# Data-Efficient Learning for Complex and Real-Time Physical Problem Solving using Augmented Simulation

Authored by:
Kei Ota, et al.
(The members in Mitsubishi Electric, Mitsubishi Electric Research Labs, and Massachusetts Institute of Technology)

Published in:
IEEE Robotics and Automation Letters
( Volume: 6, Issue: 2, April 2021)

Yahagi Takuya

# Introduction

# Introduction –Background–

- One of the goal of AI is designing robotic agents that can interact with the physical world in flexible, data-efficient and generalizable ways.

- Model-based control methods form plans vase on predefined models.
  It is data efficient, but require accurate dynamics models, which may not exist for complex tasks.
- Model-free methods rely on reinforcement learning.  These methods can solve even complex dynamics, but training these policy is inefficient. (cause of many samples)

# Introduction –Aim–

- The main aim of this research is to combine merit of these methodologies: Combining model based control methods and model free methods to achieve flexibility and data efficiency.

- They are inspired by how human learn. (cognitive science)
So, to compare against how human learn is also their aim.

# Problem Formulation

# Problem –Environment–

- Circular Maze Environment (CME) is used as research environment.
  - This is physical environment
- They consider the problem of moving the marble to the center of CME.
- At first agents learn in a physics engine, then adapts it in a real system. (sim-to-real)
- Base is model-based method.

The Image of marble

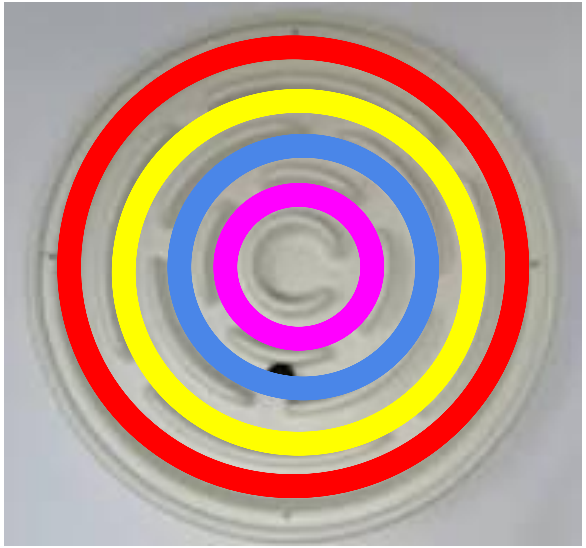
The Image of CME

# Problem –Environment–

- The CME aria are splitted into 4 Rings (Ring 1 ~ Ring 4).

- To make problems easier for agents, they changed a little goals.

  Before: To move the marble to the center of CME.
  After : To move the marble to the next inner ring.
  Then finally the marble reach the center of it.

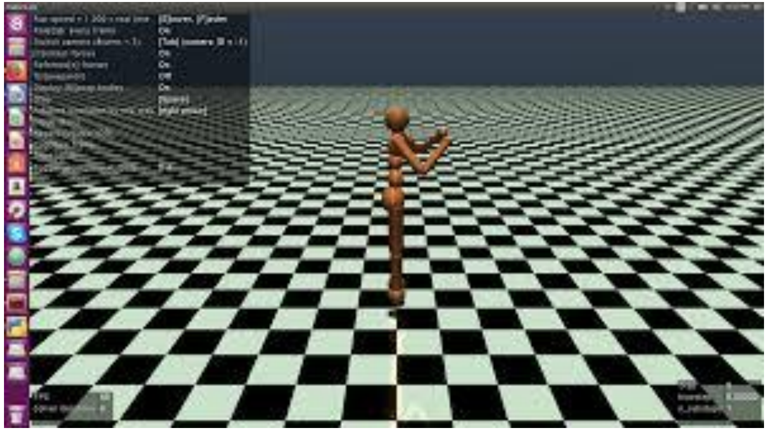


Splitted area

# Problem –Environment–

- At first agents learn in a physics engine, then adapts it in a real system.
  (It is called as sim-to-real)

- Base is model-based method.



- In this research, they represent the physics engine by $f^{PE}$ , the real system model by $f^{real}$ and the residual dynamics model by $f^{GP}$ (it represent errors between $f^{PE}$ and $f^{real}$ .

$$f^{real}(x_k, u_k) \approx f^{PE}(x_k, u_k) + f^{GP}(x_k, u_k)$$

# Problem –Environment–

- The goal of the learning agent is to learn accuracy model $\pi(u_k | x_k)$, where $u_k$ is an action ( control inputs ) and $x_k$ is a state observation.

- $x_k$ is represented as following:

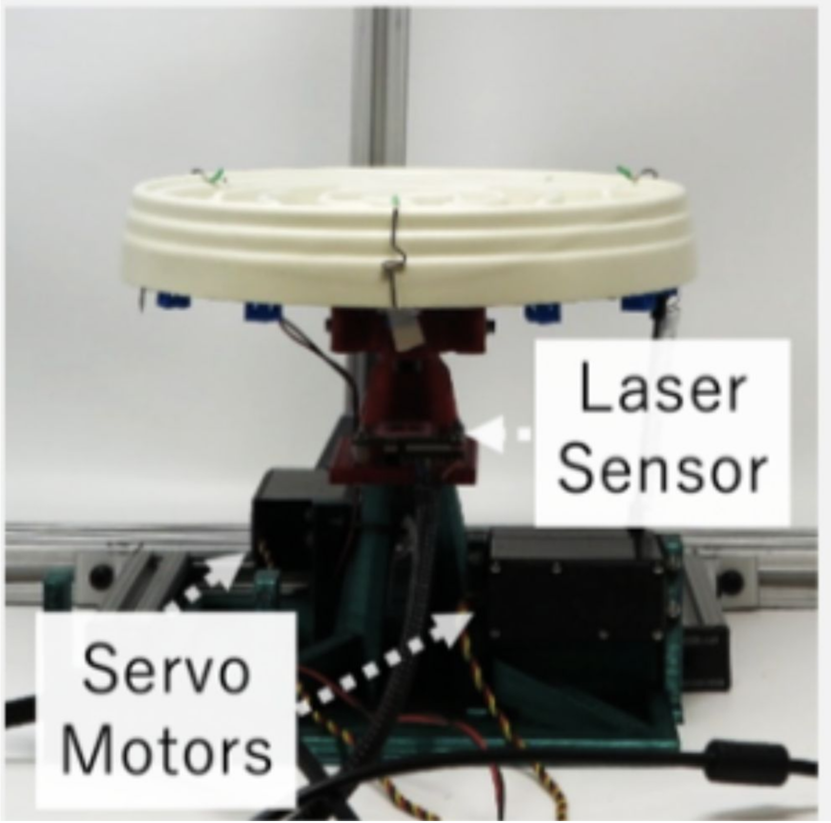
$$x = (ß, \gamma, \theta, \theta')$$

ß,γ: the value of the gradient in the x-axis and y-axis direction, respectively
θ,θ': the value of the angular position ( the direction of the marble move on) and the angular velocity of the marble.
* All of these values are continuous.

# Problem –Environment–

- ß and γ are measured by using a laser sensor.
- θ and θ’ are measured by using a camera which is set above the CME.

- The control inputs u_k consists of two variables, the gradient of X-axis and Y-axis.
  These two variables operated by servo motors like radio controller.



# Problem –research topics–

- In these environment, they researched following topic.

1. What is needed in a model-based  sim-to-real architecture for efficient learning in physical systems?
2. How can we design  a sim-to-real agent that behaves and learns in a data-efficient manner?
3. How does the performance and learning of their agent compare against how humans learn to solve these tasks?

# Approach

- Physics Models
- Sim-to-real
- Control output
- Trajectory Optimze
- Online control with NMPC

# Approach –About physics model–

- Before considering on sim-to-real, we need to consider on sim-to-sim.
- The left sim means the limited simulator model, and is represented as $f^{PE}_{red}$ .
  This model can get values only from what real system can offer.
  So, in this model, x_k has only 4 factors(ß, γ, θ, θ’).
- The right one means the full simulator model and is represented as $f^{PE}_{full}$ .
  This model can get every values from what physical system can offer.
  (example: the coordinates position of the marble info, it cannot get in the real)
- $f^{PE}_{full}$ is used instead of the real system model.
- To do so, we can check if there are any errors or lacks before more complex experiment, sim-to-real.

# Approach – sim-to-real –

- Designing sim-to-real agents have a difficulty cause of the gap between the real and simulation.
  There are two causes of the gap.
  1. Incompleteness of the physics engine :
     The physics engine can not copy the rules of physics in the real world perfectly.
  2. The physical noize. (example: controller delay, unclear images from camera, etc…)

  - So, these errors must  be improved.
    For 1: Physical parameter estimation
    For 2: Gaussian process Regression is used.

# Approach – sim-to-real –

For 1(Incompleteness): Physical parameter estimation

- They used Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to estimate 4 physical parameters, they are represented as 4 dimensional vector μ.
  1. Initialize multiple μ_1stgen randomly.
  2. For each μ_1stgen, verify fitness with using the target function.
  3. Conduct the evolution strategy.
     (example: Highly adapted μ_1stgen are selected and used to generate the next generation of individuals μ_2ndgen)
  4. The probability distribution that reflect the features of 3 used and created new μ_ngen randomly.
  5. Iterate 1~4, and if conditions of converge are met, stop algorithms and print out optimized one μ*.

# Approach – sim-to-real –

For 1(Incompleteness): Physical parameter estimation

- μ*(optimized parameter) is simulated as following:

$$\mu^* = arg_{\mu} min \frac{1}{||D||} \sum_{(x_k^{real}, u_k^{real}, x_{k+1}^{real}) \in D} ||x_{k+1}^{real} - f_{red,\mu}^{PE}(x_k^{real}, u_k^{real})||_{W_d}^2$$

where, D: the transition on the real systems

$W_d$: the weight matrix with a value which change $\theta_{k+1}$ to 1.

# Approach – sim-to-real –

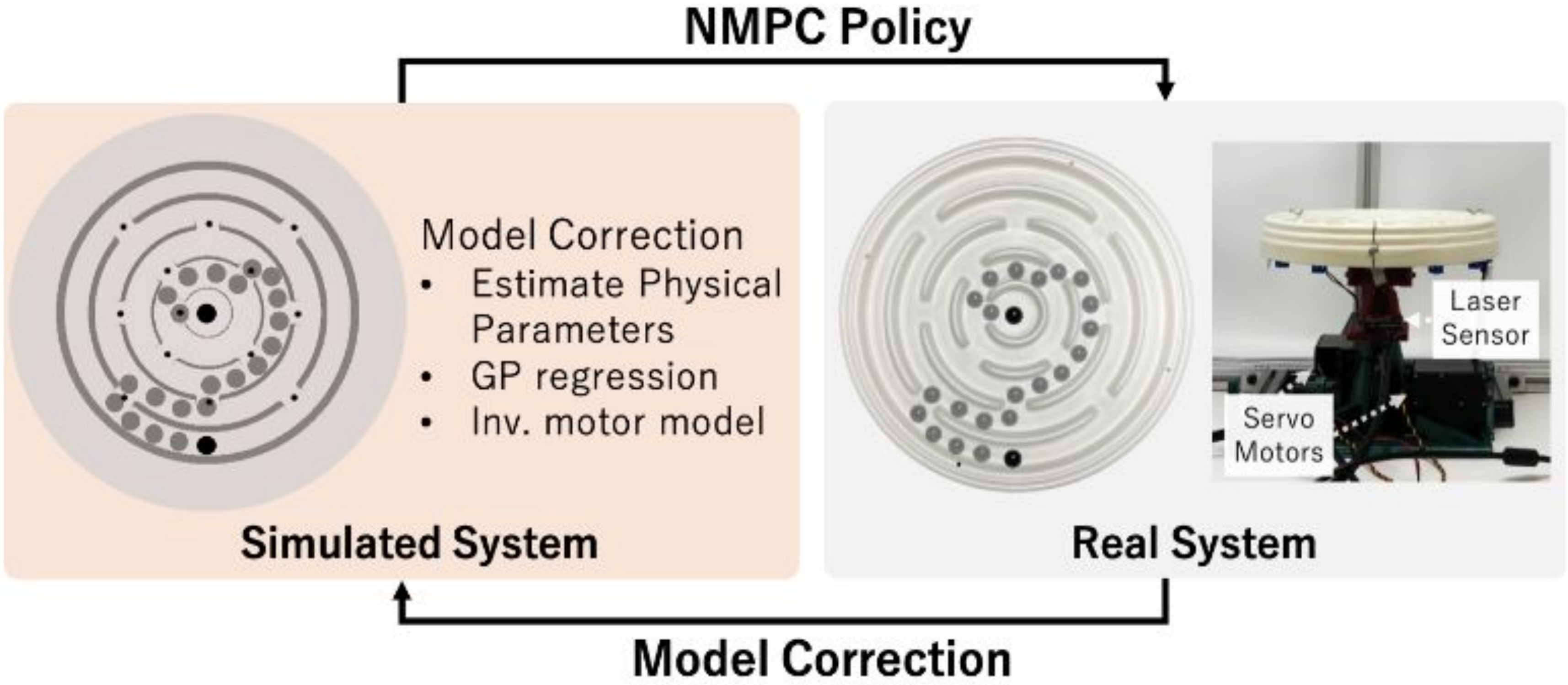
For 2(noise): Gaussian Process regression (GP)

- GP is used for decrease the mismatch between the simulator and the real system.

$$L^{GP} = \frac{1}{||D||} \sum_{(x_k^{real}, u_k^{real}, x_{k+1}^{real}) \in D} ||(x_{k+1}^{real} - \int_{red,\mu^*}^{PE}(x_k^{real}, u_k^{real})) - \int^{GP}(x_k^{real}, u_k^{real})||^2$$

- By minimizing above objectives, it learns the regression between two systems.

# Approach – The flow of estimation and data correction–



# Approach – about control input –

- Actually, control input has a problem.
- Because the controller of CME have longer time of waiting than command rate, so it causes the delay in control.
- To resolve it, they use use an inverse model for motor actuation.
  It predicts the action command (u_x, u_y) to achieve the desired state $(ß_{k+1}^{des}, \gamma_{k+1}^{des})$ , given the current state $(ß_k, \gamma_k)$ a at instant k.

- It is represented as f_imm.
- It is learned using a standard autoregressive model with external input.
- It is learn by running the CME using a sine wave input to the motor and collecting the motor response data.

# Approach – Trajectory Optimize–

- In this part, describe about the optimize algorithm of the model-based control.
- They used the iterative Linear Quadratic Regulator(LQR).

State cost: the distance from target state

$$\ell(x) = ||x - x_{target}||^2_{W},$$

W: weight matrix

control cost: adjust power of gradient input

$$\ell(u) = \lambda_u ||u||^2$$

$$\min_{x_k, u_k} \sum_{k \in [T]} \ell(x_k, u_k)$$

$$x_{k+1} = f(x_k, u_k)$$

$$x_0 = \bar{x}_0$$

# Approach – Online control with using NMPC–

- CME need to use feedback control based online model. ( real time interact)
- In this research, they used Nonlinear Model Predictive Control (NMPC).
- By using NMPC controller to track the trajectory obtained from the trajectory optimization module, to control the system in real-time.
- The controller uses the least-squares tracking cost function as following:

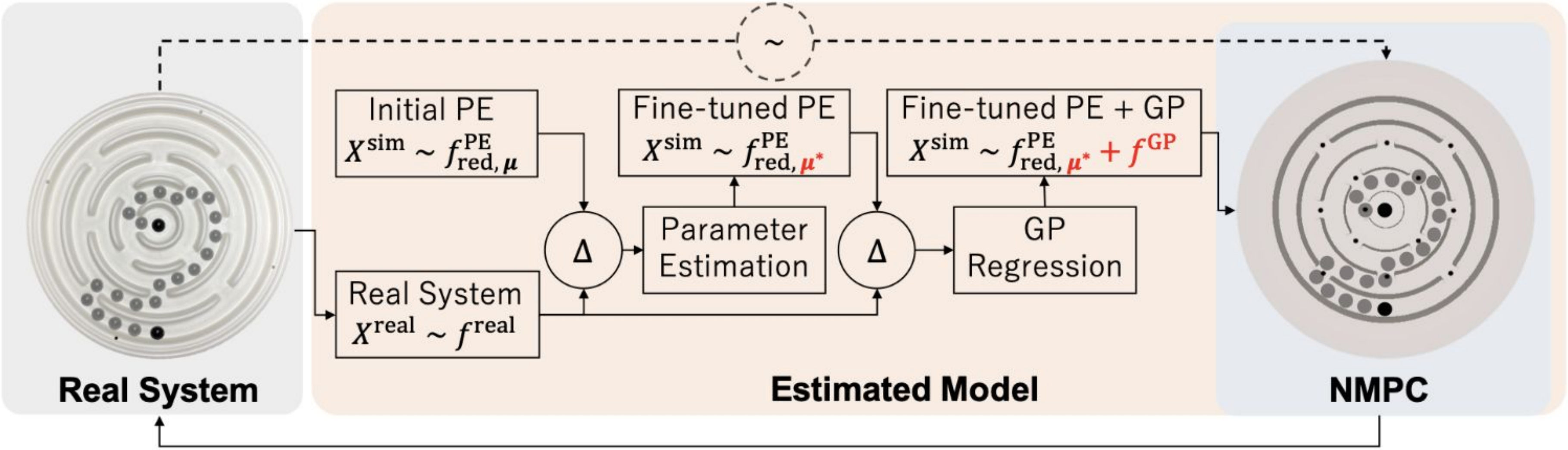
$$\ell_{tracking}(x) = ||x_k - x_k^{ref}||_Q^2$$

$x_k$ : *the system state at instant k,*

$x_k^{ref}$ : *the reference state at instant k* = next desired state

$Q$: *weight matrix*

# Approach – the flow how to make the model–



# Experiment and Results

# Experiment

- In experiment, they did three experiment:

    1. How physical parameters work?
    2. The verification of the performance of control.
    3. How does the performance and learning of their agent compare against how humans learn to solve these tasks?

# Experiment–How physical parameters work? –

- Additional settings are added about the friction parameter.
  In $f^{PE}_{full}$ , the model that is used instead of the real system in sim-to-sim, the friction parameter is decreased. Because, the ground of real system CME is smoother than the one of physics system.

- In $f^{PE}_{red}$ , the friction parameter is initialized by the default value of MuJoCo.

# Results –How physical parameters work? –

- They used NMPC to correct samples.
- As a result, μ* is converged by only ~10 transitions for each rings.

[sim-to-sim]

- The error of θ (ball position) between $f_{red}^{PE}$ and $f_{full}^{PE}$ is:

$$\approx 2e-3[rad](\approx 0.1[deg])$$

- So they conclude the CMA-ES produces accurate enough parameters in sim-to-sim.

[sim-to-real]

- The error of θ (ball position) between $f_{red}^{PE}$ and $f^{real}$ is:

$$\approx 9e-3[rad](\approx 0.5[deg])$$

- So, the effects of friction are still left.

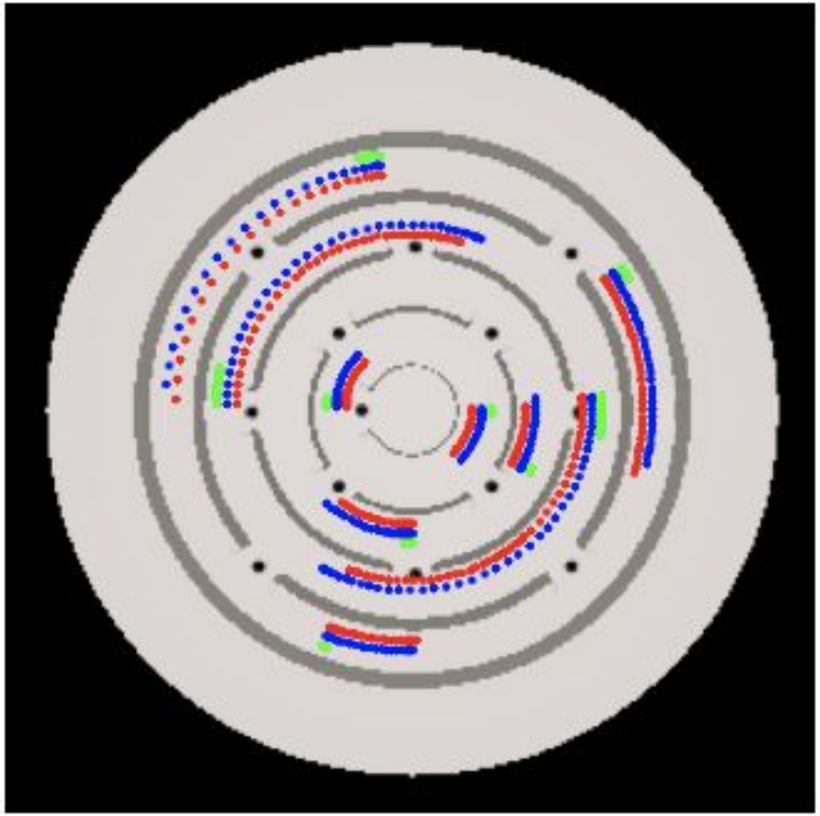
# Results –How physical parameters work? –

Points means trajectory of:

red : $f^{PE}_{full}$

blue : $f^{PE}_{red}$

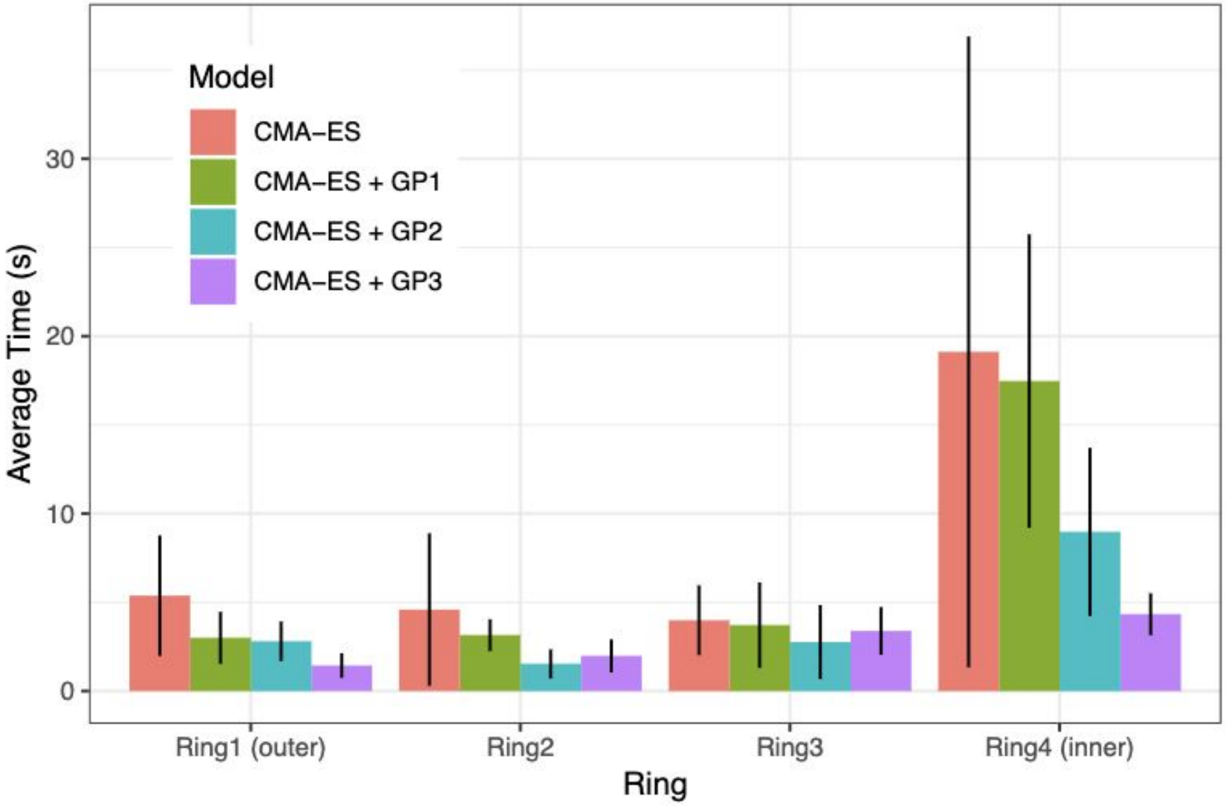
green: before estimation model



# Experiment–The verification of the performance of control–

- Because sim-to-sim agent is enough good by only CMA-ES, so omit the results of sim-to-sim.
- They use GP models to improve the CMA-ES model of sim-to-real.
- These models are divided into 4 groups depends on the number of data.
  1. CMA-ES: Without any GP modeling
  2. CMA-ES + GP1: has learned GP model from 5 rollout of the CMA-ES
  3. CMA-ES + GP2: has learned GP model from 10 rollout
     (CMA-ES:5, +GP1: 5).
  4. CMA-ES + GP3: has learned GP model from 15 rollout
     (CMA-ES:5, +GP1: 5, +GP1+GP2: 5)

# Result–The verification of the performance of control–



# Experiment–The comparison with human performance–

- 15 participants: who are not involve in this project.
- They are instructed to solve 5 times continuously.
- They have from 0 to 4 chances for learning this environment.
- So, CMA-ES (0 chances) and CMA-ES + GP1 (5 chances) is used as the target of comparison with humans.
- They control CME with using Joystick.
- They first experienced a minute without the marble.
- 3 of the participants have experience of CME with hand ( not controller )

- The researchers correct data of how much time the marble spend in each ring, and how much time they spend for this trial.

# Result–The comparison with human performance–

- Two of participants could not finish it in 15 minutes, so their data were excluded.
- The average time of how they spend in first trial is 110 sec (66 ~ 153).
  The average time of how they spend in final trial is 79 sec (37 ~ 120).
- These tendency of human is like to CMA-ES.
  ( 33 sec to 27 sec)
- However, there are no statistical reliability.

# Result–The comparison with human performance–

- For improving the power of stasti verification, they average the all of trials of human do and average between CMA-ES and CMA-ES+GP1.

TABLE I: Average time spent in each ring [sec].

| | Human | CMA-ES + GP0/1 |
|---|---|---|
| Ring 1 (outermost ring) | 22.6 | 4.18 |
| Ring 2 | 8.0 | 3.87 |
| Ring 3 | 24.3 | 3.85 |
| Ring 4 (innermost ring) | 41.1 | 18.29 |

# Consideration

# Consideration

- Suggested method show that it spend few minutes with interaction with system for learning CME.
- One of the merit of flexibility of this approach is that it can be generalized well because it is based on general physics engine (regardless of the kind).
- They try to test the generalizability and transferability of this approach when applied to different mazes and balls.
- Also, looking interfacing with common robot optimization software to make it more useful for general robotics applications for more effective use of physics engines for such problems.