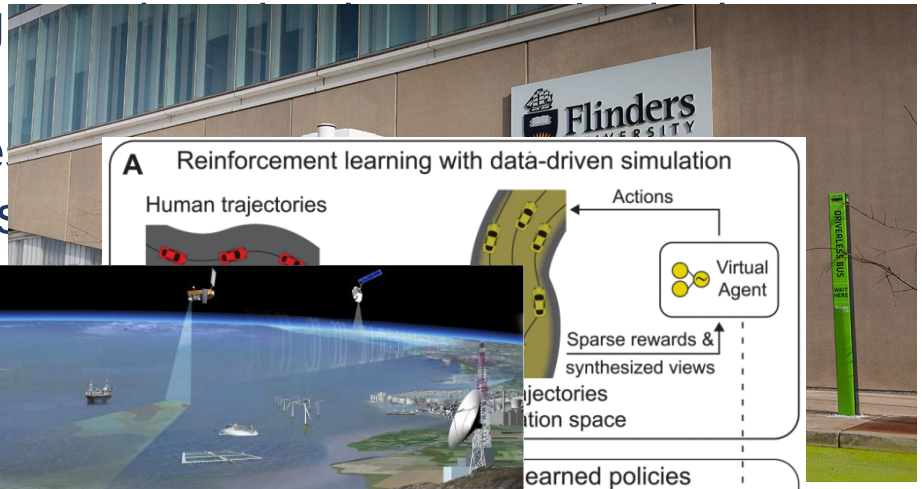
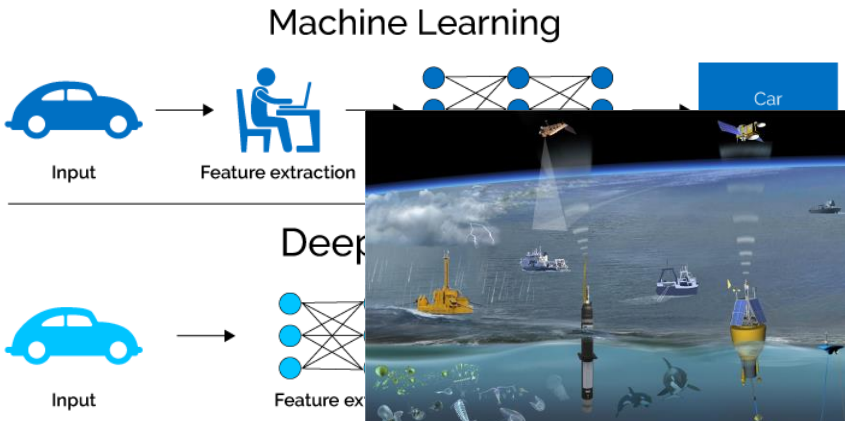
Thomas Chaffre

Reinforcement Learning and Sim-to-Real Transfer for Adaptive Control of AUV

Ambitions

1. **Autonomous vehicles** are becoming increasingly prevalent in our day-to-day activities with strengthen productivity, accuracy, operational efficiency and improved safety for human operators' and users'.





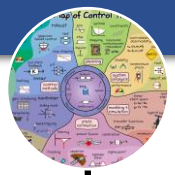
A big problem: autonomy in AUVs

Maritime robotic student projects at ENSTA Bretagne
Source: https://guerledan.ensta-bretagne.fr/actualites

Principal challenges

- Unknown dynamics
- Nonlinearities

Research spectrum



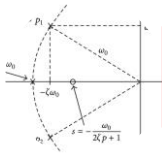
PhD Thesis

ROS package

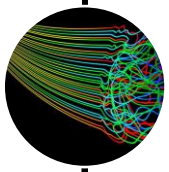
Python/C++ (GitHub)

Learning-based Adaptive Control

Multiple vehicles (ECA A9, BlueRov 2, RexRov)



Gains-Poles Mapping

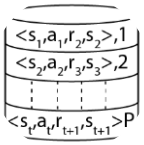


Maximum Entropy RL

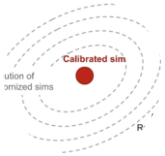
7 peer-reviewed papers

Multiple tasks (target rallying, station keeping, output regularization)

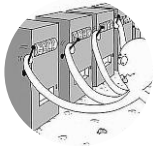
Incremental VS Direct placement



Experience Replay

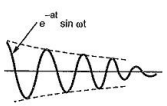


Sim-to-real Transfer



Exploration in RL

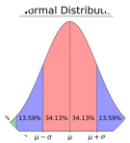
Multiple algorithms (DDPG, SAC-v1, SAC-v2, TD3)



DAPP

Bio-Inspired Experience Replay

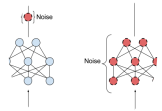
Domain Randomization



Passive exploration

Multiple domains (terrestrial, aerial, underwater)

Randomized Environmental Complexity



Parameter Noise

Historical solution: adaptive control

NASA research pilot Neil Armstrong following a mission in the first X-15 rocket plane. Image via NASA Dryden Flight Research Center.

Adaptive Control and the NASA X-15-3 Flight Revisited, by Z. T. Dydek et al., in IEEE Control systems, 2010.

"To adapt is to change something in order to make it suitable for a new use or situation."

Oxford English Dictionary

Significant barriers in AUVs

- Things we do not know how to model
- Things me can not measure
- Complexity of simulations

AGENDA (thesis overview)

Background

- Literature Review and Theory

Empirical works

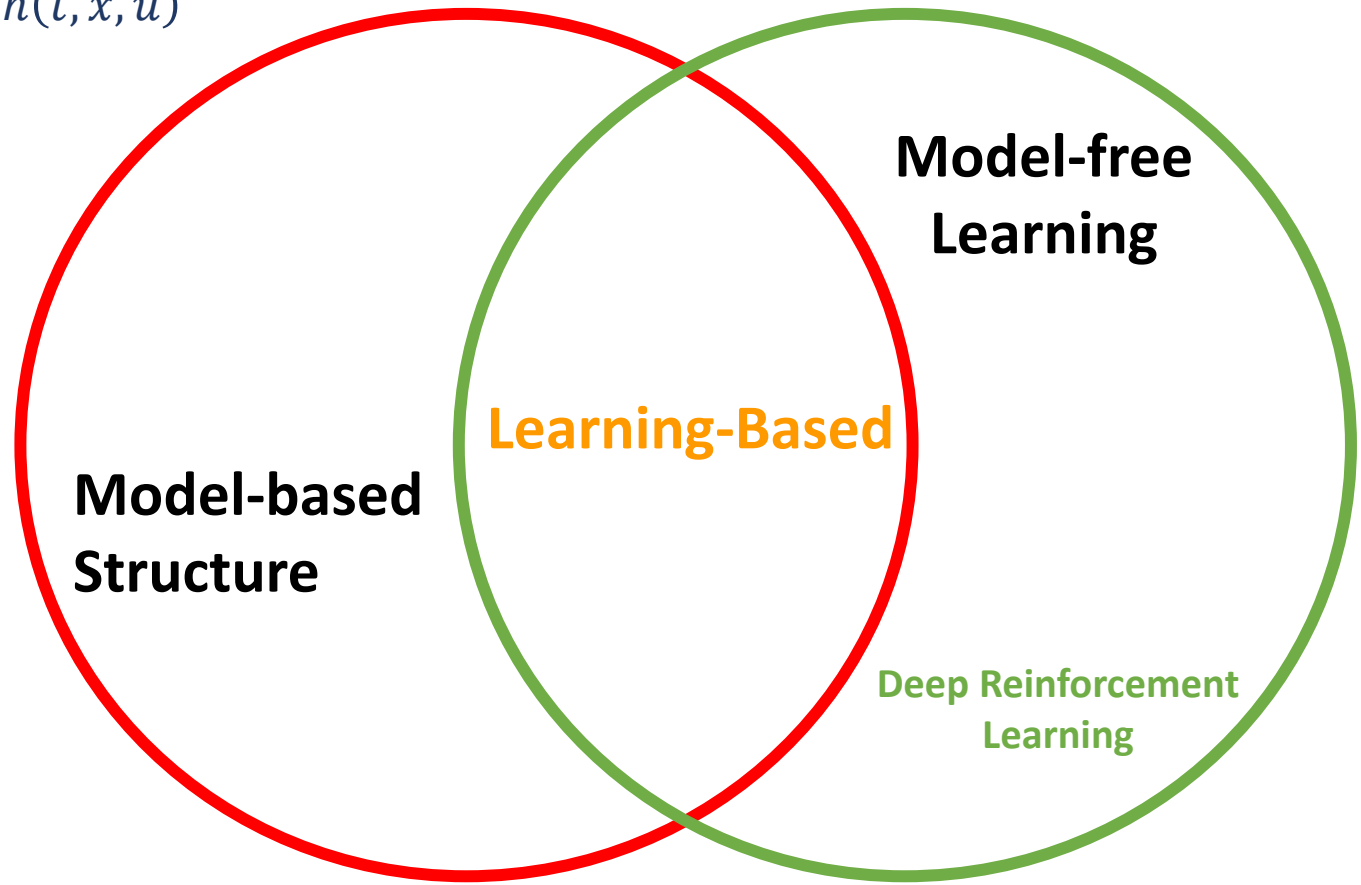
- Merging RL and Adaptive Control
- Maximum Entropy RL
- Taking inspiration from biological replay mechanism
- Sim-to-real Transfer

Conclusion

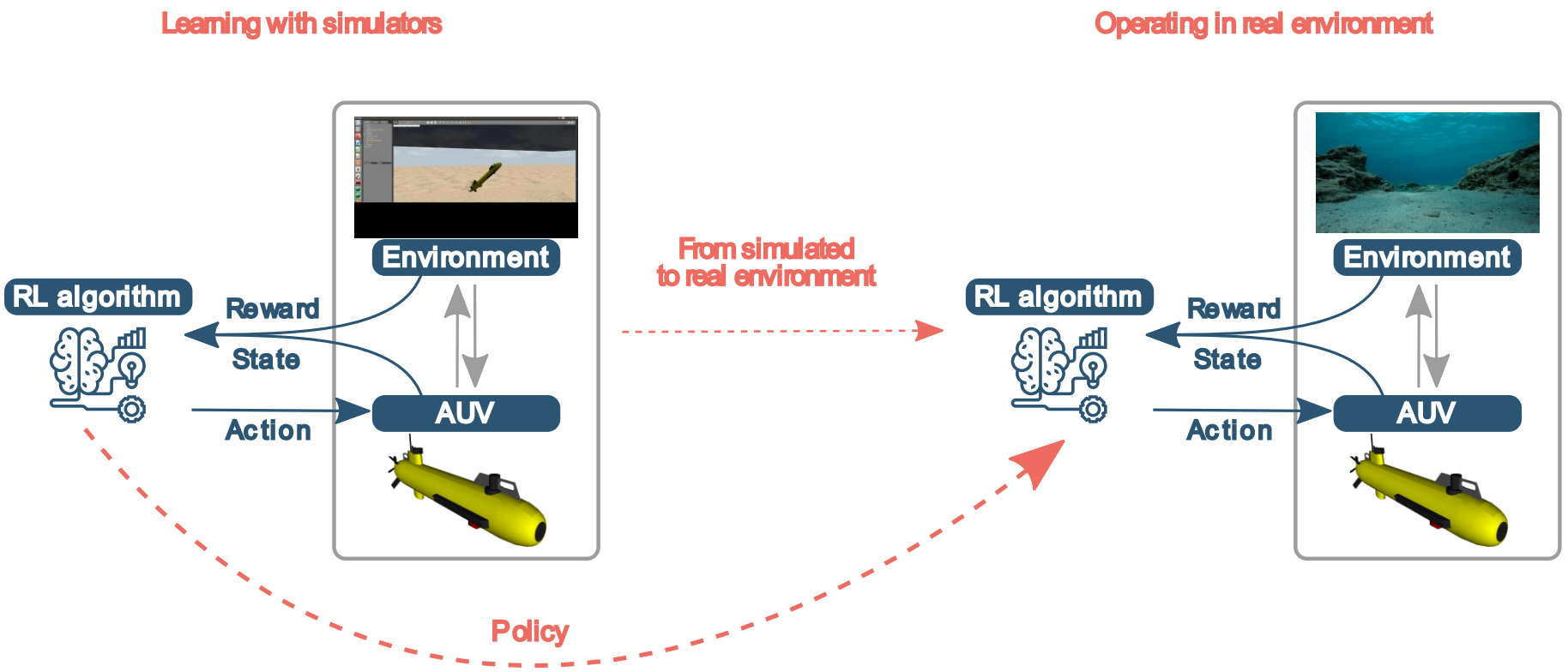
- Current and Future Research

A potential solution: learning-based

$$\begin{cases} \dot{x}(t) = f_1(t, x, u) + f_2(t, x, p) \\ y(t) = h(t, x, u) \end{cases}$$



Overall ambition



Related works

Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle, by R. Yu et al., in 36th Chinese Control Conference, 2017.

Reinforcement learning-based adaptive trajectory planning for AUVs in under-ice environments, by C. Wang et al., in OCEANS MTS/IEEE Charleston, 2018.

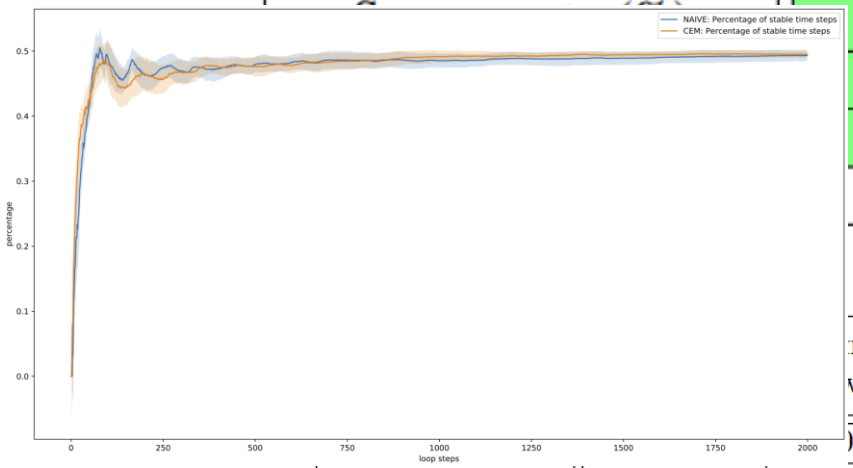
Deep learning for station keeping of AUVs, by K. B. Knudsen et al., in OCEANS 2019 MTS/IEEE SEATTLE, 2019.

Motion control of unmanned underwater vehicles via deep imitation reinforcement learning algorithm, by Z. Chu et al., in IET Intelligent Transport Systems, 2020.

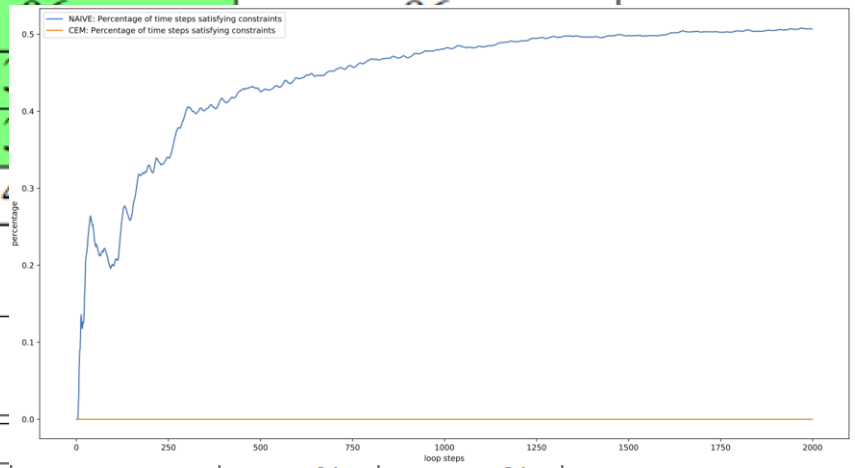
Preliminary works

PID Tuning using Cross-Entropy Deep Learning: a Lyapunov Stability Analysis, by H. Kohler, **T. Chaffre** et al., in Conference on Control Applications in Marine Systems, Robotics and Vehicles, 2022.

$$V(x) = \frac{1}{2} x^T \begin{bmatrix} M_\eta^{-1} & \alpha I & 0 \\ \alpha I & k_p & k_i \\ 0 & K_i & \alpha k_i \end{bmatrix} x,$$

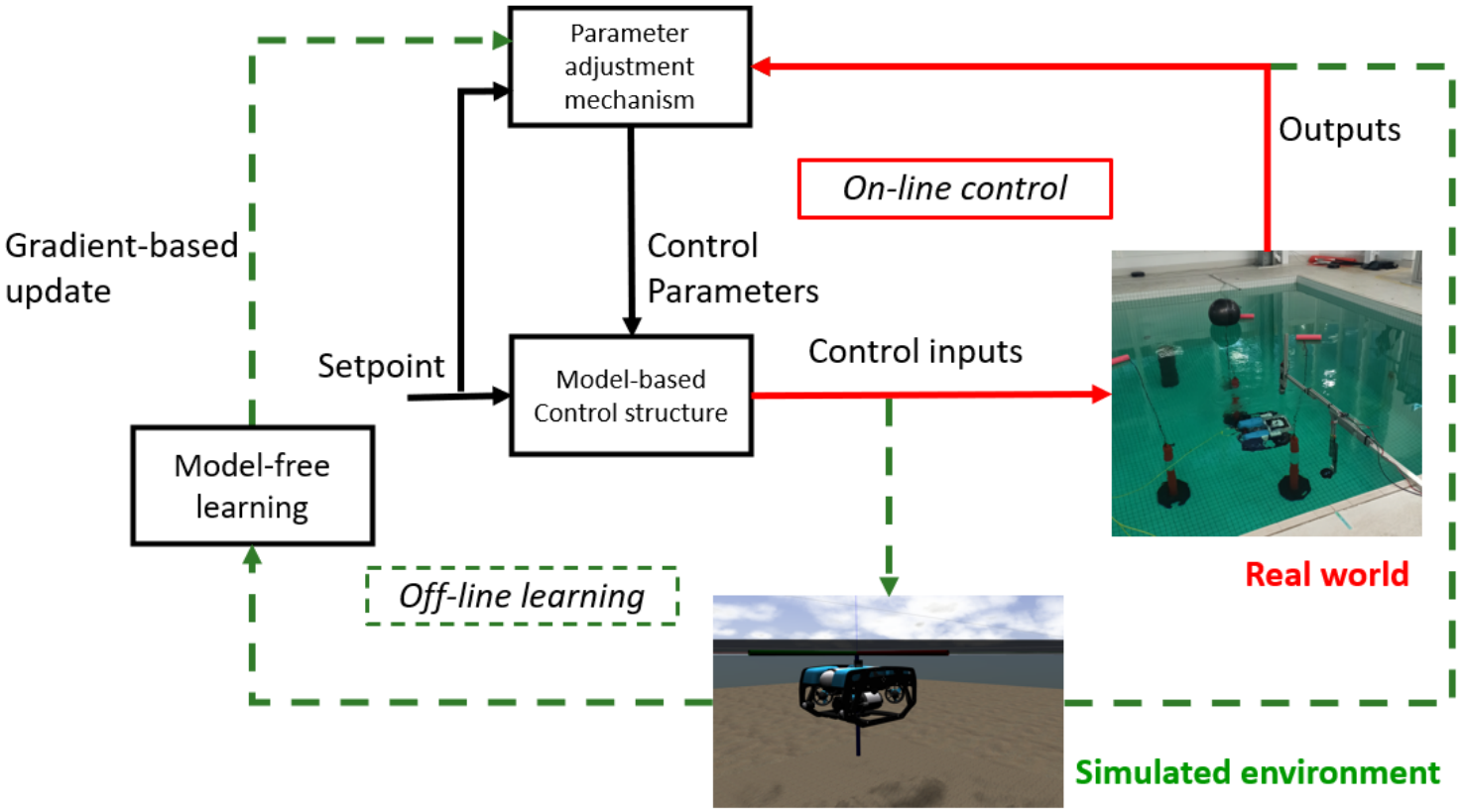


Evolution of the vehicle state stability.



Evolution of the control parameters stability.

Objective



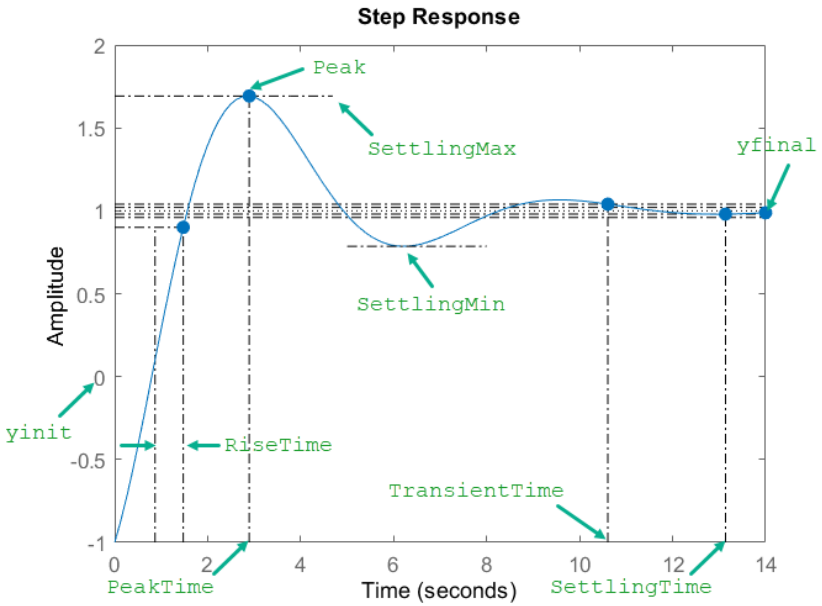
A novel learning-based adaptive control approach.

Merging RL and Adaptive Control

Model-based structure (feedback controller):

$$u = k_p e_t + k_i \min[\int e_t \, dt \, , u_{max}] + k_d\big((1-r)\dot{e}_{t-1} + r\dot{e}_t\big)$$

Pole-placement: $C(s) = \det(s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BK}))$



Merging RL and Adaptive Control

$$
\begin{aligned}
|A - BK - \lambda \boldsymbol{I}| &= -\lambda\big(\lambda(k_d + \lambda) + k_p\big) - k_i, \\
&= -\lambda^3 - \lambda^2 k_d - \lambda k_p - k_i\,, \\
&= 0.
\end{aligned}
$$

The desired λ_i are solutions of $\lambda^3 + \lambda^2 k_d + \lambda k_p + k_i = 0$.

We proposed $\lambda_i = \frac{-1}{\tau_i}$, the pole-placement design is then$*$:

$$
\begin{cases}
\frac{-1}{\tau_1^3} + \frac{k_d}{\tau_1^2} - \frac{k_p}{\tau_1} + k_i = 0 \\
\frac{-1}{\tau_2^3} + \frac{k_d}{\tau_2^2} - \frac{k_p}{\tau_2} + k_i = 0 \\
\frac{-1}{\tau_3^3} + \frac{k_d}{\tau_3^2} - \frac{k_p}{\tau_3} + k_i = 0
\end{cases}
$$

**Direct Adaptive Pole-Placement Controller using Deep Reinforcement Learning: Application to AUV Control* , T. Chaffre et al, IFAC CAMS, 2021.

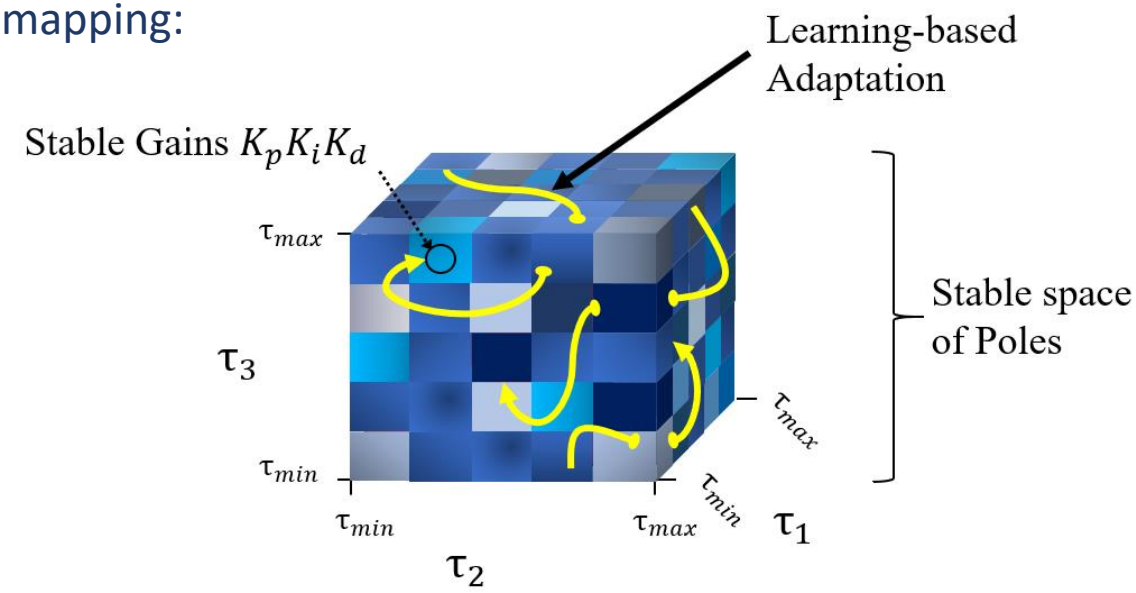
Merging RL and Adaptive Control

With $\mathrm{K}^{\mathrm{T}} = \mathrm{M}^{-1}\mathrm{N}$ we can derive the mapping:

$$k_i = \frac{1}{\tau_1 \tau_2 \tau_3}$$

$$k_p = \frac{\tau_1 + \tau_2 + \tau_3}{\tau_1 \tau_2 \tau_3}$$

$$k_d = \frac{\tau_1 \tau_2 + \tau_1 \tau_3 + \tau_2 \tau_3}{\tau_1 \tau_2 \tau_3}$$



Action space bounds:

$$\tau_{min} = 0.025 \; ; \; \tau_{max} = \frac{-\ln(0.05)}{10} = 3.33$$

Desired maximum error tolerance (in percentage)

Based on physical limits of the platform

Desired maximum settling time (in seconds)

Maximum entropy RL

Stochastic adaptive control:

$$\pi_\Omega : S \subset \mathbb{R}^{120} \rightarrow A \subset \mathbb{R}^{36},$$
$$x = [s_t] \mapsto [\lambda_i, \mu_i]$$

where τ_i is modeled by a Gaussian distributions $\mathcal{N}(\tau_i)$ such as:

$$\mathcal{N}(\tau_i) = (2\pi\mu_i)^{-1/2} \exp\left\{-\frac{1}{2\mu_i}(x-\lambda_i)^2\right\},$$

where $\lambda_i \in R, \mu_i \in R^+$ are the mean and variance of $\mathcal{N}(\tau_i)$.

Maximum entropy RL

1. Actor-Critic architecture

2. Maximum entropy framework

3. Off-policy formulation

Maximum entropy RL

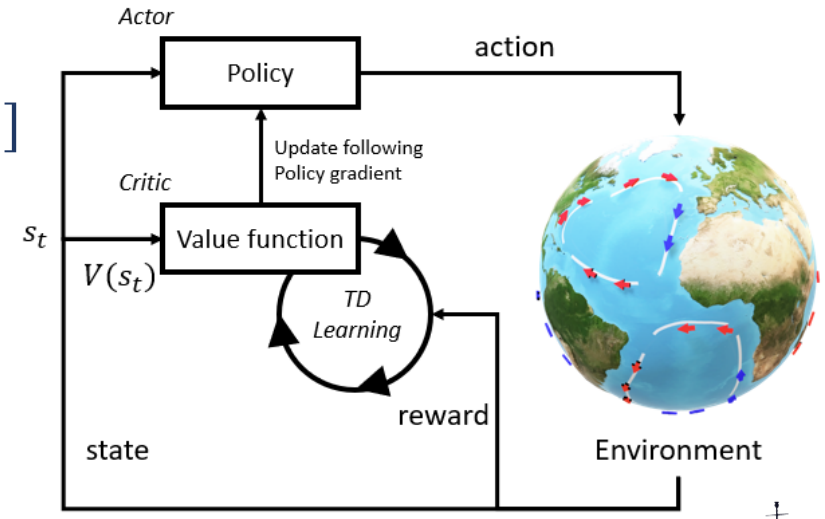
1. Actor-Critic architecture

Actor π: choose the best action to execute.
Critic Q : tells the actor how good this choice was.

Using Bellman equations:

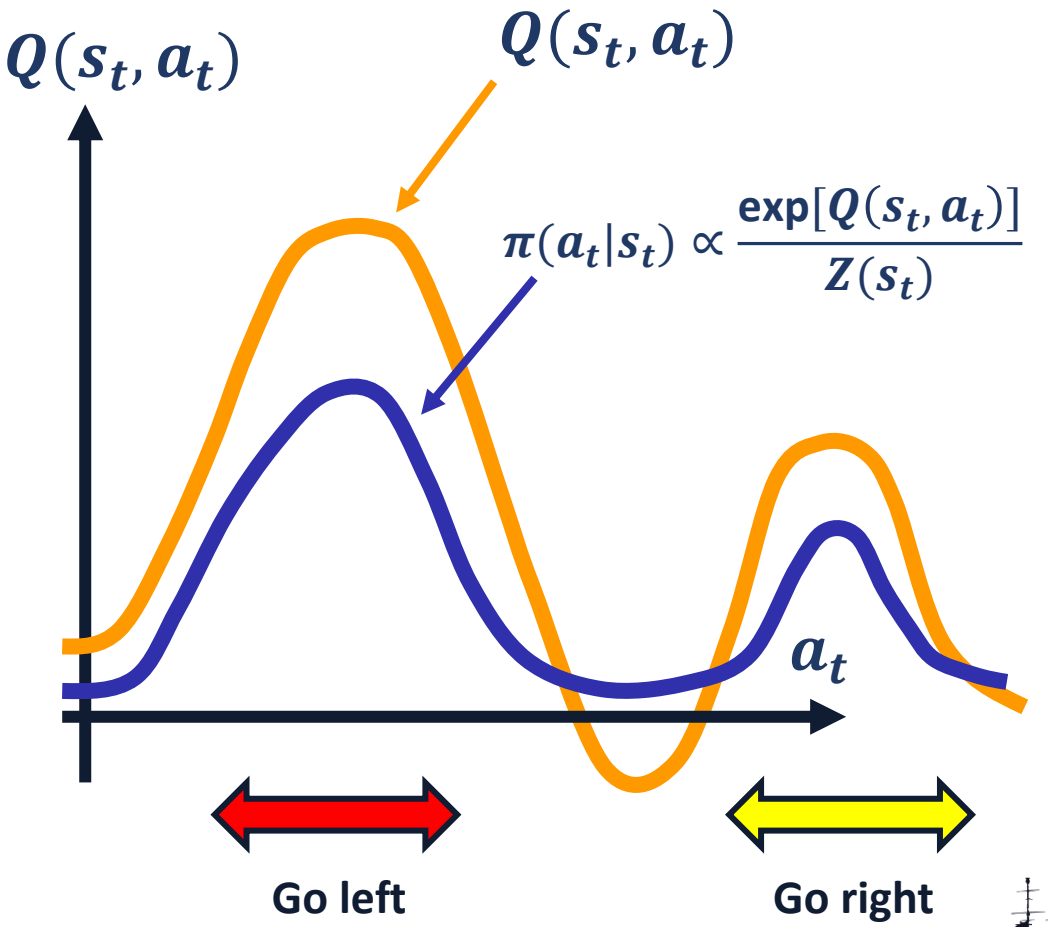
$$Q_\pi(s_t, a_t) = [\mathbb{E} \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | \, S_t = s_t, A_t = a_t]$$

$$\pi^*(a_t|s_t) = arg \max_\pi Q_\pi(s_t, a_t)$$



Maximum entropy RL

2. Maximum entropy framework



Maximum entropy RL

2. Maximum entropy framework

Equivalent to:

$$\min_{\pi} DKL(\pi(\cdot\,|s_0)||\exp\{Q(s_0,\cdot)\}))$$

Up to an additive constant, it is equal to:

$$J_{maxEntro}(\pi|s_0) = \max_{\pi} \mathbb{E}_{\pi}[Q(s_0,a_0) - \log\pi(a_0|s_0)]\,,$$

which we can further decompose as:

$$J_{maxEntrop}(\pi|s_0) = \max_{\pi} \mathbb{E}_{\pi}[\textstyle\sum_t r(s_t,a_t) + \alpha\boldsymbol{\mathcal{H}}\big(\pi(\cdot\,|s_t)\big)|s_0].$$

Maximum entropy RL

3. Off-Policy formulation

The optimal policy can be estimated using samples from any policies:

$$\nabla_\mu J(\mu) = \mathbb{E}_\beta \left[\frac{\pi_\mu(a|s)}{\beta(a|s)} Q^\pi(s,a) \nabla_\mu \log \pi_\mu(a|s) \right]$$

Experience Replay (ER) concept:

- The agent experience is store in a Replay Buffer.
- At each timestep, mini-batch gradient is performed using samples from the Replay Buffer.

Taking inspiration from biological replay mechanism

Missing biological elements*

- Replay (in biological systems) is **temporally structured**.
- Replay is **modulated by reward** and only a few selected experiences are replayed.
- Replay is **treated differently** for novel version non-novel inputs.

Taking inspiration from biological replay mechanism

Standard ER and limits

A deeper look at Experience Replay, **Richard S. Sutton et al., 2018.**

Algorithm 3: Combined-Q

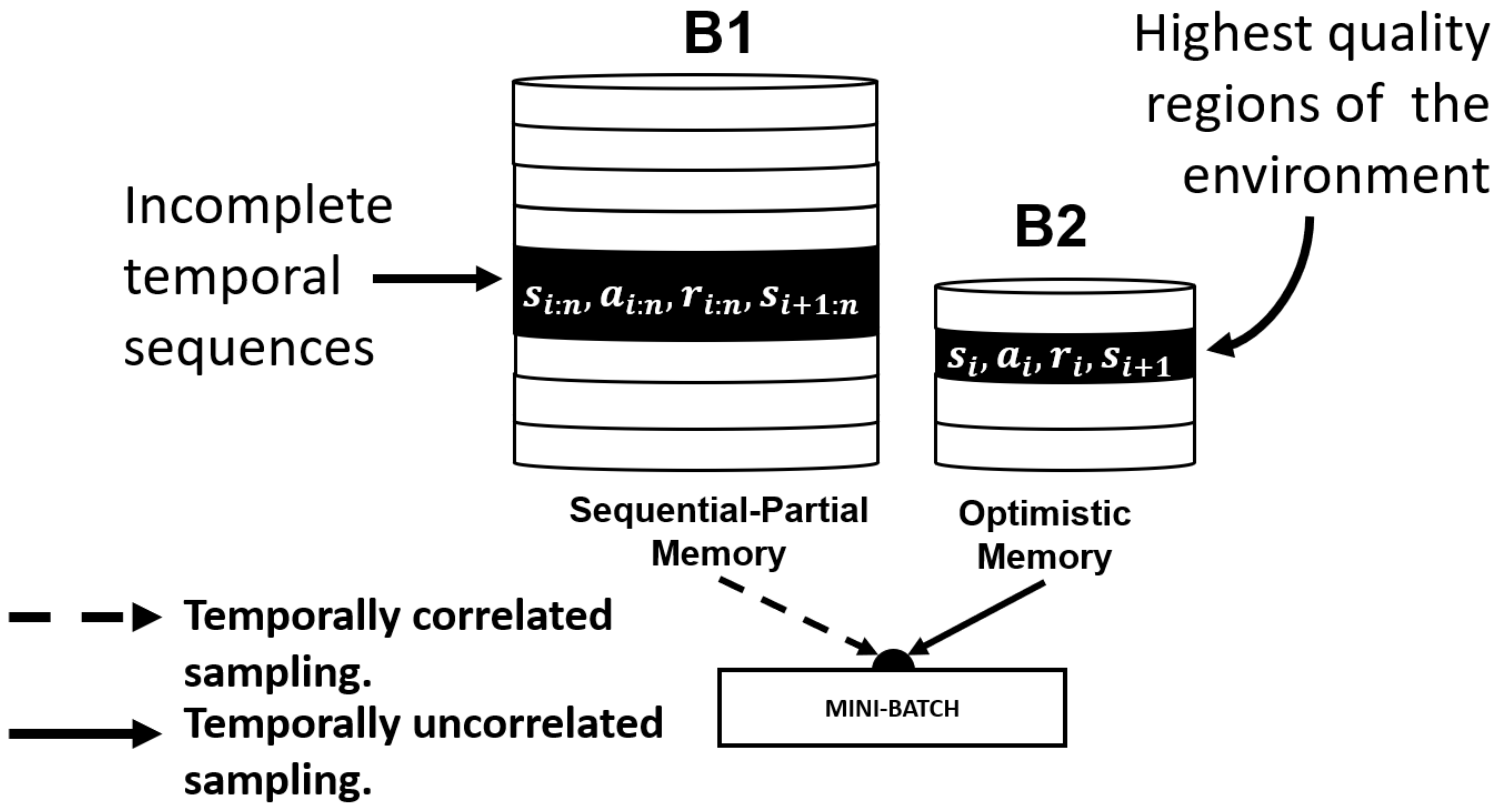
Initialize the value function Q
Initialize the replay buffer $\mathcal{M}$
while *not converged* **do**
  Get the initial state S
  while S *is not the terminal state* **do**
    Select an action A according to a ϵ-greedy policy derived from Q
    Execute the action A, get the reward R and the next state S'
    Store the transition $t = (S, A, R, S')$ into the replay buffer $\mathcal{M}$
    Sample a batch of transitions $\mathcal{B}$ from $\mathcal{M}$
    Update the value function Q with $\mathcal{B}$ and t
    $S \leftarrow S'$
  end
end

Combined Experience Replay (CER)

Revisiting fundamentals of Experience Replay, **Yoshua Bengio et al., 2020.**

- Increasing replay capacity lower chance of overfitting.
- Learning in high quality regions (as measured by reward) leads to further gains.
- Increasing replay capacity with fixed replay ratio has varying performance.

Taking inspiration from biological replay mechanism



- **Adds regularization effect**
- **Reduces age of oldest policy**

If $\mathbf{r(s_t) > \mathbb{E}_\pi[r(s_t)]}$
Because $A(s,a) = Q(s,a) - V(s)$

Sim-to-Real Transfer

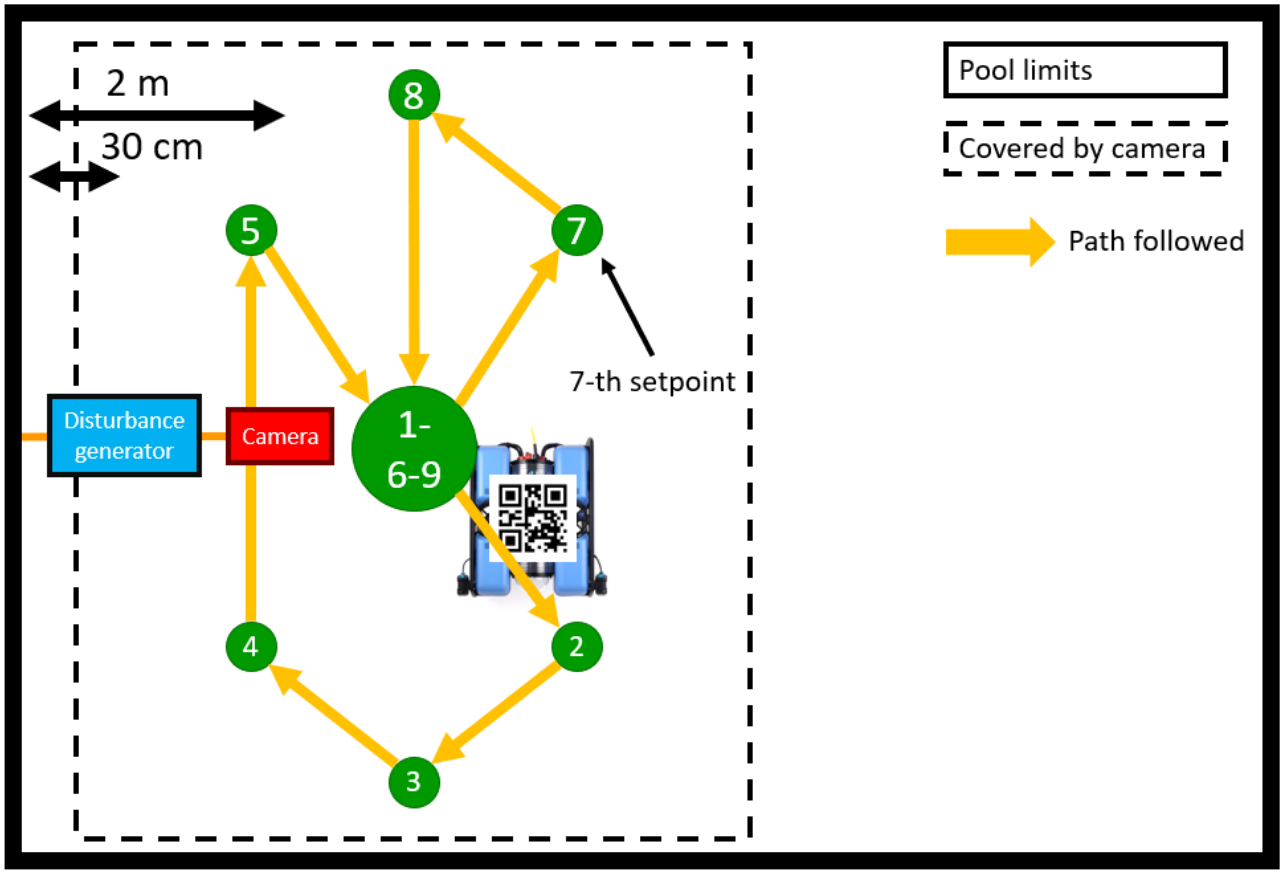
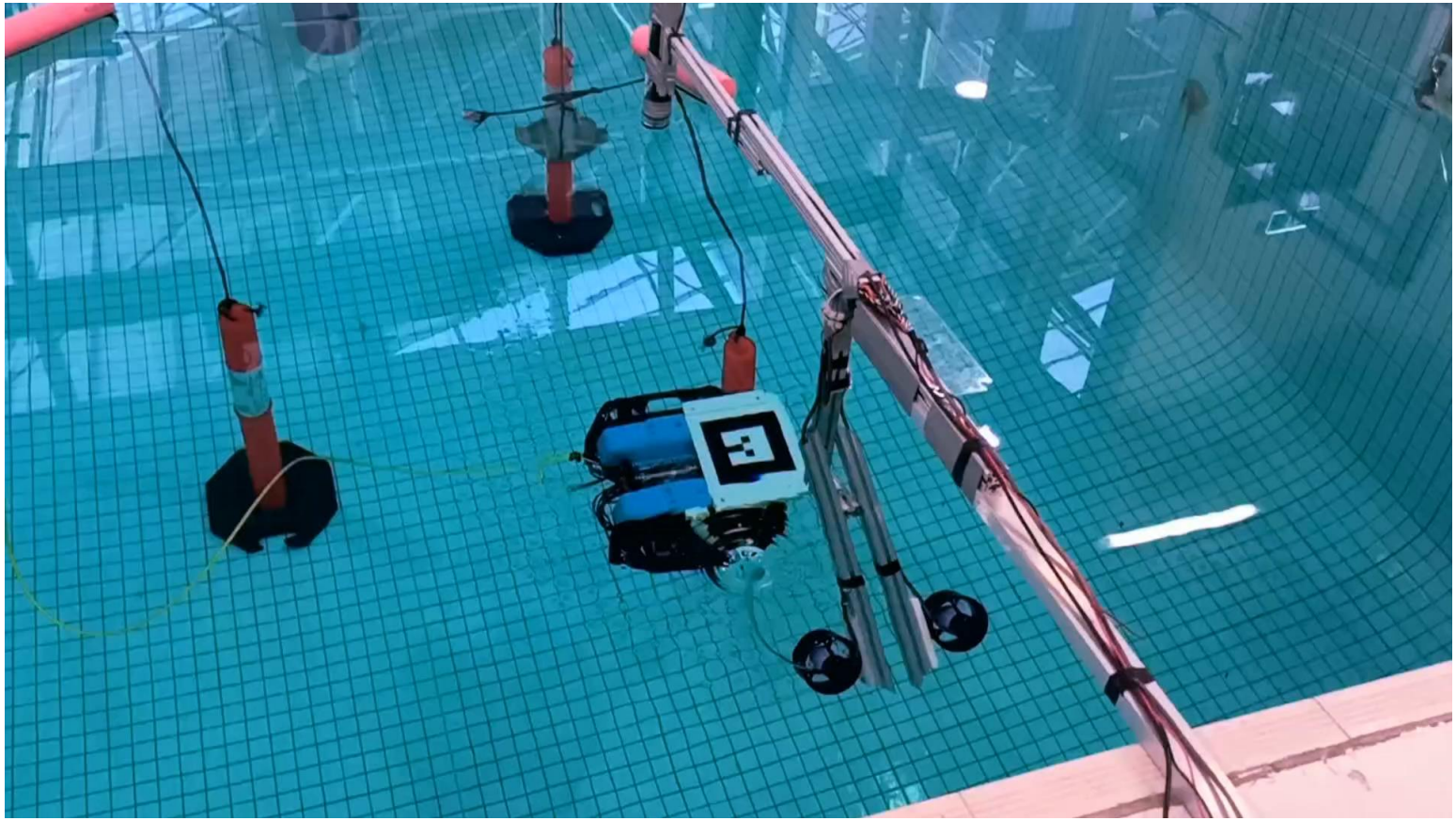


Illustration of the task for the experimental validation.

- Precision of positioning system: 3-4 centimeters after calibration.
- Evaluated 2 control systems against 2 operating conditions (with and without current disturbance).
- Each controller is performing the task 10 times and performance are average results.

Sim-to-Real Transfer



The thrusters are controlled through ESC input that we set to 1625, which according to Blue Robotics documentation gives around 8 Newtons of thrust per thruster. The total current draw for the pair is approximately 2.7A, providing a power draw of around 38 Watts.

Sim-to-Real Transfer

Reward function:

$$r(s_t) = exp^{-(e_{L2}(t))}$$

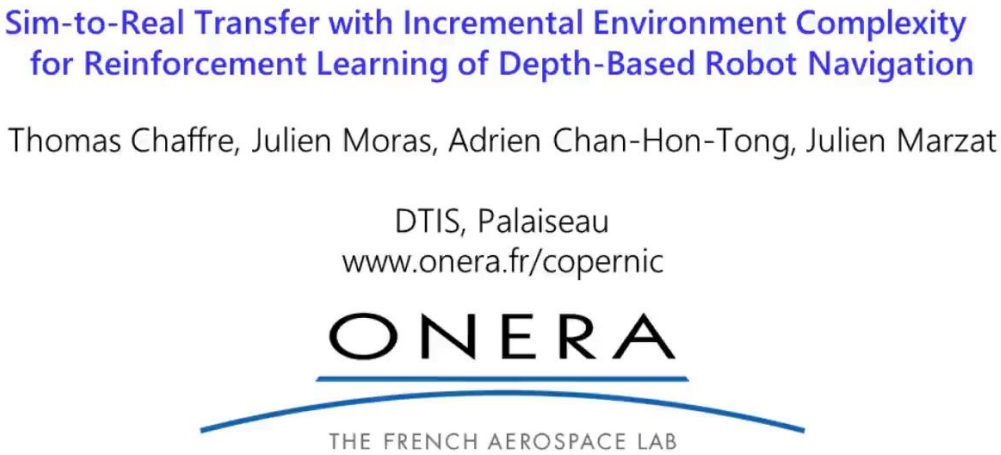
"Euclidean distance" to the setpoint

Table 5.2: *List of hyperparameters and their values for experimental validation.*

Training hyperparameter	Value
SAC version	2 (see Section 2.2.8)
Activation function	Leaky ReLU
Optimizer (all networks)	Adam [KB15]
Learning rate (all networks)	3×10^{-4}
Discount factor (γ)	0.99
Mini-batch size	256
Target network smoothing coefficient (Δ)	0.005 (see Section 2.2.8)
Delayed update trick [FHM18]	True
Critics L2 regularization	0.001
Layer Normalization [BKH16] (all networks)	True
Automatic temperature adjustment	True
Replay buffer max size	$1e6$
Replay start size	$1e4$
Experience Replay method	BIER (see Section 3.3.3)

Sim-to-Real Transfer

Randomized environmental complexity as domain randomization*



**Sim-to-Real Transfer with Incremental Environment Complexity for Reinforcement Learning of Depth-Based Robot Navigation,* T. Chaffre et al, ICINCO, 2020.

Sim-to-Real Transfer

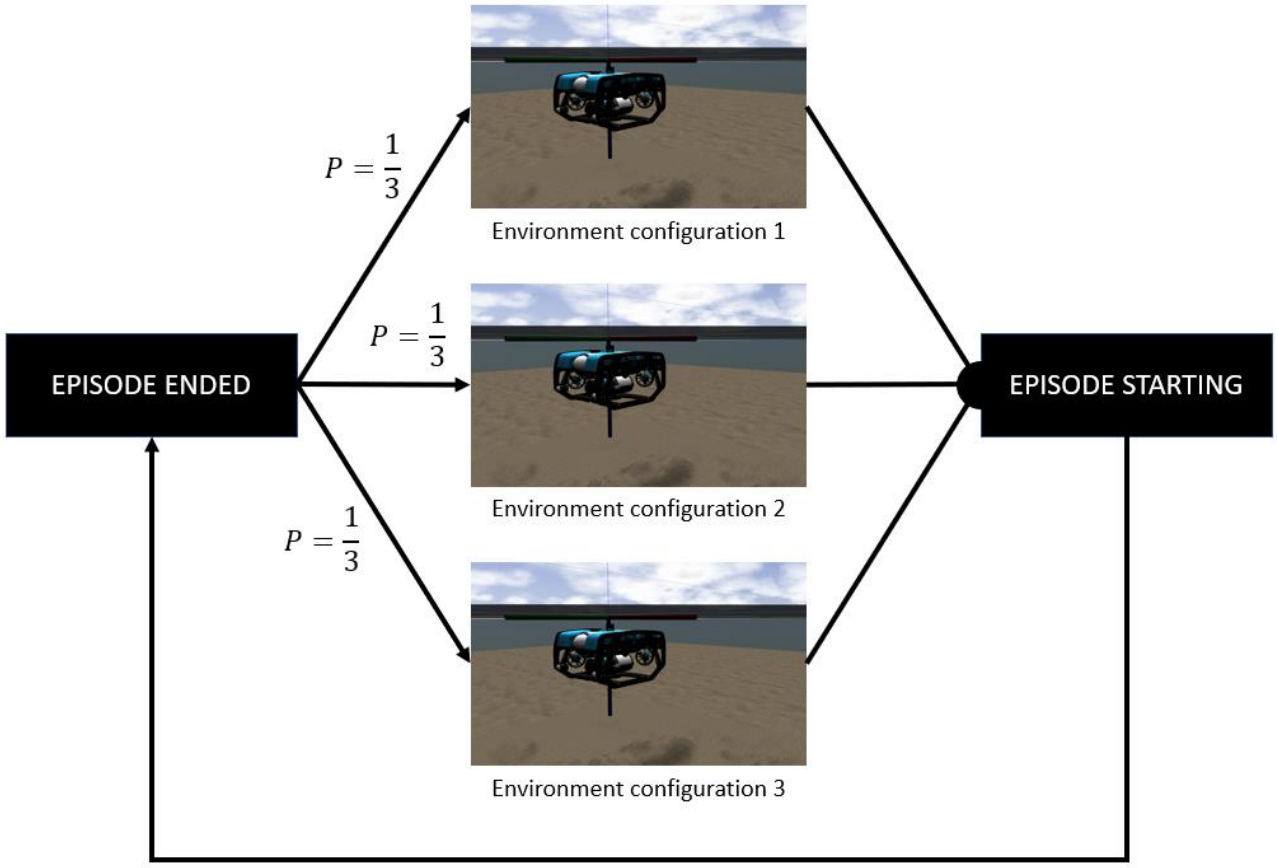
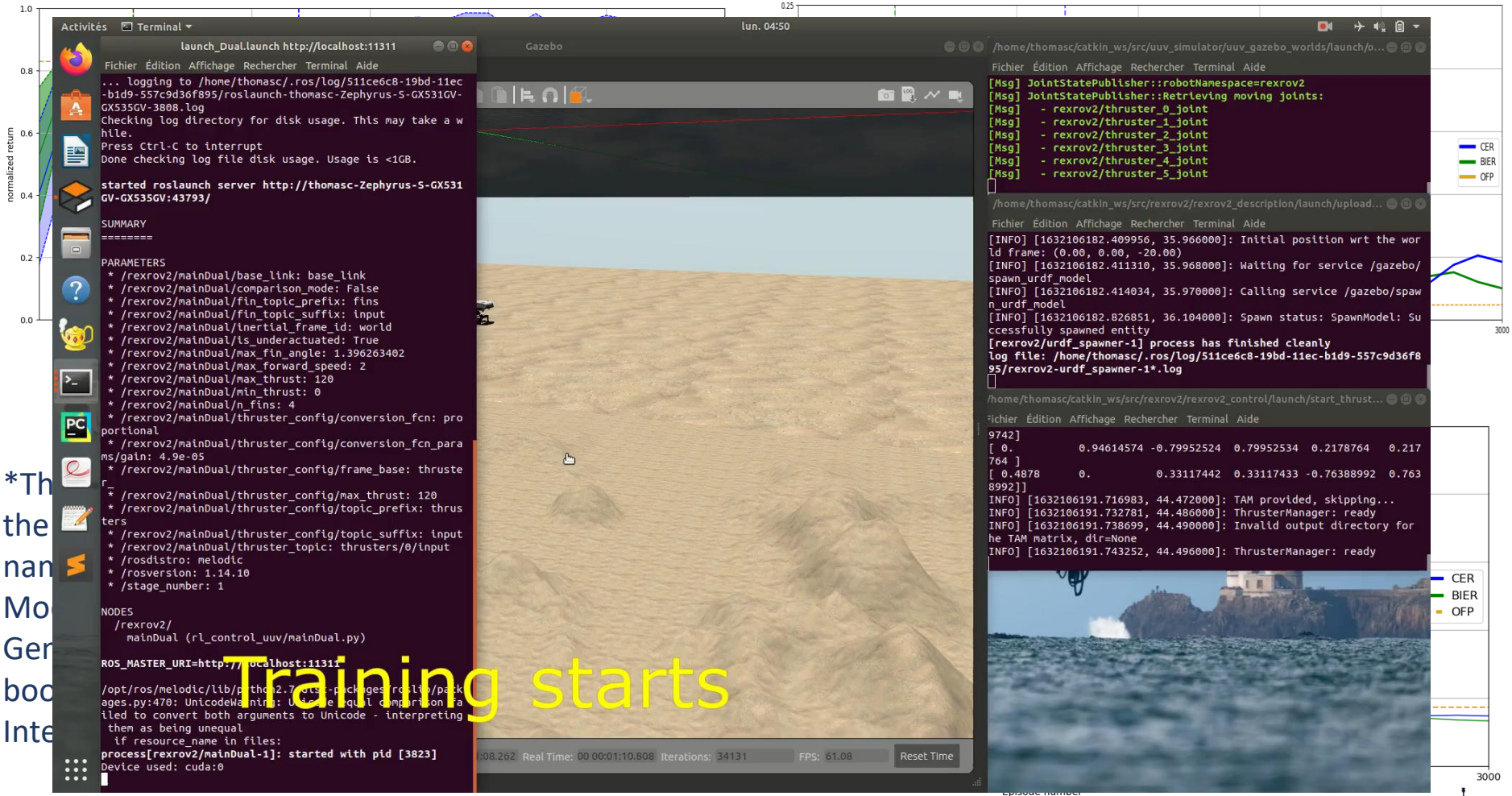


Illustration of the proposed domain randomization method. We proposed to divide the training environment in three configurations in terms of environment complexity. At the beginning of each episode, the environment characteristics are set to one of these configuration with an equal probability P for each case.

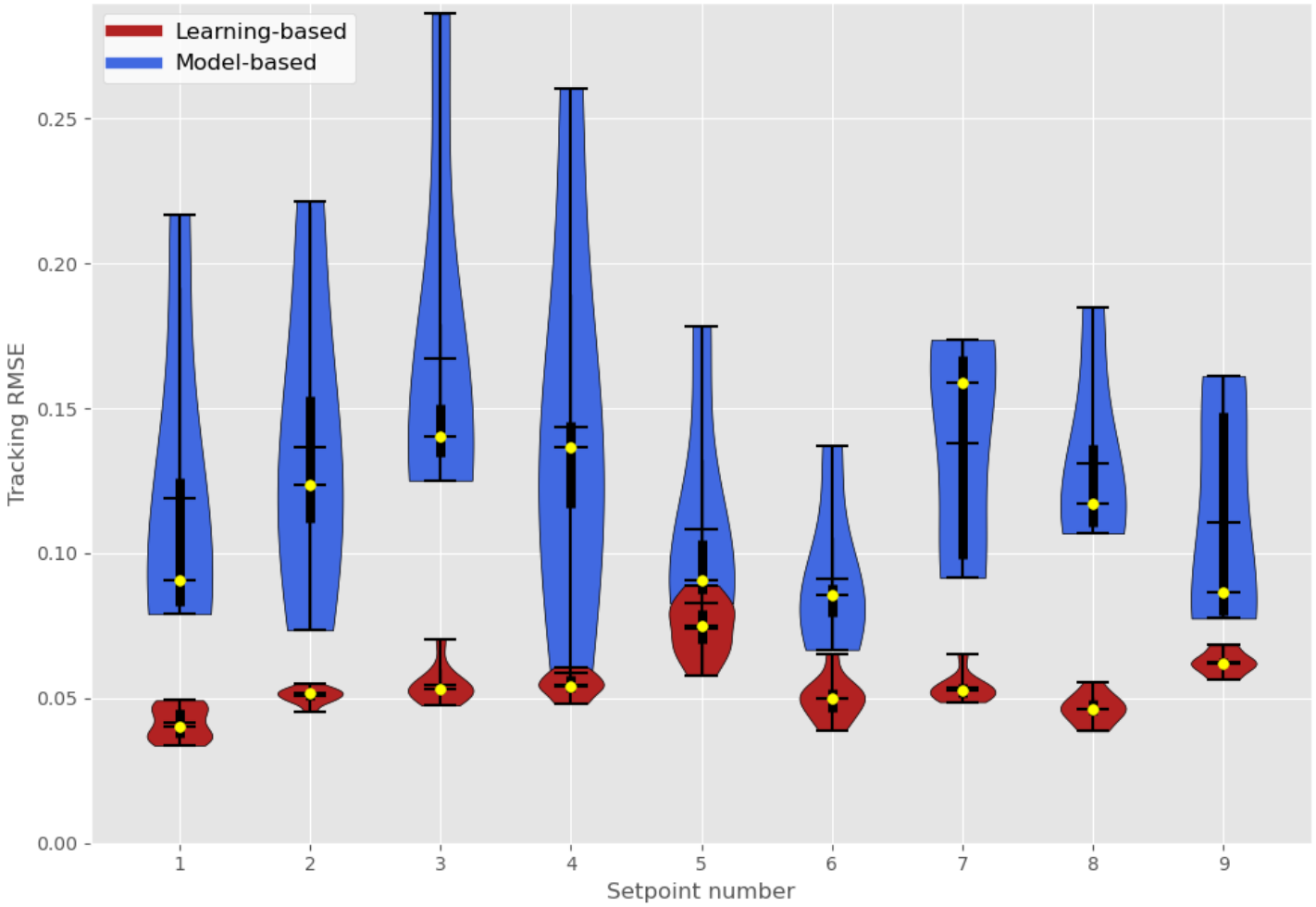
Training



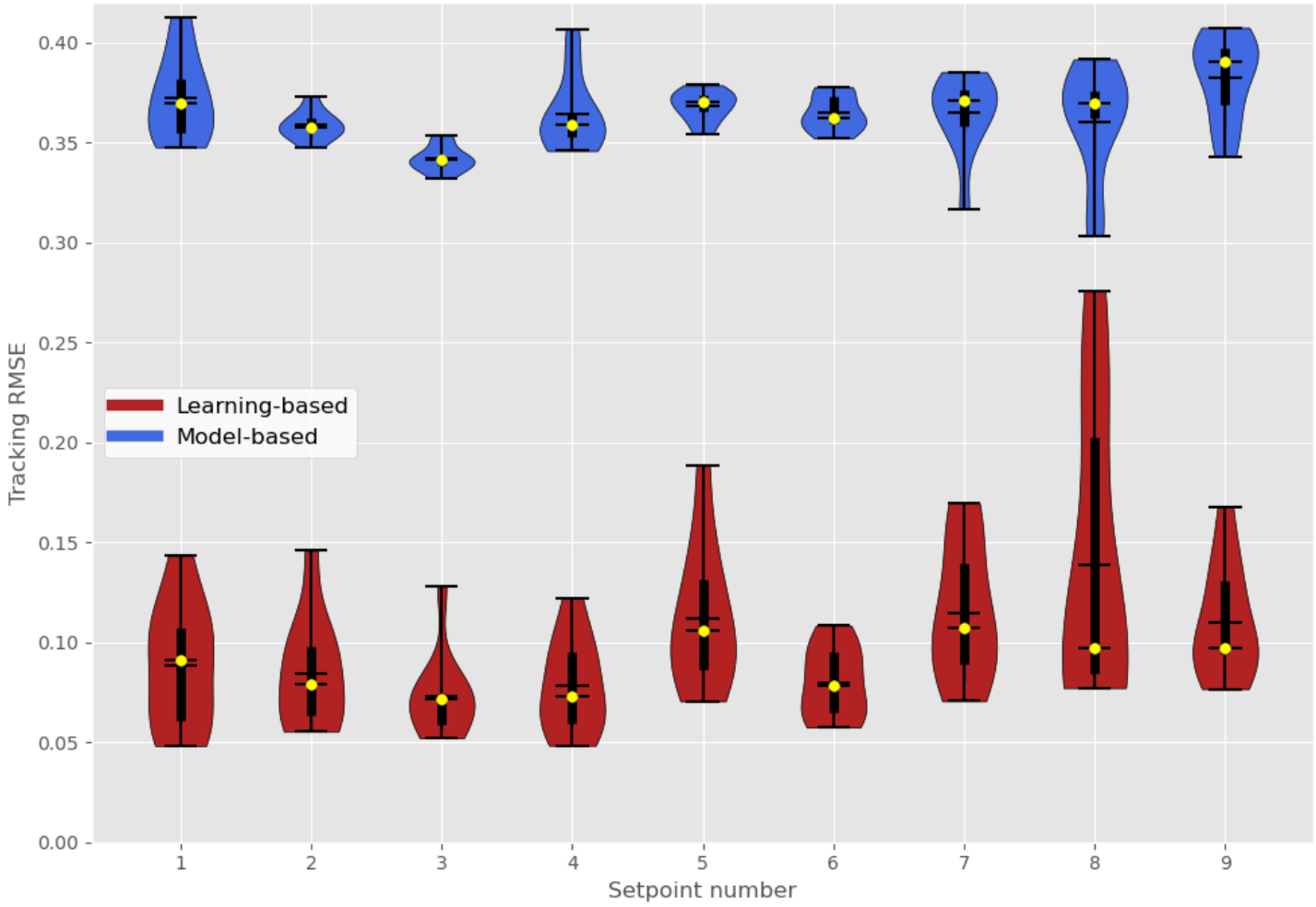
*Th
the
nam
Mo
Ger
boo
Inte

Setpoint RMSE

Results: without disturbance



Results: with disturbance



Current research

7 Peer-reviewed papers

Ros package on private GitHub

Ongoing :

- Nonlinear adaptive control structure
- Real life reinforcement learning
- Multi-tasks / multi-agents

Future research

- Offline Reinforcement Learning

Offline Q-Learning on Diverse Multi-Task Data Both Scales And Generalizes, by A. Kumar et al., 2022.
Offline RL With Realistic Datasets: Heteroskedasticity and Support Constraints, by A. Singh et al., 2022.
Pre-Training for Robots: Offline RL Enables Learning New Tasks from a Handful of Trials, by A. Kumar et al., 2022.
Don't Start From Scratch: Leveraging Prior Data to Automate Robotic Reinforcement Learning, by H. Walke et al., 2022.

- Internal disturbances

Next PhD student (Katell Lagattu CIFRE Naval Group)

- Reducing value overestimation

Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability, by D. Ghosh at al., 2021.
Value Function Spaces: Skill-Centric State Abstractions for Long-Horizon Reasoning, by D. Shah et al., 2021.
How to train your robot with deep reinforcement learning: lessons we have learned, by J. Ibarz et al., 2021.

Acknowledgments

Thank you!

t.chaffre@gmail.com

Dissertation committee:

- David Filliat, ENSTA Paris
- Tirthankar Bandyopadhyay, CSIRO

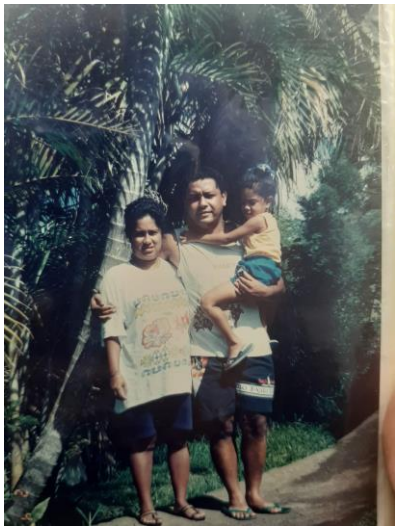
- Benoit Clement, ENSTA Bretagne
- Karl Sammut, Flinders University
- Gilles Le Chenadec, ENSTA Bretagne
- Paulo Santos, Flinders University
- Estelle Chauveau, Naval Group
- Jean-Philippe Diguet, IRL CROSSING
- Julien Marzat, ONERA
- Jonathan Wheare, Flinders University
- Andrew Lammas, Flinders University

Thank you!

t.chaffre@gmail.com

Look, a kangaroo!

Sunburn on my 1st day on campus



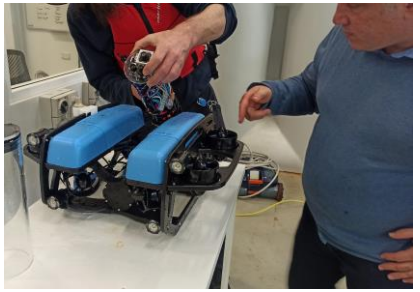
Thanks to my parents and sister

When you need Dynamic Programming to find the optimal quantity/cost tradeoff

No Kangaroos here

Presenting our work to Antoine PETIT, President of CNRS

Starting my addiction to flat white

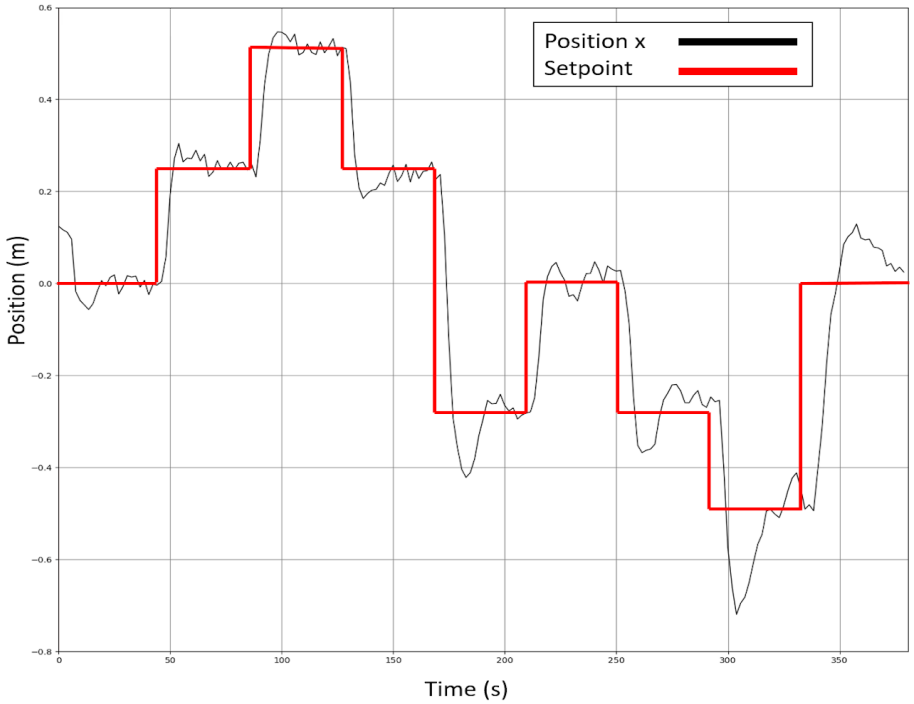


ANOTHER leak…

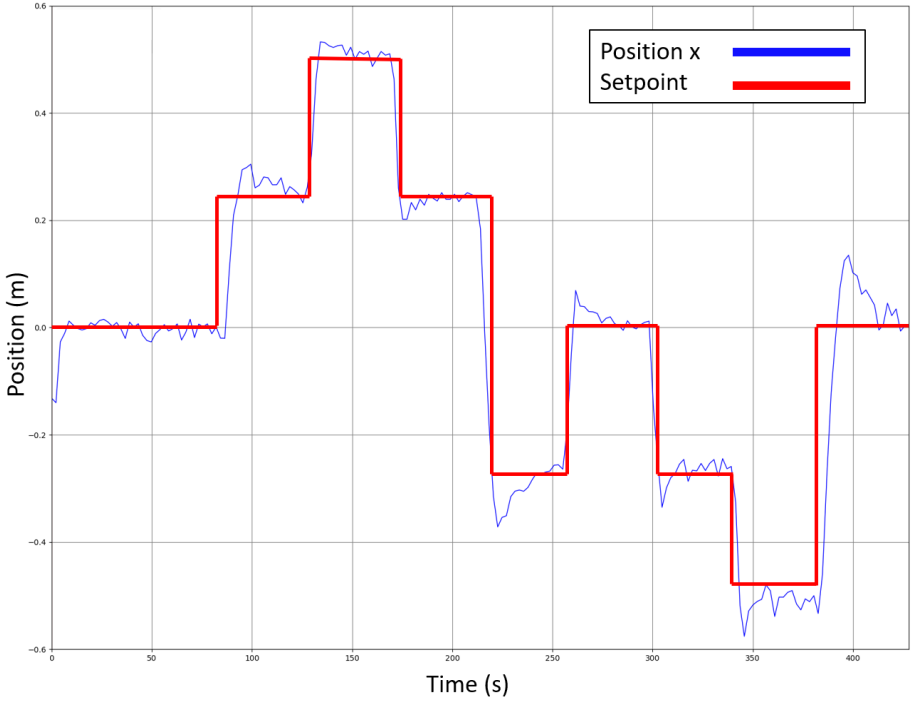
"Please acknowledge the following limitations"

Reviewer 3 be like

Trajectories: with disturbance

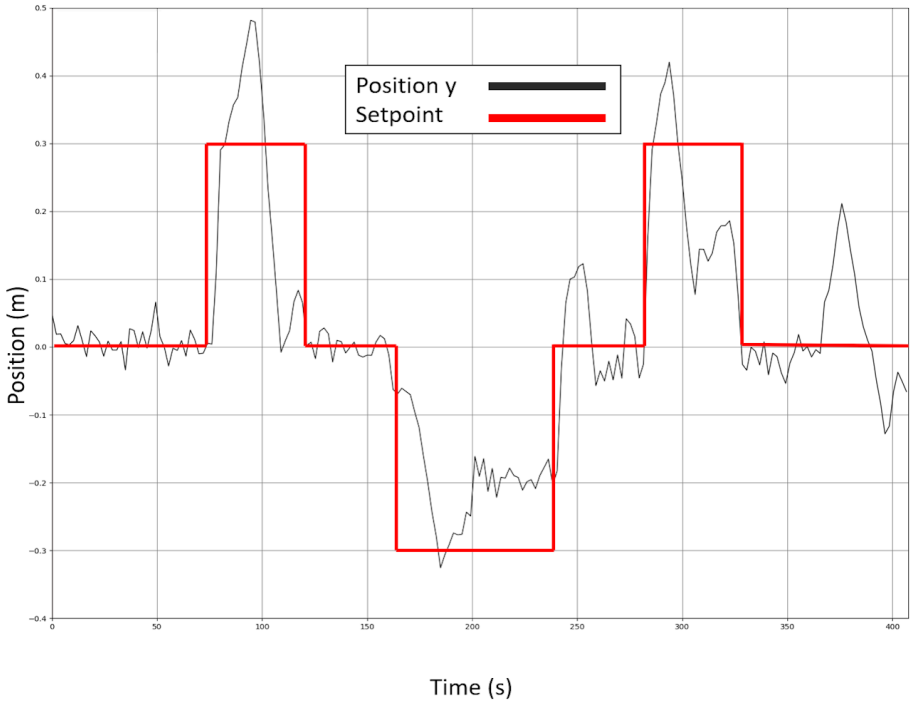


Position X with the MB controller.

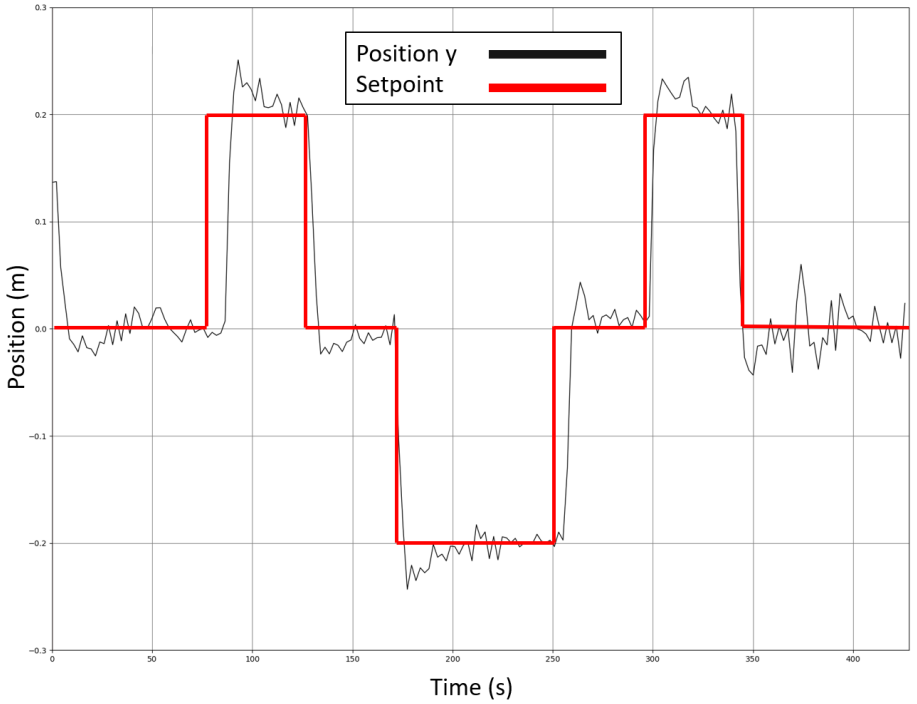


Position X with the LB controller.

Trajectories: with disturbance

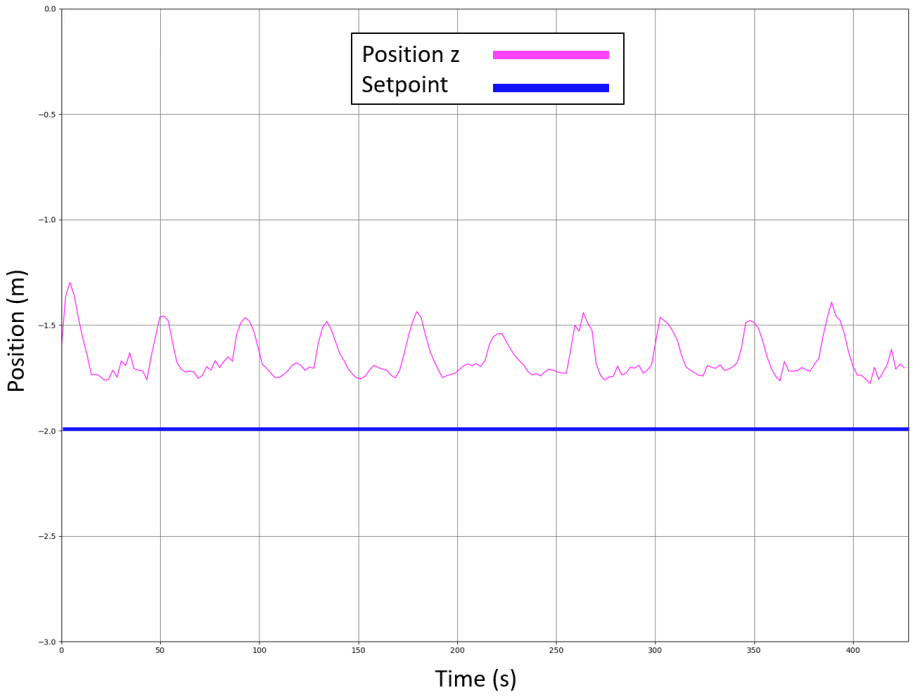


Position Y with the MB controller.

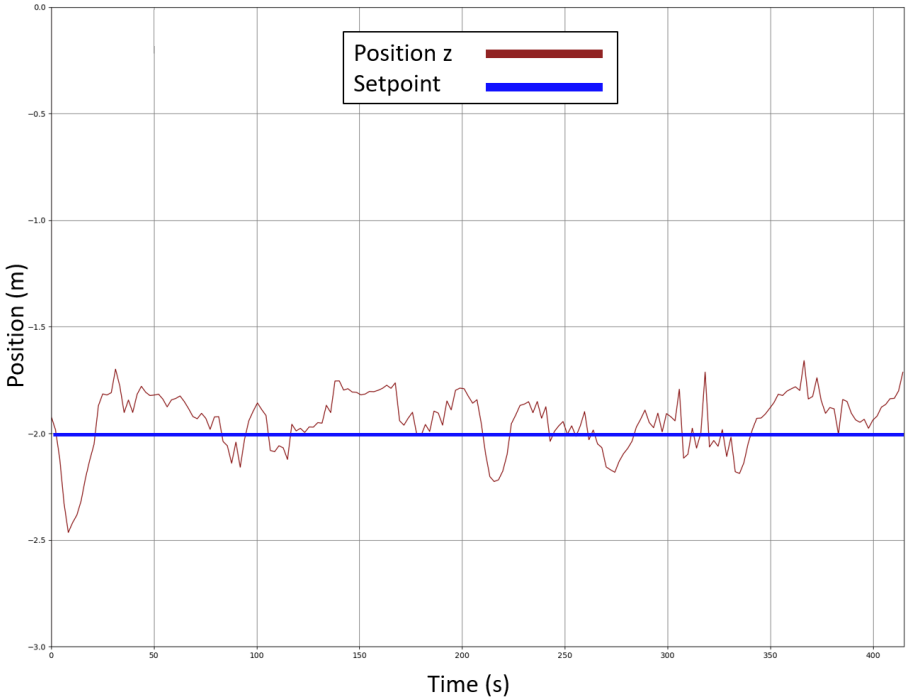


Position Y with the LB controller.

Trajectories: with disturbance

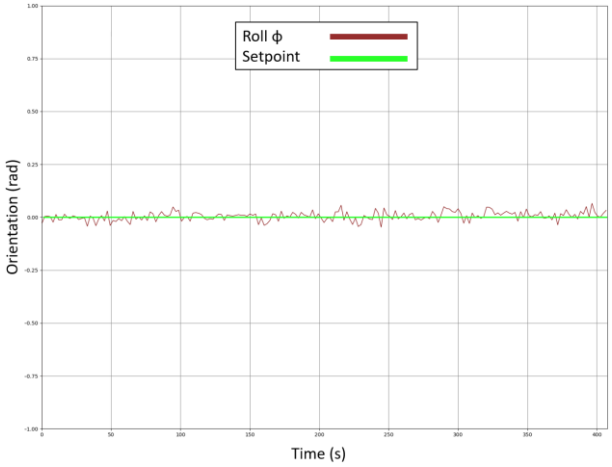


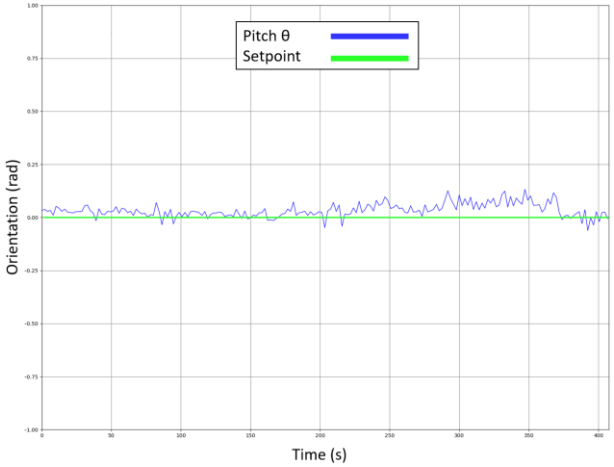
Position Z with the MB controller.

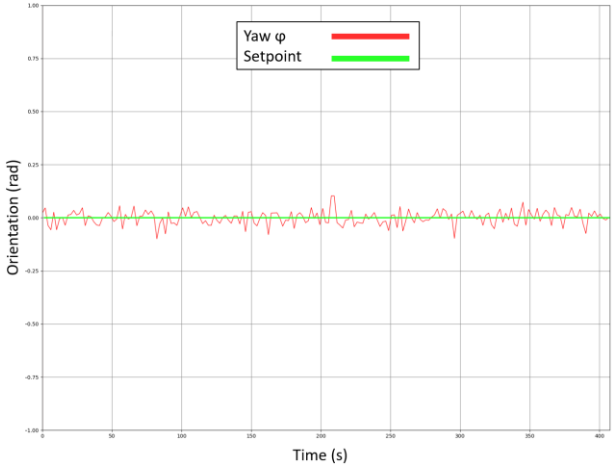


Position Z with the LB controller.

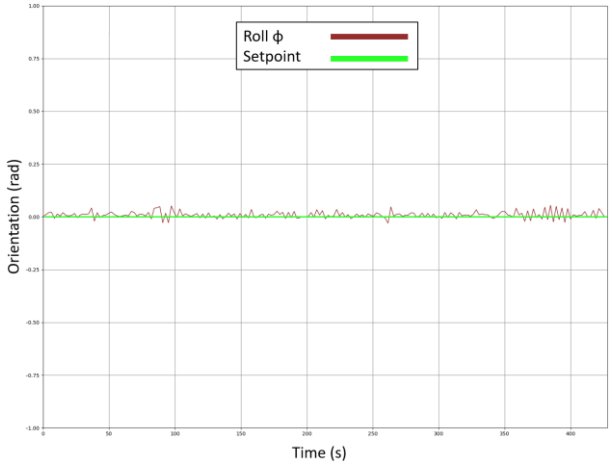
Trajectories: with disturbance

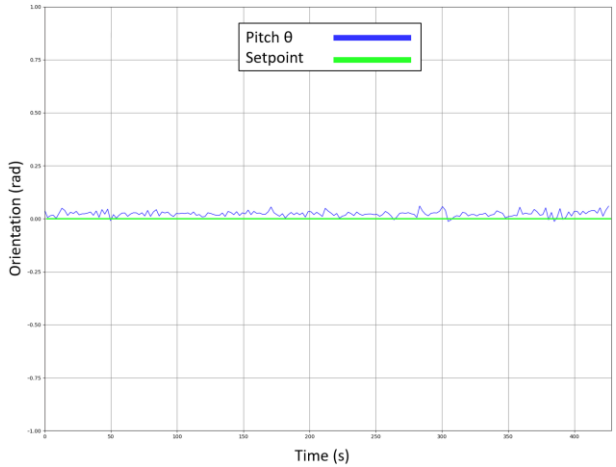


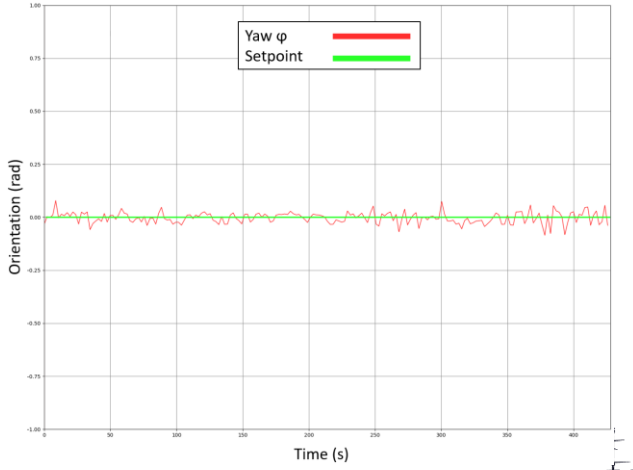




Orientation with the MB controller.







Orientation with the LB controller.

Experimental setup

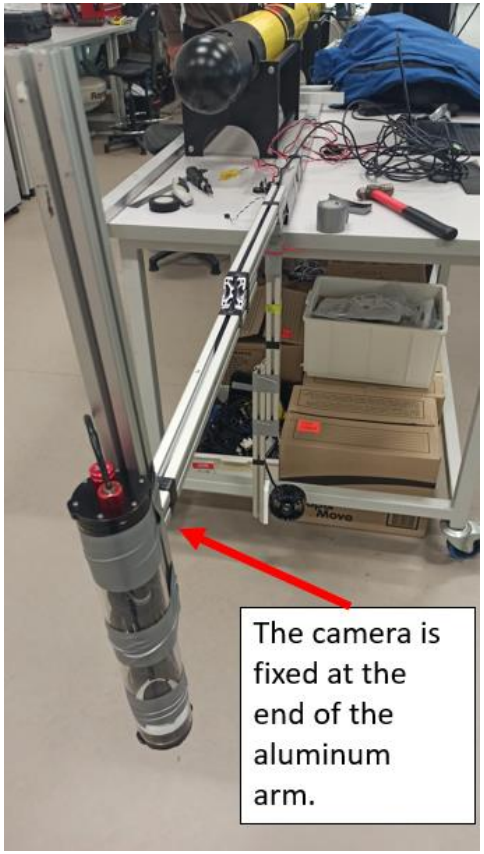
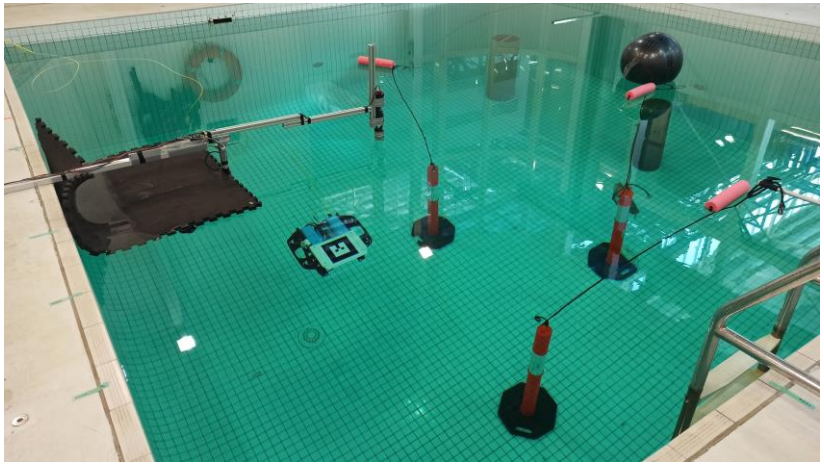
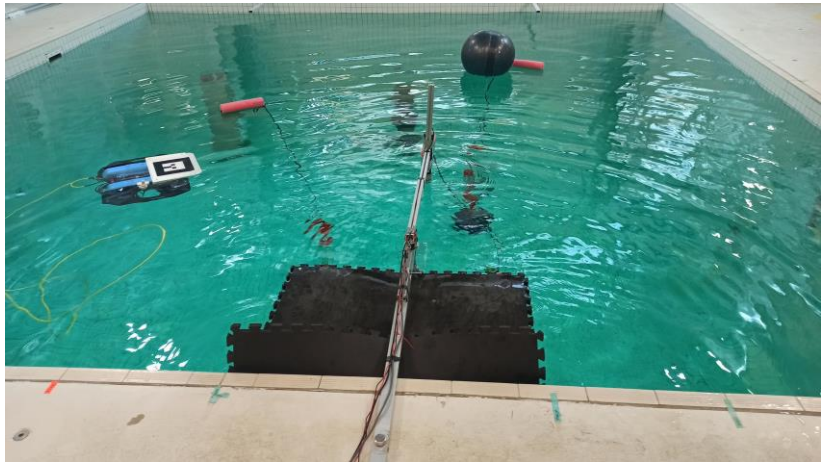
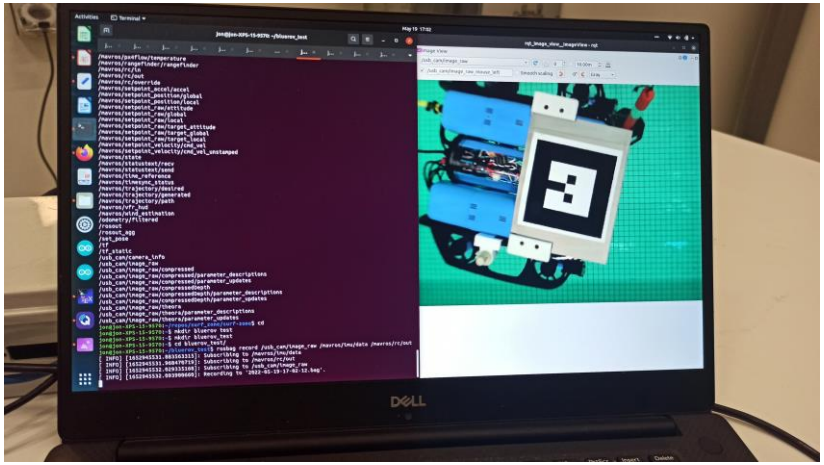


Illustration of the disturbance generator and tracking system for pose estimation.

Experimental setup







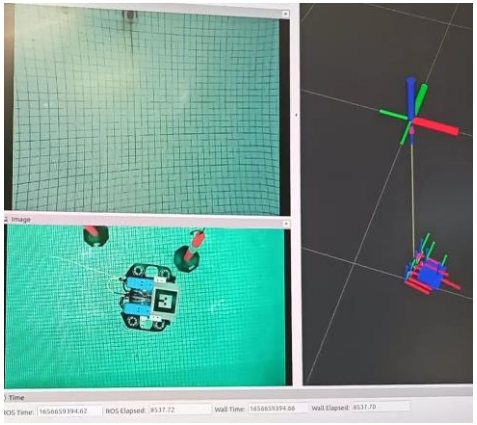
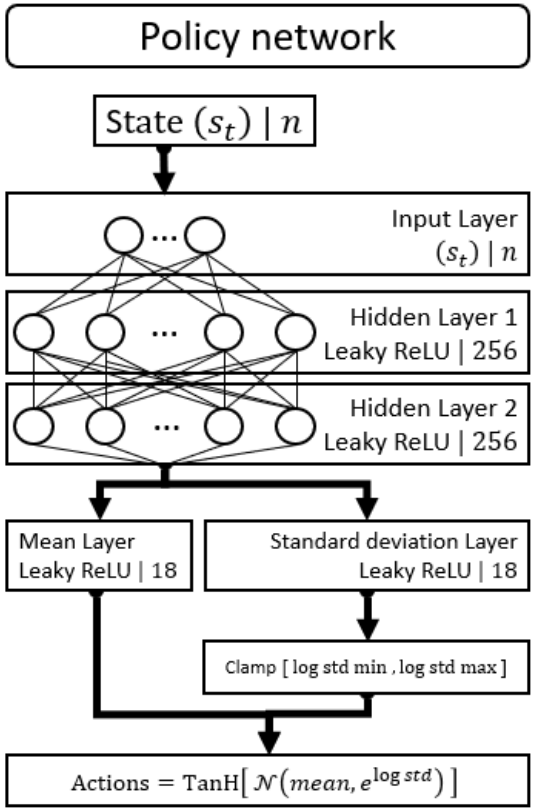
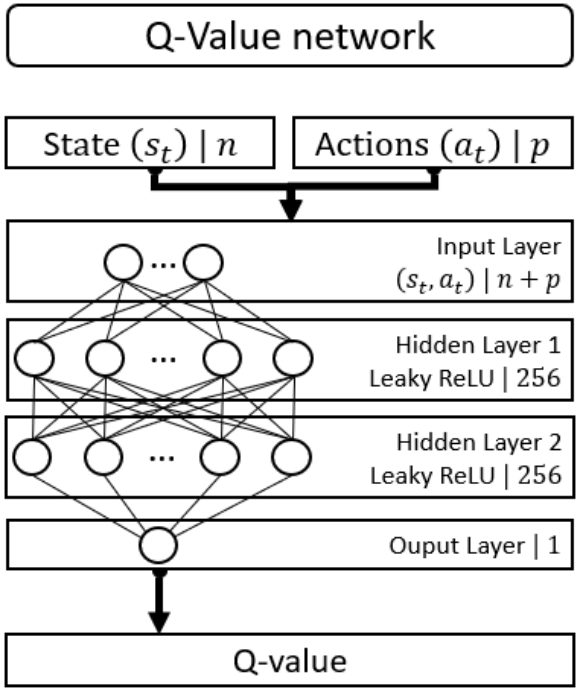
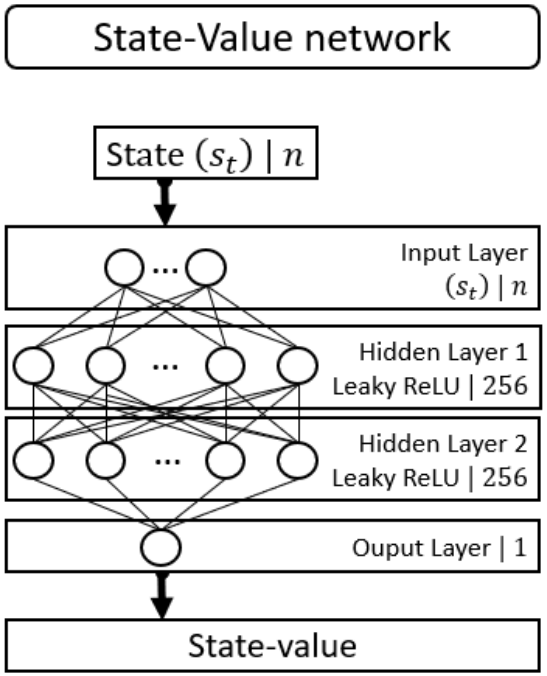


Illustration of the experiments setup.

Neural network architecture



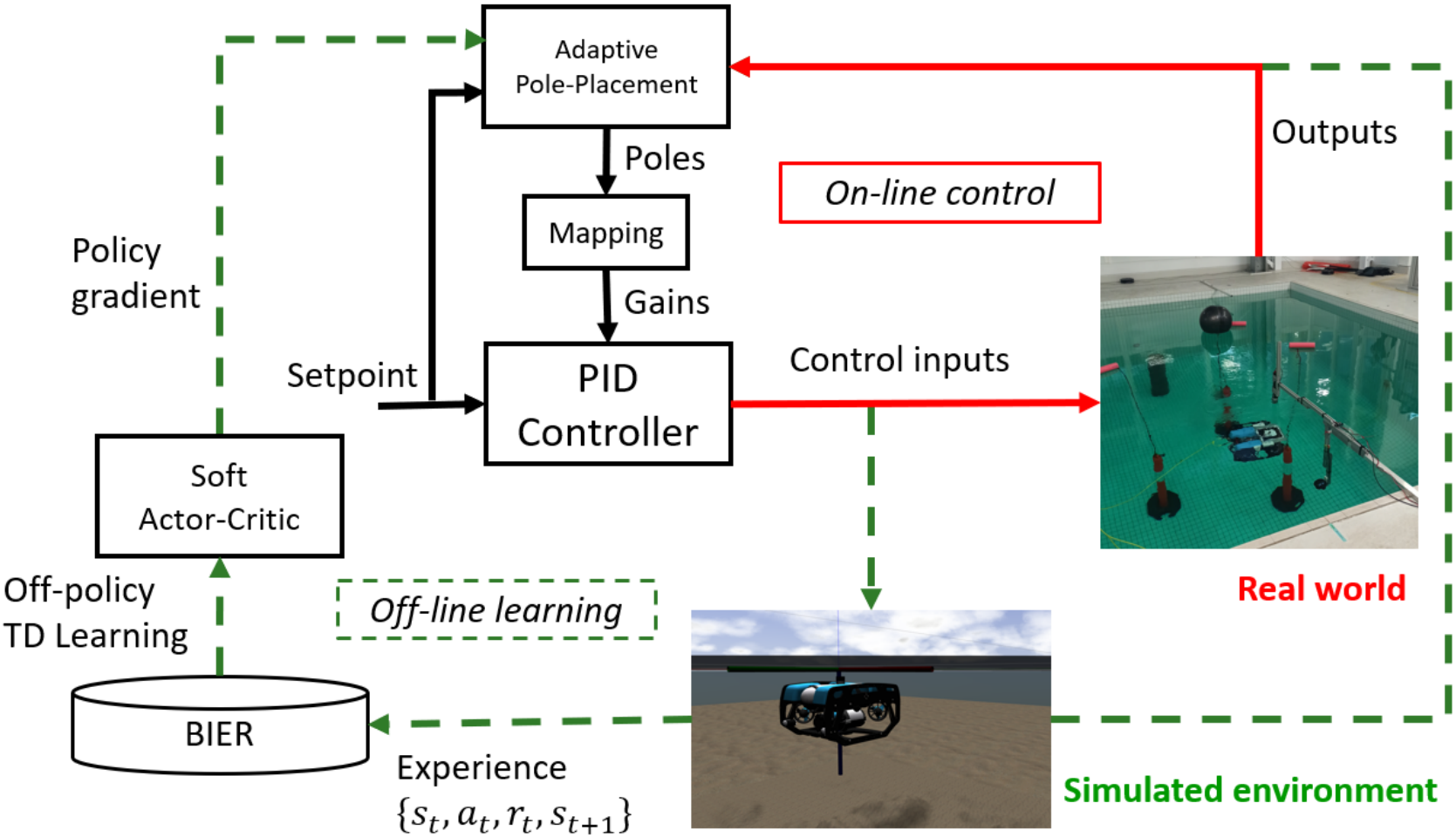




With:

- Adam
- Layer Norm ($\neq$ Batch Norm)
- L2 weight decay on Critics only (0.001)
- Soft target update ($\Delta = 0.005$) and delayed update (from TD3 algorithm)

Proposed learning-based controller



Experimental protocol

Simulated training:

- Station keeping under several process variations.
- Observation vector o_t includes: AUV state, setpoint errors, last poles estimates, last control inputs, distance to setpoint (dim = 42).
  - State vector $s_t = [o_t ; o_{t-1} ; o_{t-1} - o_t](dim = 126)$.

Simulated evaluation:

- Same task.
- 6 scenarios of varying complexity.
- Lyapunov stability analysis.

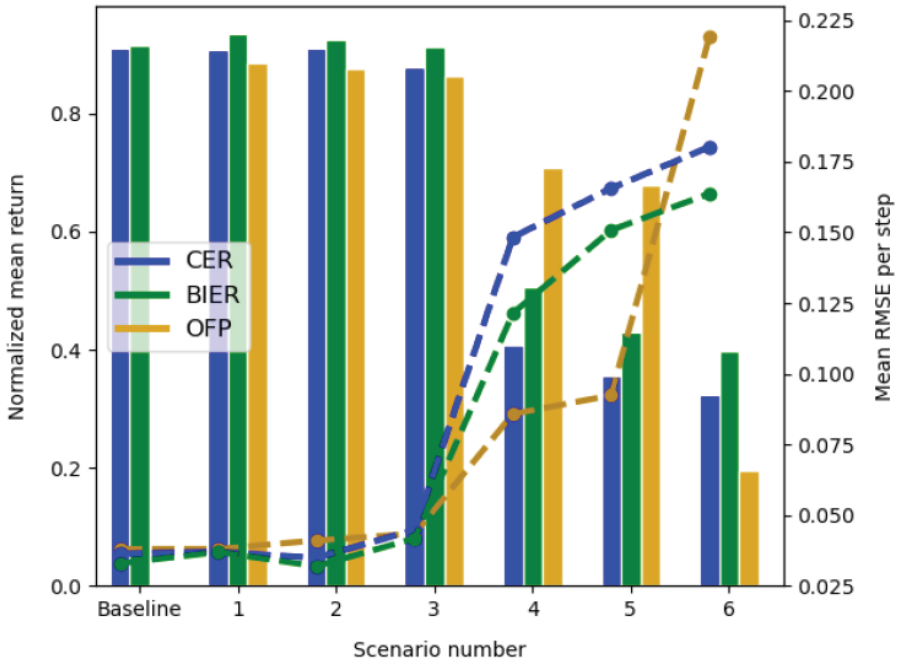
Real life evaluation:

- Same task.
- 2 scenarios of varying complexity to increase distribution shift.
- Comparison to an optimal fixed model-based version (OFP) of the controller.

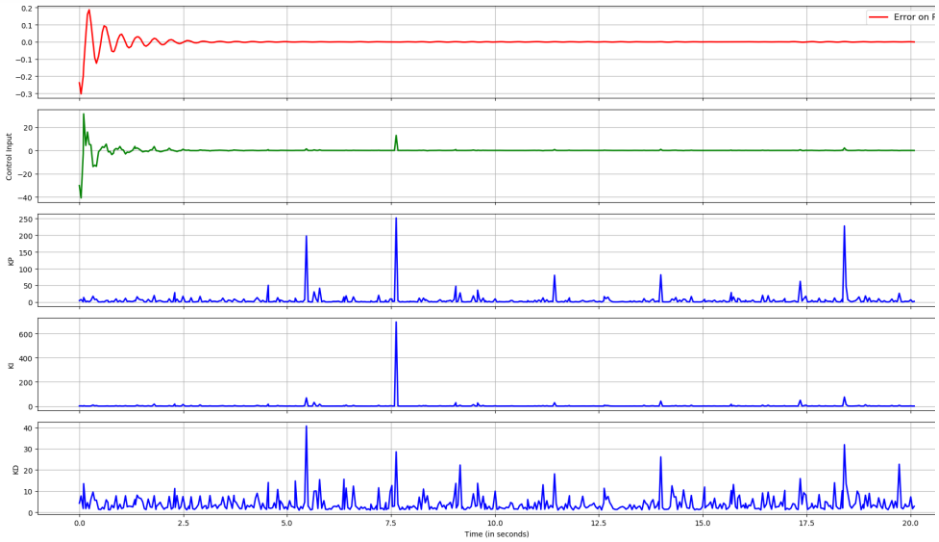
Simulated evaluation

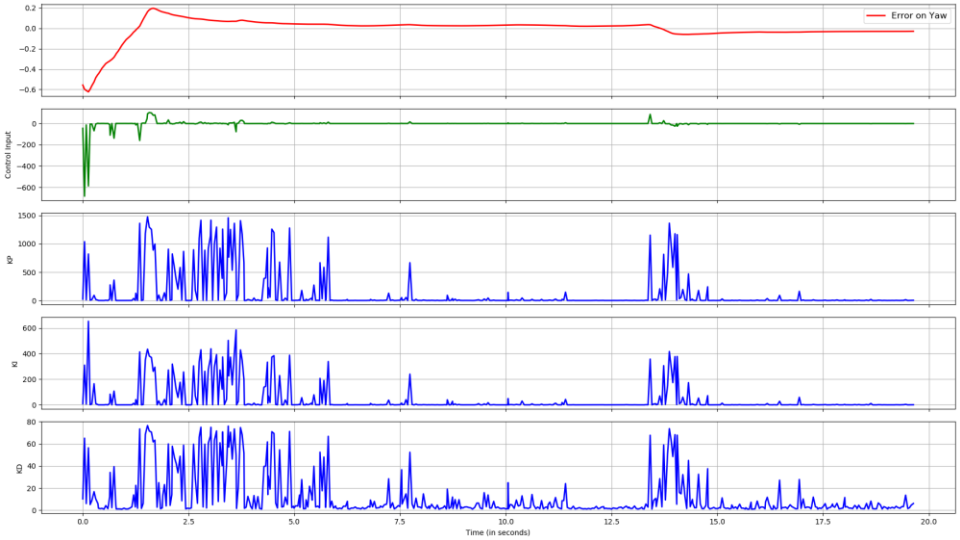
EVALUATION RESULTS FOR THE CER METHOD (LEFT) AND THE PROPOSED BIER METHOD (RIGHT).

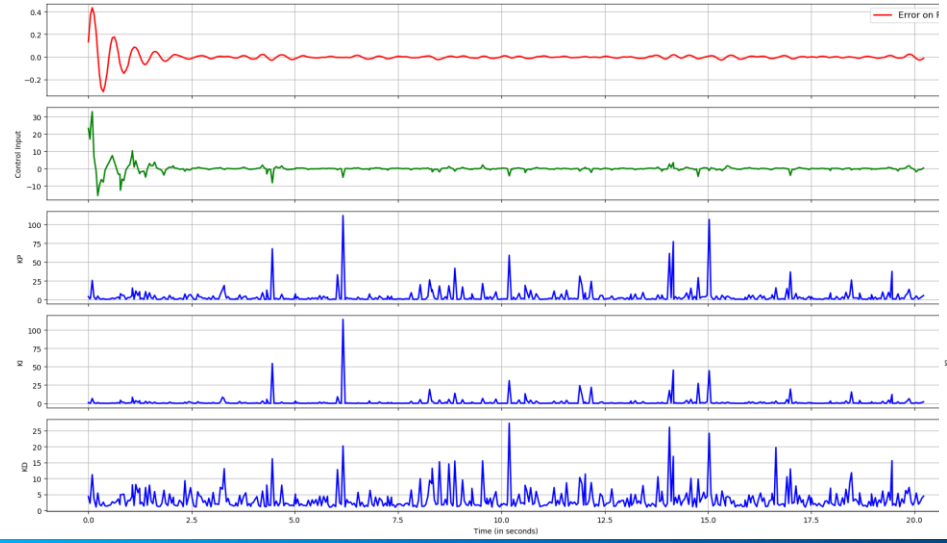
Scenario	Mean RMSE per step	Normalized mean return	Mean RMSE per step	Normalized mean return
Baseline	0.0364	0.9104 ± 0.0461	**0.0330**	**0.9219 ± 0.0250**
1	0.0370	0.9072 ± 0.0309	**0.0366**	**0.9347 ± 0.0262**
2	0.0350	0.9108 ± 0.0456	**0.0320**	**0.9244 ± 0.0240**
3	0.0448	0.8774 ± 0.0416	**0.0418**	**0.9124 ± 0.0266**
4	0.1483	0.4078 ± 0.2965	**0.1214**	**0.5071 ± 0.2530**
5	0.1656	0.3556 ± 0.2846	**0.1508**	**0.4289 ± 0.2573**
6	0.1802	0.3238 ± 0.2219	**0.1637**	**0.3966 ± 0.2167**

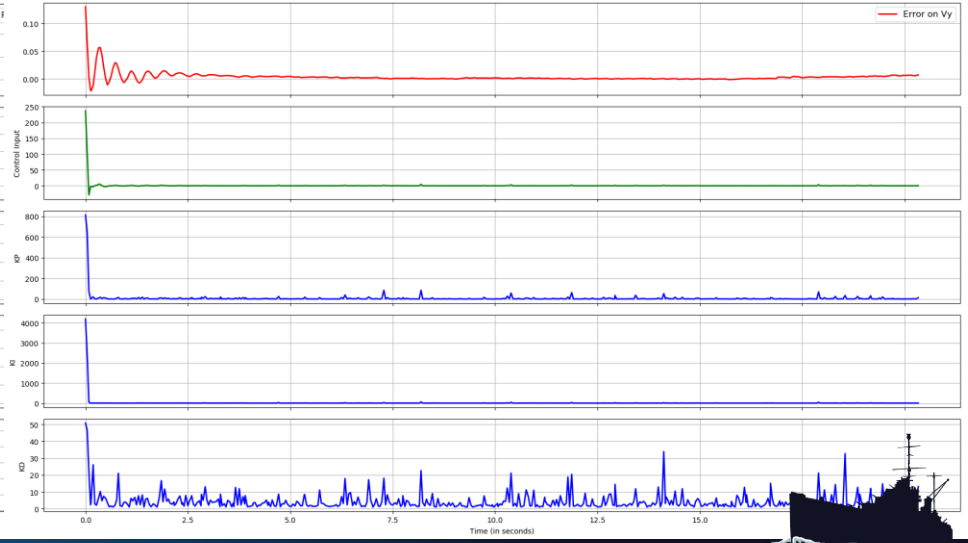


Simulated evaluation

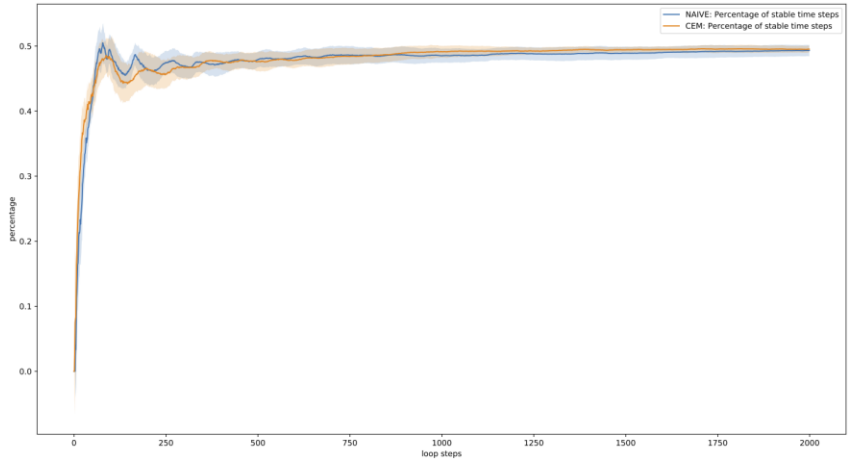


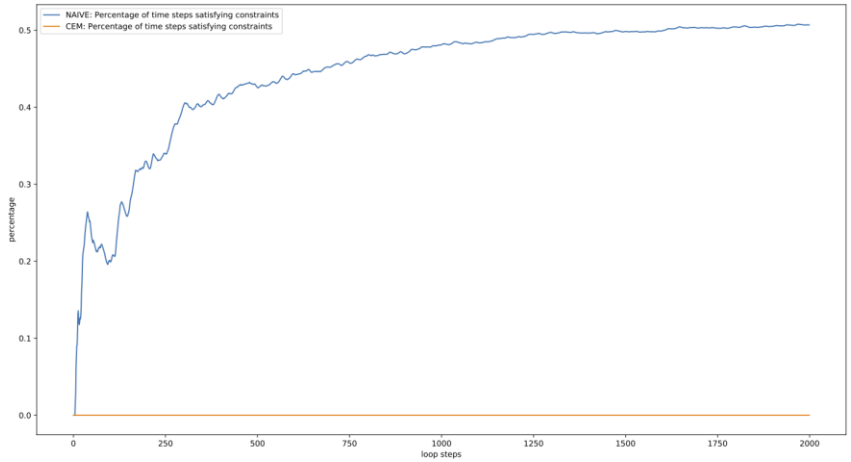


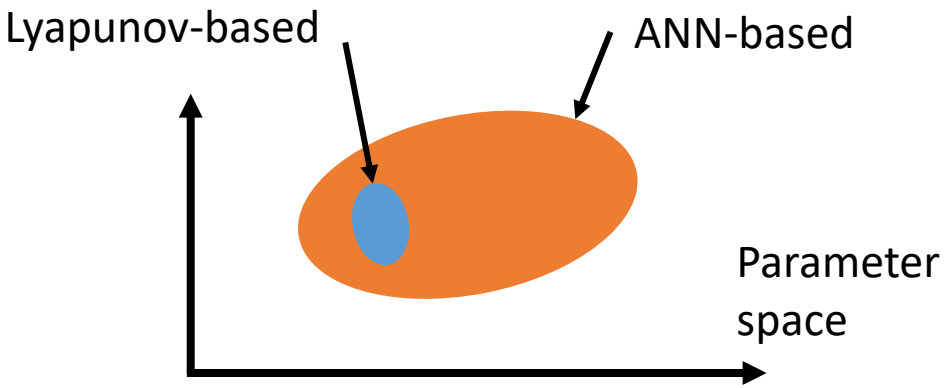




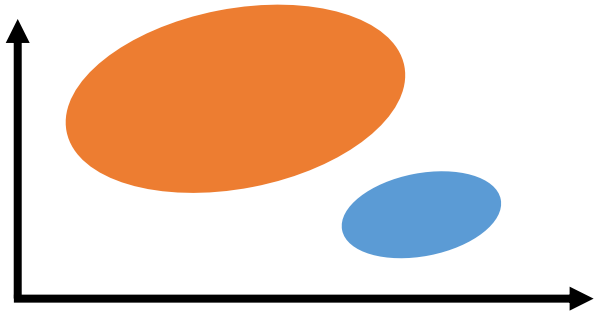
Assessing controller stability







OR



?

**PID Tuning using Cross-Entropy Deep Learning: a Lyapunov Stability Analysis,* By Hector Kohler, Benoit Clement, Thomas Chaffre, Gilles Le Chenadec, in IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles, 2022.

Experimental outcomes

- The control performance of the LB controller is between 2 and 2.4 times greater.
- The variance of the LB controller is between 1.48 and 3.1 times better smaller.
- The principal cause of failures is drift in yaw.
- The worst DoF is depth due to estimation error.

In practice

The entropy is added in the State-Value function:

$$V(s_t) = \mathbb{E}\left[Q(s_t, a_t) + \alpha \boldsymbol{\mathcal{H}}\left(\pi_\mu(.|s_t)\right)\right]$$

$$= \mathbb{E}\left[Q(s_t, a_t) - \alpha \log \pi_\mu(a_t|s_t)\right]$$

The target policy distribution is the exponential of the Q-Value distribution:

$$J_\mu(\pi) = D_{KL}\left(\pi_\mu(.|s_t) \,\middle|\middle|\, \frac{\exp(\min[Q_1(s_t,a_t), Q_2(s_t,a_t)])}{Z}\right).$$

Maximum entropy RL with automatic temperature adjustment

$$J_{maxEntrop}(\pi|s_0) = \max_\pi \mathbb{E}_\pi[\textstyle\sum_t r(s_t, a_t) + \alpha \boldsymbol{\mathcal{H}}\big(\pi(\cdot|s_t)\big)|s_0].$$

$$J_{autoEntrop}(\alpha) = \mathbb{E}_{s_t \sim D, a_t \sim \pi}[-\alpha \log \pi(\cdot|s_t) - \alpha \boldsymbol{\mathcal{H}}]^1$$

[1] *In practice , in order to make sure that α is non-negative we parametrize $\alpha_t = \exp\{\beta_t\}$ and we optimize β_t instead.*

Loss functions

Unbiased estimator of the Policy gradient:

$$\widehat{\nabla}_{\mu} J_{\mu}(\pi) = \nabla_{\mu} \log \pi_{\mu}(a_t|s_t) + \tag{J}$$

$$(\nabla_{a_t} \log \pi_{\mu}(a_t|s_t) - \nabla_{a_t} \min[Q_1(s,a), Q_2(s,a)]) \nabla_{\mu} f_{\mu}(\epsilon_t; s_t)$$

TD learning for the Critics:

$$J(V) = V(s_t) - (\min[Q_1(s_t, a_t), Q_2(s_t, a_t)] - \log \pi_{\mu}(\cdot\,|s_t))$$

$$J(Q) = Q(s_t, a_t) - \left(r_t + \gamma \times V(s_{t+1})\right)$$

The soft Q-update guarantees : $Q^{new} > Q^{old}$.[1]

And repeated policy updates **(J)** guarantees : $\pi \rightarrow \pi^*$.[1]

[1]Mathematical proofs are in appendix B of "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", T. Haarnoja et al.

Exploration strategy

Discrete environment/action spaces:

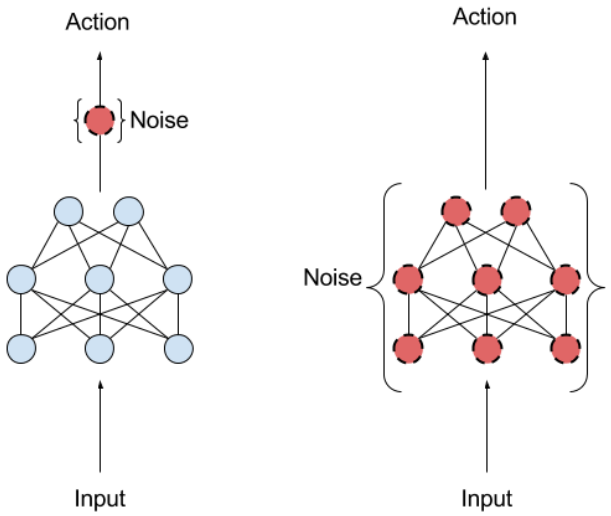
Epsilon-greedy, Thompson sampling, Upper Confidence Bounds, Boltzmann exploration…

Continuous environment/action spaces:

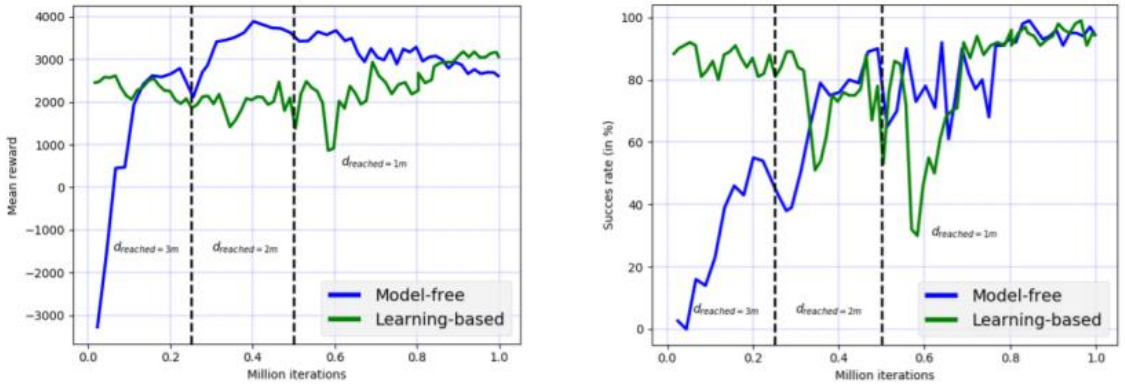
~~Epsilon-greedy, Thompson sampling, Upper Confidence Bounds, Boltzmann exploration…~~

Passive exploration → noise-based exploration (e.g., entropy):

Parameter space noise for Exploration
(M. Plappert et al., OpenAI)



Why not doing end-to-end RL?



(a) Mean reward (b) Success rate

Fig. 5: Training curves showing the mean reward and success rate computed per 100 episodes over a moving window of 100 episodes. Note that with SAC, the policy is trained to maximize also the entropy, therefore the mean action does not always correspond to the optimal action for the maximum return objective.

Controller type	Mean step number	Mean total reward	Mean reward per step	Success rate	Positive reward rate
Fixed Poles PID	488	710.797	1.454	50.6%	61.197%
Model-free	357	2238.970	6.256	74.2%	86.056%
Learning-based	**281**	**4034.434**	**14.346**	**91.6%**	**89.2%**

Table 1: Evaluation results.

Learning-based vs Model-free Adaptive Control of a MAV under Wind Gust (T. Chaffre et al, Book Chapter, LNEE by SPRINGER 2021)

Classification of RL methods

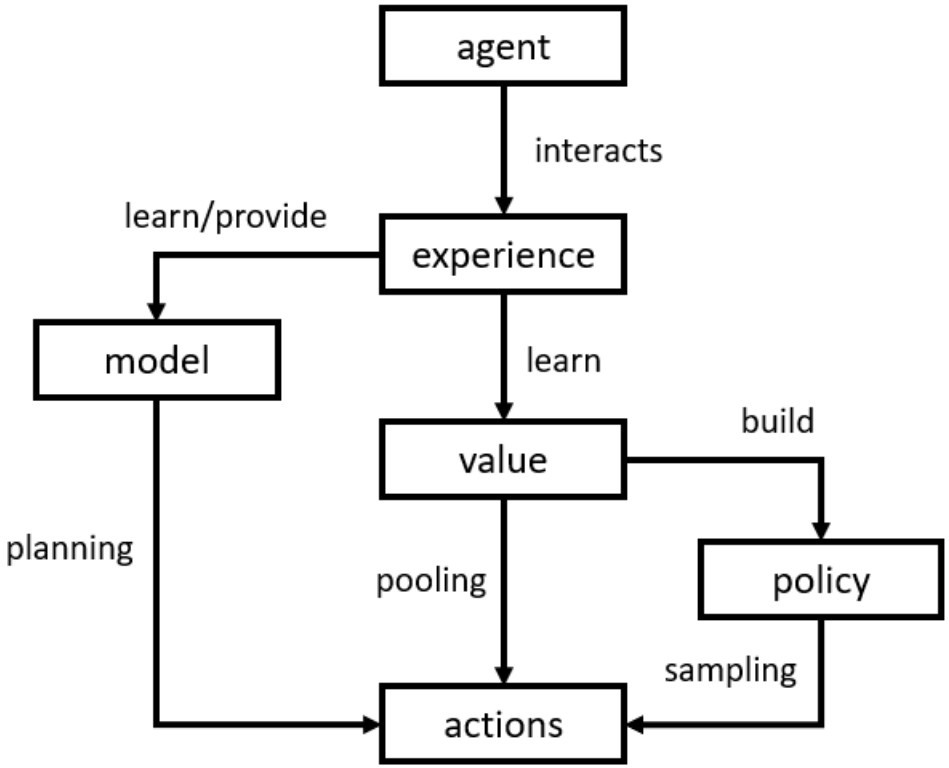
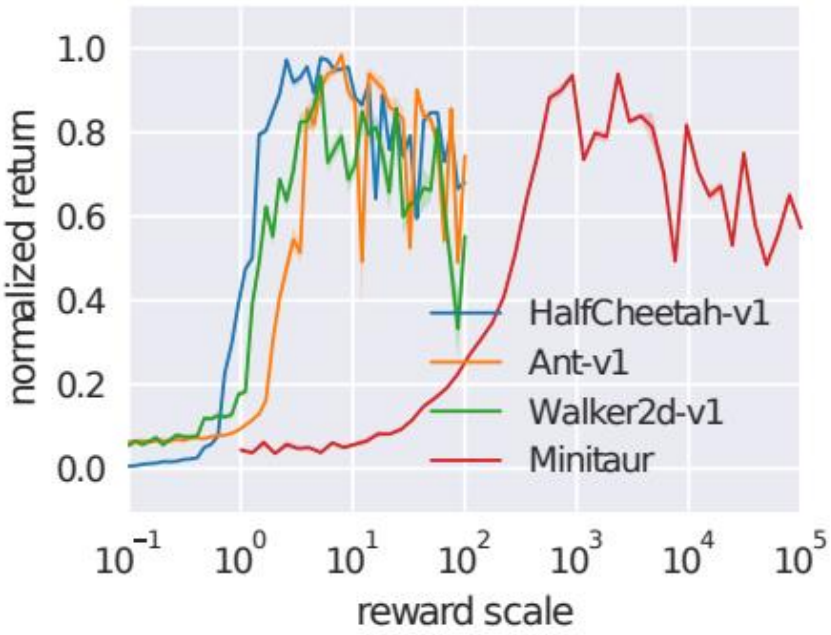


Illustration of RL methods based on the nature of the decision making process. The difference between the solutions methods is captured by how the actions are determined: in model based methods, the actions are the result of deterministic planning, in value based methods the actions are pooled over the entire set of possible actions, and in policy gradient methods the actions are sampled from a probability density function

SAC Limits



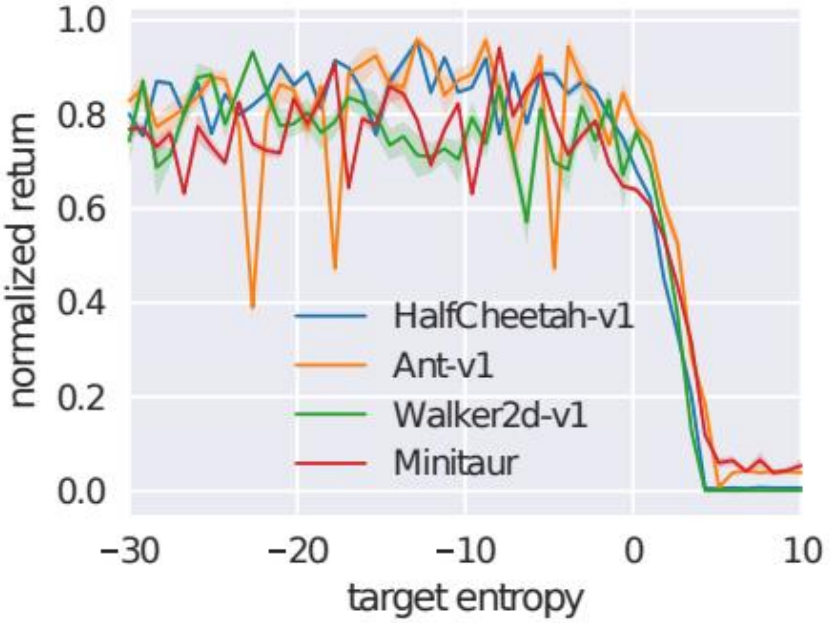


Illustration of performance with fixed (left) and automatic temperature parameter (right).

Moving target values