



5<sup>th</sup> Joint International Symposium on  
Deformation Monitoring

20 -22 June 2022

Polytechnic City of Innovation, Valencia, Spain

[jisdm2022.webs.upv.es](http://jisdm2022.webs.upv.es)

# Adaptive spatial discretization using reinforcement learning

Jemil Avers Butt<sup>1,2</sup>, Andreas Wieser<sup>1</sup>

<sup>1</sup>ETH Zurich, Institute of Geodesy and Photogrammetry, Zurich; <sup>2</sup>Atlas optimization GmbH, Zurich

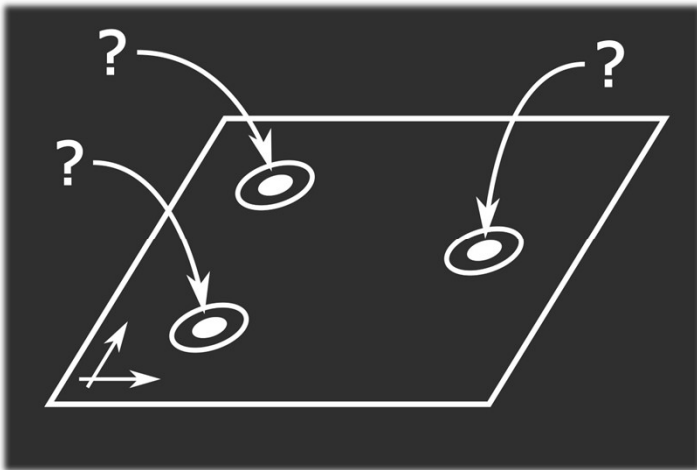
**ETH** zürich



# Content

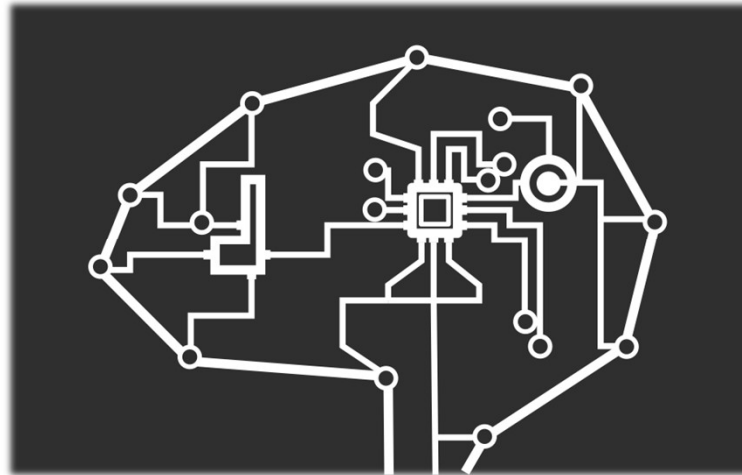
## 1. Problem

Spatial discretization



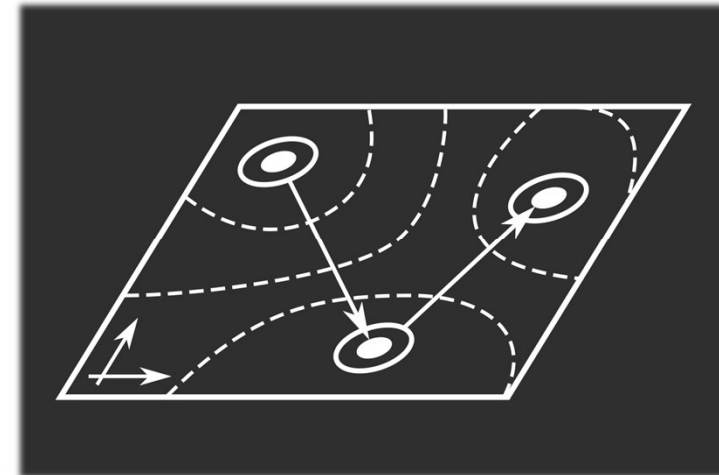
## 2. Method

Reinforcement learning

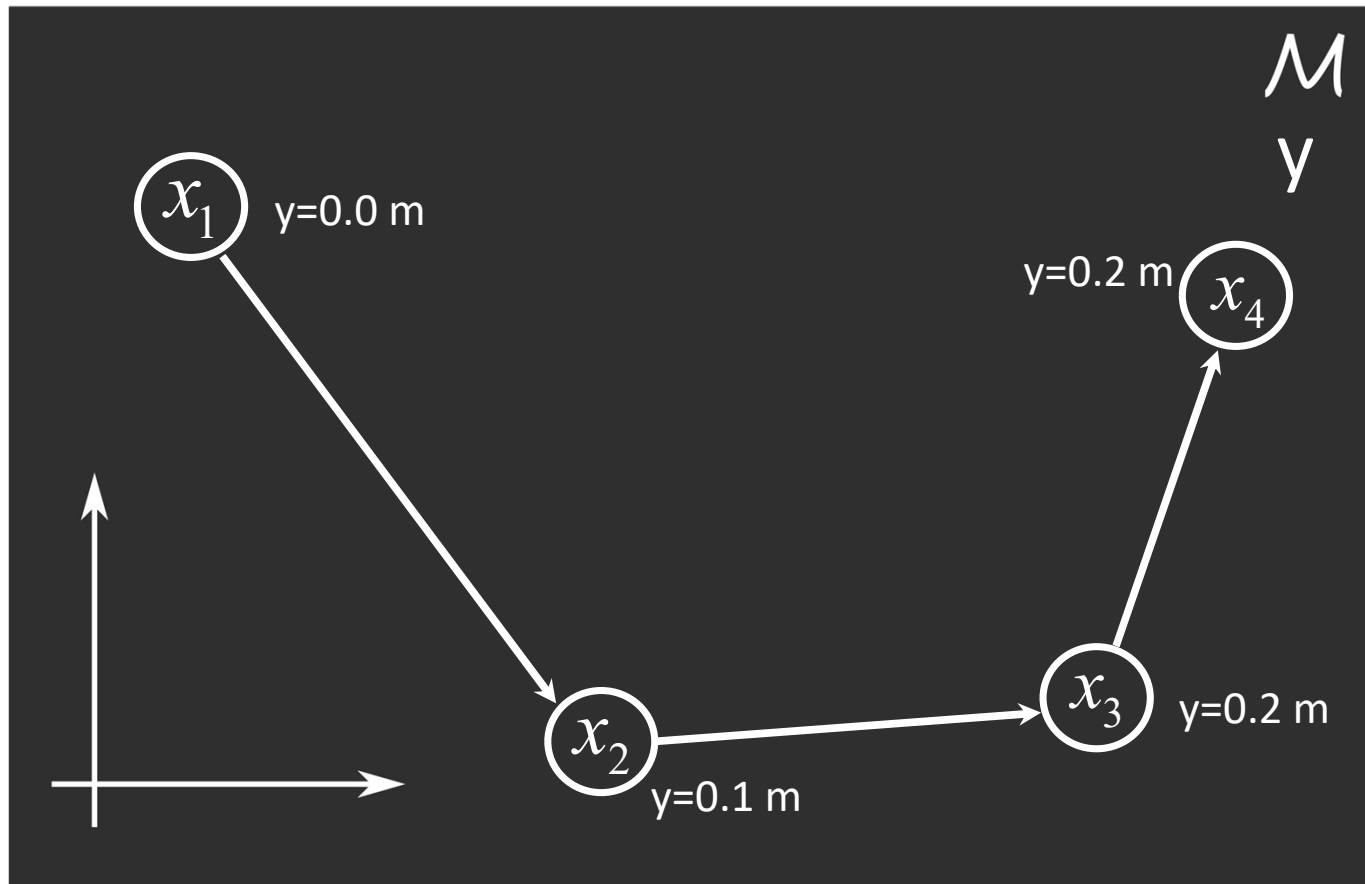
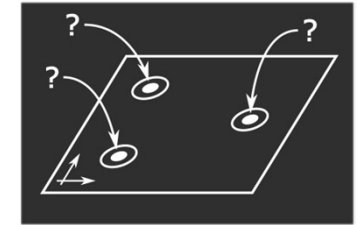


## 3. Results

Better sampling



# Problem - Specification



## Spatial discretization

- Object  $M$
- Interested in  $y$
- Sequence of  $x_1, x_2, \dots$

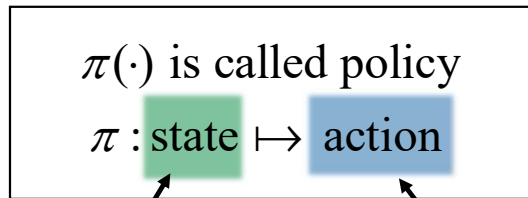
## Questions

- Best sequence?
- System knowledge?
- Adapt to observations?

## Answer

- We need a policy  $\pi$  !

# Problem – Hierarchies of Solutions

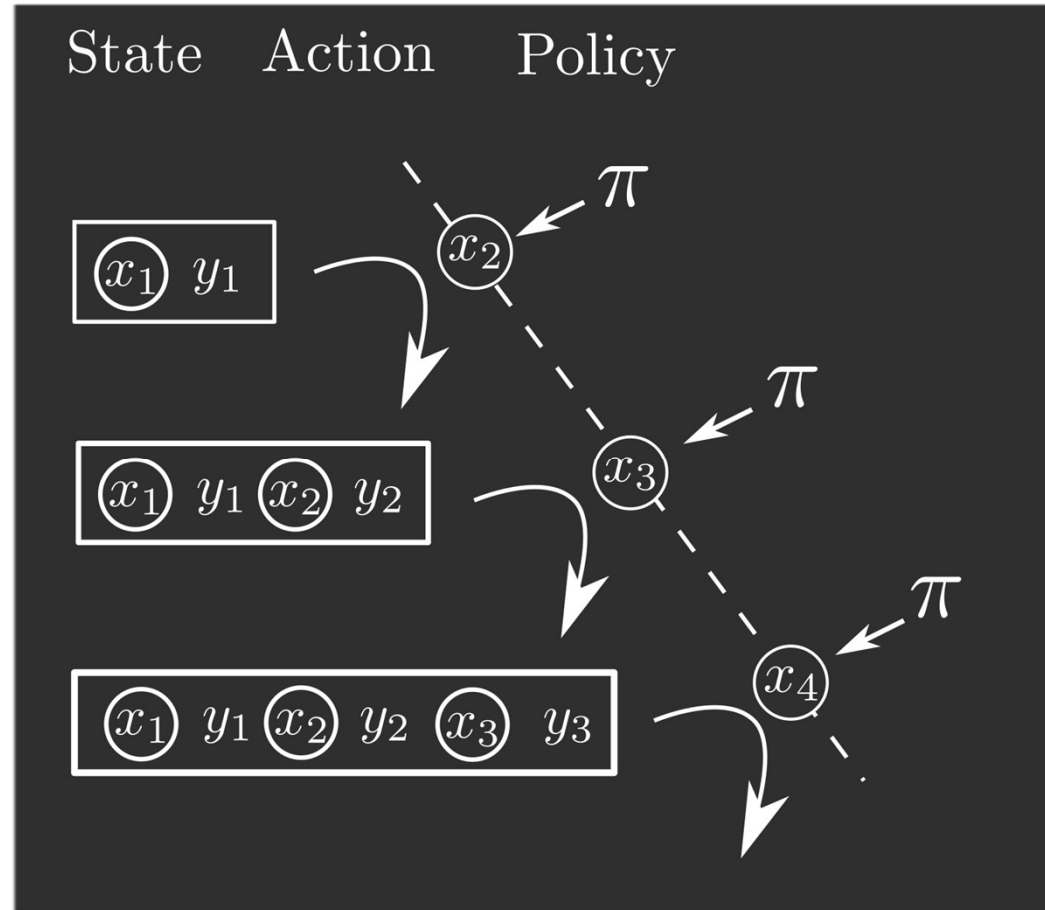


What we know

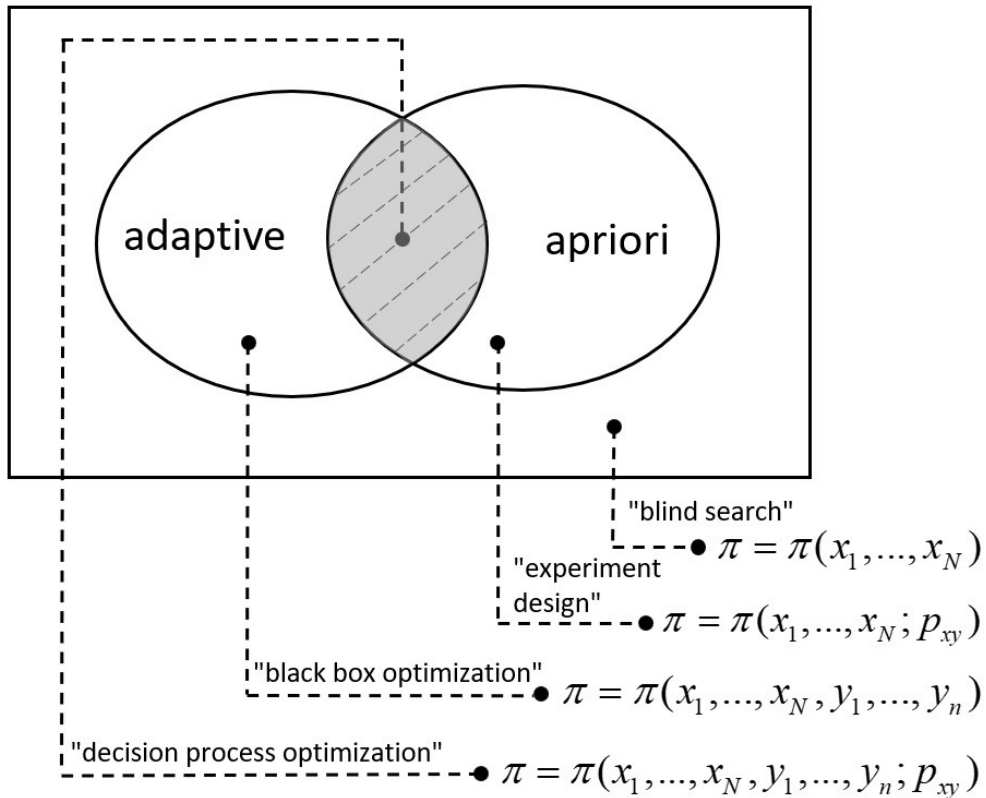
Next measurement

We want

- Adaptivity  
 Next sample locations influenced by previous findings  
 $\rightarrow \pi = f(x_1, \dots, x_N, y_1, \dots, y_N)$
- Prior knowledge  
 Sampling scheme makes use of dependences between  $x, y$   
 $\rightarrow \pi = f(x_1, \dots, x_N; p_{xy})$



# Problem – State of the art



"Blind search" :

Random numbers, pseudorandom, grids, ...

[Kuipers & Niederreiter]

"Experiment design" :

Minimization of  $\text{tr}(\Sigma)$ ,  $\text{det}(\Sigma)$ , Entropy, ...

- > Geodetic networks, geostatistical sampling

[Grafarend & Sanso], [Angulo et al]

"Black box optimization" :

Sequences of samples based assumptions.

-> Problems with unknown structure

[Fu], [Alarie et al]

"Decision process optimization" :

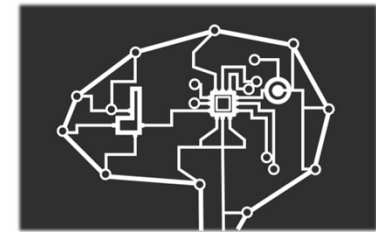
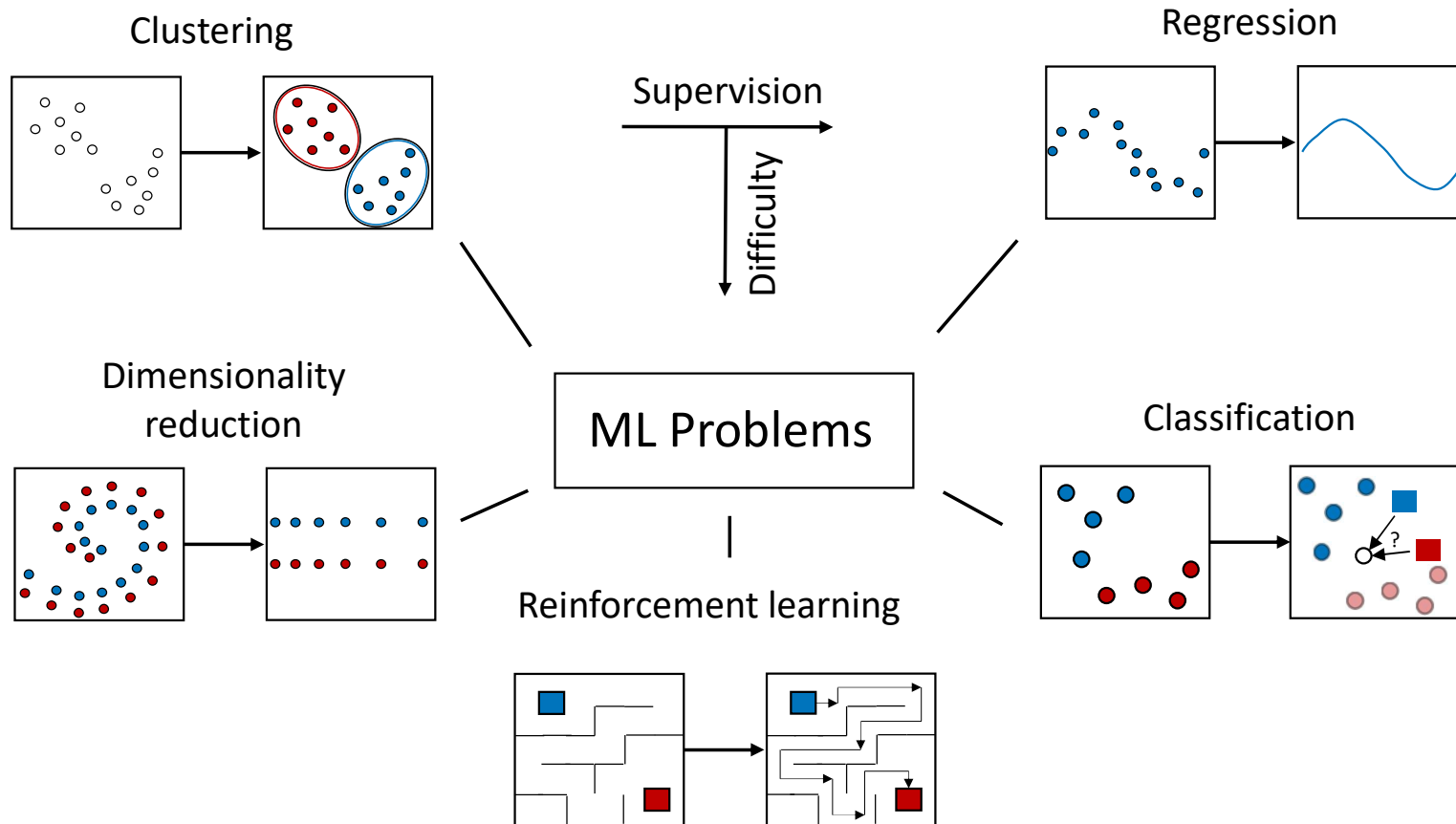
Adaptive policies to maximize reward

-> Optimal control in economy and games

[Sutton & Barto],

[Feinberg & Shwartz]

# Method – RL in ML



RL:

- Optimal control
- No right/wrong
- Learn from experience
- Agent interacts with environment
- Difficult task
- Computationally intense

# Method – RL Policy gradients

Policy function is ANN

- Determines decisions
- Make decisions optimal

1. Expected return depends on params

$$\eta(\pi) = E_{\pi p} \left[ \sum_{k=1}^{n-1} \gamma^k r(s_k, a_k) \right]$$

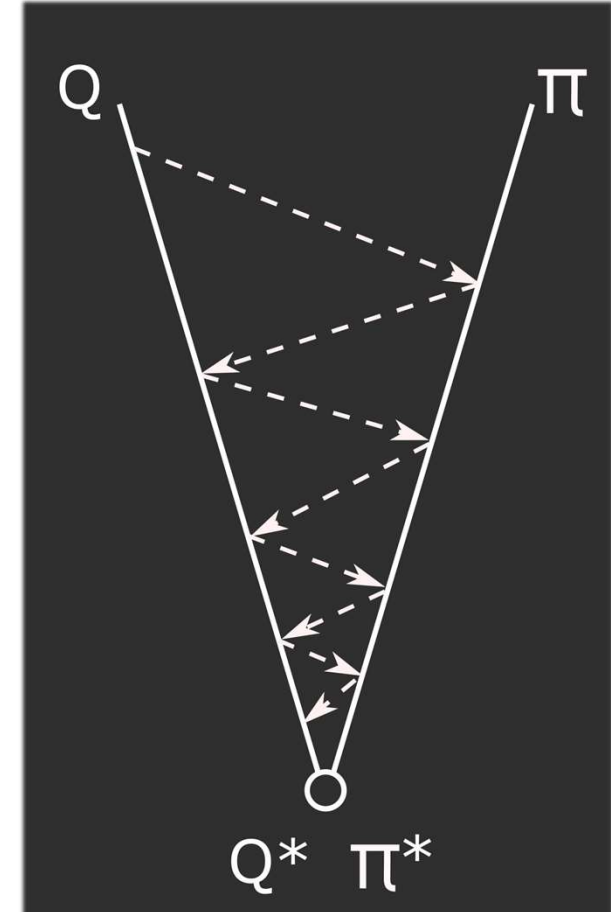
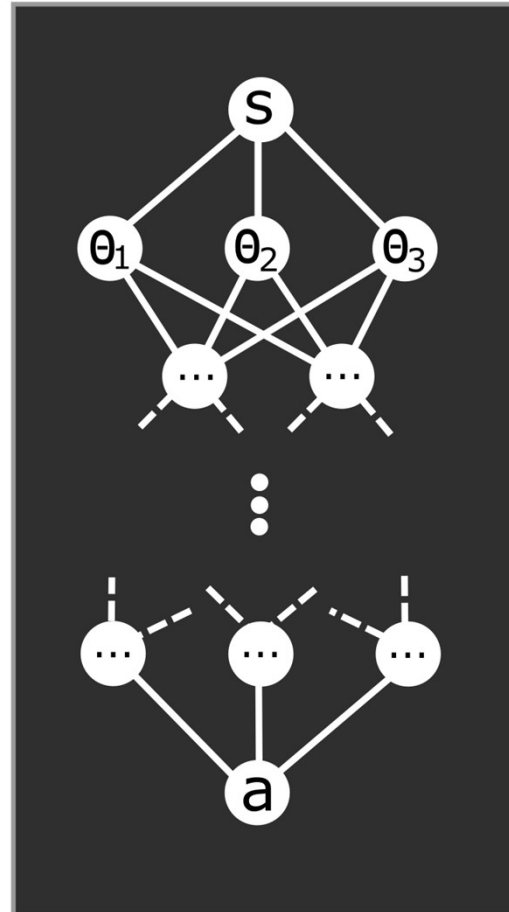
$$\eta(\pi) = \int_{(S \times A)^{n+1}} \sum_{k=0}^{n-1} \gamma^k r(s_k, a_k) p_{\theta}(\xi) d\xi$$

2. Gradient depends on observations

$$\nabla_{\theta} \eta = \int_{(S \times A)^{n+1}} \sum_{k=0}^{n-1} \gamma^k r(s_k, a_k) \nabla_{\theta} [\log p_{\theta}(\xi)] p_{\theta}(\xi) d\xi$$

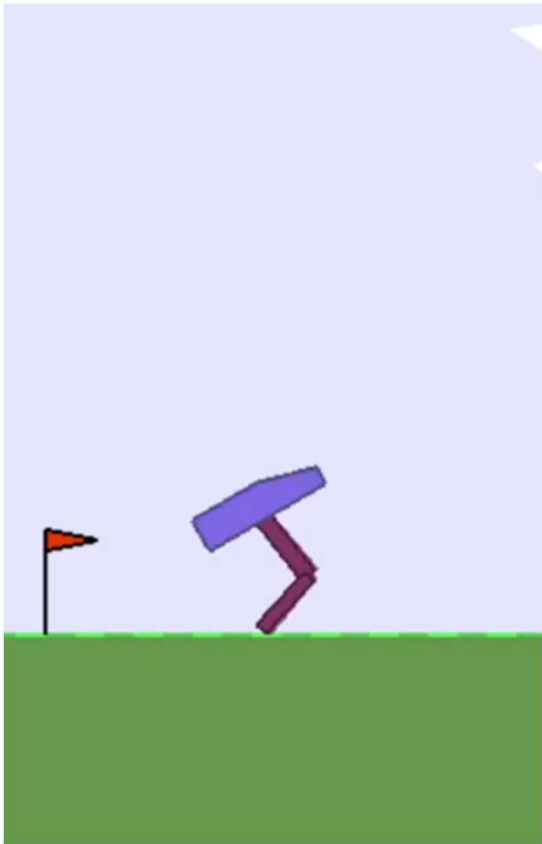
$$\hat{\nabla}_{\theta} \eta = \frac{1}{m} \sum_{j=1}^m \left( \sum_{k=0}^{n-1} \gamma^k r(s_k^j, a_k^j) \right) \nabla_{\theta} [\log p_{\theta}(\xi_j)]$$

$$= \frac{1}{m} \sum_{j=1}^m \left( \sum_{k=0}^{n-1} \gamma^k r(s_k^j, a_k^j) \right) \sum_{i=0}^{n-1} \nabla_{\theta} \log \pi_{\theta}(s_i^j, a_i^j).$$

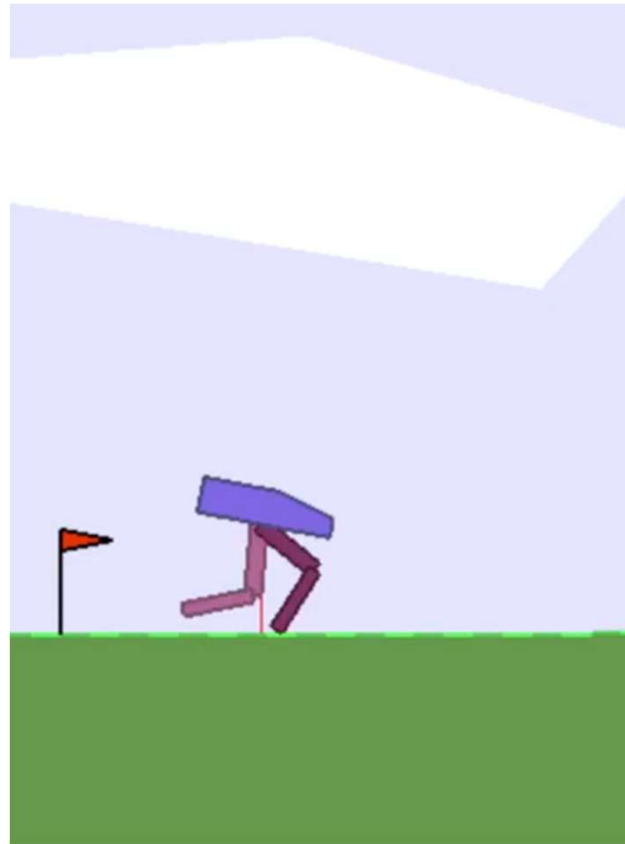


# Method – Showcase

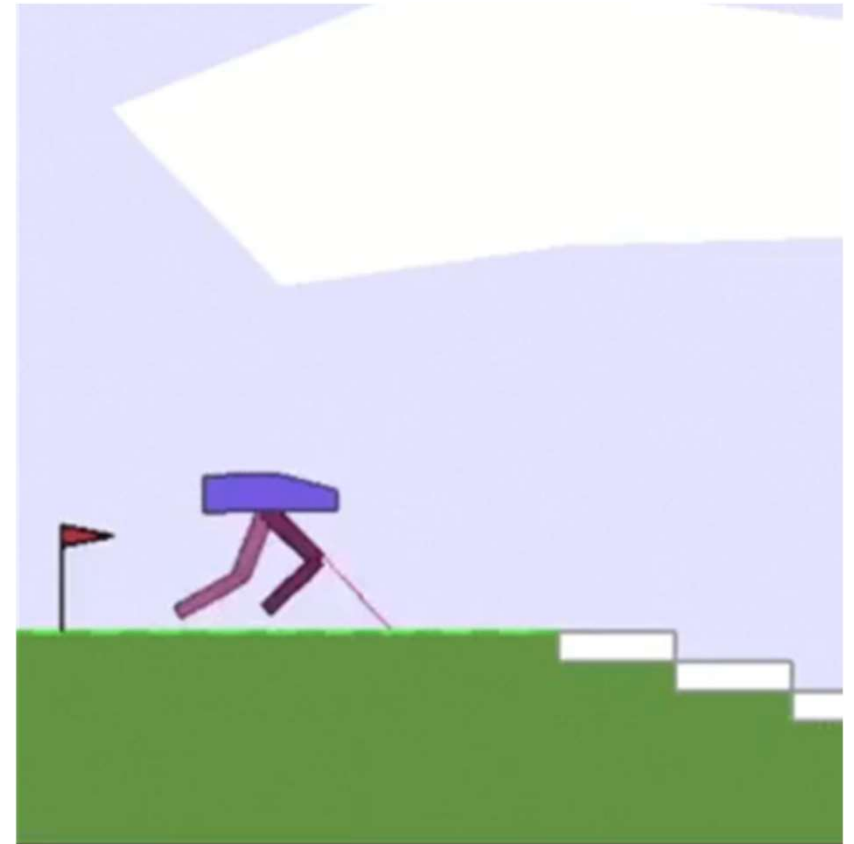
Training: 3 min



Training: 3 h



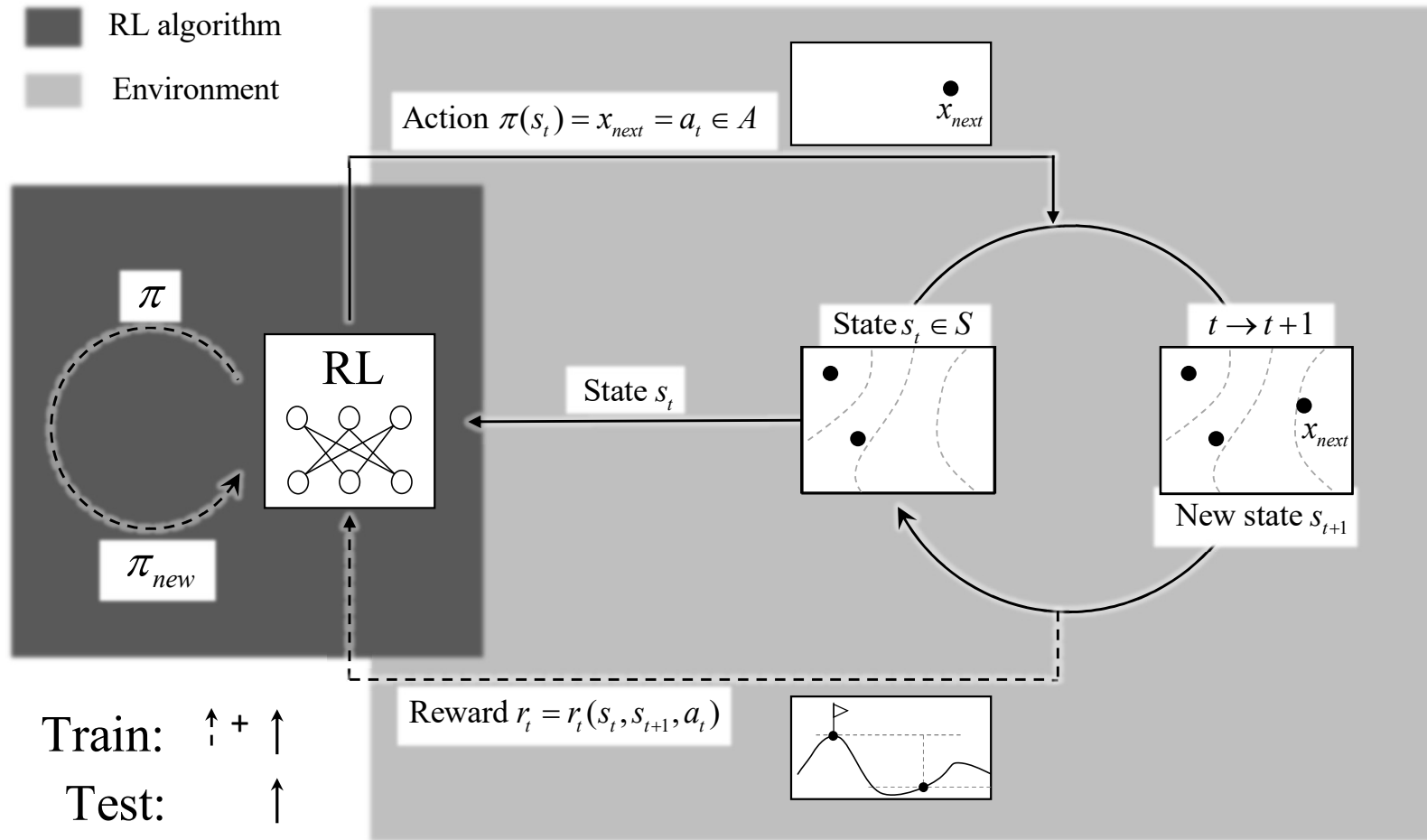
Training: 3 d



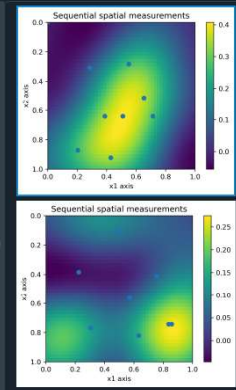
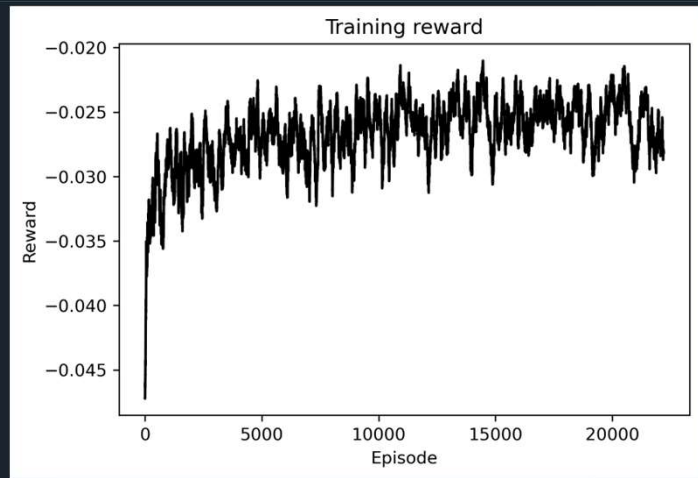
Trained with a3c by David Griffiths  
[https://github.com/dgriff777/a3c\\_continuous](https://github.com/dgriff777/a3c_continuous)



# Method – Spatial discretization as RL



```
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/jemil/Desktop/Programming/Python/Atlas_Optimization/Optimal_Monitoring/Compilation_Paper/benchmark_random_def_2D.py
random_def_1D.py | Playing_games_Lunar_lander_discrete.py | Playing_games_bipedal_walker.py | benchmark_random_def_2D.py
1 """
2 The goal of this script is to train a TD3 RL algorithm on the random deformation
3 task and compare the cumulative rewards to the ones gathered by alternative
4 discretization strategies.
5 For this, do the following
6 1. Definitions and imports
7 2. Train with stable baselines
8 3. Apply alternative methods
9 4. Summarize and plot results
10 """
11
12 """
13 1. Definitions and imports
14 """
15 # i) Import basics and custom environment
16
17 import numpy as np
18 import time
19 from scipy.optimize import basinhopping
20 import class_random_def_2D_env as def_2D
21
22 # ii) Import stable baselines
23 from stable_baselines3 import TD3
24 from stable_baselines3.common.env_checker import check_env
25
26 # iii) Initialize and check
27 np.random.seed(0)
28 def_2D_env = def_2D.Env()
29 def_2D_env.reset()
30 # check_env(def_2D_env)
31
32 """
33 2. Train with stable baselines
34 """
35 # i) Train a TD3 Model
36
37 # start_time=time.time()
38 # model = TD3("MlpPolicy", def_2D_env, verbose=1, seed=0)
39 # model.learn(total_timesteps=100000)
40 # end_time=time.time()
41
42 # model.save('./Saved_models/trained_benchmark_random_def_2D')
43 model = TD3.load('./Saved_models/trained_benchmark_random_def_2D')
44
45 """
46 3. Apply alternative methods
47 """
48 # Note: All actions are in [-1,1]x[-1,1] and get mapped to [0,1]x[0,1] by
49 # the environment, translating input actions from the symmetric box space
50 # [-1,1]x[-1,1] to indices
51
52 # i) Grid based sampling
53
54 def grid_based_sampling(environment):
55     grid_x1 = np.kron(np.array([-1/3, 1/3, 1]), np.array([1, 1, 1]))
56     grid_x2 = np.kron(np.array([1, 1, 1]), np.array([-1/3, 1/3, 1]))
57     grid = np.vstack((grid_x1, grid_x2))
58     action = grid[:, environment.epoch]
59     return action
60
61 # ii) Pseudo random sampling
62
63 def pseudo_random_sampling(environment):
64     halton_sequence = np.array([[1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, 1/16, 9/17,
```



```
Console I/O
[ 0.07423399]
[ 0.36265202]
[ 0.28539754]
[ 0.35397827]
[-0.02127369]
Reward is -0.0012474026886026413
Measured locations are [[0.20408163 0.8719487]
[0.55102041 0.28205128]
[0.3877551 0.64102564]
[0.51020408 0.64102564]
[0.28571429 0.30769231]
[0.65306122 0.51282051]
[0.71428571 0.64102564]
[0.42857143 0.92307692]
[0. 0. ]]
Measurements are [[ 0.27411945]
[ 0.27618975]
[ 0.35367949]
[ 0.40609153]
[ 0.07423399]
[ 0.36265202]
[ 0.28539754]
[ 0.35397827]
[-0.02127369]]
Reward means of different methods
[-0.04441617 -0.05062521 -0.08014935 -0.03963356 -0.04539691 -0.03688655]
Reward standard deviations of different methods
[0.04121231 0.03924319 0.06256735 0.03201321 0.03924964 0.0434375 ]

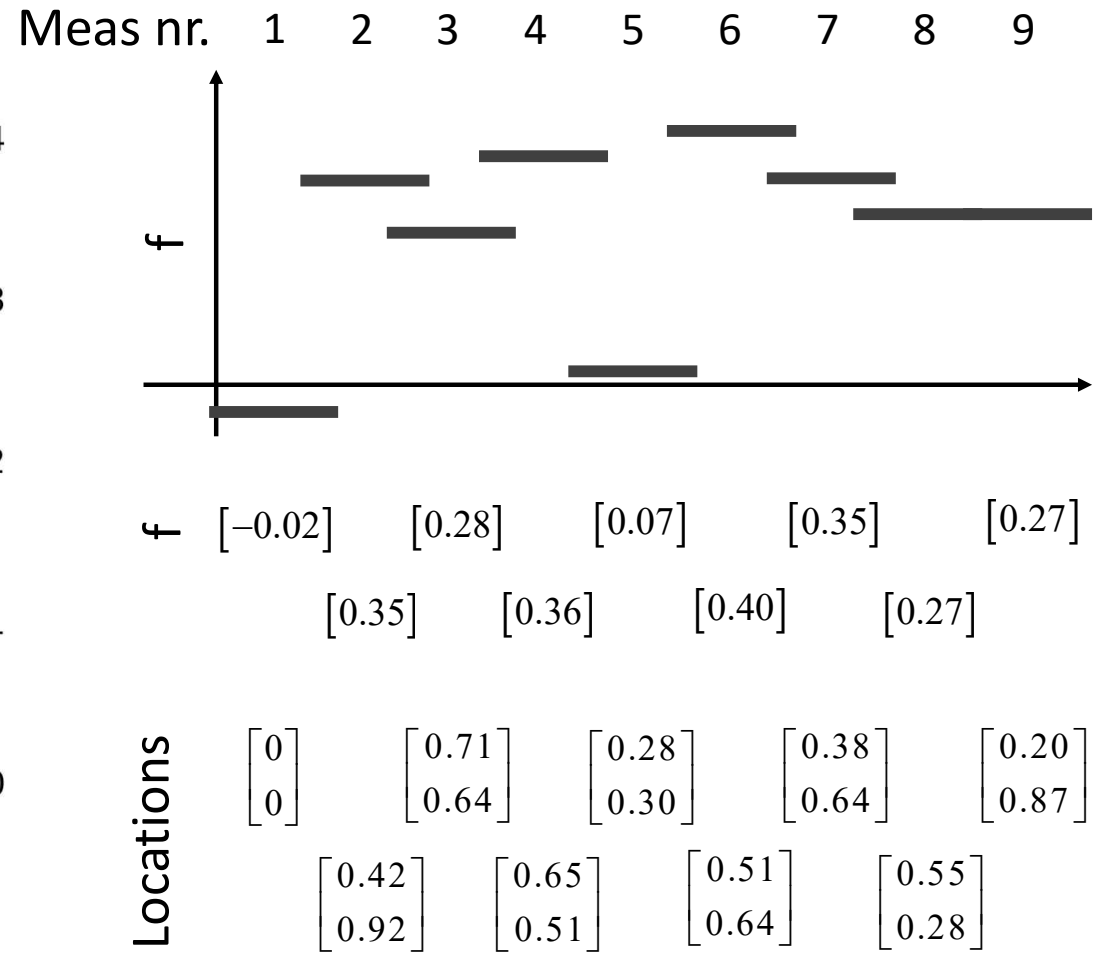
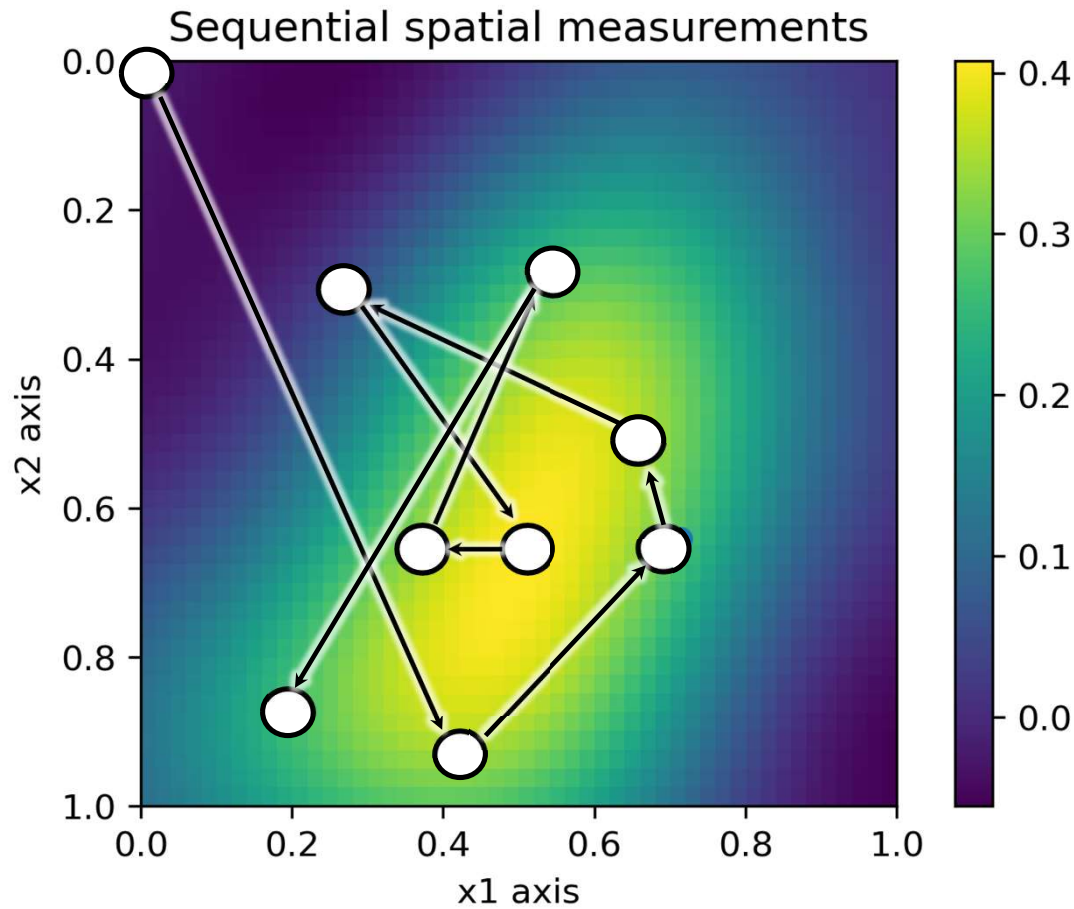
Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

In [6]: n_episodes=1
.....: for k in range(n_episodes):
.....:     done=False
```

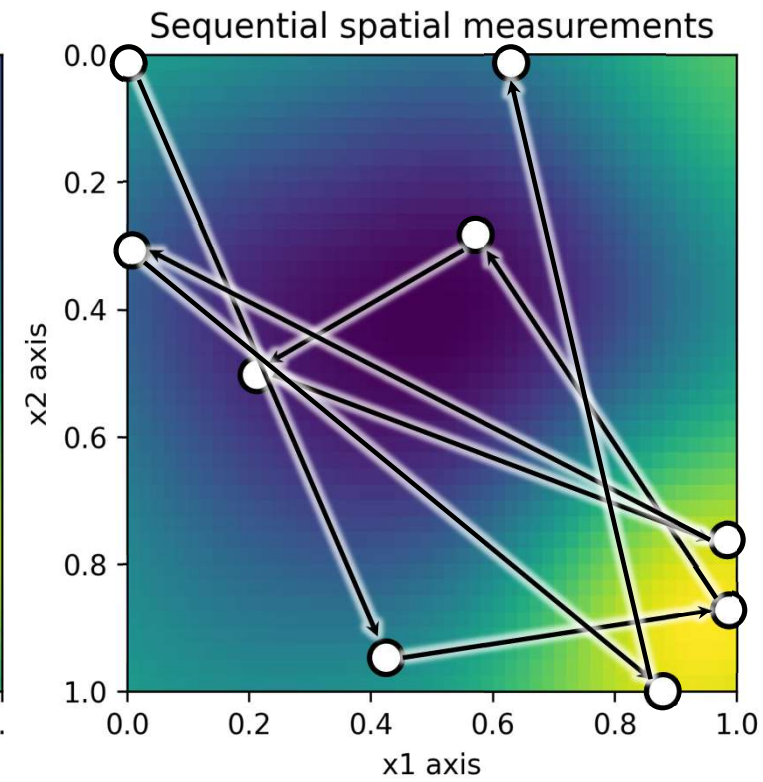
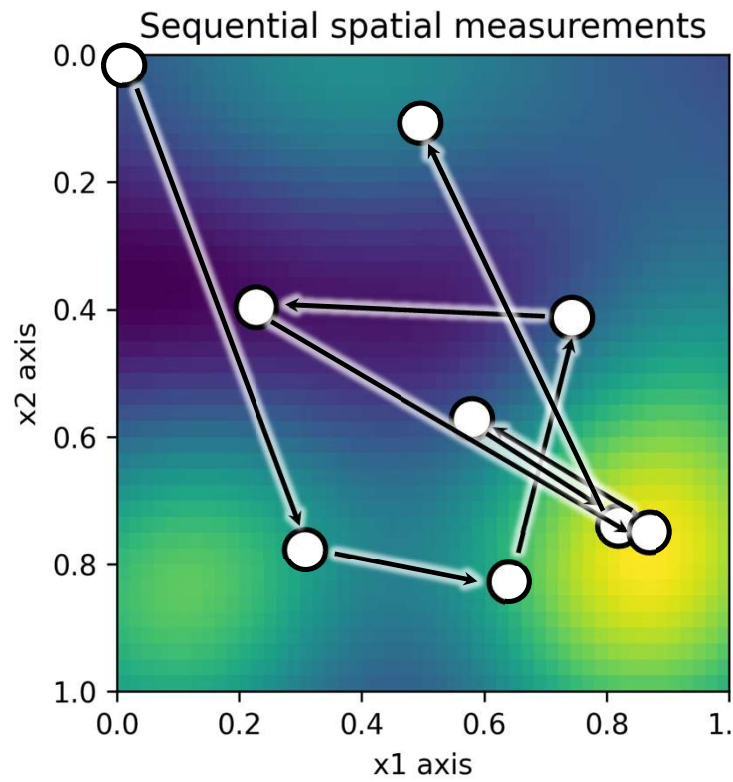
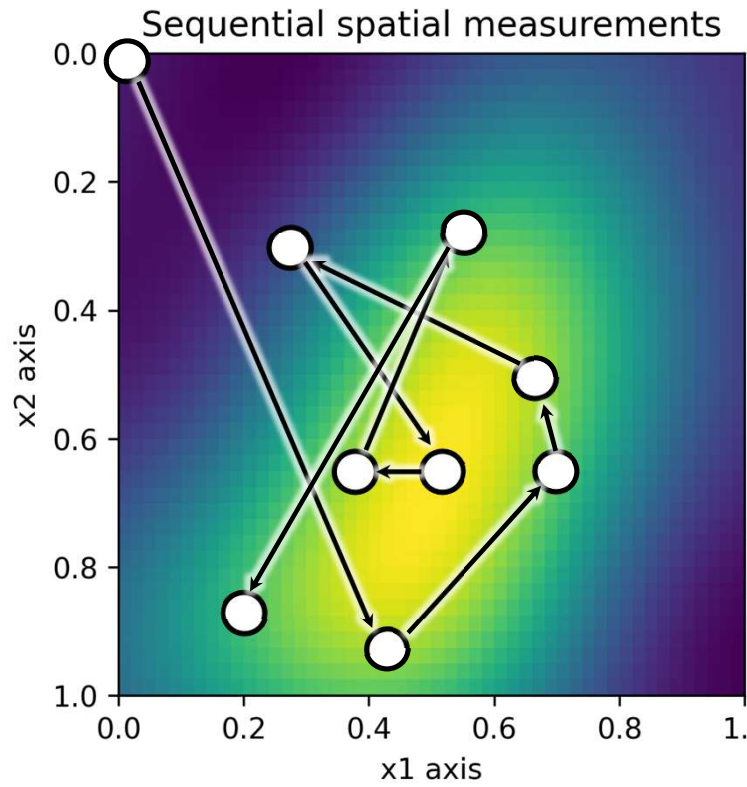
Stable-Baselines3: Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and N. Dorman (2019). Stable Baselines 3. GitHub repository: <https://github.com/DLR-RM/stable-baselines3>

[https://github.com/jemil-butt/Optimal\\_Discretization\\_RL](https://github.com/jemil-butt/Optimal_Discretization_RL)

# Results – Illustration

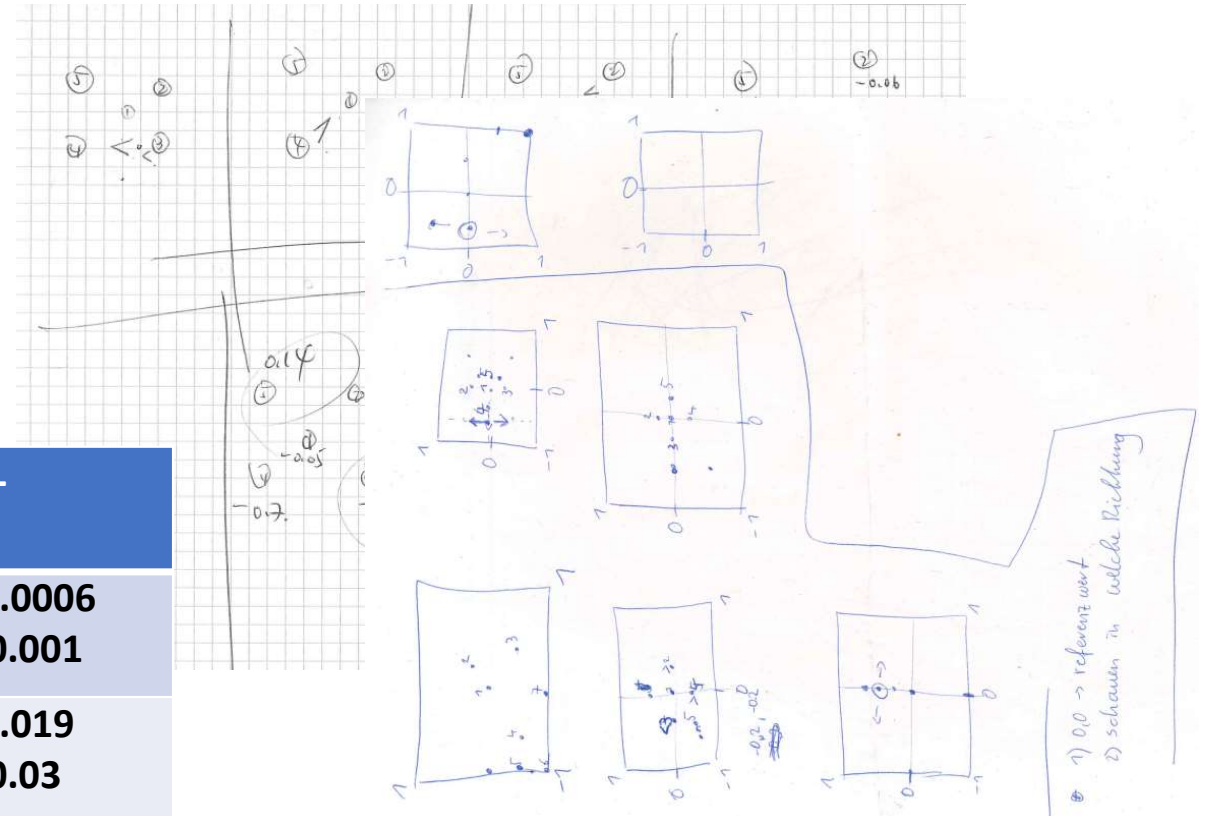


# Results – Illustration



# Results – Comparison

Method/ Task	Random	Quadratur e	Exp. design	RL
beam bending	-0.05 $\pm 0.1$	-0.065 $\pm 0.07$	-0.0025 $\pm 0.005$	<b>-0.0006</b> <b><math>\pm 0.001</math></b>
1D def.	-0.07 $\pm 0.08$	-0.047 $\pm 0.04$	-0.035 $\pm 0.04$	<b>-0.019</b> <b><math>\pm 0.03</math></b>
Def. tracking	-0.28 $\pm 0.19$	N. A	-0.24 $\pm 0.14$	<b>-0.067</b> <b><math>\pm 0.038</math></b>
2D def.	-0.08 $\pm 0.063$	-0.04 $\pm 0.032$	-0.045 $\pm 0.039$	<b>-0.031</b> <b><math>\pm 0.043</math></b>



Human*	Human 1	Human 2	Human 3
2D def.	-0.078	-0.015	-0.026

# Results – Conclusion

**Problem:** Have some object or process. Want to know something.  
Decide where to measure.

**Solution:** Make model of process. Use it for simulations.  
RL gets optimal\* sequences of decisions.

## What's good!

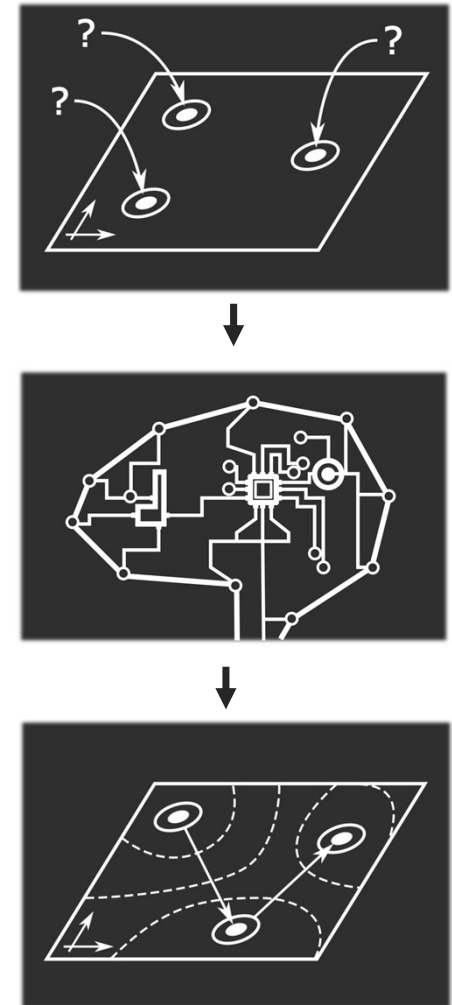
- Really flexible
- Optimal decisions under uncertainty
- Don't need explicit probabilistic models
- Modelling work done by computer
- Higher score than alternatives

## What's bad!

- No guarantees
- No measure of uncertainty
- Need model for simulation
- Training sloooow ...

## Applications!

- Adaptive monitoring
- Episodic repositioning



# Sources

- Ai, B., Jia, M., Xu, H., Xu, J., Wen, Z., Li, B. and D. Zhang (2021). Coverage path planning for maritime search and rescue using reinforcement learning, *Ocean Engineering*, Volume 241.
- Alarie, S., Audet, C., Gheribi, A.E., Kokkolaras, M., and S. Le Digabel. (2021). Two decades of blackbox optimization applications, *EURO Journal on Computational Optimization*, 9.
- Angulo, J.M., Ruiz-Medina, M.D., Alonso, F.J. and M.C. Bueso (2005). Generalized approaches to spatial sampling design. *Environmetrics*, Vol 16, No. 5.
- Feinberg, E. A. and A. Shwartz (2012). *Handbook of Markov Decision Processes. Methods and Applications*. Berlin Heidelberg (Springer Science & Business Media).
- Fu, M. C. *Handbook of Simulation Optimization*. Berlin, Heidelberg: Springer, 2014.
- Grafarend, Erik W. and F. Sansò (2012). *Optimization and Design of Geodetic Networks*. Berlin Heidelberg (Springer Science & Business Media).
- Sutton, R.S. and A.G. Barto (1992). *Reinforcement Learning*. Berlin Heidelberg (Springer Science & Business Media).
- Wiering, M. and M. Otterlo (2012). *Reinforcement Learning. State-of-the-Art*. Berlin Heidelberg (Springer Science & Business Media).
- Zuluaga, J. G. C., Leidig, J. P., Trefftz, C., and G. Wolffe, (2021). Deep Reinforcement Learning for Autonomous Search and Rescue, *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, pp. 521-524.

# Thank you for your time!

Now:  
Q & A

