A dramatic painting depicting a scene from Aztec culture. In the center, a priest in elaborate feathered headdress and traditional clothing stands on a raised stone platform, his arms raised in a gesture of offering or sacrifice. Below him, a massive crowd of people, mostly women, looks up in awe or participation. To the right, another figure in a detailed costume, possibly a soldier or a participant in a ritual, holds a large red banner with gold symbols. The background features a massive pyramid with multiple levels and steps, silhouetted against a bright sky. The overall atmosphere is one of a significant religious or cultural event.

algorithms for robotics

Michal CHOVANEC, PhD.

overview

- noise reduction : digital filters, Kalman filter
- control : PID, LQR, MPC
- motor controll : park, clark transformations, real time PID
- planning, decision making : image segmentation, reinforcement learning

digital filters

- low pass filter (1st order)

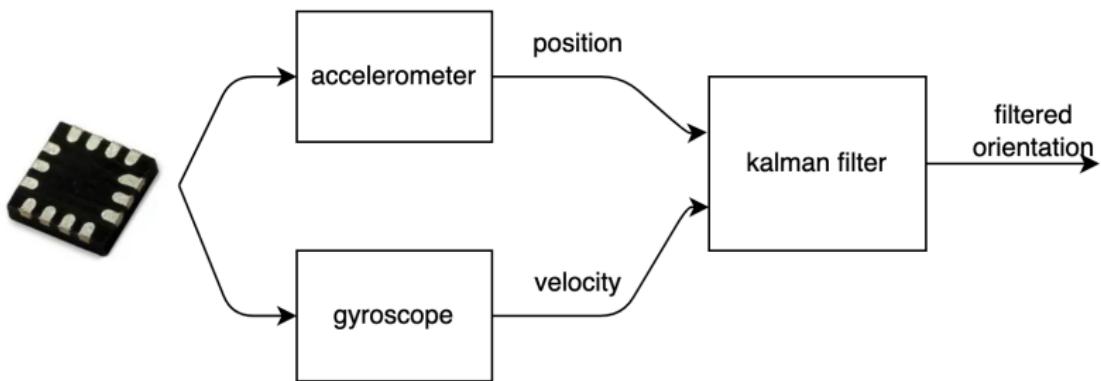
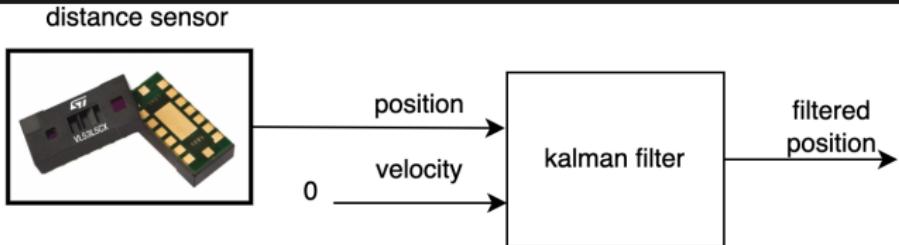
$$y_n = (1 - a)y_{n-1} + ax_n$$

$$a = \frac{\Delta t}{RC + \Delta t}$$

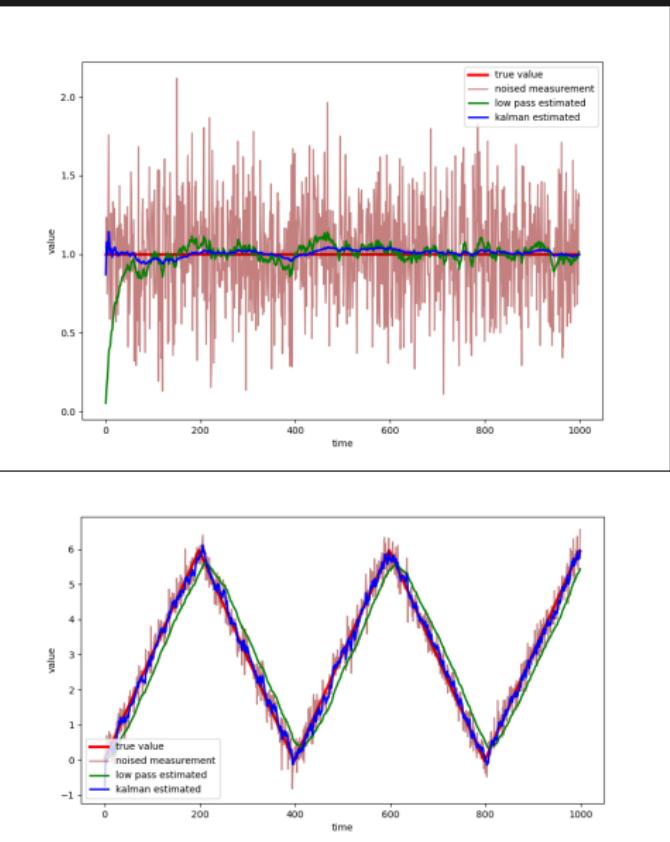
- resonant filter (2nd order)

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2}$$

digital filters - usage



digital filters - low pass vs Kalman



1D kalman is simple

```
class KalmanFilter:

    def __init__(self, dims = 1):

        #initial guess
        self.x_hat = torch.zeros(dims)
        self.p      = 0.5*torch.ones(dims)

    ...

    z   - position measurement
    dz  - velocity measurement
    pz  - position measurement uncertainty
    pdz - velocity measurement uncertainty
    ...

    def step(self, z, dz, pz, pdz, dt = 1):

        #1, prediction
        #predict the state and uncertainty
        self.x_hat = self.x_hat + dz*dt
        self.p     = self.p + pdz*(dt**2)

        #2, kalman gain
        k       = self.p/(self.p + pz)

        #3, update
        self.x_hat = self.x_hat + k*(z - self.x_hat)
        self.p     = (1.0 - k)*self.p

    return self.x_hat
```

- 1, calibrate sensor variance
(pz, pdz)
- 2, measure position z
and velocity v
- 3, dynamics model

$$z_{n-1} = z_n + v_n dt$$

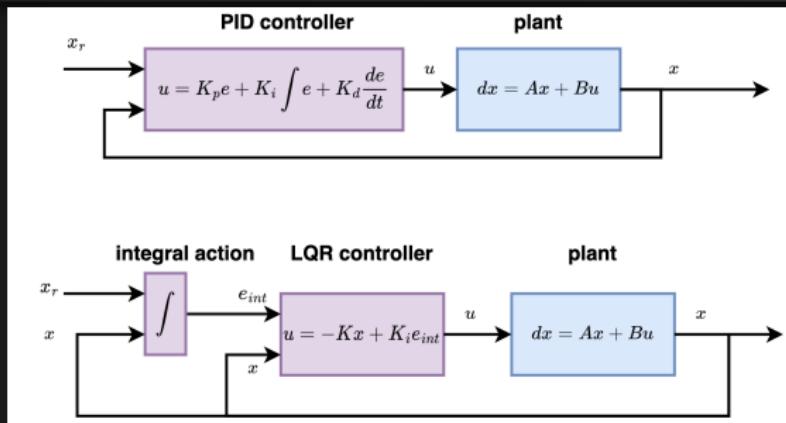
- 4, compute filter step

usage



- sensors de-noising
- sensors fusion
(gyro + accelerometer),
GPS
- navigation, error checking
- Apollo guidance computer
(Apollo 8, 1968)
- Hakuto lander crash !!!
(2023, April 2023)

controll : PID vs LQR



- PID
- single input, single output
- simple systems (motor control, temperature ...)
- empirical "tunning"
- LQR
- multiple inputs, multiple outputs
- complex dynamics (drones, F16, rockets, walking robot, segway)
- exact solution
- robust

servo state space model

1st order motor
velocity model

$$\frac{d\omega(t)}{dt} = -\frac{a}{b}\omega(t) + \frac{1}{b}u(t)$$

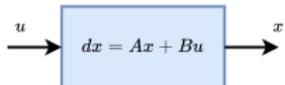
wheel angle

$$\frac{d\theta(t)}{dt} = \omega(t)$$

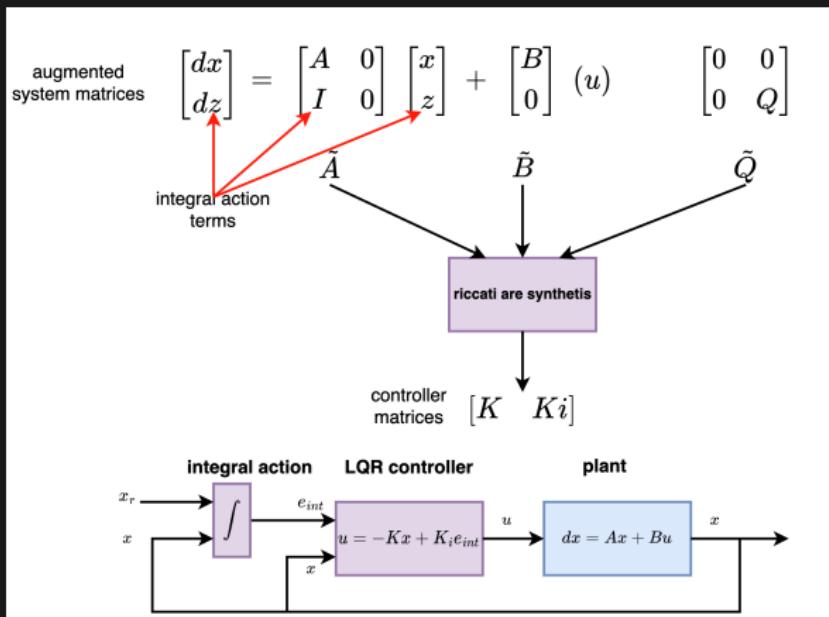
state space model

$$\begin{pmatrix} dx_0 \\ dx_1 \end{pmatrix} = \begin{pmatrix} -\frac{a}{b} & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \begin{pmatrix} 1/b \\ 0 \end{pmatrix} (u_0)$$

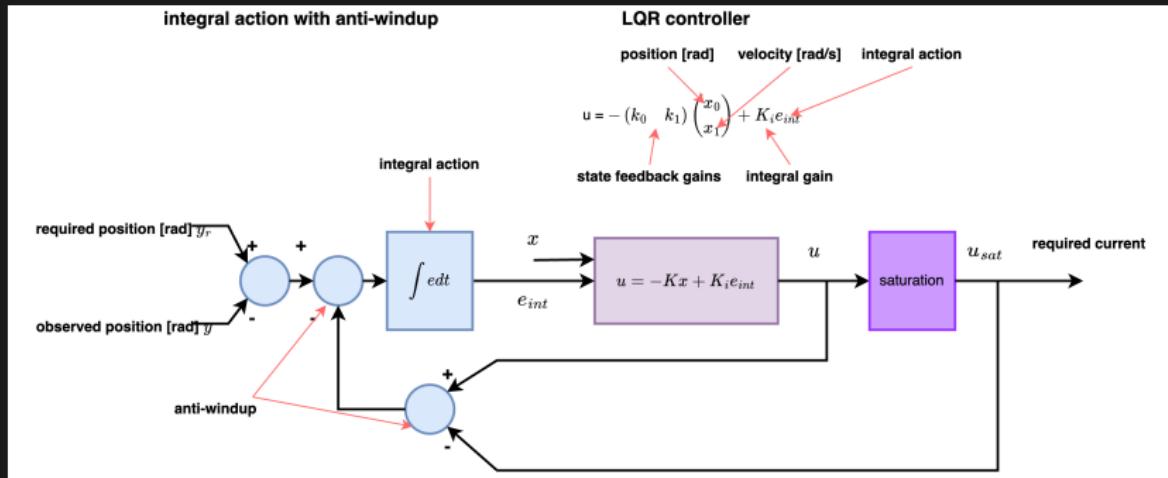
$$\frac{dx}{dt} = Ax(t) + Bu(t)$$



controller synthesis



LQR for servo

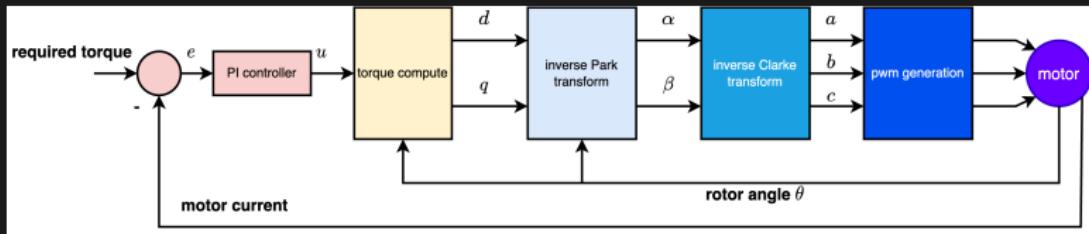


three phase motor controll



- MCU : stm32f051
 - ARM cortex m0, 48MHz
- encoder : AS5600, 12bit
- driver : MP6540H, 5A, 50V

field oriented control



field oriented control

trapezoidal control

- highest maximum speed
- lowest switching losses
- easiest implementation

- highest power output
- lowest noise
- best torque ripple
- maximum motor efficiency
- coding experience needed

field oriented controll

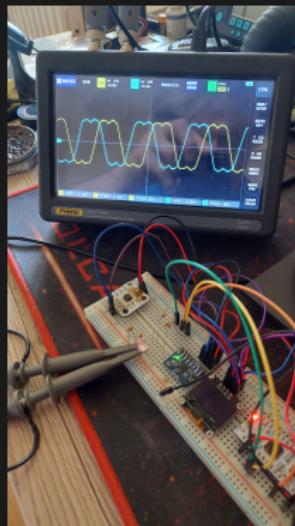
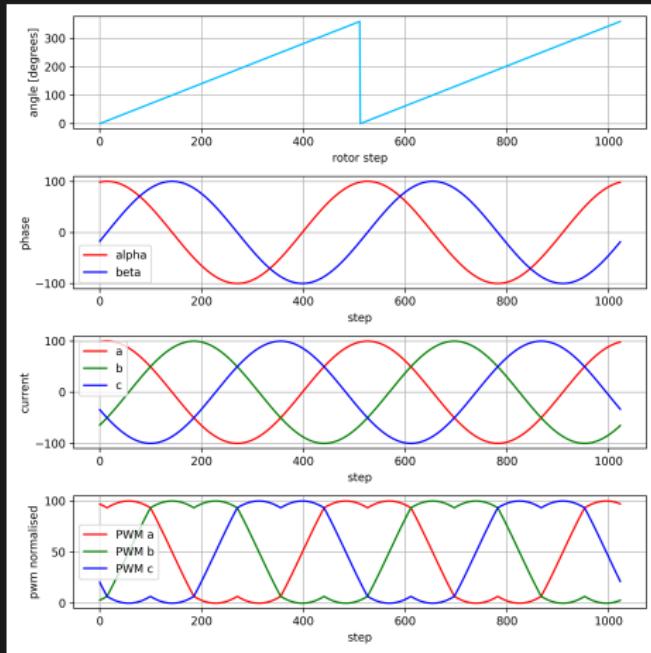
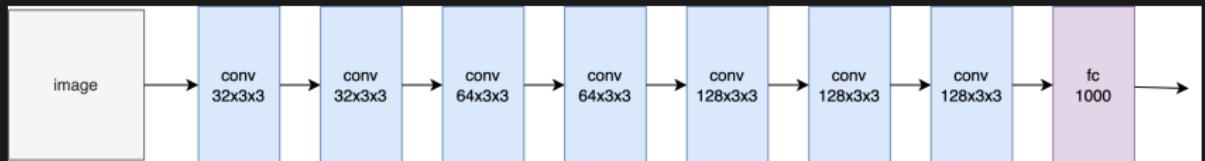


image segmentation



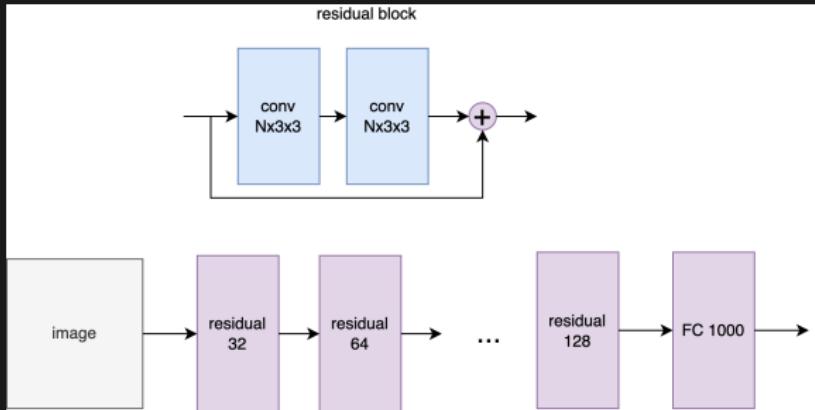
- deep learning
- convolutional neural networks
- resnet18, resnet50, u-net, mask2former ...

convolutional neural network



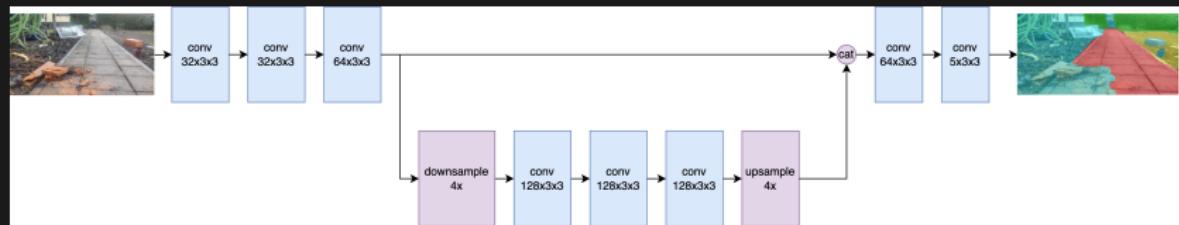
- use 3x3 kernels
- strides instead of poolings
- ReLU activation
- Adam optimizer

resnet



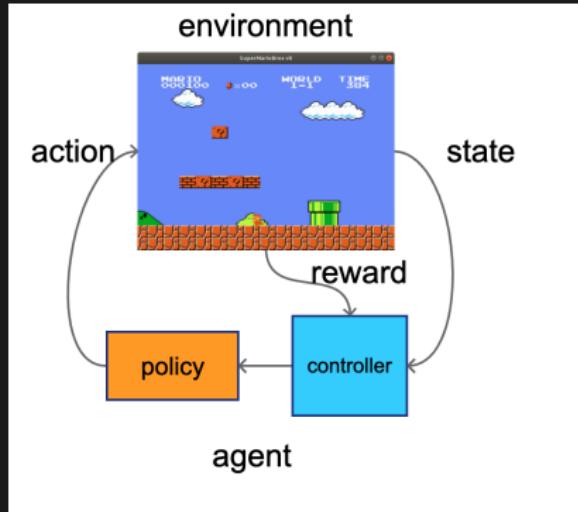
- skip connections
- resnet18, 50, 110
- AlphaGO, AlphaZero

segmentation



- multiple sizes views
- 10FPS on NVIDIA jetson nano

reinforcement learning



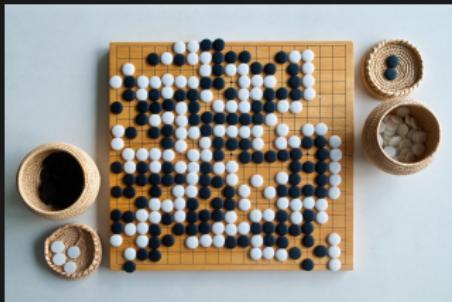
- obtain state
- select action
- execute action
- learn from experiences

deep reinforcement learning - playing Atari



2013, Mnih (DeepMind) : Playing Atari with Deep Reinforcement Learning, <https://arxiv.org/pdf/1312.5602.pdf>

reinforcement learning - path to superhuman score

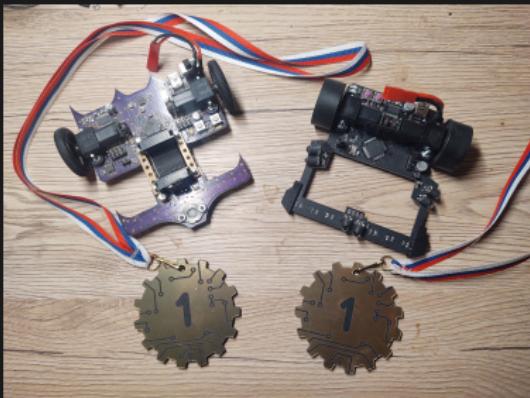
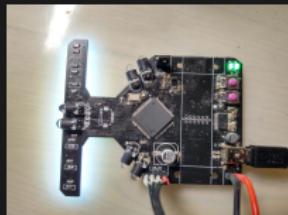


- 2016 AlphaGo,
<https://www.nature.com/articles/nature16961>
- 2020 MuZero, <https://arxiv.org/pdf/1911.08265.pdf>
- AlphaZero, AlphaStar ...

reinforcement learning - robotics



motoko uprising - line follower

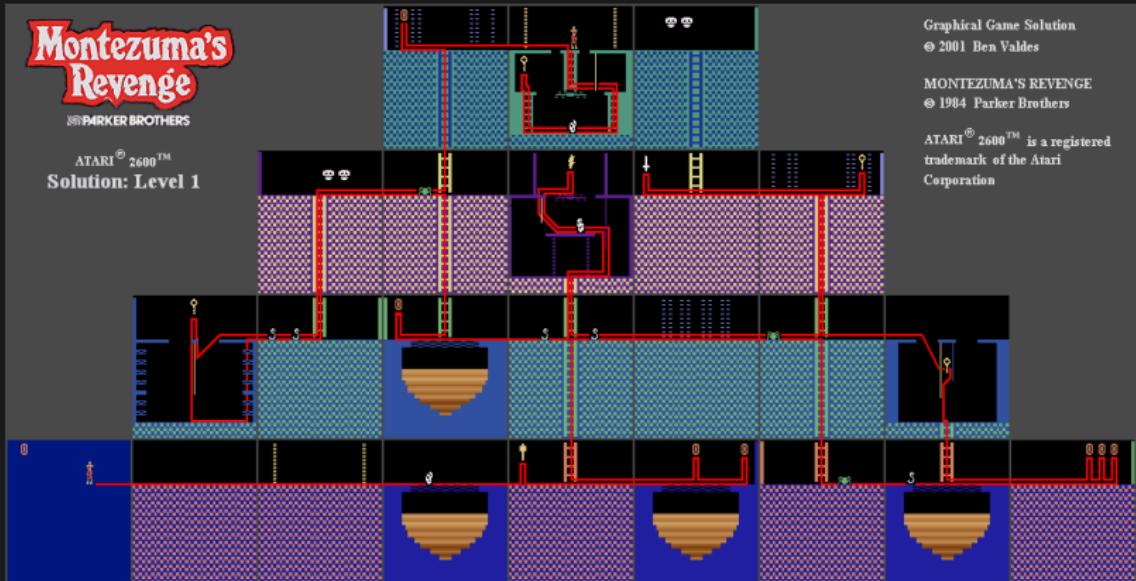


video : <https://www.youtube.com/watch?v=xUAJ1LA6Xwc>

unsolved problems

- RL sample efficiency
- sparse rewards
- generalisation

Montezuma's revenge



- extreme sparse rewards
- huge state space
- need to return hundreds of steps back
- still not solved - without game state save/load

state of the art score

source : <https://paperswithcode.com/sota/atari-games-on-atari-2600-montezumas-revenge>

year	name	score
2015	Deep Reinforcement Learning with Double Q-learning	0
2017	Curiosity-driven Exploration by Self-supervised Prediction ^a	0
2021	MuZero	2500
2018	Count-Based Exploration with Neural Density Models ^b	3705
2019	Exploration by Random Network Distillation ^c	8152
2021	GoExplore* ^d	43 000

* requires environment state saving/loading

^a<https://arxiv.org/abs/1705.05363>

^b<https://arxiv.org/abs/1703.01310>

^c<https://arxiv.org/abs/1810.12894>

^d<https://arxiv.org/abs/2004.12919>

exploration by self supervised exploitation

Exploration by self-supervised exploitation

Matej Pecháč^{a,c}, Michal Chovanec^b, Igor Farkaš^a

^a*Department of Applied Informatics, Comenius University, Bratislava, Slovak Republic*

^b*Photoneo, Ltd., Bratislava, Slovak Republic*

^c*Correspondence: matej.pechac@fmph.uniba.sk*

- self supervised model regularisation
- reached state of the art score (25 000 +) on Montezuma's Revenge
- first algorithm solving procgen hard exploration seeds

recommended sources

- book : Maxim Lapan, 2020, Deep Reinforcement Learning Hands-On second edition
- book : Enes Bilgin, 2020, Mastering Reinforcement Learning with Python
- youtuber : Yannic Kilcher, link
- youtuber : Two Minute Papers, link
- web : Paper With Code, link
- web : Intellabs, link

Q&A



- <https://github.com/michalnand/>
- michal.nand@gmail.com