Neuroevolution for Deep Reinforcement Learning Problems

David Ha
Google Brain
Tokyo, Japan
hadavid@google.com

http://gecco-2019.sigevo.org/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-6748-6/19/07.
https://doi.org/10.1145/3319619.3323370

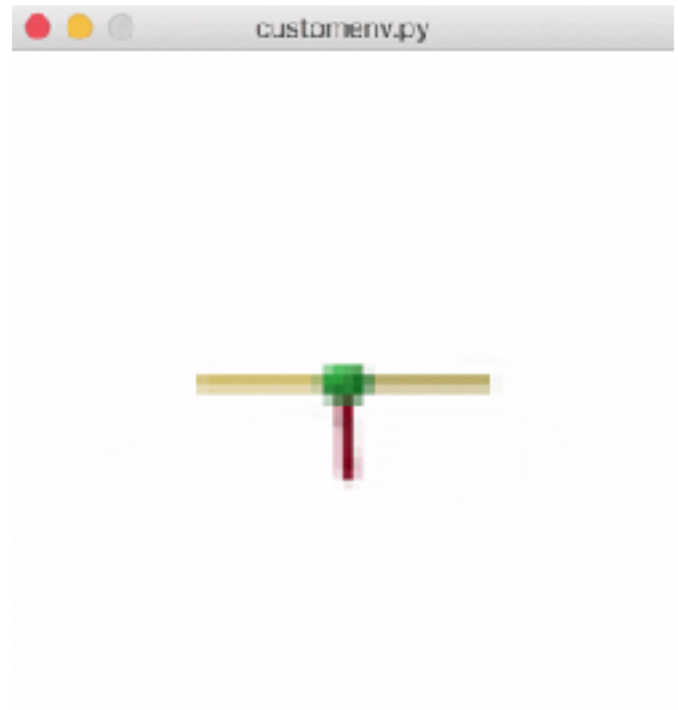
Instructor

- **David Ha** is currently a research scientist at Google Brain. His research interests include Recurrent Neural Networks, Creative AI, Evolutionary Computing, and Robotics. Prior to joining Google, He worked at Goldman Sachs as a Managing Director, where he co-ran the fixed-income trading business in Japan. He obtained undergraduate and graduate degrees in Engineering Science and Applied Math from the University of Toronto.

Tutorial Agenda

- Introduction
- Review of OpenAI Gym Environment
- Brief Review of Neuroevolution
- Neuroevolution for Continuous Control Tasks
  - Case Study: PyBullet and Roboschool Environments
  - Case Study: Bipedal Walker in OpenAI Gym
  - Case Study: Sim2Real Applications
- Brief Overview of Deep Generative Models
- World Models: Combine Generative Models with Evolution
- Questions & Discussion

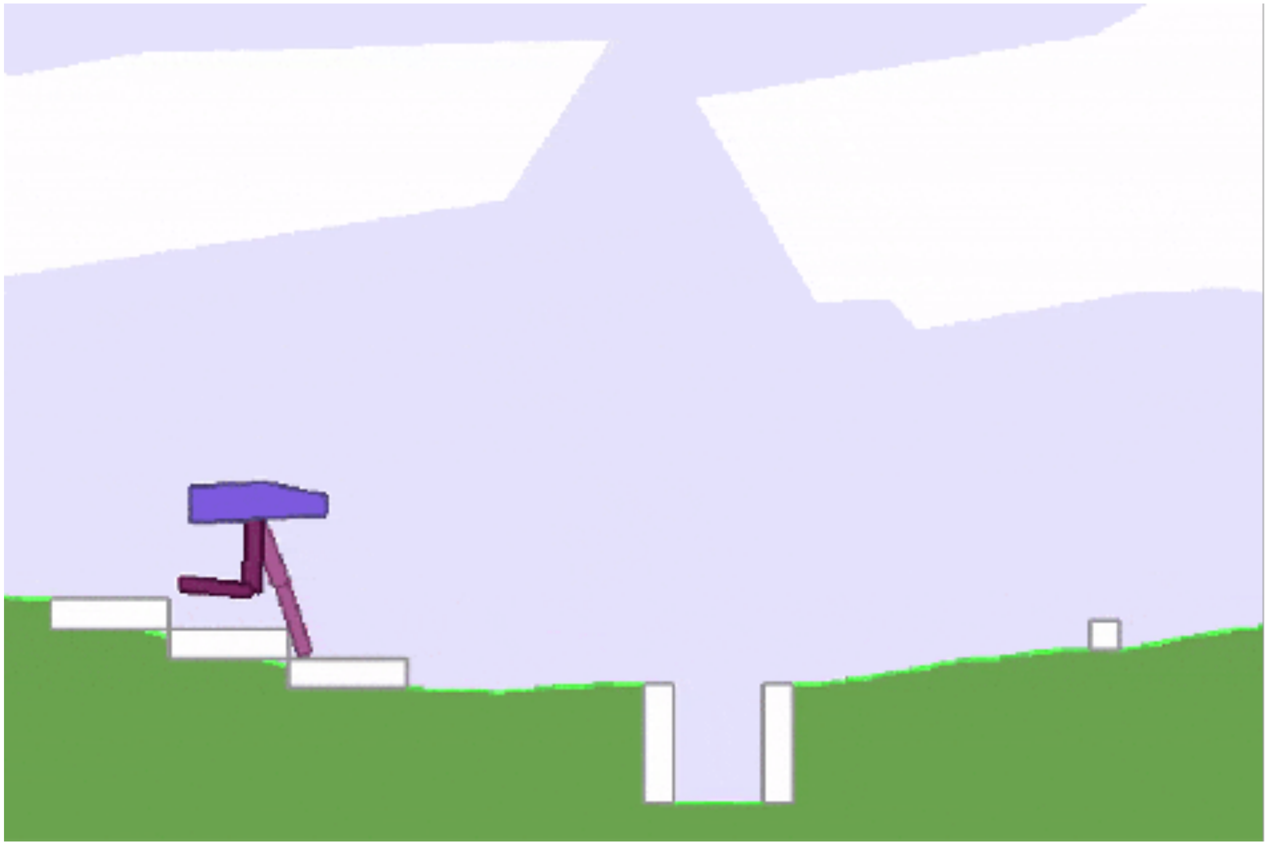
OpenAI Gym Environments




```
import gym
env = gym.make(‘‘Pendulum-v1’’)
observation = env.reset()
for _ in range(1000):
  env.render()
  # your agent here (just random actions)
  action = env.action_space.sample()
  obs, reward, done, info = env.step(action)
```

- Standard interface for single-agent RL environment.
- The simple API has been adopted by RL community.

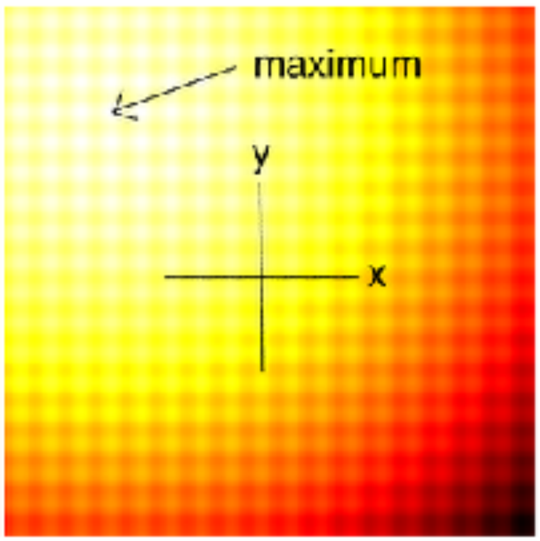
Why Evolve Weights for RL?



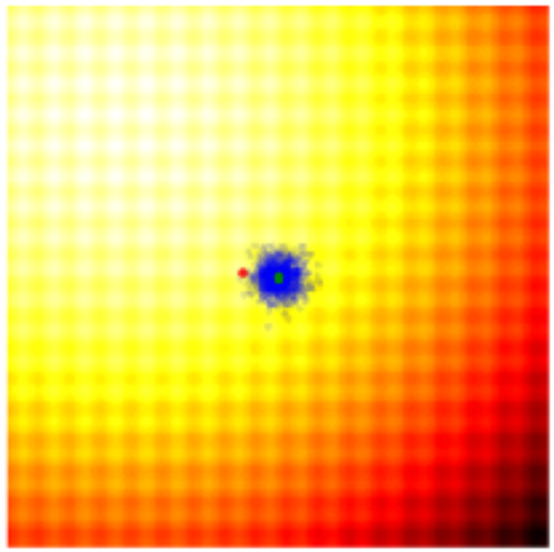
- Credit assignment, especially long term rewards, is hard!
- Easy to get stuck in local optima.
- Why Not?

Neuroevolution Algorithms

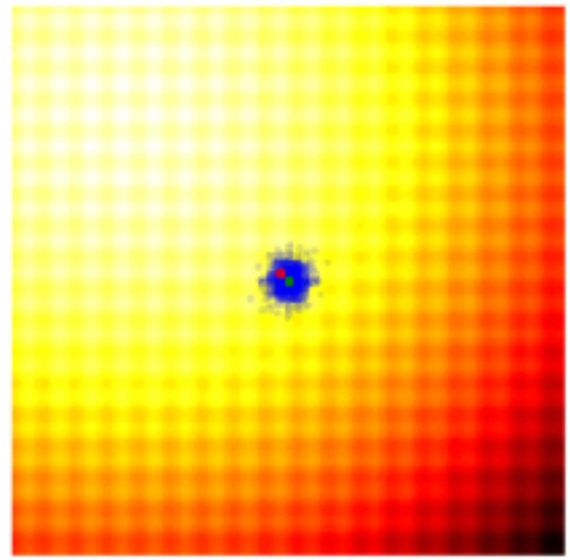
❖ Many out there: GA, NEAT, CMA-ES, PEPG, RS



Optimisation Problem:
Shifted 2D Rastrigin

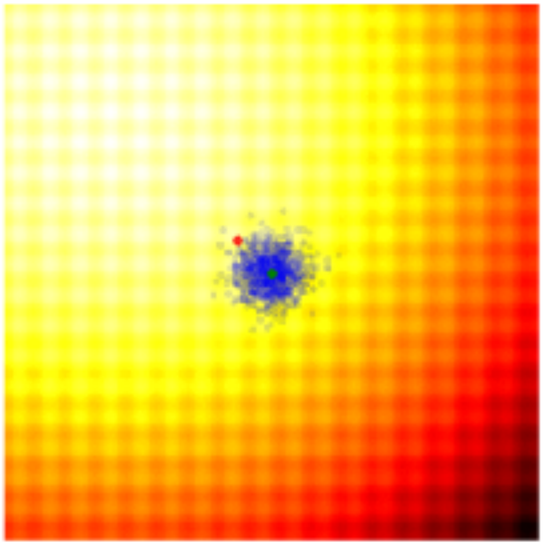


Genetic Algorithm

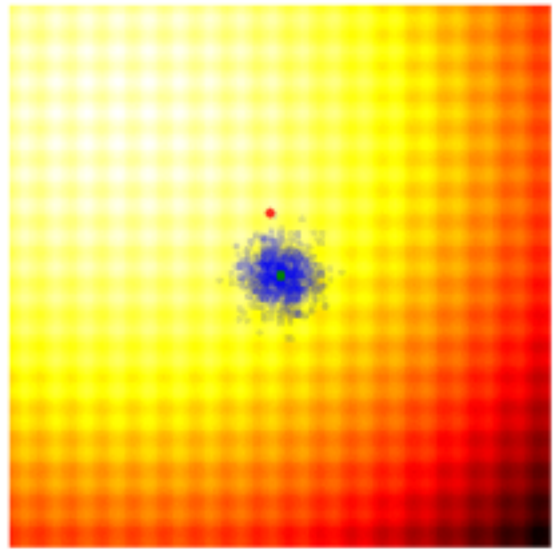


Evolution Strategies

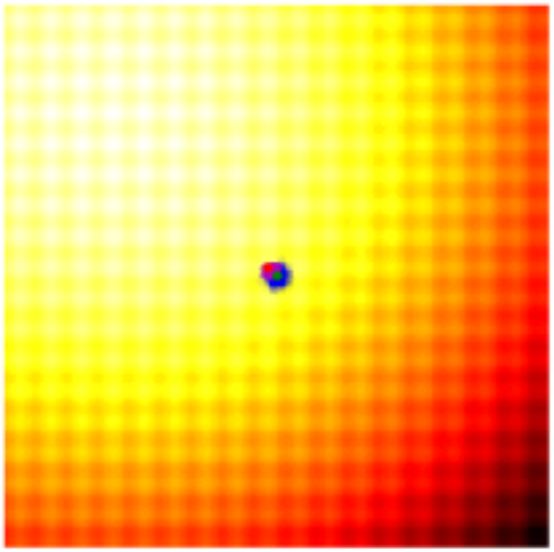
Neuroevolution Algorithms



REINFORCE (OpenAI)



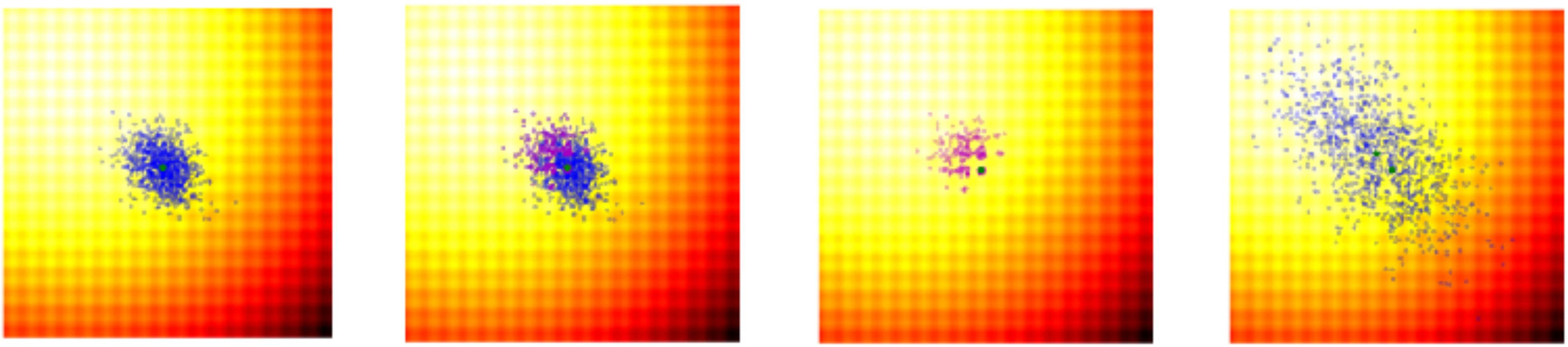
Parameter-Exploring Policy Gradients



CMA-ES

- ❖ We want to use a standard interface for all these methods.
- ❖ Standardise Gym Environments to use with Neuroevolution.

CMA-ES for Dummies



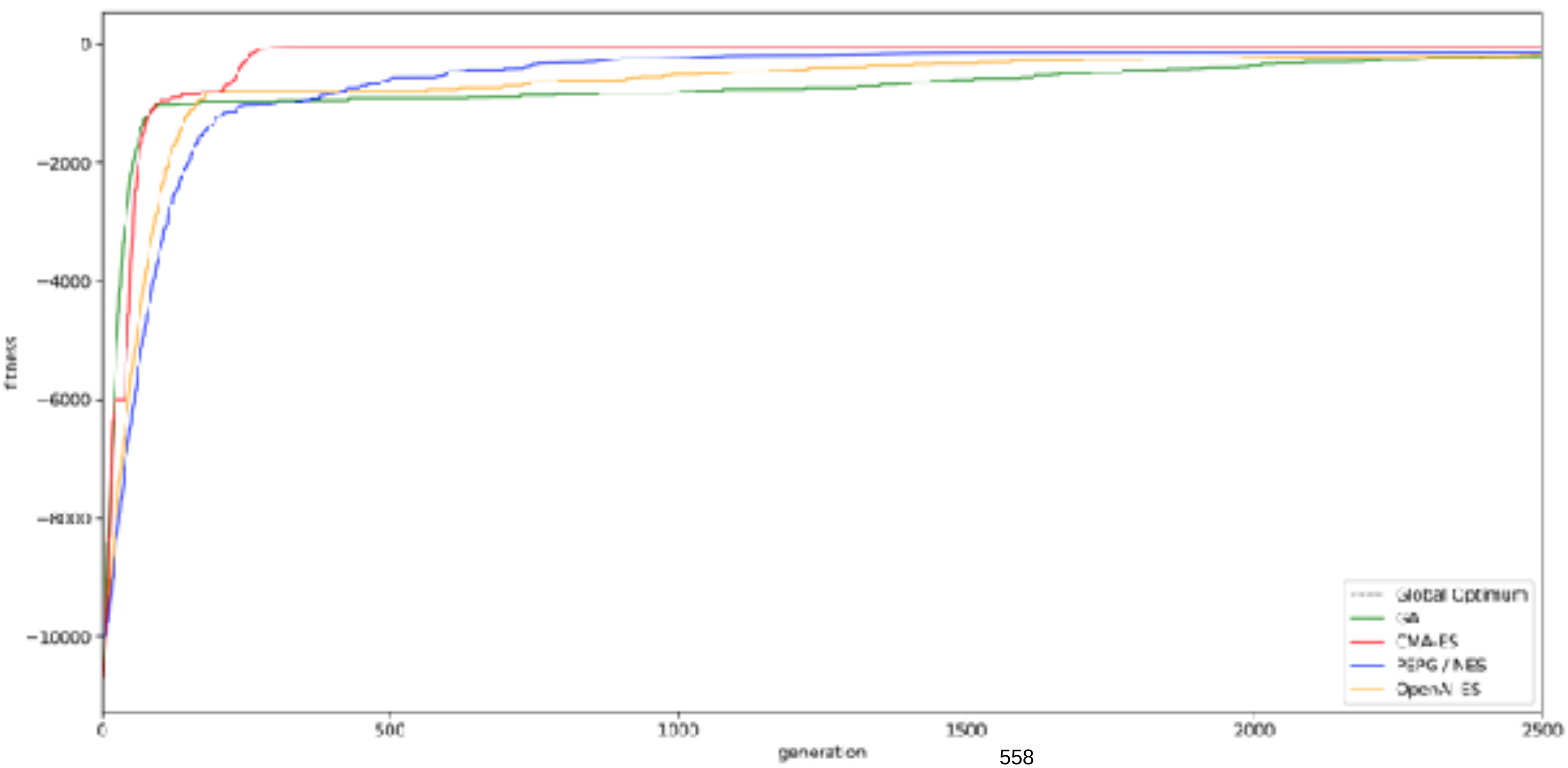
Step 1 Step 2 Step 3 Step 4

1. Calculate the fitness score of each candidate solution in generation.
2. Isolates the best 25% of the population in generation, in purple.
3. Using only the best solutions, and using the mean of the *current* generation (the green dot), calculate the covariance matrix of the next generation.
4. Sample a new set of candidate solutions using the updated mean and covariance matrix.

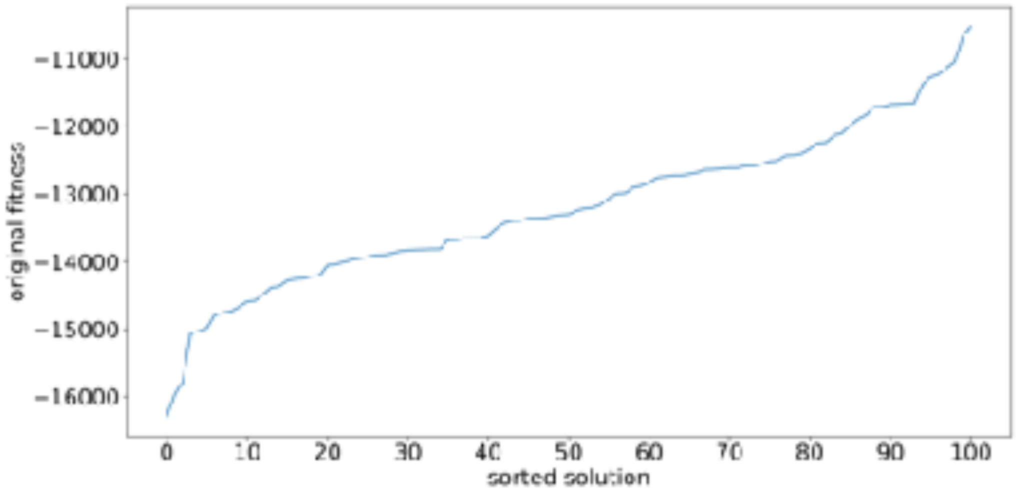
100-D Rastrigin Function

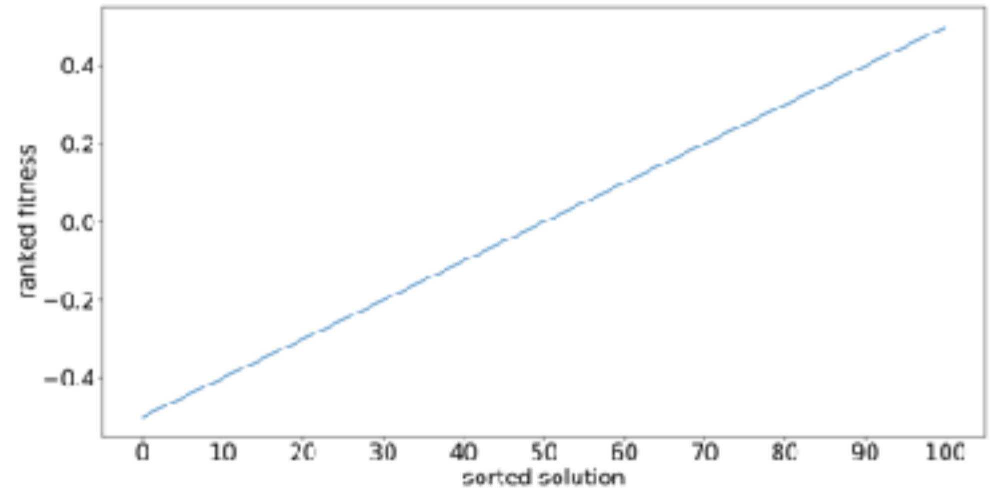
```
[ 10.          10.          10.          10.          10.
  10.99495864 10.99495864  8.01008777 10.99495864  9.00504136
  10.         10.          10.          9.00504136 10.
  9.00504136  10.          10.99495864  8.01008776 10.99495864
  10.99495864 10.          9.00504137  10.99495863 10.99495864
  9.00504136  10.          9.00504136  10.99495863 10.
  10.99495863  8.01008776  10.99495863 10.          10.
  10.99495863  9.00504137  10.99495863  9.99999999  10.
  10.          10.          10.          10.          10.
  10.99495864 10.          10.          9.00504137   9.00504137
  10.          9.00504136  10.99495863 10.99495864 10.00000001
  9.00504136  10.          9.00504136  10.          10.
  10.          9.99999999  10.          10.          9.99999999
  10.          9.00504137  10.          10.          9.00504136
  9.00504137  10.          9.00504137   9.99999999 10.
  9.00504136  9.00504136  10.00000001  10.          10.
  9.00504136  10.99495864 10.          10.99495864 10.
  10.00000001 10.00000001 10.          9.00504136  10.99495863
  9.00504136  10.99495865 10.          10.          10.00000001
  9.99999999  10.          10.99495864  9.99999999 10.        ]
```

- Final solution that CMA-ES discovered for 100-D Rastrigin function.
- Global optimal solution is a 100-dimensional vector of exactly 10.



What is Fitness Shaping?

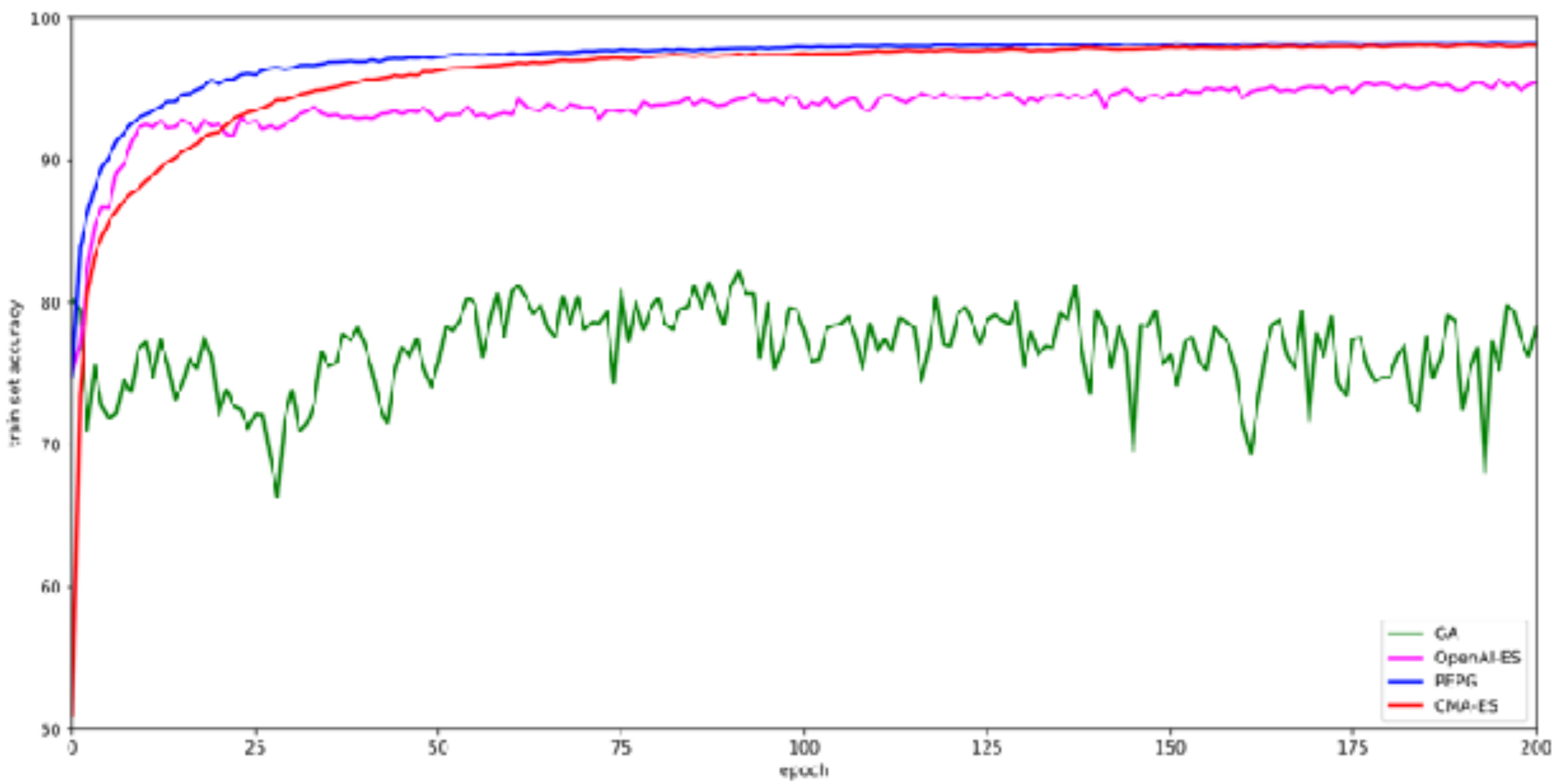




- If we have large outliers, the gradient estimation might become dominated by this outliers and increase the chance of the algorithm being stuck in a local optimum.
- Automatically normalises rewards.

MNIST with Neuroevolution

Method	Train Set	Test Set
Adam (SGD)	99.8	98.9
Simple GA	82.1	82.4
CMA-ES	98.4	98.1
OpenAI-ES	96.0	96.2
PEPG	98.5	98.0



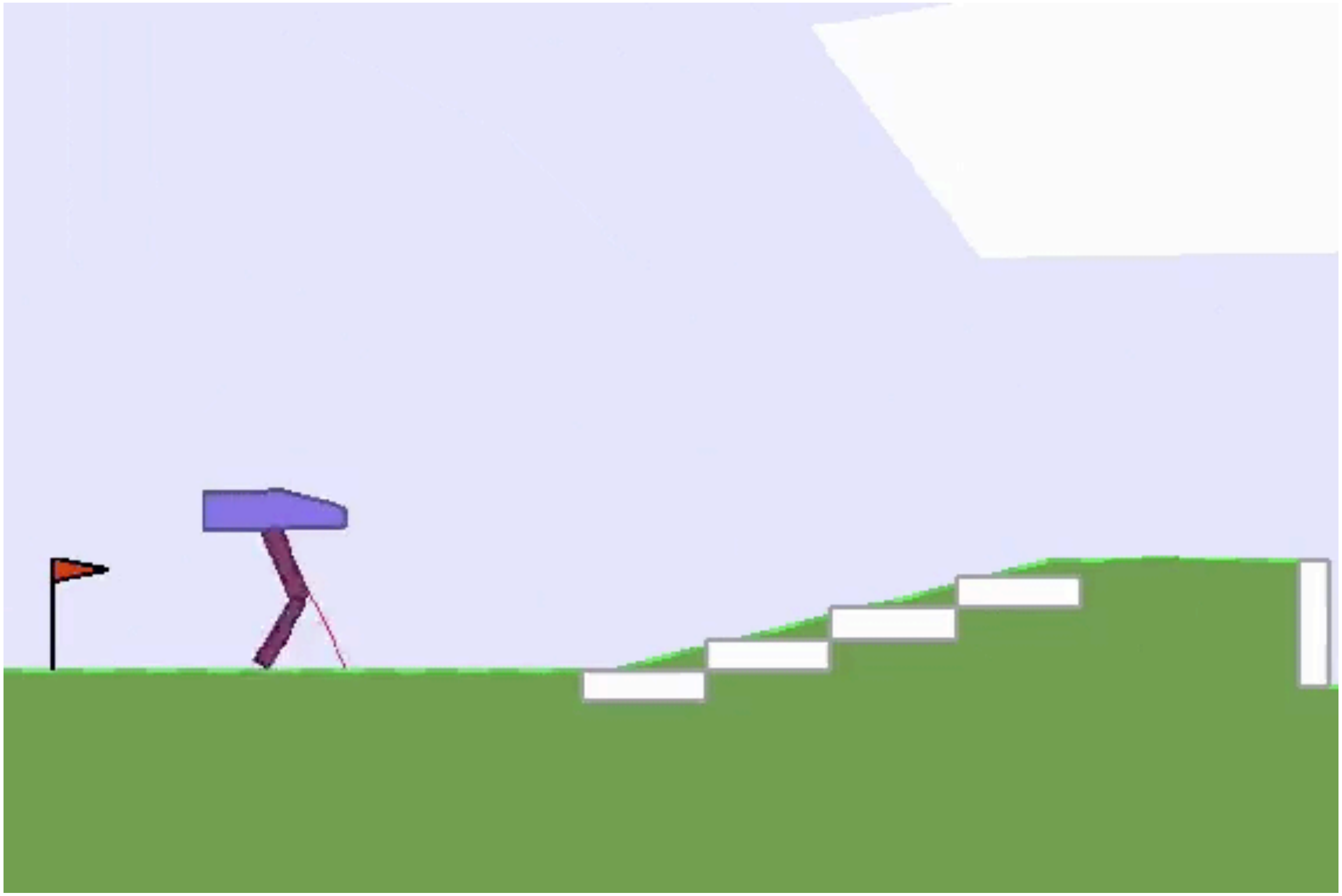
Neuroevolution with OpenAI Gym

```python
env = gym.make('worlddomination-v0')
# use our favourite algorithm
solver = OurNeuroevolutionAlgo()
while True:
  # ask the ES to give set of params
  solutions = solver.ask()
  # create array to hold the results
  fitlist = np.zeros(solver.popsize)
  # evaluate for each given solution
  for i in range(solver.popsize):
    # init the agent with a solution
    agent = Agent(solutions[i])
    # rollout env with this agent
    fitlist[i] = rollout(agent, env)
  # give scores results back to ES
  solver.tell(fitness_list)
  # get best param & fitness from ES
  bestsol, bestfit = solver.result()
  # see if our task is solved
  if bestfit > MY_REQUIREMENT:
    break
```

```python
def rollout(agent, env):
  obs = env.reset()
  done = False
  total_reward = 0
  while not done:
    a = agent.get_action(obs)
    obs, reward, done = env.step(a)
    total_reward += reward
  return total_reward
```

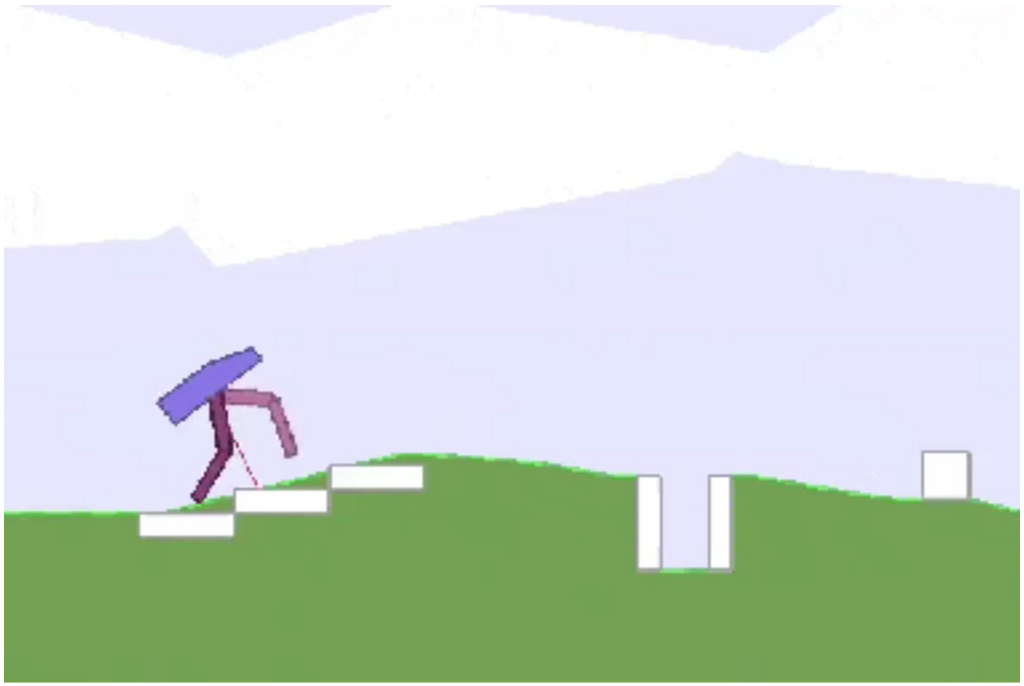
- We only care about the terminal, cumulative reward.
- Simple implementation at http://github.com/hardmaru/estool

ES Solved BipedalWalkerHardcore-v0

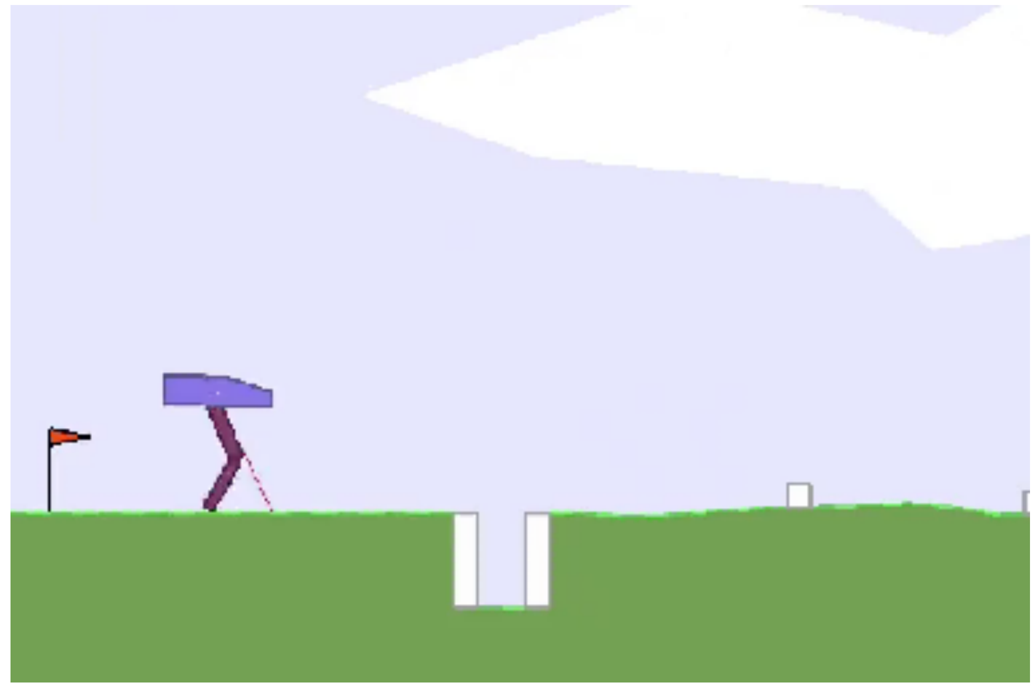


❖ Simple averaging technique -> Much more robust policies.

ES Solved BipedalWalkerHardcore-v0



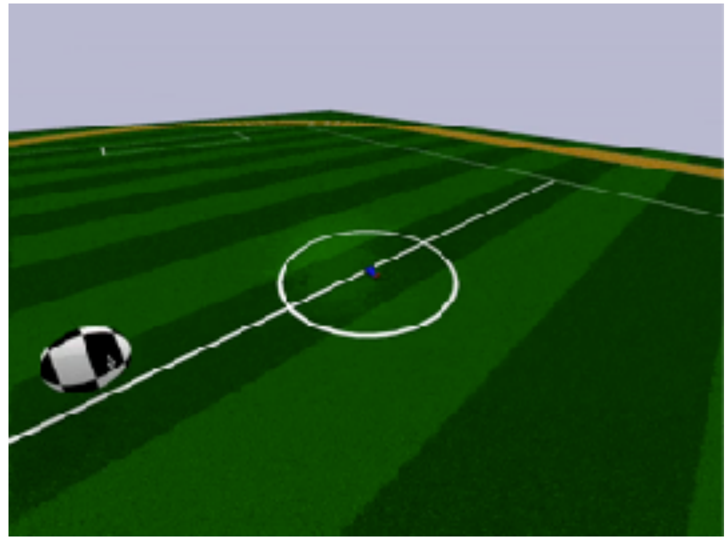
CMA-ES

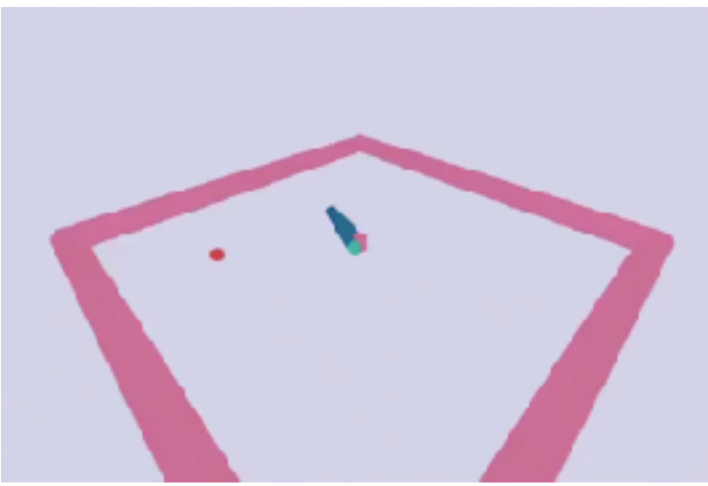


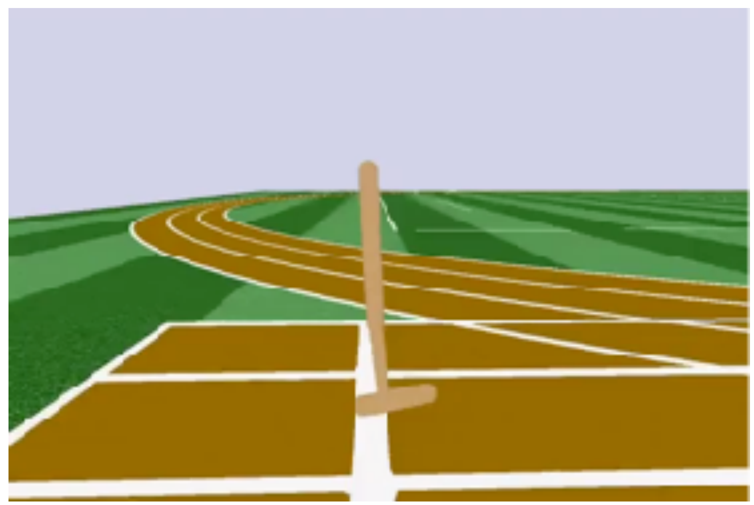
PEPG

❖ First solution to achieve score > 300 over 100 random trials.

ESTool with PyBullet, Roboschool

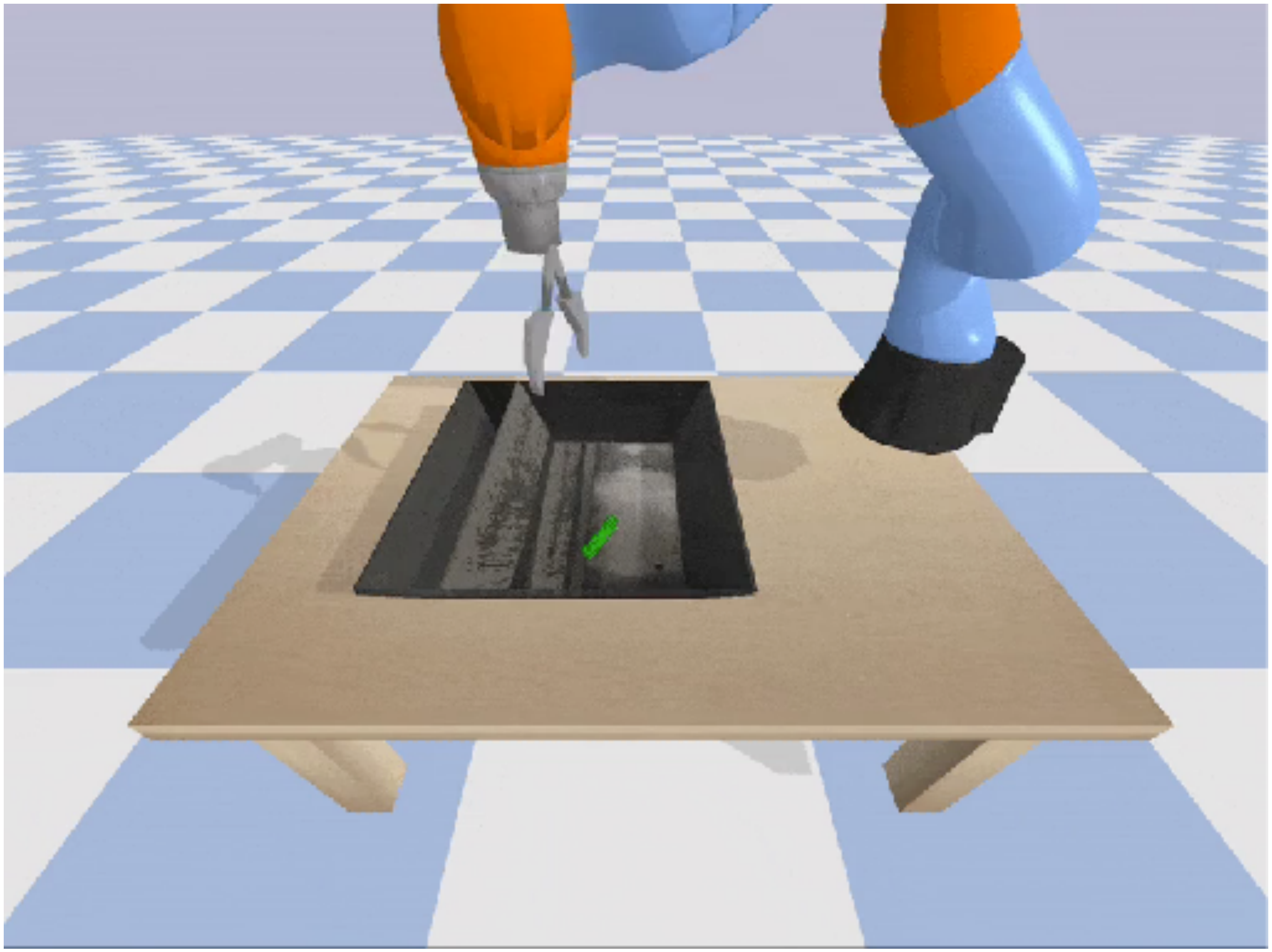






- We can solve a large set of standard continuous control tasks.

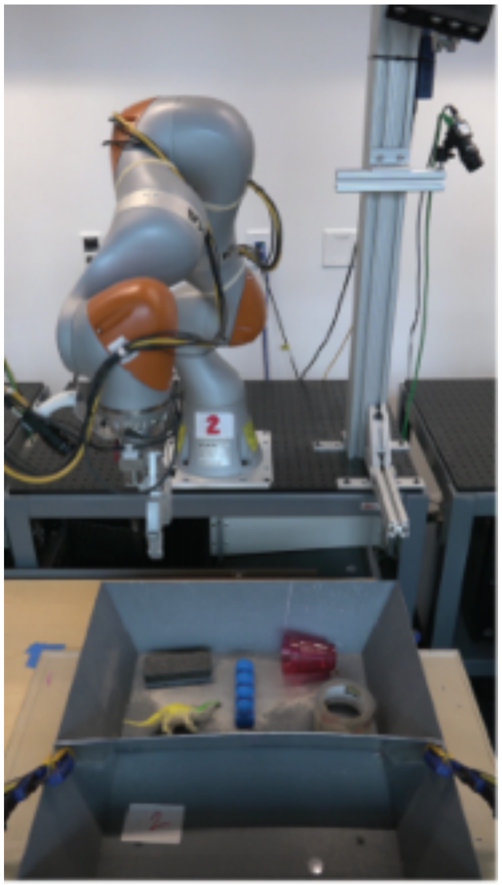
ESTool with PyBullet Kuka Arm

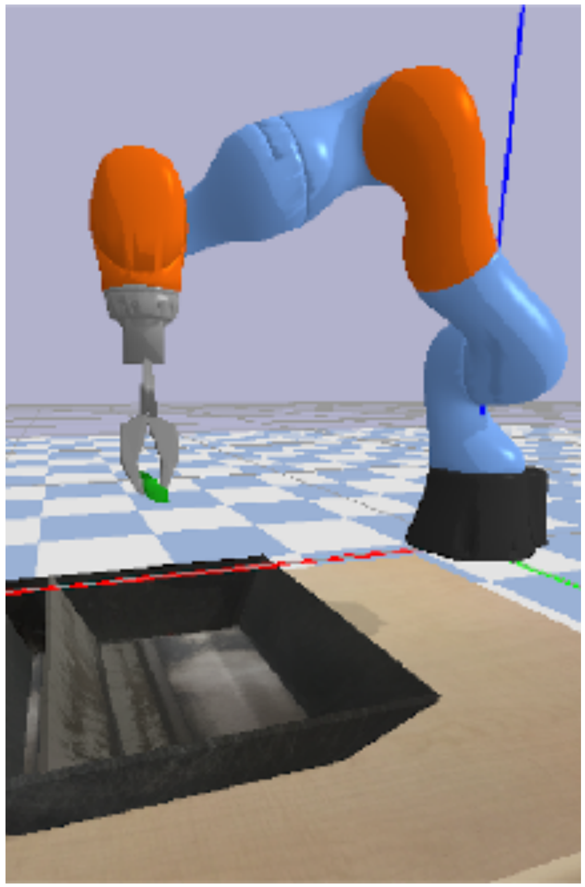


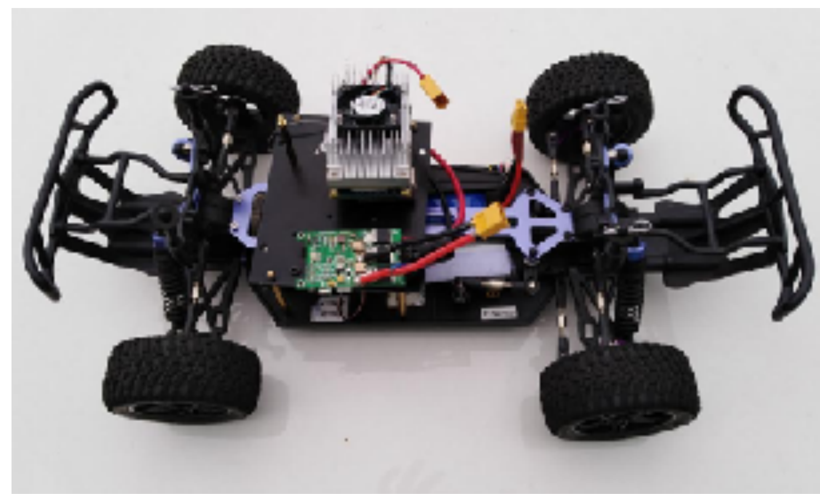
❖ Kuka grasping tasks easily solvable. Can incorporate vision.

Transfer Learning PyBullet Models

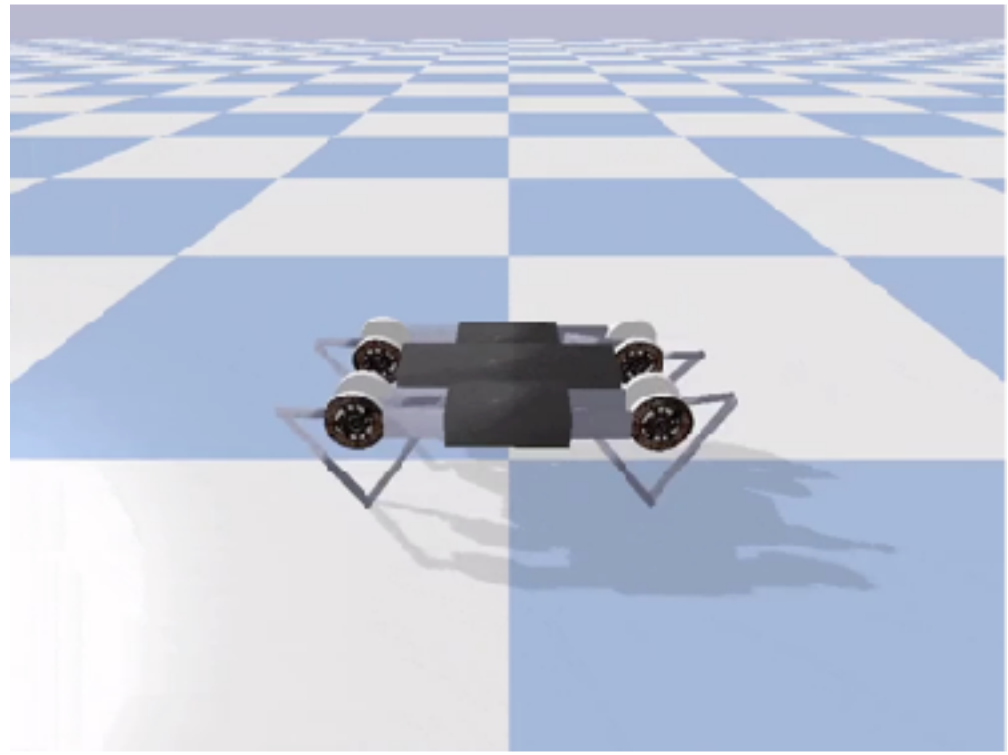
- MIT Racecar, Minitaur, Kuka Arm







PyBullet Minitaur Task

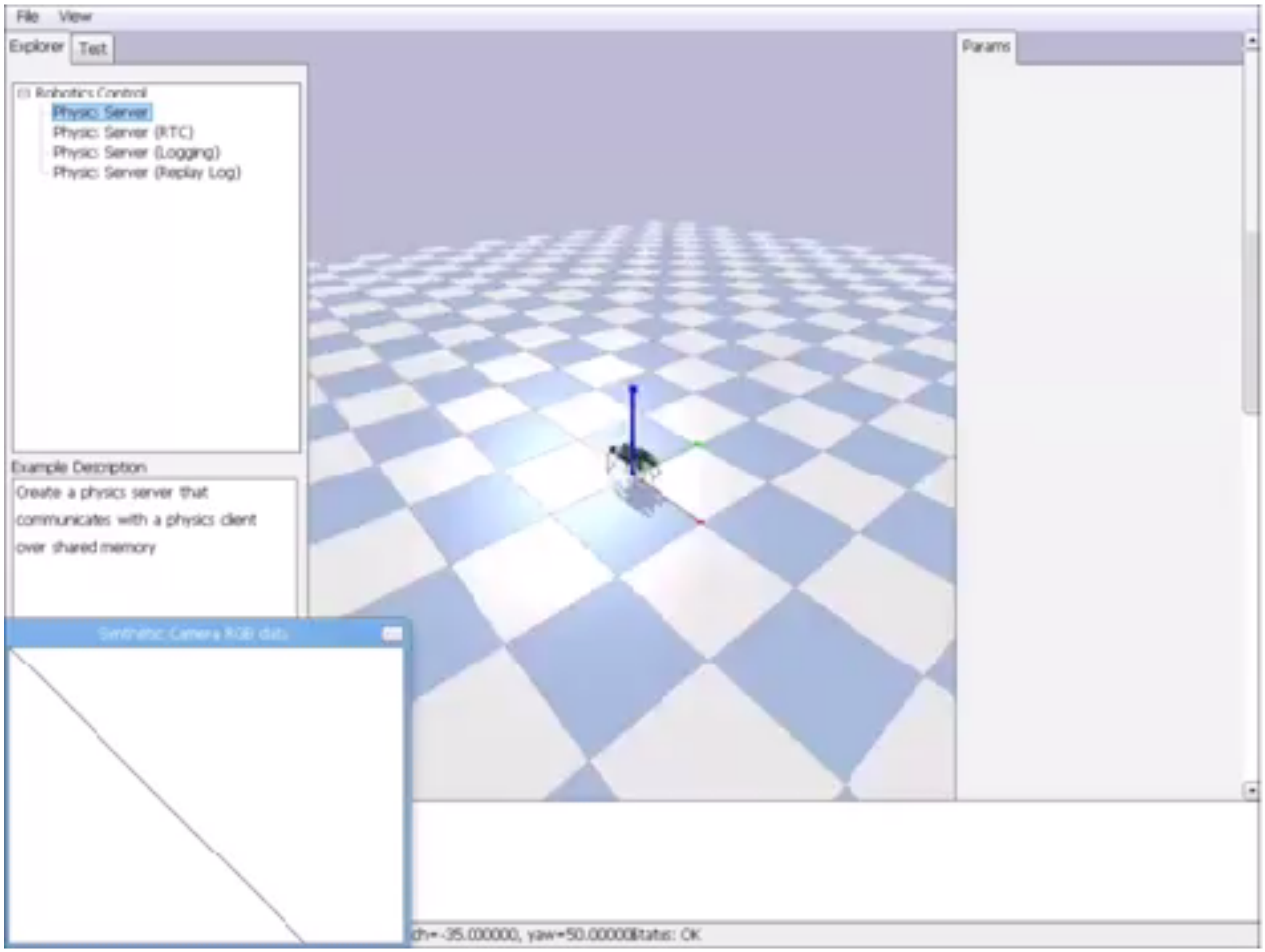


- PyBullet includes realistic models of actual robots.
- Useful to experiment with transfer learning.

Sim2Real Minitaur

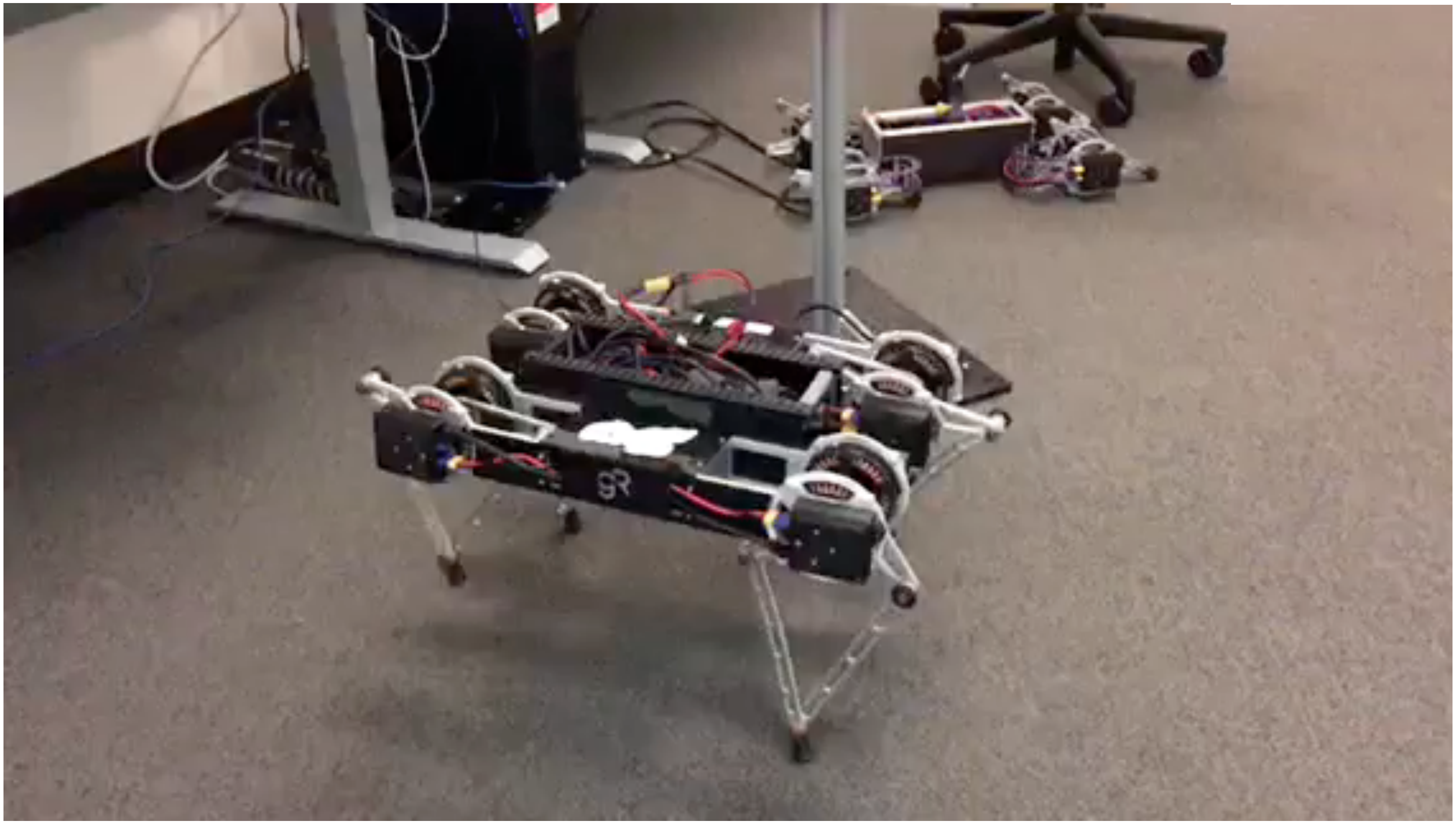
❖ Difficult to transfer learned policy from simulation to reality.

Simpler Task - Stand up with 2 legs



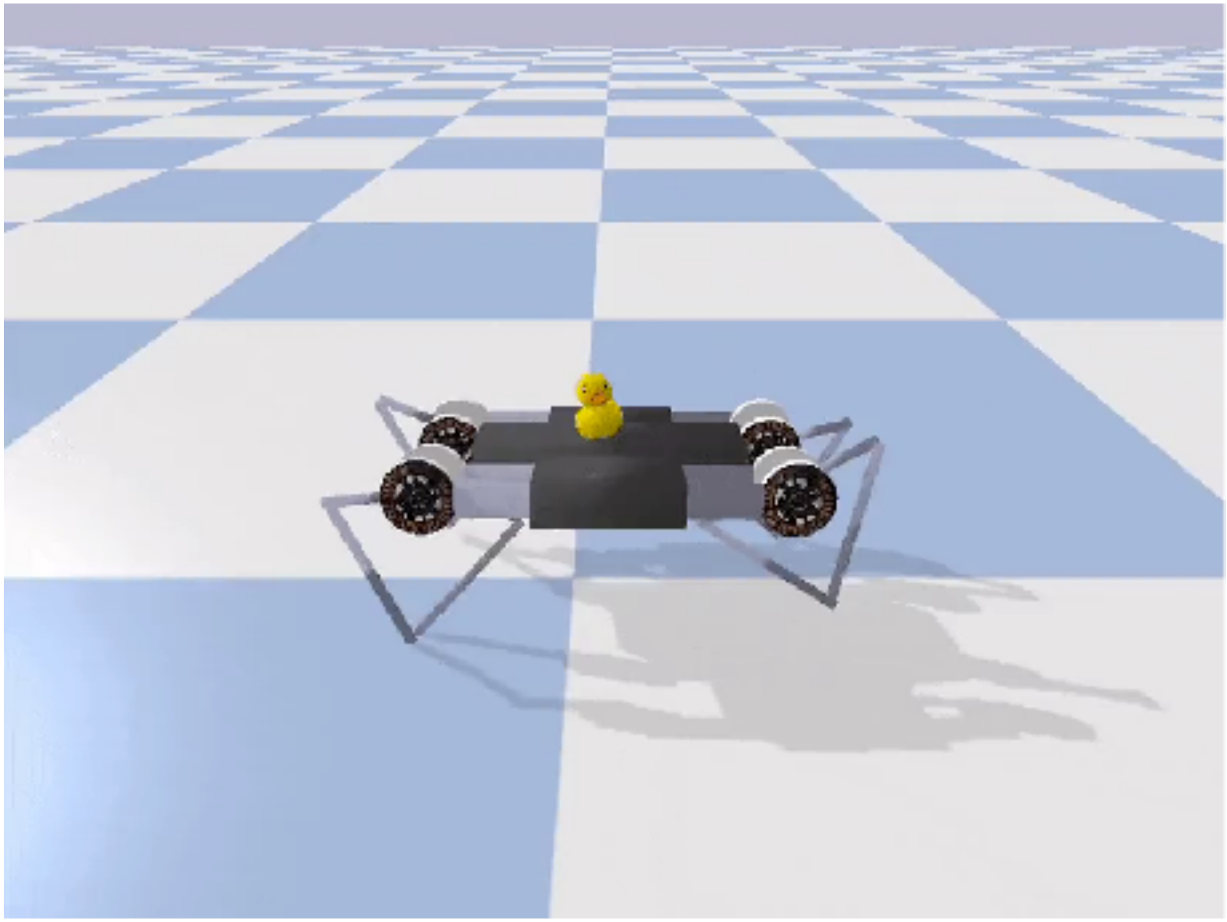
❖ Optimizing Simulations with Noise-Tolerant Structured Exploration (ICRA 2018)

Simpler Task - Stand up with 2 legs



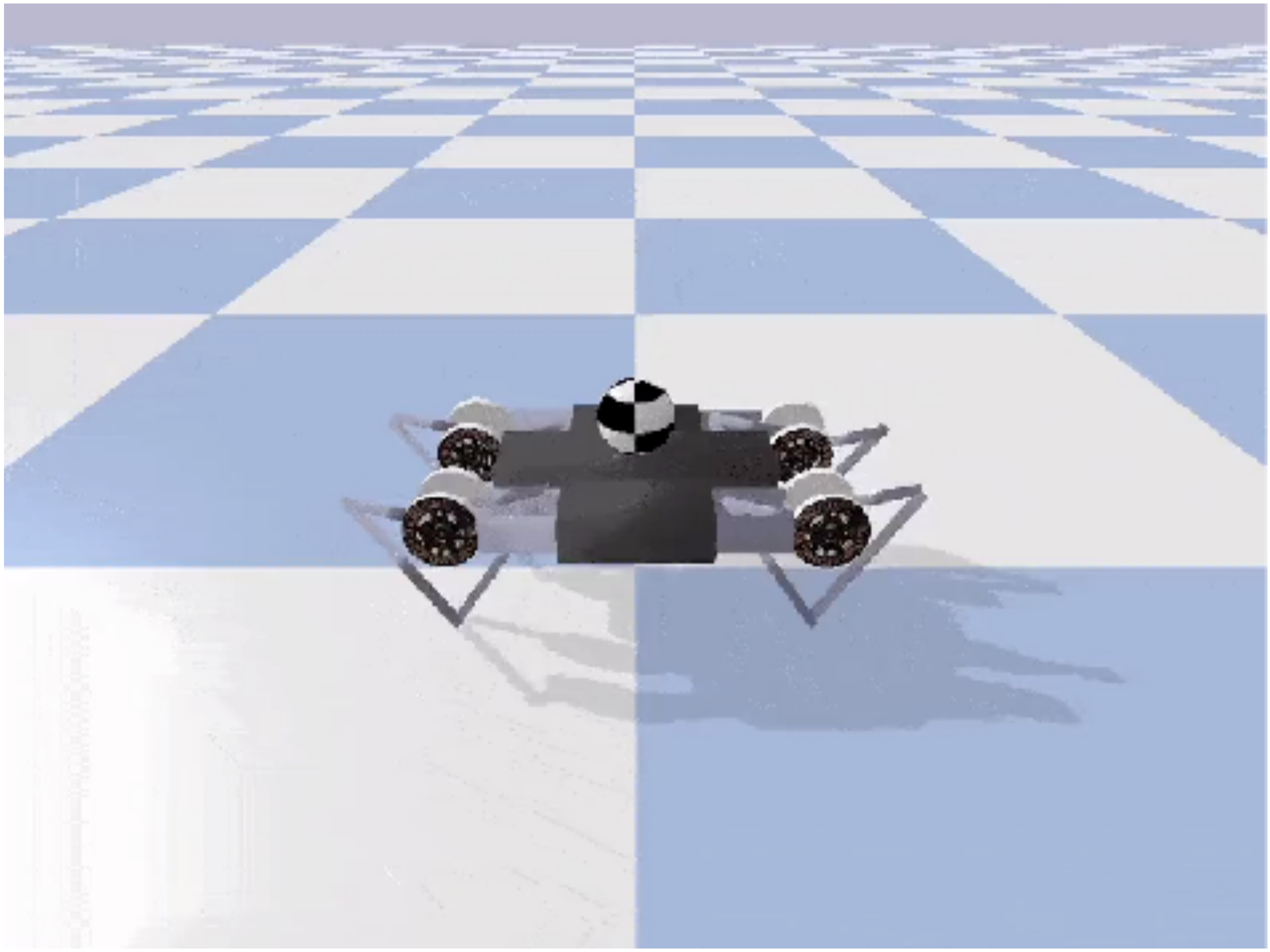
- Optimizing Simulations with Noise-Tolerant Structured Exploration (ICRA 2018)

Robust Minitaur Environments



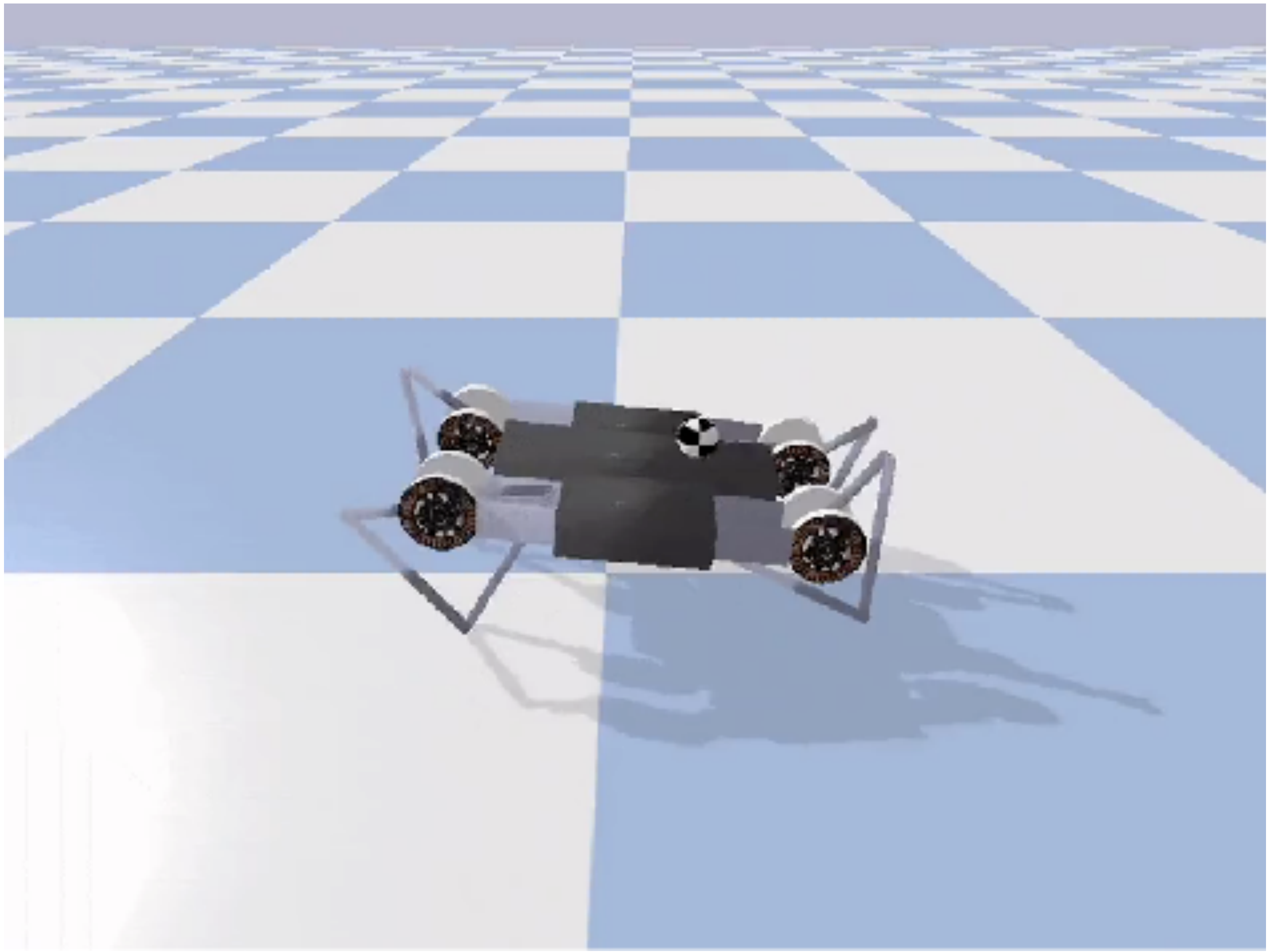
- Add more difficult task to the existing environment.
- End rollout once either objective failed.

Robust Minitaur Environments



❖ Adding a ball made it cheat. Must be careful with objectives.

Robust Minitaur Environments



- Showed it who is boss.

Sim2Real Minitaur

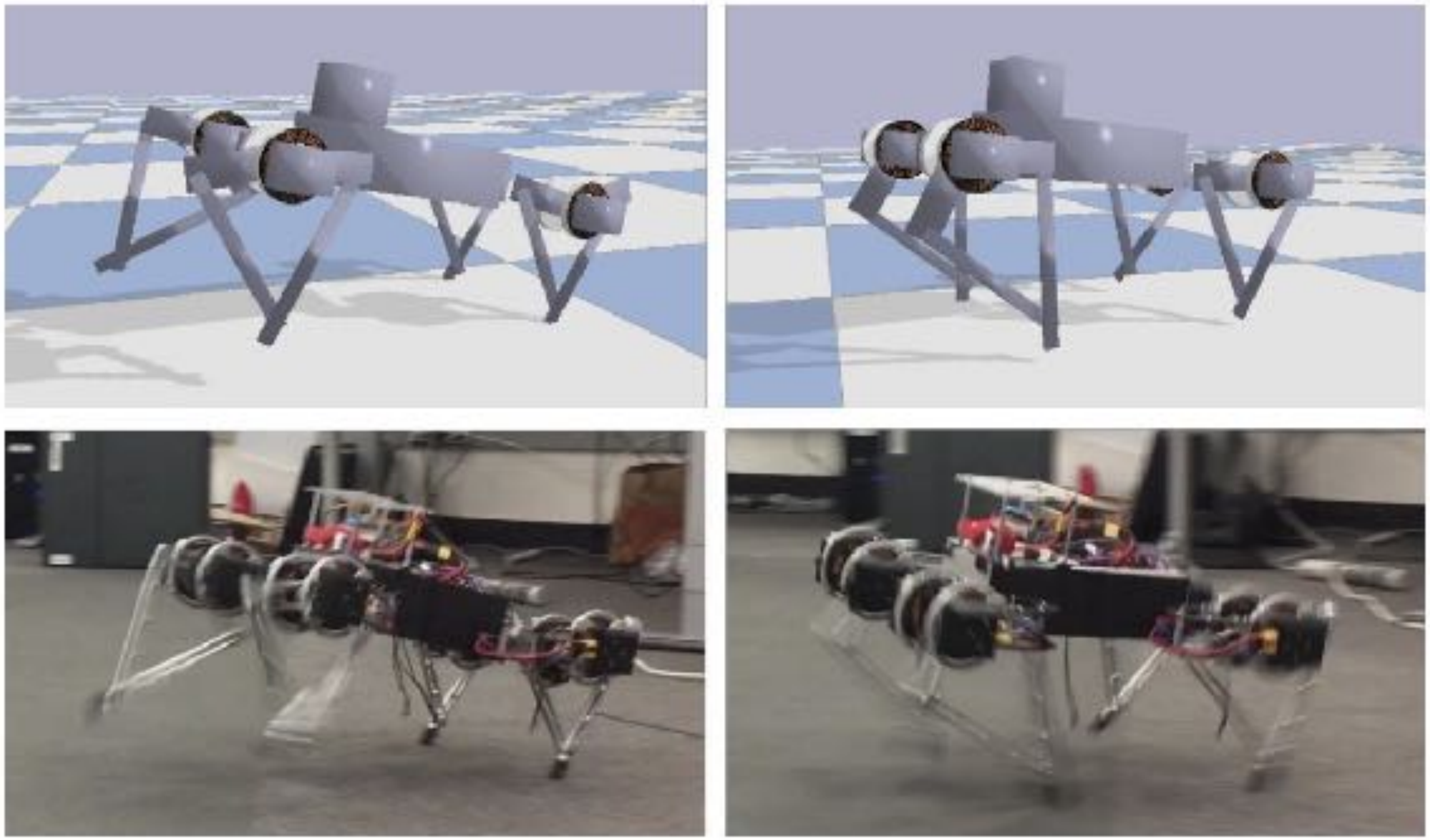
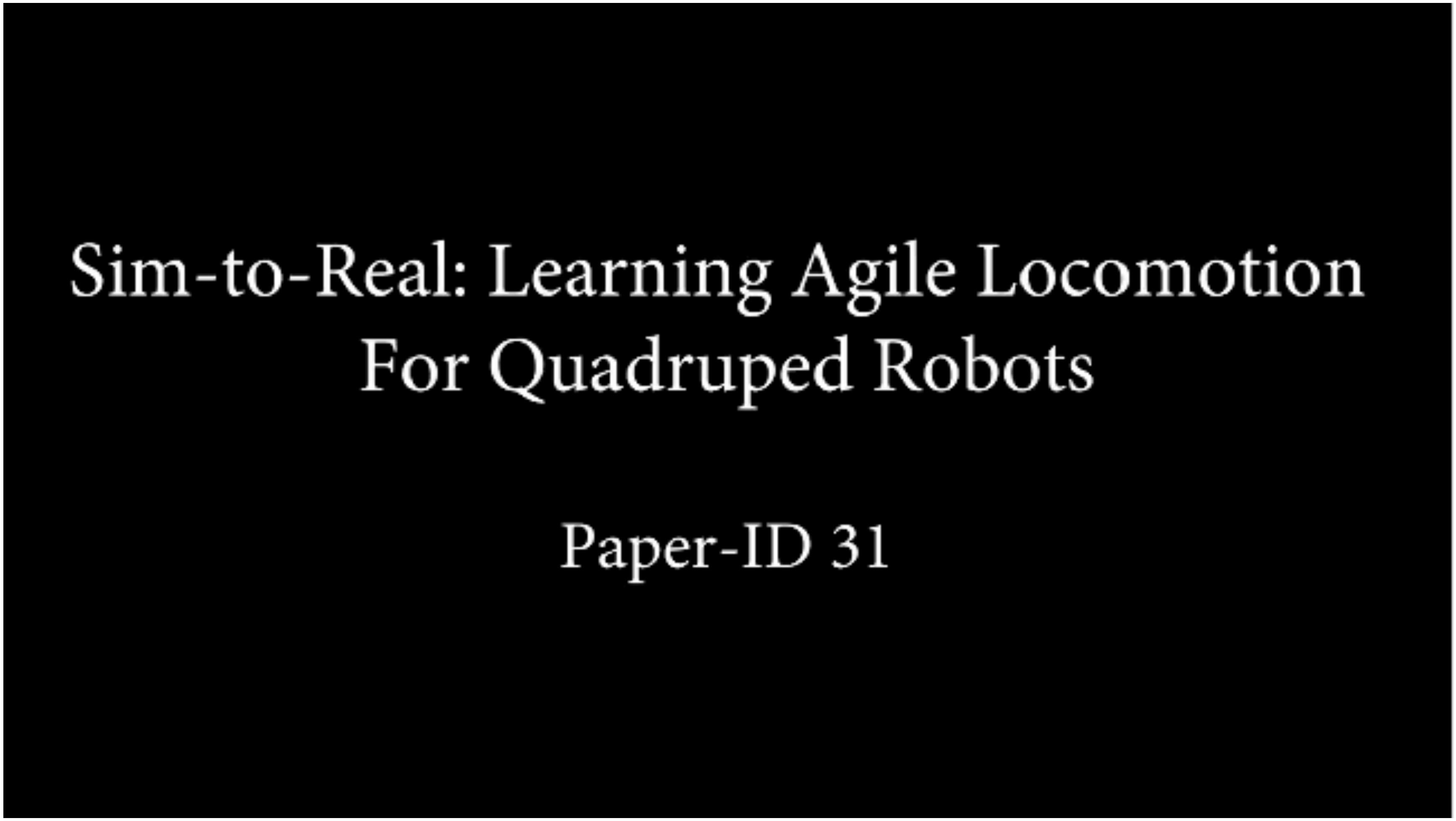


Fig. 1: The simulated and the real Minitaurs learned to gallop using deep reinforcement learning.

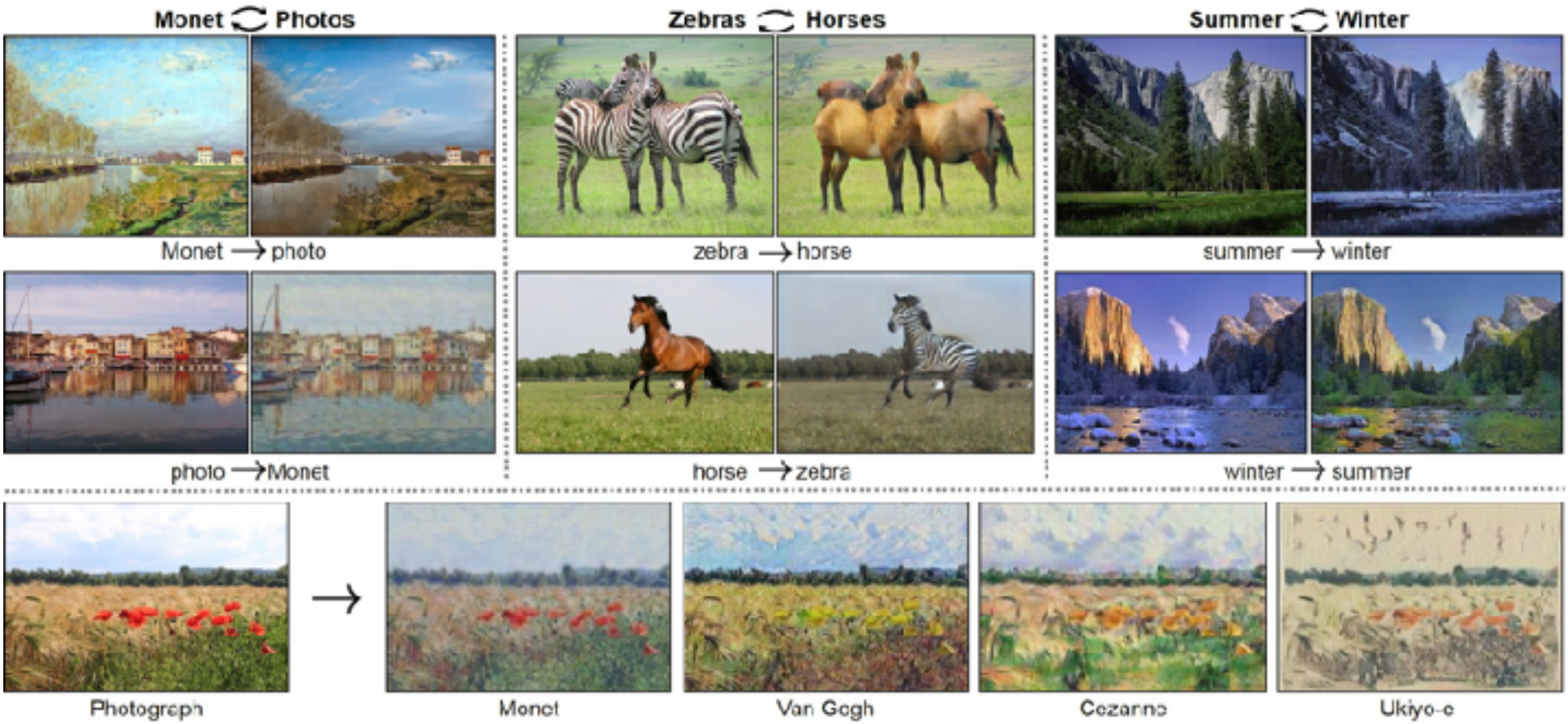
- Sim-to-Real: Learning Agile Locomotion For Quadruped Robots (RSS 2018).

Sim2Real Minitaur



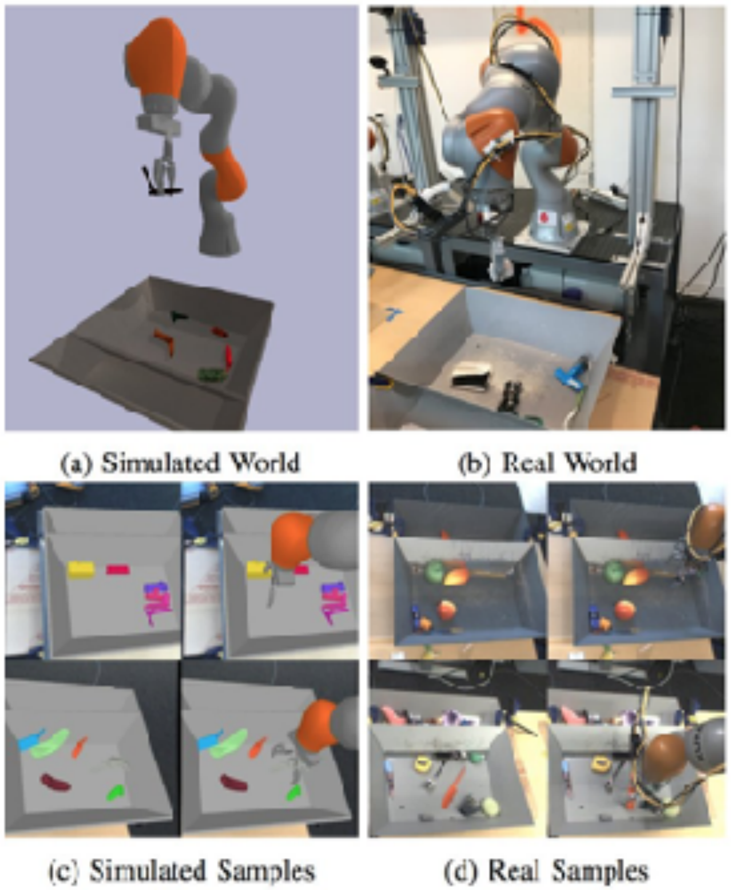
❖ Sim-to-Real: Learning Agile Locomotion For Quadruped Robots (RSS 2018).

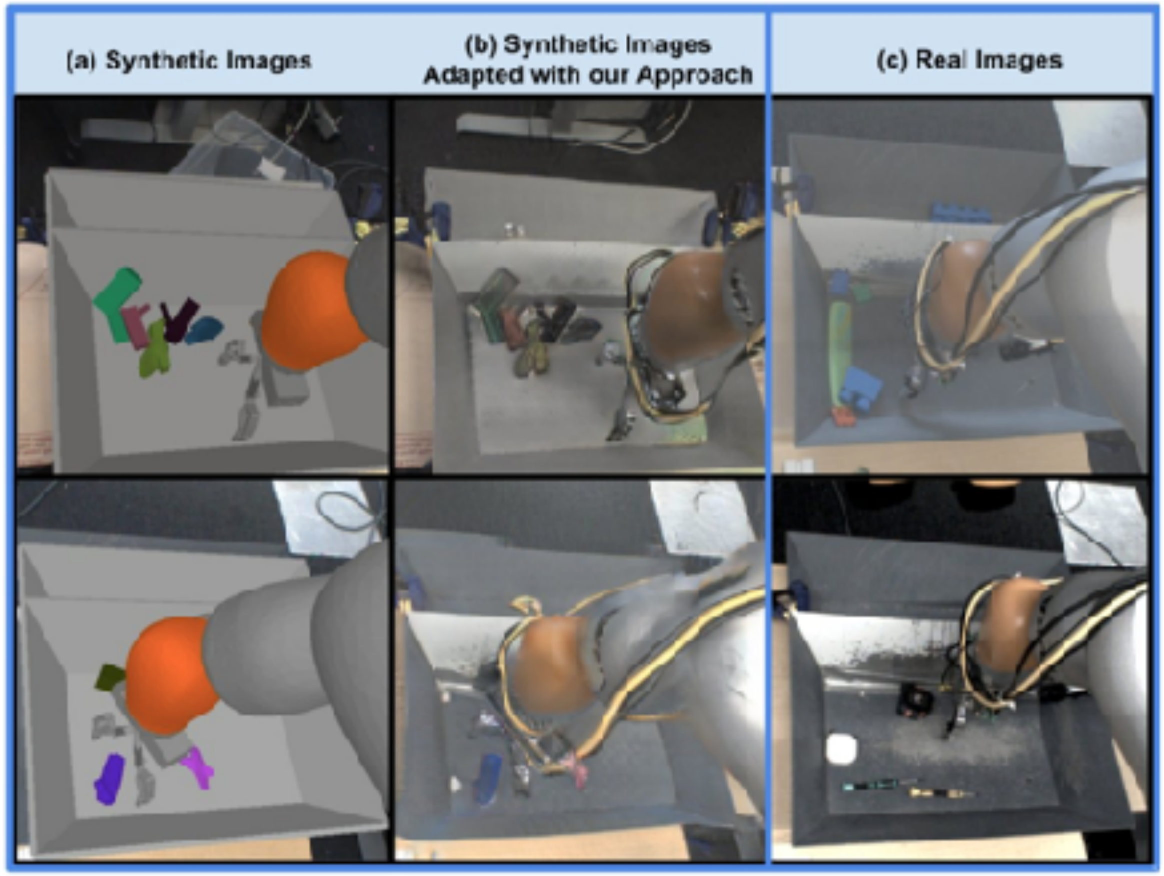
Sim2Real Kuka Arm: Apply "CycleGAN"



❖ Unpaired Image-to-Image Translation using CycleGAN(Zhu, Park et al. 2017)

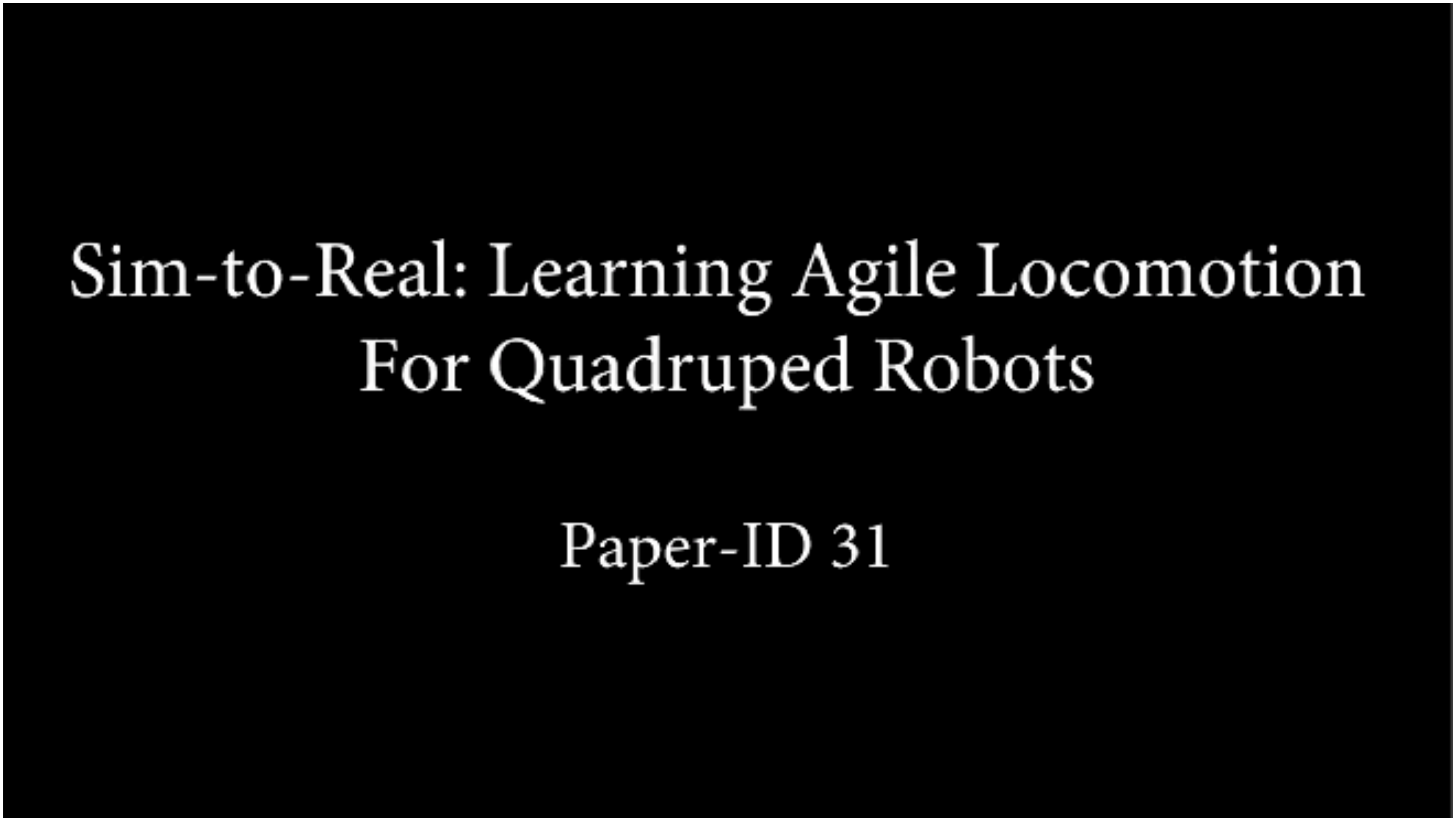
Sim2Real Kuka Arm: Apply “CycleGAN”





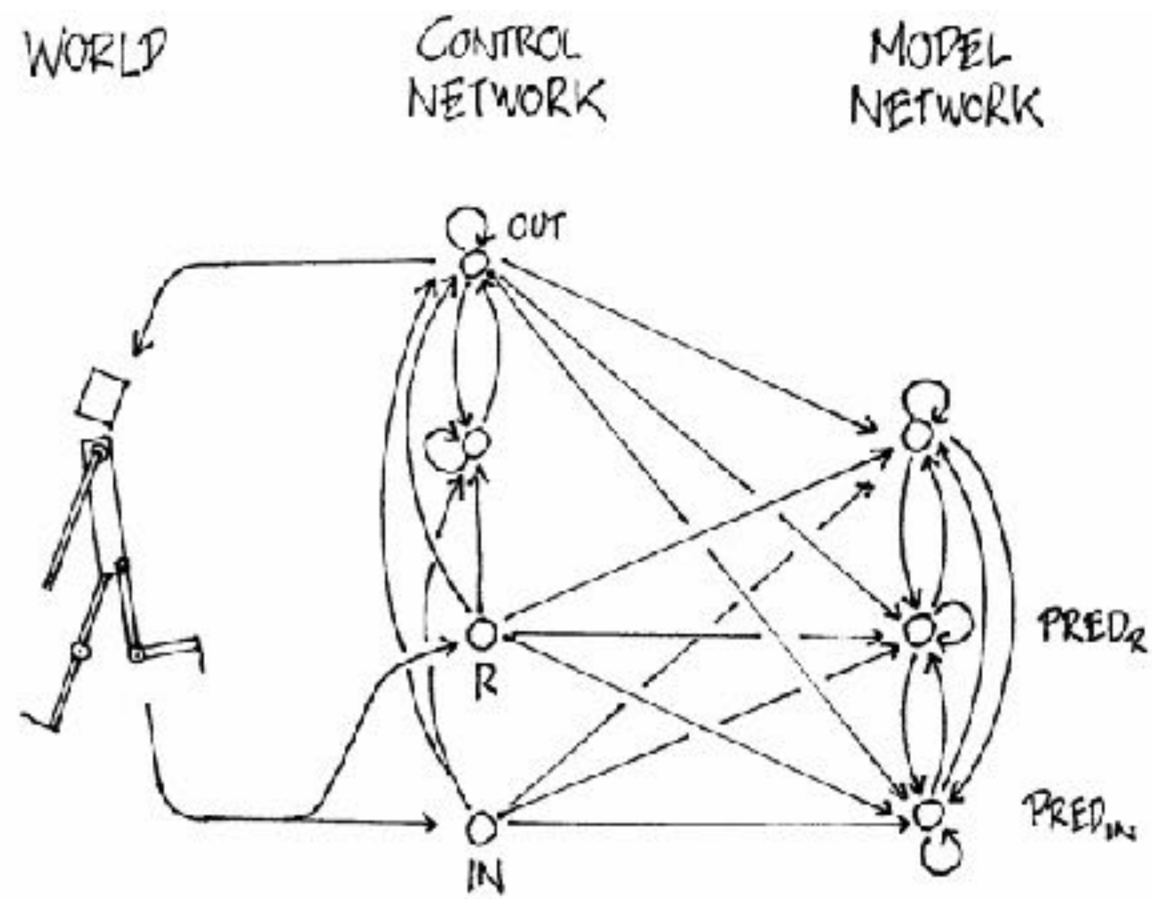
- Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping (2017)

Sim2Real Kuka Arm



- Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping (2017)

World Models

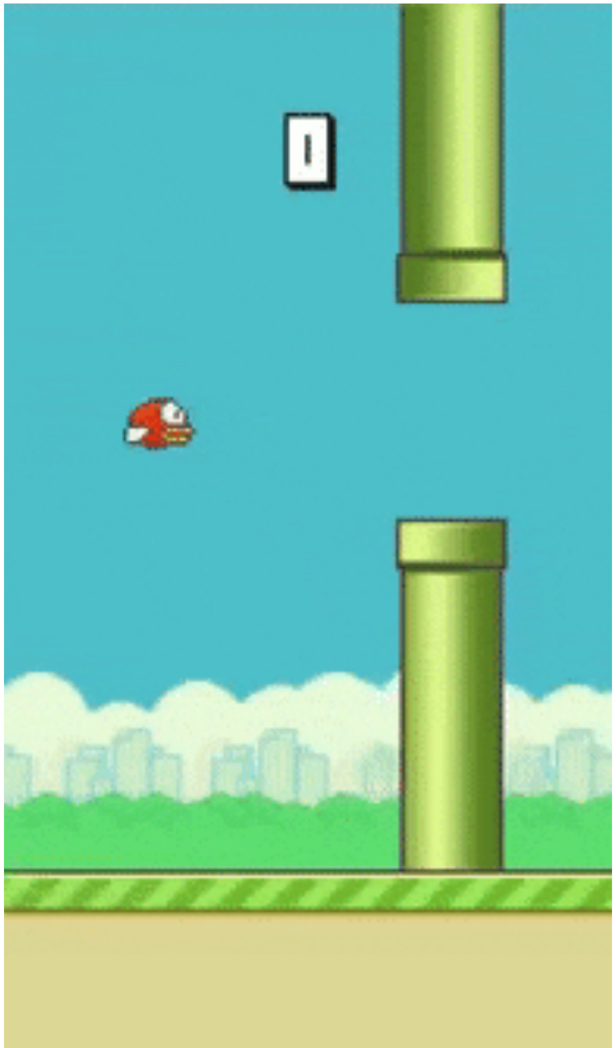


- Ha and Schmidhuber (2018)
- Schmidhuber (1990, 1991, 2015)

We have a predictive model of the world.

- Schmidhuber (1990, 1991, 2015)

Basic Algorithmic Information Theory Argument

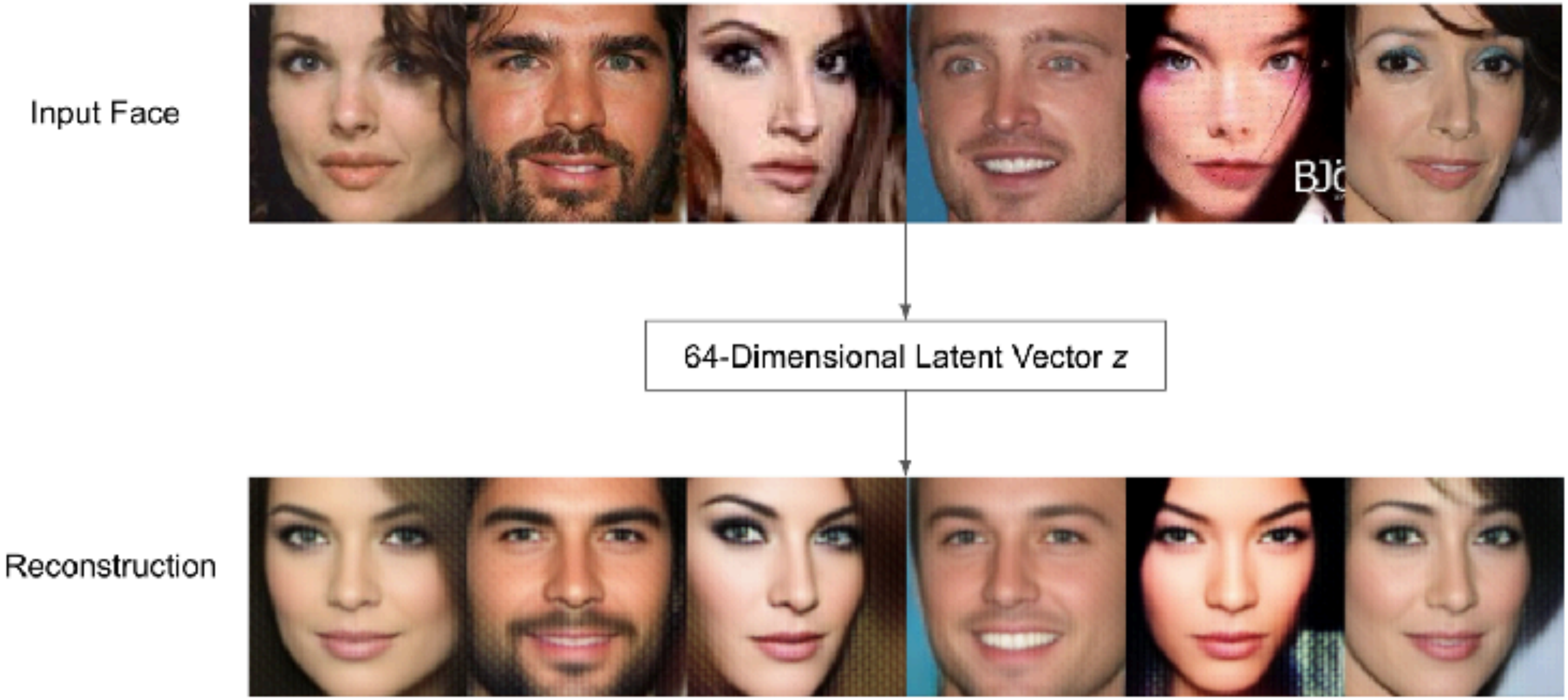


- Many RL tasks requires representation learning and predicting the future.

- We can efficiently train highly expressive models to learn representations of space and time with backprop.

- Use these representations as features to learn a compact policy, using Neuroevolution, to achieve the task.

Generative Model: Variational Autoencoder

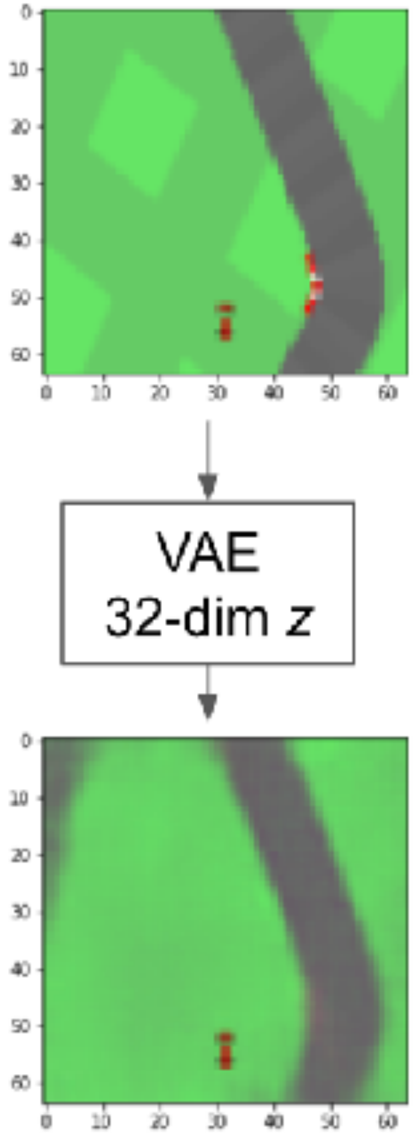


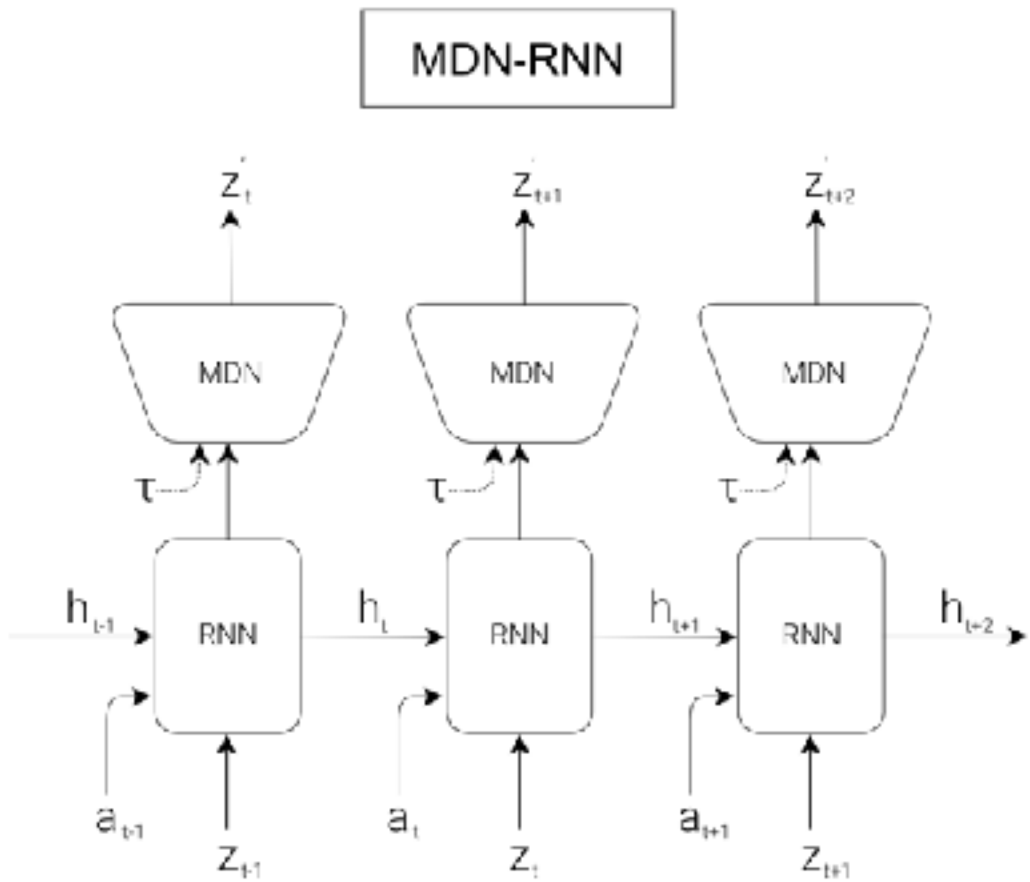
- Compress Famous People into 64D Floats with Gaussian Prior

Generative Model: Mixture Density Networks

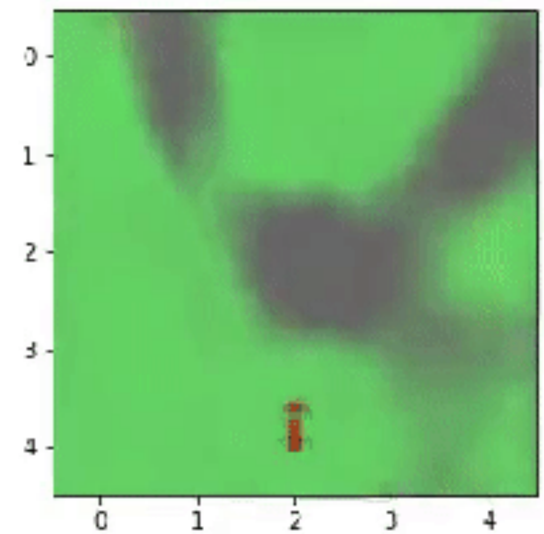
- Train an RNN to predict the probability distribution of next pen strokes. Model PDF as a Mixture of Gaussian distribution.

Learning a Generative Model of a Gym Environment





$$P(z_{t+1} \mid a_t, z_t, h_t)$$



Train VAE and MDN-RNN with Deep Learning.
Train controller policy with Neuroevolution to determine action:

$$a_t = W_c \, [z_t \; h_t] \; + b_c$$

Car Racing Task

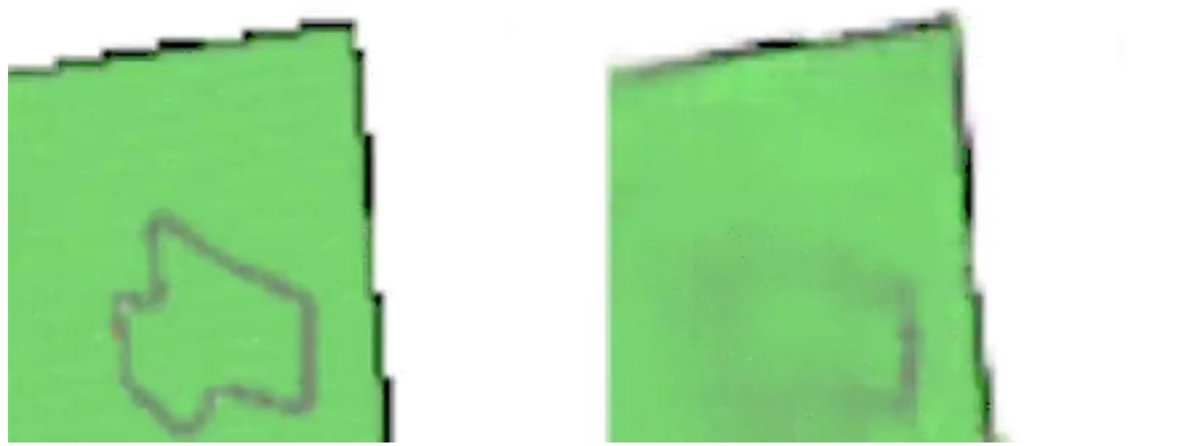
- Continuous control task to learn from pixels in a top-down racing environment.
- Maps are randomly generated for each trial.
- Actions Space: left/right, accelerate, break
- Cumulative Reward is:

  1000 x Fraction of Tiles Visited - 0.1 x Time Taken
- Episode finishes when all tiles are visited or when t > 1000
- For example, if you have finished in 732 frames, your reward is 1000 - 0.1*732 = 926.8 points.
- Task considered solved when Avg Score > 900 over 100 random trials

Car Racing Task - Training Procedure

1. Collect 10,000 rollouts from a random policy.
2. Train VAE (V) to encode frames into $z \in \mathcal{R}^{32}$.
3. Train MDN-RNN (M) to model $P(z_{t+1} \mid a_t, z_t, h_t)$.
4. Define Controller (C) as $a_t = W_c \; [z_t \; h_t] \; + \; b_c$.
5. Use CMA-ES to solve for a W_c and b_c that maximizes the expected cumulative reward.

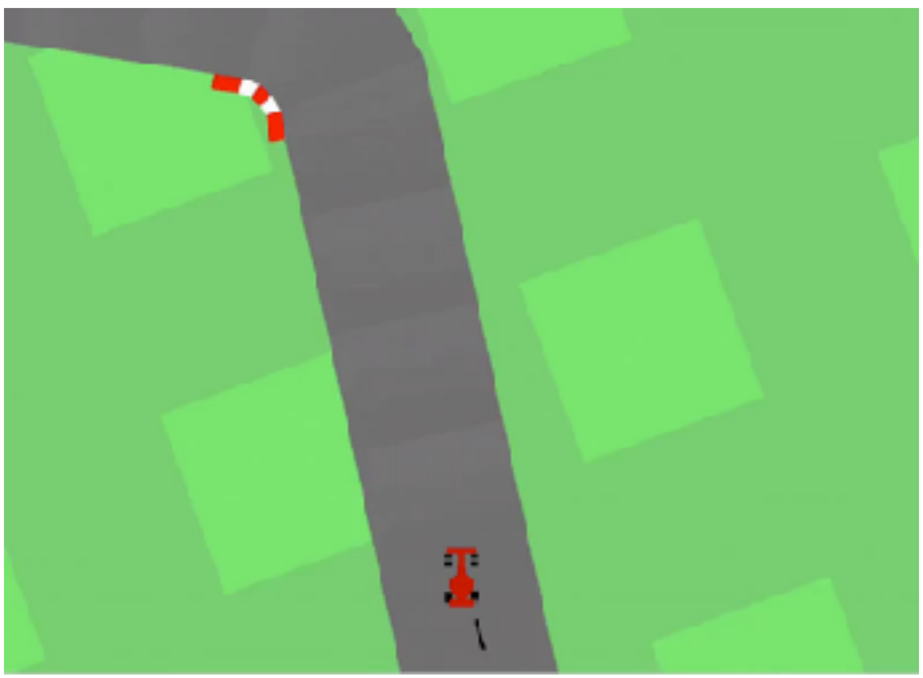
MODEL	PARAMETER COUNT
VAE	4,348,547
MDN-RNN	422,368
CONTROLLER	867



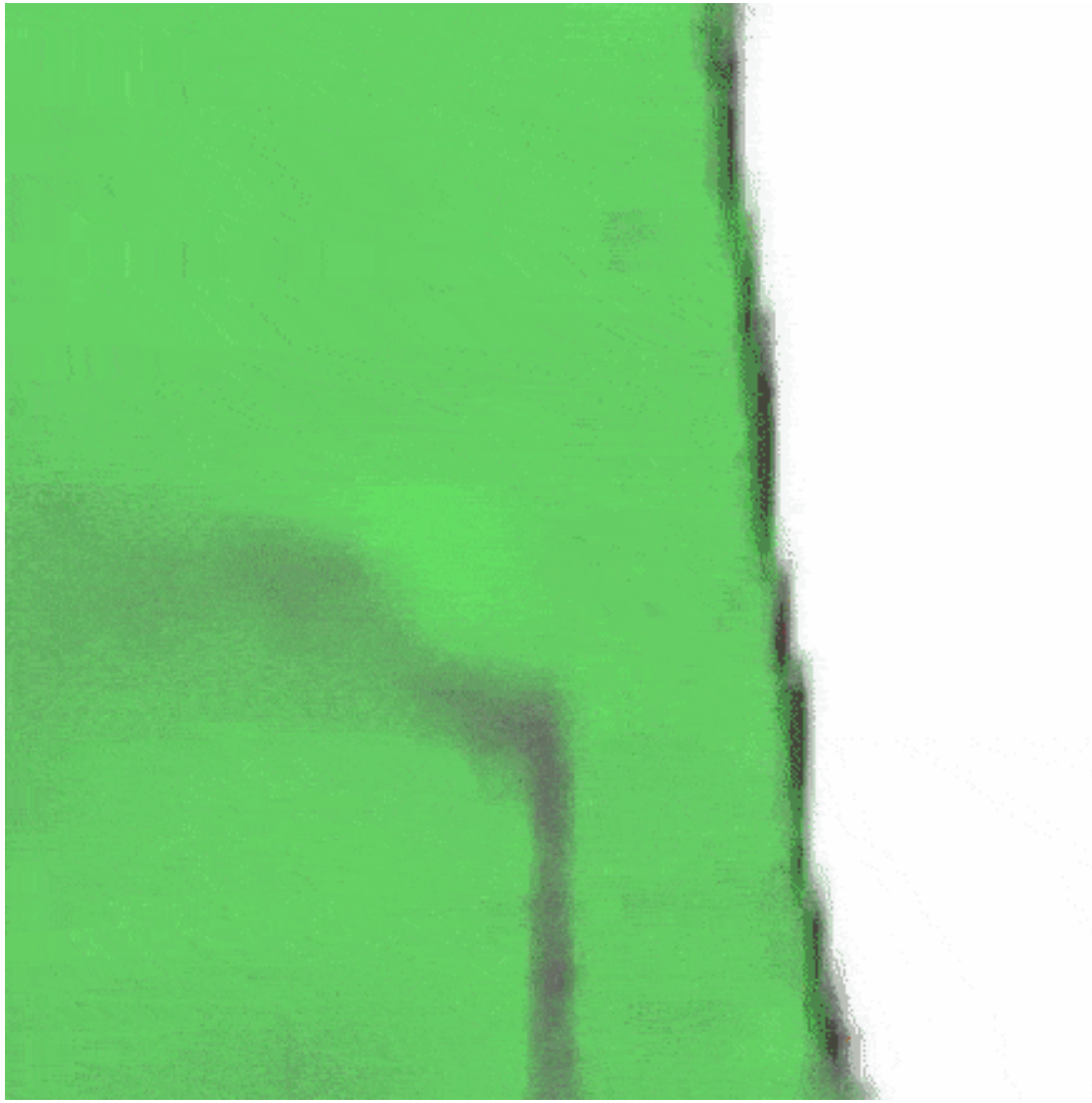
Car Racing Task - Results

Method	Avg. Score
DQN (Prieur, 2017)	343 ± 18
A3C (continuous) (Jang et al., 2017)	591 ± 45
A3C (discrete) (Khan & Elibol, 2016)	652 ± 10
ceobillionaire (Gym Leaderboard)	838 ± 11
V model	632 ± 251
V model with hidden layer	788 ± 141
Full World Model	$\mathbf{906 \pm 21}$

Table 1. `CarRacing-v0` scores achieved using various methods.



Car Racing Dreams



VizDoom: Take Cover

- Avoid Fireballs from Monsters
- Actions Space:

  [left / stay put / right]
- Reward is 1 for each frame survived.
- Max Reward = 2100 time steps
- Task is considered solved when average reward of 100 runs > 750 time steps

VizDoom: Take Cover - Training Procedure

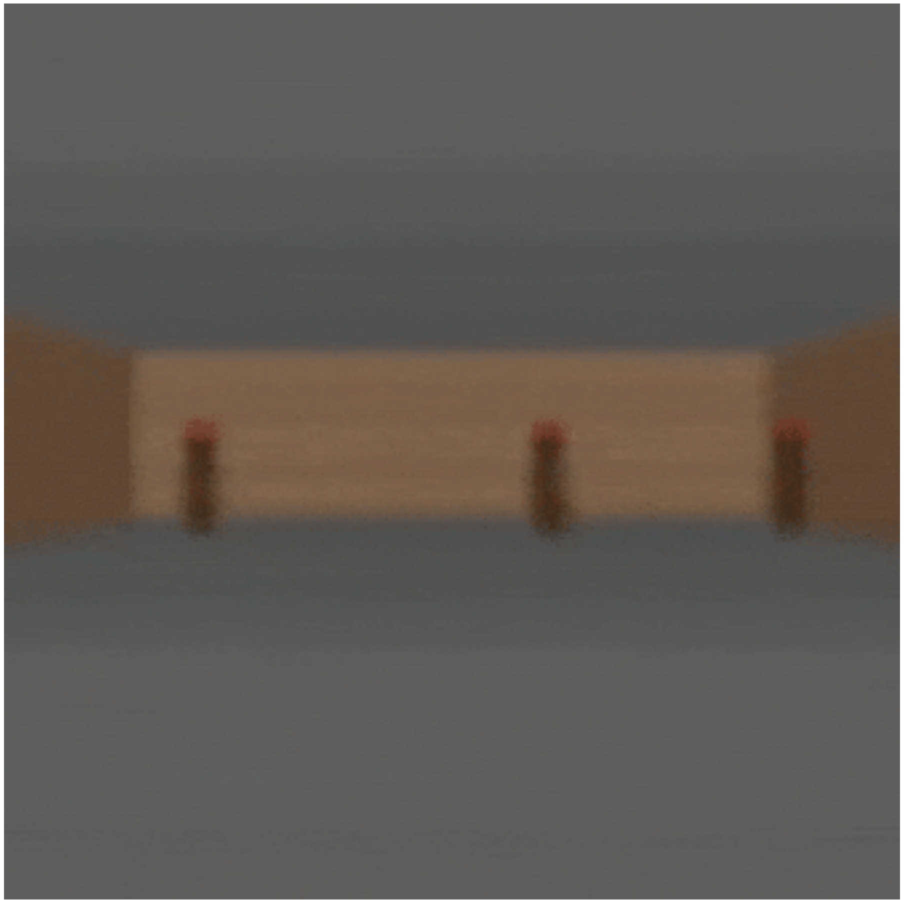
1. Collect 10,000 rollouts from a random policy.
2. Train VAE (V) to encode each frame into a latent vector $z \in \mathcal{R}^{64}$, and use V to convert the images collected from (1) into the latent space representation.
3. Train MDN-RNN (M) to model $P(z_{t+1}, d_{t+1} \mid a_t, z_t, h_t)$.
4. Define Controller (C) as $a_t = W_c \, [z_t \; h_t]$.
5. Use CMA-ES to solve for a W_c that maximizes the expected survival time inside the virtual environment.
6. Use learned policy from (5) on actual environment.

Model	Parameter Count
VAE	4,446,915
MDN-RNN	1,678,785
Controller	1,088

VizDoom: Take Cover - Adversarial Policy

Temperature τ	Virtual Score	Actual Score
0.10	2086 ± 140	193 ± 58
0.50	2060 ± 277	196 ± 50
1.00	1145 ± 690	868 ± 511
1.15	918 ± 546	1092 ± 556
1.30	732 ± 269	753 ± 139
Random Policy	N/A	210 ± 108
Gym Leader	N/A	820 ± 58

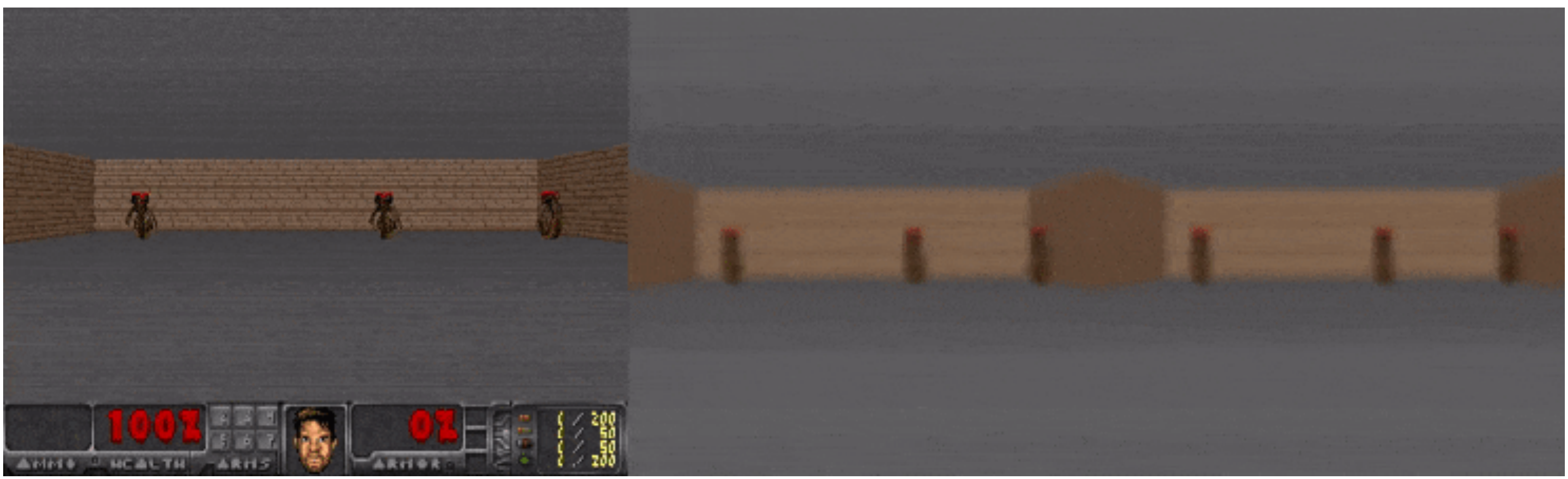
Table 2. Take Cover scores at various temperature settings.



- Agent learned actions to take advantage of flaws of virtual environment.
- Adjust temperature parameter in the sampling to control uncertainty.

Iterative Training Policy

1. Initialize M, C with random model parameters.
2. Rollout to actual environment N times. Save all actions a_t and observations x_t during rollouts to storage.
3. Train M to model $P(x_{t+1}, r_{t+1}, a_{t+1}, d_{t+1} | x_t, a_t, h_t)$ and train C to optimize expected rewards inside of M.
4. Go back to (2) if task has not been completed.



Discussion

- Hidden states + policy contain PDF of the future. No need to roll out future scenarios.
- Train in "latent-space" / "thought vector" land.
- Agent has access to all the hidden secret variables of the "game engine".
- TensorFlow Virtual environment is much more efficient tha VizDoom. Works on multi-threaded environment, MPI, GPU acceleration.
- Gaussian prior for VAE's z useful for correcting "bad samples" from MDN-RNN.
- Potentially useful to "explain" why an agent took a certain action (in both space and time).
- Camera-based robotic applications.
- Non-pixel-based complex tasks, such as Humanoid.