPs
Course Outline

• Understand state-of-the-art in robotics and robot learning
• Formulate robot learning problems as MDPs
• Investigate alternative ways of solving MDPs
• Applying these techniques to solve robotic tasks
Typically, useful to incorporate problem-specific insights.

Goal (300, 400)

Mapper

Spatial Representation of the World

Planner

Neural Network

Locomotion: Combining with low-level control

Deep Drone Racing: Learning Agile Flight in Dynamic Environments
Kaufmann, et al. CoRL 2018
Manipulation: Use of specialized hardware

Learning to Grasp and Re-grasp using Vision and Touch
Calandra, et al. RAL 2018
Course Outline

• Understand state-of-the-art in robotics and robot learning
• Formulate robot learning problems as MDPs
• Investigate alternative ways of solving MDPs
• Applying these techniques to solve robotic tasks
• Perspectives
Perspectives

• Representations vs Behaviors
• Big Data vs Clever Algorithms
• Lessons from Cognitive Science, Psychology, Neuroscience
• ...
Course Outline

• Understand state-of-the-art in robotics and robot learning
• Formulate robot learning problems as MDPs
• Investigate alternative ways of solving MDPs
• Applying these techniques to solve robotic tasks
• Perspectives
Today, we will…

• Course outline
• Course logistics
• Get to know each other
Course Logistics

Instructor: Saurabh Gupta

TA: Aditya Prakash

http://saurabhg.web.illinois.edu/teaching/ece598sg/fa2022
Today, we will…

• Course outline
• Course logistics
• Get to know each other
Thank you