Micro-Data Reinforcement Learning for Adaptive Robots

Konstantinos Chatzilygeroudis
Adjunct Lecturer, CEID, University of Patras
& Computer Vision Team Leader, Metargus

March 26, 2021

About the speaker

- CEID Graduate: 2009-2014
- PhD in Machine Learning/Robotics (INRIA): 2015-2018
- Post-doc in Robotics (EPFL): 2018-2020
- CEID Adjunct Lecturer: Oct 2020-now
- Metargus: Jan 2021-now

Robotics (1)

Robotics (1)

Robotics (2)

[1] Boston Dynamics, 2020

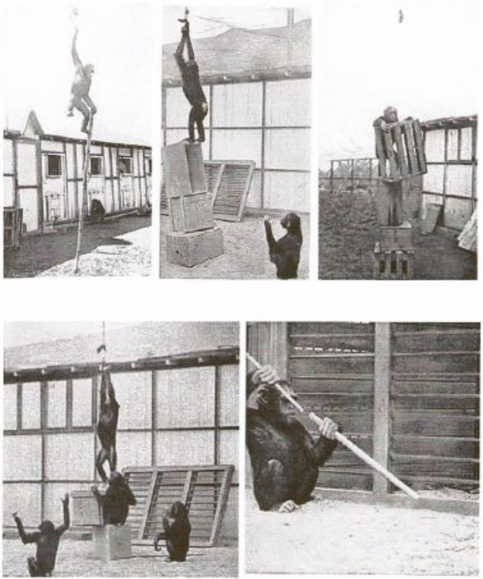
[2] `https://shorturl.at/oryAE`

Robotics - Limitations

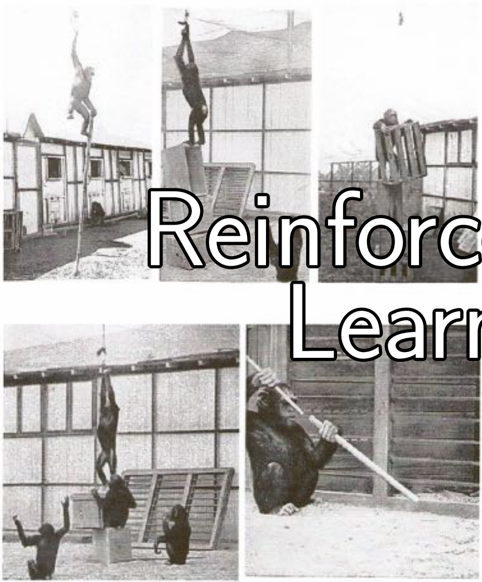
[1]DARPA Robotics Challenge, 2015

Animal Adaptation

Trial and Error Learning



Trial and Error Learning



Reinforcement Learning

Problem formulation

Reinforcement Learning (RL) for Robotics
We consider dynamical systems of the form:

$$\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w} \tag{1}$$

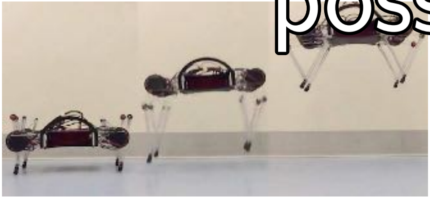
with $\boldsymbol{x} \in \mathbb{R}^E$, $\boldsymbol{u} \in \mathbb{R}^F$, i.i.d. Gaussian system noise $\boldsymbol{w}$, and unknown transition dynamics F.

We seek to find a *policy* π, $\boldsymbol{u} = \pi(\boldsymbol{x}|\boldsymbol{\theta})$, which maximizes the *expected long-term reward* in **as little interaction time as possible** (i.e., we want a **data-efficient** algorithm):

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} r(\boldsymbol{x}_t)\middle|\boldsymbol{\theta}\right] \tag{2}$$

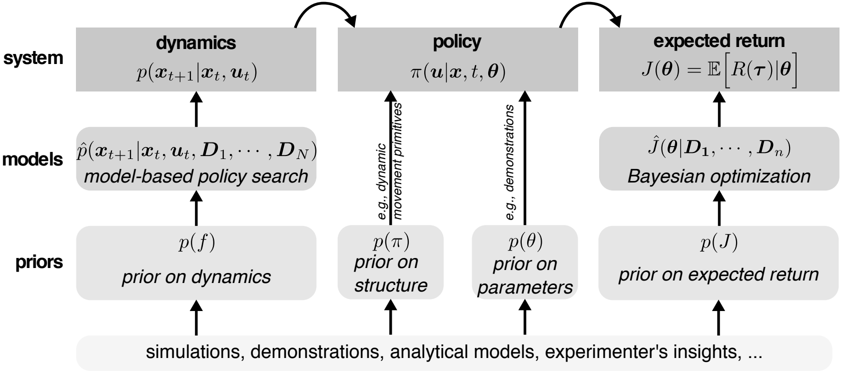
where $r(\boldsymbol{x}_t)$ is the immediate reward of being in state $\boldsymbol{x}$ at time t.

Micro-Data Reinforcement Learning



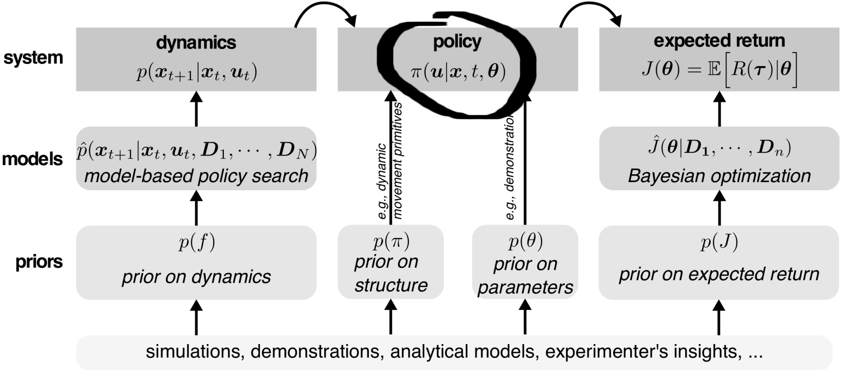
Only a few trials possible!

Strategies of Micro-Data Reinforcement Learning



[1] Chatzilygeroudis, K., Vassiliades, V., Stulp, F., Calinon, S., Mouret, J.-B. "A survey on policy search algorithms for learning robot controllers in a handful of trials", *IEEE Transactions on Robotics* 2019

Micro-Data Reinforcement Learning: No prior



State-of-the-art: Policy Gradient Algorithms

Policy search:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}} J(\boldsymbol{\theta}) \tag{3}$$

Stochastic Policy Gradients[1]:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\Big[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log\pi(\mathbf{u}_t|\mathbf{x}_t,\boldsymbol{\theta}) A_t\Big] \tag{4}$$

where $A_t = \hat{Q}^\pi(\mathbf{x}_t, \mathbf{u}_t)$.

Big variance in the gradient estimation, and thus slow convergence (or even divergence)!

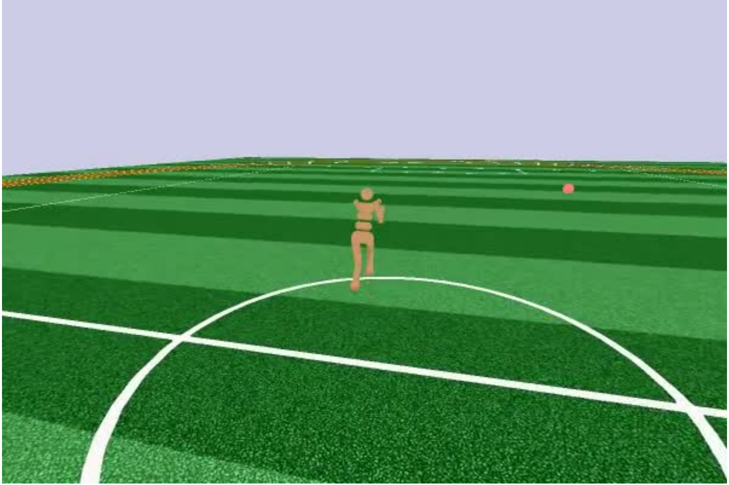
Trust Region Policy Optimization (TRPO)[2] and Proximal Policy Optimization (PPO)[3] use an extra constraint to reduce the variance and provide monotonic improvement guarantees.

[1] Sutton, R., et al. "Policy gradient methods for reinforcement learning with function approximation", NIPS, 2000

[2] Schulman, J., et al. "Trust region policy optimization", ICML, 2015

[3] Schulman, J., et al. "Proximal policy optimization algorithms", 2017

State-of-the-art: PPO Results

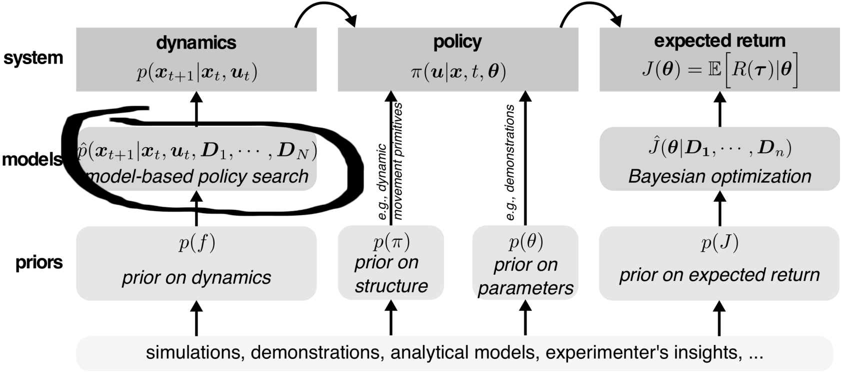


More than 80 hours of simulated training!

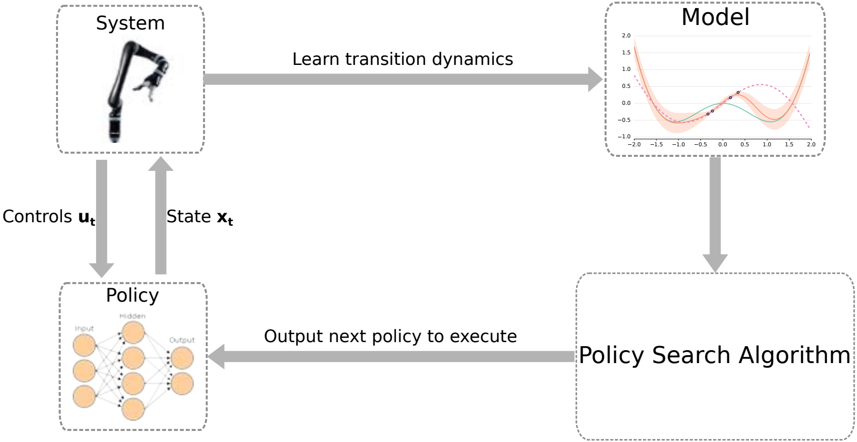
[1] Schulman, J., et al. "Proximal policy optimization algorithms", 2017

[2] https://blog.openai.com/openai-baselines-ppo/

Micro-Data Reinforcement Learning: Model-based

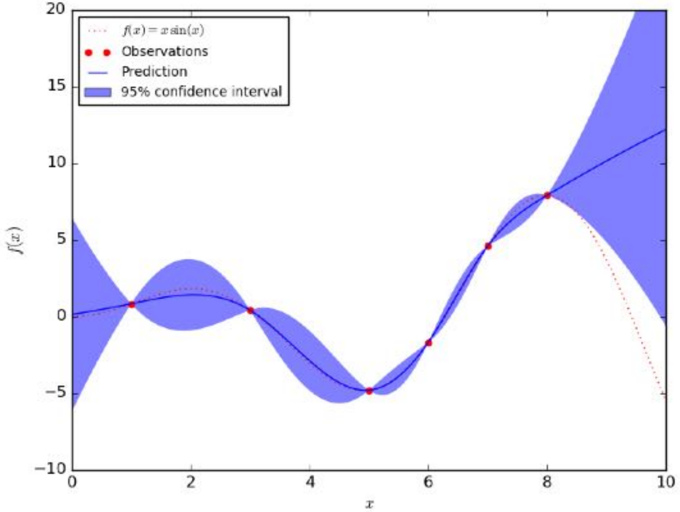


Model-based policy search



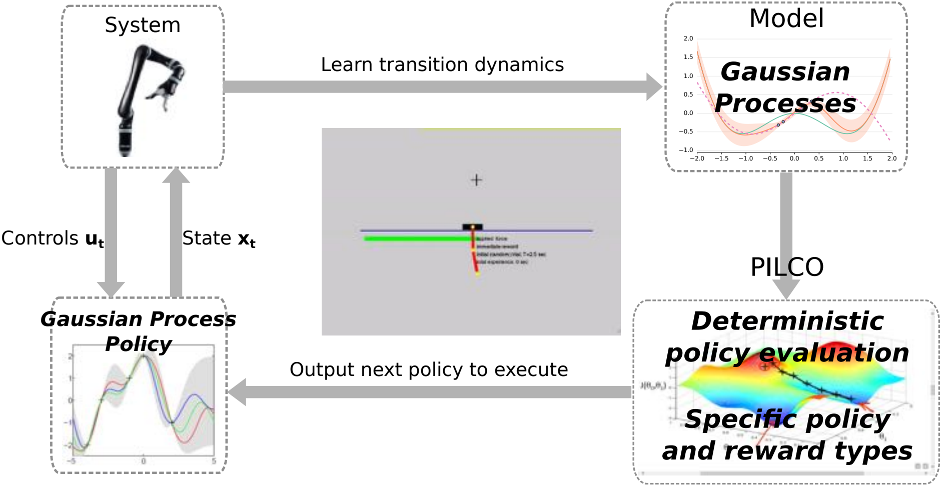
[1] Deisenroth, M.P., Neumann, G., Peters, J. "A survey on policy search for robotics", Foundations and Trends in Robotics, 2013

Gaussian Processes



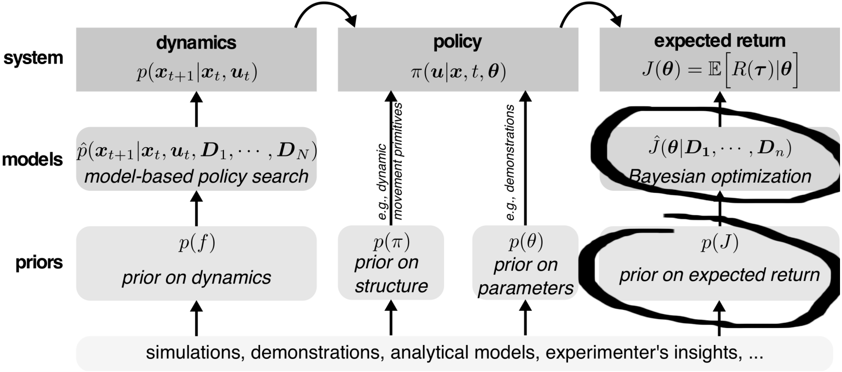
[1] Rasmussen, CE. "Gaussian processes in machine learning", 2004

State-of-the-art: PILCO[1]

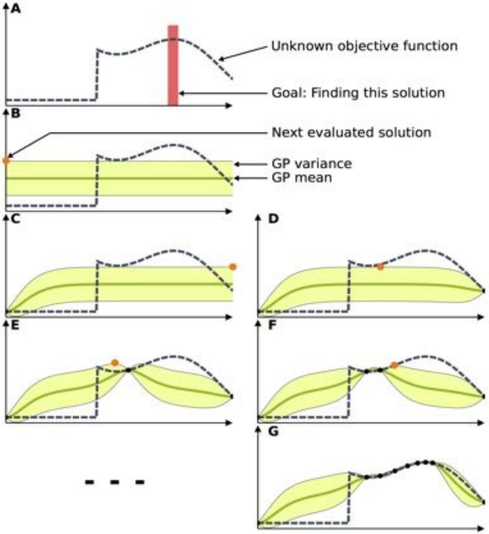


[1] Deisenroth, D., et al. "Gaussian Processes for Data-Efficient Learning in Robotics and Control" IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014

Micro-Data Reinforcement Learning: Surrogate Models



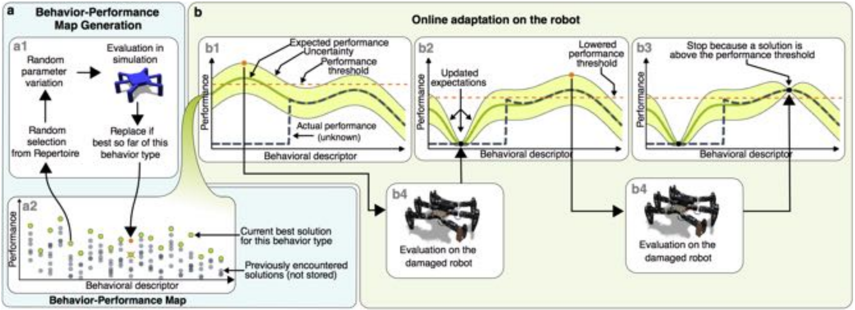
State-of-the-art: Bayesian Optimization

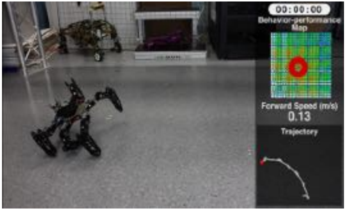


[1] Brochu, E., Cora, V.M., De Freitas, N. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning", 2010

State-of-the-art: Learning for Damage Recovery

Intelligent Trial and Error Algorithm (IT&E)[1]





[1] Cully, A. et al. "Robots that can adapt like animals", in Nature, vol. 521, no. 7553, pp. 503–507, 2015

State-of-the-art: Quadratic Programming-based Control

But we can have analytical models:

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \\ \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= \ddot{\boldsymbol{x}} \end{aligned} \quad (5)$$

And solve an optimization:

$$\begin{aligned} &\min_{\mathcal{X}} - \frac{1}{2}\mathcal{X}^T\boldsymbol{G}\mathcal{X} + \boldsymbol{g}^T\mathcal{X} \\ &\text{s.t. } \begin{bmatrix}\boldsymbol{M}(\boldsymbol{q}) & -\boldsymbol{S} & -\boldsymbol{J}(\boldsymbol{q})^T\end{bmatrix}\mathcal{X} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) = 0 \end{aligned} \quad (6)$$

where

$$\mathcal{X} = \begin{bmatrix}\ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T \quad (7)$$

State-of-the-art: QP-based Control (2)

[1]LARSEN Inria: `https://www.youtube.com/watch?v=-An7Ju3ge0I&ab_channel=LarsenInria`

State-of-the-art: Limitations

Learning Methods

Pure episodic approach:

- The robot starts in the same initial state at each episode;
- Learning process separated from operation.

Scaling issues with complex robots:

- Exponentially more data are needed as the dimensionality of state/action space increases;
- Data-efficient approaches do not take advantage of multi-core architectures.

Traditional Methods

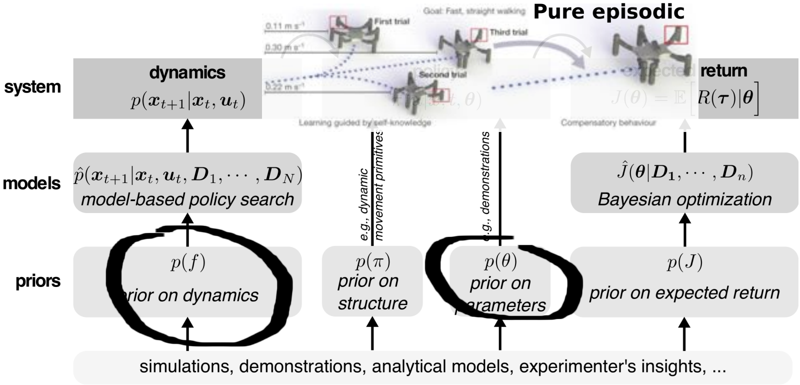
Require a lot of tuning:

- Hard to find hyper-parameters;
- Different parameters for each task.

Accurate models required:

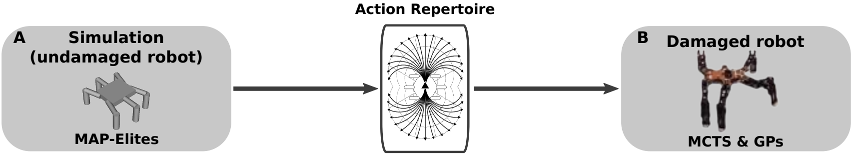
- Model-based methods (e.g., QP-control) do not work with in accurate models;
- Difficult to incorporate learning.

Reset-free Trial and Error Learning for Robot Damage Recovery



[1] Chatzilygeroudis, K., Vassiliades, V. and Mouret, J. B. "Reset-free Trial-and-Error Learning for Robot Damage Recovery", RAS, 2018.

Reset-free Trial and Error Learning for Robot Damage Recovery (RTE)



Learning and correcting the repertoire

MAP-Elites[1] uses a **simulated intact** robot and **requires**:

- A parameterized policy, $\pi_{\boldsymbol{\theta}}$
- An action descriptor, $\boldsymbol{a}$, that describes the task space
- A performance measure, $\mathrm{performance}(\boldsymbol{\theta})$

It provides:

- A diverse set of locally (with respect to $\boldsymbol{a}$) optimized policies, A
- A mapping between the task space, (or the set of actions A), and the policy space Θ; i.e., $A \rightarrow \Theta$
- A mapping between actions and relative outcomes, $M : A \rightarrow O$;

We use Gaussian Processes with a non-zero mean function to correct the repertoire:

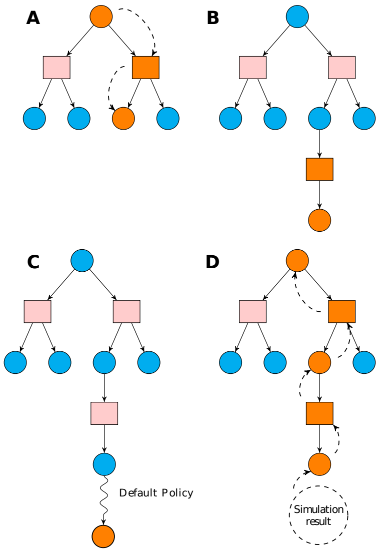
$$
\begin{aligned}
&p(f(\boldsymbol{a})|D_{1:t}, \boldsymbol{a}) \sim \mathcal{N}(\mu(\boldsymbol{a}), \sigma^2(\boldsymbol{a})) \\
&\mu(\boldsymbol{a}) = M(\boldsymbol{a}) + \boldsymbol{k}^T K^{-1}(D_{1:t} - M(\boldsymbol{a}_{1:t})) \\
&\sigma^2(\boldsymbol{a}) = k(\boldsymbol{a}, \boldsymbol{a}) - \boldsymbol{k}^T K^{-1}\boldsymbol{k}
\end{aligned} \tag{8}
$$

[1] Mouret, J.-B., and Clune, J. "Illuminating search spaces by mapping elites.", 2015.

Planning with MCTS

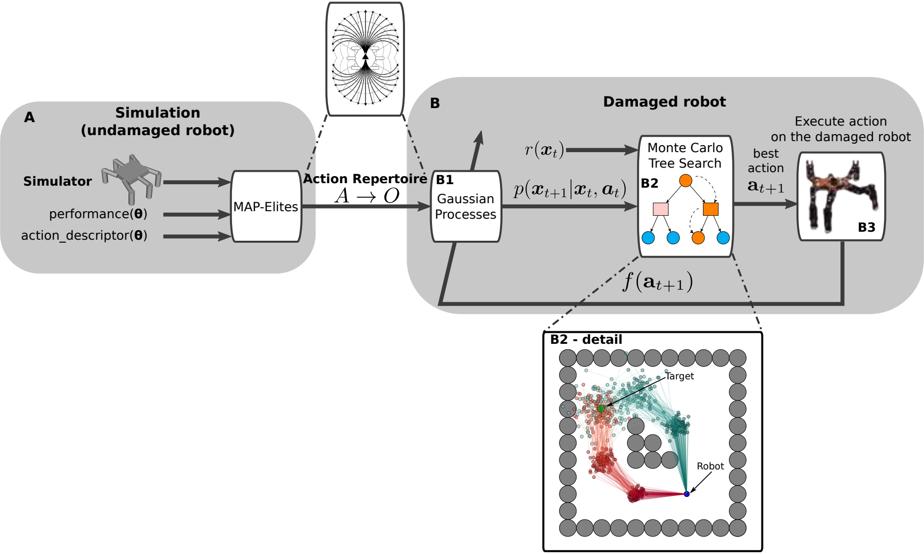
Monte Carlo Tree Search:

- is a planning algorithm that is:
  - best-first, sample-based, and
  - anytime
- finds optimal decisions
  - by taking random samples, and
  - building a search tree according to their results
- treats as a **black-box** the model of the environment (handles uncertainty)
- has successfully solved RL problems with:
  - stochastic transitions,
  - continuous state spaces, and
  - high branching factors



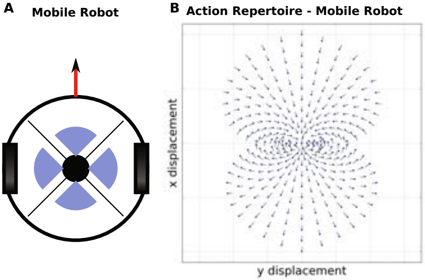
[1] https://github.com/resibots/mcts

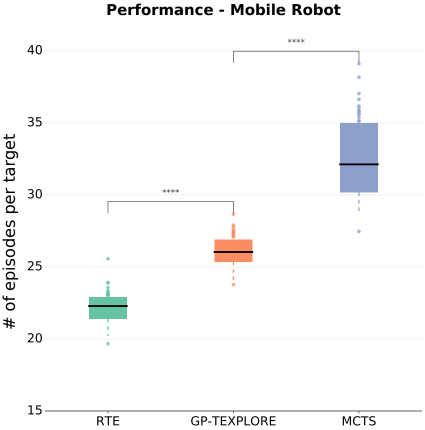
RTE Overview



Results: Mobile robot

A velocity-controlled differential drive robot has to learn to navigate again after one of the motors only achieves half of the desired velocity.

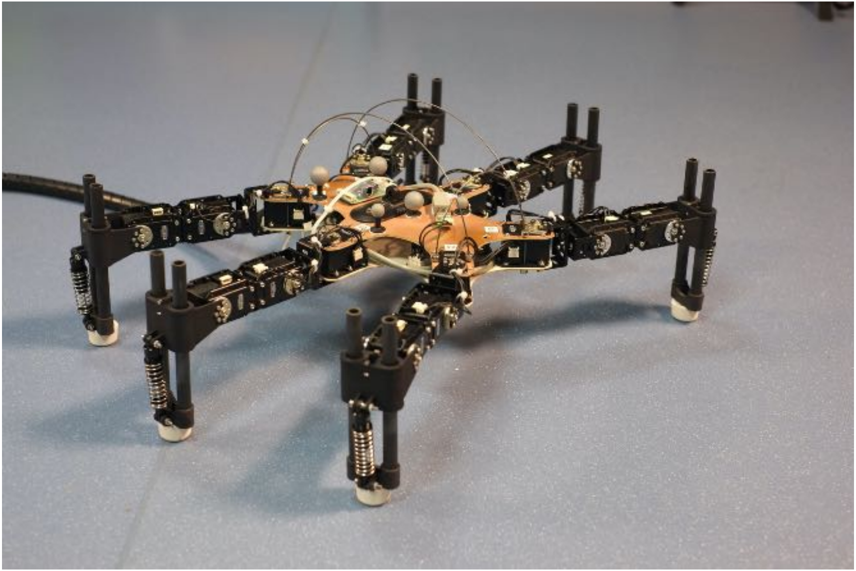




50 replicates

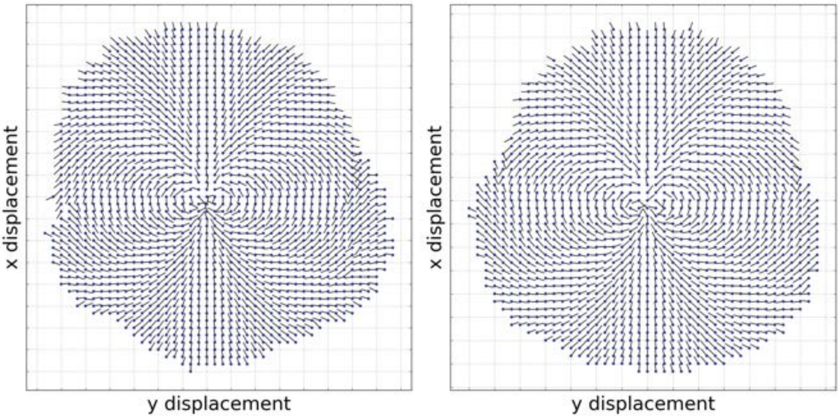
[1] Hester, T., Stone, P. "TEXPLORE: real-time sample-efficient reinforcement learning for robots", Machine learning, 2013

Simulated hexapod robot



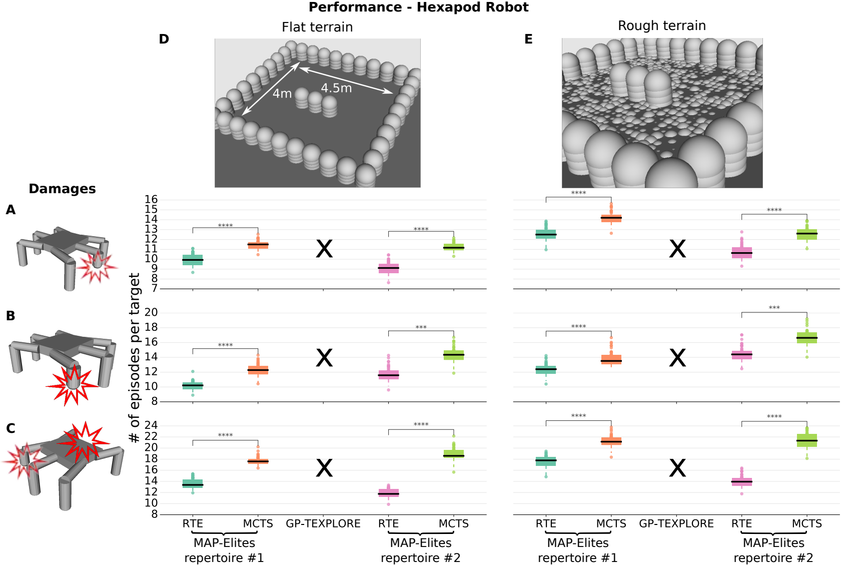
Results - Simulated hexapod robot

Action Repertoires - Hexapod Robot



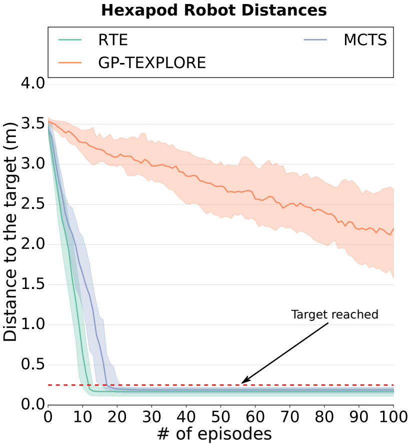
Action Repertoire #1 Action Repertoire #2

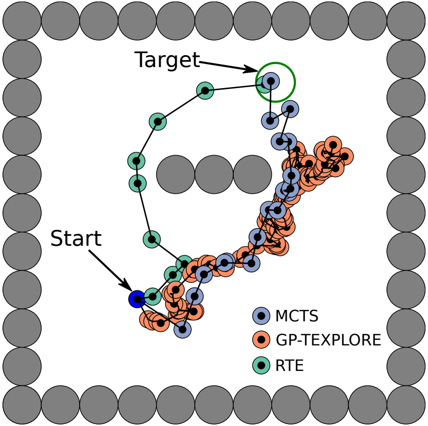
Results - Simulated hexapod robot (2)



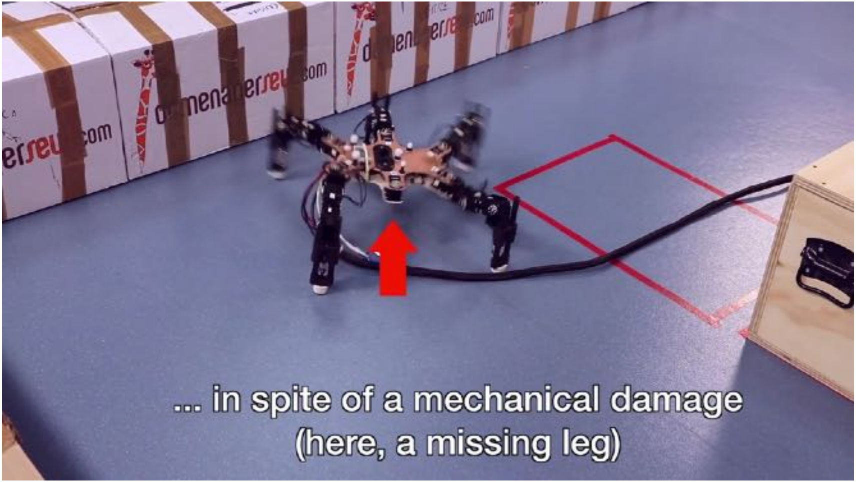
50 replicates

Results: Simulated hexapod robot (3)

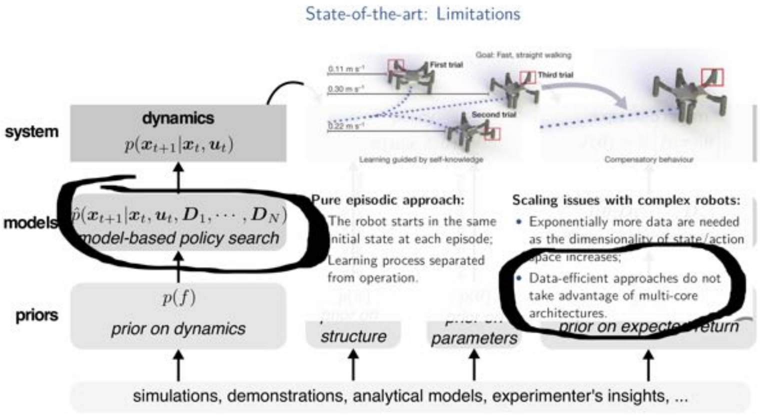




Video - Physical hexapod robot

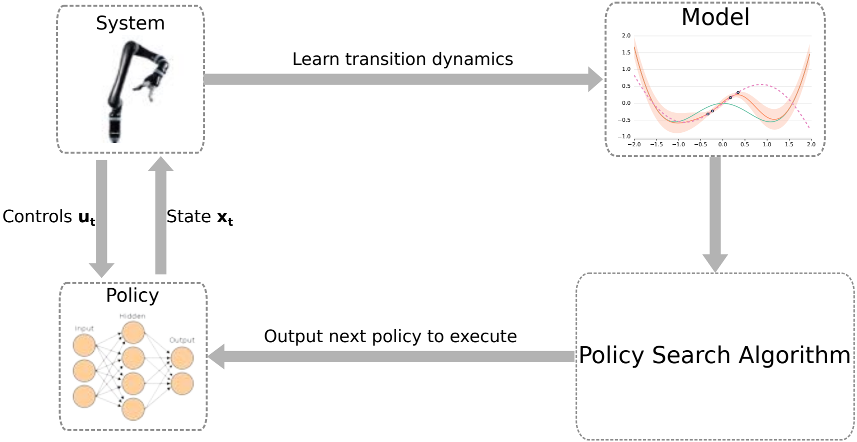


Black-Box Data-efficient Policy Search for Robotics



[1] Chatzilygeroudis, K., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., and Mouret, J. B. "Black-Box Data-efficient Policy Search for Robotics", IROS, 2017

Model-based policy search - Reminder



[1] Deisenroth, M.P., Neumann, G., Peters, J. "A survey on policy search for robotics", Foundations and Trends in Robotics, 2013

Black-box Data-efficient RObot Policy Search (Black-DROPS)

Key Idea #1 *Implicit Policy Evaluation*; treat each rollout as a noisy measurement of the *expected long-term reward*:

$$G(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + N(\boldsymbol{\theta}) \tag{9}$$

$$\begin{aligned}\mathbb{E}\Big[G(\boldsymbol{\theta})\Big] &= \mathbb{E}\Big[J(\boldsymbol{\theta}) + N(\boldsymbol{\theta})\Big] = \mathbb{E}\Big[J(\boldsymbol{\theta})\Big] + \mathbb{E}\Big[N(\boldsymbol{\theta})\Big] \\ &= J(\boldsymbol{\theta}) + \mathbb{E}\Big[N(\boldsymbol{\theta})\Big] (\text{since } \mathbb{E}\Big[\mathbb{E}[x]\Big] = \mathbb{E}[x])\end{aligned} \tag{10}$$

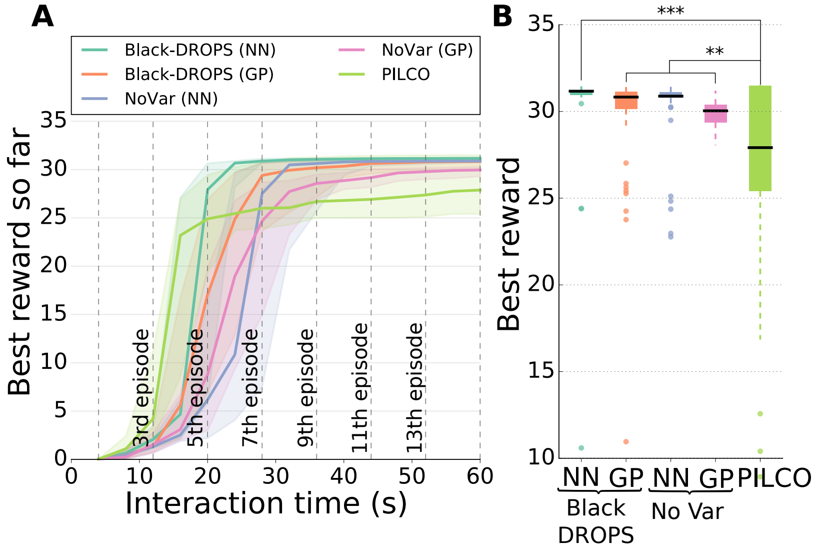
Key Idea #2 Use a black-box optimizer that:

- Is suited for noisy optimization;
- Can take advantage of multi-core computers.

We use a modified version of IPOP-CMA-ES[1] that fulfills the above properties.

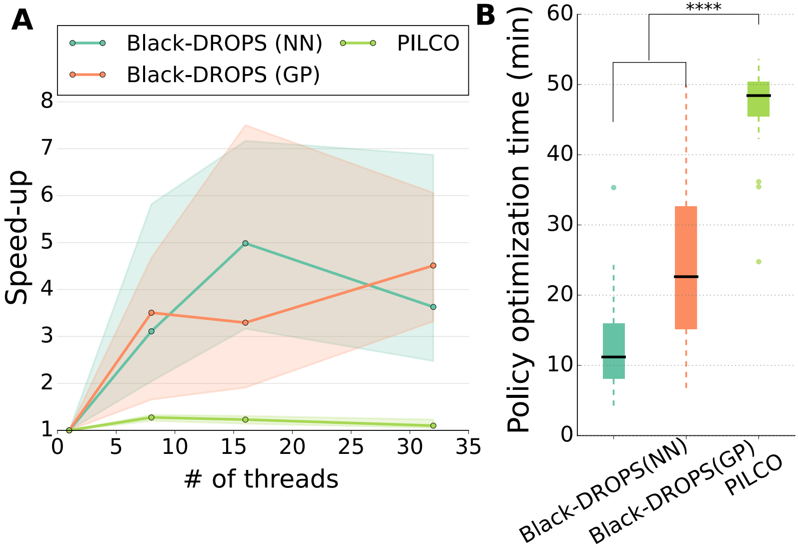
[1] Hansen, N., Ostermeier, A. "Completely derandomized self-adaptation in evolution strategies", Evolutionary Computation, 2001

Results - Noiseless Cartpole Simulation



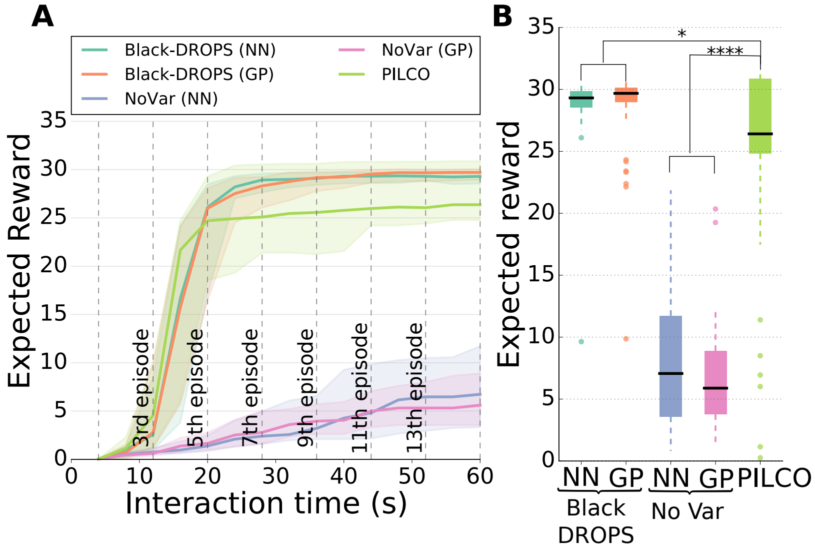
80 replicates

Results - Cartpole Simulation (Scaling)



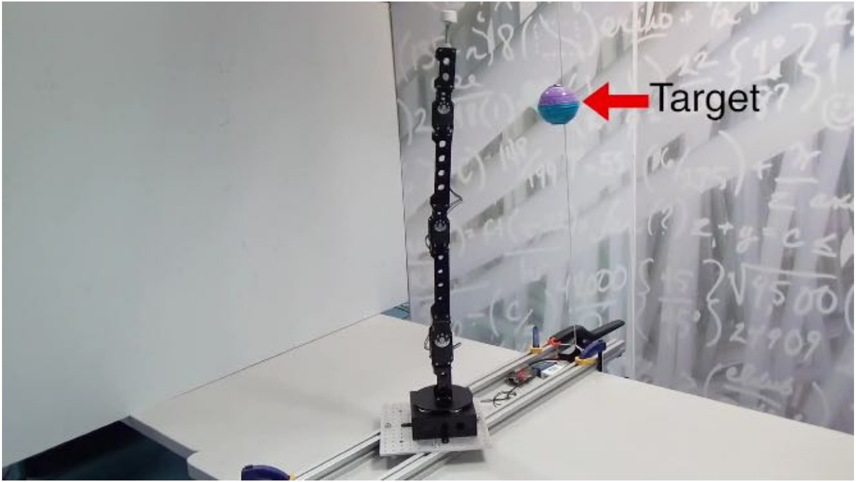
80 replicates

Results - Noisy Cartpole Simulation

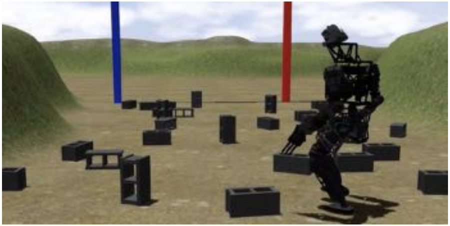


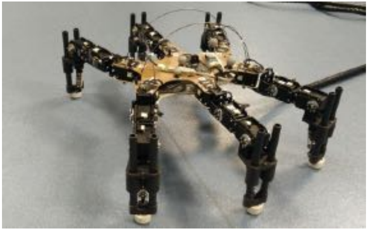
80 replicates

Video - Physical Robot Experiments

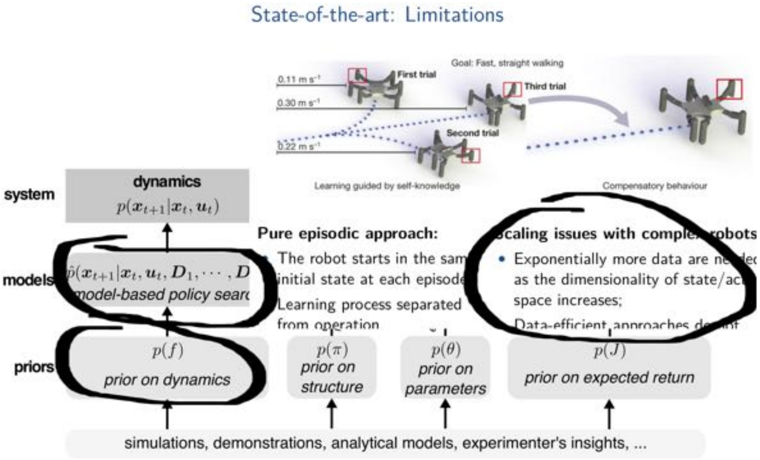


But model-based policy search does not scale!!





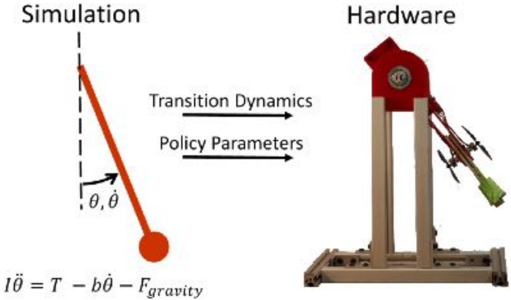
Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics



[1] Chatzilygeroudis, K., and Mouret, J.-B. “Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics”, ICRA, 2018.

Priors to the rescue!

We can use dynamic simulators as priors for the dynamics model.



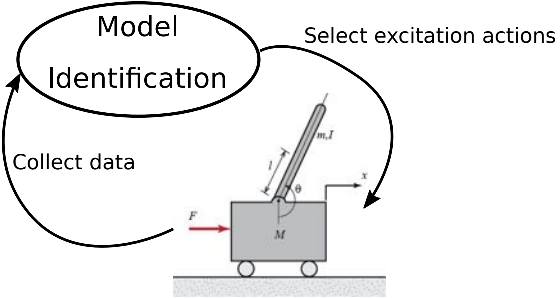
[1] Cutler, M., and How, J. P. "Efficient reinforcement learning for robots using informative simulated priors." ICRA, 2015.

[2] Saveriano, M., Yin, Y., Falco, P., and Lee, D. "Data-Efficient Control Policy Search using Residual Dynamics Learning.", IROS, 2017.

Model Identification

When equations are available, identifying the parameters from data would be the classic approach, but:

- Assumes the actual system can be fully captured by the available equations;
- Proper excitation of the system required for non-trivial cases.



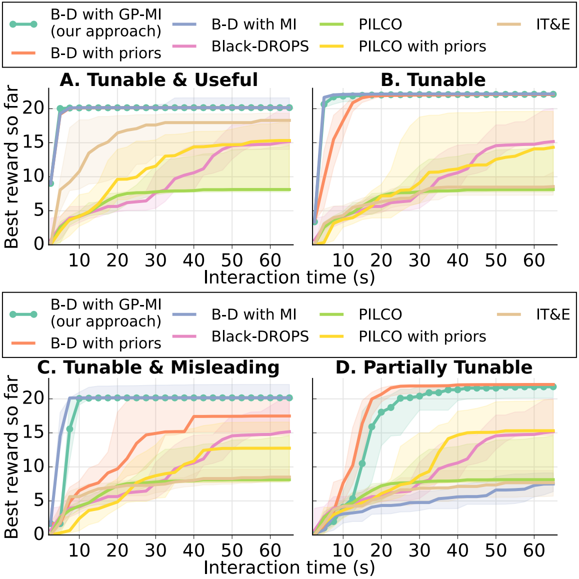
Gaussian Processes with Parameterized Black-Box Priors

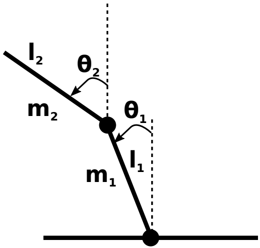
- Let's re-write the equation of our dynamic system:

$$\boldsymbol{x}_{t+1} = \underbrace{M(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\phi}_M) + f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\phi}_K)}_{F(\boldsymbol{x}_t, \boldsymbol{u}_t)} + \boldsymbol{w} \tag{11}$$

- Each $\boldsymbol{\phi}_M$ corresponds to a different parameterization of the mean model M.
- We use GPs to model F and $M(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\phi}_M)$ is the mean function of the GPs; $f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\phi}_K)$ captures what cannot be captured by the mean function.
- Now we **can jointly optimize** for $\boldsymbol{\phi}_M$ and $\boldsymbol{\phi}_K$ through Maximum Likelihood Estimation (MLE).
- The modeling procedure can balance between non-parametric modeling and model identification (we call it **GP-MI**).

Results - Pendubot Simulation



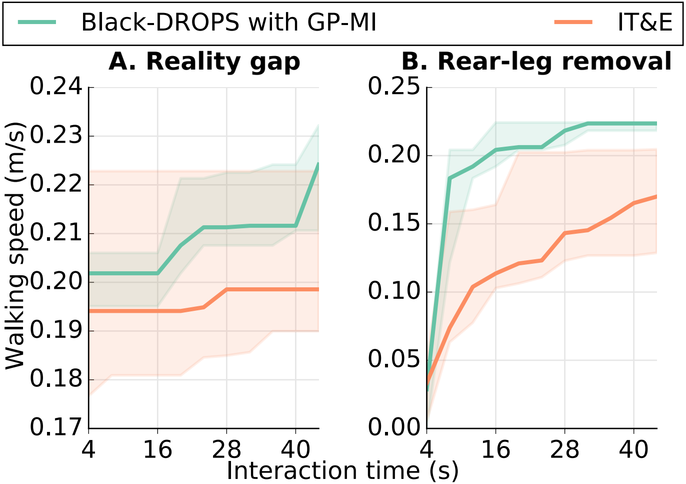


Pendubot system

30 replicates

Results - Physical Hexapod Robot Experiments

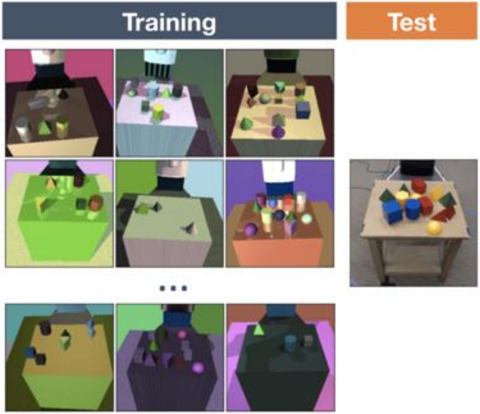
Real Hexapod Robot Experiments



5 replicates

[1] Cully, A. et al. "Robots that can adapt like animals", in Nature, vol. 521, no. 7553, pp. 503–507, 2015

Sim2Real Methods



[1] Tobin, J. et al. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World", IROS, 2017

Sim2Real Methods

Main intuitions:

- Simulators give us the ground truth data;
- Simulators can be fast;
- Simulators are reproducible;
- Simulators allow easy customization.

Main steps:

Domain Randomization (DR)

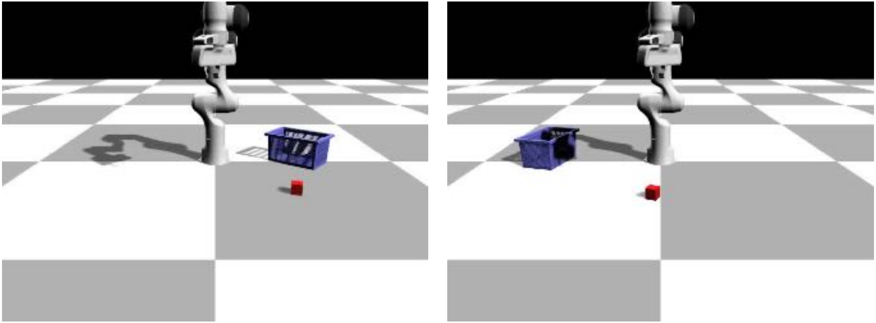
- For each episode, select a random initialization of the world dynamics;
- Optimize the policy for a few steps (e.g., via RL);
- Create a new episode.

Imitation Learning (IL)

- Solve/Learn the task in simulation;
- Collect many many variations of the solution;
- Learn the policy via supervised learning.

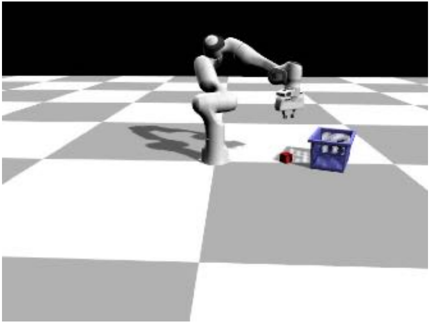
DR + IL gives best results...

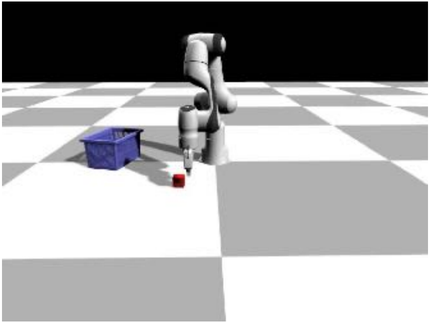
Imitation Learning using a Simulator



[1] Konstantinos Tsinganos, CEID

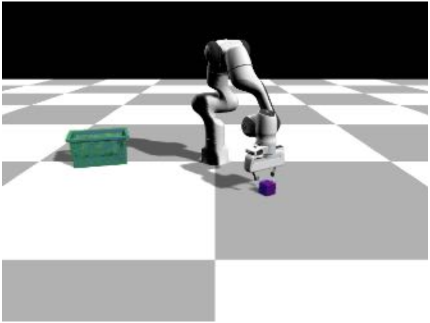
Imitation Learning using a Simulator (2)

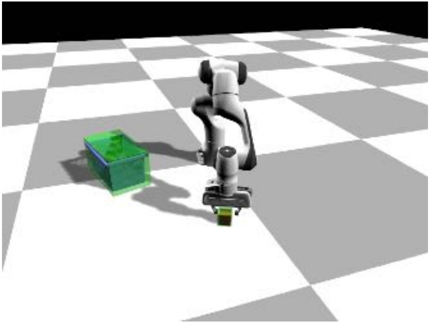




[1] Konstantinos Tsinganos, CEID

Visual/Realistic Sim2Real





[1] Konstantinos Tsinganos, CEID

Combining Learning with Traditional Methods & Exploration

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ UNIVERSITY OF PATRAS

LASA

EPFL

But we can have analytical models:

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_{\boldsymbol{g}}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \\ \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= \ddot{\boldsymbol{x}} \end{aligned} \tag{12}$$

And solve an optimization:

$$\begin{aligned} &\min_{\mathcal{X}} -\frac{1}{2}\mathcal{X}^T\boldsymbol{G}\mathcal{X} + \boldsymbol{g}^T\mathcal{X} \\ &\text{s.t. } \begin{bmatrix}\boldsymbol{M}(\boldsymbol{q}) & -\boldsymbol{S} & -\boldsymbol{J}(\boldsymbol{q})^T\end{bmatrix}\mathcal{X} + \boldsymbol{C}_{\boldsymbol{g}}(\boldsymbol{q},\dot{\boldsymbol{q}}) = 0 \end{aligned} \tag{13}$$

where

$$\mathcal{X} = \begin{bmatrix}\ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T \tag{14}$$

Combining QP-control with Learning

Inverse Dynamics:

$$\begin{aligned} \boldsymbol{S}\boldsymbol{\tau} &= \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \\ \boldsymbol{\tau} &= \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \end{aligned} \tag{15}$$

Learning Inverse Dynamics Models:

$$\boldsymbol{\tau} = f_{\text{joint}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}_{\text{desired}}) \tag{16}$$

$$f_{\text{joint}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}_{\text{desired}}) = f_{\text{analytic}} + e_{\text{joint}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}_{\text{desired}}) \tag{17}$$

But e_{joint} needs to be linear wrt $\mathcal{X} = \begin{bmatrix}\ddot{\boldsymbol{q}}_{\text{desired}} & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T$!
Let's linearize: We are in a state $(\boldsymbol{q}_t,\dot{\boldsymbol{q}}_t,\ddot{\boldsymbol{q}}_t)$ and we take the first two terms of the Taylor series expansion of $e_{\text{joint}}(\cdot,\cdot,\cdot)$:

$$\begin{aligned} f_{\text{learned}}(\boldsymbol{q}_t,\dot{\boldsymbol{q}}_t,\ddot{\boldsymbol{q}}_{t+1}) &= f_{\text{analytic}}(\boldsymbol{q}_t,\dot{\boldsymbol{q}}_t,\ddot{\boldsymbol{q}}_{t+1}) + e_{\text{joint}}(\boldsymbol{q}_t,\dot{\boldsymbol{q}}_t,\ddot{\boldsymbol{q}}_t) \\ &+ (\ddot{\boldsymbol{q}}_{t+1} - \ddot{\boldsymbol{q}}_t)\frac{\partial e_{\text{joint}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})}{\partial \ddot{\boldsymbol{q}}}\Big|_{\substack{\boldsymbol{q}=\boldsymbol{q}_t \\ \dot{\boldsymbol{q}}=\dot{\boldsymbol{q}}_t \\ \ddot{\boldsymbol{q}}=\ddot{\boldsymbol{q}}_t}} \end{aligned} \tag{18}$$

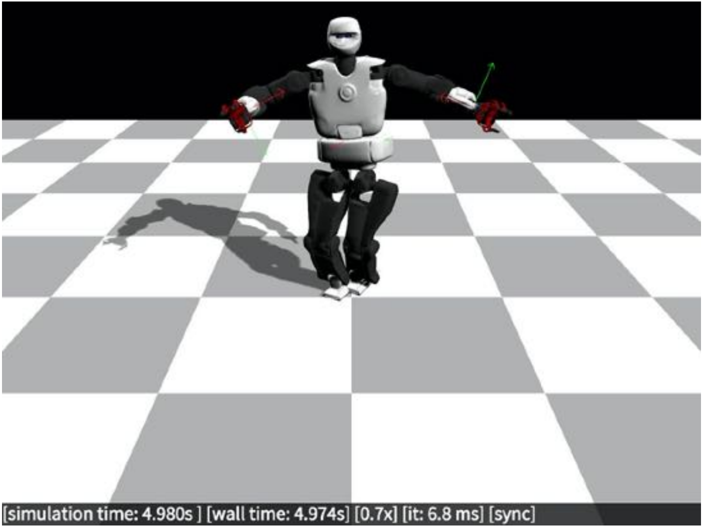
Combining QP-control with Learning (2)

Algorithm Self-correcting QP

Design the task specifications: $\boldsymbol{x}_d^*(t)$
Configure the adaptive controller and the model learning procedure
for $n = 1 \rightarrow N_{\text{episodes}}$ **do** ▷ For each episode
    for $t = 0 \rightarrow T$ **do** ▷ For each time-step
        Get $\ddot{\boldsymbol{x}}_r^*$ from adaptive controller
        Compute the cost function for the QP given the $\ddot{\boldsymbol{x}}_r^*$
        Get $\boldsymbol{\tau}$ from the QP with the updated cost function, and the learned model $f_{\text{learned}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}}_{\boldsymbol{r}}^*)$
        Apply $\boldsymbol{\tau}$ to the robot and collect data
        Update the adaptive controller
    Inverse dynamics model learning

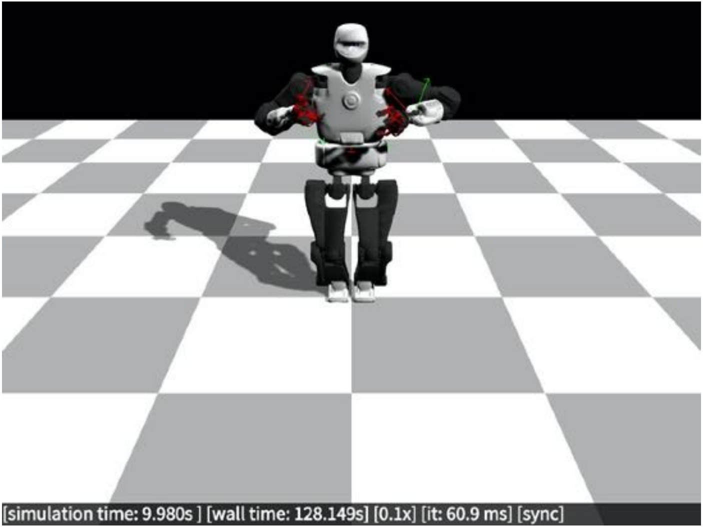
Combining QP-control with Learning (3)

Before Learning



Combining QP-control with Learning (4)

After Learning

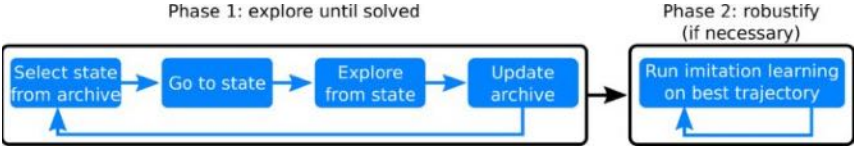


Combining QP-control with Learning (5)

Learning on a physical robot

Discovering interesting behaviors using exploration

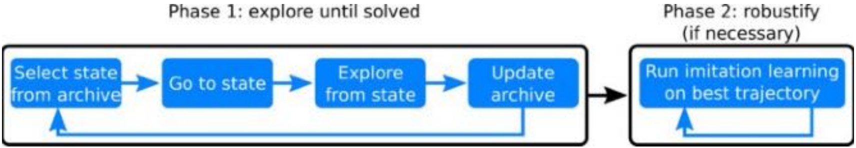
Go-Explore: a New Approach for Hard-Exploration Problems



[1] Ecoffet, A. et al. "First return, then explore", Nature, 2021

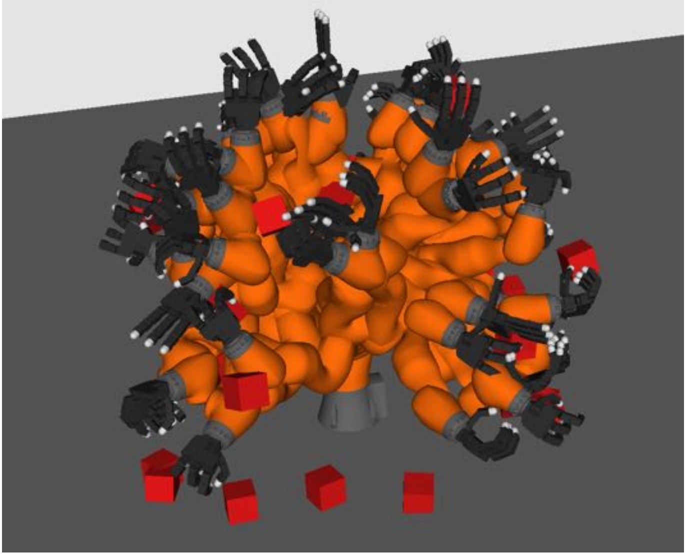
Discovering interesting behaviors using exploration

Go-Explore: a New Approach for Hard-Exploration Problems

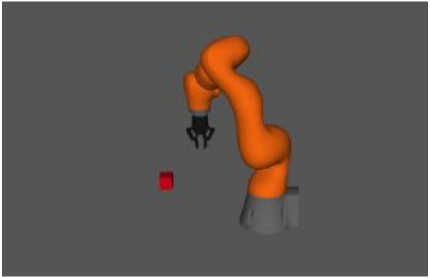


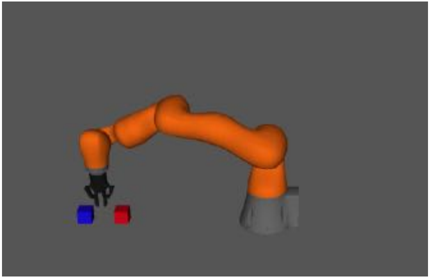
[1]Ecoffet, A. et al. "First return, then explore", Nature, 2021

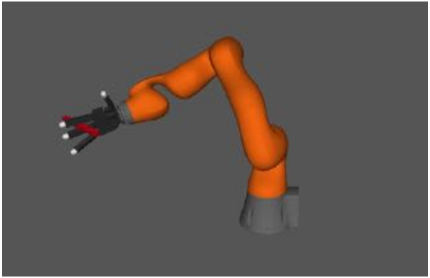
Discovering interesting behaviors using exploration (2)



Discovering interesting behaviors using exploration (3)







[1] Andrea Mussati, EPFL

Automated Sports Player Tracking and Statistics

Thank you!
Any questions?

costashatz@upatras.gr

http://costashatz.github.io