UiO : Department of Informatics
University of Oslo

Kyrre Glette
Robotics and Intelligent Systems group, Department of Informatics
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion

Evolutionary and adaptive robotics: from simulation to reality

Evolutionary and adaptive robotics (ER)

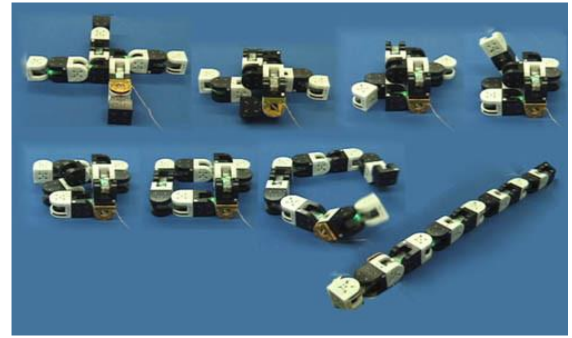
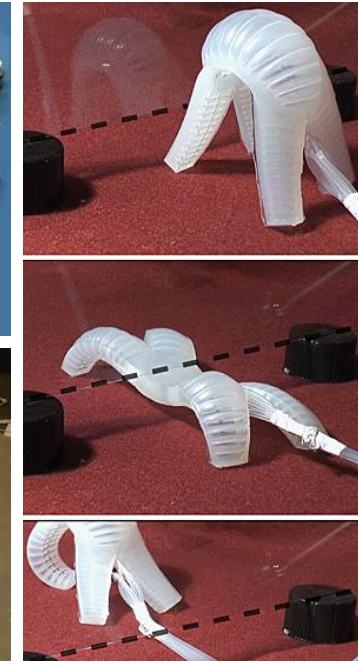
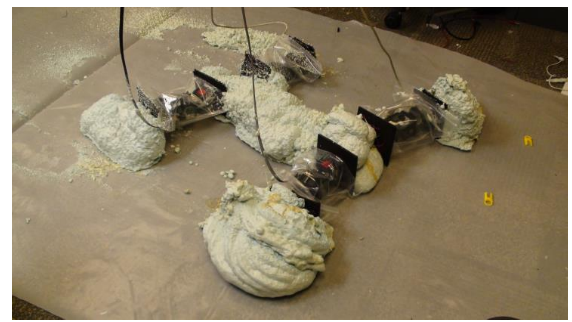
- Motivation
- Some basic concepts of *evolutionary algorithms* and other AI concepts with applications to robotics
  - More details about the algorithms in IN3050
  - Only a small taste of what's possible with AI and ML in robotics[1]
- Connections to our research in ROBIN
  - Robots and experiments

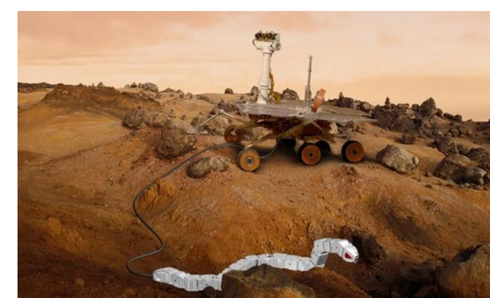
1) E.g. check this site to stay updated: https://spectrum.ieee.org/tag/robotics

Need for resilient and adaptive robots!

Motivation

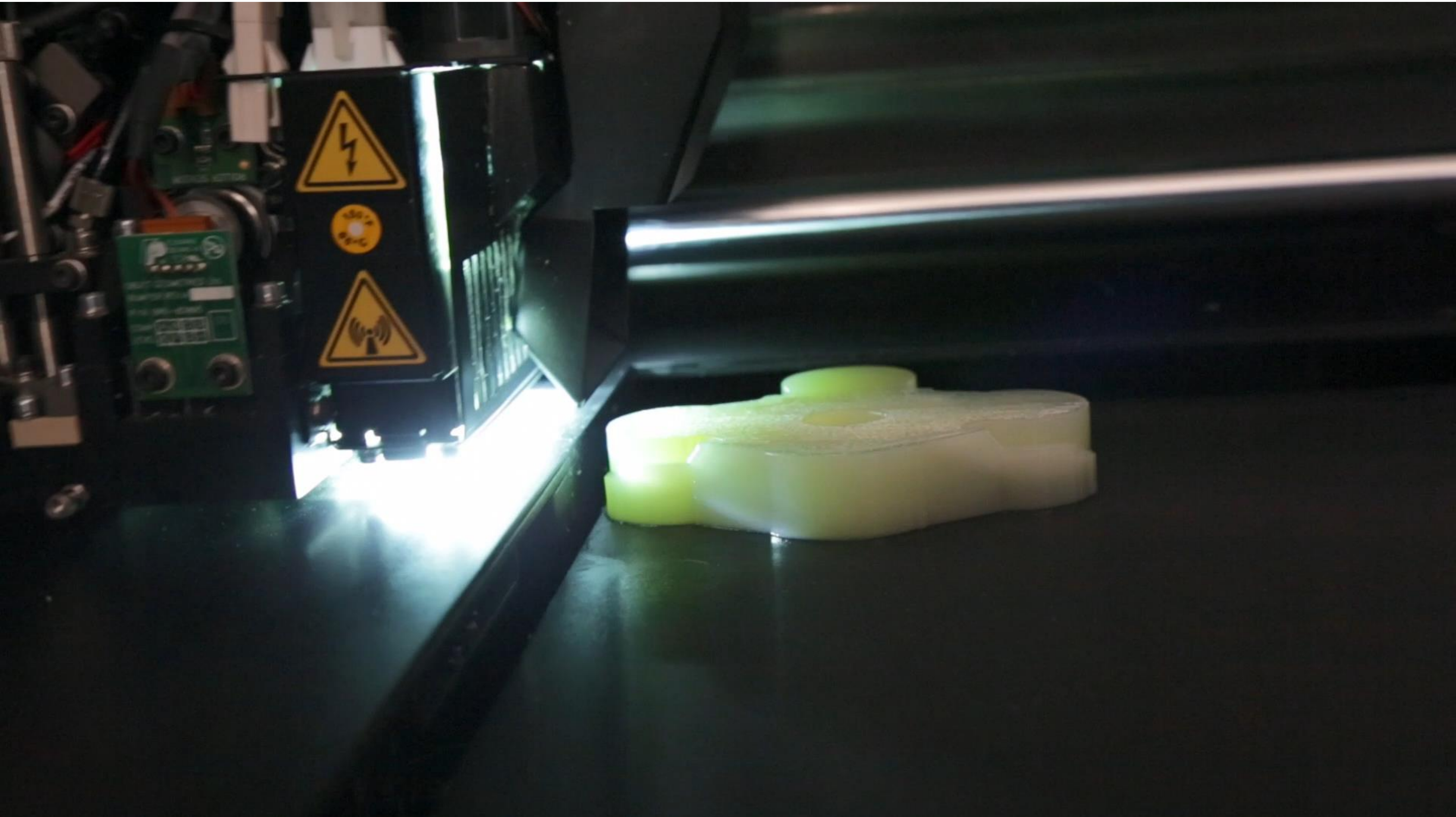
- New building blocks
  - Large space of robot body-behavior configurations
- Unseen and changing environments
  - Robots could / should adapt both body and behavior
- We need automatic design
  - To efficiently explore the design space
  - To autonomously adapt to the environment

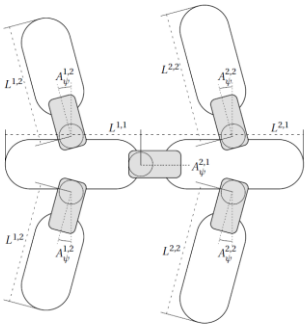


How can we automatically design a range of body-behavior approaches?

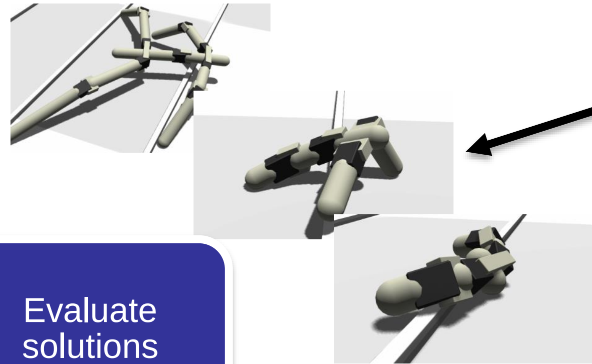
Our approach (Samuelsen et al.)



Evolutionary algorithm
Evolutionary robotics



Objective/fitness function:
Forward movement



Initialize random solutions

Evaluate solutions

Good solution to problem?

yes

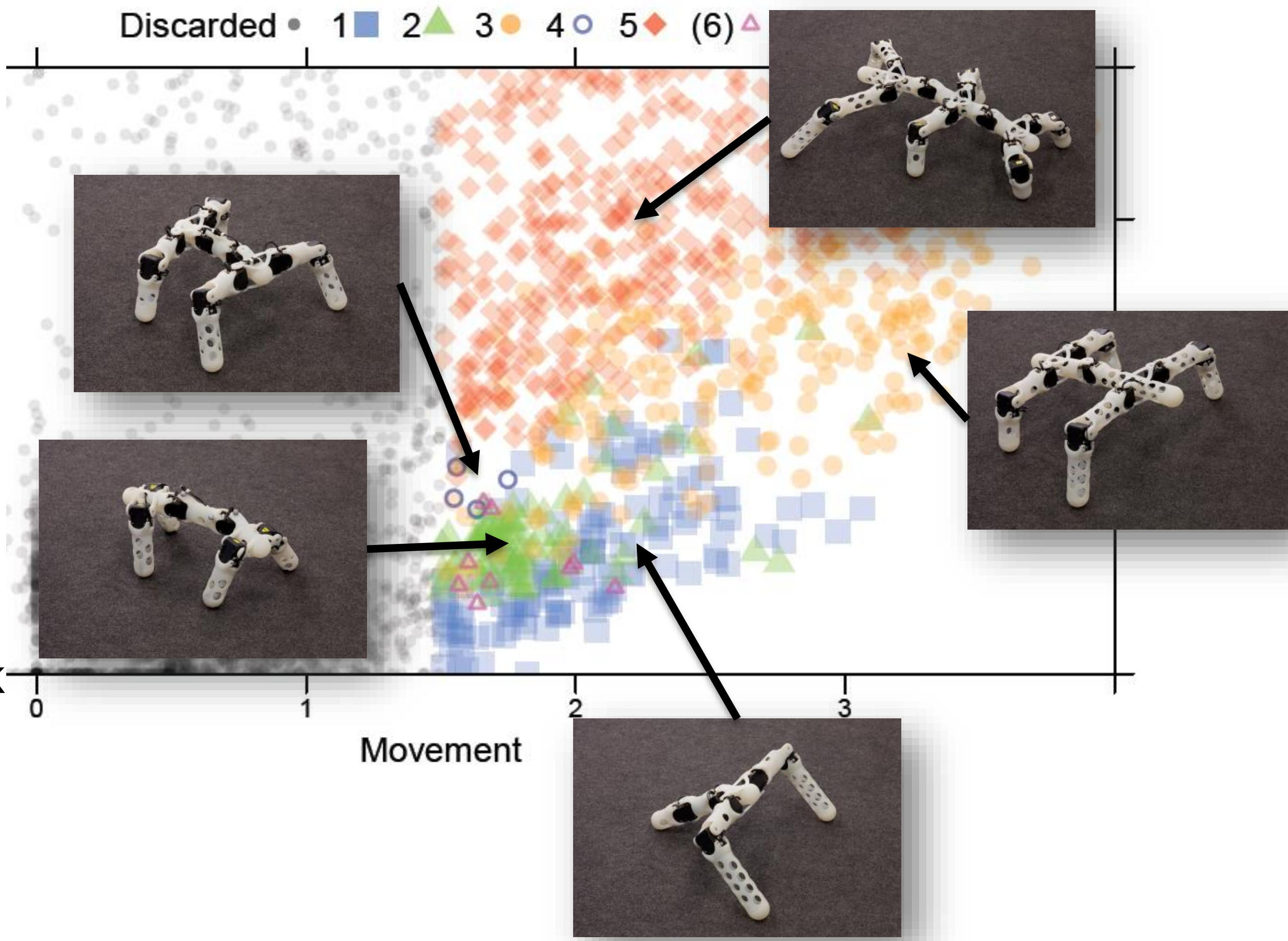
Use/produce solution

no

Create new solutions from good solutions

Evolve diverse solutions and select different concepts

- Multi-objective evolution
  – Tradeoffs speed and weight
- Evolve diversity
  – Morphological distance metric
  – Clustering to pick morphologically diverse solutions



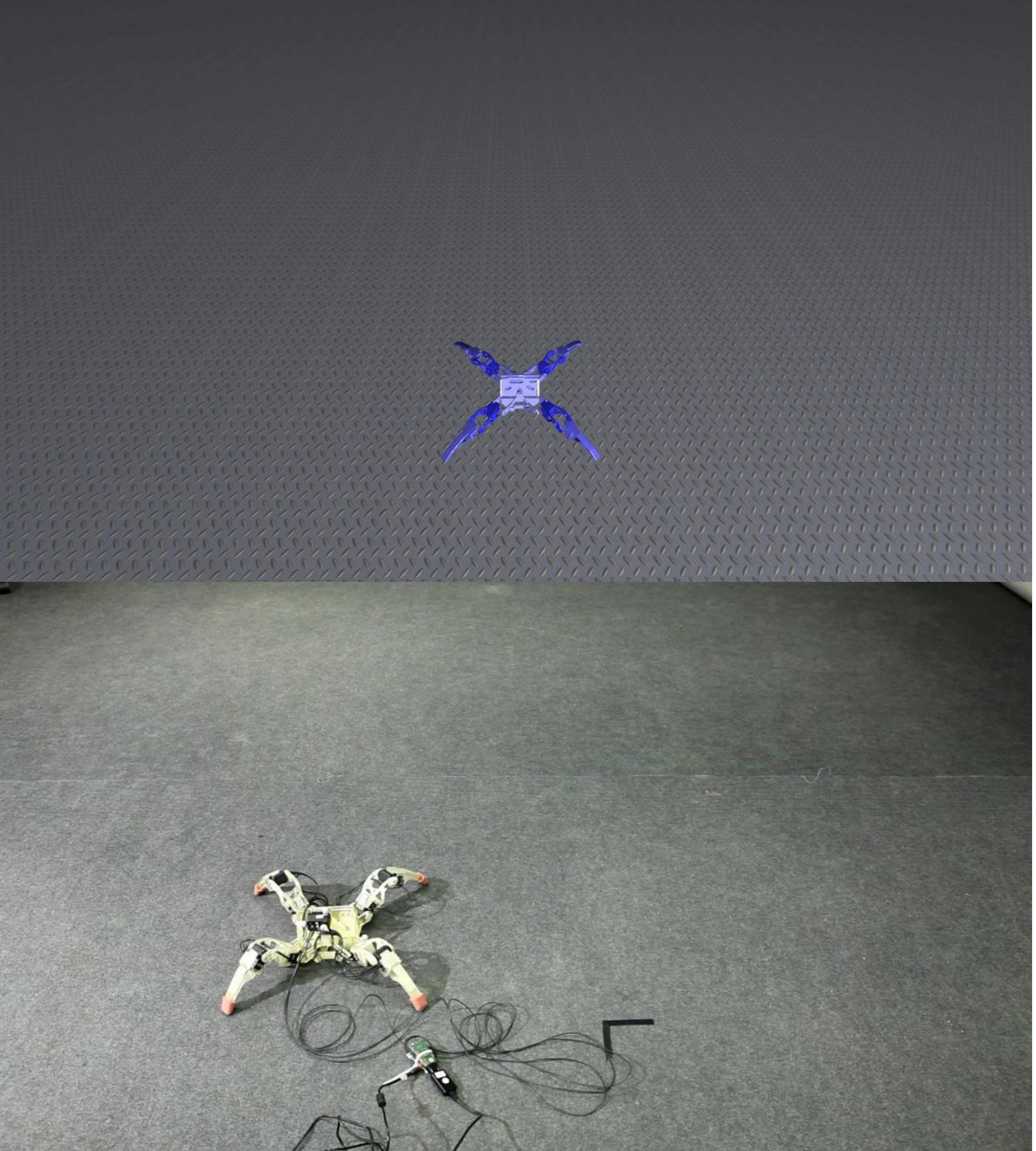
Challenge: Reality gap

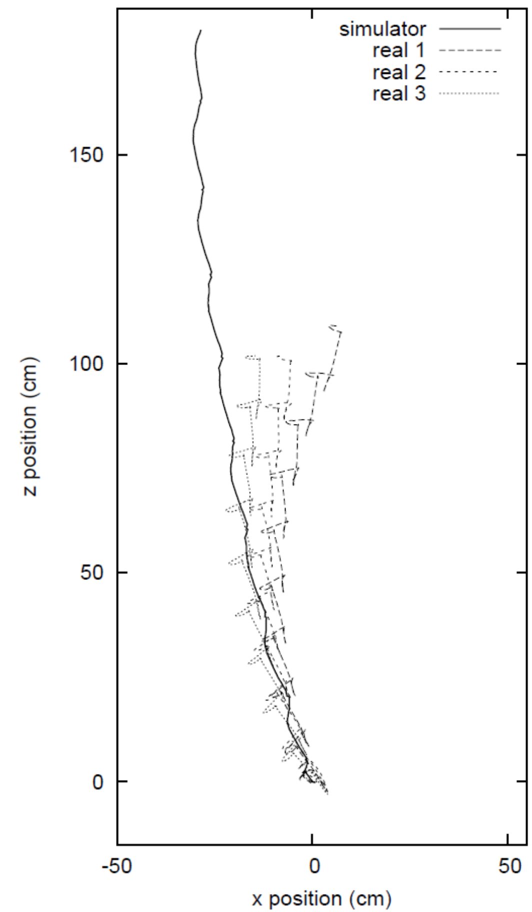
- Simulators are fast and can evaluate solutions in parallel, *but:*

- A simulator cannot capture all aspects of reality
- Evolved solutions may exploit features of the simulator not present in reality

→ The solutions evolved in simulation behave differently when applied to the real robot!

Reality gap example





How to deal with the reality gap?

- Ideas?

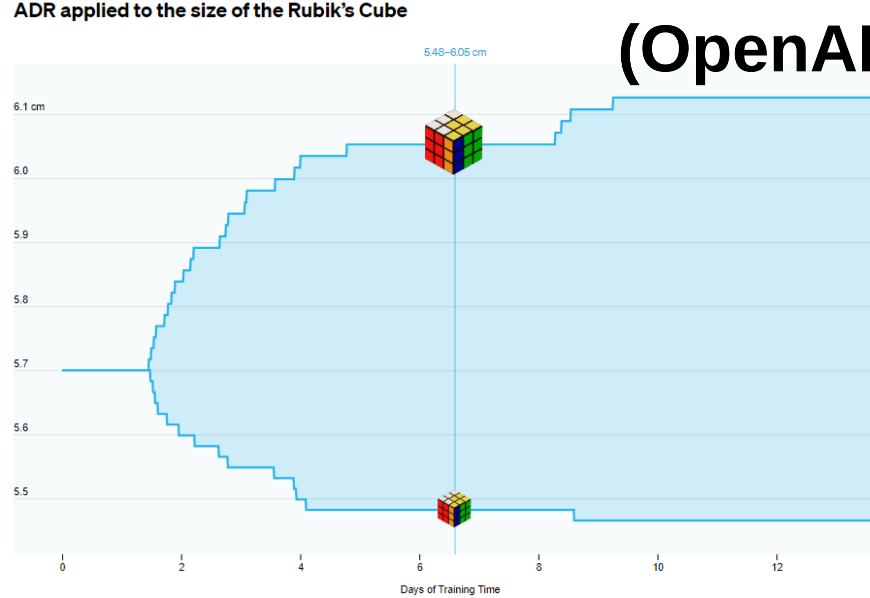
How to deal with the reality gap

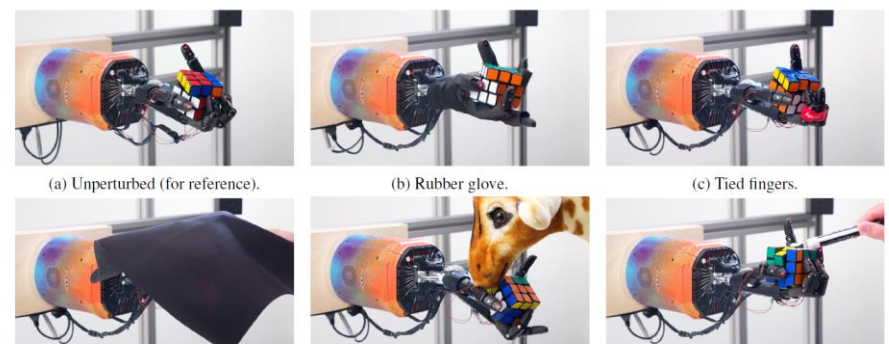
1. Increase simulation fidelity
   – Manually: do more precise measurements, increase computation spent on solving physics equations
   – Automatically: measure deviation simulation-reality, auto-tune simulator for smaller deviation
2. Encourage robustness
   – Manually: E.g. Encourage slow, static movements, *add noise*
   – Automatically: Avoid solution types that transfer poorly
3. Online learning after deployment on real robot
   – Can use evolution, reinforcement learning, or other method

Reality gap: adding noise / domain randomization (OpenAI)

- Deep reinforcement learning
- Noise encourages robust behaviors
- Progressively adding more «noise» / variation in the simulation
  - E.g. changing the cube size
- Lots of computation!
  - 64 V100 GPUs and 920 32-core CPUs training for several months
  - 13 000 «years of experience»

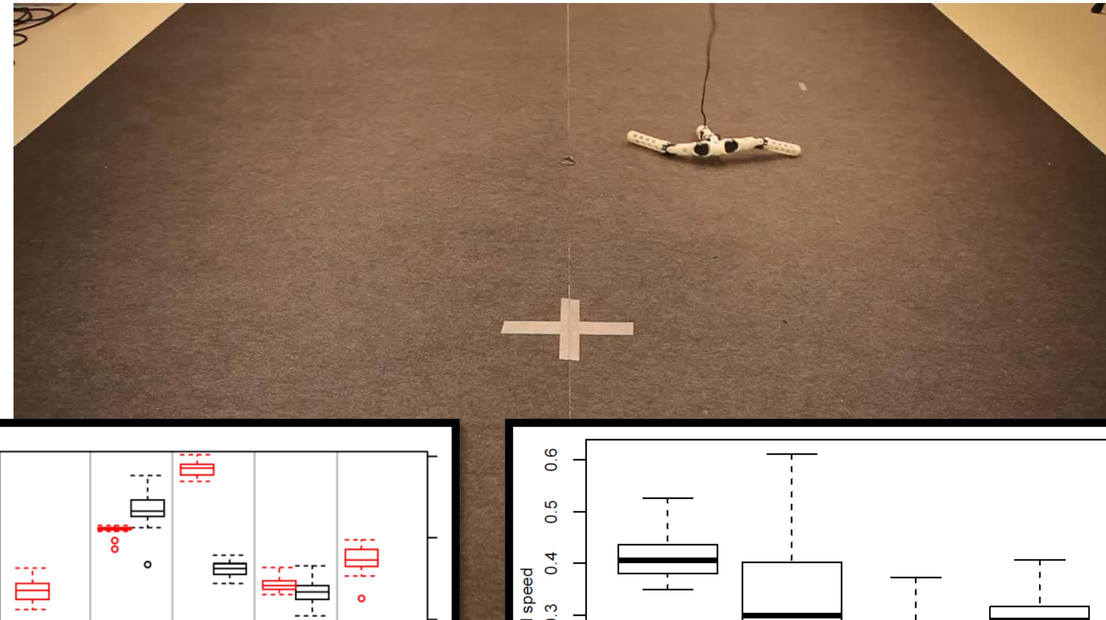
OpenAI Solving Rubik's Cube with a Robot Hand
https://arxiv.org/abs/1910.07113
https://openai.com/research/solving-rubiks-cube

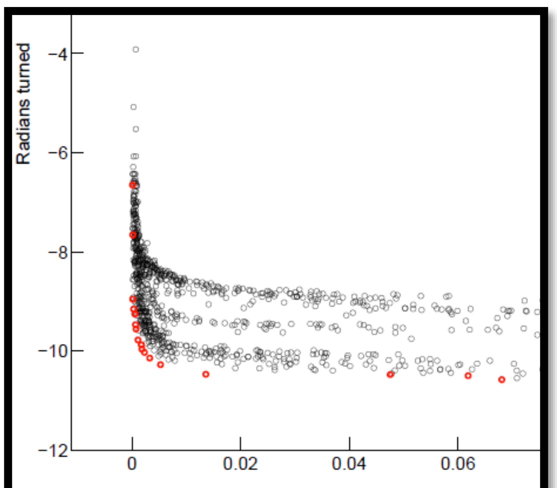


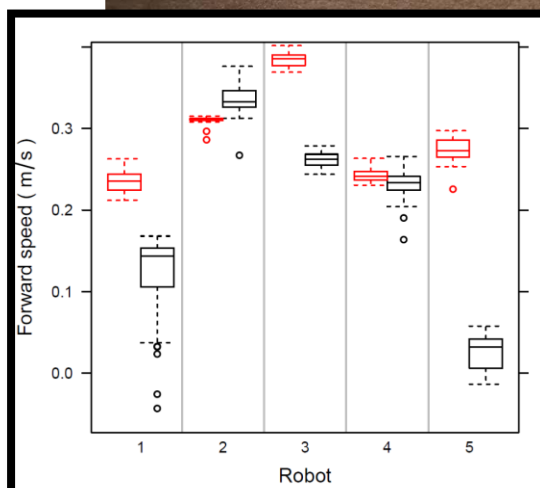

(a) Unperturbed (for reference). (b) Rubber glove. (c) Tied fingers.
(d) Blanket occlusion and perturbation. (e) Plush giraffe perturbation.[17] (f) Pen perturbation.

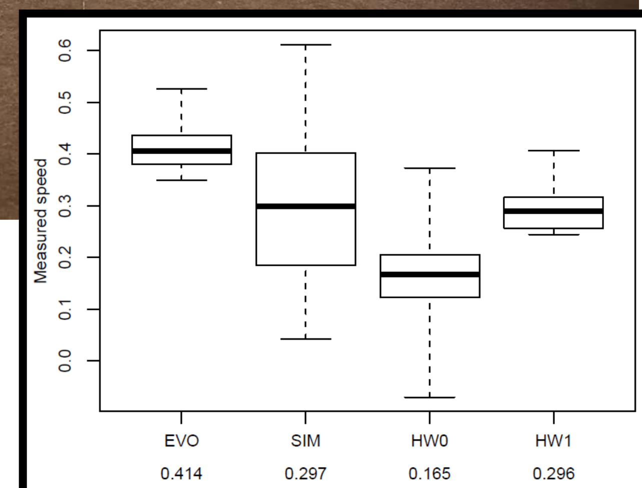
Reality gap: Evaluate and adapt solutions in the real world

- Learn to turn for arena evaluation
- Check real world performance
- Continued learning



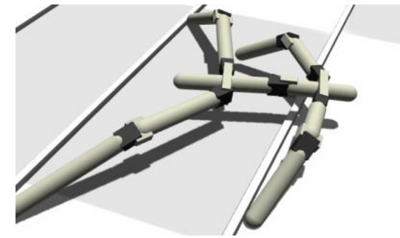


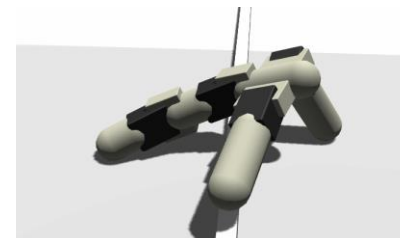


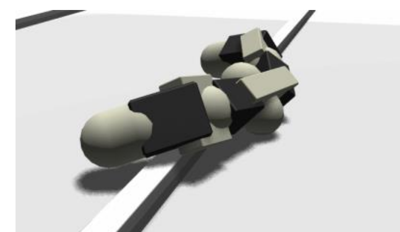


Morphology evolution challenges

- Reality gap can be large due to exploitation of simulator
- Time-consuming to produce one real-world instance of body+controller







Can we automatically search for bodies exploiting real-world characteristics?

Our approach (Nygaard et al.)

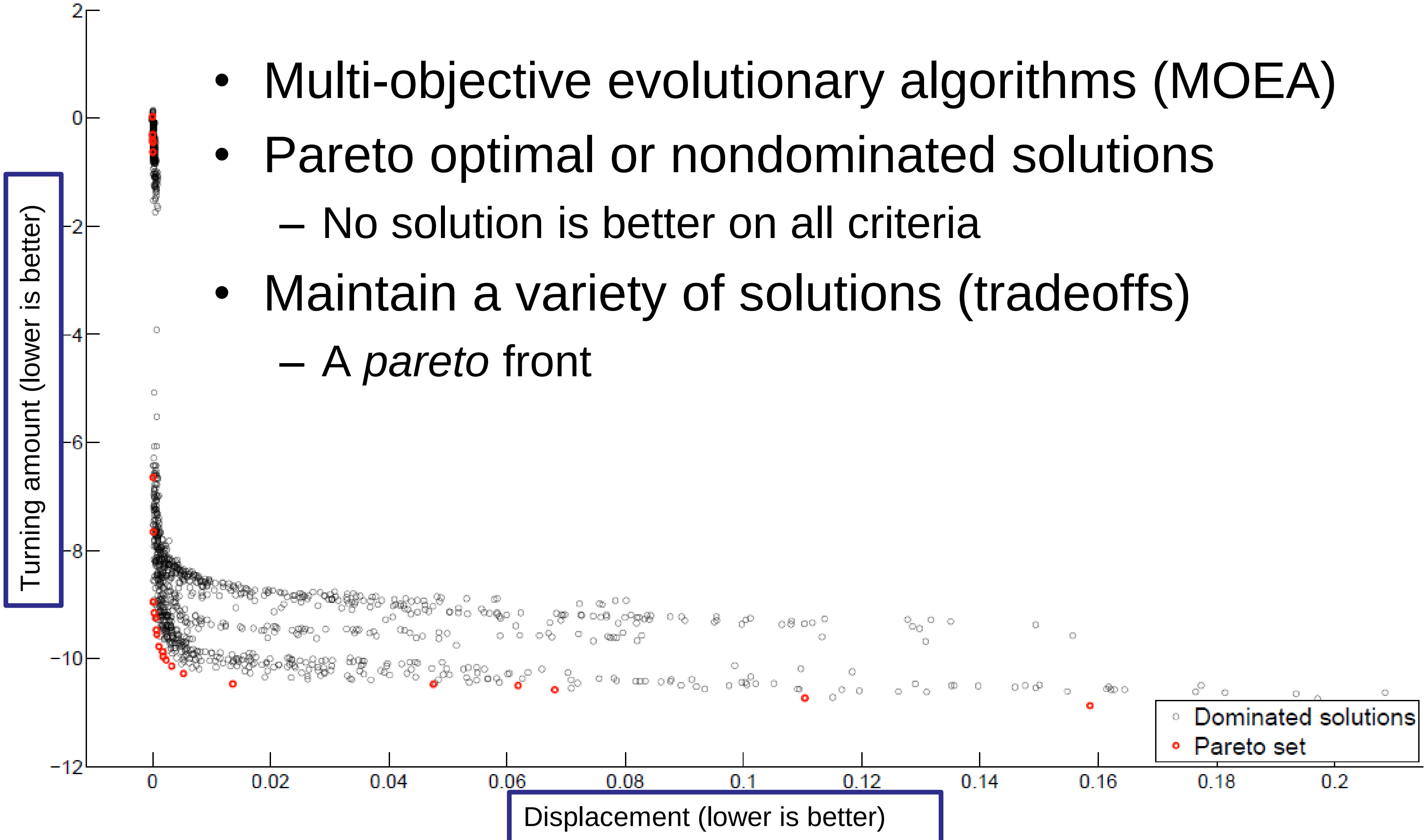
DyRET: Dynamic Robot for Embodied Testing

- Self-reconfiguring robot platform
  - Reconfiguration mechanism too slow to be actively used in gait
- Allows testing multiple morphological combinations using the same robot
- Real world evolution of morphology and control
- Adaptation to environment
- Open source and hardware

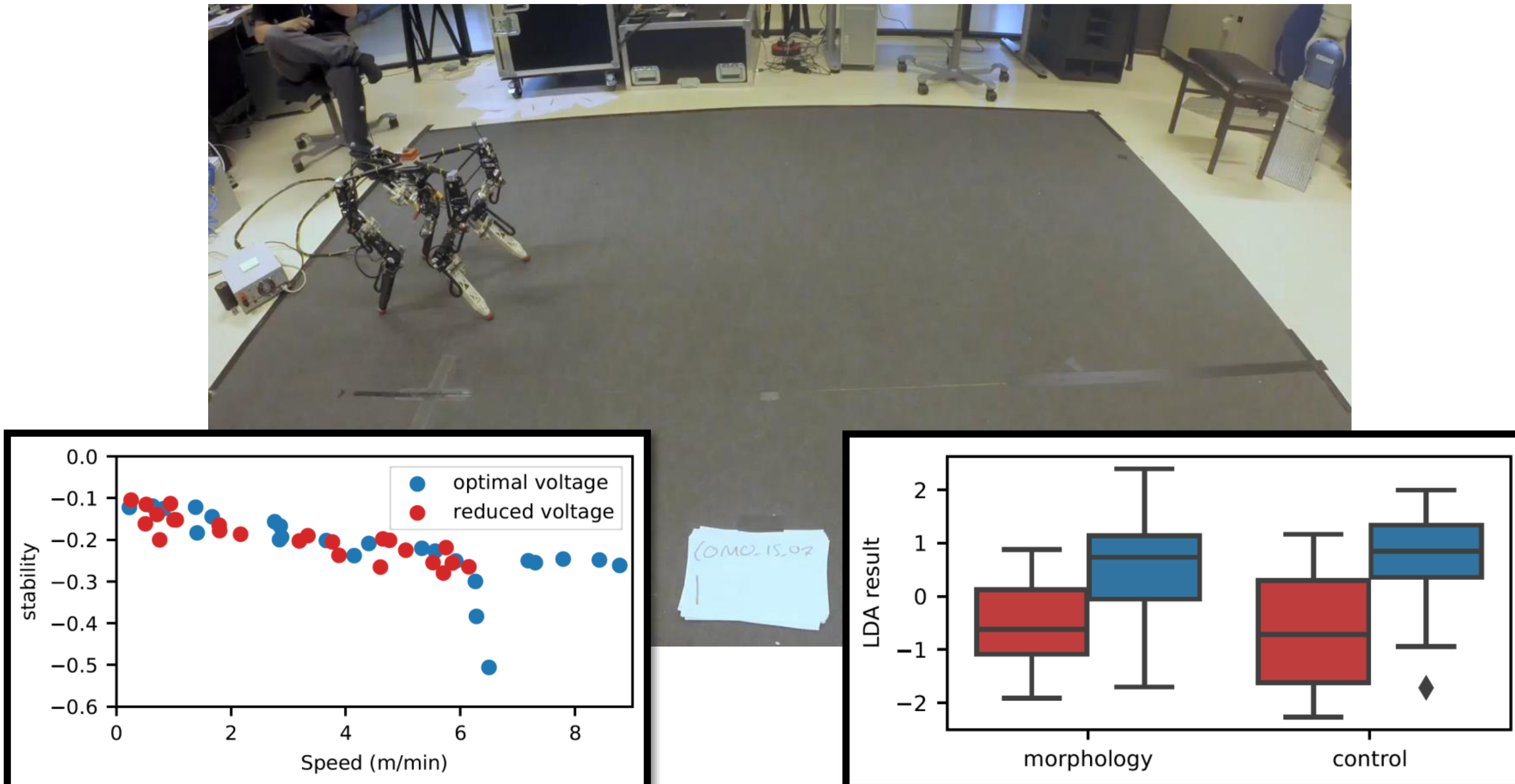
https://github.com/dyret-robot/dyret_documentation

Optimization for multiple solution criteria

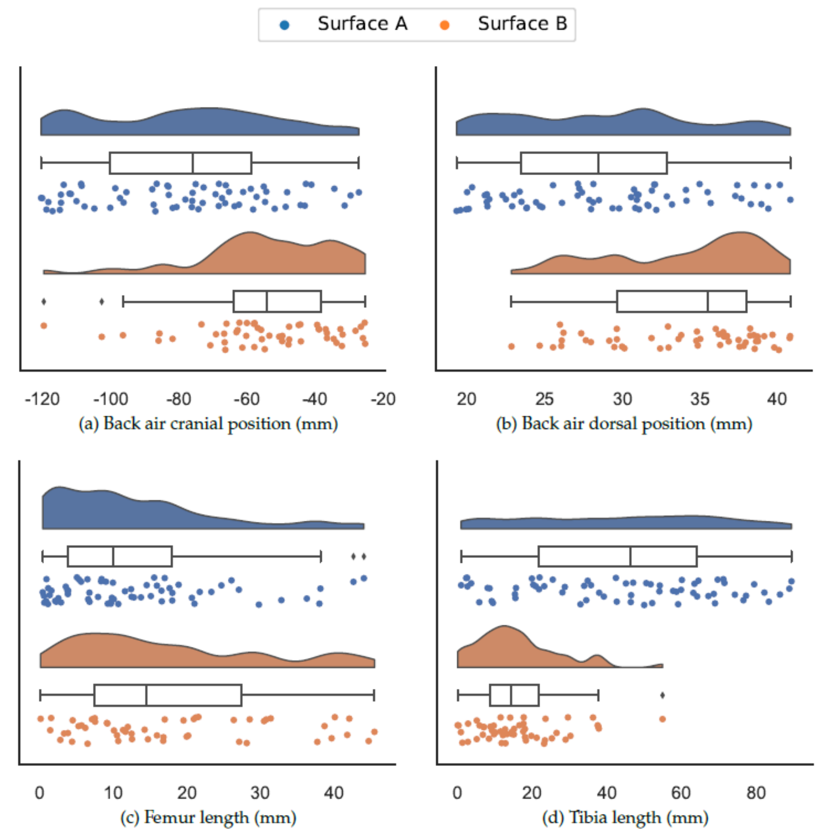
- Multi-objective evolutionary algorithms (MOEA)
- Pareto optimal or nondominated solutions
  – No solution is better on all criteria
- Maintain a variety of solutions (tradeoffs)
  – A *pareto* front



Real world morphology and control evolution

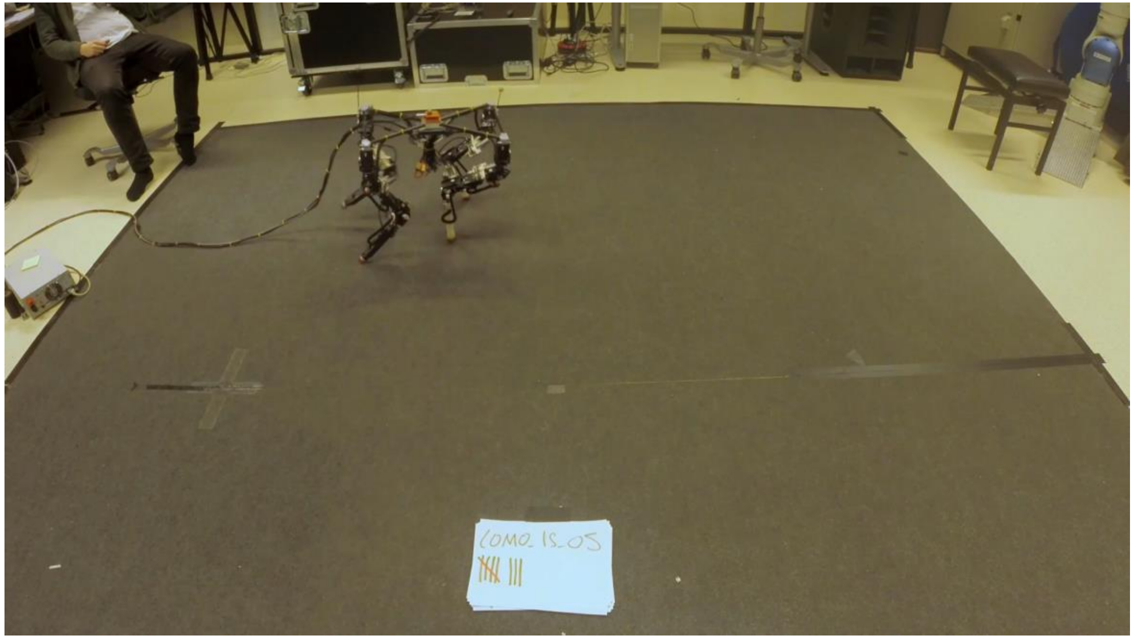


Evolution finds different bodies for different surfaces



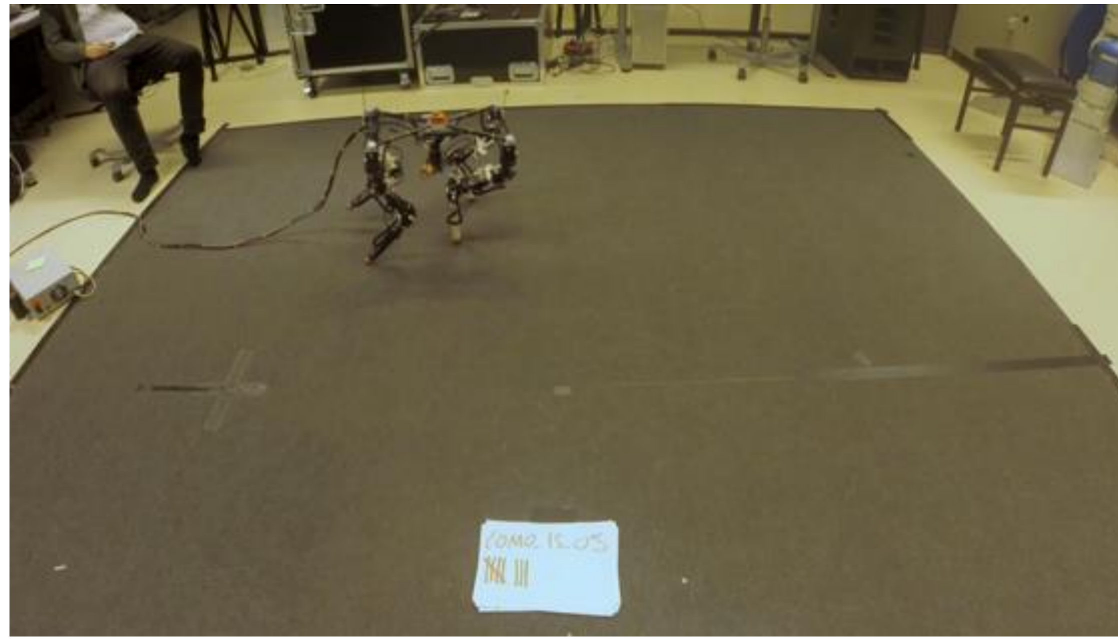
Challenges of real-world evolution

- Ideas?



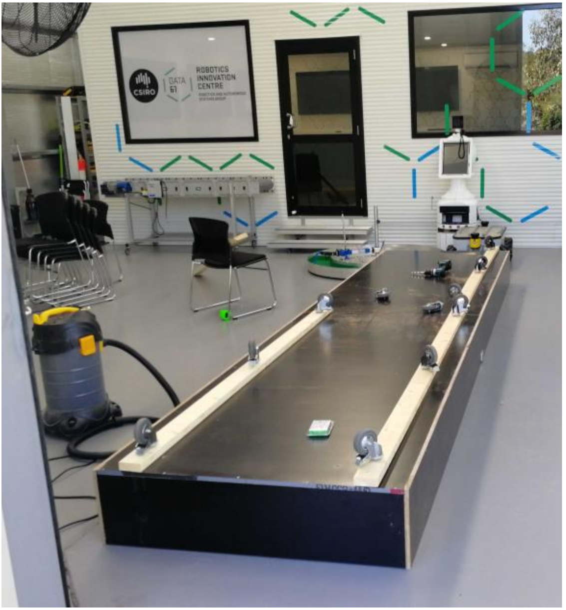
Challenges of real-world evolution

- Evaluation budget
  - One evaluation takes time!
  - This restricts the algorithms
- Wear and tear
  - Robot characteristics may change
- Less exploration
  - Morphology – cannot vary too much
  - Control / gait – cannot fall all the time



How to adapt a robot's body on the fly using machine learning

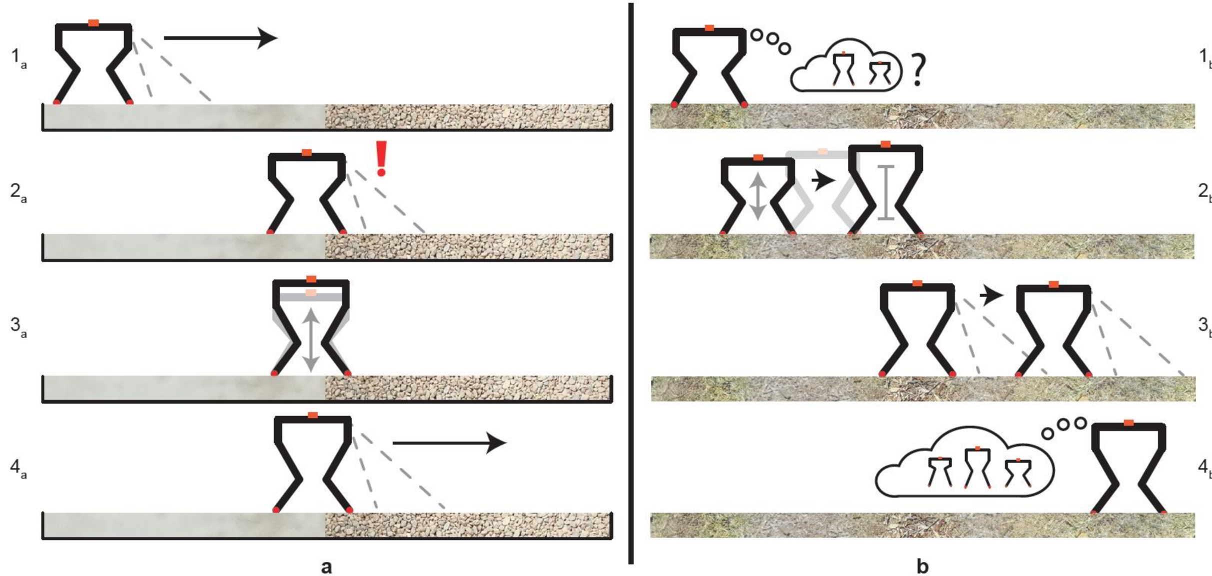
More terrains at CSIRO

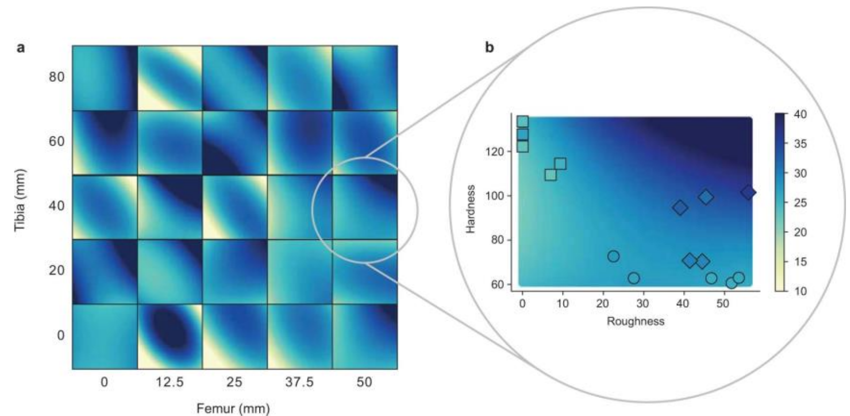


Real-world Embodied AI Through a Morphologically Adaptive Quadruped Robot

Building an ML model

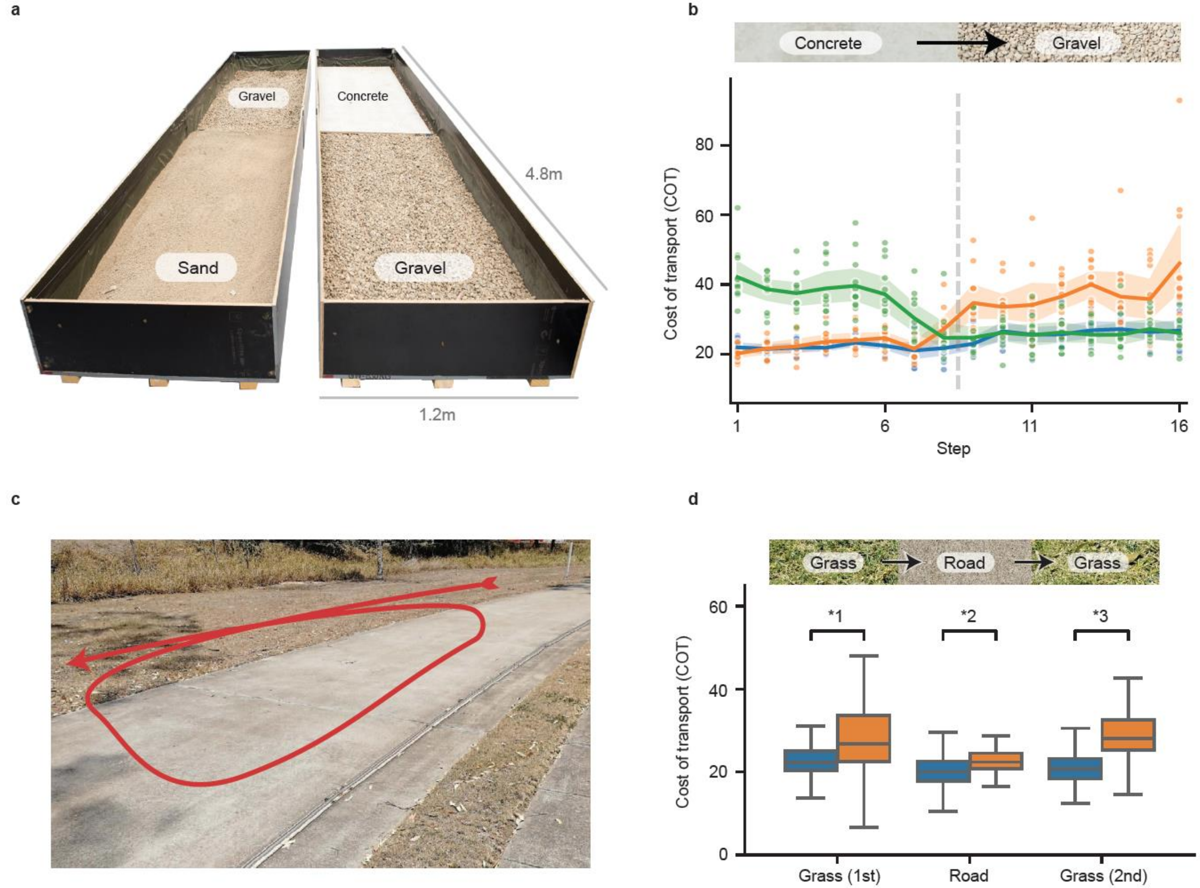
- Terrain sensing & characterization
- Choose body configuration
- Measure performance
- Update model





Different morphologies preferred in different environments

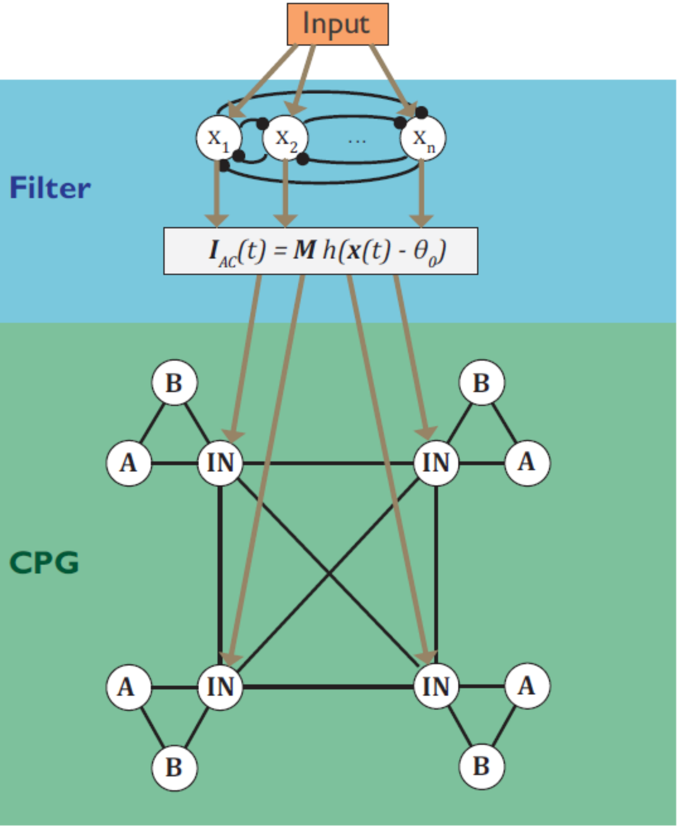
- Morphology can be a part of problem solving
  - Embodied AI
- Demonstrated on a real robot / real terrain
- Better than using the best fixed morphology

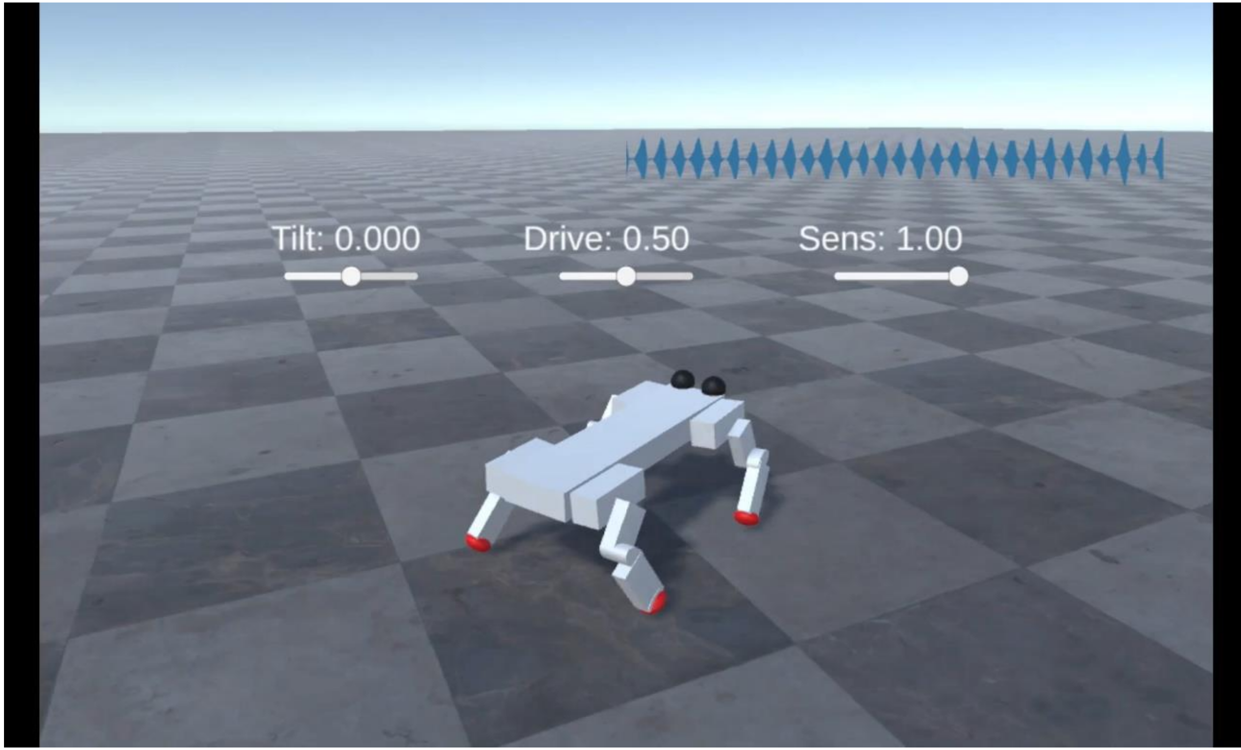


Alternatives to evolutionary robotics

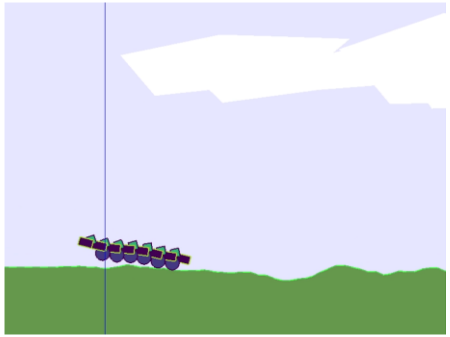
- Bayesian optimization
  - Builds models of performance based on observed data
    (surrogate models)
  - Can be much more data efficient, but less exploration
  - Often used for real-world optimization of robot controllers
- (Deep) reinforcement learning
  - Typically uses a neural network to control for each time step
  - Makes use of data from each simulation time step
  - More data efficient, but less exploration

Bonus: Can we make robots dance? (Szorkovszky)

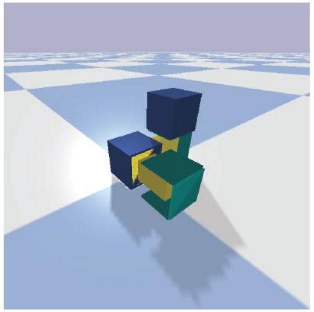


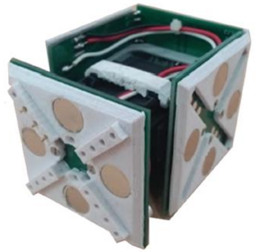


Choose platform according to task

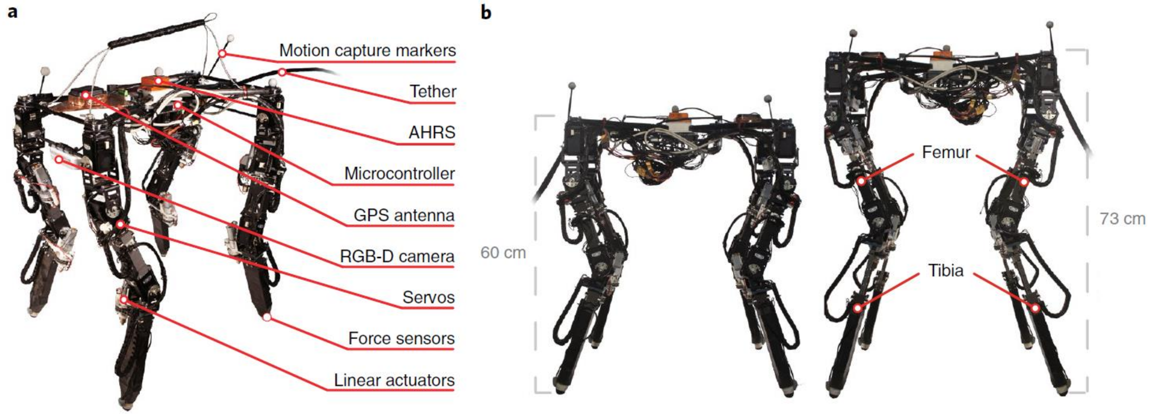


- Variable realism and evaluation cost
  - E.g. 2D vs. 3D simulation
  - Hardware in the loop?



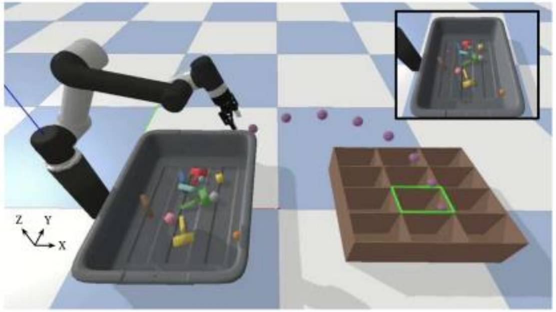


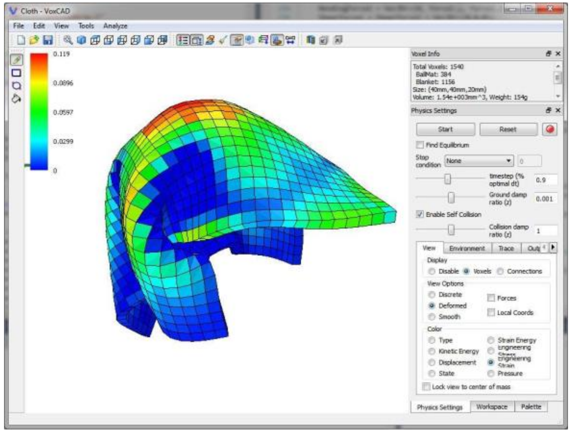
- AI "gyms" – simple task-specific setup
- ROS – complete

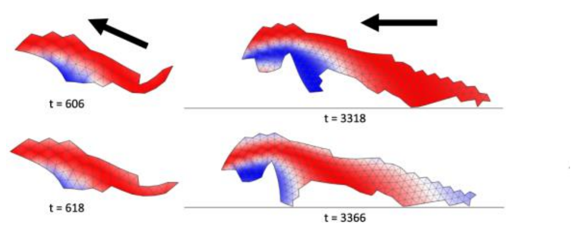


Simulation systems

- Rigid body physics simulations
  - PhysX, Bullet, MuJoCo, DART, Box2D, …
- Other simulation systems
  - Voxcad
  - Custom mass-spring-damper systems
  - Soft models incorporated in rigid-body engines
- Higher realism (slower)
  - FEM simulations
  - SOFA framework: rigid, deformable, fluid
- High-level wrappers
  - Robot-centric: Isaac Sim, Coppeliasim, Gazebo
  - Game-centric: Unreal Engine, Unity
  - «Gyms»: Gymnasium, evolution-gym

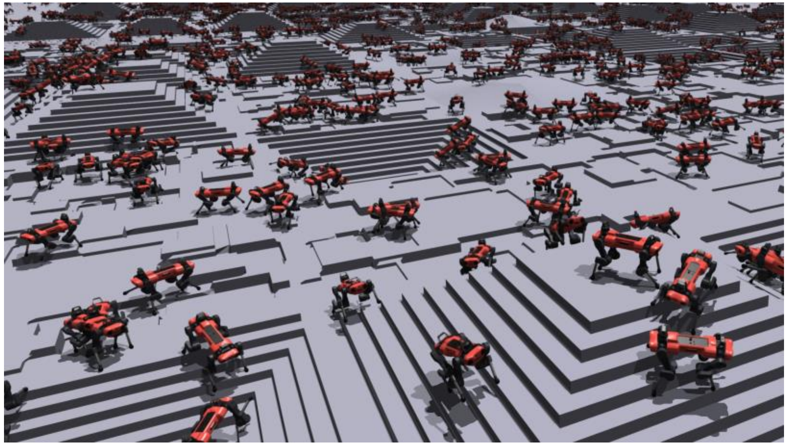


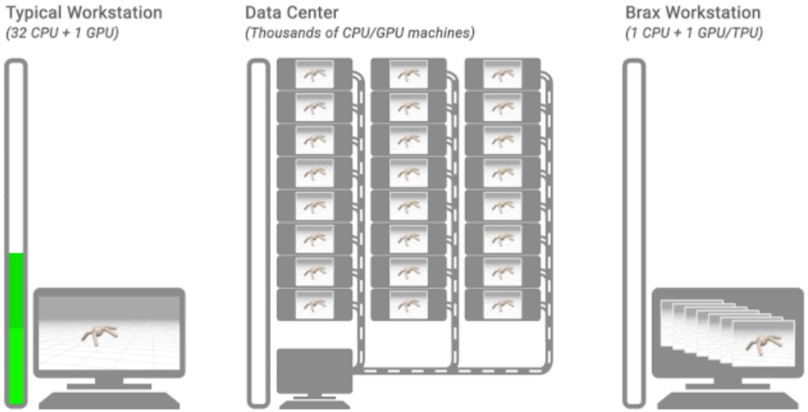




New simulators leverage GPUs for speedups

- E.g. BRAX, IsaacSim
- Simulation code rewritten to work well on GPU architecture
- 1000s of simulations in parallel on a single GPU
- The learning algorithm can run on the same GPU - reduced latency

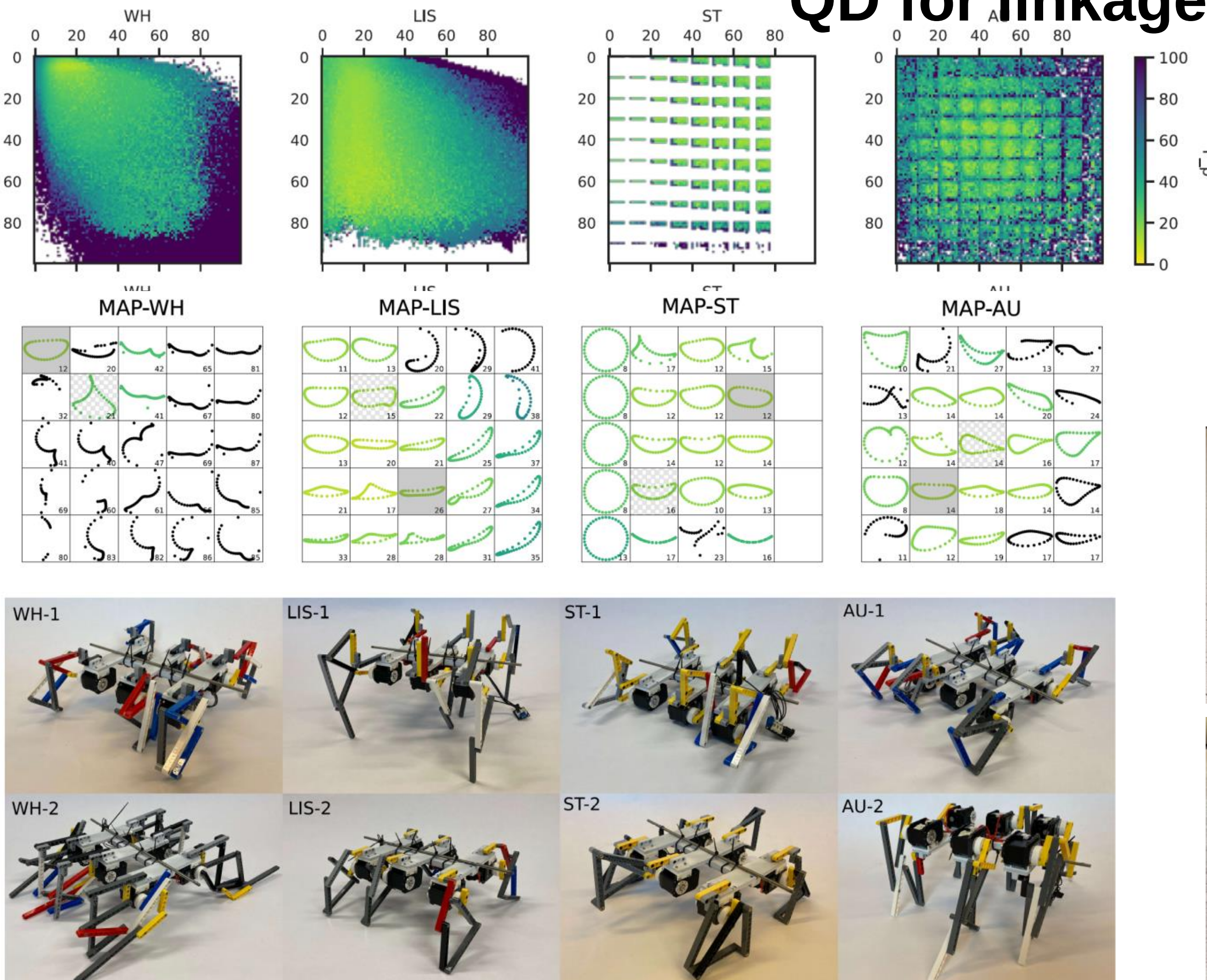


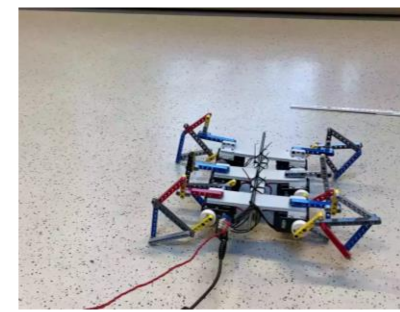


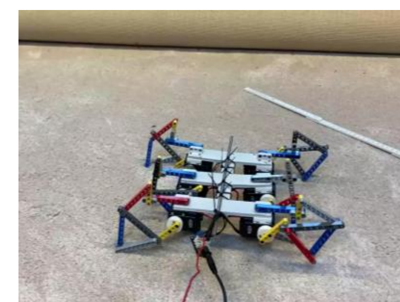
https://research.google/blog/speeding-up-reinforcement-learning-with-a-new-physics-simulation-engine/

QD for linkage robot design

(Norstein et al.)

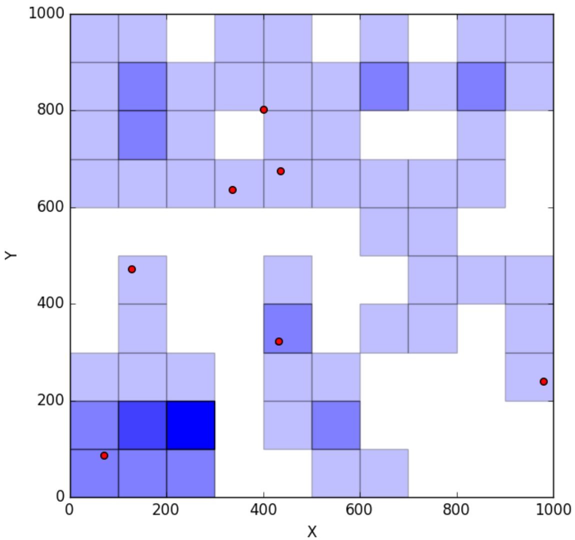


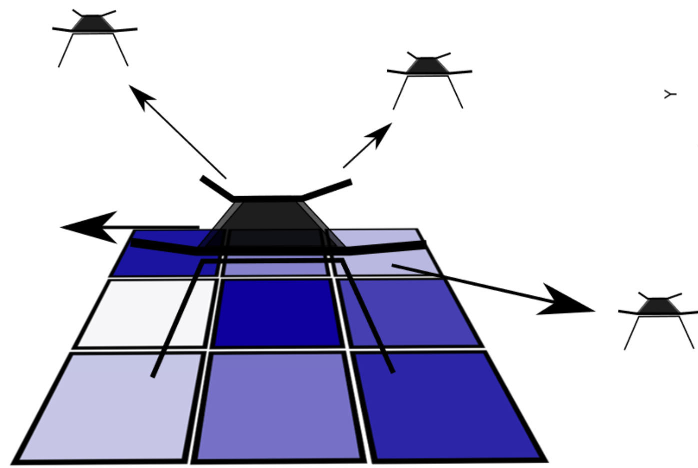




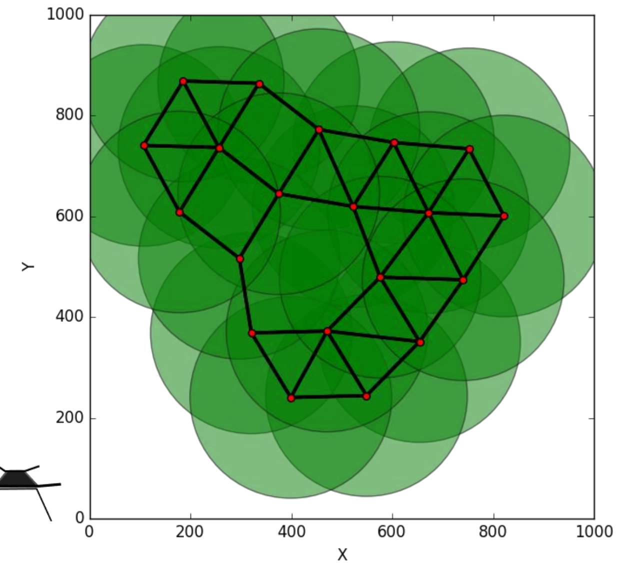
Multi-function swarm (Engebråten et al.)

Exploration (search)

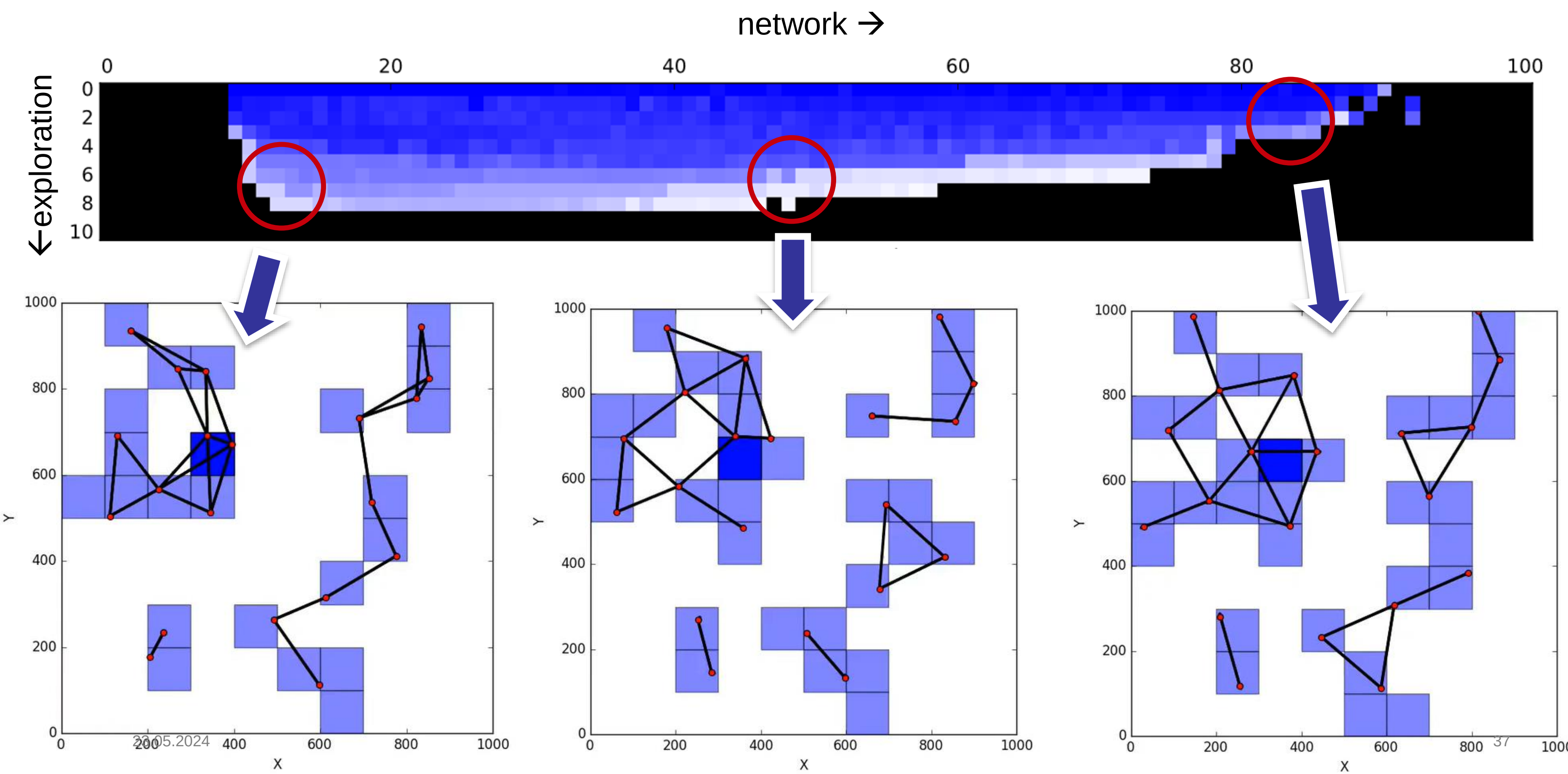




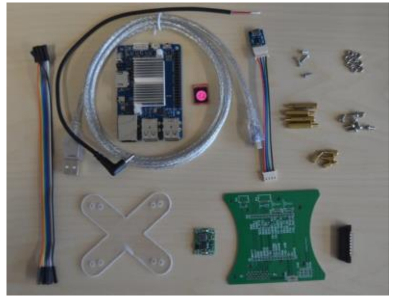
Networking (maintaining coverage)

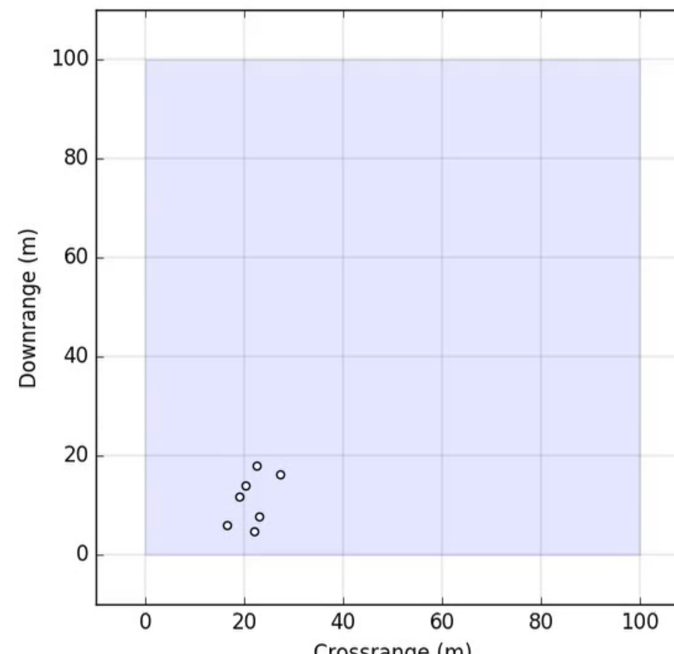


Controller repertoire generation



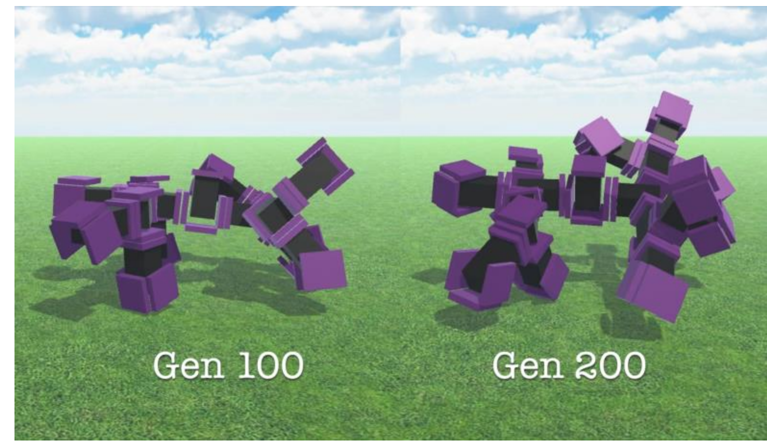
Real-world testing





Example MSc project: Evolving modular robots

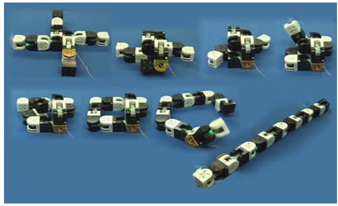
- Simulation only
  - Building blocks based loosely on real modular robot
- Investigating impact of control mechanisms on evolved robots
  - Centralized vs decentralized control approach
  - Effects on morphology evolution



Kvalsund: Search Space Traversal in Co-Optimized Modular Robots https://www.duo.uio.no/handle/10852/95656

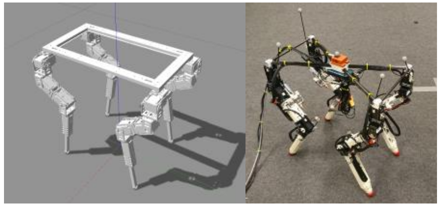
Summary

- Evolutionary robotics can be useful for adaptation, optimization, design exploration
  - Optimization
  - Exploring many different solutions
  - Exploring trade-off solutions



- Co-evolution of body and control is possible
  - Wholistic design for given tasks / environments

- From simulation to hardware
  - Sample cost
  - The reality gap



Questions

Bonus material

[Submitted on 25 Sep 2023]

Extreme Parkour with Legged Robots

Xuxin Cheng, Kexin Shi, Ananye Agarwal, Deepak Pathak

CoRL: Quadruped parkour (CMU)

https://extreme-parkour.github.io/

Quadruped parkour

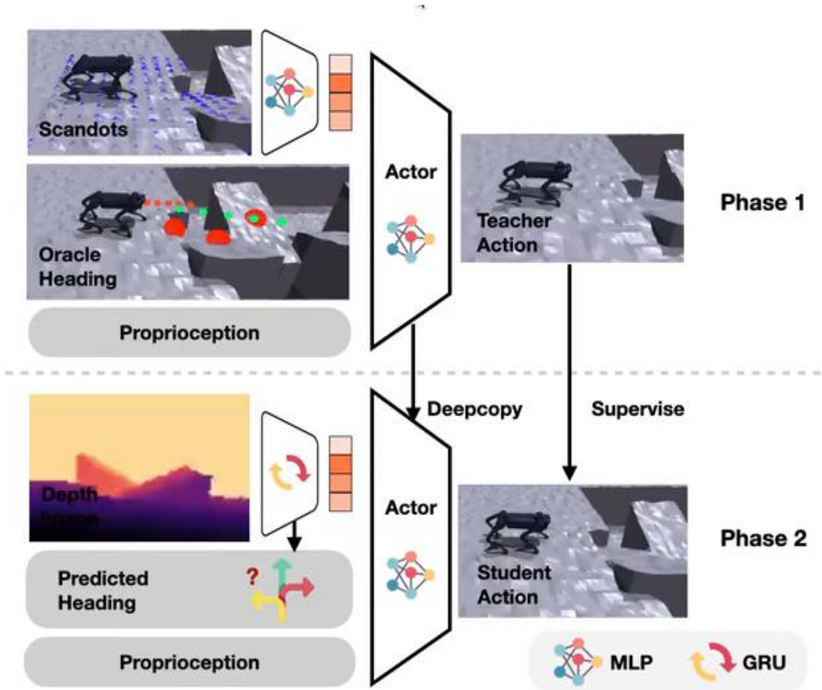


Figure 2: Training overview. In phase 1, we use RL to learn a locomotion policy with access to privileged information like environment parameters and scandots [2] in addition to heading direction from waypoints. We use Regularized Online Adaptation (ROA)[9] to train an estimator to recover environmental information from the history of observations. In phase 2, we distill from scandots into a policy that operates from onboard depth and *automatically decides its heading (yaw) direction* conditioned on the obstacle.

- Single end-to-end NN
- Vision backbone + policy
- Depth image + proprioception

- Train 20hrs RTX3090
- Run Jetson NX 50Hz

Google DeepMind

https://robotics-transformer2.github.io

28 July 2023

RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich

Google DeepMind. Authors listed in alphabetical order, with contributions listed in Appendix A.

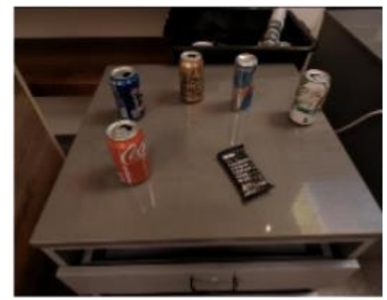
Robotic foundation models

https://robotics-transformer2.github.io/

Prompt:
Given <img> Instruction:
Bring me a drink.
Prediction:
Plan: pick 7up can.
Action: 1 143 129 123 145 114 115 127

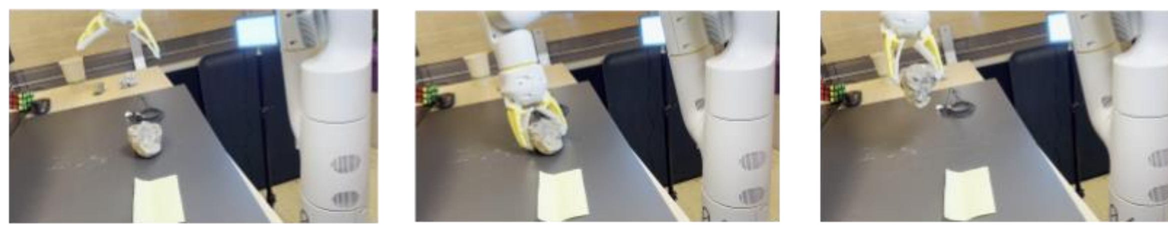
Prompt:
Given <img> Instruction:
Move all the objects together.
Prediction:
Plan: move green can near green rice chip bag.
Action: 1 128 126 127 135 123 119 127

Prompt:
Given <img> Instruction:
Pick the object that is different from all other objects
Prediction:
Plan: pick rxbar chocolate. Action: 1 128 129 125 131 125 128 127



Prompt:
Given <img> Instruction:
Move the green objects together.
Prediction:
Plan: move green can near green rice chip bag.
Action: 1 130 129 121 131 127 128 127

Prompt:
Given <img> I need to hammer a nail, what object from the scene might be useful?
Prediction:
Rocks. Action: 1 129 138 122 132 135 106 127



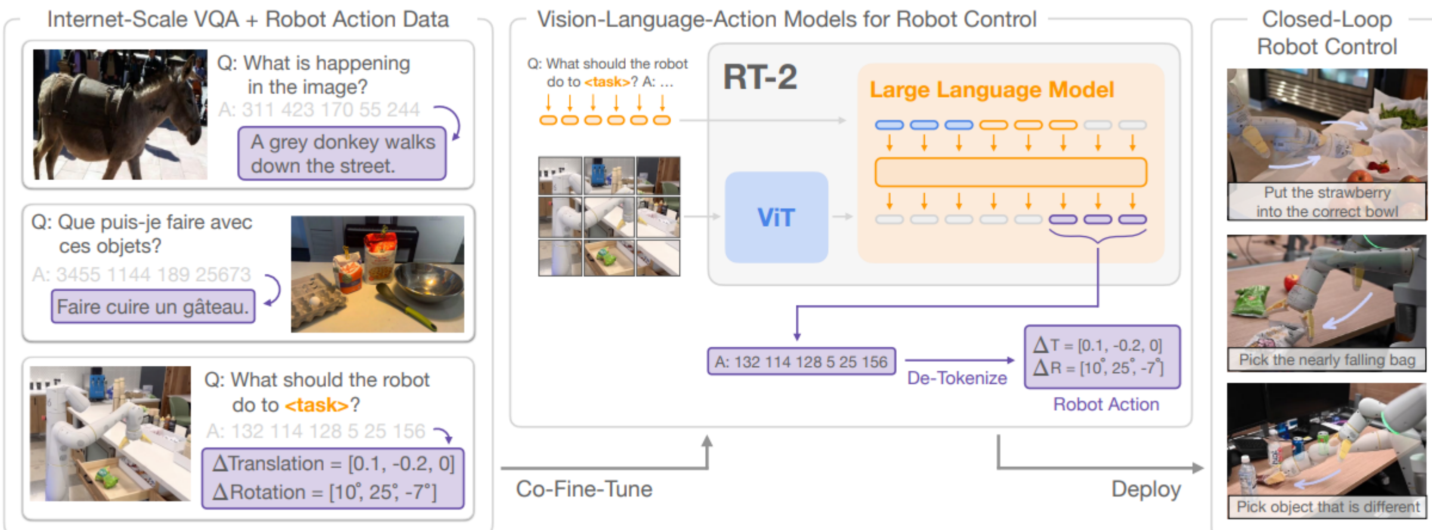


Figure 1 | RT-2 overview: we represent robot actions as another language, which can be cast into text tokens and trained together with Internet-scale vision-language datasets. During inference, the text tokens are de-tokenized into robot actions, enabling closed loop control. This allows us to leverage the backbone and pretraining of vision-language models in learning robotic policies, transferring some of their generalization, semantic understanding, and reasoning to robotic control. We demonstrate examples of RT-2 execution on the project website: robotics-transformer2.github.io.