Learning robots and the ACM prize of Pieter Abbeel

Wouter Caarls
Departamento de Engenharia Elétrica, PUC-Rio

wouter@ele.puc-rio.br

Seminário PESC, 31 de agosto, 2022

Teaser

[Stanford Unviersity, 2007]

Pieter Abbeel

- Born in Antwerp, Belgium, 1977.
- MSc (electrical engineering), KU Leuven, 2000
- PhD (computer science), Stanford, 2008
- Professor, UC Berkeley
- Director of the Berkeley Robot Learning Lab
- Co-Director of the Berkeley Artificial Intelligence Research (BAIR) lab

Main research topics: robotics and machine learning with particular focus on deep reinforcement learning, deep imitation learning, deep unsupervised learning, meta-learning, learning-to-learn, and AI safety.

Pieter Abbeel

- Born in Antwerp, Belgium, 1977.
- MSc (electrical engineering), KU Leuven, 2000
- PhD (computer science), Stanford, 2008
- Professor, UC Berkeley
- Director of the Berkeley Robot Learning Lab
- Co-Director of the Berkeley Artificial Intelligence Research (BAIR) lab

Main research topics: robotics and machine learning with particular focus on deep reinforcement learning, deep imitation learning, deep unsupervised learning, meta-learning, learning-to-learn, and AI safety.

Outline

Robotics

Robot

1. a machine that resembles a living creature in being capable of moving independently (as by walking or rolling on wheels) and performing complex actions (such as grasping and moving objects)
2. a. device that automatically performs complicated, often repetitive tasks (as in an industrial assembly line)
   b. a mechanism guided by automatic controls

– Merriam-Webster, 2022

Robotics

Robot

1. a machine that resembles a living creature in being capable of moving independently (as by walking or rolling on wheels) and performing complex actions (such as grasping and moving objects)
2. a. device that automatically performs complicated, often repetitive tasks (as in an industrial assembly line)
   b. a mechanism guided by automatic controls

– Merriam-Webster, 2022

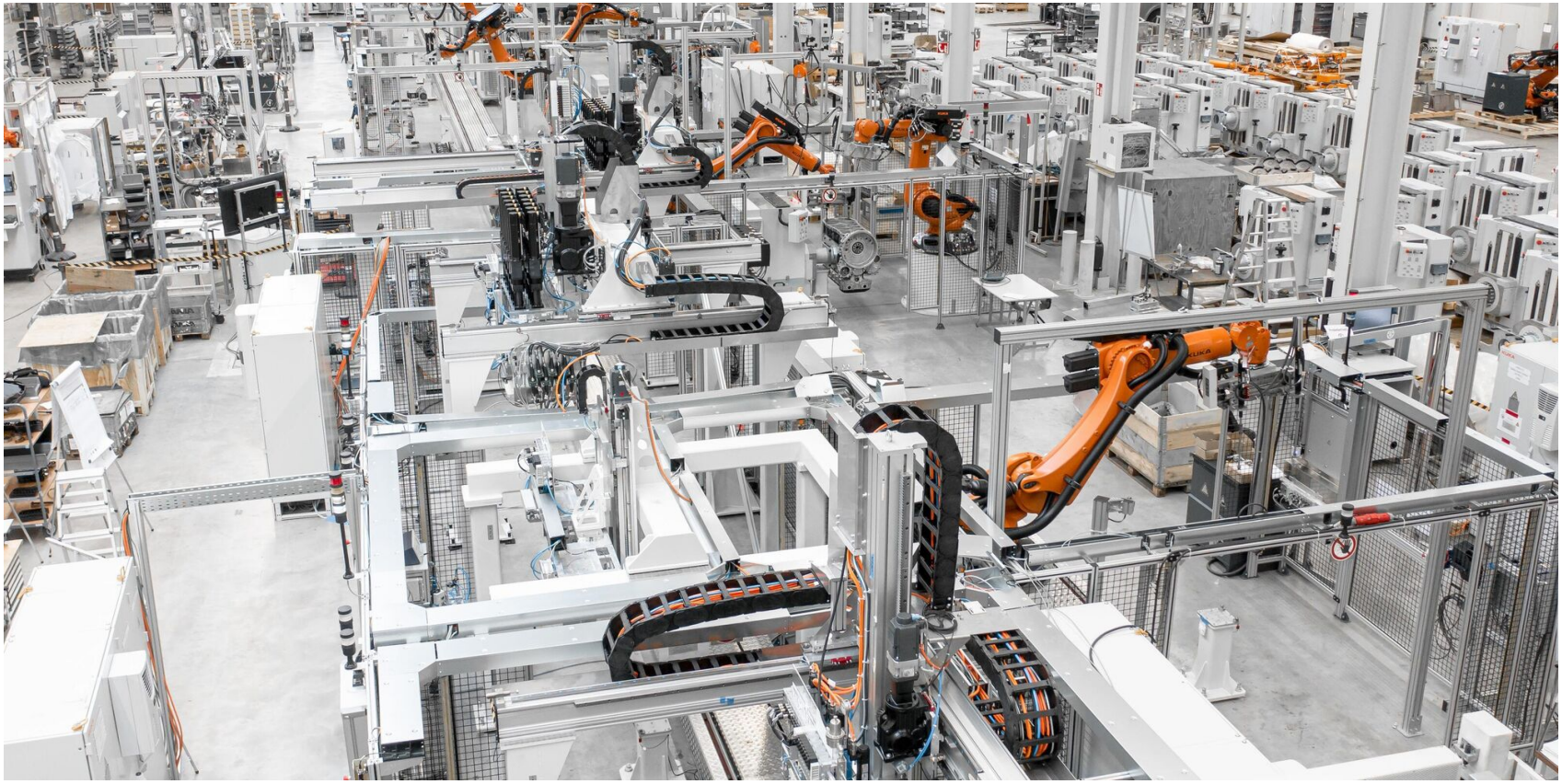
"I can't define a robot, but I know one when I see one."

– Joseph Engelberger, pioneer in industrial robotics

Humanoid robots

[Boston Dynamics, 2022]

Industrial robots



[KUKA, 2022]

Intelligent vehicles

[Tesla Motors, 2022]

Control

Robot

1. a machine that resembles a living creature in being capable of moving independently (as by walking or rolling on wheels) and performing complex actions (such as grasping and moving objects)
2. a. device that automatically performs complicated, often repetitive tasks (as in an industrial assembly line)
   b. a mechanism guided by automatic controls

– Merriam-Webster, 2022

The fact that robots perform complicated actions is part of the definition. Specifying *how* those actions are performed is called control.

Control

Open-loop

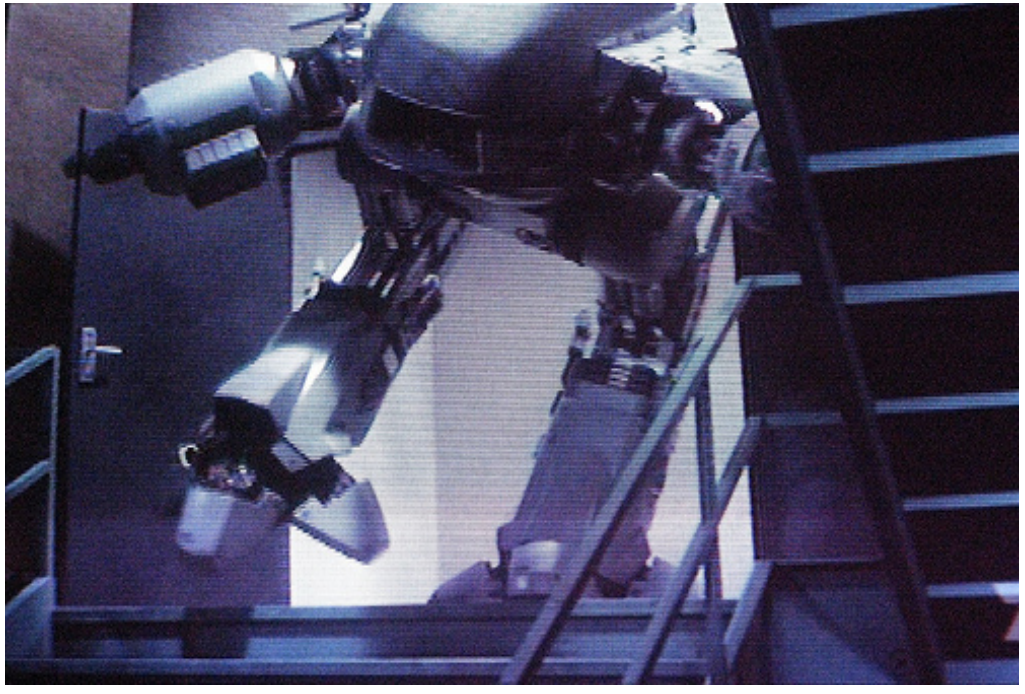
Many complicated actions can be specified in open-loop: the robot moves from position to position, actuating some end-effector when appropriate (grasping, welding, etc.).

[Haanstra, 1958]

Control

Closed-loop

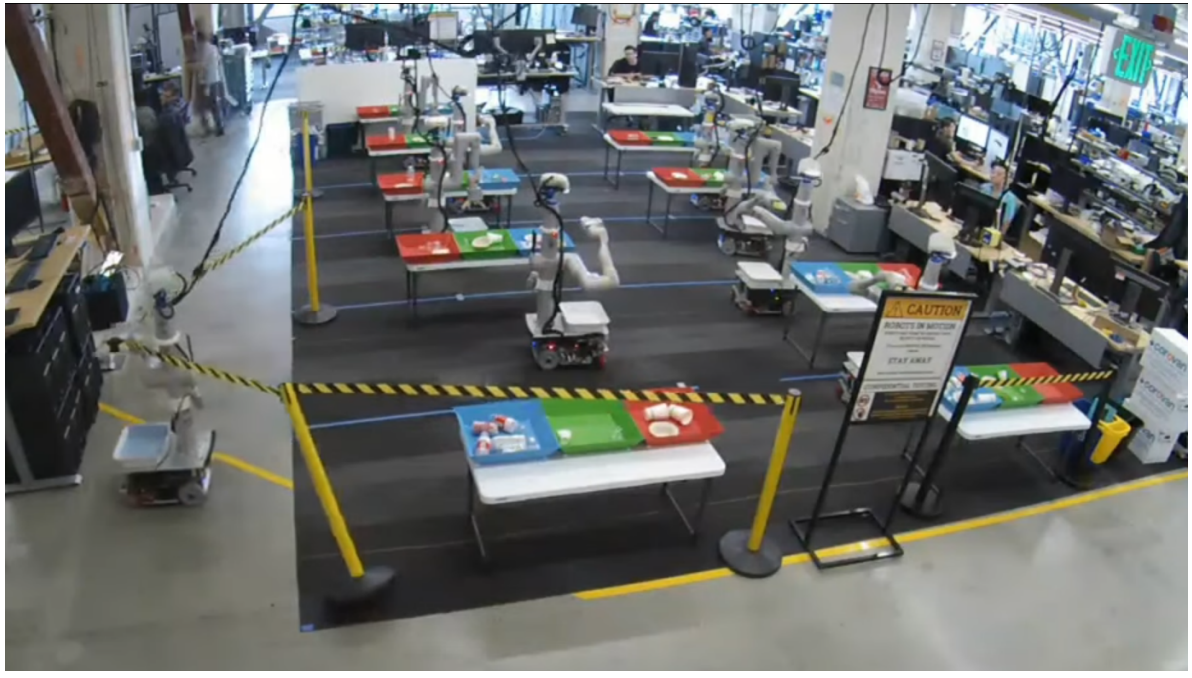
In more interesting scenarios, the system needs to react to the current state of the environment in a closed loop. This is possible if we can model the response of the environment to the controller's outputs.



[Verhoeven, 1987]

Control
Learning

If we don't have a model of the environment, it should be learned from data, or the controller should be derived without modeling, through trial and error.



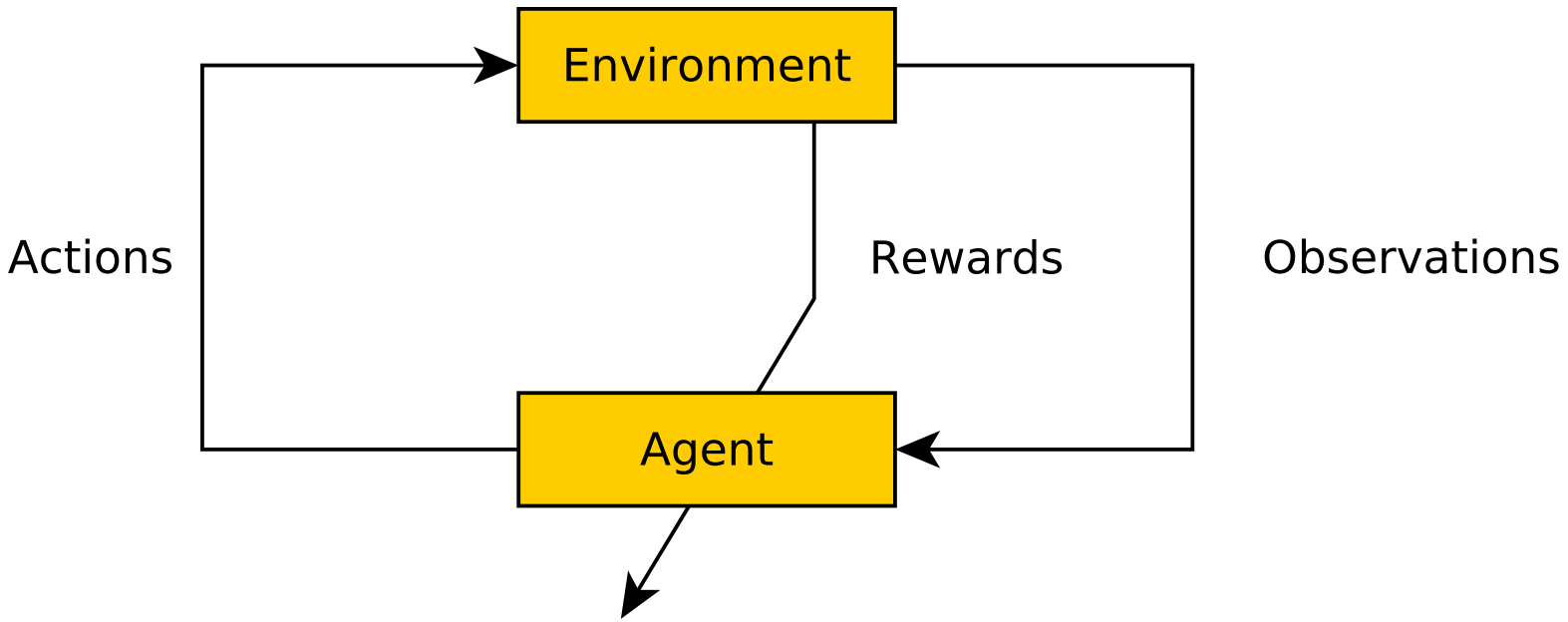
[Everyday Robots, 2021]

Reinforcement learning

[punchinell0, 2008]

Process

Reinforcement learning (RL) is optimizing a control policy by trial and error, through interaction with an environment.



Goal

Find actions that maximize the reward received over the lifetime of the agent.

Ingredients

Any reinforcement learning process must define

Environment (Stochastic) dynamical system the agent interacts with (e.g. robot + environment)

Observation Sensor readings that give information about the current state of the environment (e.g. camera image)

Agent Software that learns the control policy that maps states to actions

Action Actuation applied to the environment (e.g. motor torque)

Reward Signal being optimized by the agent (e.g. forward velocity, amount of dust)

Return

RL solves a sequential decision process: in every state, the action that should be taken is the one that maximizes the expected sum of future rewards (return):

$$\pi(s) = \arg\max_a \mathbb{E}\left[R_t | s_t = s, a_t = a\right]$$
$$= \arg\max_a \mathbb{E}\left[\sum_k r_{t+k+1} | s_t = s, a_t = a\right].$$

Most reinforcement learning algorithms try to estimate the return in a value function

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_k r_{t+k+1} | s_t = s, a_t = a\right].$$

Optimization

The optimal value function Q^* can be written recursively as

$$Q^*(s,a) = \mathbb{E}\left[r + \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\right].$$

To estimate this value function, samples of the right hand side of this equation are used to train an approximator $\hat{Q}$ by generating a training set

$$(s_t, a_t) \rightarrow (r_{t+1} + \max_{a'} \hat{Q}(s_{t+1}, a'))$$

and feeding it into a well-known machine learning algorithm.

Moving target

In RL, the training set depends on the approximator output. This makes it much harder than supervised learning.

Actor-critic algorithms

The policy can be derived from the value function as

$$\pi(s) = \arg\max_a Q(s, a),$$

but the arg max function requires a discrete action set to maximize over. In robotics, discrete actions lead to chattering. To avoid that, we can approximate the policy $\hat{\pi}$ directly and maximize

$$\hat{Q}(s_t, \hat{\pi}(s_t))$$

by adjusting the policy parameters in the direction of larger estimated expected returns. This is called actor-critic RL, where π is the actor and Q is the critic.

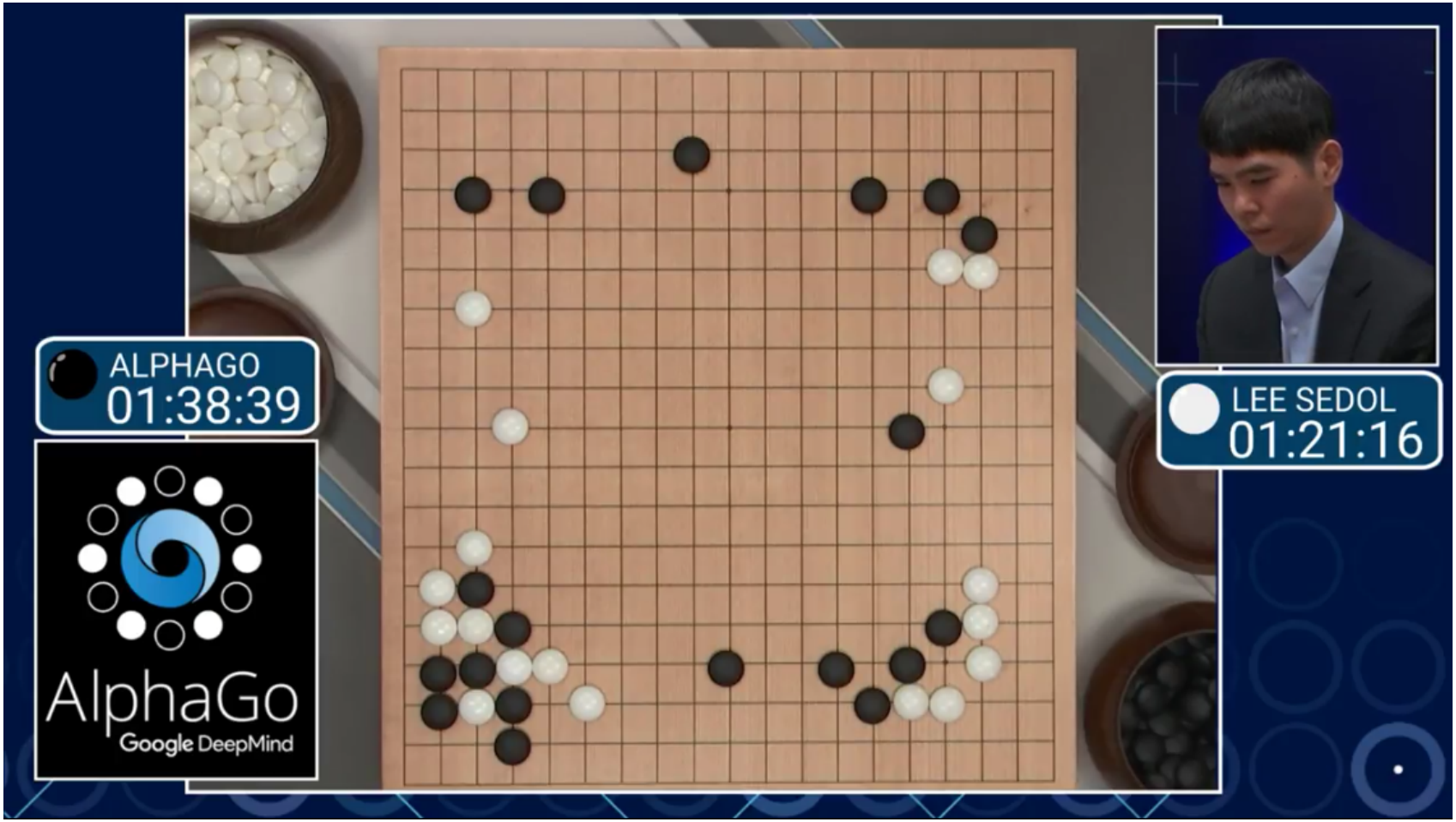
Game playing

A popular benchmark area for reinforcement learning algorithms is game playing. Some of its advantages over directly tackling robotics are

- The environment is precisely defined (the rules of the game / program code)
- Games can be played faster than real-time
- Many games can be played in parallel
- No possibility of breaking something through trial and error
- Performance can be directly compared with humans

Currently, the most popular games are board games (chess, Go) and Atari games (Pong, Qbert).

AlphaGo

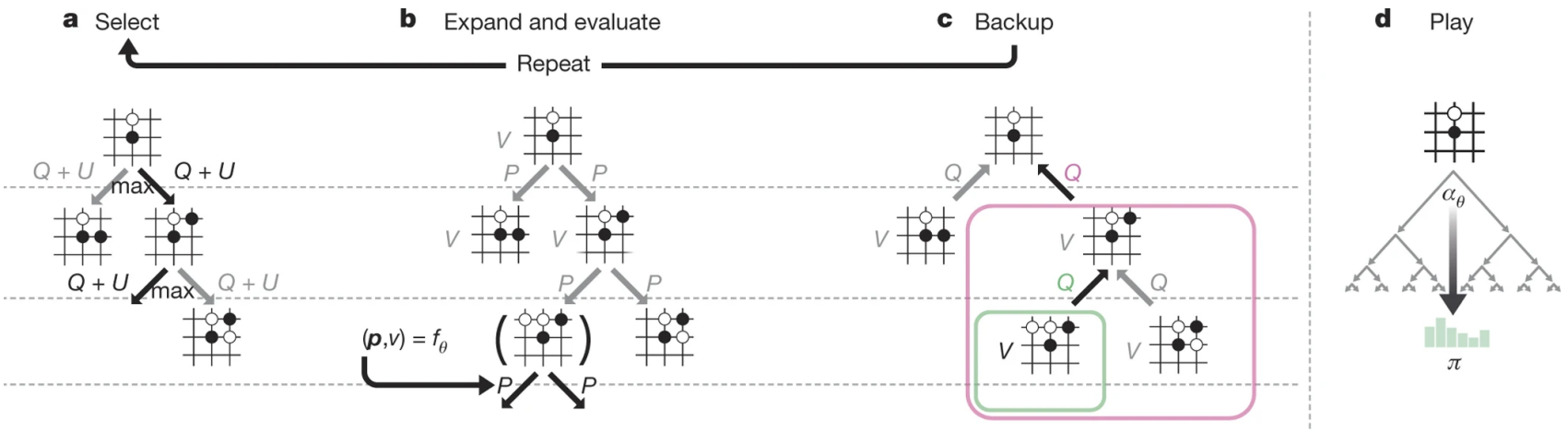


Silver et al., Mastering the game of Go with Deep Neural Networks & Tree Search, in: Nature, 2016

AlphaGo Zero

Action selection

AlphaGo selects actions using Monte Carlo tree search.

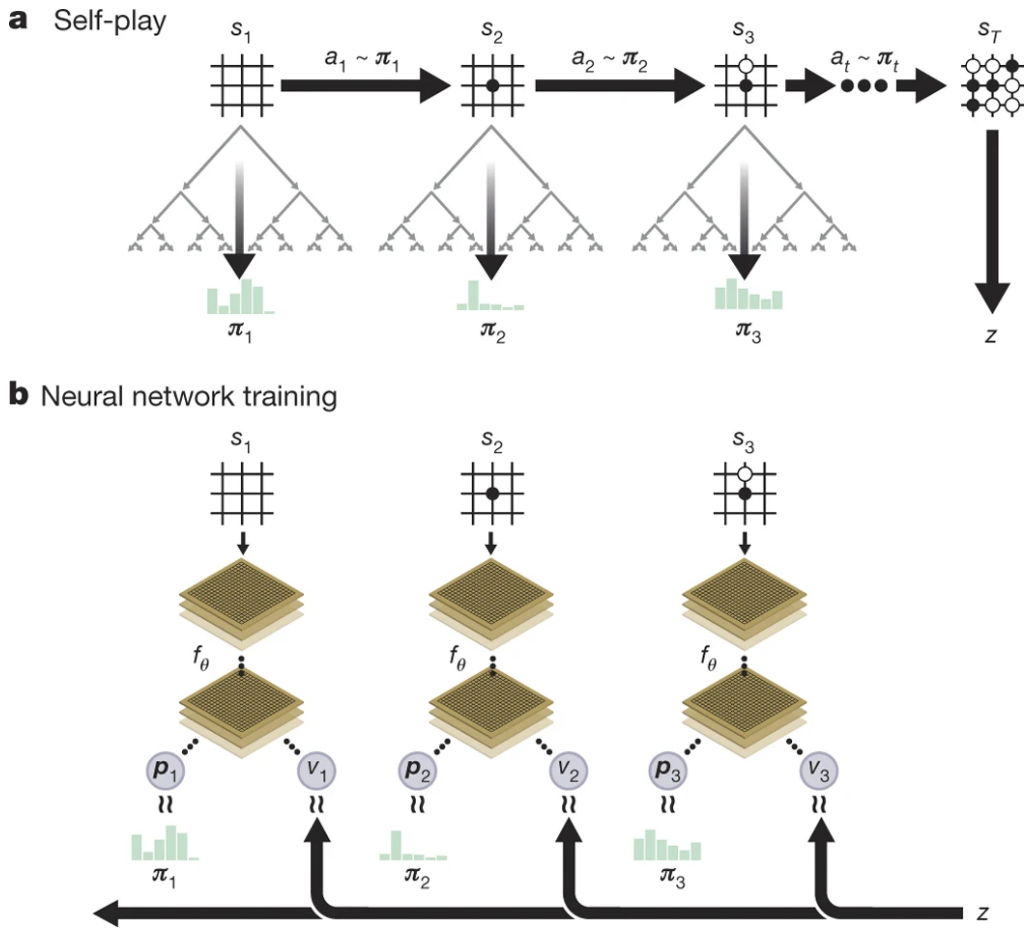


Starting from the current position, moves are selected sequentially by the policy and kept in a tree. When the end of the tree is reached, the value function evaluates the chance of winning from that position.

Silver et al., Mastering the game of Go without human knowledge, in: Nature, 2017

AlphaGo Zero

Training



Silver et al., Mastering the game of Go without human knowledge, in: Nature, 2017

EfficientZero

Improvements

AlphaGo Zero requires a known dynamics model. The latest version (MuZero[1]) uses model learning to avoid this. *No prior knowledge about the rules of the game is necessary*.
MuZero still requires many millions of games of self-play to reach superhuman level. EfficientZero[2] reduces this by

1. better training of the transition model (self-supervised consistency);
2. using the model to predict sums of rewards instead of singular rewards (reward prefix);
3. predicting value targets using imaginary experience (off-policy correction).

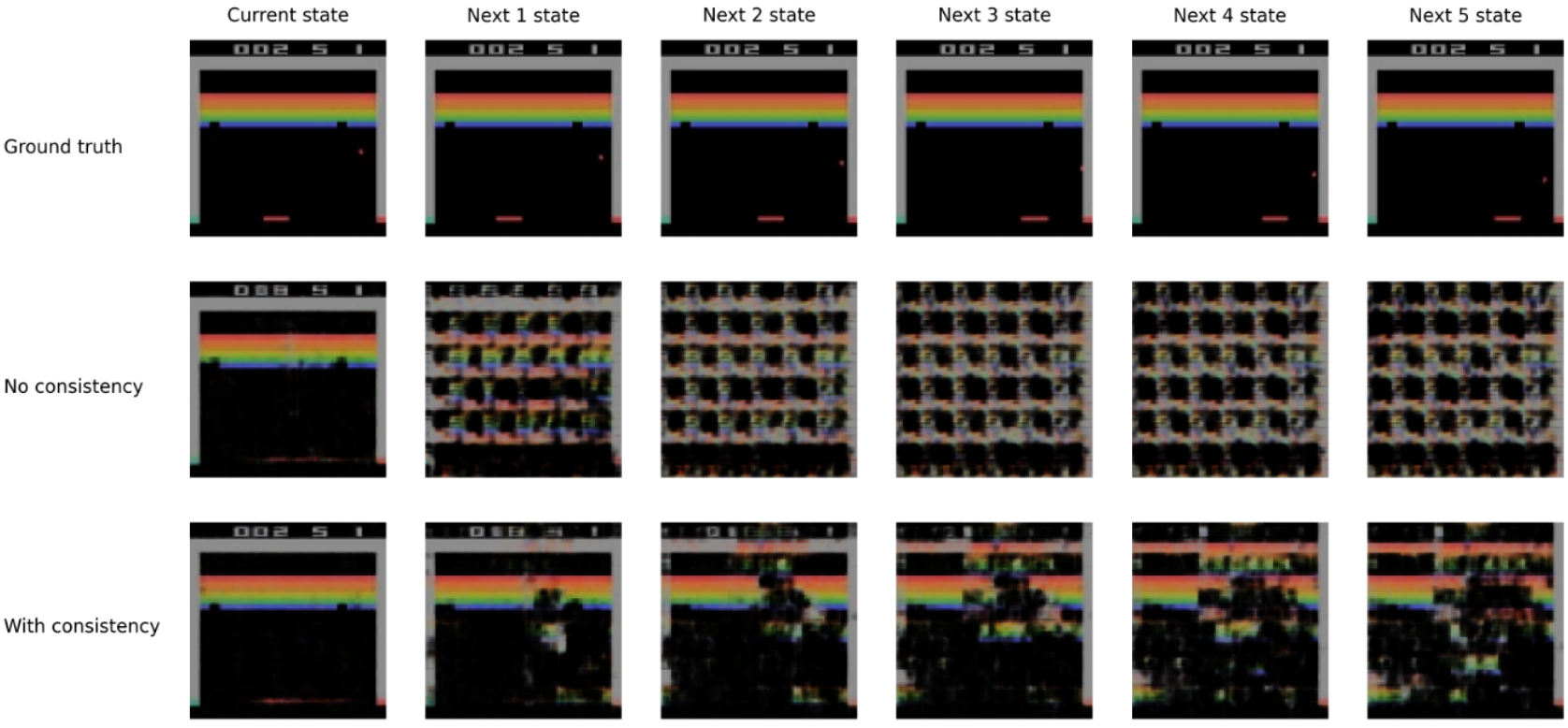
[1] Schrittwieser et al., Mastering Atari, Go, chess and shogi by planning with a learned model, in: Nature, 2020

[2] W. Ye, S. Liu, T. Kurutach, **P. Abbeel**, and Y. Gao, Mastering Atari Games with Limited Data, in: NeurIPS, 2021.

EfficientZero

Atari games

Game	Full	w.o. consistency	w.o. value prefix	w.o. off-policy correction
Normed Mean	**1.943**	0.881	1.482	1.475
Normed Median	**1.090**	0.340	0.552	0.836



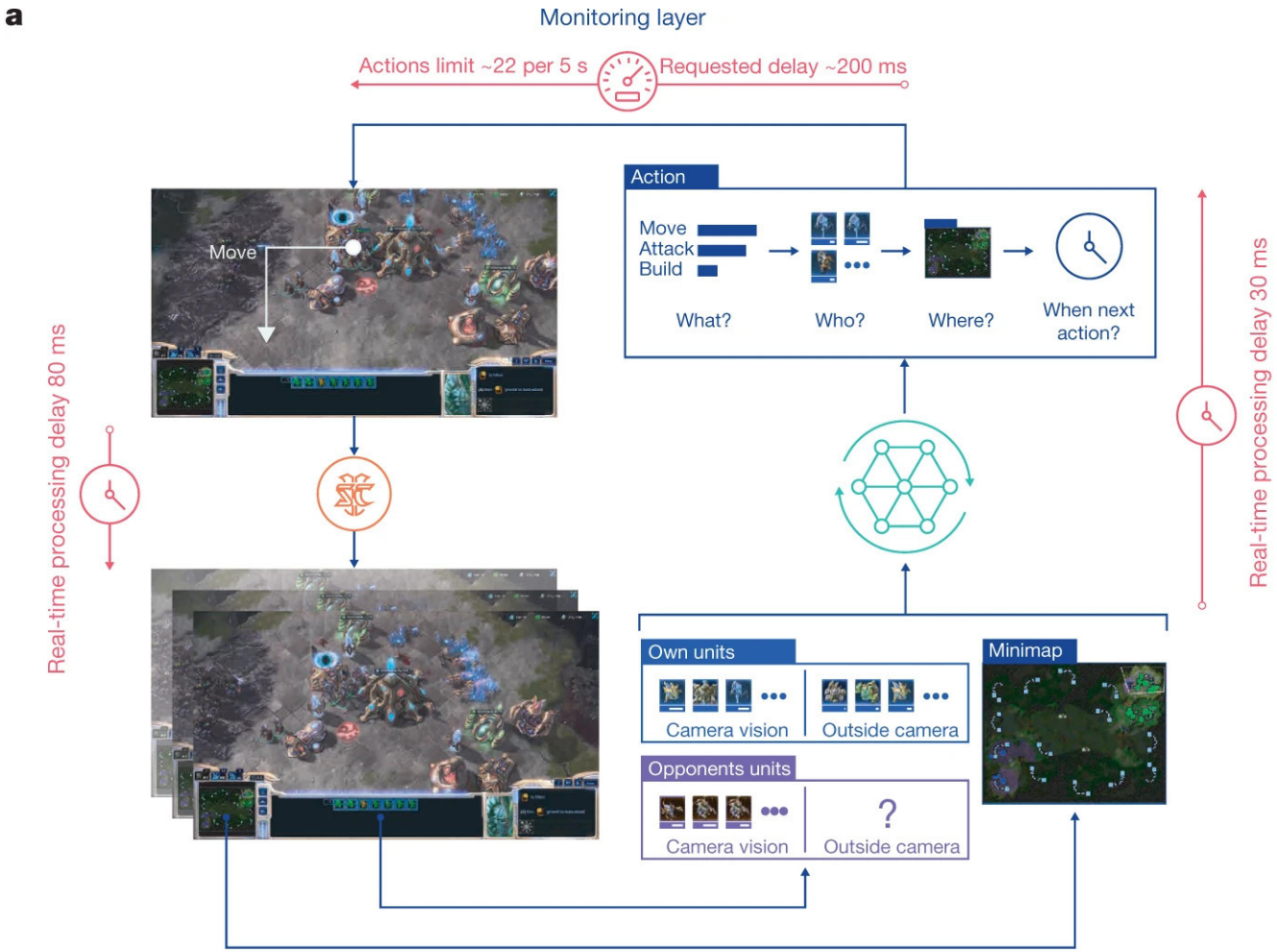
Partial observability

In contrast to Go, in many games and other systems the observations given by the environment are not enough to uniquely define the state of the system. Think of

- objects moving outside the screen, or outside the camera view;
- object velocities not being measured directly;
- object properties not being apparent from a camera image.

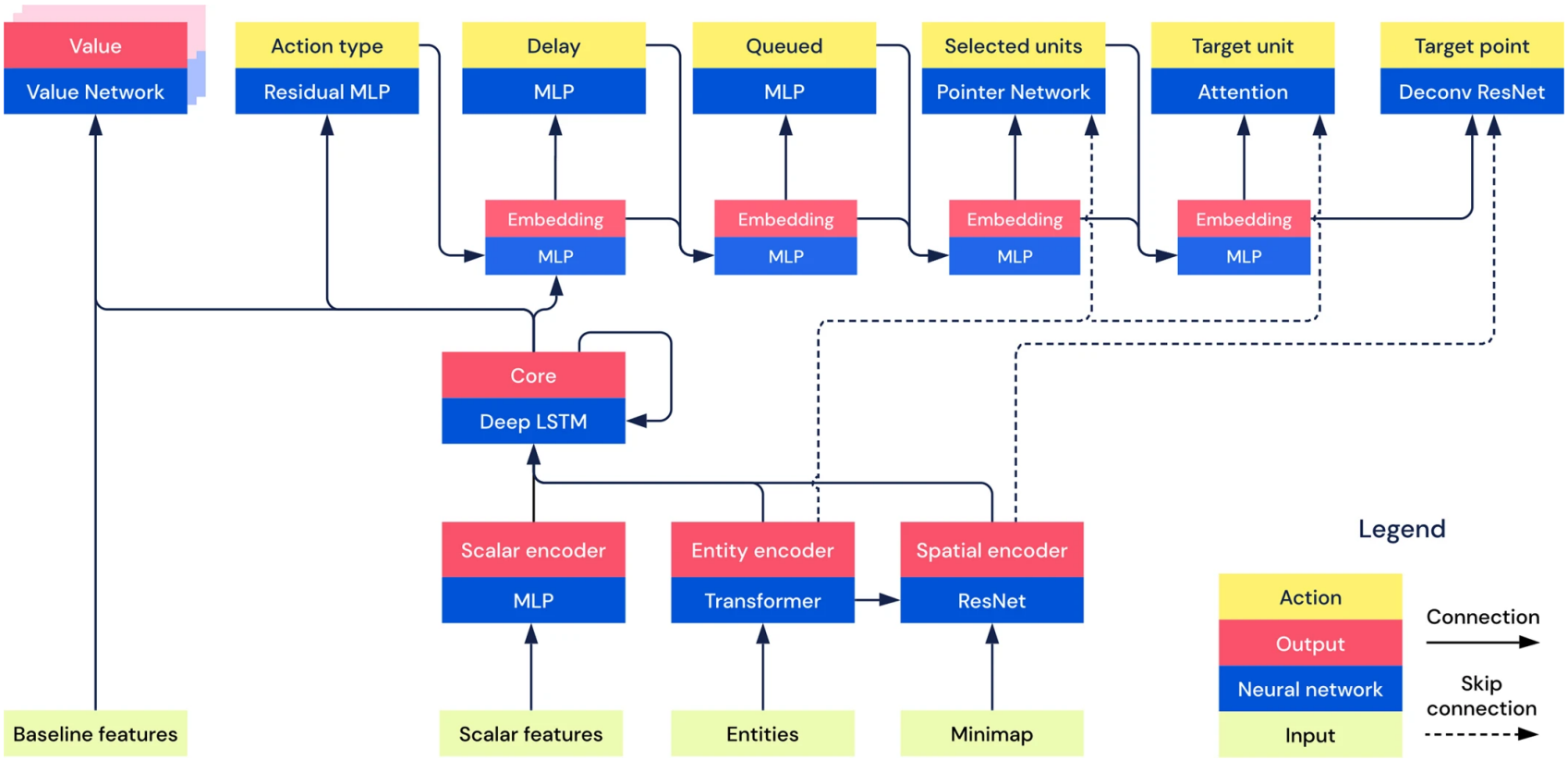
Such partial observability can be dealt with by concatenating observation sequences (short-term), or estimating a hidden state (long-term). The latter is usually done by approximating the value function and policy using a recurrent neural network.

AlphaStar



Vinyals et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, in: Nature, 2019

AlphaStar


Value
Value Network
Action type
Residual MLP
Delay
MLP
Queued
MLP
Selected units
Pointer Network
Target unit
Attention
Target point
Deconv ResNet
Embedding
MLP
Embedding
MLP
Embedding
MLP
Embedding
MLP
Core
Deep LSTM
Scalar encoder
MLP
Entity encoder
Transformer
Spatial encoder
ResNet
Baseline features
Scalar features
Entities
Minimap
Legend
Action
Output
Neural network
Input
Connection
Skip connection

Real-world systems

As opposed to games, real-world systems such as robots

- are only partially defined by the designer (robot), the rest is uncertain (environment);
- are limited to real-time (slow!);
- are expensive to run multiple copies of;
- can break through distructive actions or wear and tear.

As such, it is very important that the applied algorithms require few training episodes, and can deal with any environmental conditions.

Imitation learning

One way to reduce training time is through imitation learning, in which a human demonstrates the task to be performed. This can take various forms, such as:

- using the provided demonstration as a reference trajectory to track;
- initializing the policy with the actions taken by the human during demonstration, as a starting point for reinforcement learning;
- inferring the task from the demonstration, and using it for behavioral cloning or reinforcement learning.

Helicopter

The helicopter from the introduction used the reference trajectory method, solving three important problems:

Extracting a reference trajectory from noisy demonstrations

The demonstrations are very noisy, with differences in both position and timing. These need to be aligned in order to find a "mean"trajectory.

Learning a time-varying dynamics model

The observations (6d pose + 6d velocity) do not model the airflow, rotor speed, etc., and are therefore not a good basis for predicting the next state.

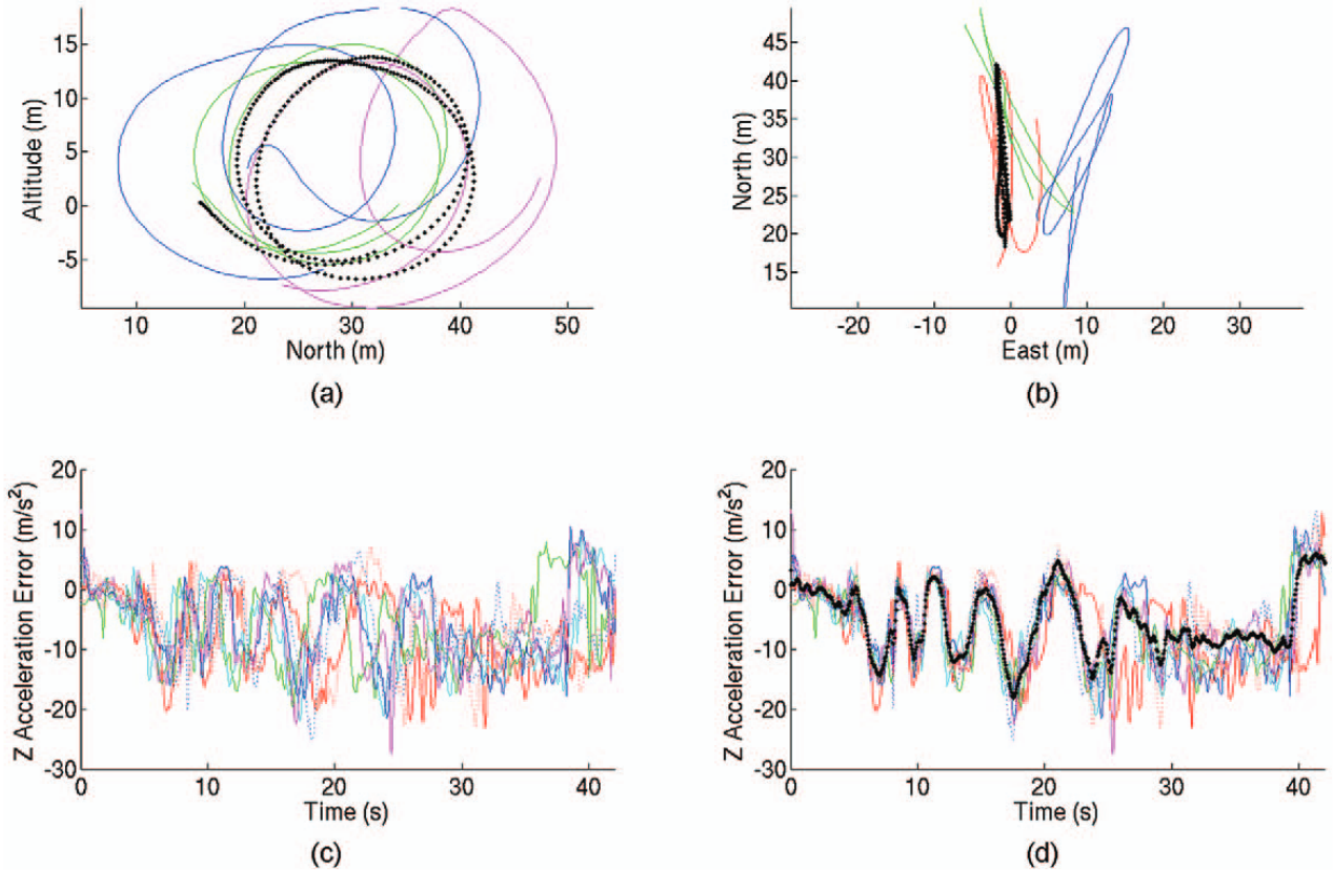
Tracking the reference trajectory

When wind pushes the helicopter off-trajectory, basic state-feedback controllers perform badly. Instead, a receding-horizon controller is used.

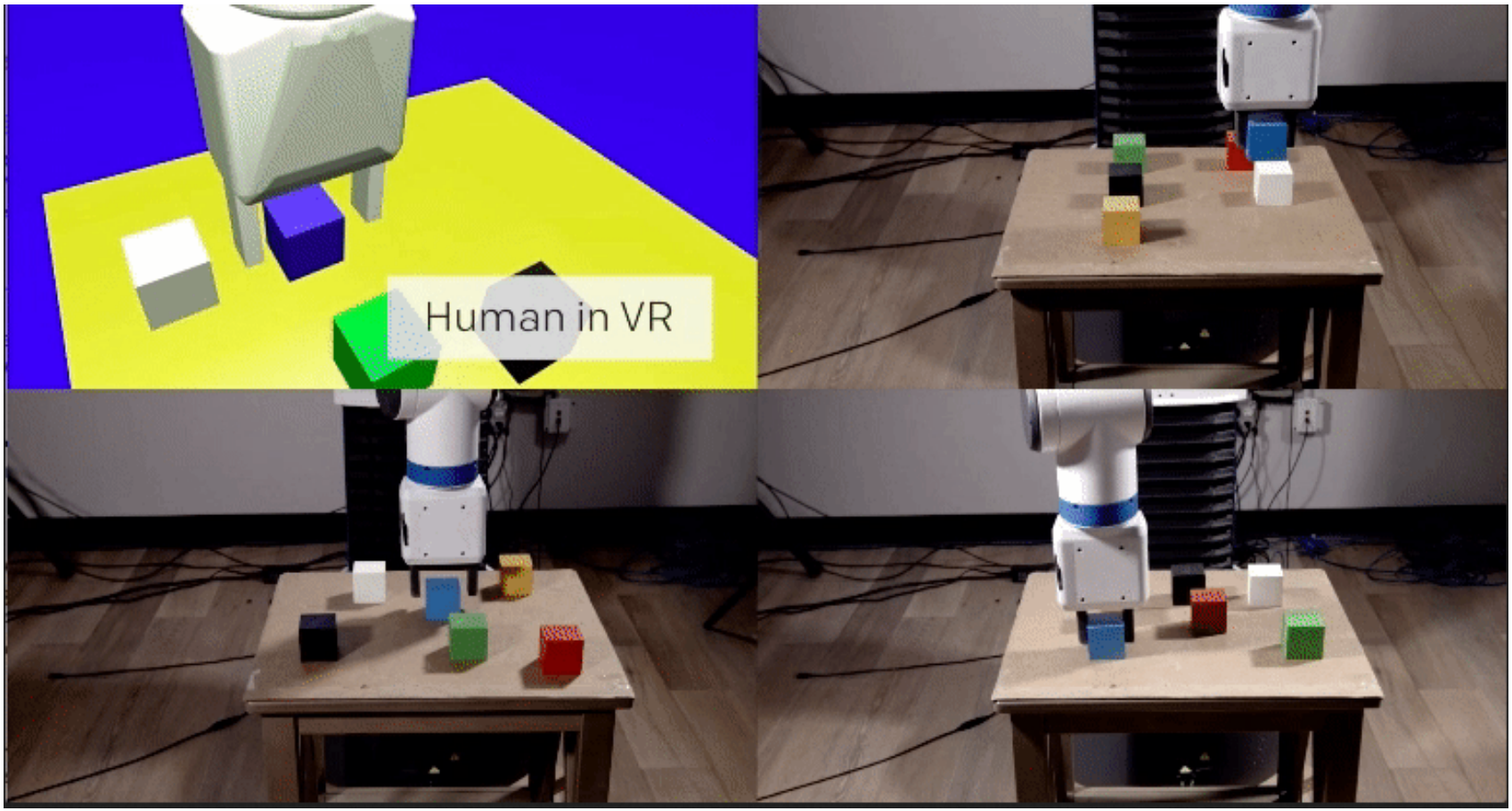
P. Abbeel, A. Coates, and A.Y. Ng, Autonomous Helicopter Aerobatics through Apprenticeship Learning, in: IJRR, 2010

Trajectory learning

The reference trajectory is found by assuming the demonstrations are all observations of a single, hidden trajectory. The most likely sequence of hidden states can by found by solving a Hidden Markov Model.



Block stacking



Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, **P. Abbeel**, and W. Zaremba, One-shot imitation learning, in: NIPS, 2017

Stochastic actor-critic

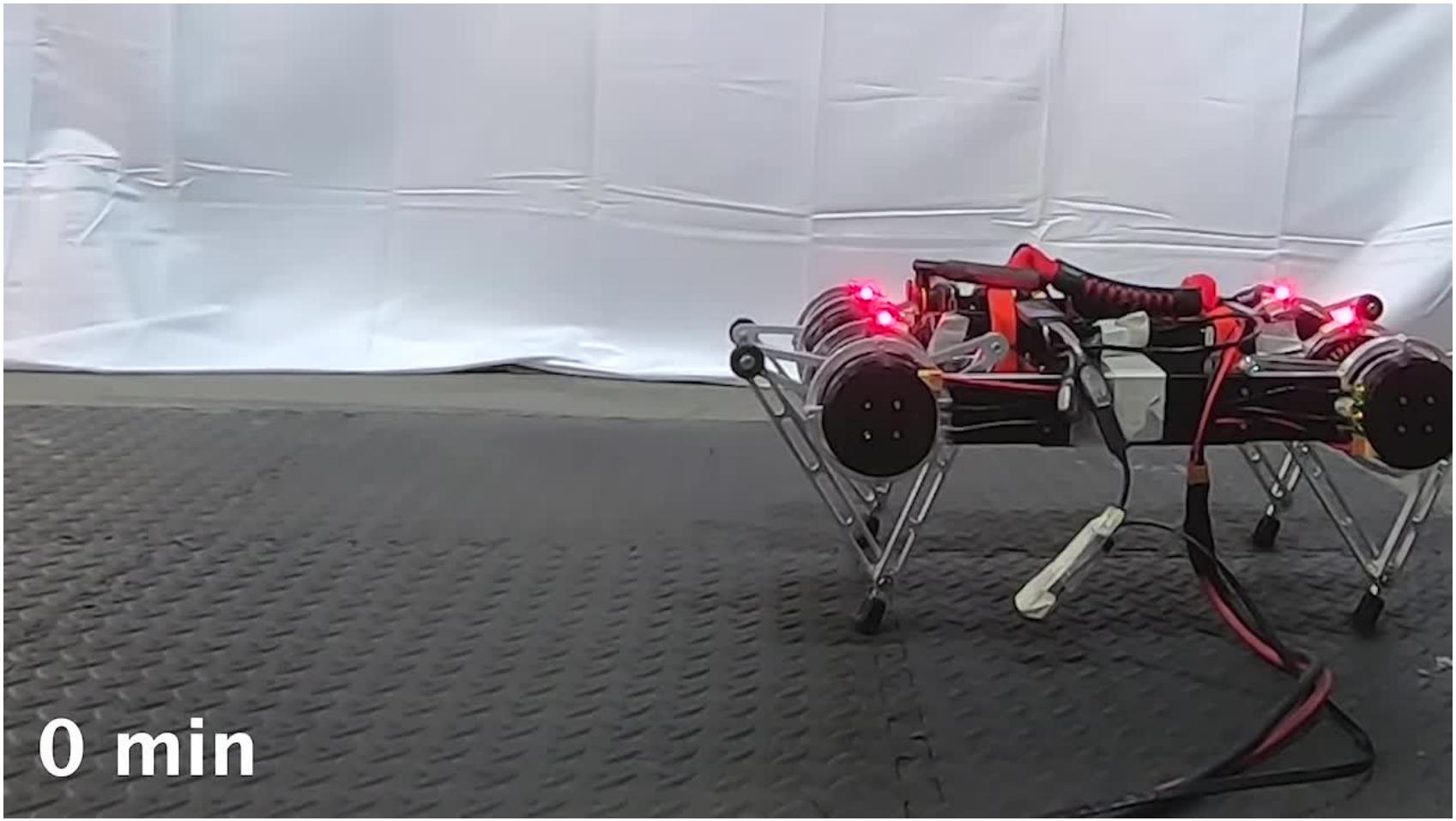
An important part of any reinforcement learning technique is ensuring sufficient exploration. Usually, this is done by adding noise to an otherwise deterministic control policy. In contrast, SAC[3] uses an explicitly stochastic actor $\pi(a|s)$ that uses an entropy augmented reward

$$r_t + \mathcal{H}(\pi(\cdot|s_t)).$$

As such, it learns to find the maximum return while acting as randomly as possible. The increased exploration greatly improves the robustness.

[3] T. Haarnoja, A. Zhou, **P. Abbeel**, and S. Levine, Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, in: ICML, 2018.

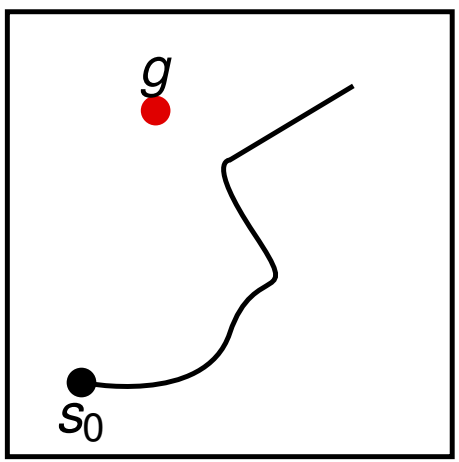
Learning to walk



T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, Learning to Walk via Deep Reinforcement Learning, in: RSS, 2021

Multiple goals

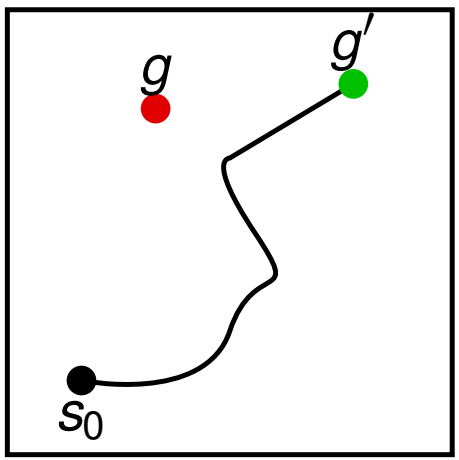
Many (robotic) tasks are goal-oriented: reach for an object, move to a certain position, etc. But as long as the goal is not reached, the agent has no feedback.



[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, **P. Abbeel**, and W. Zaremba, Hindsight Experience Replay, in: NIPS, 2017

Multiple goals

Many (robotic) tasks are goal-oriented: reach for an object, move to a certain position, etc. But as long as the goal is not reached, the agent has no feedback.

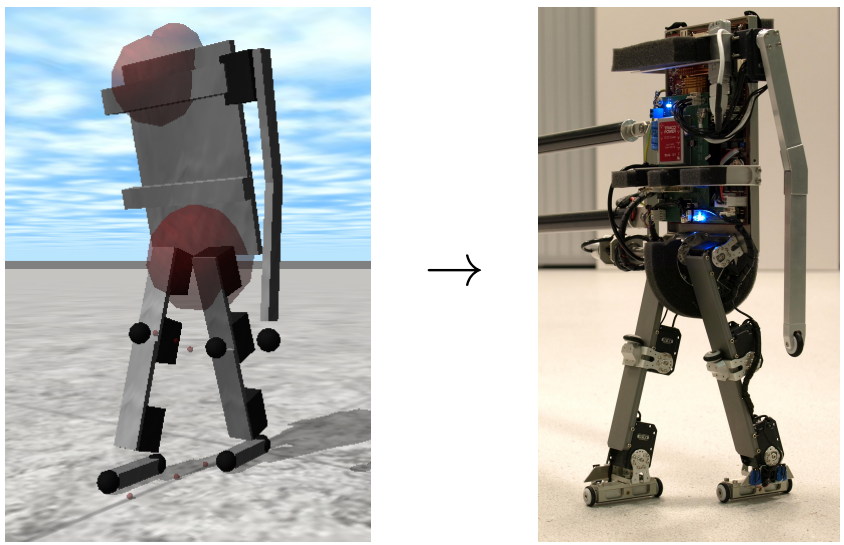


Hindsight Experience Replay[4] recognizes that some goal is always reached, and reinterprets the trajectory as if that is where it wanted to go all along.

[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, **P. Abbeel**, and W. Zaremba, Hindsight Experience Replay, in: NIPS, 2017

Pre-training

Even with state of the art algorithms, solving tasks from scratch in the real world still requires too many interactions. It is therefore common to pre-train in simulation, and then fine-tune in the real world.

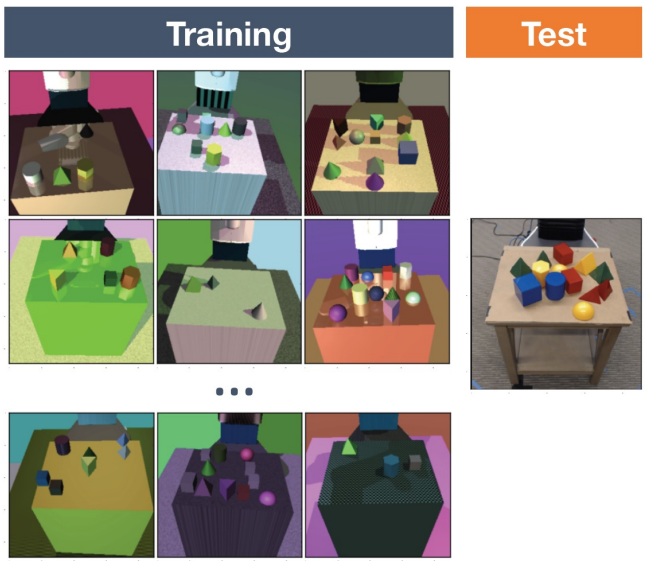


[Schuitema, 2010]

However, the reality gap often drastically reduces the performance while the policy moves towards a new local optimum.

Domain randomization

In domain randomization[5], the observations (camera images) generated by the simulator are randomly modified.

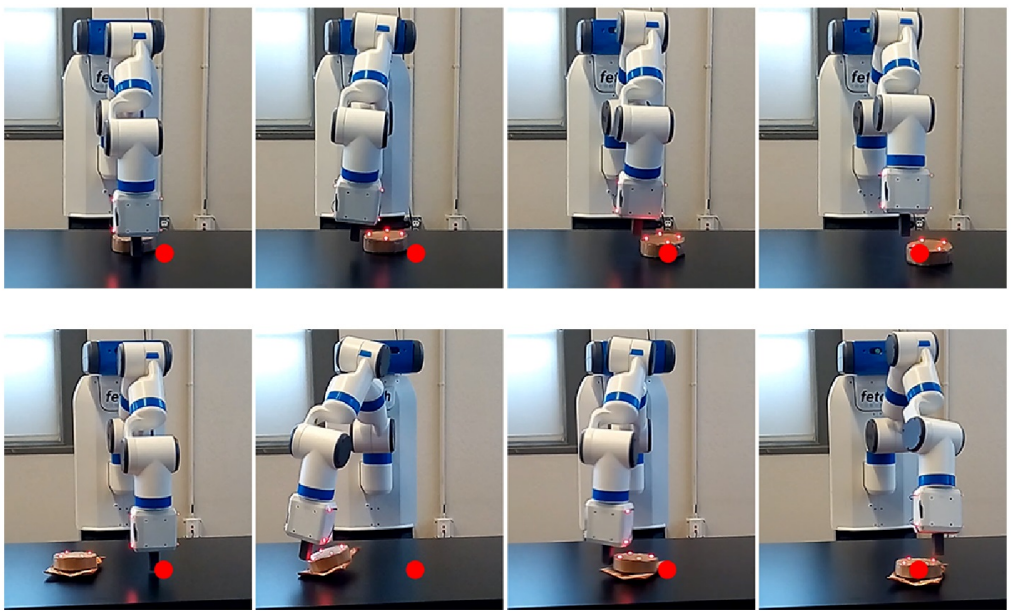


The intuition is to learn a policy that works in any of the simulated situations. If the real world falls within that distribution, it can be quickly adapted to.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and **P. Abbeel**, Domain randomization for transferring deep neural networks from simulation to the real world, in: IROS, 2017

Dynamics randomization

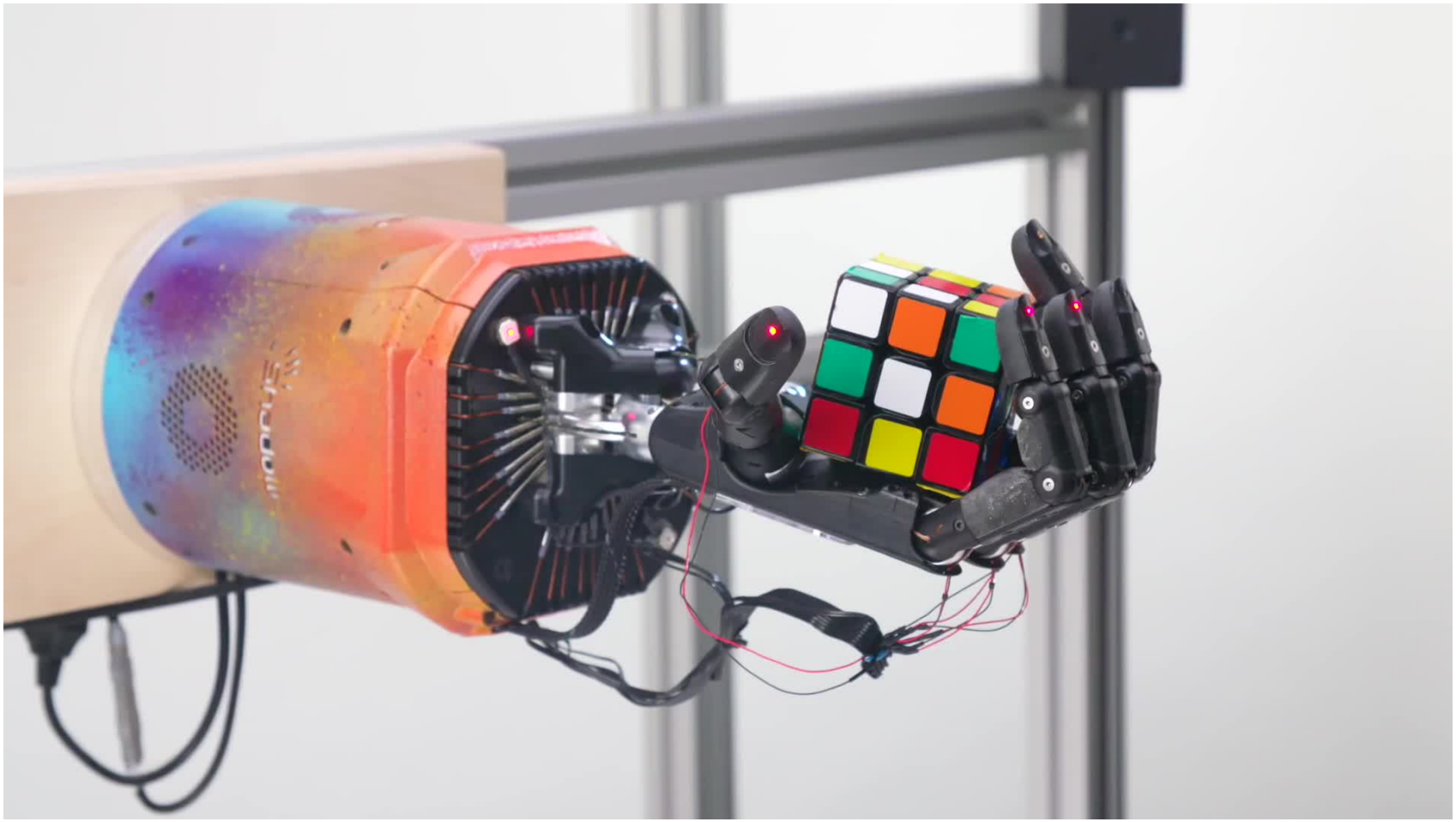
Similarly, in dynamics randomization[6] the dynamics (mass, damping, friction, etc.) of the system are randomized.



Additionally, using a recurrent network to estimate a hidden state, the system can adapt to the actual dynamics instead of learning an overly cautious policy.

[6] X. Bin Peng, M. Andrychowicz, W. Zaremba, and **P. Abbeel**, Sim-to-real transfer of robotic control with dynamics randomization, in: ICRA, 2018

Complex manipulation



OpenAI, Solving Rubik's Cube with a Robot Hand, 2019

Learning robots and the ACM prize of Pieter Abbeel

Wouter Caarls
Departamento de Engenharia Elétrica, PUC-Rio

wouter@ele.puc-rio.br

Seminário PESC, 31 de agosto, 2022