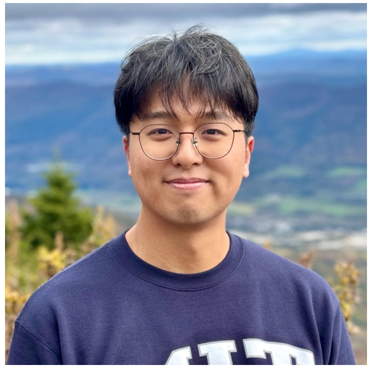
Position: Automatic Environment Shaping is the Next Frontier in RL

Younghyo Park*, Gabriel B. Margolis*, and Pulkit Agrawal

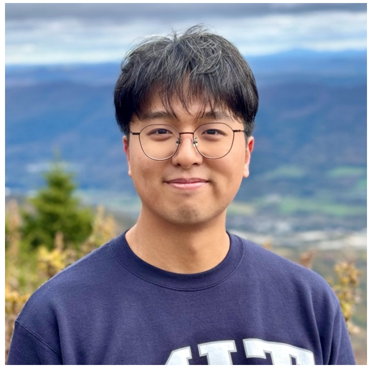


ICML
International Conference
On Machine Learning

Massachusetts Institute of Technology
Improbable AI

Position: Automatic Environment Shaping is the Next Frontier in RL

Younghyo Park*, Gabriel B. Margolis*, and Pulkit Agrawal ⟵ We all work on Robotics!

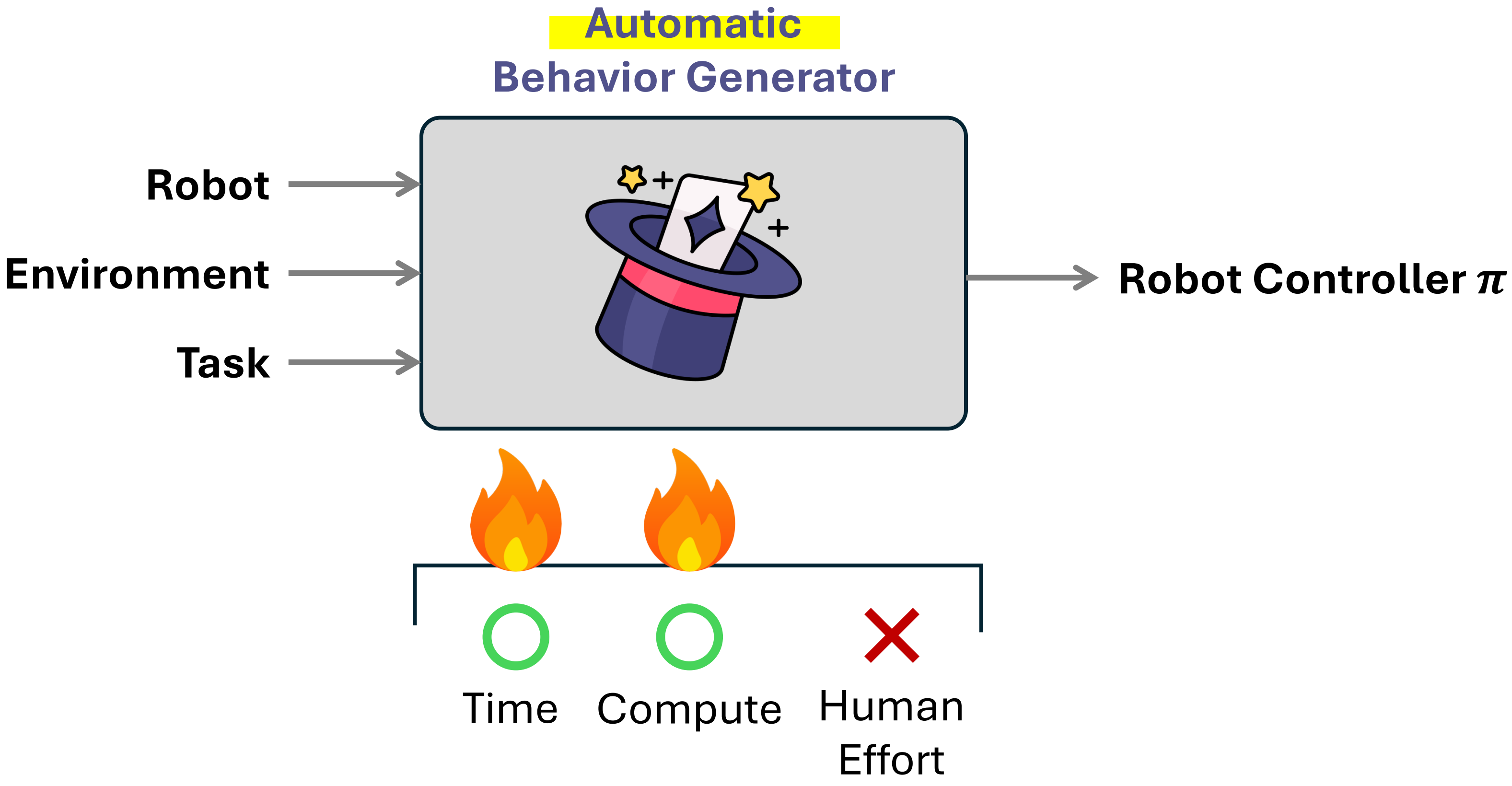


ICML International Conference On Machine Learning

Massachusetts Institute of Technology

Improbable AI

Dream of Every Roboticist



Dream of Every Roboticist

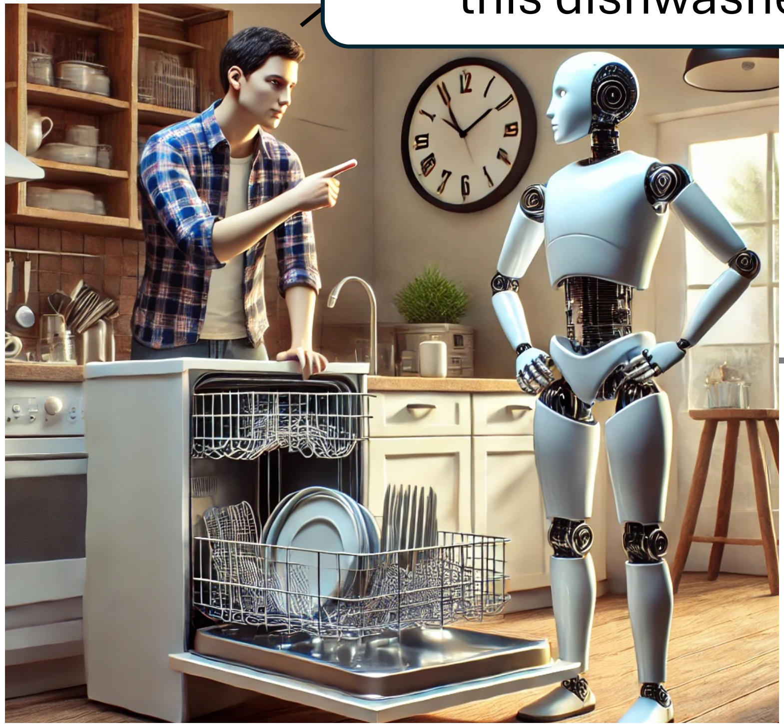
I want you to learn how to chop this ginger by tmrw.

Automatic Behavior Generator

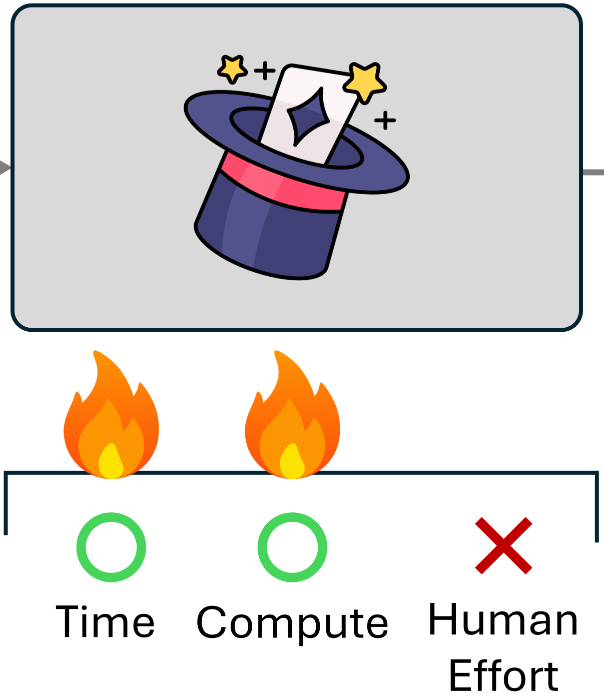
Time | Compute | Human Effort

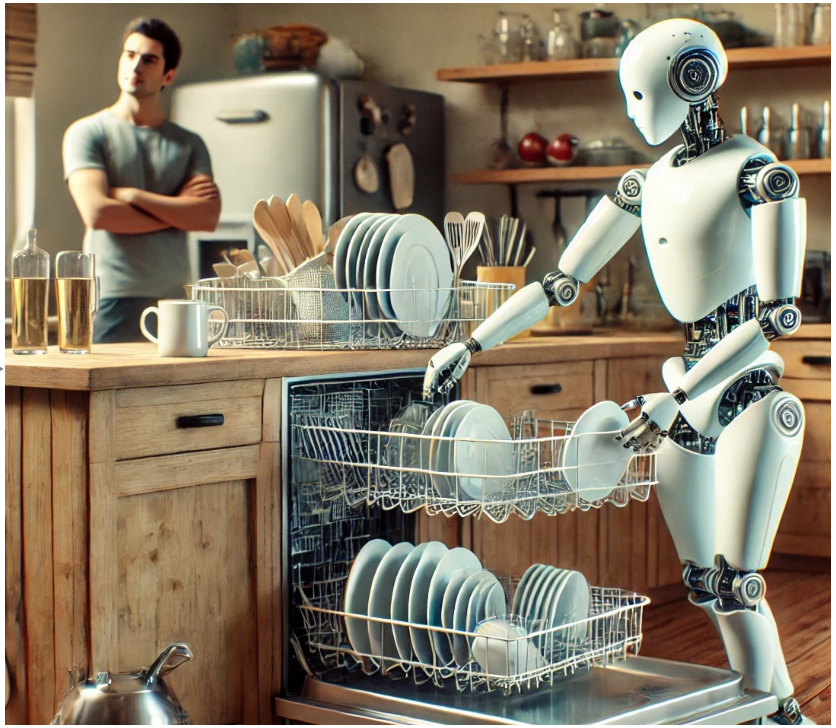
Dream of Every Roboticist

I want you to learn how to unload this dishwasher by 5pm.



Automatic Behavior Generator

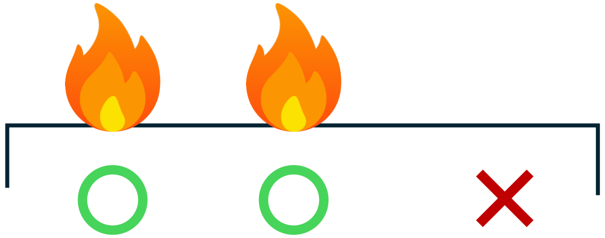


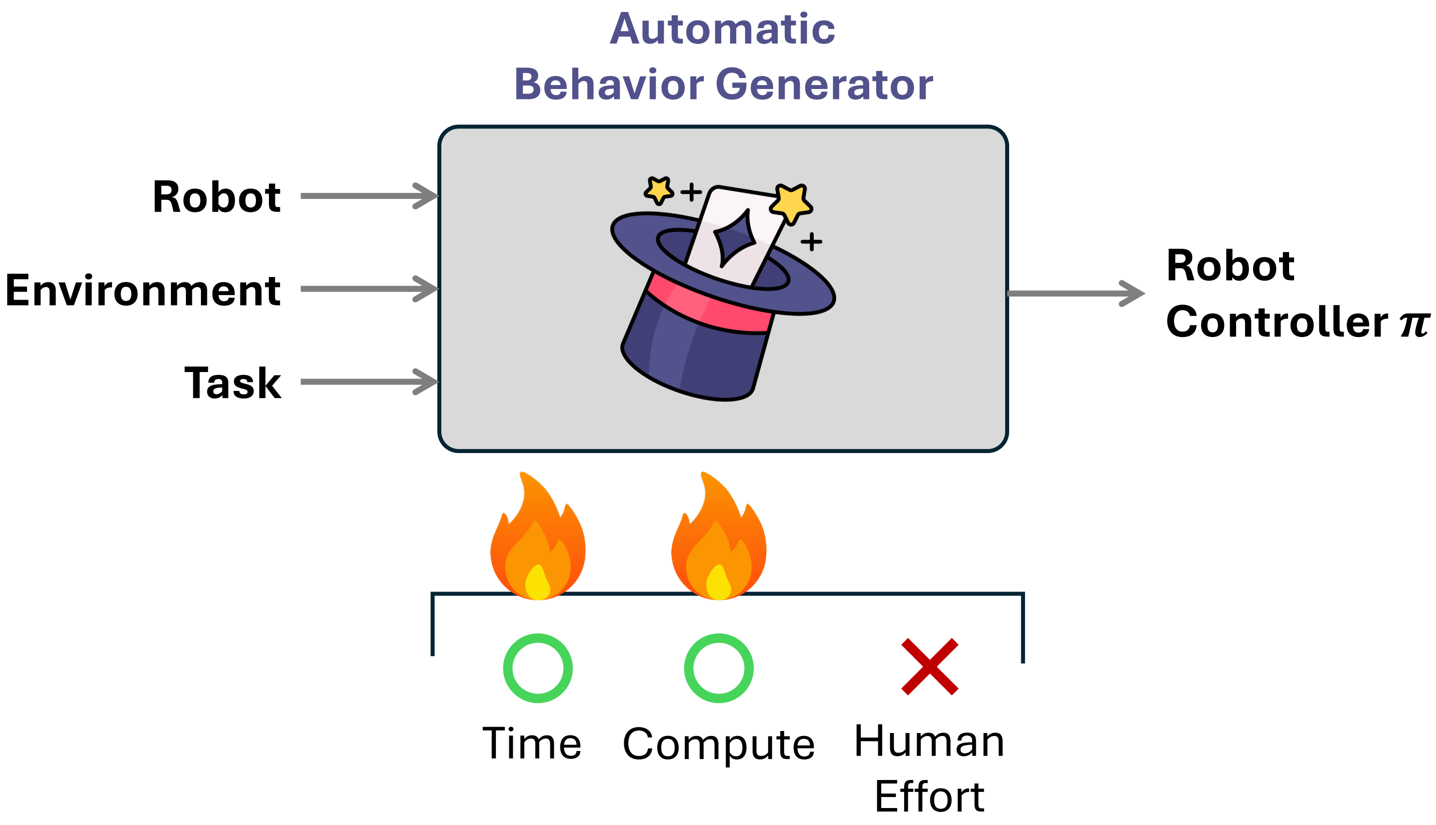


Dream of Every Roboticist

I want you to learn how to unpack all these boxes by tmrw.

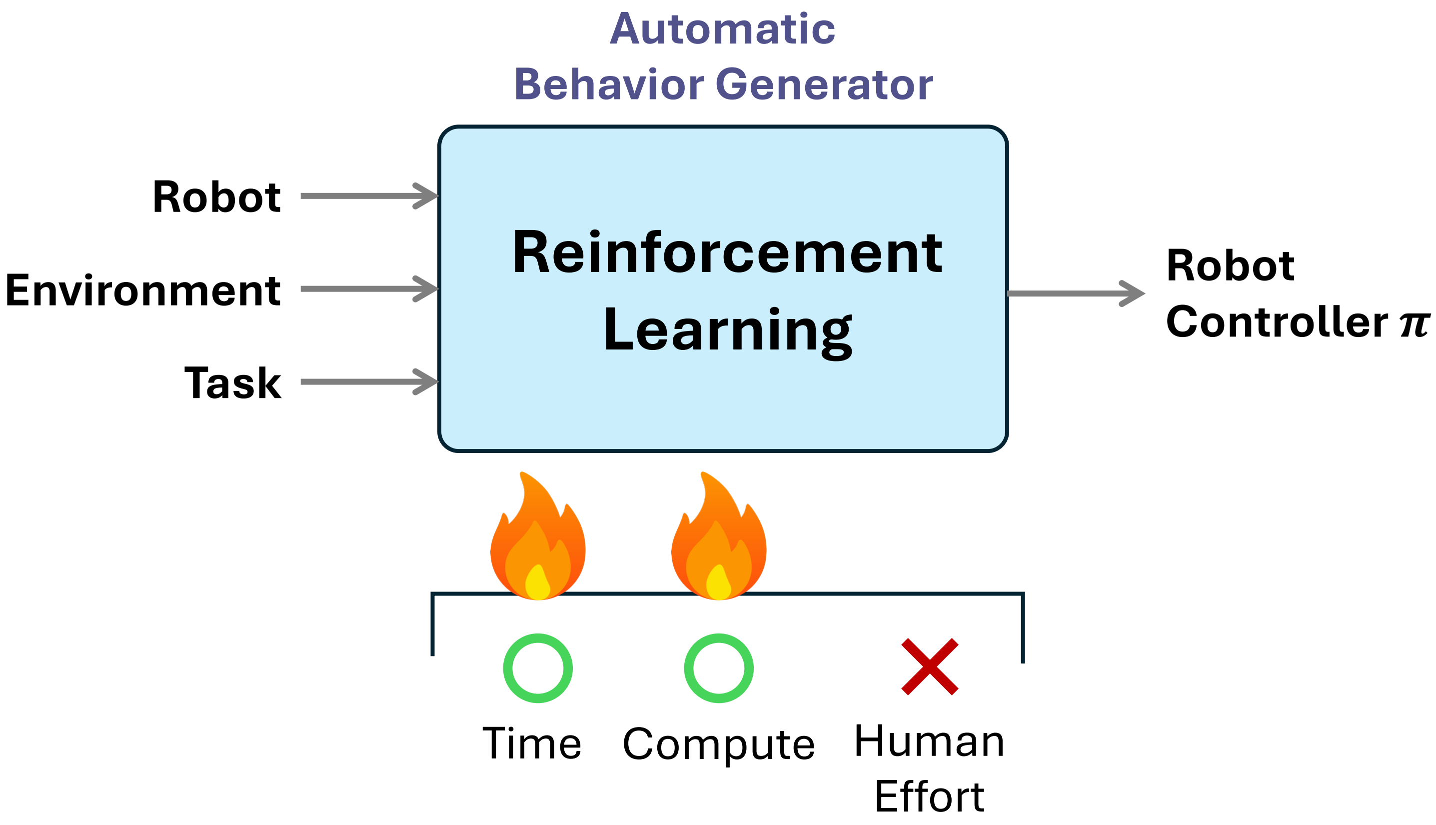
Automatic Behavior Generator



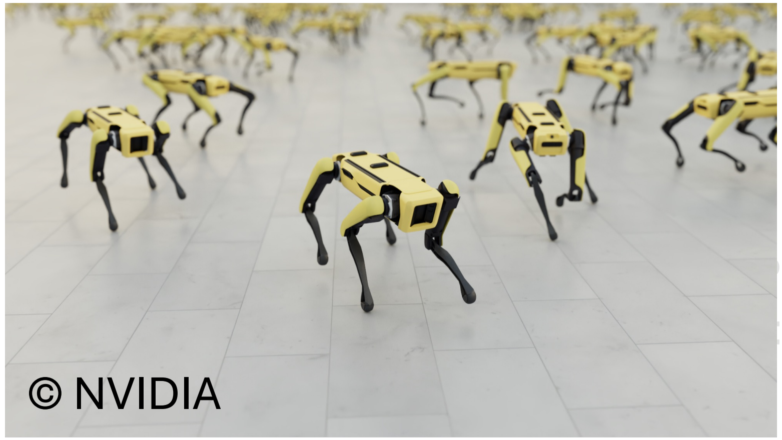


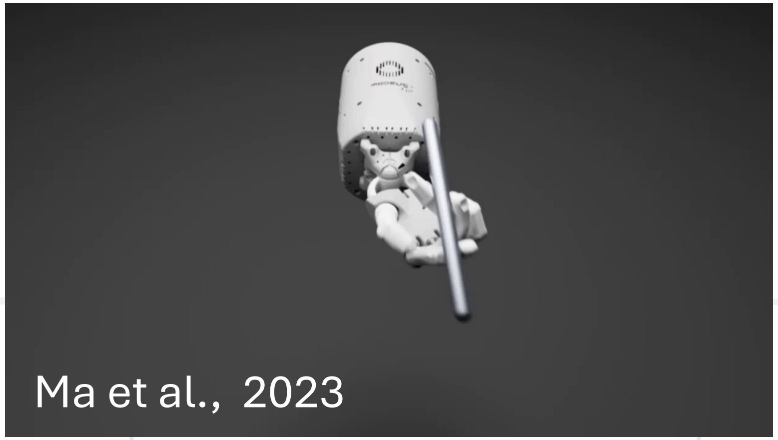
Too good to be true?

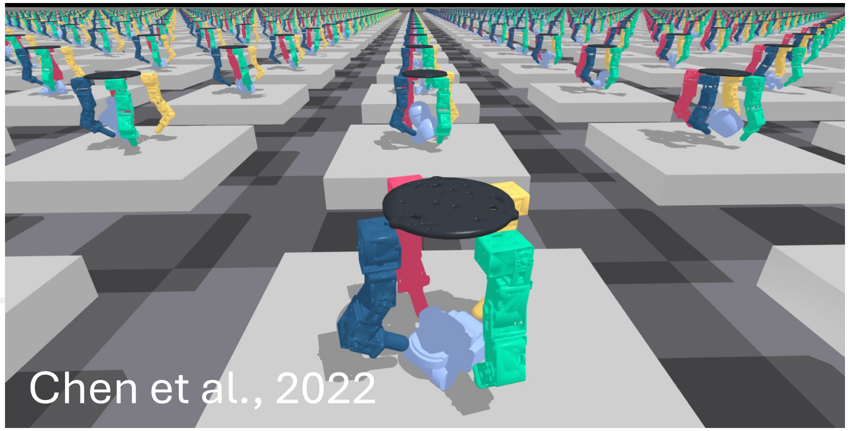
Promise of RL



Promise of RL vs Reality


© NVIDIA

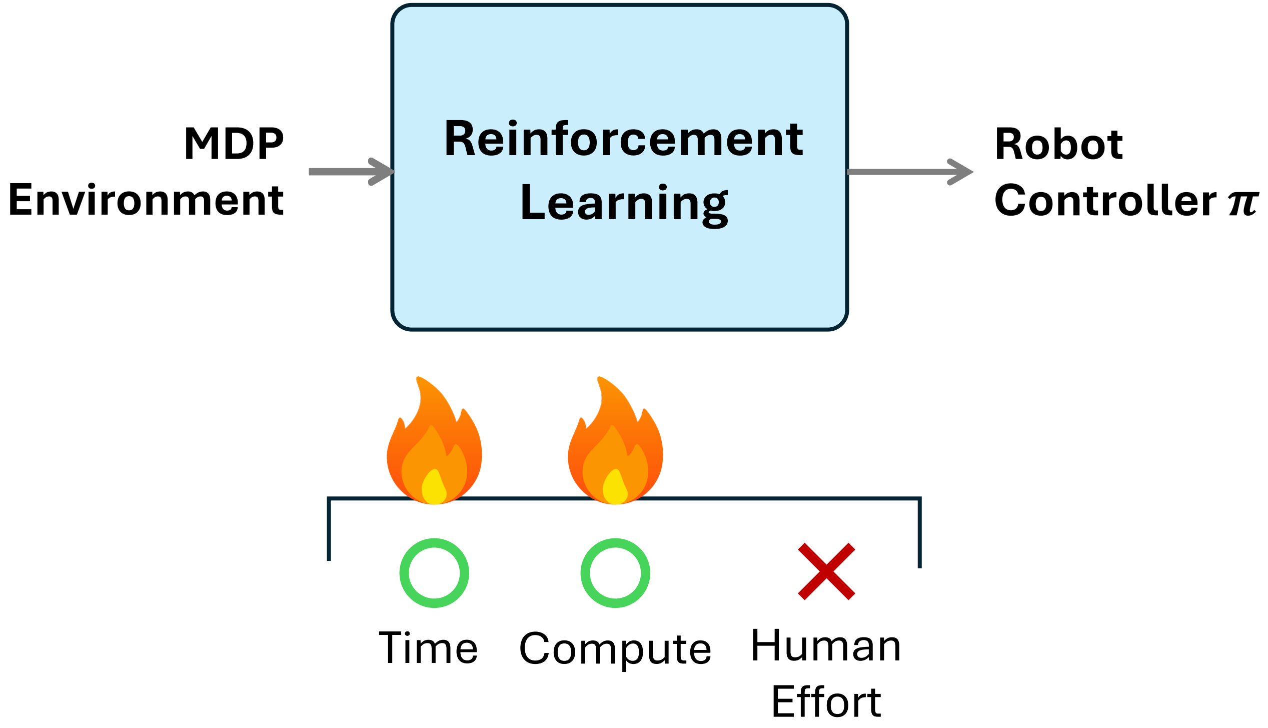

Ma et al., 2023


Chen et al., 2022

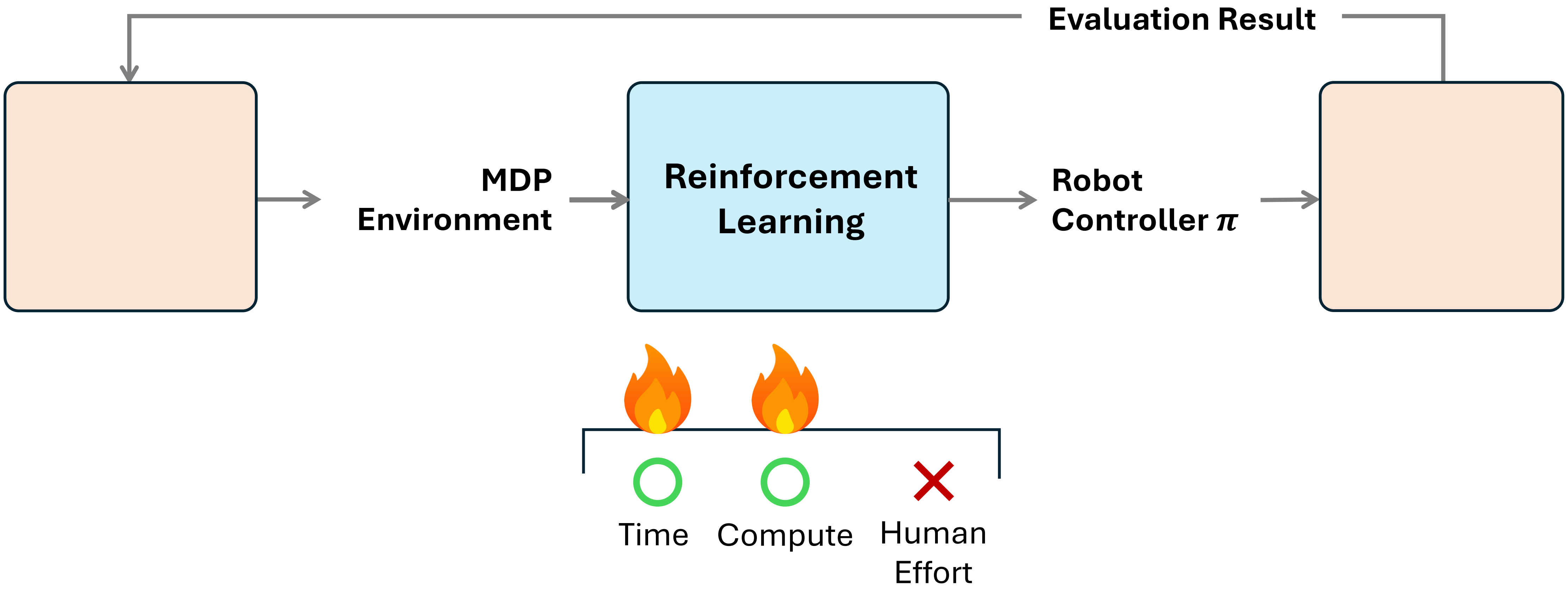
Not necessarily True.

“Reinforcement Learning, in theory,
is a general-purpose, automated optimal control solver for any MDP setting.”

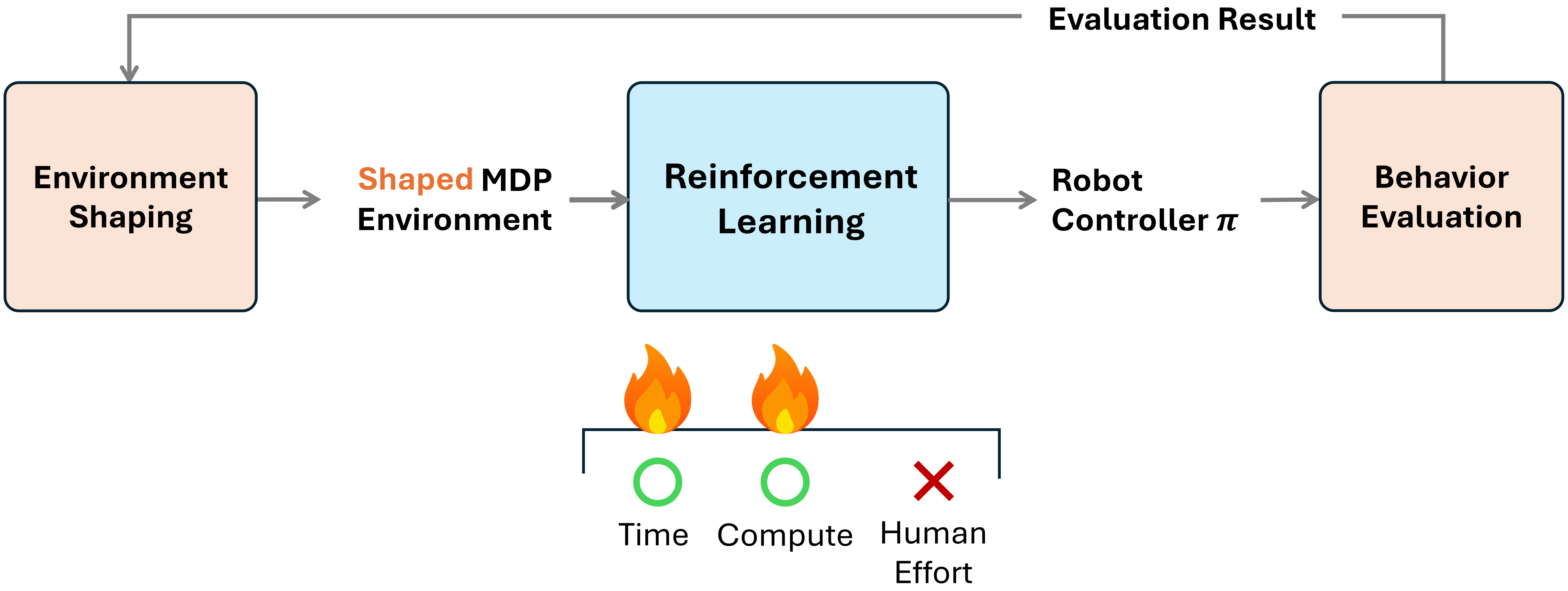
Promise of RL vs Reality



Promise of RL vs Reality

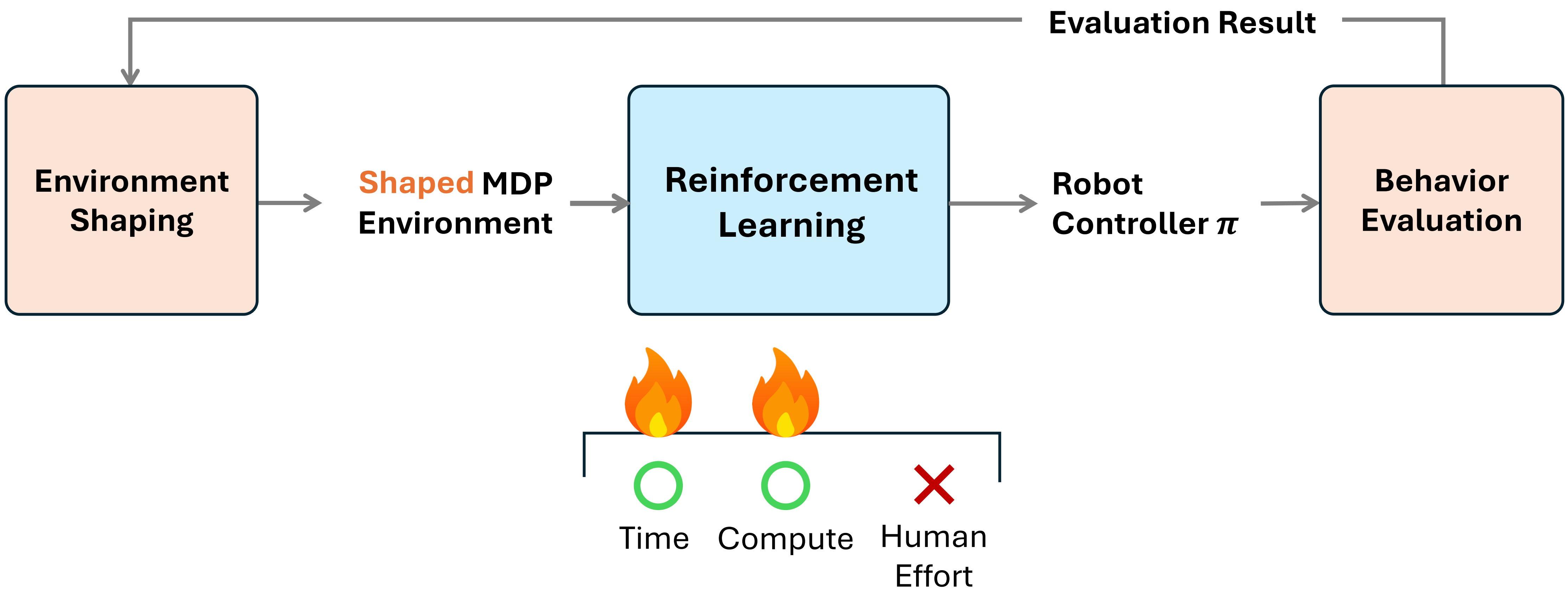


Promise of RL vs Reality



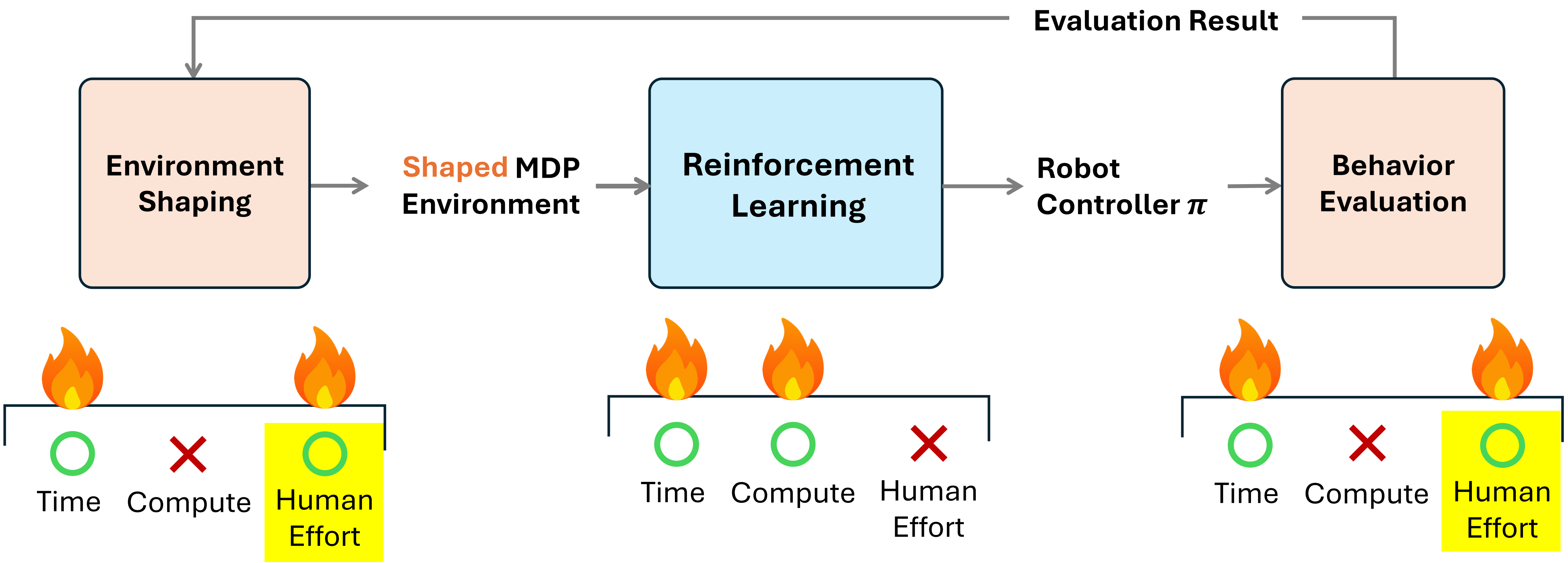
Fixing RL vs Environment Shaping

Promise of RL vs Reality

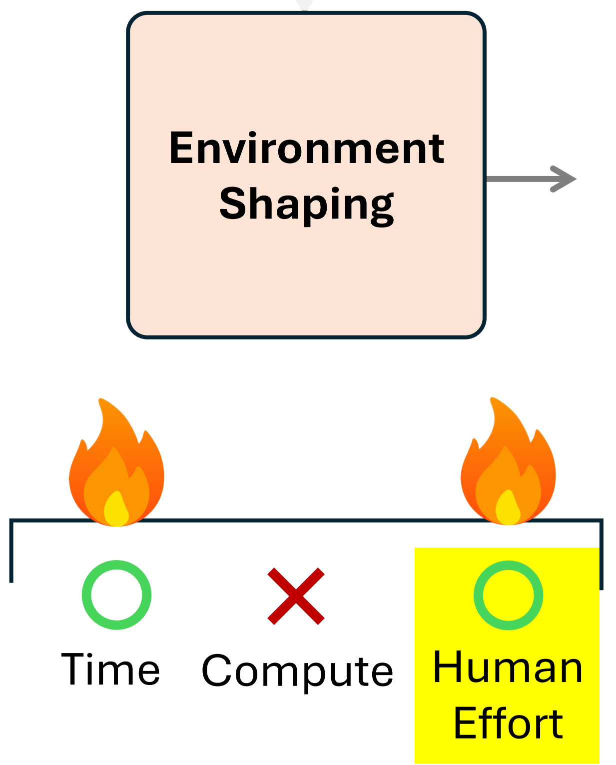


Fixing RL vs **Environment Shaping**

Promise of RL vs Reality



Promise of RL vs Reality



Reward Shaping

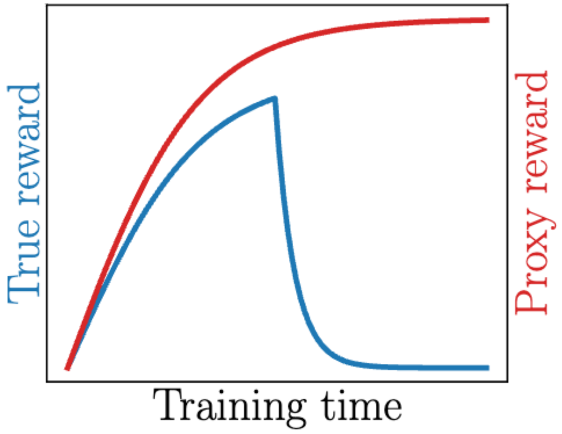


Figure 2: An illustration of reward hacking when optimizing a hackable proxy. The true reward first increases and then drops off, while the proxy reward continues to increase.

[1] Skalse et al., Defining and Characterizing Reward Hacking [NeurIPS 2022]

Promise of RL vs Reality

Evaluation Result

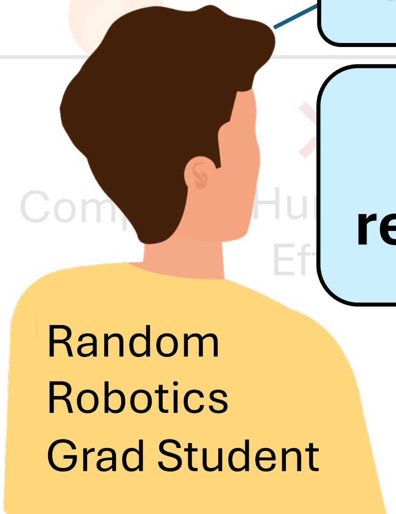
Environment Shaping

Reward Shaping

```
env.compute_reward()
```

How about using RL?

Time	Compute	Human Effort
○	✕	○



NO, THANK YOU

I really don't want to do **reward shaping**; I hate that.

Promise of RL vs Reality

Environment Shaping

Time
Compute
Human Effort

Reward Shaping

```
env.compute_reward()
```

Action Space Shaping

```
env.get_actuation(...)
```

Observation Shaping

```
env.get_observation()
```

Terminal Cond. Shaping

```
env.check_terminal()
```

Reset Strategy Shaping

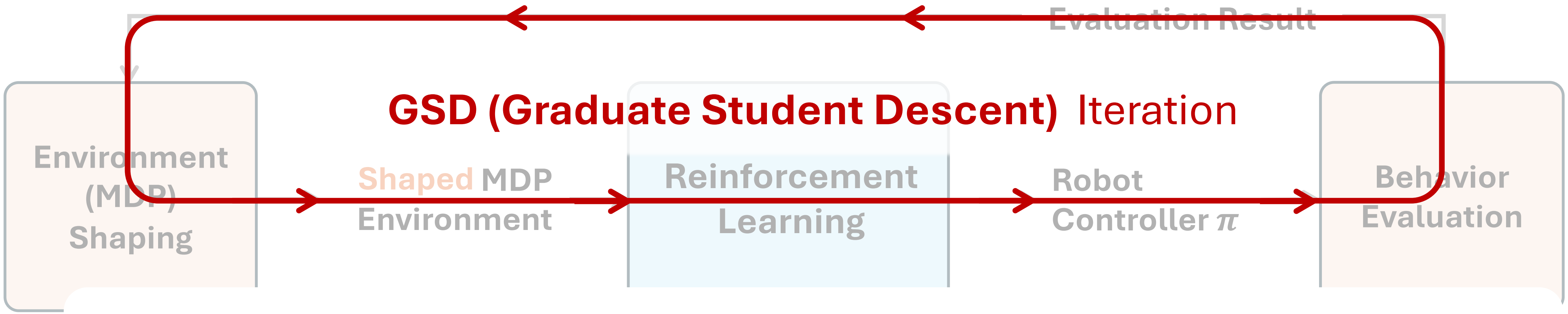
```
env.reset()
```

Curriculum Shaping

```
env.sample_goal()
```

Promise of RL vs Reality

One and Only Optimizer for this shaping problem:

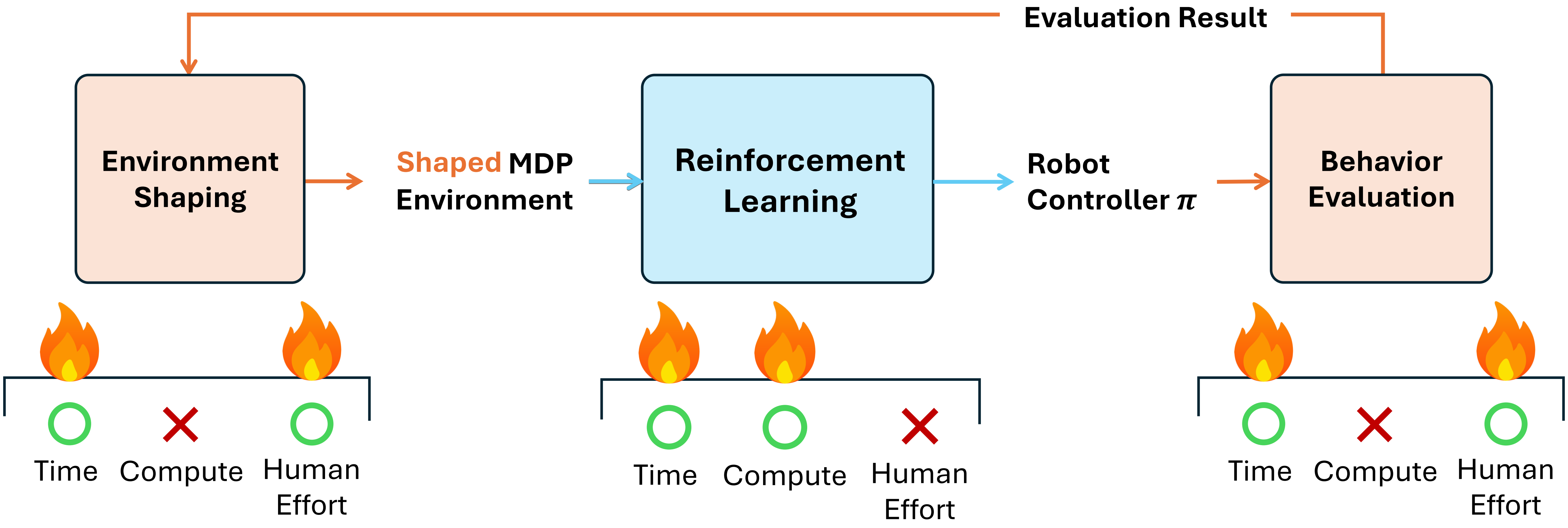


graduate student descent

Noun [edit]

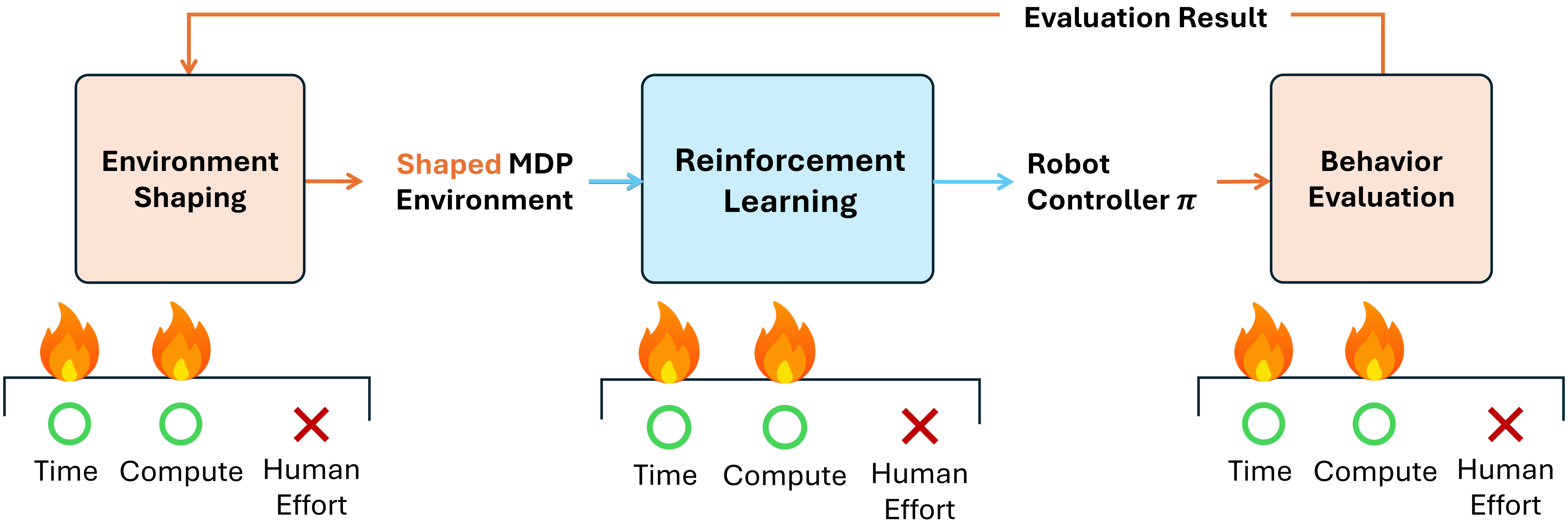
1. (*machine learning*, *humorous*) The process of choosing hyperparameters manually and in an ad-hoc manner, typical of work assigned to a graduate student.

[2] https://en.wiktionary.org/wiki/graduate_student_descent



We argue that:

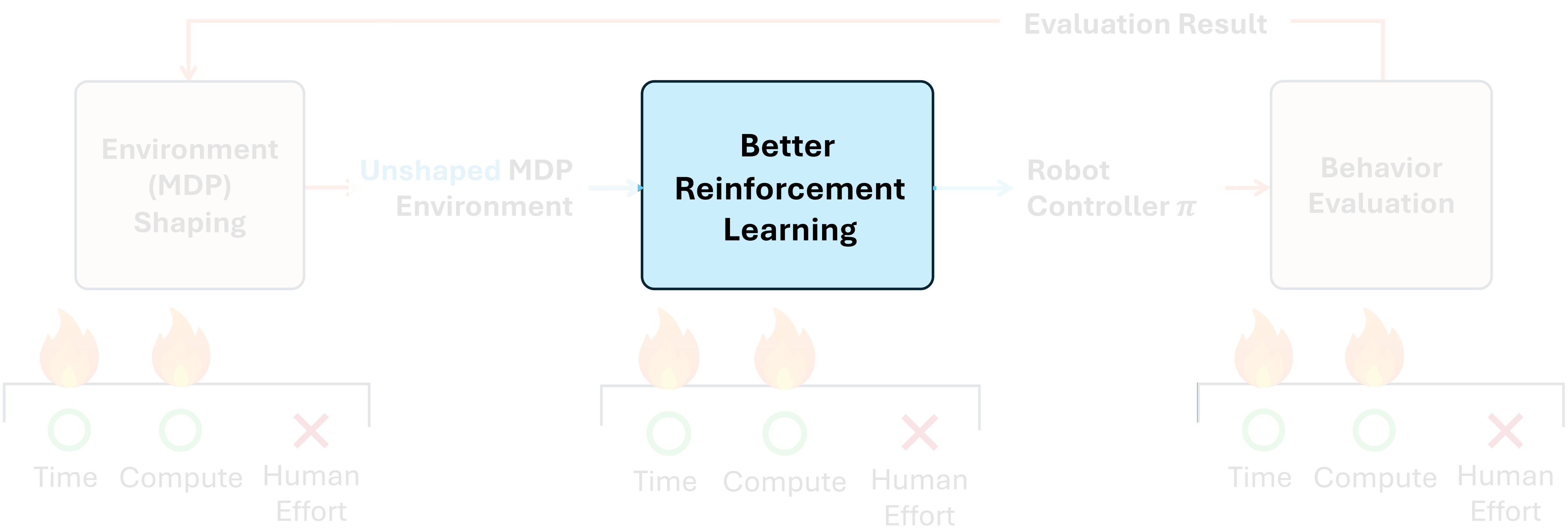
The community should focus on the following things to scale up the success of RL for many domains.



We argue that:

The community should focus on the following things to scale up the success of RL for many domains.

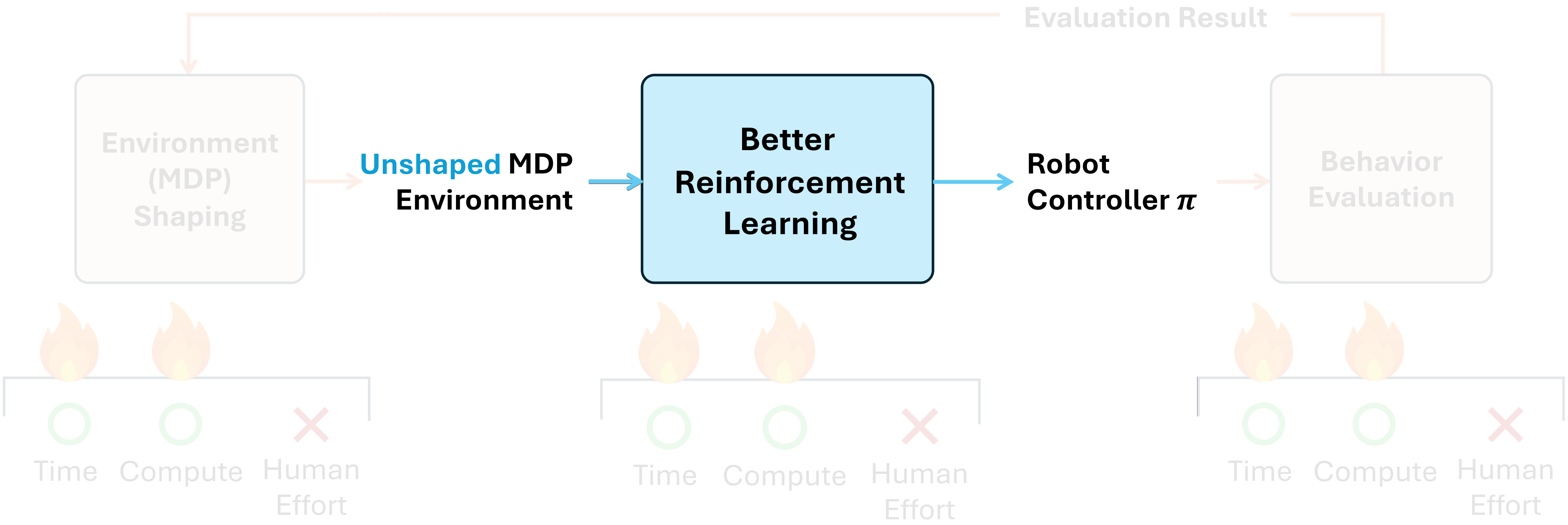
> 1. Prioritize research of **automating** the heuristic process of **Environment Shaping**.



We argue that:

The community should focus on the following things to scale up the success of RL for many domains.

> 1. Prioritize research of **automating** the heuristic process of **Environment Shaping**.
> 2. Develop **better RL algorithms** that doesn't require heuristic environment shaping in the first place.



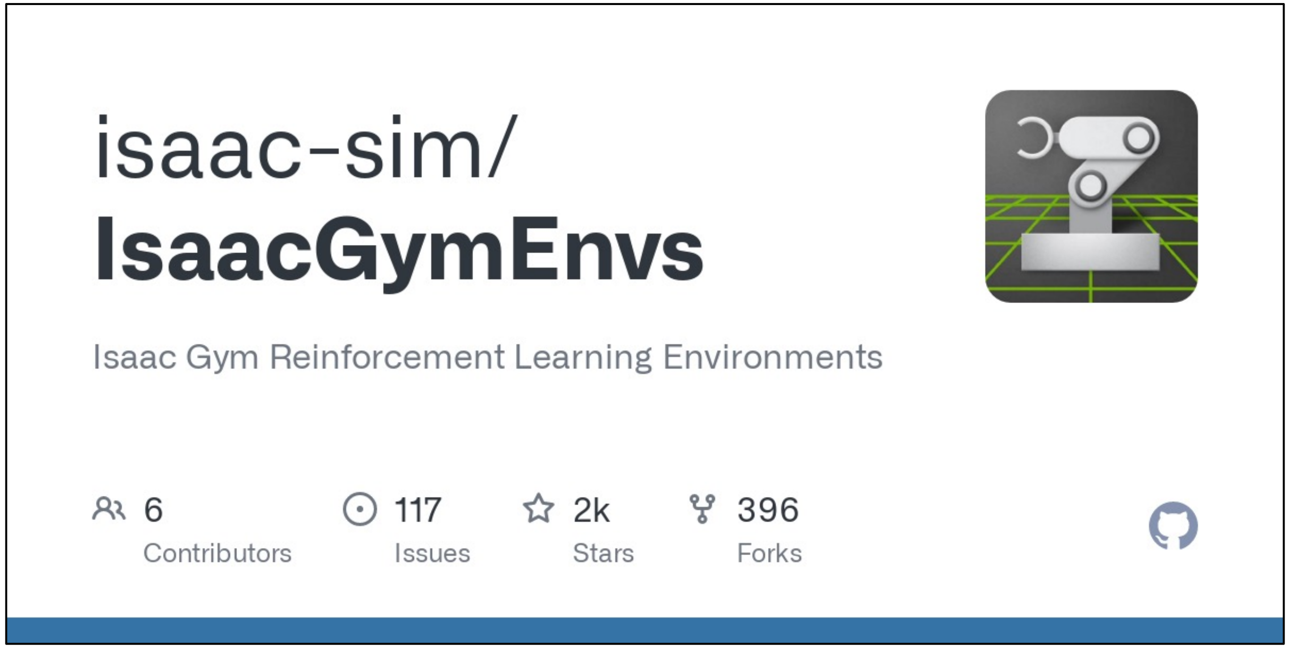
We argue that:

The community should focus on the following things to scale up the success of RL for many domains.

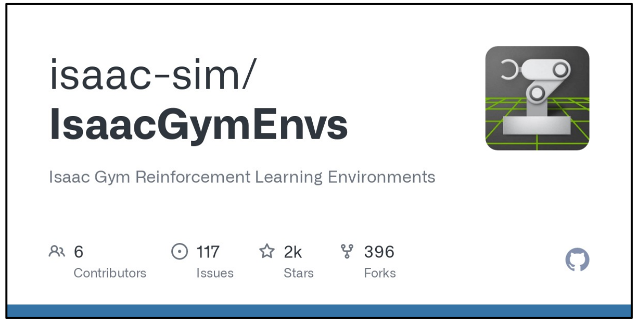
> 1. Prioritize research of **automating** the heuristic process of **Environment Shaping**.
> 2. Develop **better RL algorithms** that doesn't require heuristic environment shaping in the first place.
> 3. Benchmarking RL algorithms on **unshaped environments**.

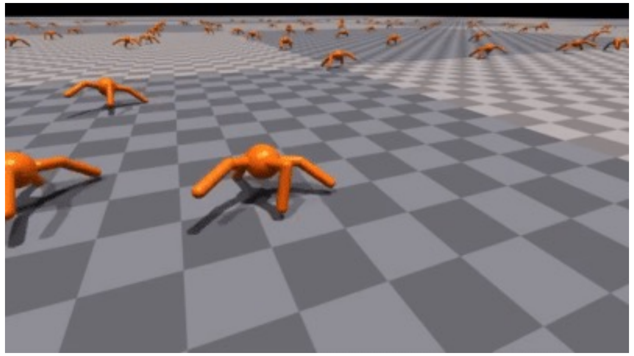
Examples of Environment Shaping

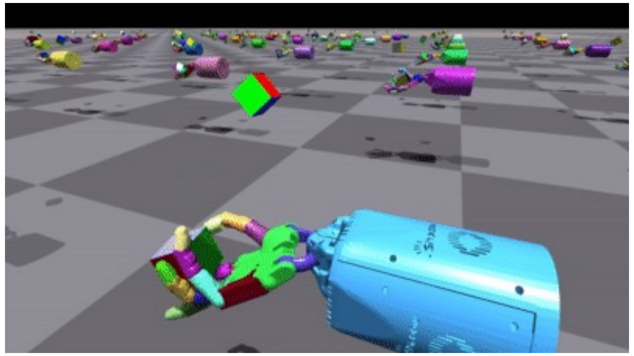
and how crucial they are to make RL work.

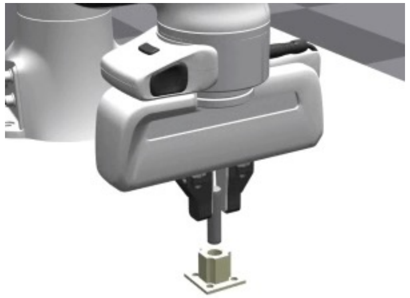


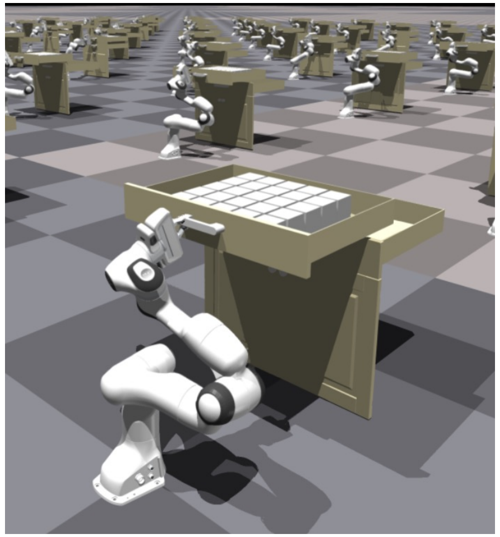
Examples of Environment Shaping

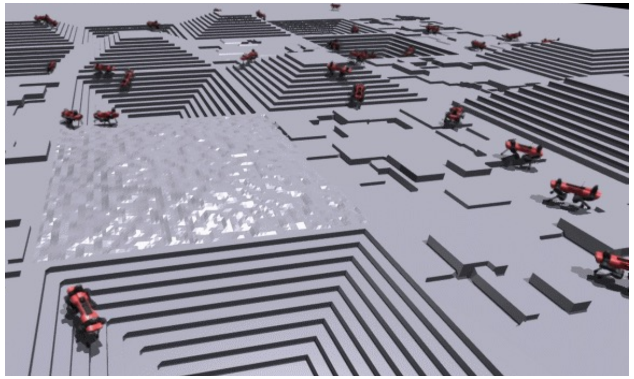


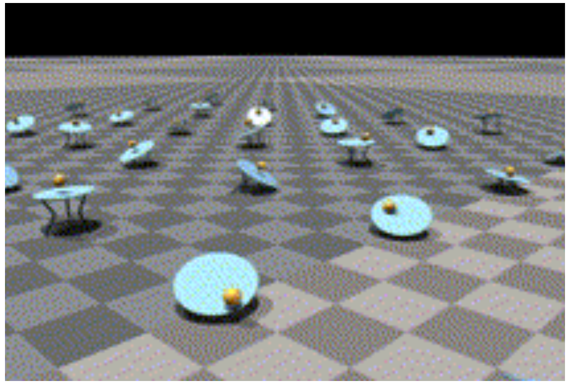








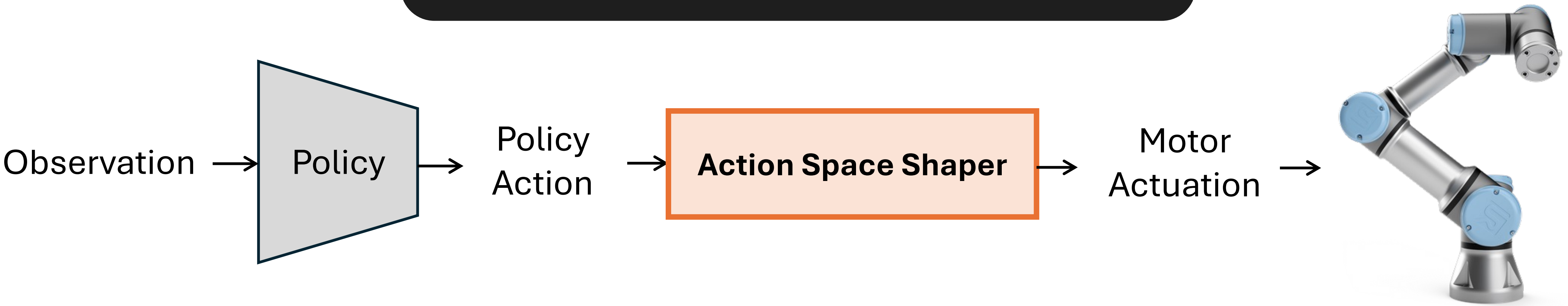




Examples of Environment Shaping

A. Action Space Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```

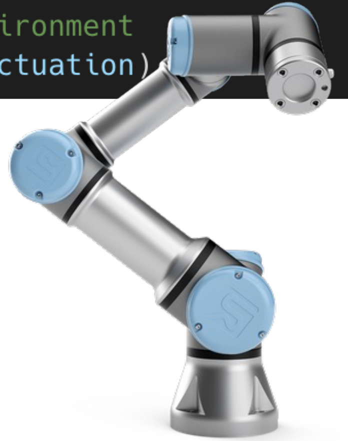


Examples of Environment Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```

A. Action Space Shaping

Observation → Policy → Policy Action → **Action Space Shaper** → Motor Actuation →



Unshaped Action Space

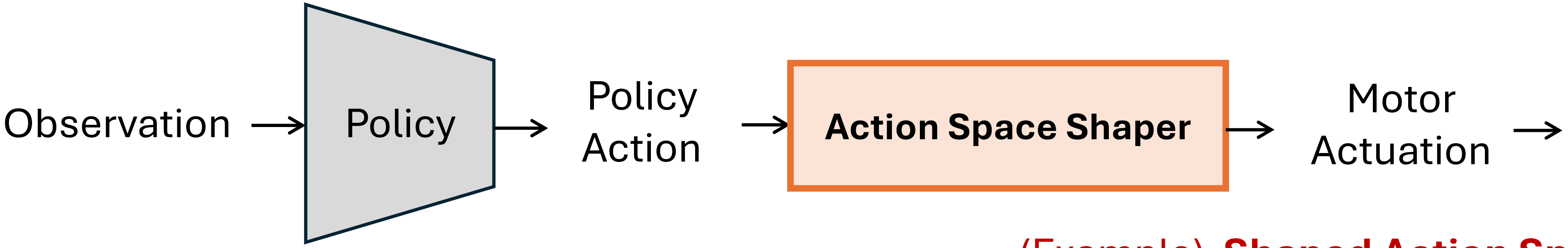
Letting the neural network policy directly predict acceptable motor commands

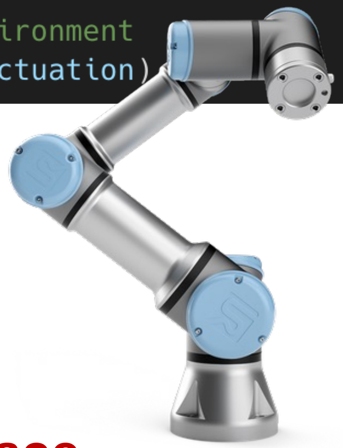
```
def get_actuation(self, res):
  # policy output will be directly
  #           sent as motor command
  return res
```

Examples of Environment Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```

A. Action Space Shaping





Unshaped Action Space

Letting the neural network policy directly predict acceptable motor commands

```
def get_actuation(self, res):
  # policy output will be directly
  #            sent as motor command
  return res
```

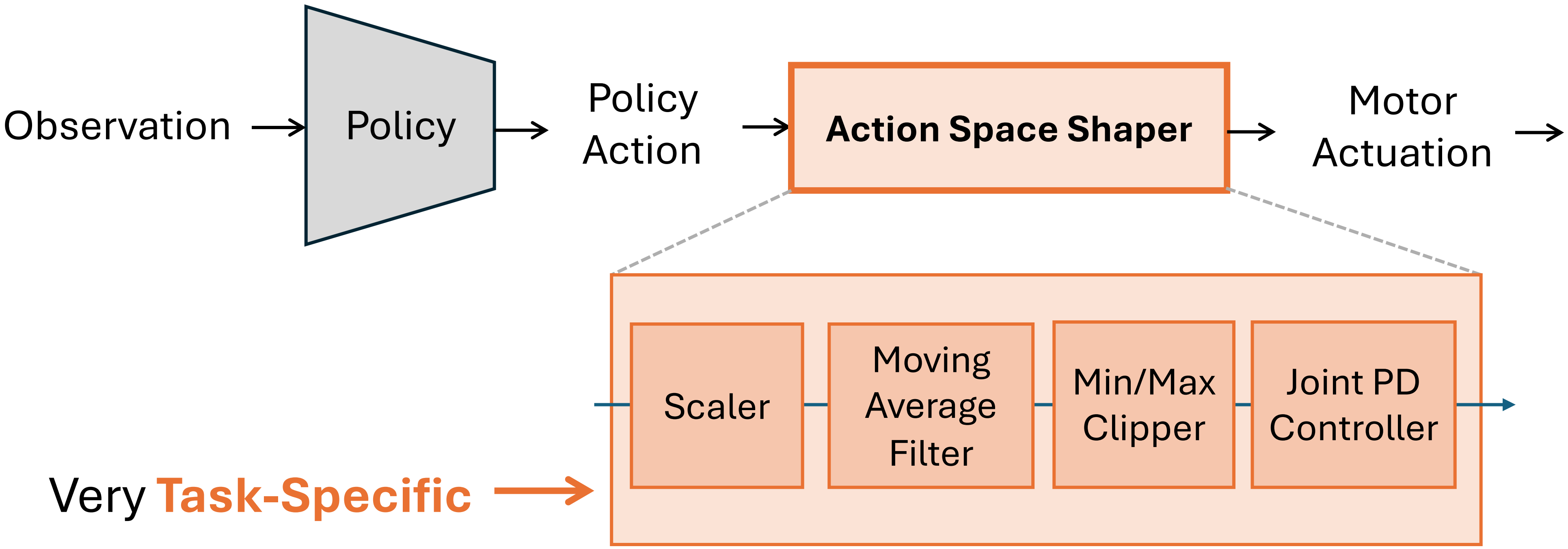
(Example) **Shaped Action Space**

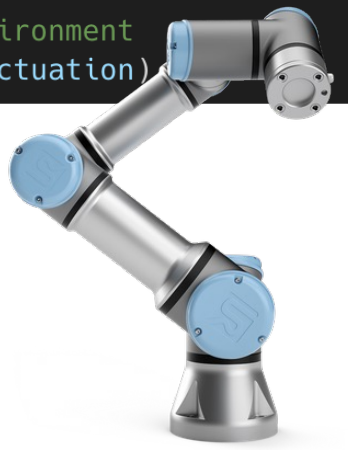
```
def get_actuation(self, res):

  # policy designed to predict absolute joint targets.

  # scale the policy output
  cur_target = scale(res, lower_bound, upper_bound)

  # compute the moving average of targets
  cur_target = coefs['alpha'] * cur_target
             + (1 - coefs['alpha']) * prev_target
  prev_target = cur_target

  # compute the motor torques with PD controller
  torques = coefs['pgain'] * (cur_target - self.dof_pos)
          - coefs['dgain'] * self.dof_vel

  return torques
```

Examples of Environment Shaping

A. Action Space Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```



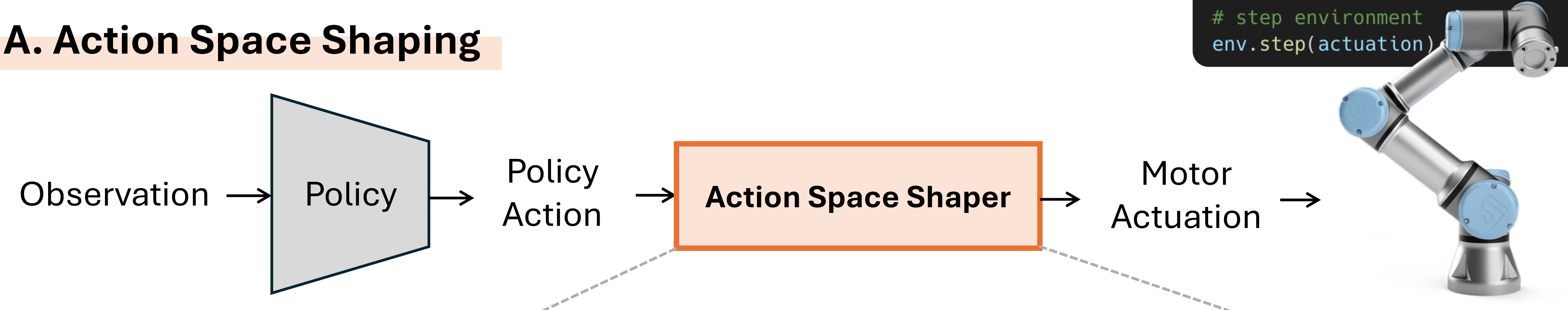


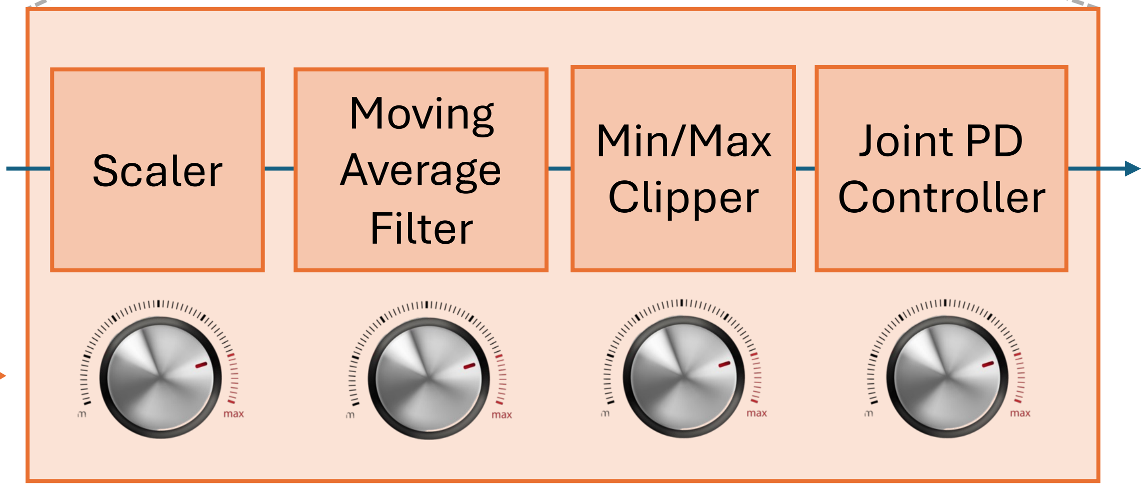
(Example) **Shaped Action Space**

Examples of Environment Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```

A. Action Space Shaping





It introduces all these **extra knobs** to tune!

(Example) **Shaped Action Space**

Examples of Environment Shaping

```
res = policy(obs)
actuation = env.get_actuation(res)

# step environment
env.step(actuation)
```

A. Action Space Shaping

Observation → Policy → Policy Action → **Action Space Shaper** → Motor Actuation →

It's a necessary evil; PPO cannot solve most tasks without action space shaping.

AllegroHand	Reward	Change
all shaped	38777	–
sparse reward	0	↓ 38777
unshaped action space	21530	↓ 17247
unshaped observation space	2114	↓ 36663
no early termination	0	↓ 38777
single initial state	0	↓ 38777
single goal state	141155	↑ 102378

Anymal	Reward	Change
all shaped	−45	–
sparse reward	−2789	↓ 2744
unshaped action space	−2499	↓ 2454
unshaped observation space	−2656	↓ 2611
no early termination	−43	↑ 2
single initial state	−17	↑ 28
single goal state	−2516	↓ 2470

Humanoid	Reward	Change
all shaped	7554	–
sparse reward	5237	↓ 2317
unshaped action space	67	↓ 7487
unshaped observation space	0	↓ 7554
no early termination	705	↓ 6849
single initial state	5735	↓ 1819

Examples of Environment Shaping

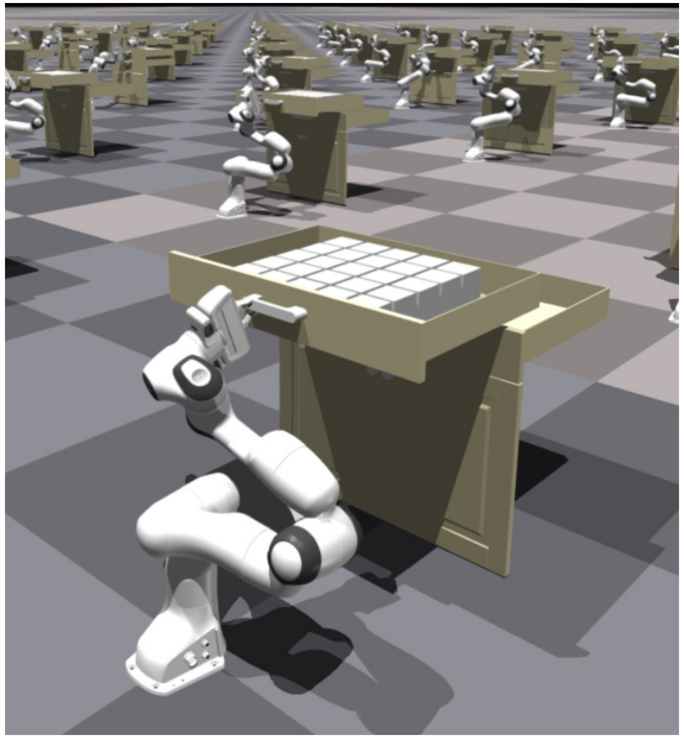
B. Observation Space Shaping

```
obs = env.get_observation()
```

Examples of Environment Shaping

```
obs = env.get_observation()
```

B. Observation Space Shaping



Unshaped Observation Space

Concatenation of entire raw oracle states.

```
def get_observation(self):

  obs = torch.cat([self.dof_pos, self.dof_vel,
                   self.drawer_state], dim = -1)
  return obs
```

Examples of Environment Shaping

```
obs = env.get_observation()
```

B. Observation Space Shaping

Entire oracle states
Available from simulation → **Observation Space Shaper** → Observation → Policy

Unshaped Observation Space

Concatenation of entire raw oracle states.

```
def get_observation(self):

  obs = torch.cat([self.dof_pos, self.dof_vel,
                   self.drawer_state], dim = -1)
  return obs
```

Shaped Observation Space

Concatenation of hand-crafted terms carefully engineered for this specific task.

```
def get_observation(self):

    # gripper pose
    ef_pose = forward_kinematics(self.joint_pos)

    # drawer handle pose
    handle_pose = self.drawer_handle_pose

    # vector between gripper and drawer handle
    vec = torch.inverse(ef_pose) @ handle_pose

    obs = torch.cat([scale(self.joint_pos),
                     scale(self.joint_vel),
                     scale(self.drawer_opened),
                     ef_pose, handle_pose, vec], dim = -1)
    return obs
```

Examples of Environment Shaping

```
obs = env.get_observation()
```

B. Observation Space Shaping

It's a necessary evil; PPO cannot solve most tasks without observation space shaping.

AllegroHand	Reward	Change
all shaped	38777	–
sparse reward	0	↓ 38777
unshaped action space	21530	↓ 17247
unshaped observation space	2114	↓ 36663
no early termination	0	↓ 38777
single initial state	0	↓ 38777
single goal state	141155	↑ 102378

Anymal	Reward	Change
all shaped	−45	–
sparse reward	−2789	↓ 2744
unshaped action space	−2499	↓ 2454
unshaped observation space	−2656	↓ 2611
no early termination	−43	↑ 2
single initial state	−17	↑ 28
single goal state	−2516	↓ 2470

Humanoid	Reward	Change
all shaped	7554	–
sparse reward	5237	↓ 2317
unshaped action space	67	↓ 7487
unshaped observation space	0	↓ 7554
no early termination	705	↓ 6849
single initial state	5735	↓ 1819

Examples of Environment Shaping

C. Terminal Condition Shaping

D. Reset Strategy Shaping

E. Curriculum Shaping

AllegroHand	Reward	Change
all shaped	38777	–
sparse reward	0	↓ 38777
unshaped action space	21530	↓ 17247
unshaped observation space	2114	↓ 36663
no early termination	0	↓ 38777
single initial state	0	↓ 38777
single goal state	141155	↑ 102378

Anymal	Reward	Change
all shaped	−45	–
sparse reward	−2789	↓ 2744
unshaped action space	−2499	↓ 2454
unshaped observation space	−2656	↓ 2611
no early termination	−43	↑ 2
single initial state	−17	↑ 28
single goal state	−2516	↓ 2470

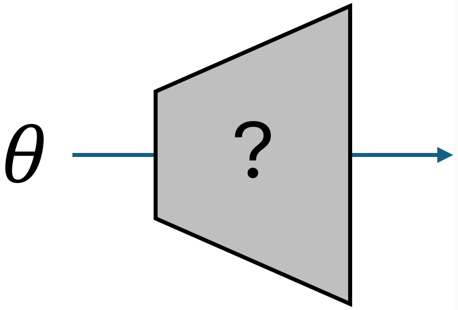
Humanoid	Reward	Change
all shaped	7554	–
sparse reward	5237	↓ 2317
unshaped action space	67	↓ 7487
unshaped observation space	0	↓ 7554
no early termination	705	↓ 6849
single initial state	5735	↓ 1819

Check out our paper to see more examples of environment shaping and how it's affecting the RL performance.

Examples of Environment Shaping

Automating Environment Shaping

How should we parametrize different environment shapers?



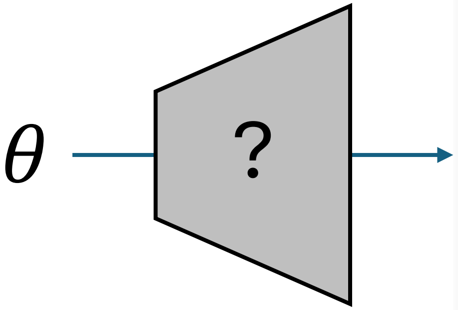

```
def get_observation(self):

    # gri
    ef_po

    # dra
    handl

    # vec
    vec =

    obs =


    retur
```

```
def get_actuation(self, res):

    # po

    # sc
    cur_

    # co
    cur_

    prev

    # co
    torq

    retu
```

```
def compute_reward(self):
    # Forward velocity reward
    forward_velocity = self.center_vel[2]
    reward_forward_velocity = self.coef['forward_velocity'] * forward_velocit

    # Goal velocity alignment
    velocity_diff = torch.norm(self.center_vel - self.goal_vel)
    reward_goal_velocity = -self.coef['goal_velocity'] * velocity_diff

    # Energy efficiency reward (negative of the sum of absolute velocities)
    energy_efficiency = torch.sum(torch.abs(self.dof_vel))
    reward_energy_efficiency = -self.coef['energy_efficiency'] * energy_effic

    # Joint acceleration penalty
    joint_acceleration = torch.norm(self.dof_vel[1:] - self.dof_vel[:-1])
    reward_joint_acceleration_penalty = -self.coef['joint_acceleration_penalt
joint_acceleration
```

Automating Environment Shaping

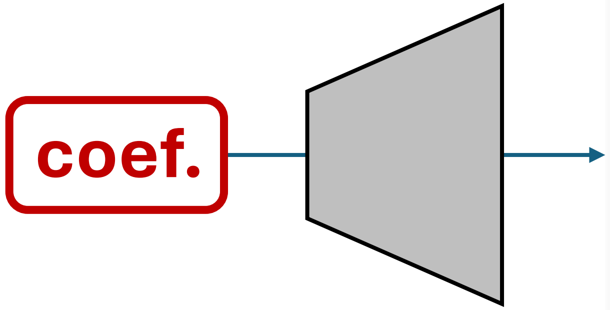
How should we parametrize different environment shapers?




```python
def compute_reward(self):
    # Forward velocity reward
    forward_velocity = self.center_vel[2]
    reward_forward_velocity = self.coef['forward_velocity'] * forward_velocity

    # Goal velocity alignment
    velocity_diff = torch.norm(self.center_vel - self.goal_vel)
    reward_goal_velocity = -self.coef['goal_velocity'] * velocity_diff

    # Energy efficiency reward (negative of the sum of absolute velocities)
    energy_efficiency = torch.sum(torch.abs(self.dof_vel))
    reward_energy_efficiency = -self.coef['energy_efficiency'] * energy_efficiency

    # Joint acceleration penalty
    joint_acceleration = torch.norm(self.dof_vel[1:] - self.dof_vel[:-1])
    reward_joint_acceleration_penalty = -self.coef['joint_acceleration_penalty'] *
joint_acceleration

    # Joint velocity penalty
    joint_velocity_penalty = torch.norm(self.dof_vel)
    reward_joint_velocity_penalty = -self.coef['joint_velocity_penalty'] *
joint_velocity_penalty
```

Automating Environment Shaping

How should we parametrize different environment shapers?



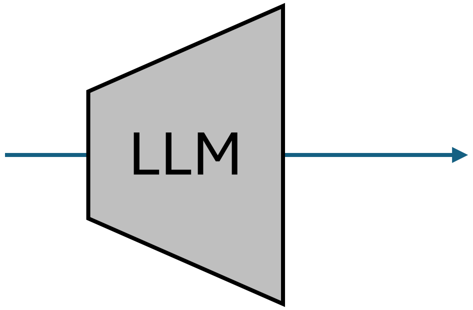

```
def compute_reward(self):
    # Forward velocity reward
    forward_velocity = self.center_vel[2]
    reward_forward_velocity = self.coef['forward_velocity'] * forward_velocity

    # Goal velocity alignment
    velocity_diff = torch.norm(self.center_vel - self.goal_vel)
    reward_goal_velocity = -self.coef['goal_velocity'] * velocity_diff

    # Energy efficiency reward (negative of the sum of absolute velocities)
    energy_efficiency = torch.sum(torch.abs(self.dof_vel))
    reward_energy_efficiency = -self.coef['energy_efficiency'] * energy_efficiency

    # Joint acceleration penalty
    joint_acceleration = torch.norm(self.dof_vel[1:] - self.dof_vel[:-1])
    reward_joint_acceleration_penalty = -self.coef['joint_acceleration_penalty'] *
joint_acceleration

el)
int_velocity_penalty'] *
joint_velocity_penalty
```

What if we want to try out different reward terms?

Automating **Environment Shaping**

How should we parametrize different environment shapers?

Prompt



Optimizing Environment Shaping as a ***Code Generation Problem***

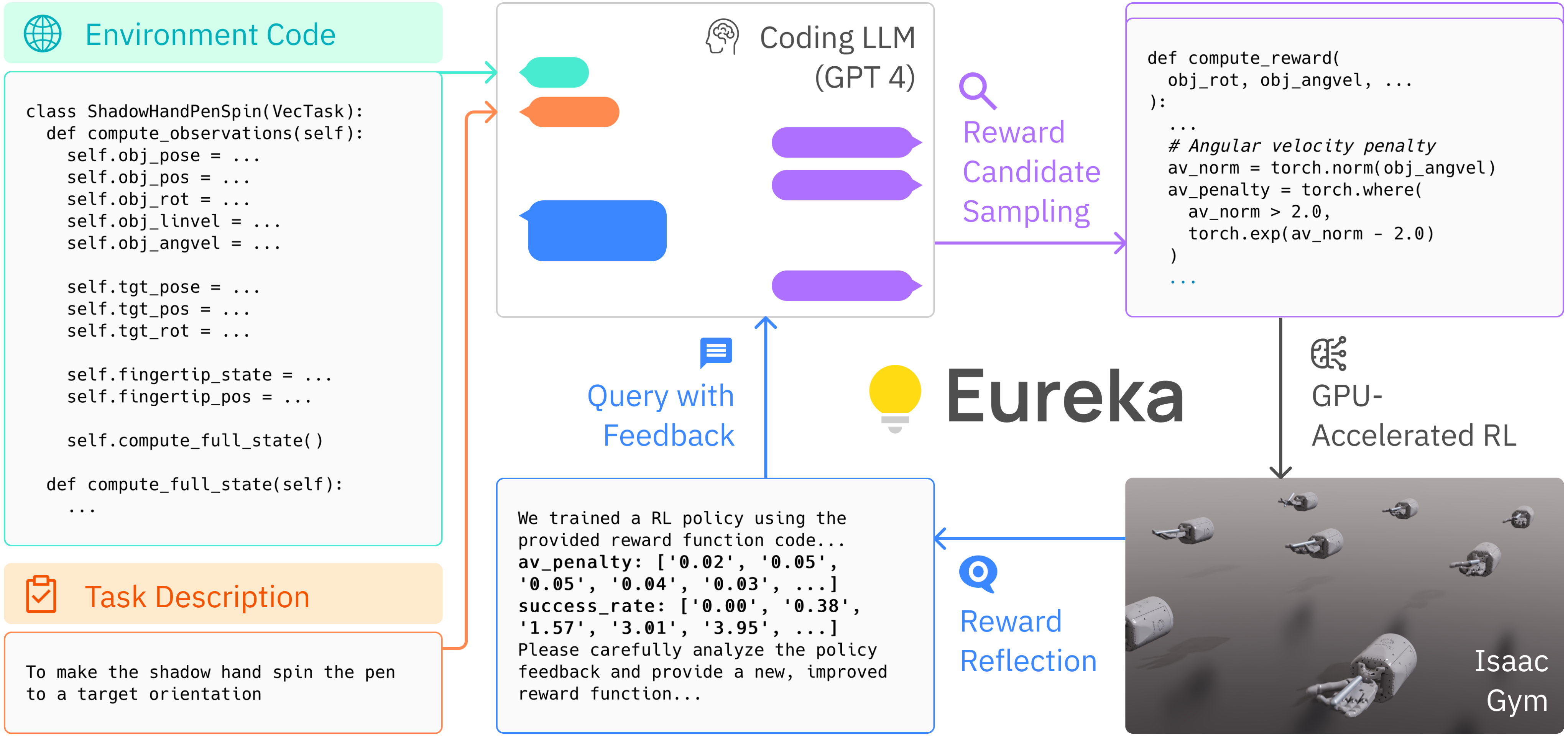
```python
def get_observation(self):

    # gripper pose
    ef_pose = forward_kinematics(self.joint_pos)

    # drawer handle pose
    handle_pose = self.drawer_handle_pose

    # vector between gripper and drawer handle
    vec = torch.inverse(ef_pose) @ handle_pose

    obs = torch.cat([scale(self.joint_pos),
                     scale(self.joint_vel),
                     scale(self.drawer_opened),
                     ef_pose, handle_pose, vec], dim = -1)
    return obs
```

```python
def get_actuation(self, res):

  # policy designed to predict absolute joint targets.

  # scale the policy output
  cur_target = scale(res, lower_bound, upper_bound)

  # compute the moving average of targets
  cur_target = coe
             + (
  prev_target = cu

  # compute the mo
  torques = coefs[                                  f_pos)
          - coefs['dgain'] * self.dof_vel

  return torques
```

```python
def compute_reward(self):
    # Forward velocity reward
    forward_velocity = self.center_vel[2]
    reward_forward_velocity = self.coef['forward_velocity'] * forward_velocity

    # Goal velocity alignment
    velocity_diff = torch.norm(self.center_vel - self.goal_vel)
    reward_goal_velocity = -self.coef['goal_velocity'] * velocity_diff

    # Energy efficiency reward (negative of the sum of absolute velocities)
    energy_efficiency = torch.sum(torch.abs(self.dof_vel))
    reward_energy_efficiency = -self.coef['energy_efficiency'] * energy_efficiency

    # Joint acceleration penalty
    joint_acceleration = torch.norm(self.dof_vel[1:] - self.dof_vel[:-1])
    reward_joint_acceleration_penalty = -self.coef['joint_acceleration_penalty'] *
joint_acceleration

    # Joint velocity penalty
    joint_velocity_penalty = torch.norm(self.dof_vel)
    reward_joint_velocity_penalty = -self.coef['joint_velocity_penalty'] *
joint_velocity_penalty

    # Fall penalty (if the center of mass is too low)
    fall_penalty = 0.0
    if self.com_height < 0.5:
        fall_penalty = self.coef['fall_penalty']
    reward_fall_penalty = -fall_penalty

    # Upright bonus
    upright_bonus = self.dof_pos[0]
    reward_upright_bonus = self.coef['upright_bonus'] * upright_bonus

    # Smoothness bonus (negative of the jerk)
    jerk = torch.norm(self.dof_vel[2:] - 2*self.dof_vel[1:-1] + self.dof_vel[:-2])
    reward_smoothness_bonus = -self.coef['smoothness_bonus'] * jerk

        reward_energy_efficiency +
        reward_joint_acceleration_penalty +
        reward_joint_velocity_penalty +
        reward_fall_penalty +
        reward_upright_bonus +
        reward_smoothness_bonus
    )

    return total_reward
```

Representing Environment Shaping with ***Python Code***

Automating Environment Shaping



Ma et al., *Eureka: Human-level reward design via coding large language models* (2023)

Automating Environment Shaping

Can we extend the LLM-based automation paradigm to other environment components?

shaping component	Eureka (Ma et al., 2023)	Human Design (Makoviychuk et al., 2021)	Automation Performance
*reward	0.986	0.973	$\uparrow$ 0.013
†observation	0.967	0.973	$\downarrow$ 0.006
†action	0.982	0.973	$\uparrow$ 0.009
$^{\circ}$reward $\times$ observation	0.196	0.973	$\downarrow$ 0.777
$^{\circ}$reward $\times$ action	0.536	0.973	$\downarrow$ 0.437
$^{\circ}$reward $\times$ observation $\times$ action	N/A	0.973	N/A

Automating Environment Shaping

Can we extend the LLM-based automation paradigm to other environment components?

shaping component	Eureka (Ma et al., 2023)	Human Design (Makoviychuk et al., 2021)	Automation Performance
*reward	0.986	0.973	↑ 0.013
†observation	0.967	0.973	↓ 0.006
†action	0.982	0.973	↑ 0.009
°reward × observation	0.196	0.973	↓ 0.777
°reward × action	0.536	0.973	↓ 0.437
°reward × observation × action	N/A	0.973	N/A

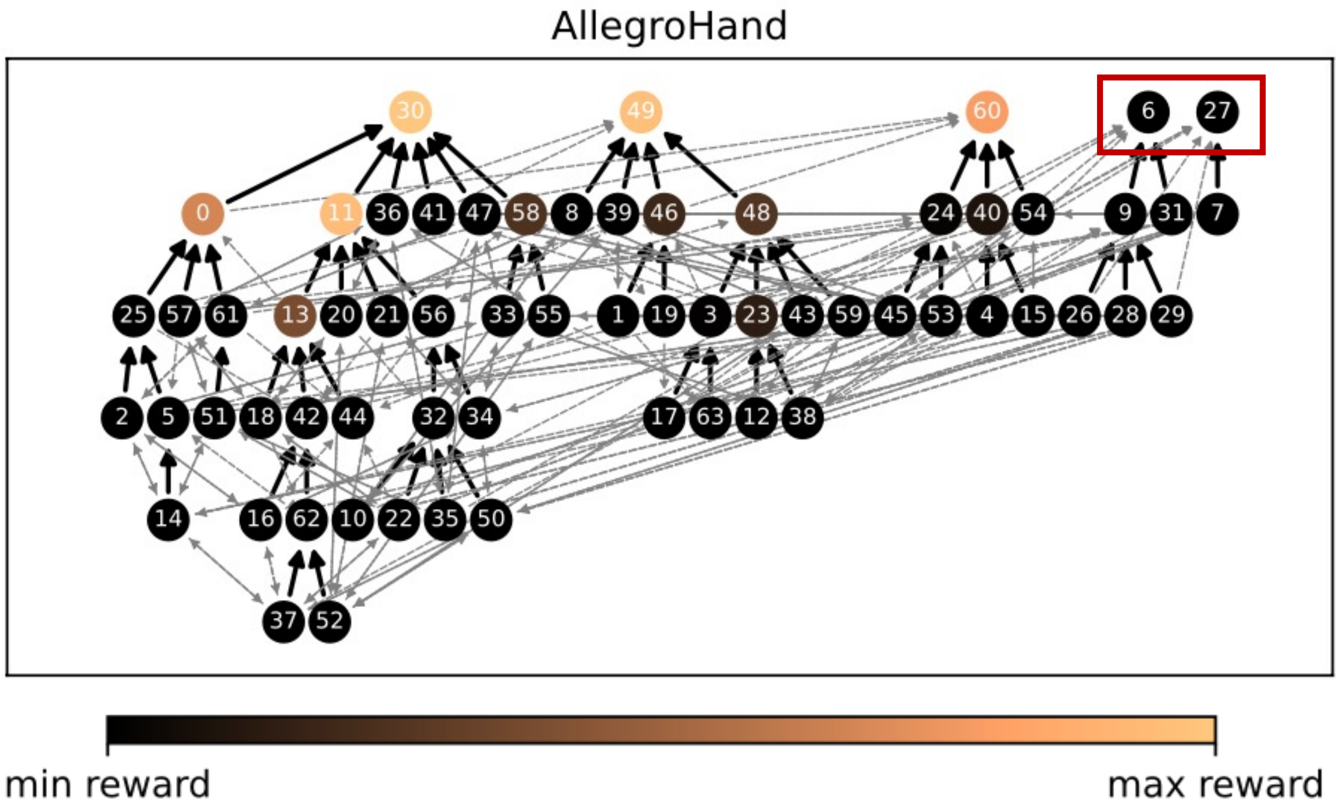
Automating Environment Shaping

Can we extend the LLM-based automation paradigm to other environment components?

shaping component	Eureka (Ma et al., 2023)	Human Design (Makoviychuk et al., 2021)	Automation Performance
*reward	0.986	0.973	$\uparrow$ 0.013
†observation	0.967	0.973	$\downarrow$ 0.006
†action	0.982	0.973	$\uparrow$ 0.009
$^{\circ}$reward $\times$ observation	0.196	0.973	$\downarrow$ 0.777
$^{\circ}$reward $\times$ action	0.536	0.973	$\downarrow$ 0.437
$^{\circ}$reward $\times$ observation $\times$ action	N/A	0.973	N/A

Automating Environment Shaping

But we need to shape the entire environment as its entirety;



Sequential Optimization

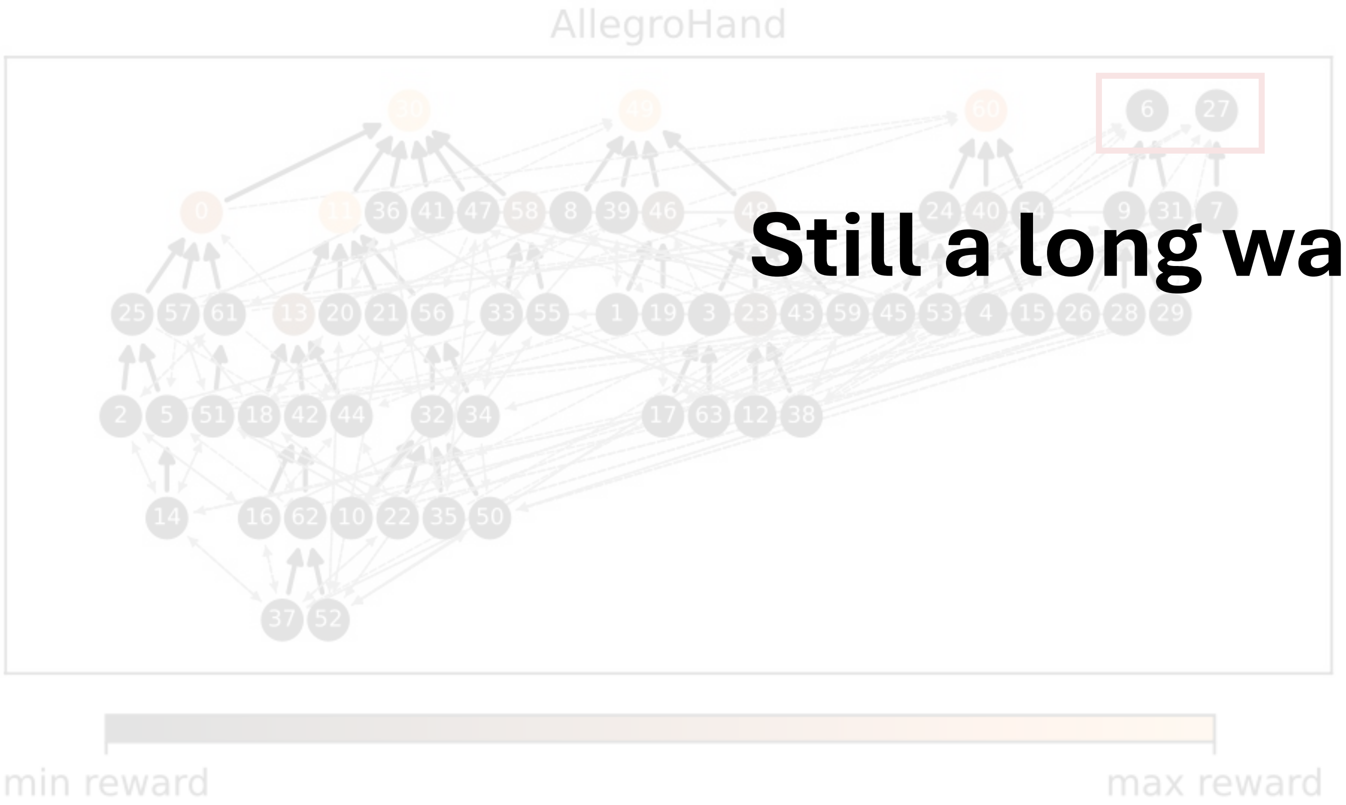
(i.e., shape the reward →
shape the action space →
shape the observation space)

leads to Local Optima!

Automating Environment Shaping

Shaping the entire environment is much harder than shaping one component.

but the optimization landscape is non-convex, we need joint optimization.



Still a long way to go!

Sequential Optimization

(i.e., shape the reward →
shape the action space →
shape the observation space)

leads to Local Optima!

Path forward

We argued that:

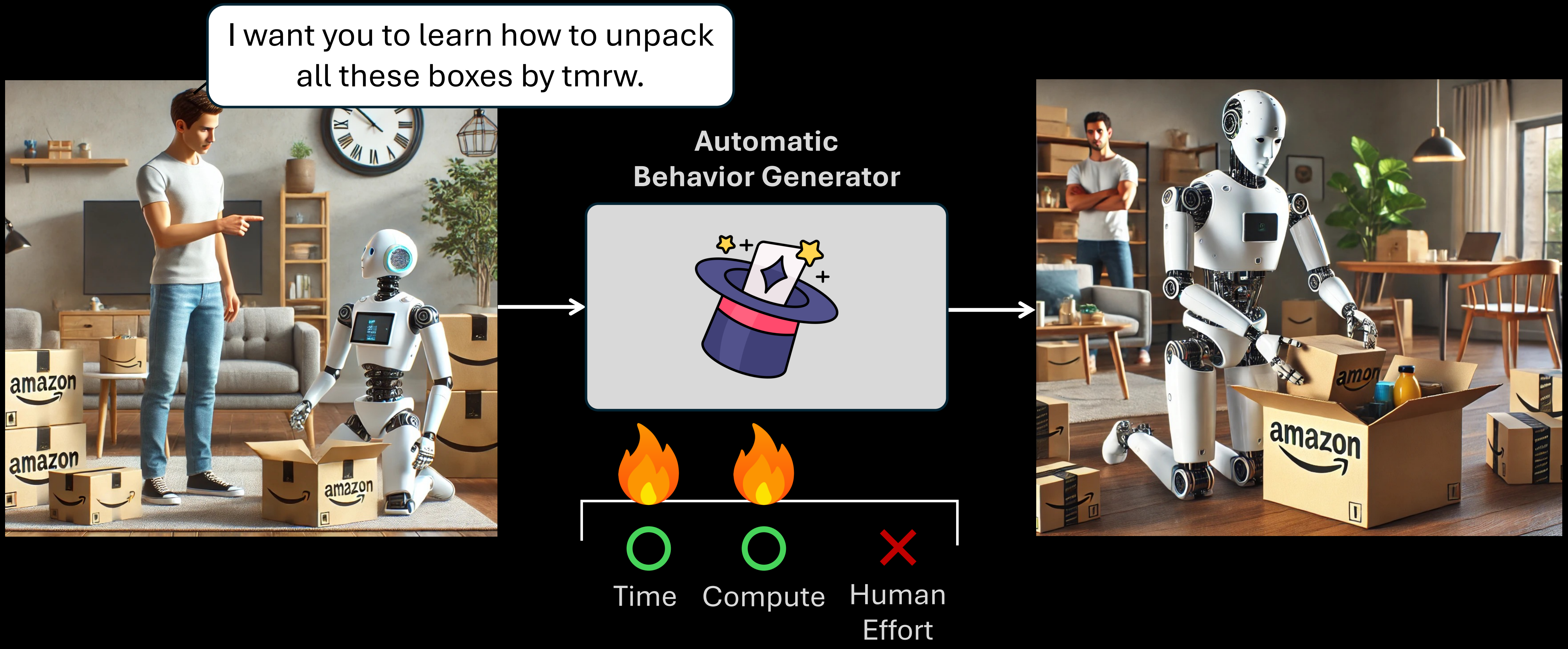
The community should focus on the following things to scale up the success of RL:

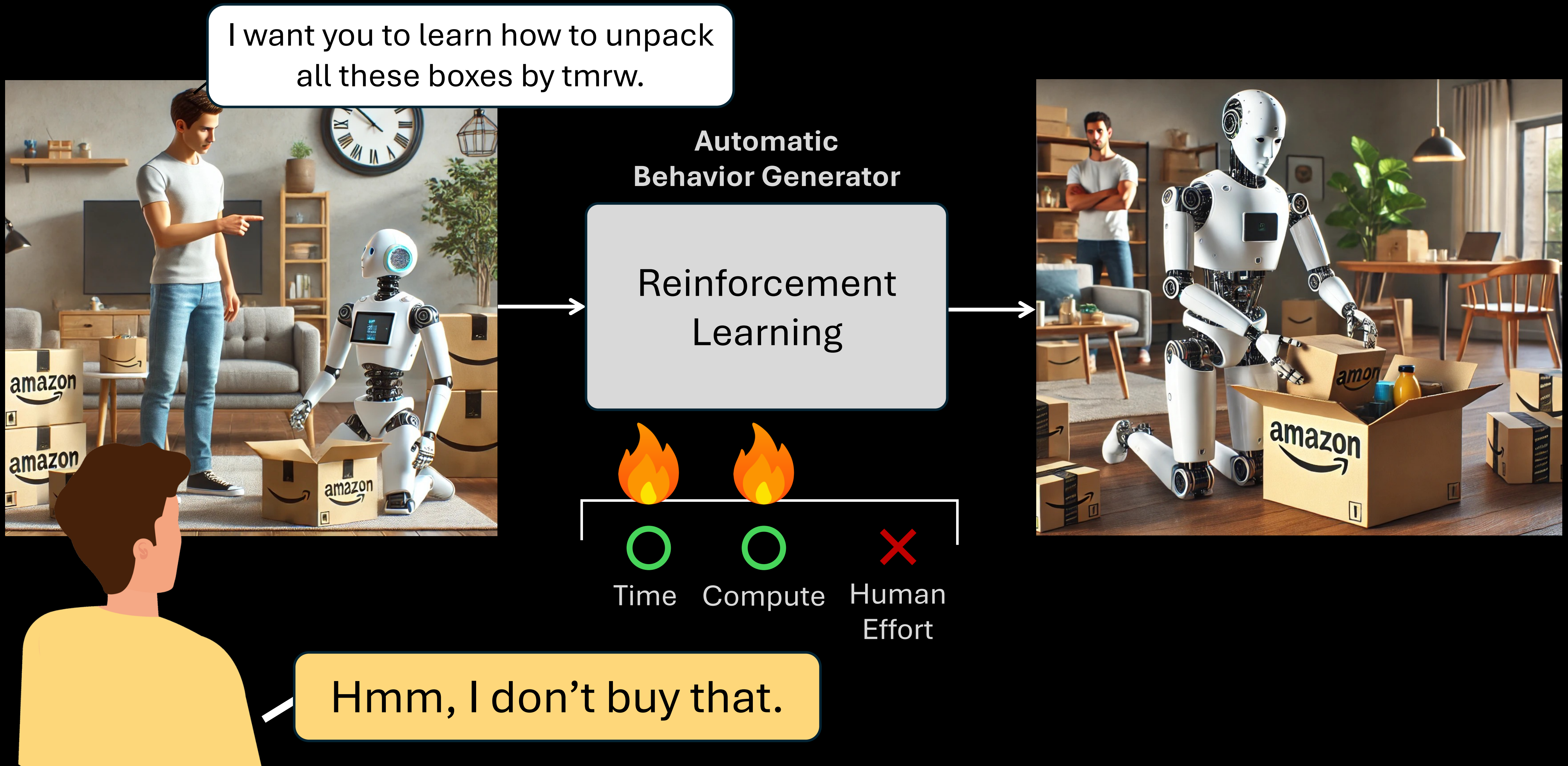
> 1. Prioritize research of **automating** the heuristic process of **Environment Shaping**.
> 2. Develop **better RL algorithms** that doesn't require heuristic environment shaping in the first place.
> 3. Benchmarking RL algorithms on **unshaped environments**.

EnvCoderBench

- Collection of **unshaped** robotics RL environments
- Easy-to-use API for LLM-based automated shaping
- Support for parallel RL training of multiple shaping candidates for efficient evaluation.

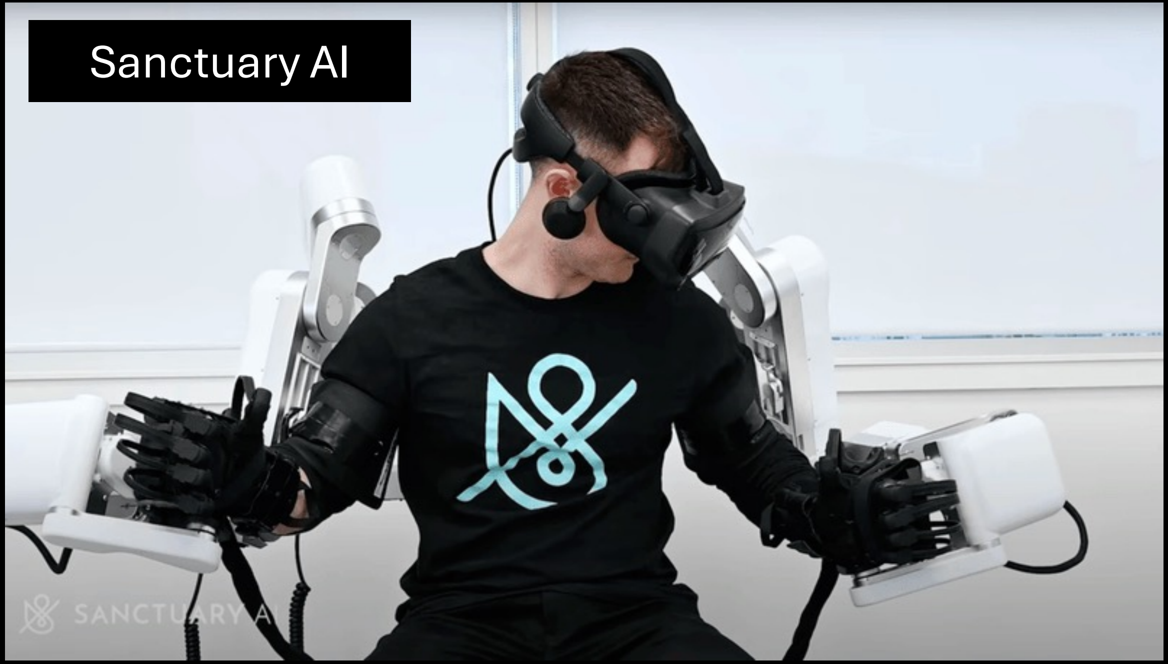
Opposing Positions

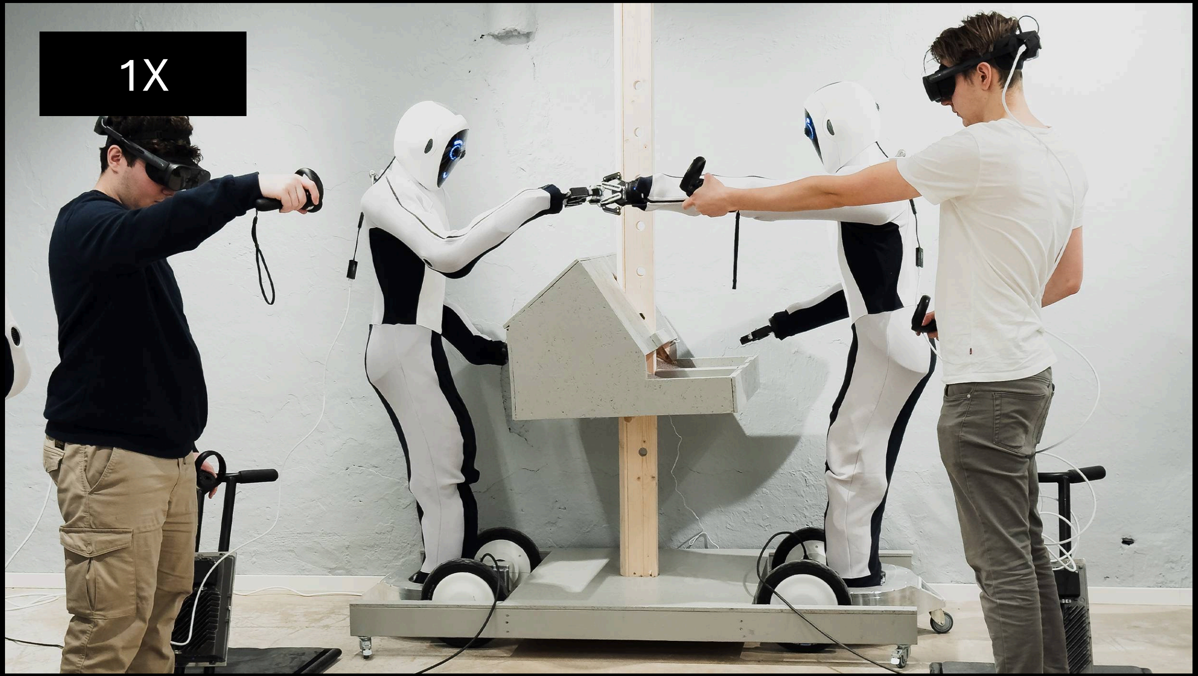

I want you to learn how to unpack all these boxes by tmrw.
Automatic
Behavior Generator
Time
Compute
Human
Effort
amazon

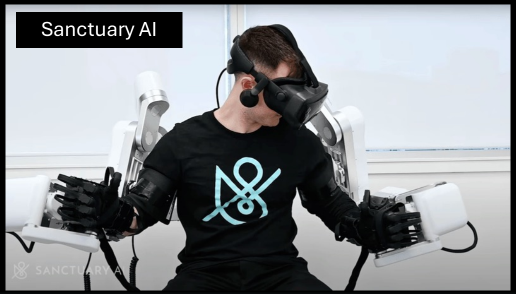

I want you to learn how to unpack all these boxes by tmrw.
Automatic Behavior Generator
Reinforcement Learning
Time
Compute
Human Effort
Hmm, I don't buy that.

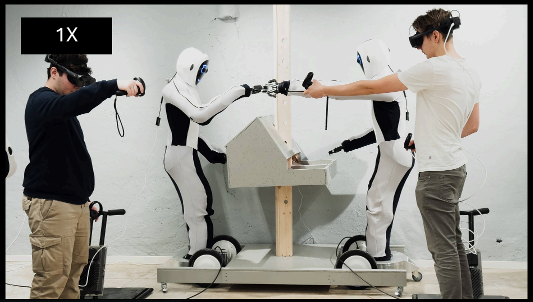

4x teleoperated
Google DeepMind
Improved Gripper Responsiveness


Tesla


Sanctuary AI
SANCTUARY AI


1X





~~Reinforcement Learning~~

Robotic Foundation Model

Time

Compute

Human Effort

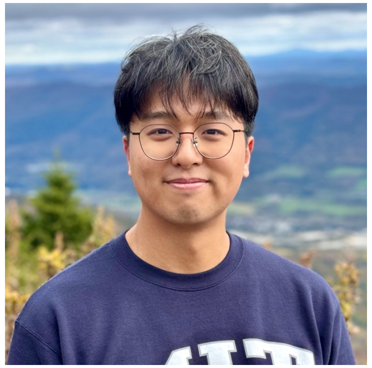
We still believe in the **power of RL** as a tool to generate robust, generalizable, super-human behaviors that cannot be easily achieved with Imitation Learning.

The behaviors generated by RL can also be another data source to train those foundation models;

Making RL easier to use will be the start of a **virtuous data cycle for embodied intelligence**.

Position: Automatic Environment Shaping is the Next Frontier in RL

Younghyo Park*, Gabriel B. Margolis*, and Pulkit Agrawal



ICML International Conference On Machine Learning

Massachusetts Institute of Technology

Improbable AI