

# How to train your robot

Modular robot learning for autonomous visual navigation

Michel Aractingi, Gianluca Monaci, Tomi Silander  
NAVER LABS Europe

# CONTENTS

1. Introduction
2. Overview of Robot Learning
3. Modular Reinforcement Learning
4. Visual Navigation Experiments
5. Concluding Remarks



# 1. Introduction

# 1.1 Mobile Robots

## NAVER HQ2: “project 1784”

- World's 1<sup>st</sup> robot-friendly building
- Robots provide all sorts of services
- Robots move in complex environments, interact with people, other robots and infrastructure



# 1.1 Mobile Robots

So these robots will have to be very smart, right?

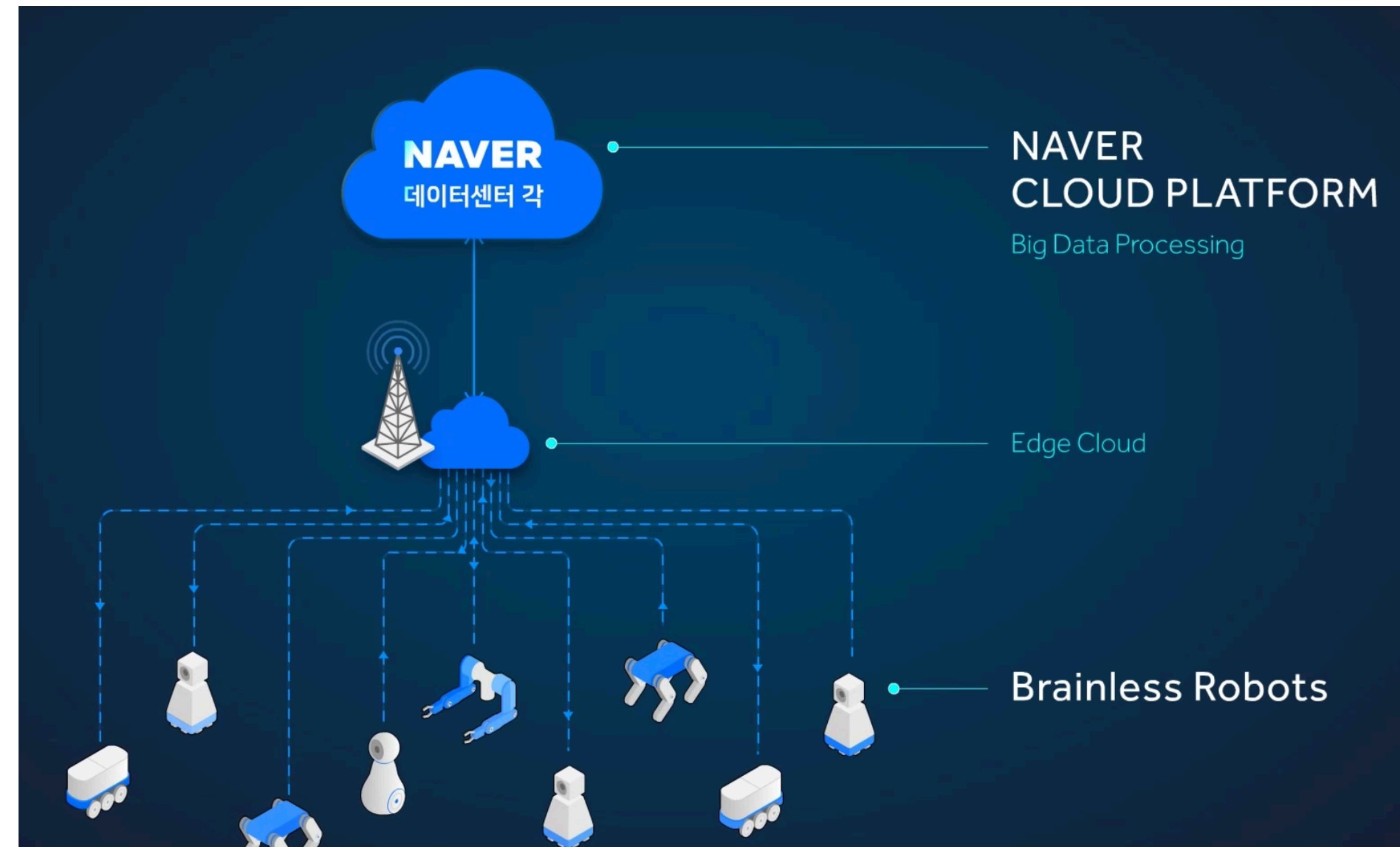


# 1.1 Mobile Robots

So these robots will have to be very smart, right?

## Brainless Robots

- Limited computation in robot, most intelligence in the cloud





# 1.1 Mobile Robots

## AROUND

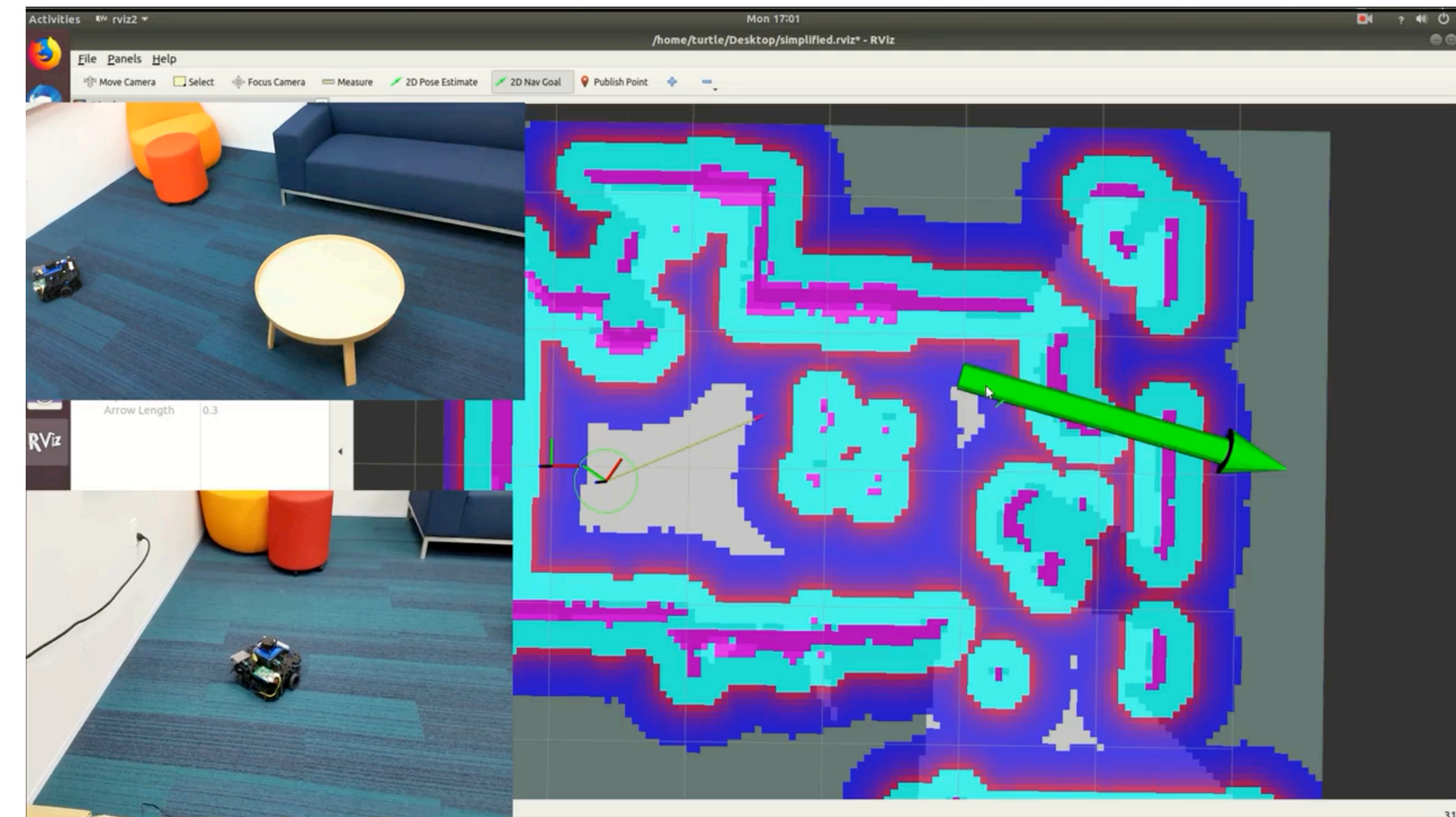
- NAVER LABS mobile robot platform
- Can navigate in complex, crowded environments
- Uses only RGBD cameras, without Lidar
- Guidance, delivery services and many more



# 1.2 Robot Navigation

Accurately modelled robot navigates using Lidar in known, precisely mapped, static environment

- Robot navigates on the map



<https://navigation.ros.org>



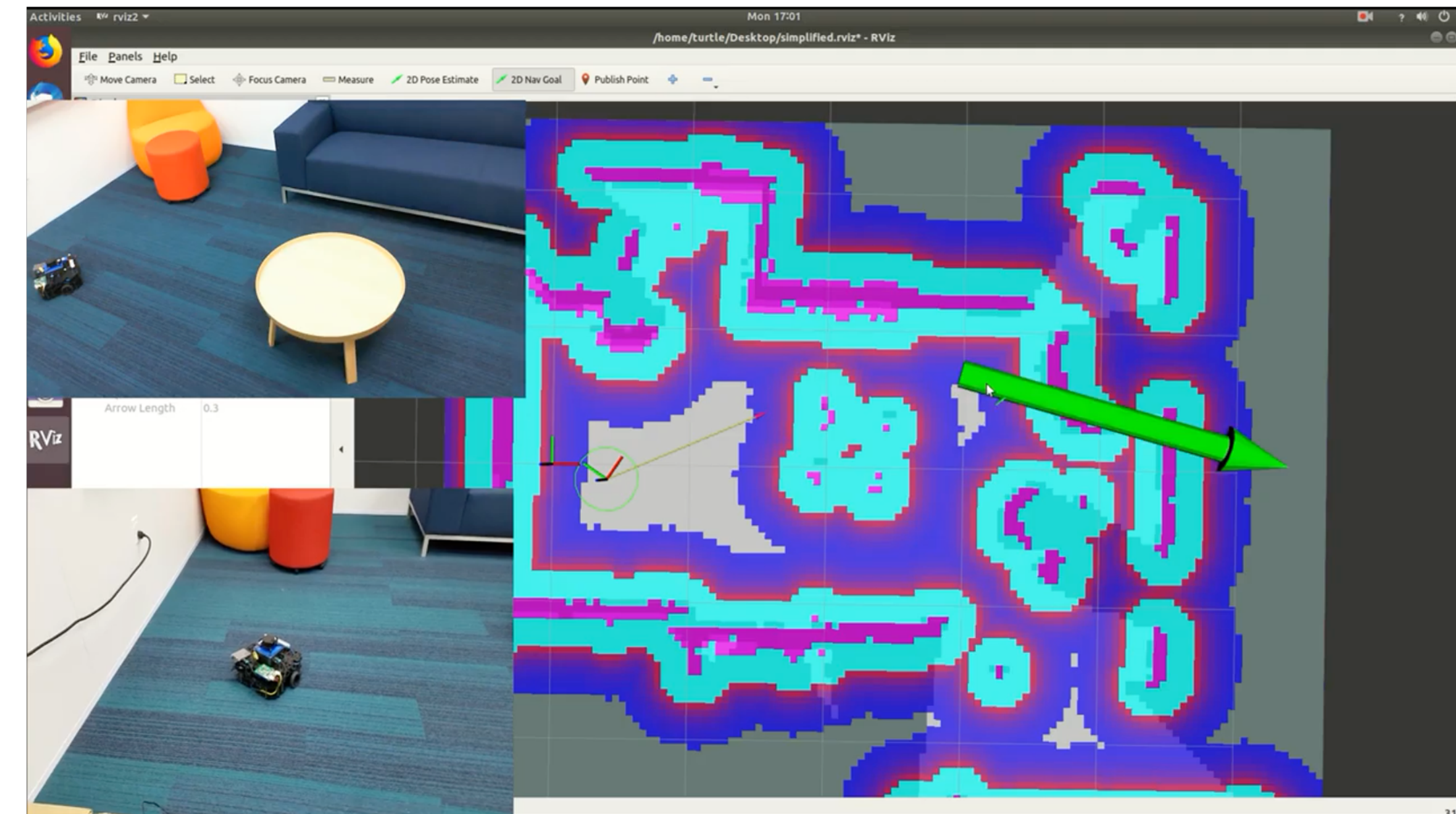
# 1.2 Robot Navigation

Accurately modelled robot navigates using Lidar in known, precisely mapped, static environment

- Robot navigates on the map

## Issues

- Known robot: impractical



<https://navigation.ros.org>

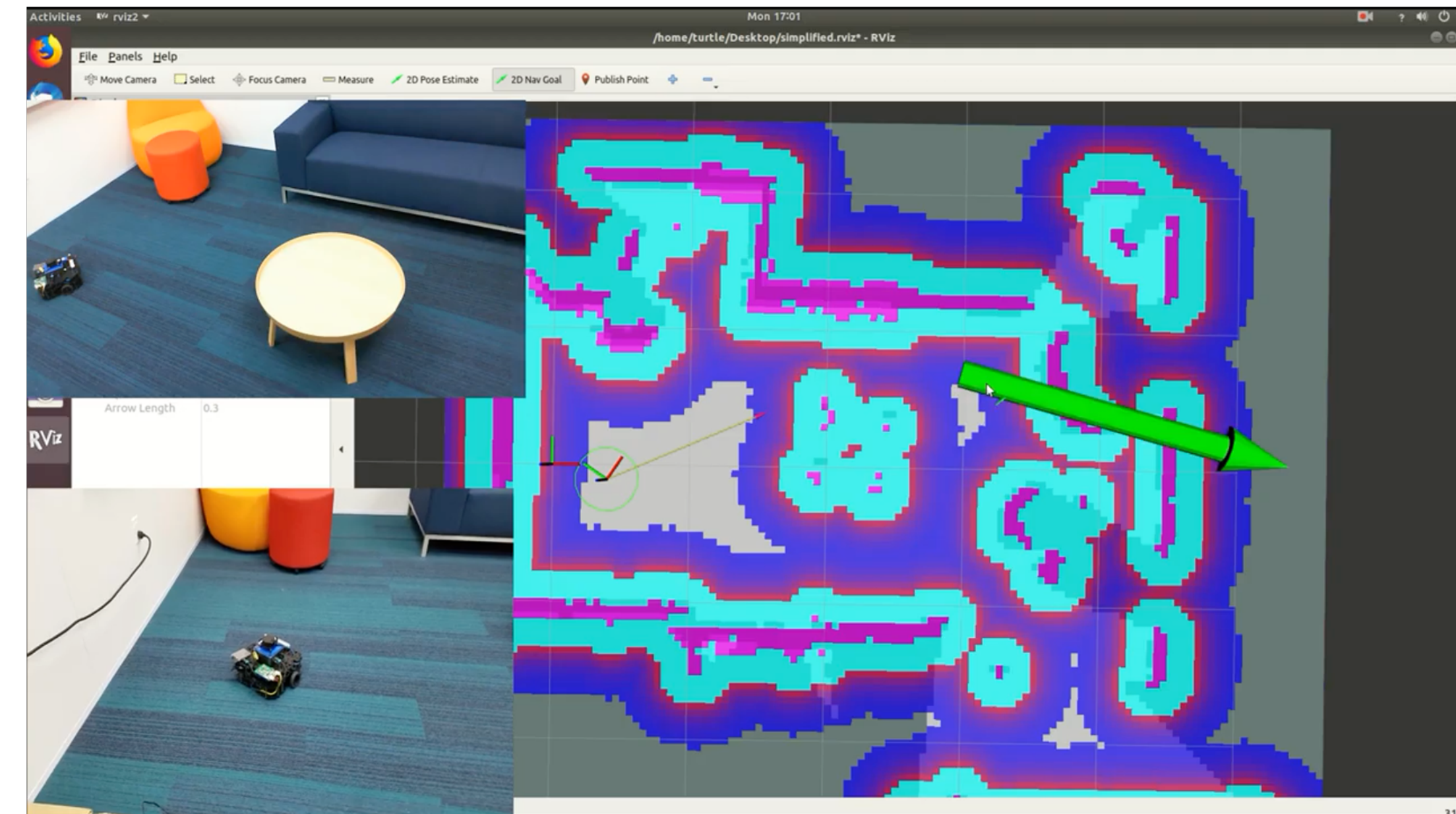
# 1.2 Robot Navigation

Accurately modelled robot navigates using **Lidar** in known, precisely mapped, static environment

- Robot navigates on the map

## Issues

- Known robot: impractical
- **Lidar**: range and resolution limitations, little semantic information, expensive



<https://navigation.ros.org>



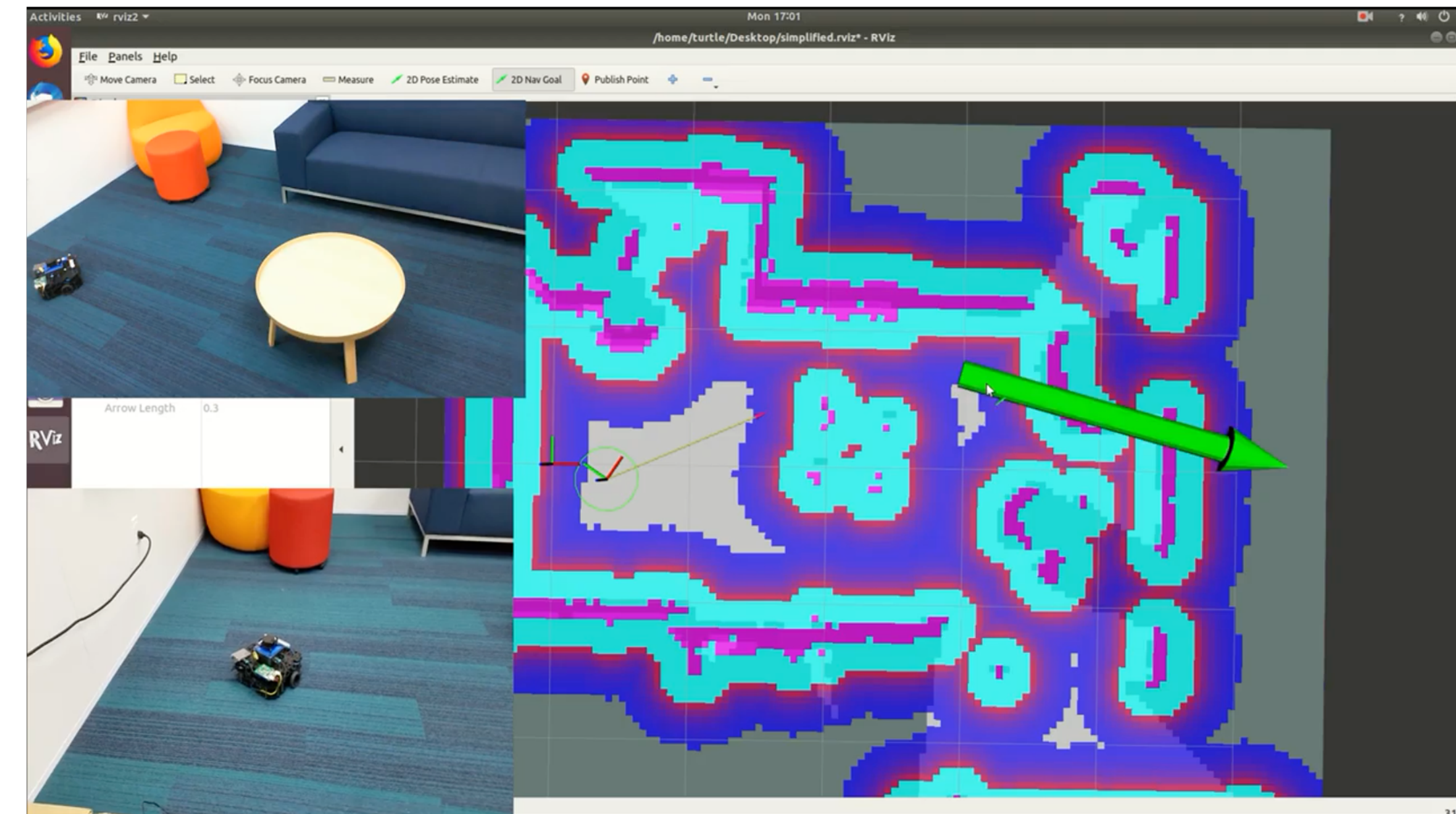
# 1.2 Robot Navigation

Accurately modelled robot navigates using Lidar in known, precisely mapped, static environment

- Robot navigates on the map

## Issues

- Known robot: impractical
- Lidar: range and resolution limitations, little semantic information, expensive
- Accurate metric map: expensive



<https://navigation.ros.org>

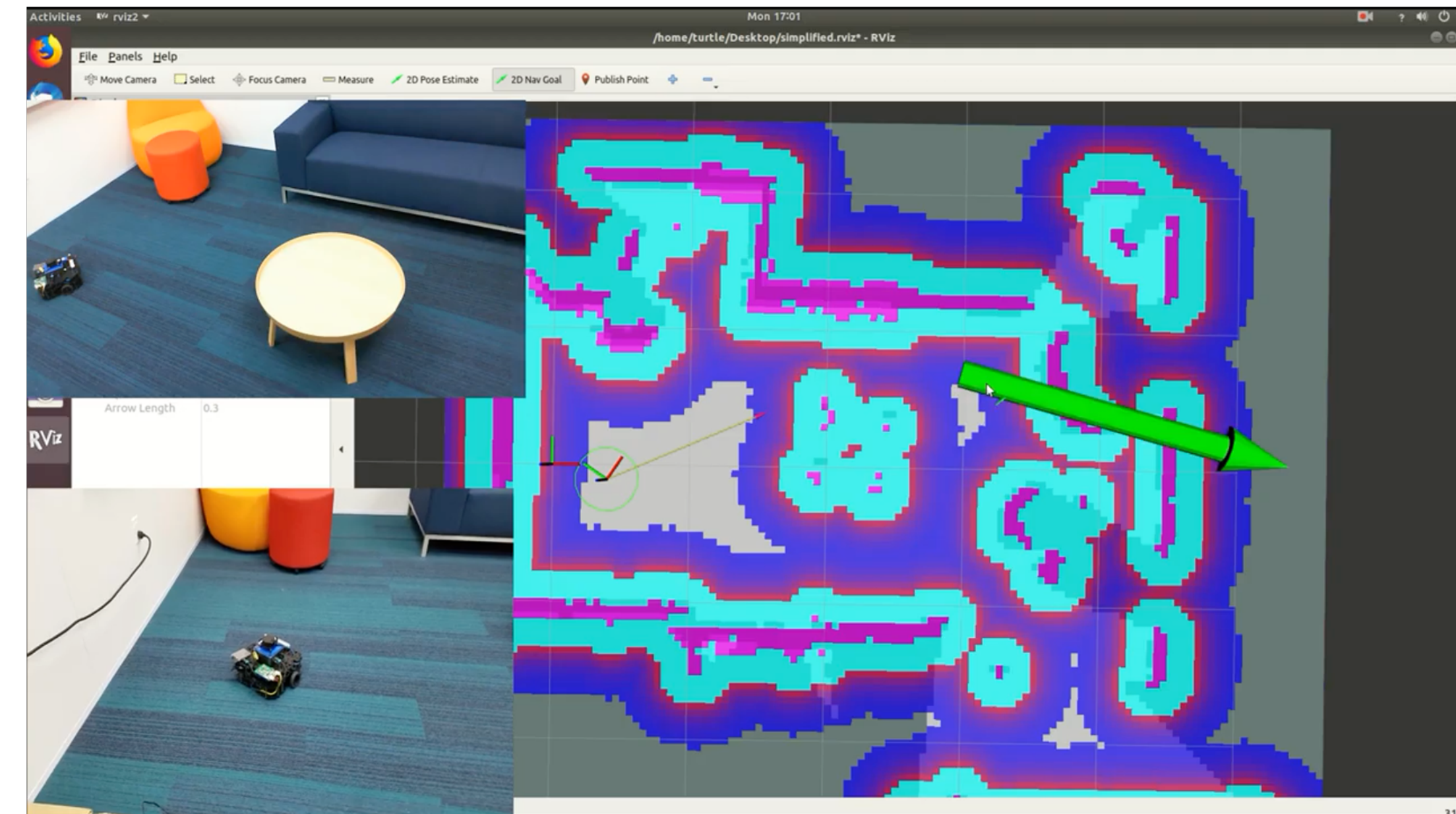
# 1.2 Robot Navigation

Accurately modelled robot navigates using Lidar in known, precisely mapped, static environment

- Robot navigates on the map

## Issues

- Known robot: impractical
- Lidar: range and resolution limitations, little semantic information, expensive
- Accurate metric map: expensive
- Static environment: unrealistic



<https://navigation.ros.org>



# 1.2 Robot Navigation

## Promising Directions

### Visual Localization

- Uses RGB camera, no Lidar needed
- Accurate metric map not always necessary





# 1.2 Robot Navigation

## Promising Directions

### Visual Localization

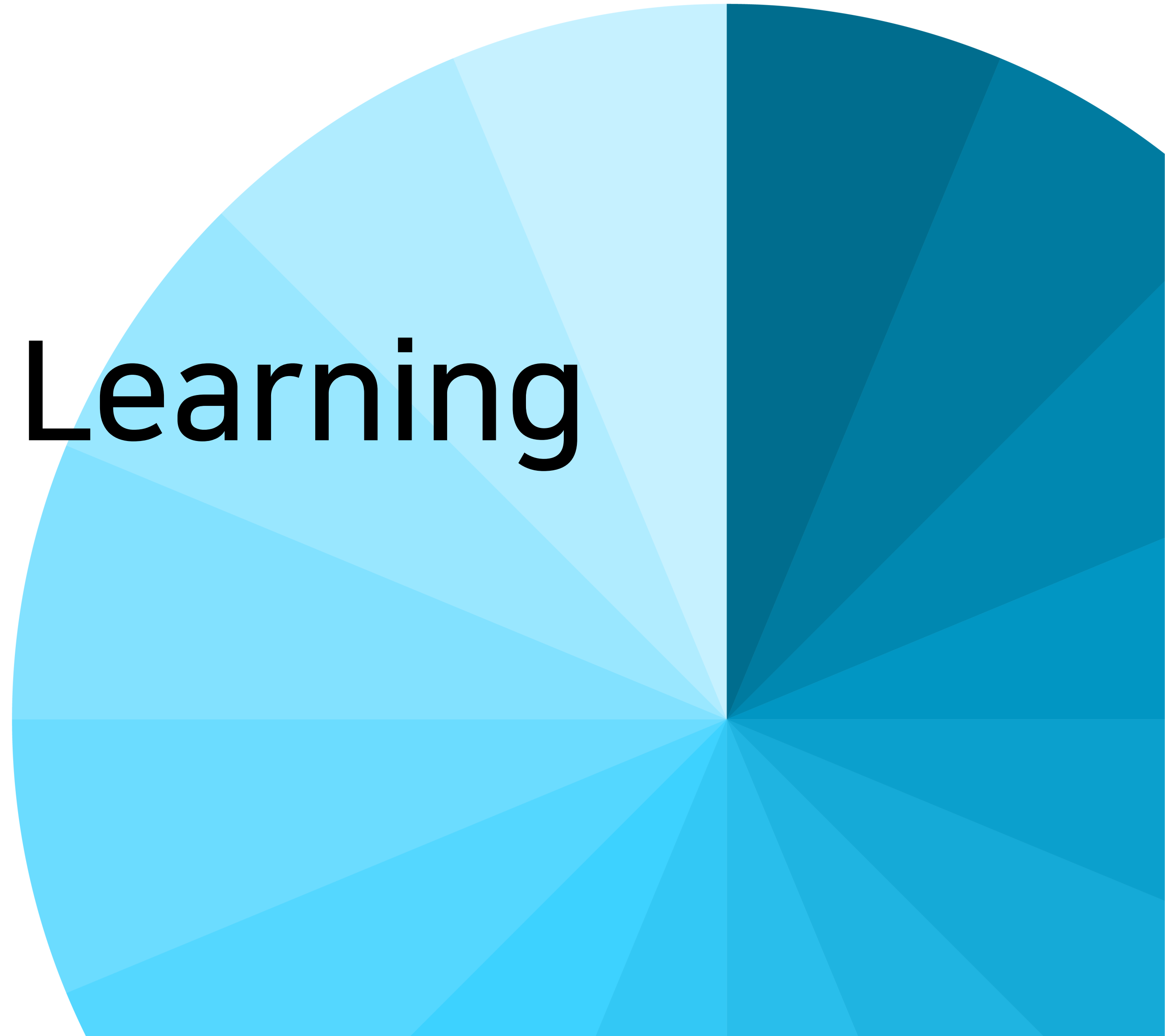
- Uses RGB camera, no Lidar needed
- Accurate metric map not always necessary

### Learn to Navigate

- Robust to observation and model errors: accurate robot, sensor, and environment models not necessary
- Learn to deal with dynamic environments



## 2. Robot Learning





## 2.1 Robot Learning Options

### Traditional control stack



- Lots of well-understood models, priors
- Fits well with Robotic Operating System architectures
- Interpretable
- Errors propagate across layers without other modules knowing
- Modules not optimized for the end task

## 2.1 Robot Learning Options

### “End-to-end” Learning



Deep Learning Model



- Recent exciting progress in Deep Learning
- Robust to noise and partial observations
- Optimized for the end task
- Hard to interpret
- Hard to include prior knowledge

# 2.1 Robot Learning Options

## Warning: Robot Learning is hard

- Trying all sort of things might be dangerous  
It breaks robots and labs
- Training real robots is really slow  
Limited speed, hw issues
- Humans often needed to set labels/rewards  
Hard, time consuming



<https://clips.twitch.tv/FunAmazingWrenFrankerZ>



# 2.1 Robot Learning Options

## Imitation Learning

- Supervised learning
- Preferred option for robot learning

## Advantages

- Well-known deep learning paradigms and tools
- Safe and sample-efficient

## Issues

- (Very) hard to provide examples or labels
- What about new situations?
- Cool if robots could learn by themselves



<https://developer.nvidia.com/blog/deep-learning-self-driving-cars>



<https://roboclette.ch>

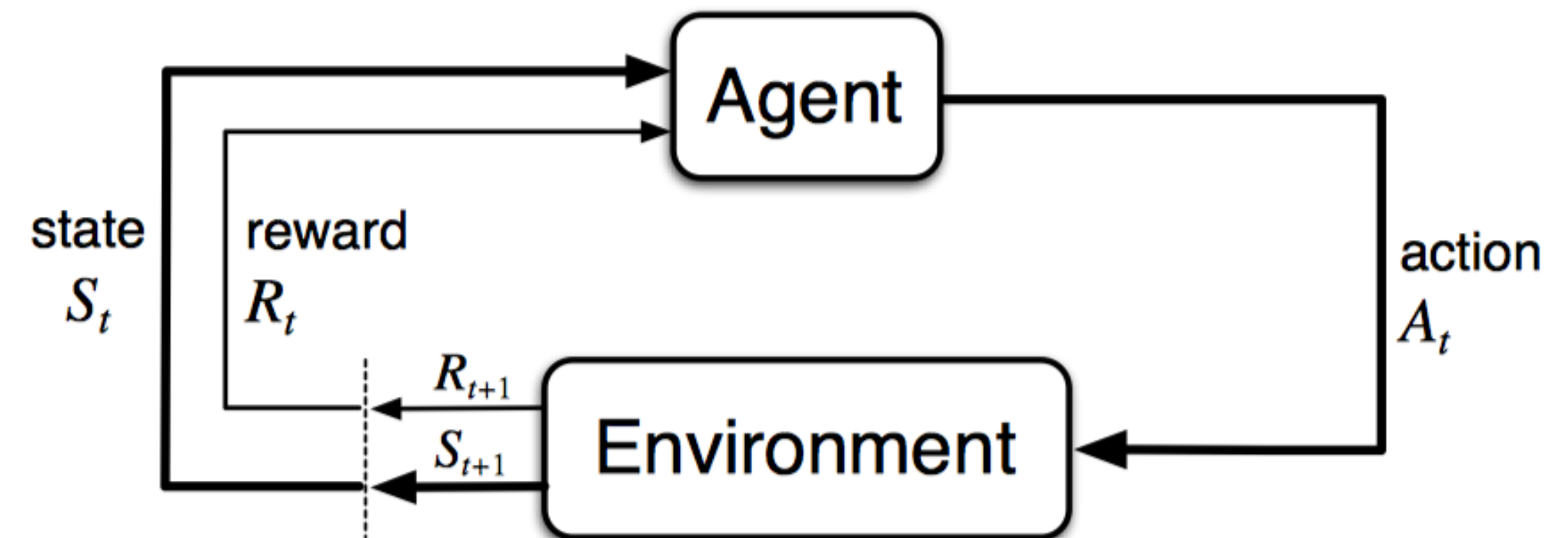
## 2.2 Reinforcement Learning

### Formalism to learn decision-making from experience

- Agent acts ( $A$ ) on the environment, and doing so it modifies environment's state  $S$  and gets a reward  $R$

### Goal

- Learn a policy  $\pi(A|S)$  to maximize accumulated rewards



## 2.2 Reinforcement Learning

### Formalism to learn decision-making from experience

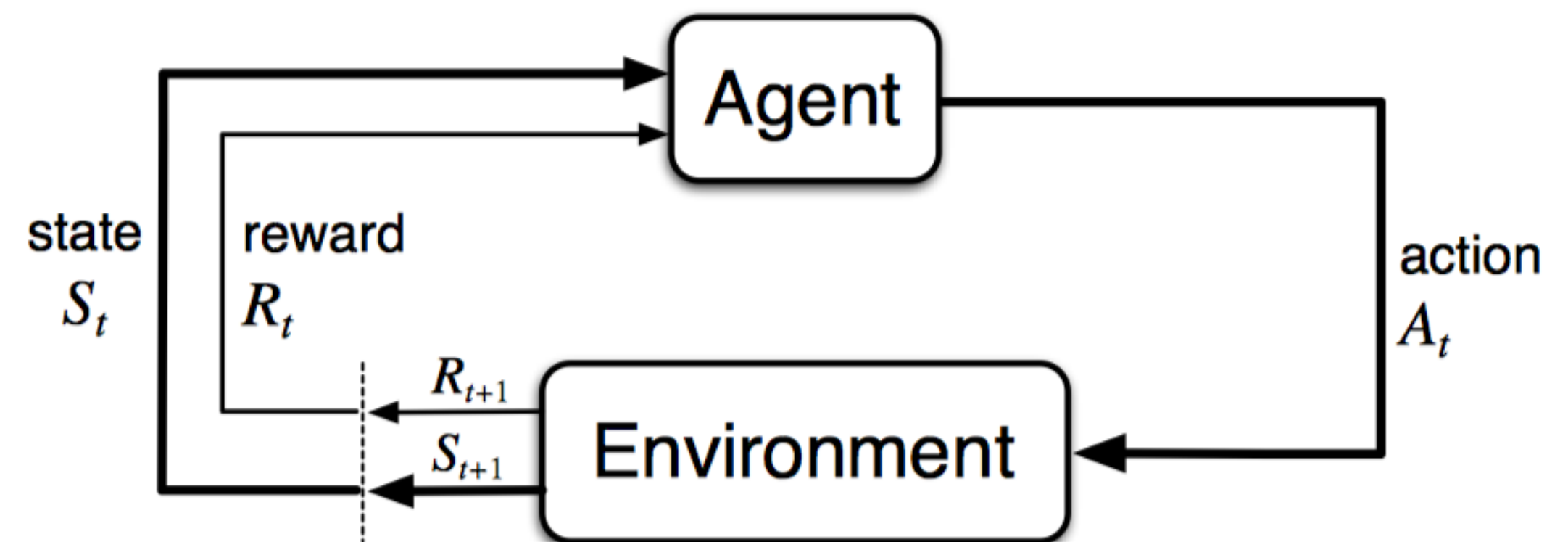
- Agent acts ( $A$ ) on the environment, and doing so it modifies environment's state  $S$  and gets a reward  $R$

### Goal

- Learn a policy  $\pi(A|S)$  to maximize accumulated rewards

### Deep RL

- Model (and learn) policy with Deep Neural Network





# 2.2 Reinforcement Learning

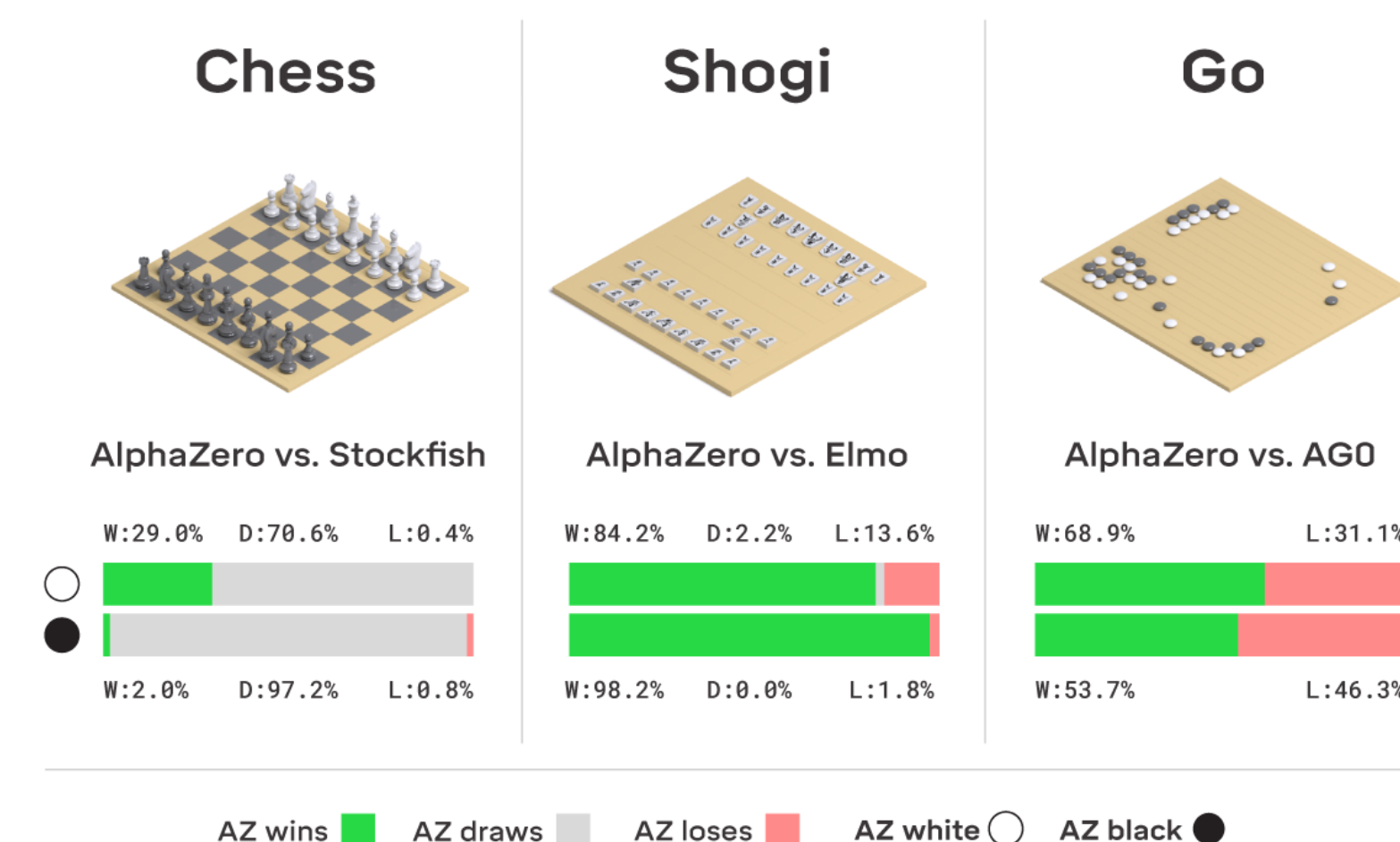
Deep Reinforcement Learning (RL)  
has received lots of interest lately



Playing Atari with Deep Reinforcement Learning, Mnih et al., 2015



Emergence of Locomotion Behaviours in Rich Environments, Heess et al., 2017



A general reinforcement learning algorithm that masters chess, shogi and Go through self-play, Silver et al., 2017

## 2.2 Reinforcement Learning

### Success in game-like settings

- Cannot break robots / labs
- Fast gaming environments
- Easy to set rewards

## 2.2 Reinforcement Learning

### Success in game-like settings

- Cannot break robots / labs
- Fast gaming environments
- Easy to set rewards



<https://sites.google.com/view/simopt>

### Idea

- Use simulation environments for robot learning



<http://gibsonenv.stanford.edu>



## 2.2 Reinforcement Learning

### Simulation environments

- Safe
- Fast and parallelizable
- Procedural generation of experiments
- Quantitative results
- Simulation-to-real gap
- Quickly becomes complex for complex robots, environments, use-cases

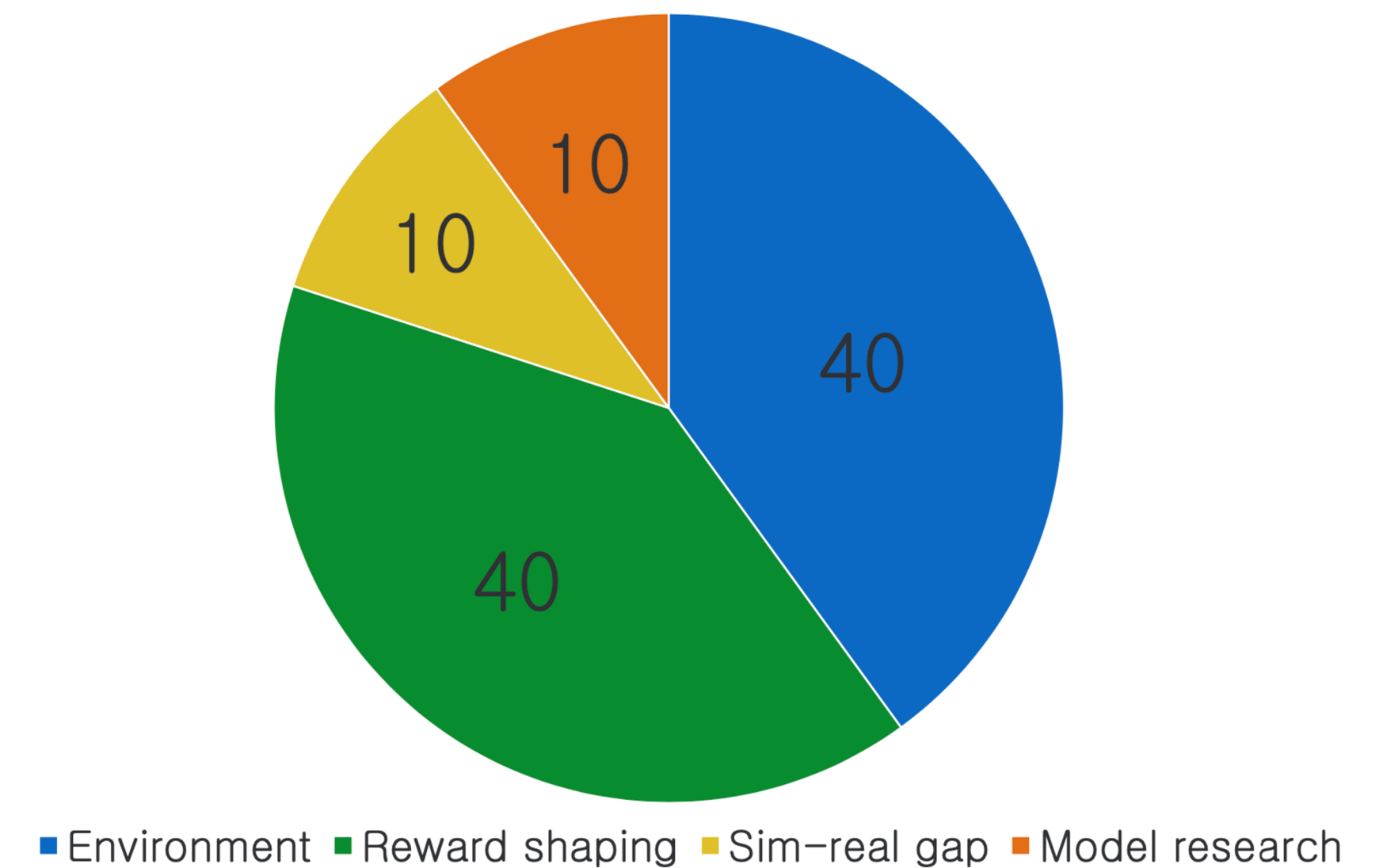
## 2.2 Reinforcement Learning

### Simulation environments

- Safe
- Fast and parallelizable
- Procedural generation of experiments
- Quantitative results
- Simulation-to-real gap
- Quickly becomes complex for complex robots, environments, use-cases

### 3.1 Deep RL in practice

40/40/10/10 rule



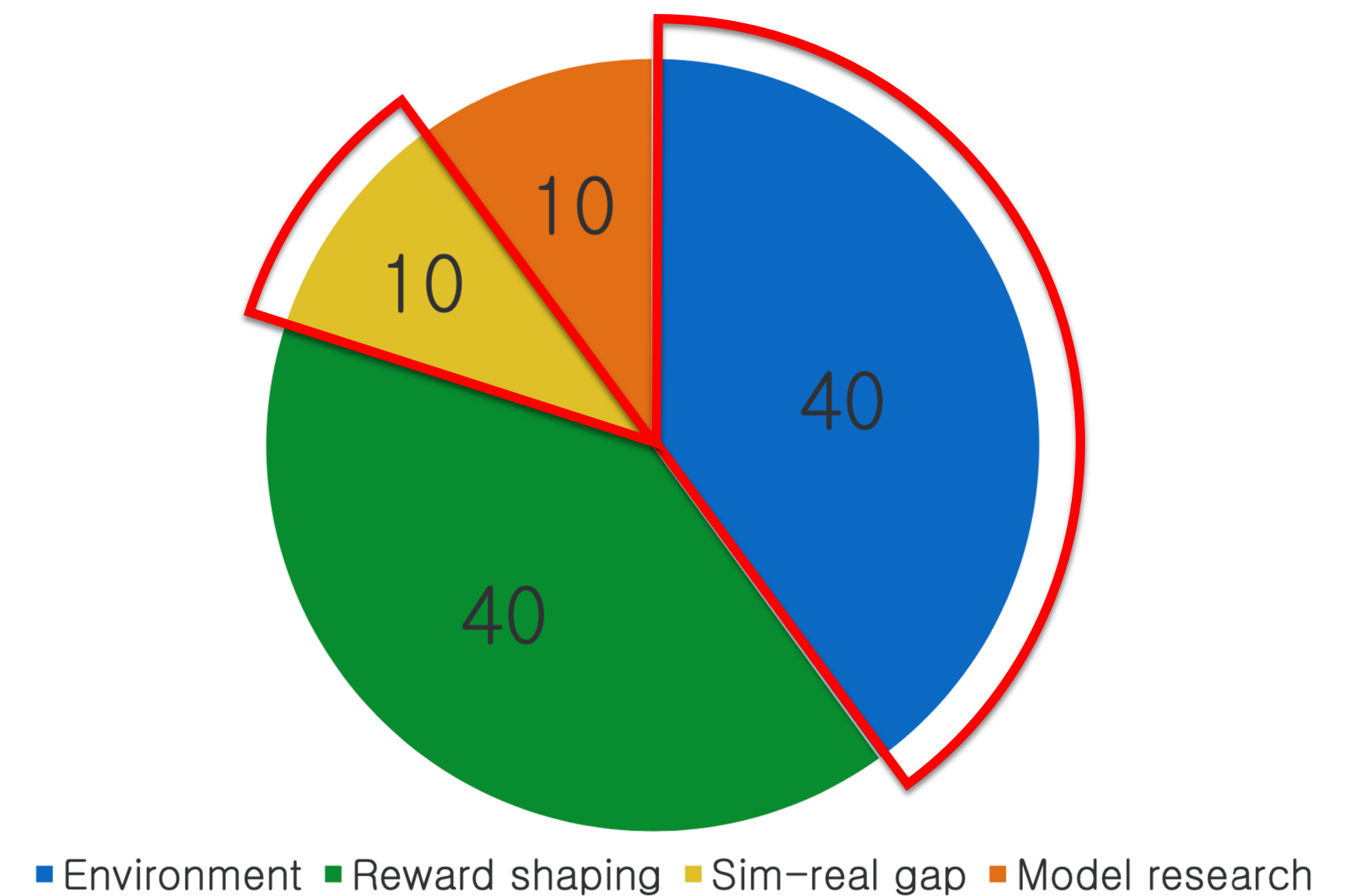
## 2.2 Reinforcement Learning

### Simulation environments

- Safe
- Fast and parallelizable
- Procedural generation of experiments
- Quantitative results
- Simulation-to-real gap
- Quickly becomes complex for complex robots, environments, use-cases

### 3.1 Deep RL in practice

40/40/10/10 rule





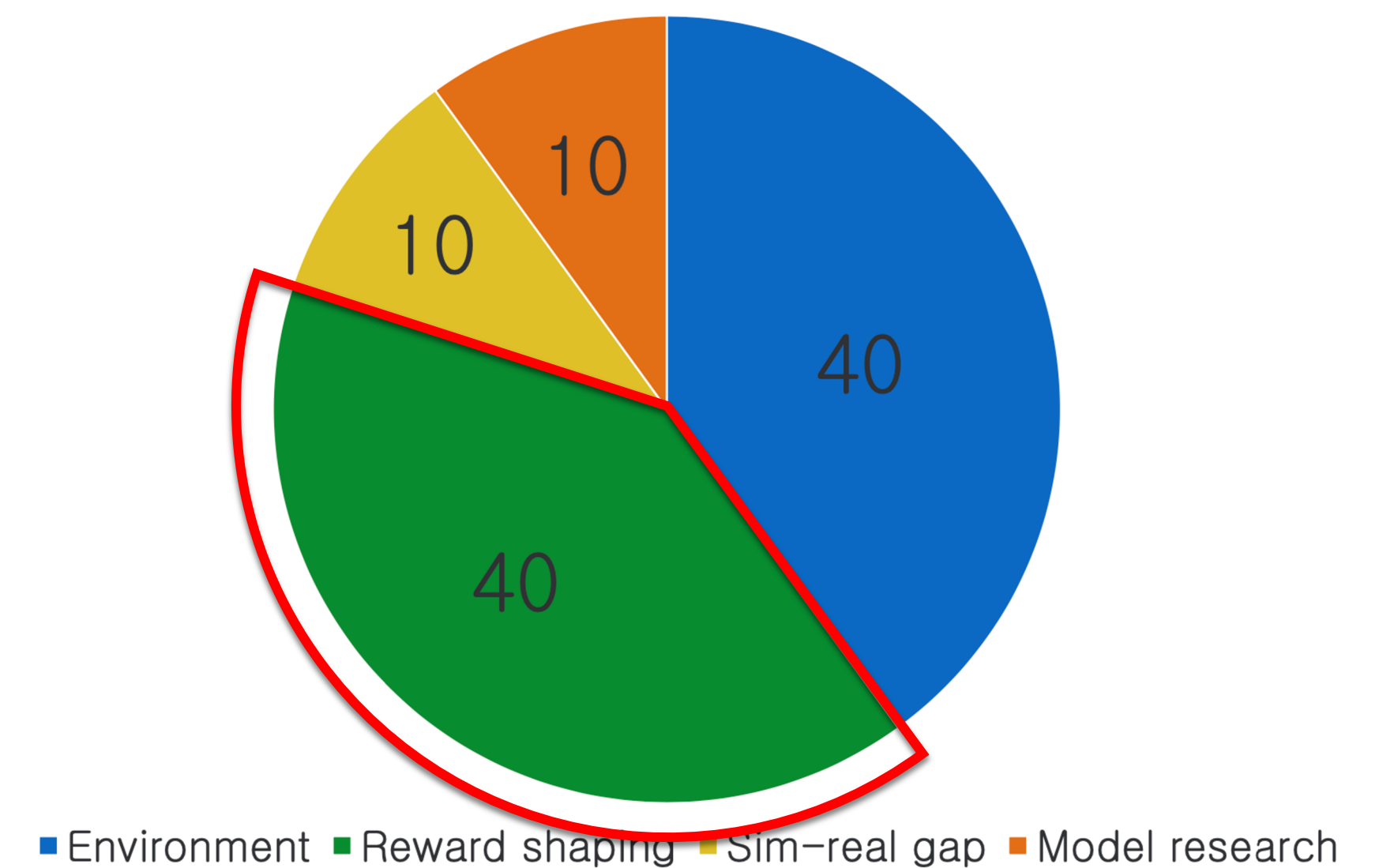
## 2.2 Reinforcement Learning

### Reward shaping

- The way the reward is formulated
- Determines agent behaviour

### 3.1 Deep RL in practice

40/40/10/10 rule



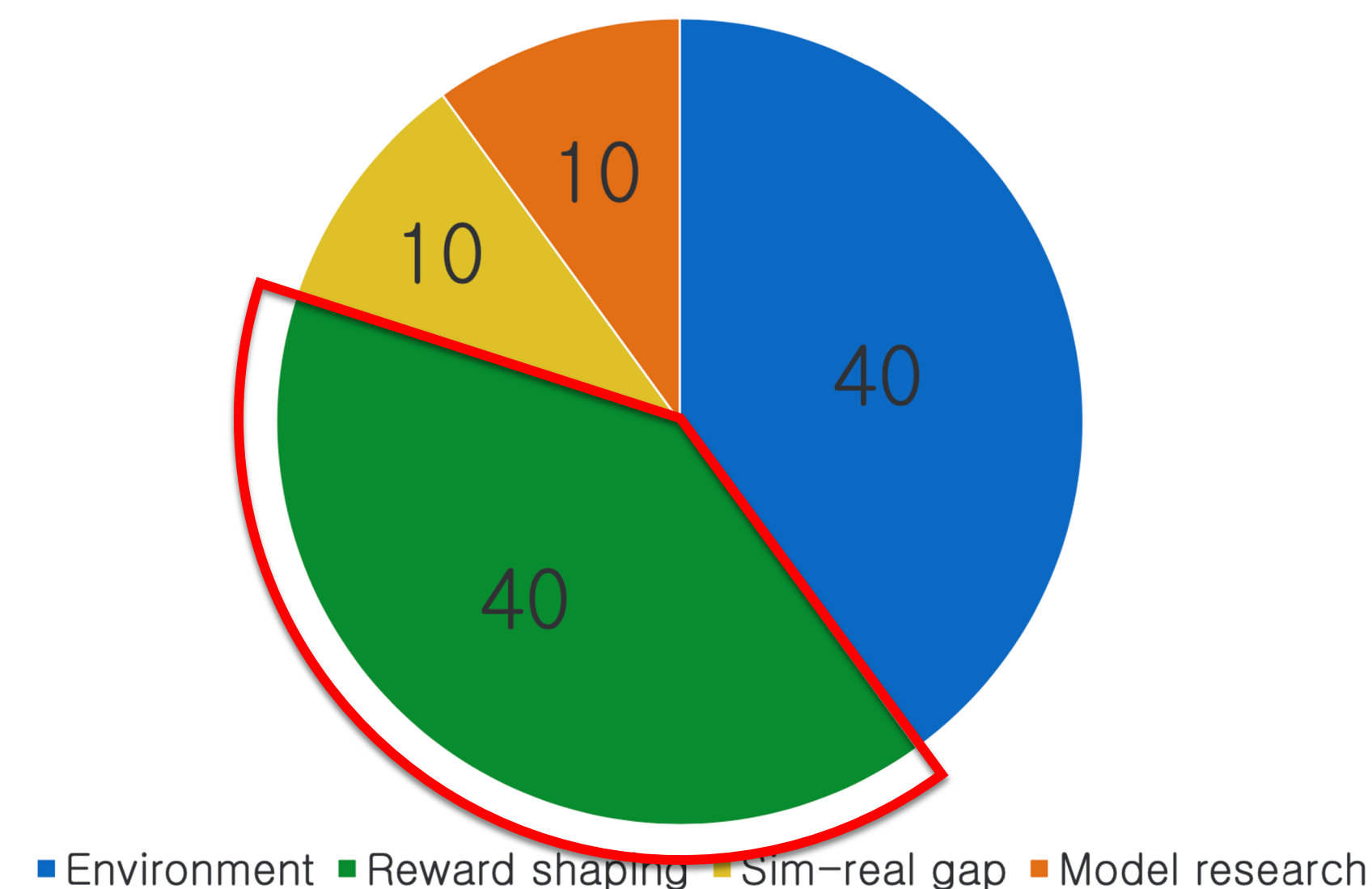
## 2.2 Reinforcement Learning

### Reward shaping

- The way the reward is formulated
- Determines agent behaviour
- Hard to predict result for complex agents and behaviours
- Takes a lot of time and experience

### 3.1 Deep RL in practice

40/40/10/10 rule





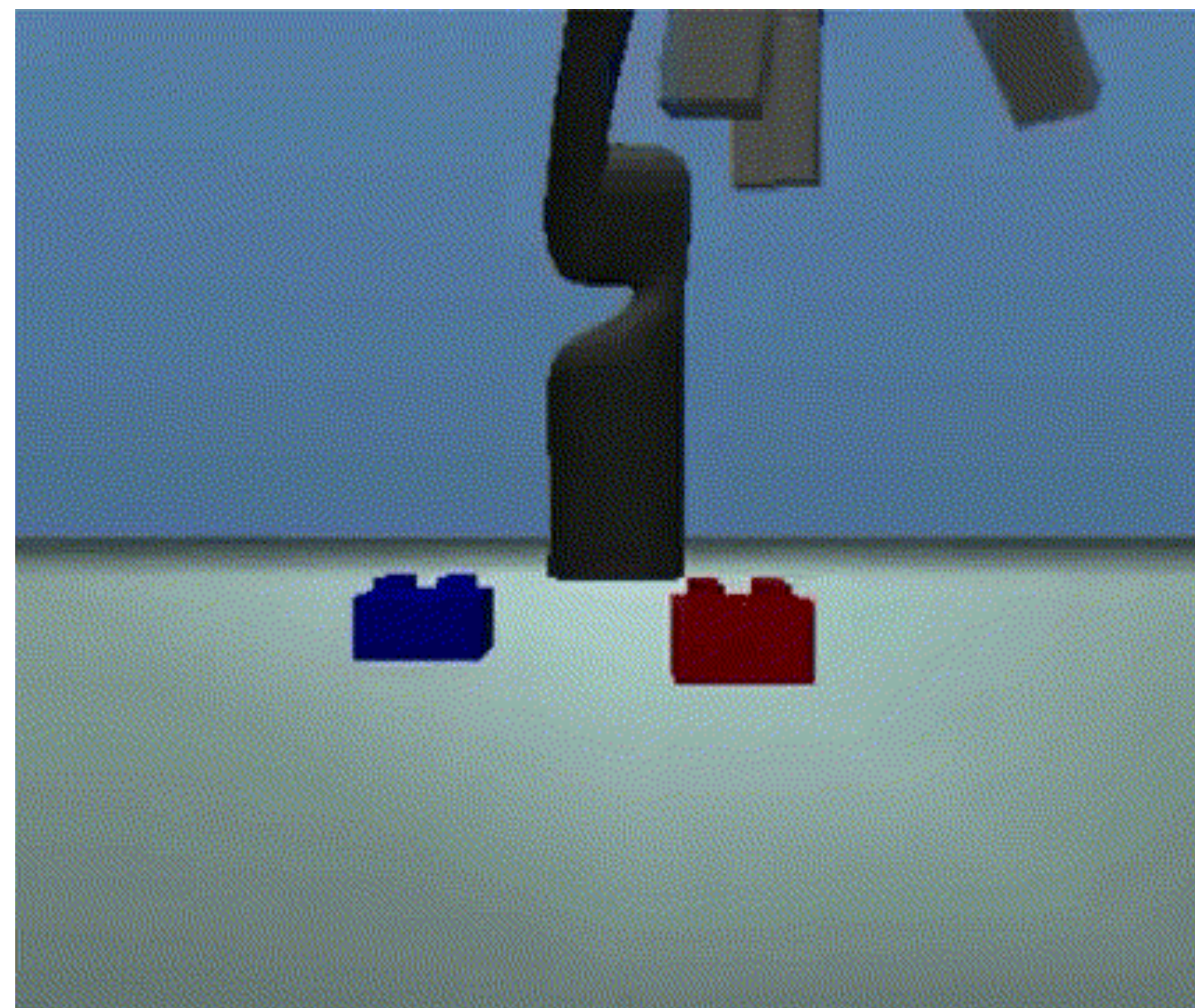
## 2.2 Reinforcement Learning

### Task

- Stack the red block on top of the blue one

### Reward

- Height of the bottom face of the red block when the arm is not touching the block



Data-efficient Deep Reinforcement Learning for Dexterous Manipulation, I. Popov et al., 2017



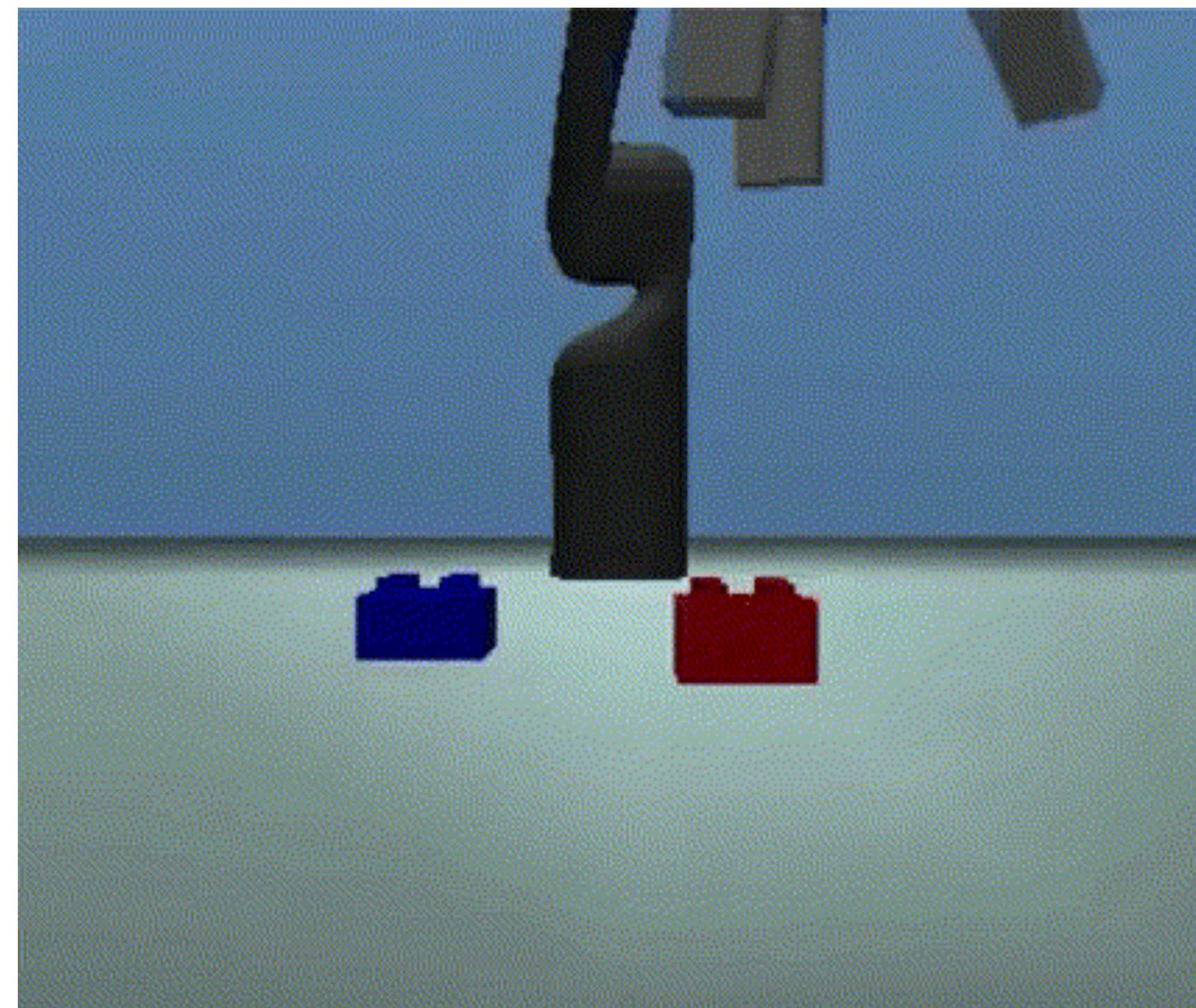
## 2.2 Reinforcement Learning

### Task

- Stack the red block on top of the blue one

### Reward

- Height of the bottom face of the red block when the arm is not touching the block



Data-efficient Deep Reinforcement Learning for Dexterous Manipulation, I. Popov et al., 2017



## 2.3 Best of Both Worlds?

### (Deep) RL

- Powerful modern tools
- Robust to noise
- Optimized for the end task
  
- Heavy data requirements
- Complex reward shaping
- Hard to interpret
- Hard to include prior knowledge

## 2.3 Best of Both Worlds?

### (Deep) RL

- Powerful modern tools
- Robust to noise
- Optimized for the end task
- Heavy data requirements
- Complex reward shaping
- Hard to interpret
- Hard to include prior knowledge

### Classic modular architecture

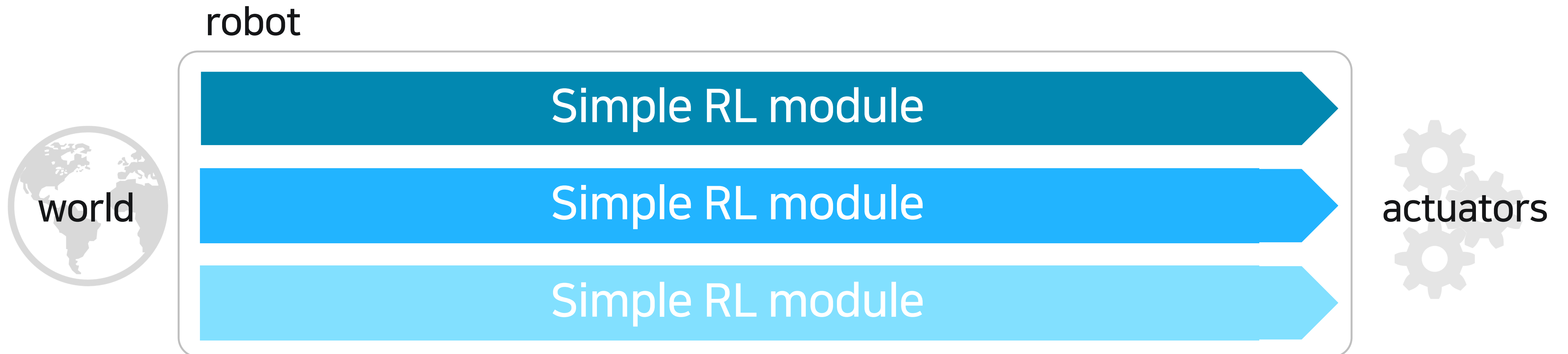
- Lots of well-understood models
- Useful priors
- Fits with Robotic Operating System
- Interpretable
- Errors propagate across layers
- Modules not optimized for end task



## 2.3 Best of Both Worlds?

### Idea

- Combine advantages of RL and model-based modular architectures
- Robot decomposed into simple learnable components that are jointly trained to realize high-level coordinated actions

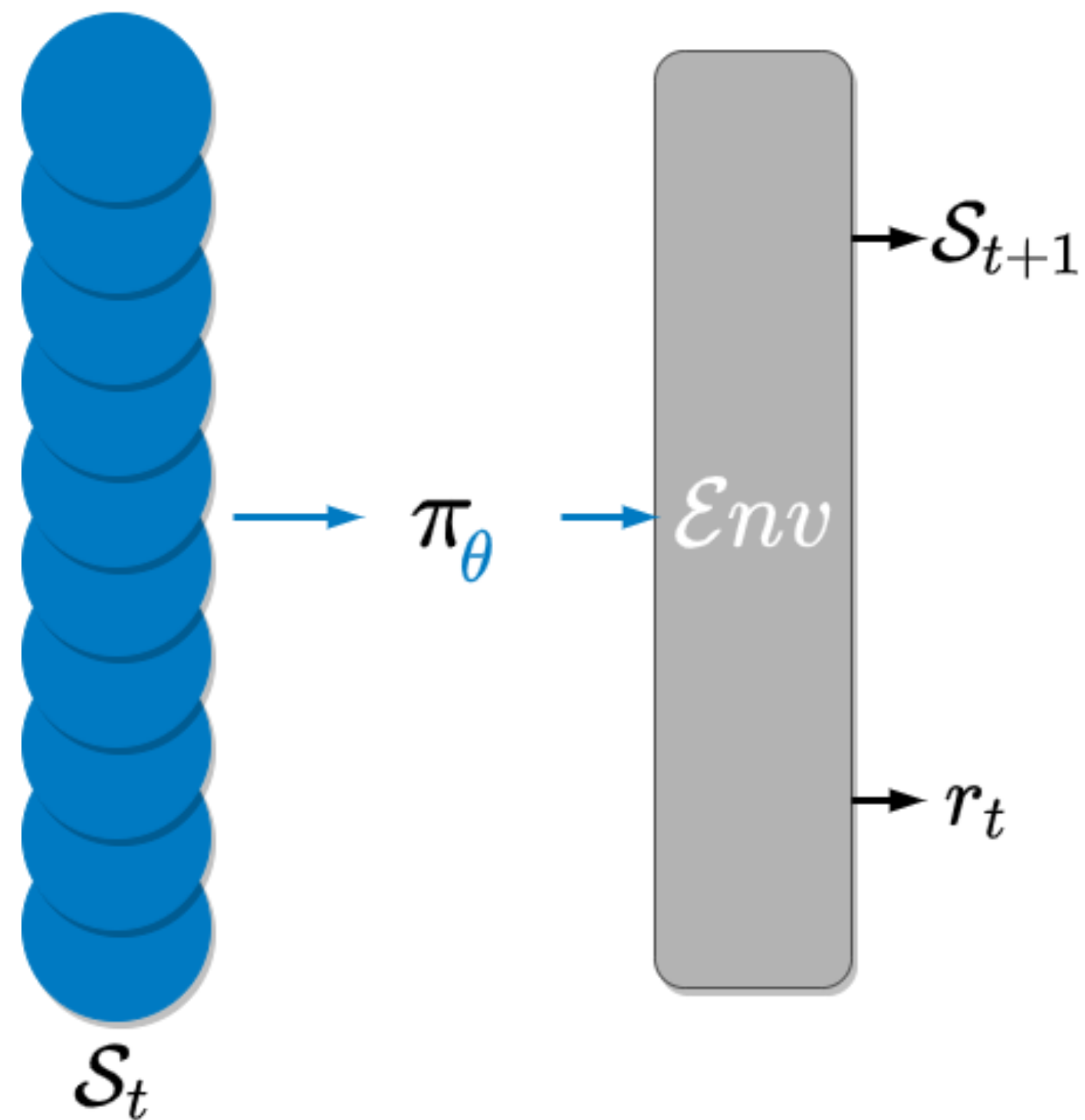


# 3. Modular Reinforcement Learning

# 3.1 Motivation for Modular RL

## "Classic" RL

- Learn  $\pi_\theta$

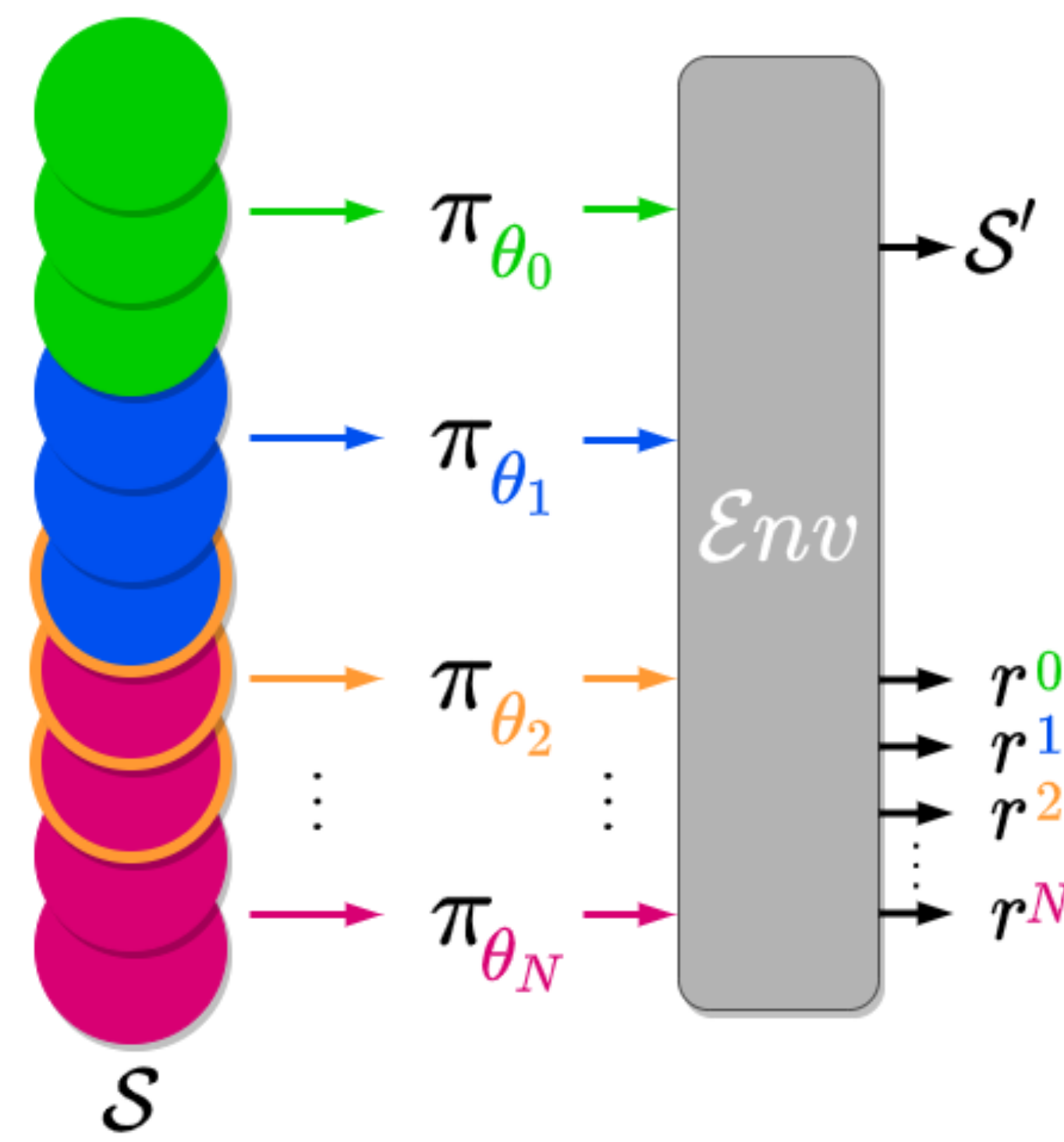
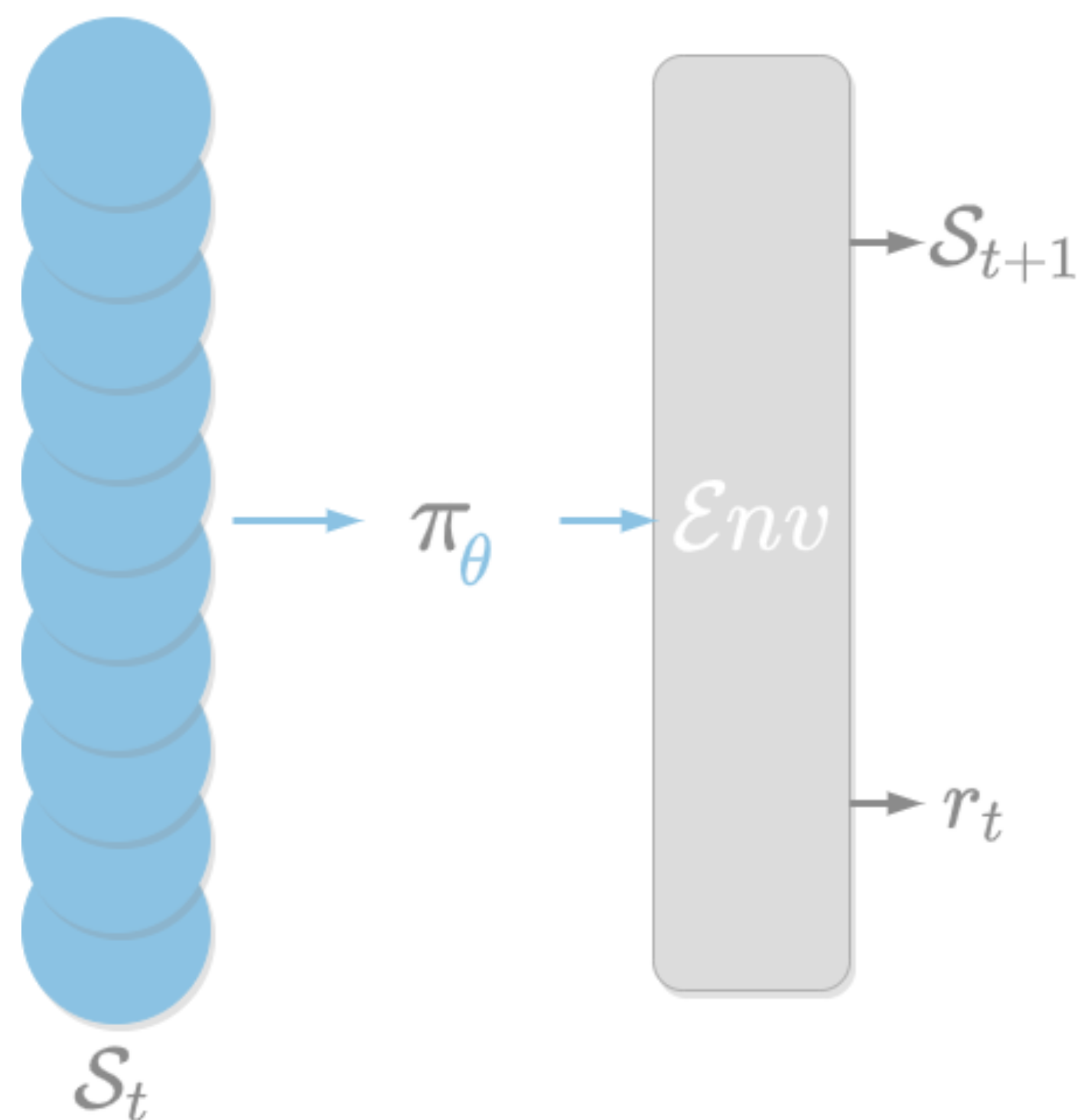




# 3.1 Motivation for Modular RL

## Modular RL

- Decompose  $\pi_\theta$  into  $\pi_{\theta_0}, \dots, \pi_{\theta_N}$
- Learn  $\pi_{\theta_0}, \dots, \pi_{\theta_N}$  simultaneously



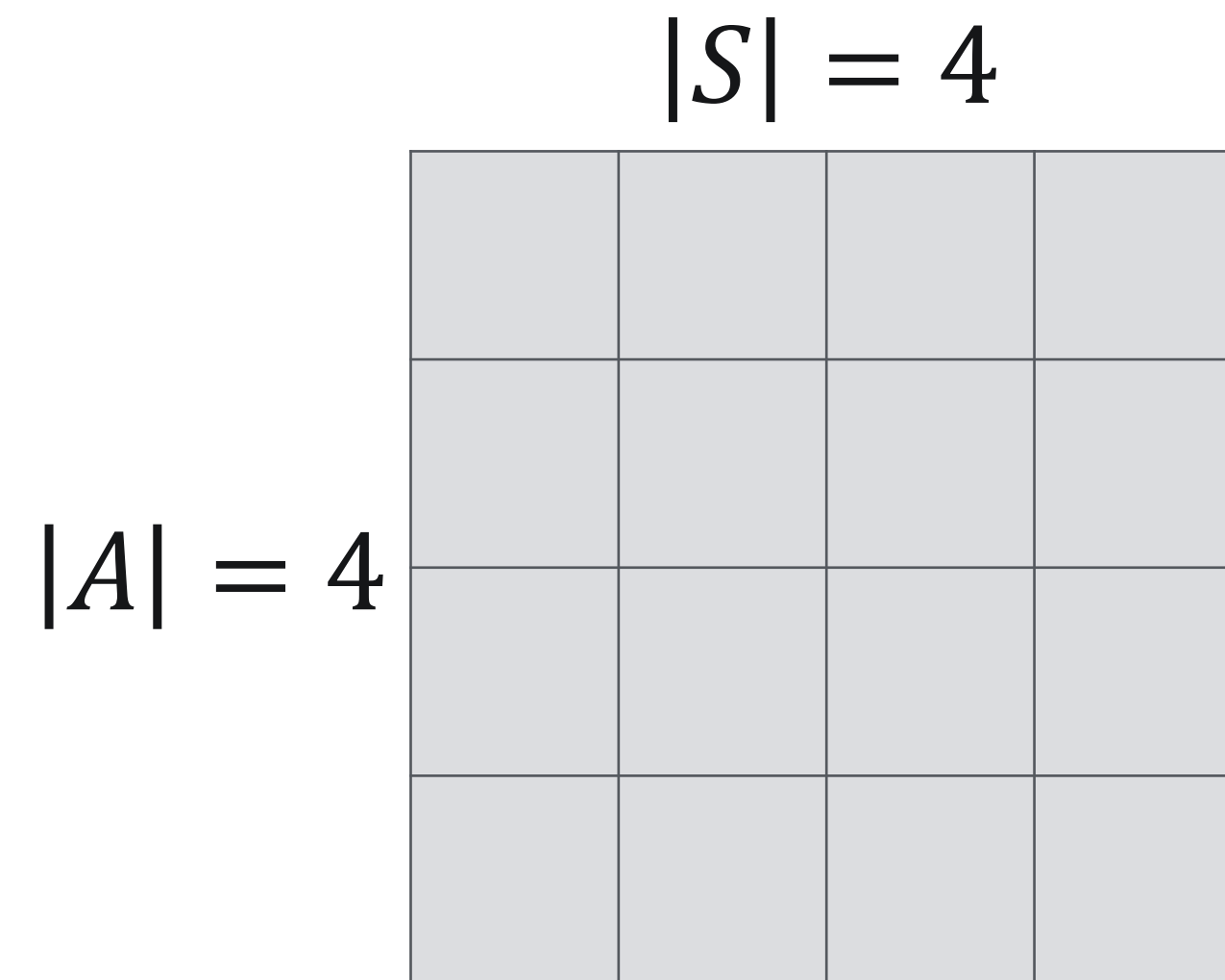
# 3.1 Motivation for Modular RL

## Why it should work

- Decomposition introduces useful **prior knowledge** and makes system more **interpretable**
- Learning introduces **robustness** and compensates for potential errors
- Decomposition in simple components **reduces sample complexity** and need for **reward shaping**
- **Fits with Robot Operating System** modular architecture
- Easy to include **model-based components**, or different learning paradigms
- Arguably **simpler to transfer to**, or **train on**, robot

## 3.1 Motivation for Modular RL

Number of samples needed by RL algorithms to learn a near-optimal policy is of the order  $|S||A| \log(|S||A|)$

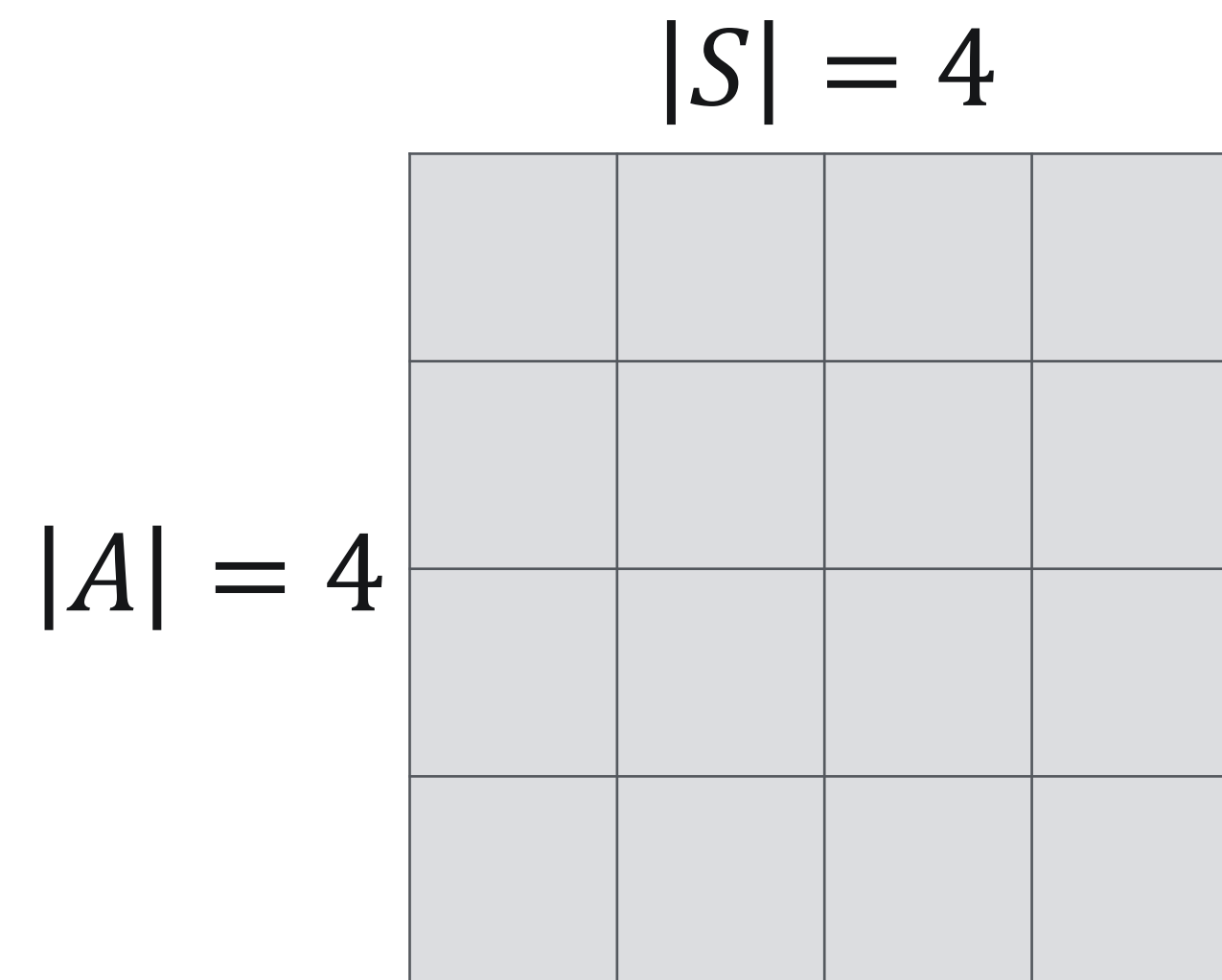


$$16 \log(16) = 64$$

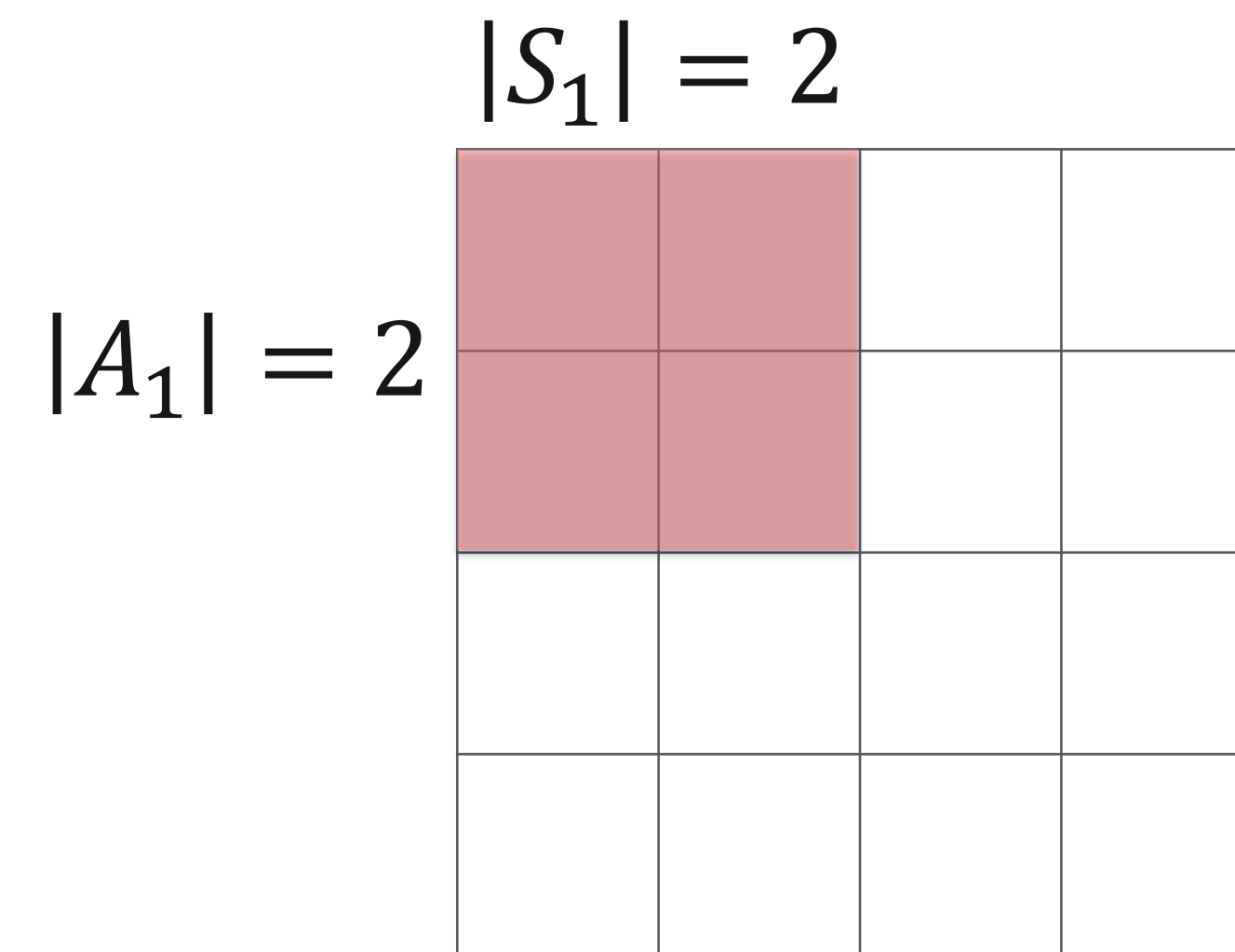


# 3.1 Motivation for Modular RL

Number of samples needed by RL algorithms to learn a near-optimal policy is of the order  $|S||A| \log(|S||A|)$



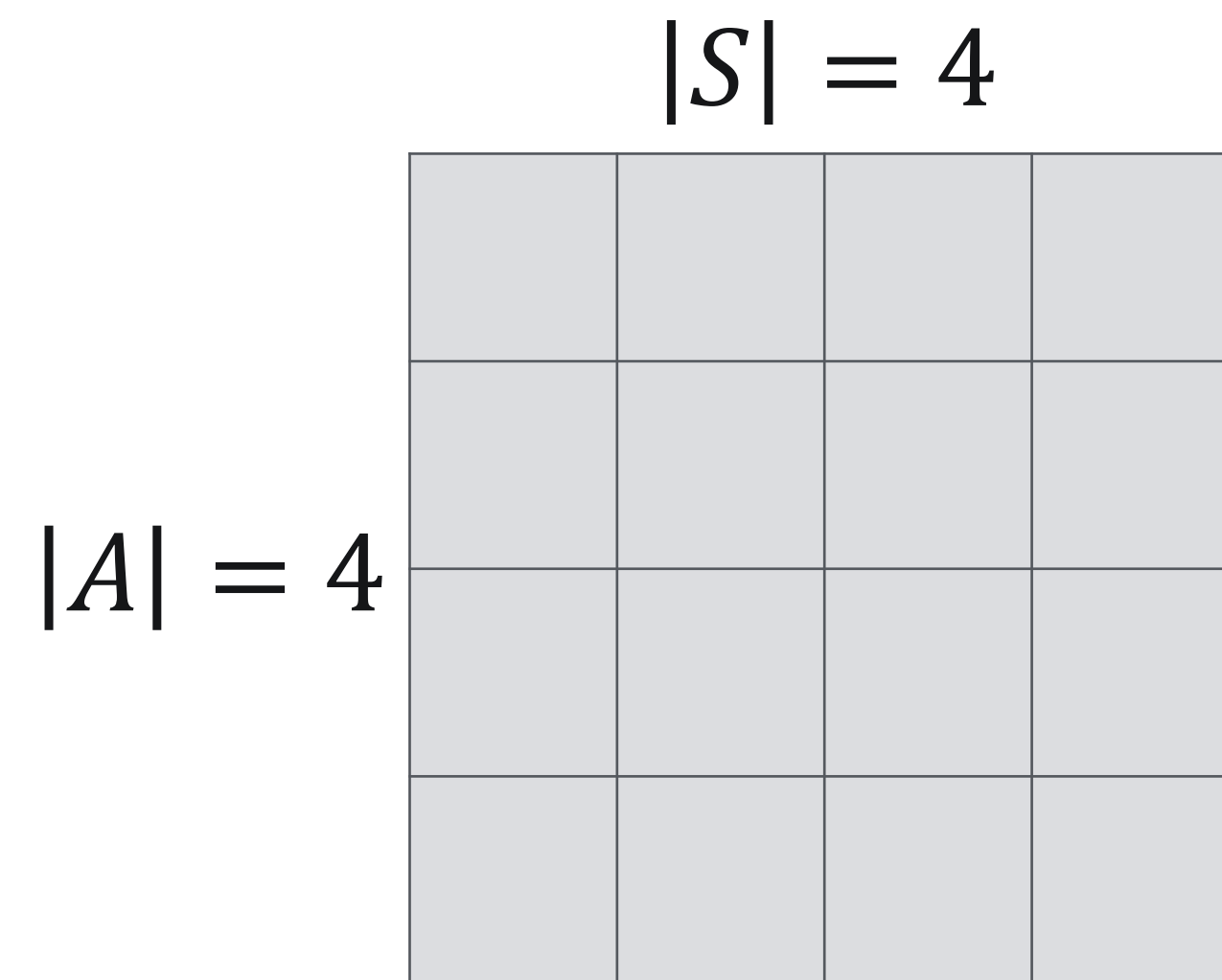
$$16 \log(16) = 64$$



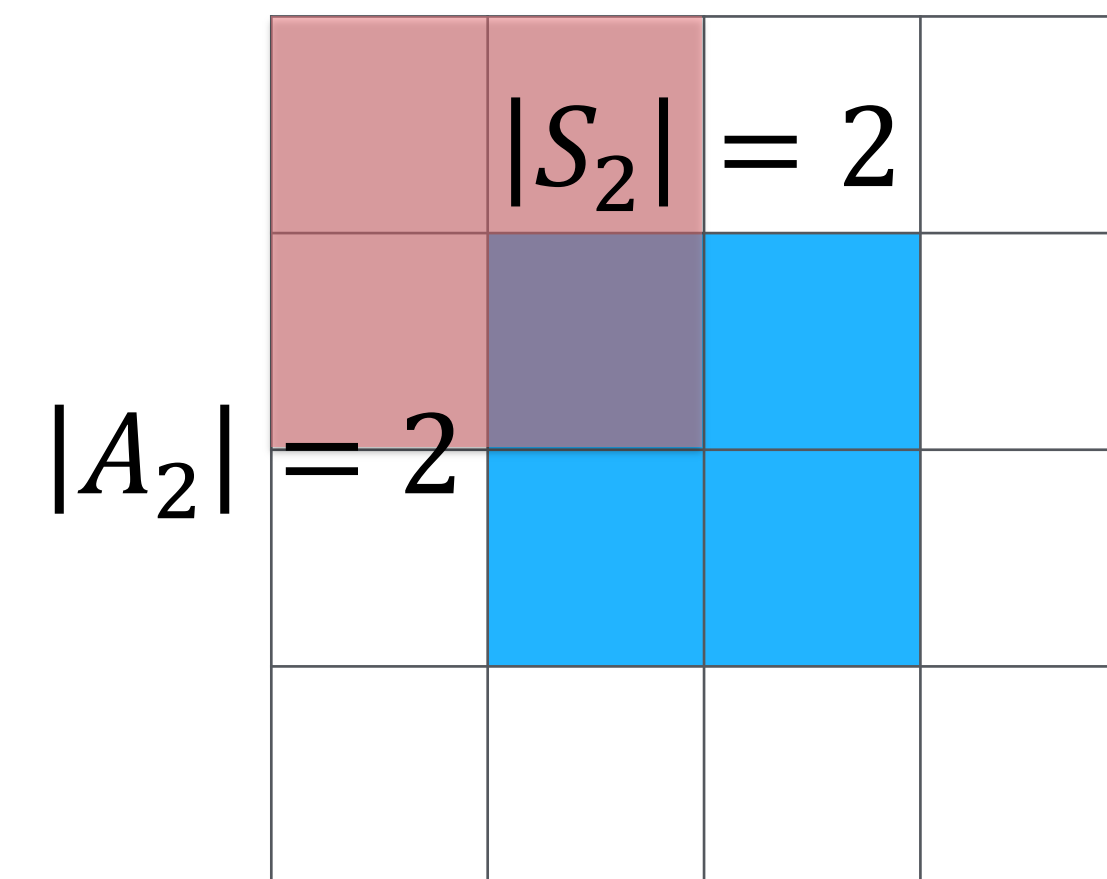
$$4 \log(4)$$

# 3.1 Motivation for Modular RL

Number of samples needed by RL algorithms to learn a near-optimal policy is of the order  $|S||A| \log(|S||A|)$



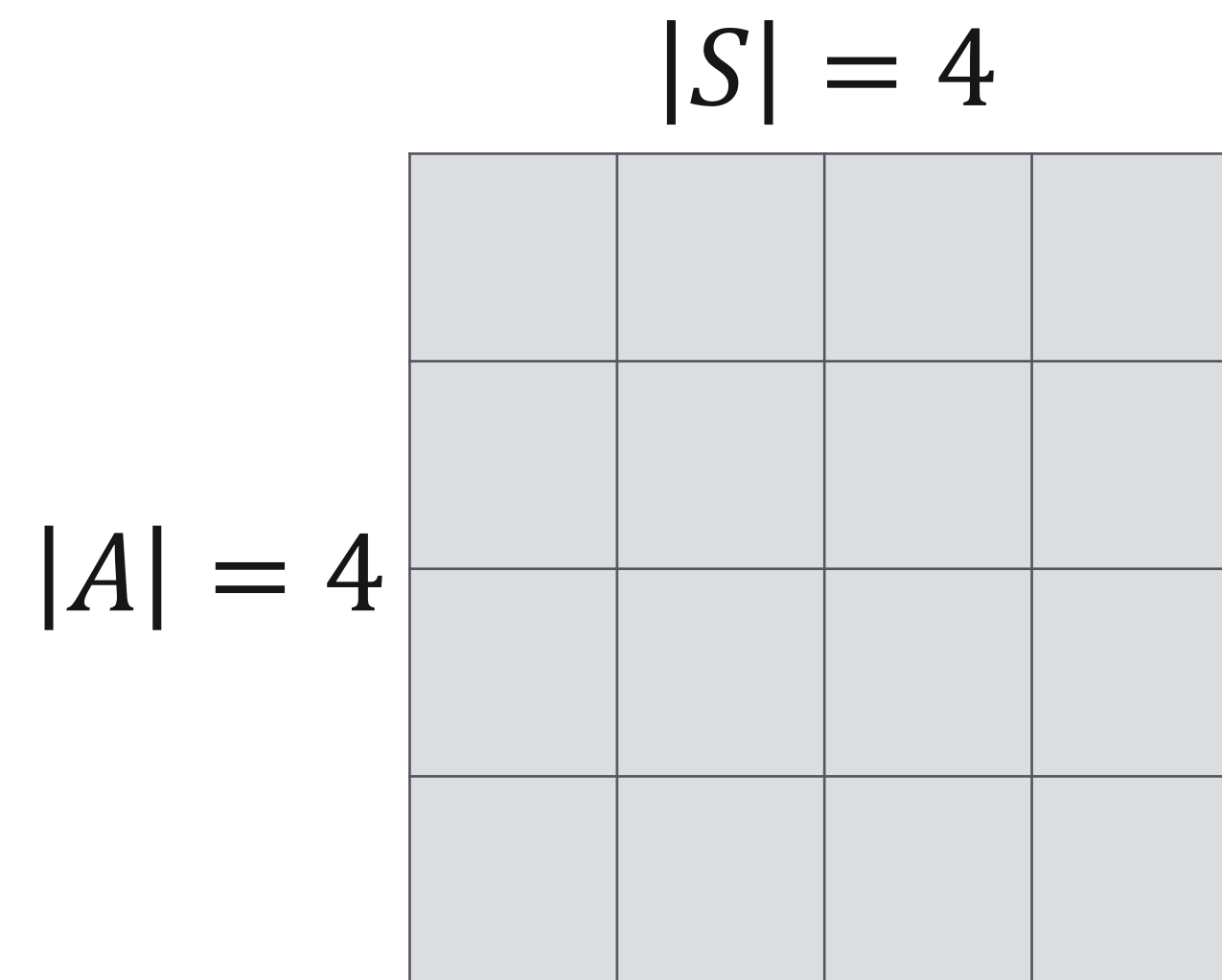
$$16 \log(16) = 64$$



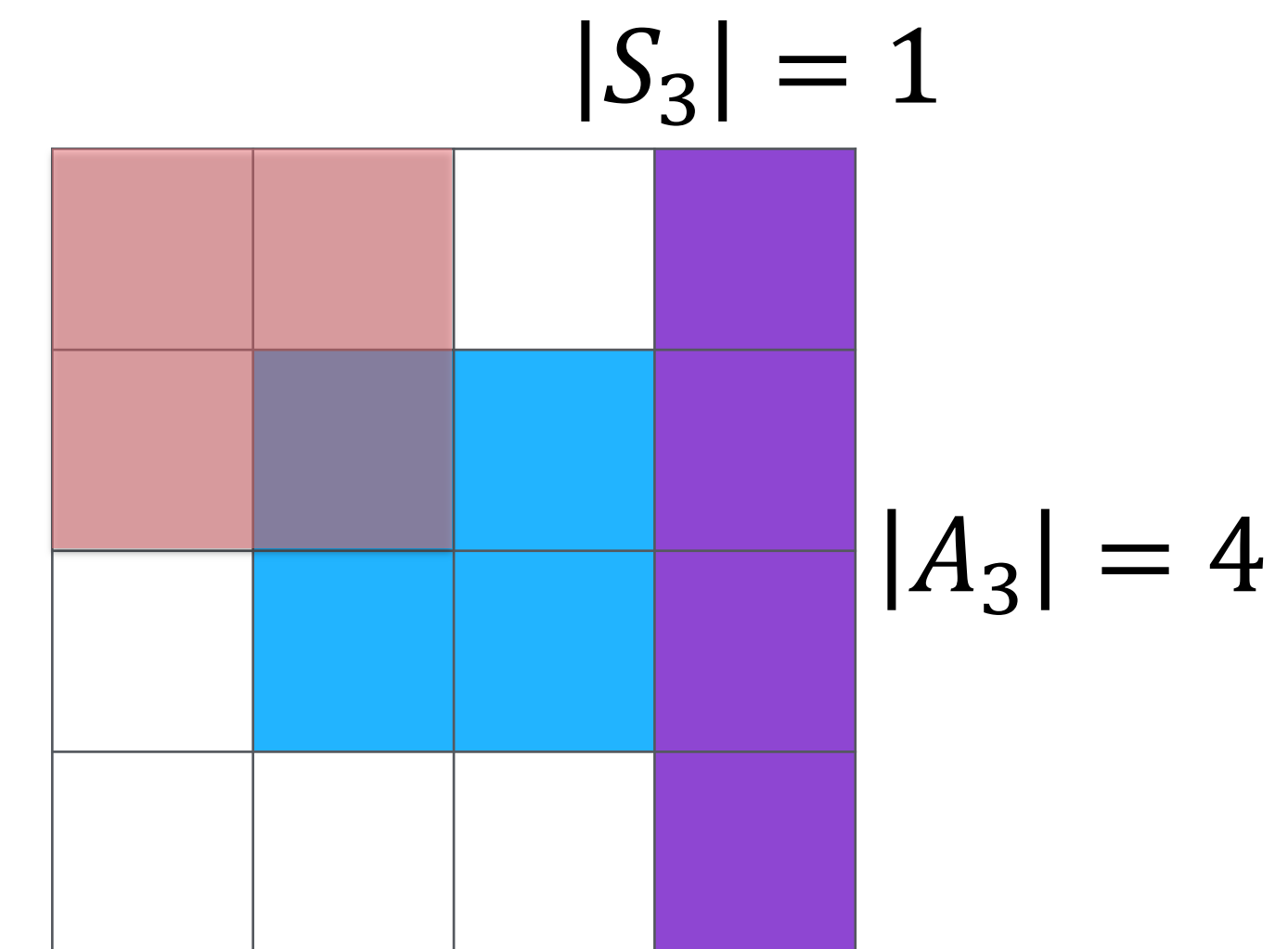
$$4 \log(4) + 4 \log(4)$$

# 3.1 Motivation for Modular RL

Number of samples needed by RL algorithms to learn a near-optimal policy is of the order  $|S||A| \log(|S||A|)$



$$16 \log(16) = 64$$

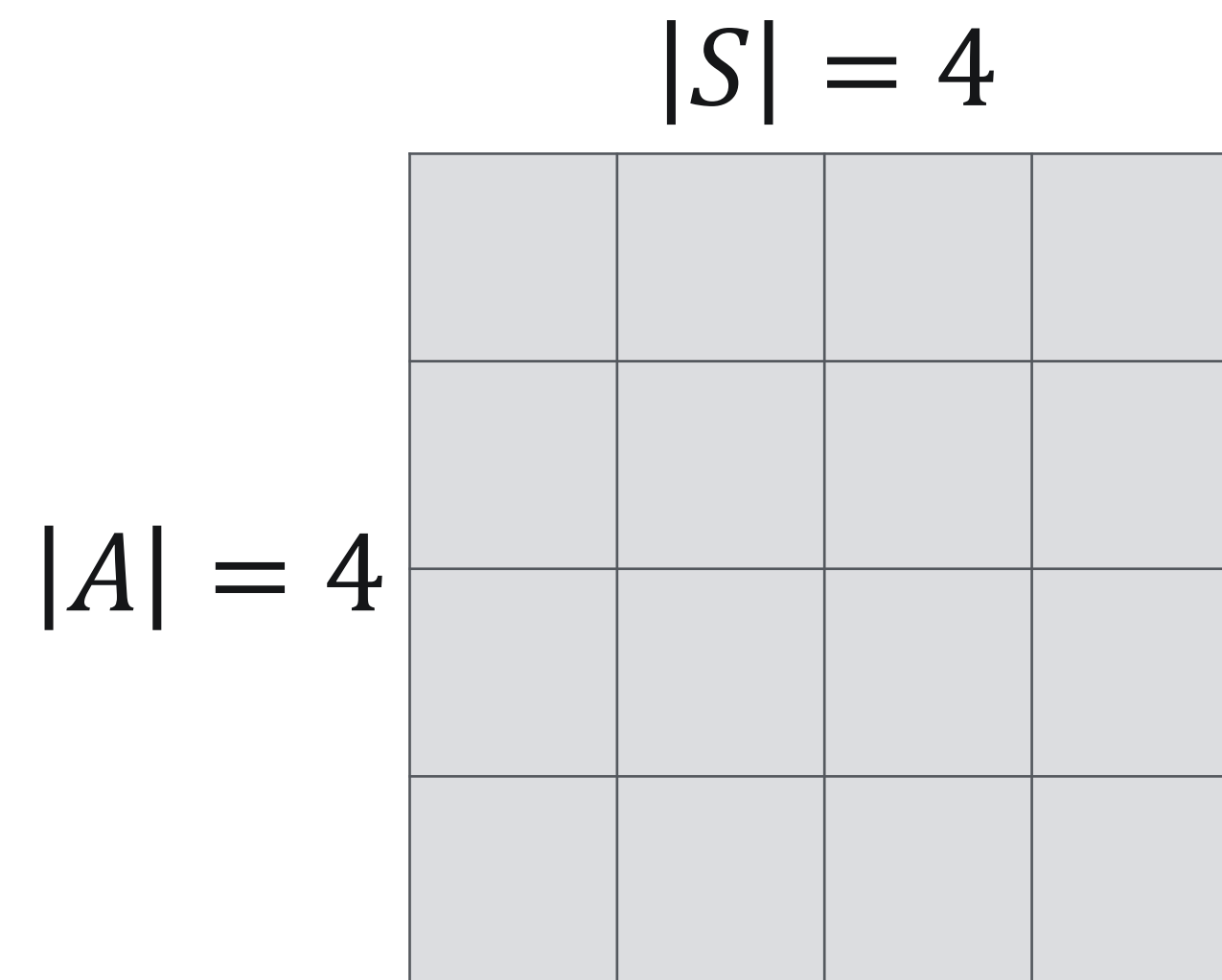


$$4 \log(4) + 4 \log(4) + 4 \log(4)$$

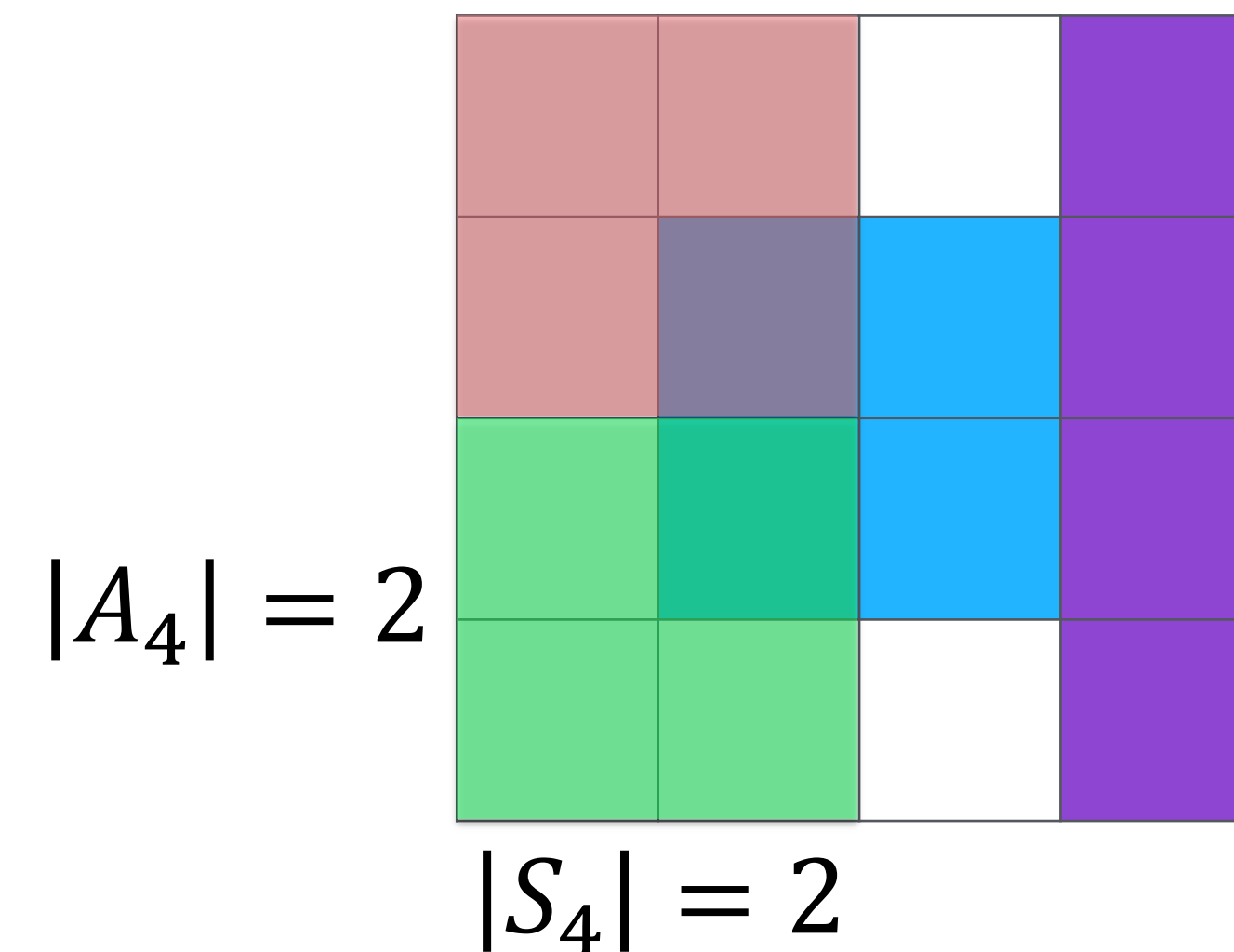


# 3.1 Motivation for Modular RL

Number of samples needed by RL algorithms to learn a near-optimal policy is of the order  $|S||A| \log(|S||A|)$



$$16 \log(16) = 64$$

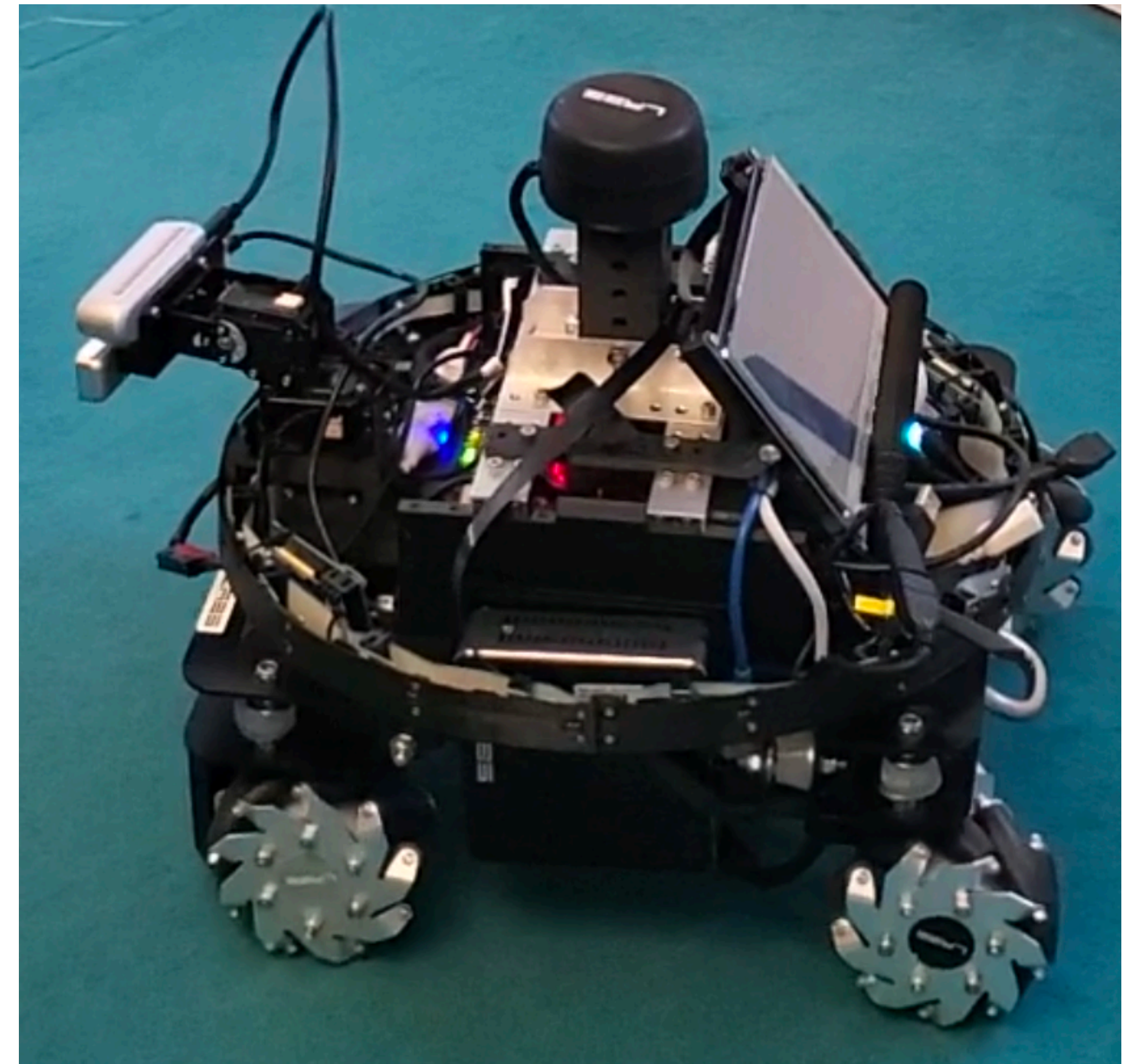


$$4 \log(4) + 4 \log(4) + 4 \log(4) + 4 \log(4) = 8 + 8 + 8 + 8 = 32$$

## 3.2 Example: Visual Navigation

### Robot with moving camera

- Actuators
  - Wheels
  - Camera pan and tilt
- Sensors
  - RGBD Camera
  - Inertial Measurement Unit
  - Camera position

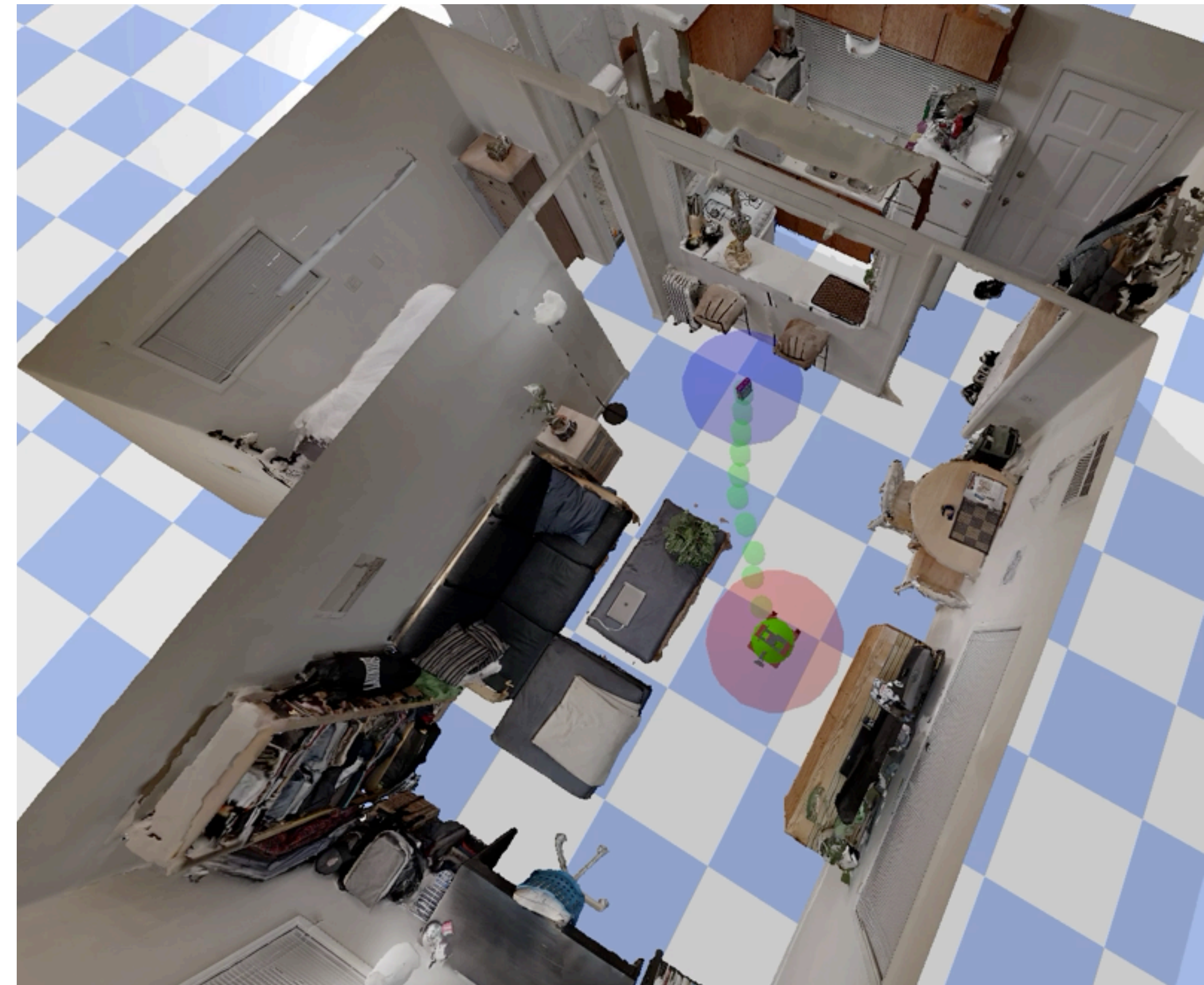




## 3.2 Example: Visual Navigation

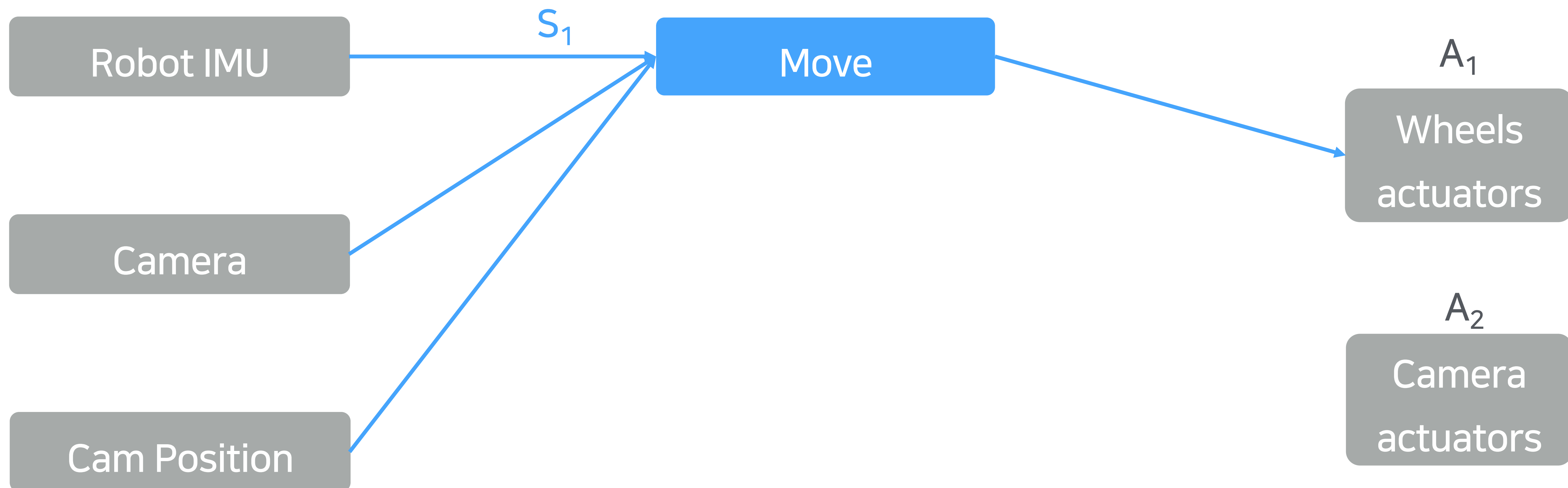
Task: Find a given object and navigate to it while avoiding obstacles

- Camera **scans** the environments
- Robot body **aligns** to the camera
- Once the object is detected, the camera keeps the object in **focus**
- Robot **moves** towards the target
- And robot **avoids** collisions

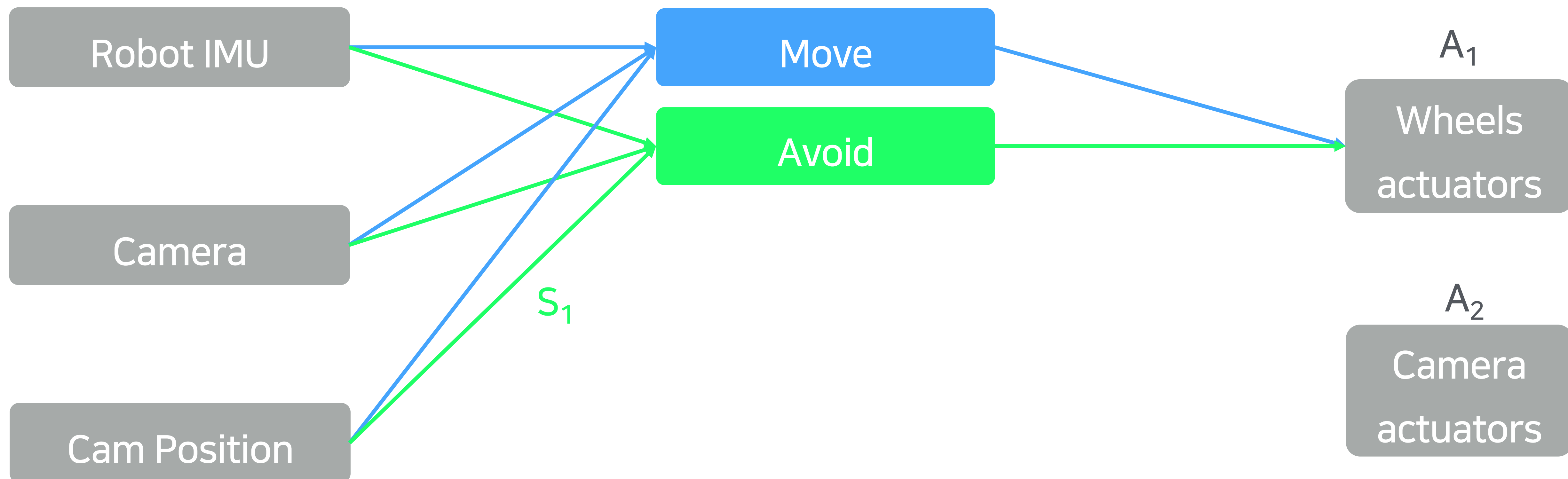




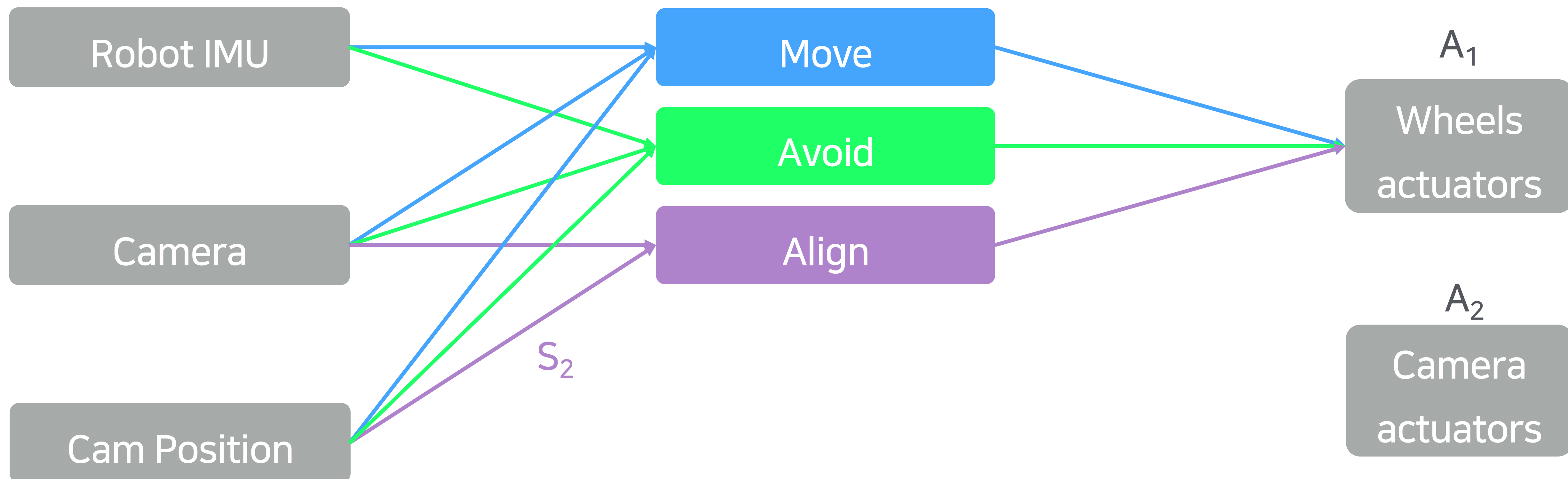
## 3.2 Example: Visual Navigation



## 3.2 Example: Visual Navigation

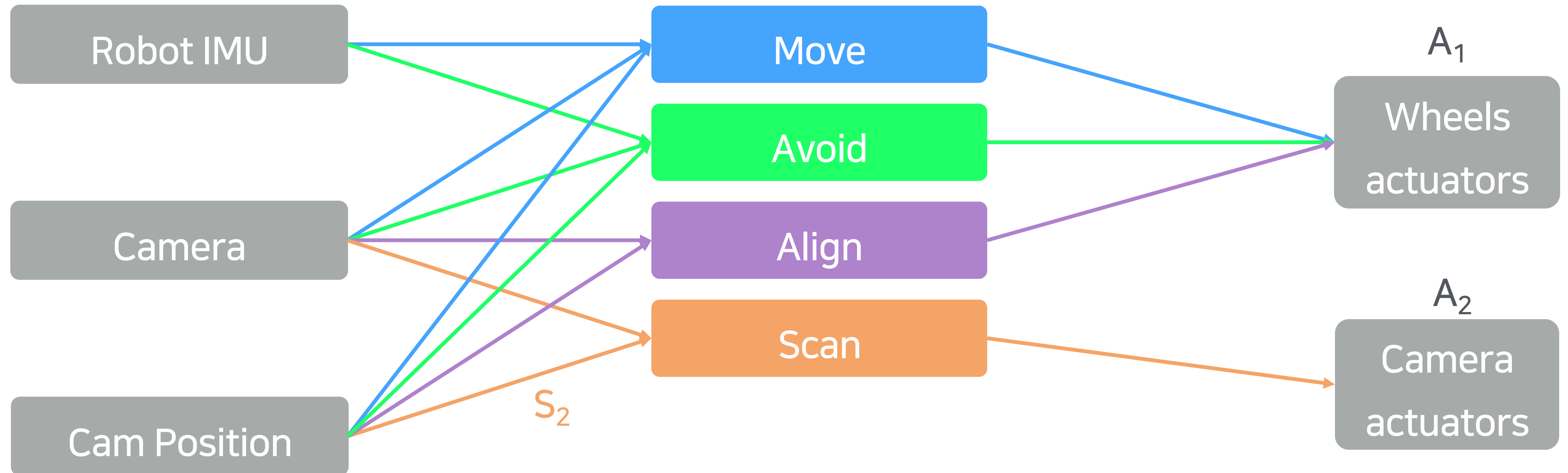


## 3.2 Example: Visual Navigation

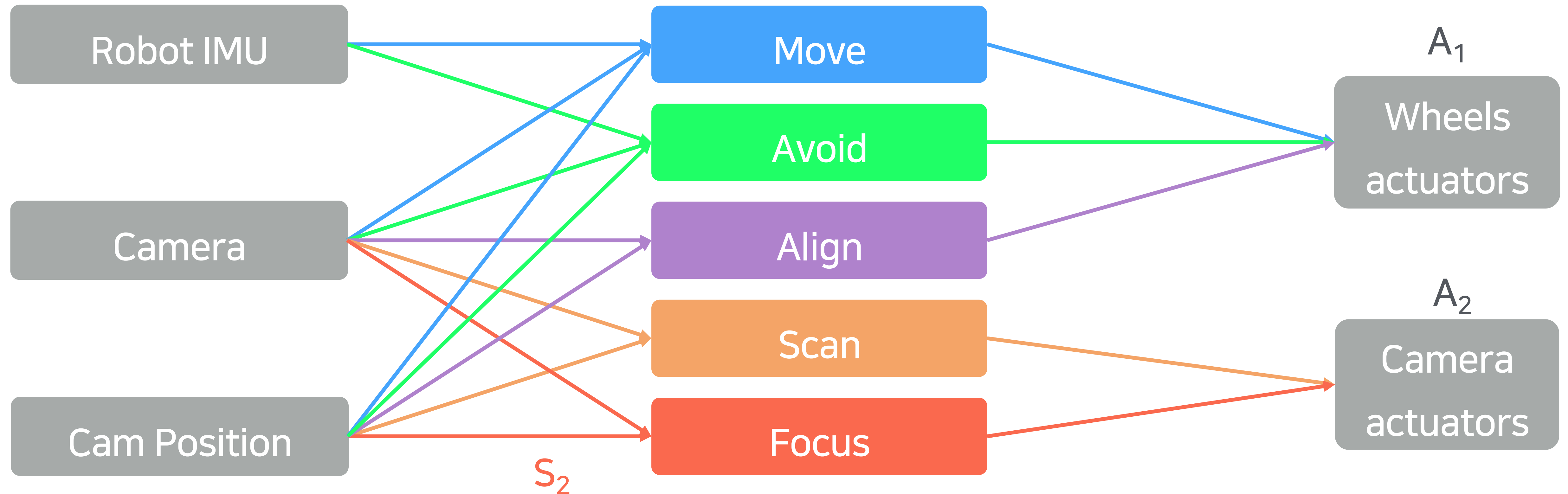




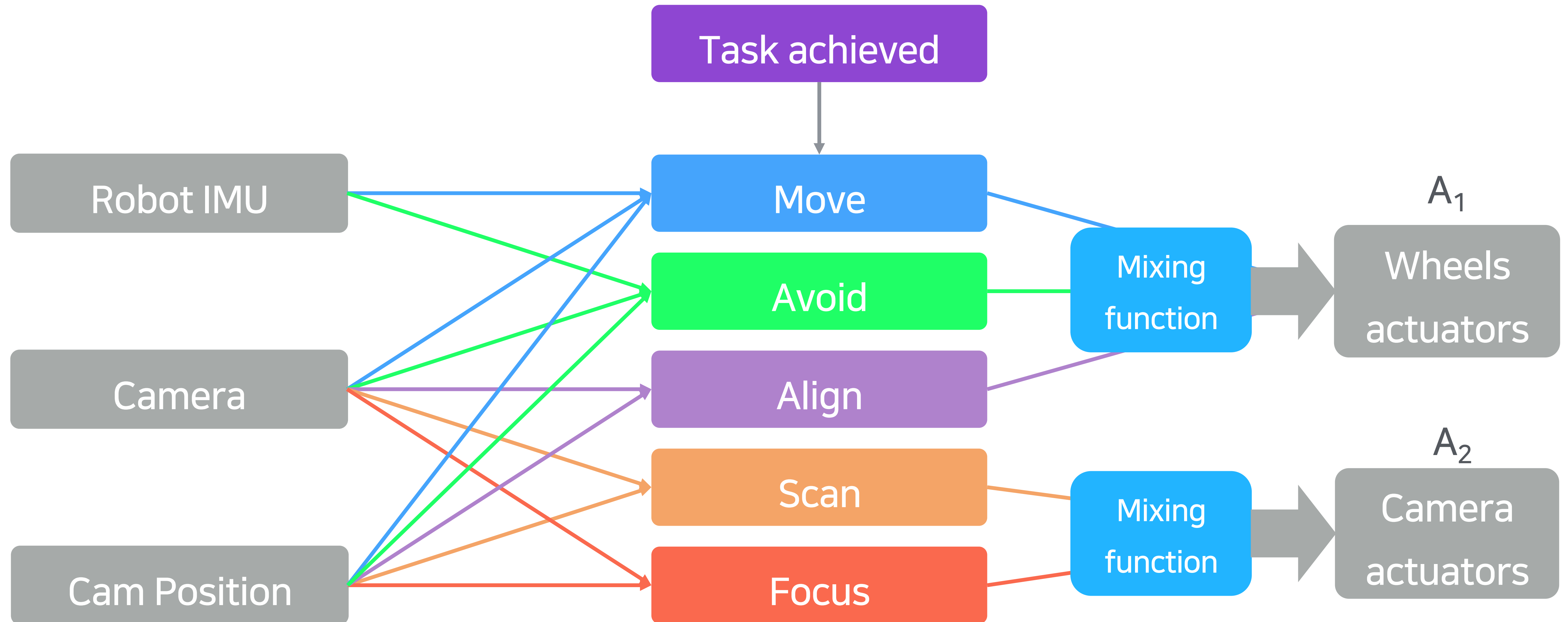
## 3.2 Example: Visual Navigation



## 3.2 Example: Visual Navigation



## 3.2 Example: Visual Navigation





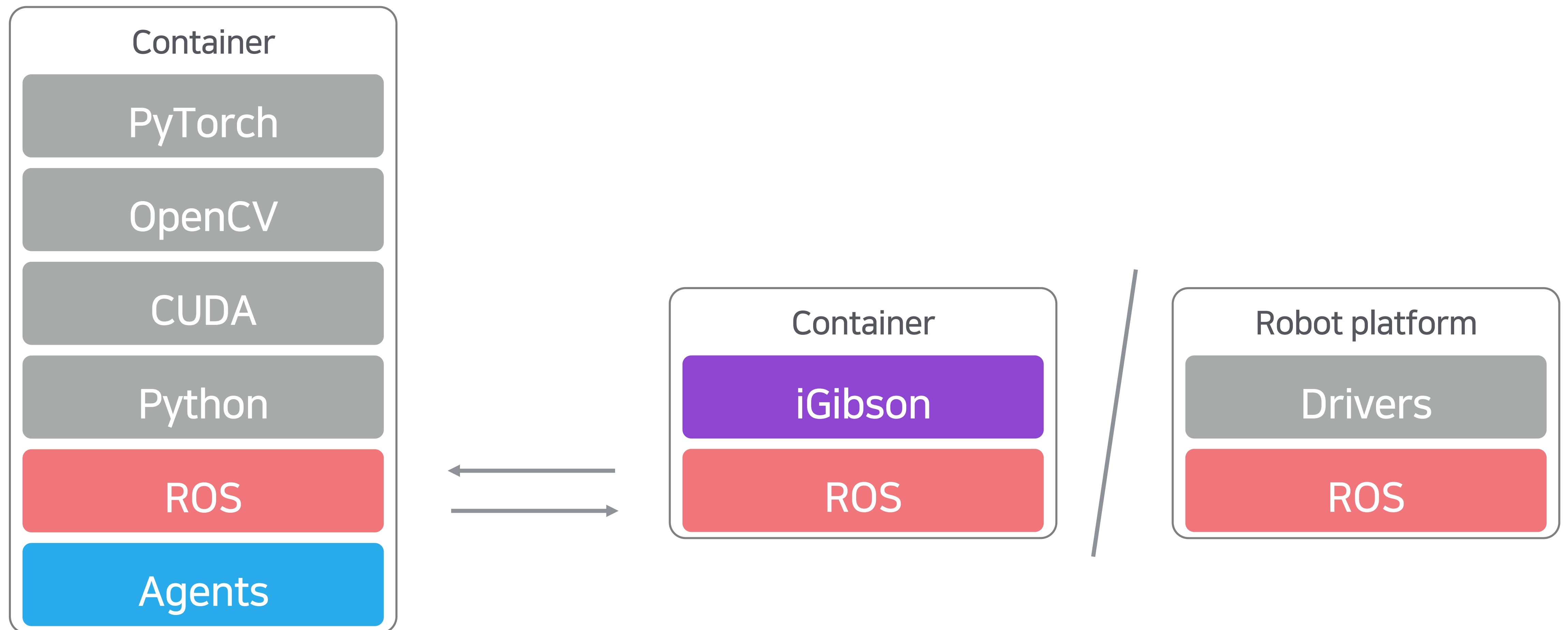
# 3.2 Example: Visual Navigation

| Agent           | How                           | Params | Rewards                                                                                                     |
|-----------------|-------------------------------|--------|-------------------------------------------------------------------------------------------------------------|
| Move            | Learned with RL               | 12     | $R_{move} = \begin{cases} 1, & \text{if robot close to target} \\ 0, & \text{otherwise.} \end{cases}$       |
| Avoid           | Model-based                   |        |                                                                                                             |
| Align           | Model-based                   |        |                                                                                                             |
| Scan            | Model-based                   |        |                                                                                                             |
| Focus           | Learned with RL               | 6      | $R_{focus} = \begin{cases} 1, & \text{if object close to img center} \\ 0, & \text{otherwise.} \end{cases}$ |
| Mixing function | Learned with RL               | 18     | $R_{mix} = R_{move}$                                                                                        |
| Task achieved   | Learned with self-supervision | 4      |                                                                                                             |

# 4. Navigation Experiments

# 4.1 Visual Navigation in Simulator

## Software Stack

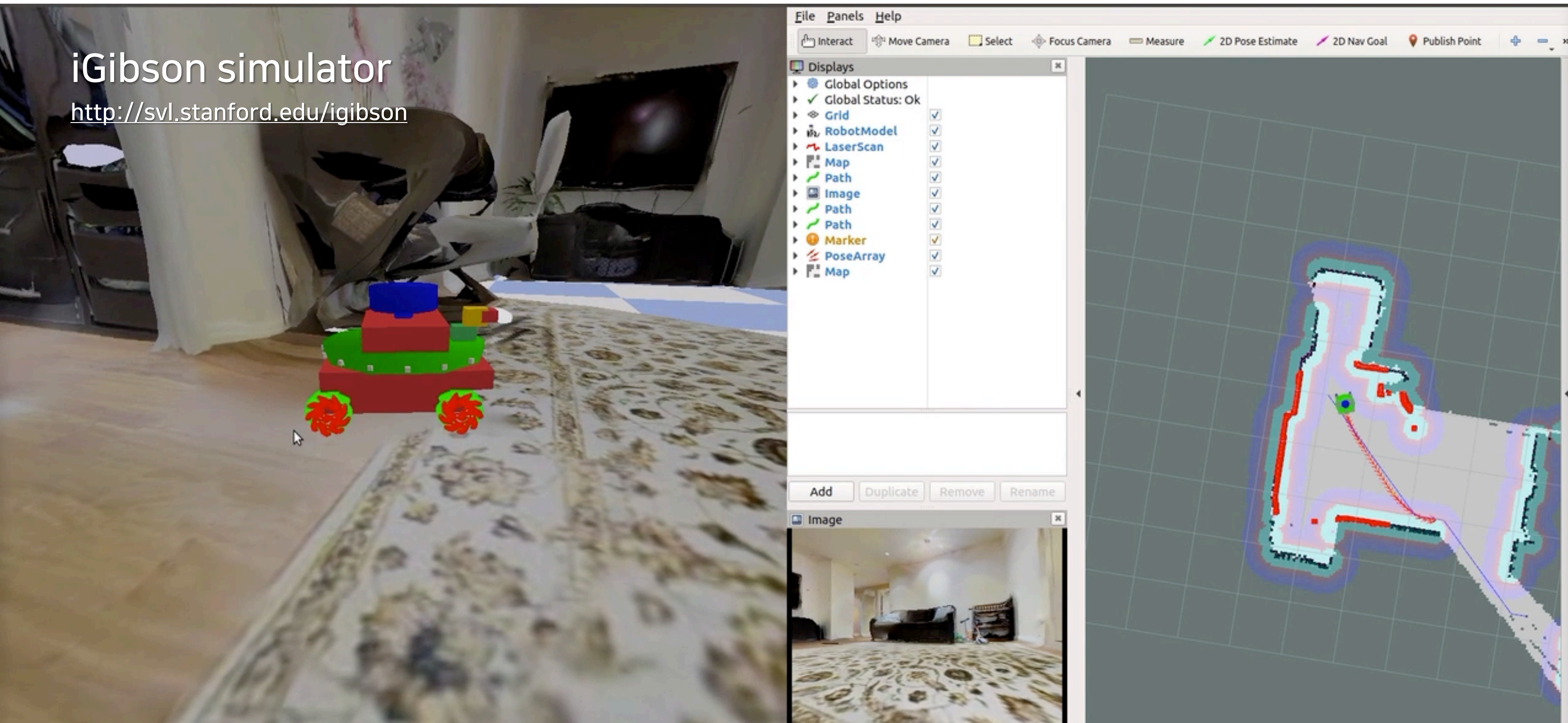




# 4.1 Visual Navigation in Simulator

iGibson simulator

<http://svl.stanford.edu/igibson>



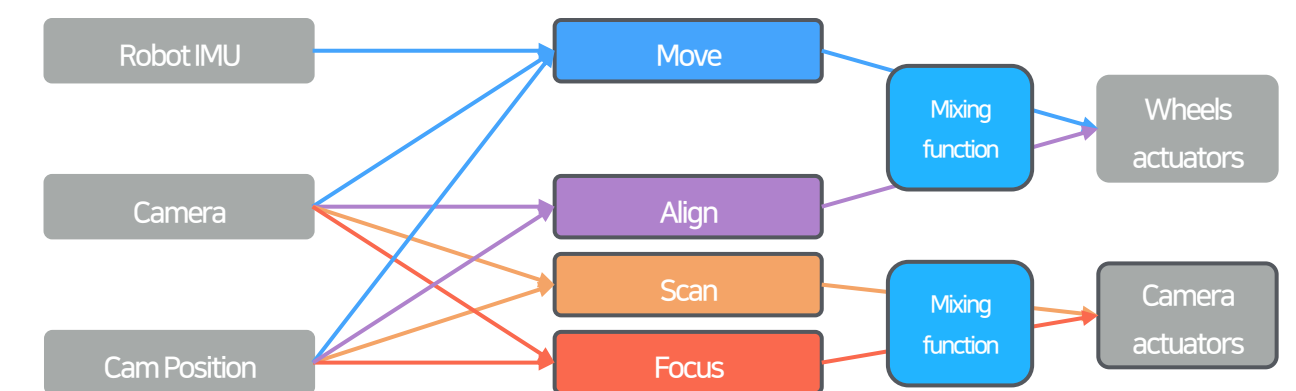


# 4.1 Visual Navigation in Simulator

## Three agents

- Multi-agent modular RL

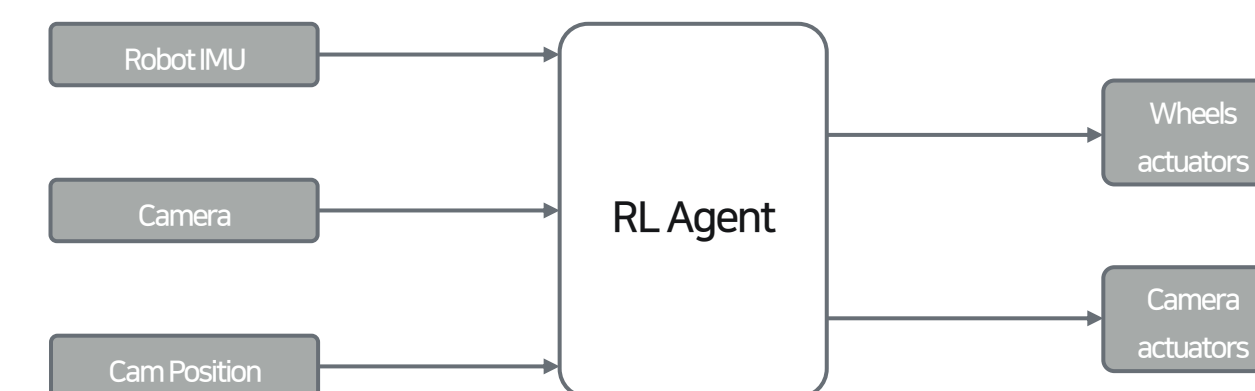
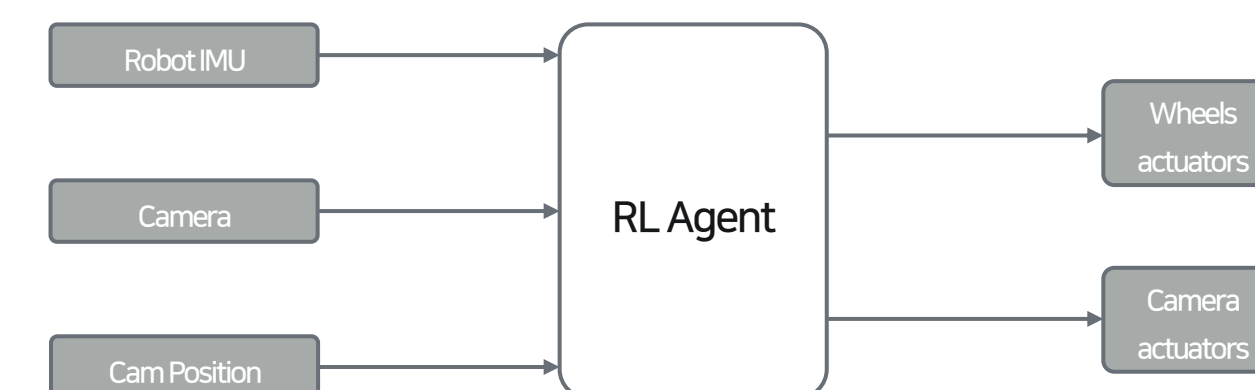
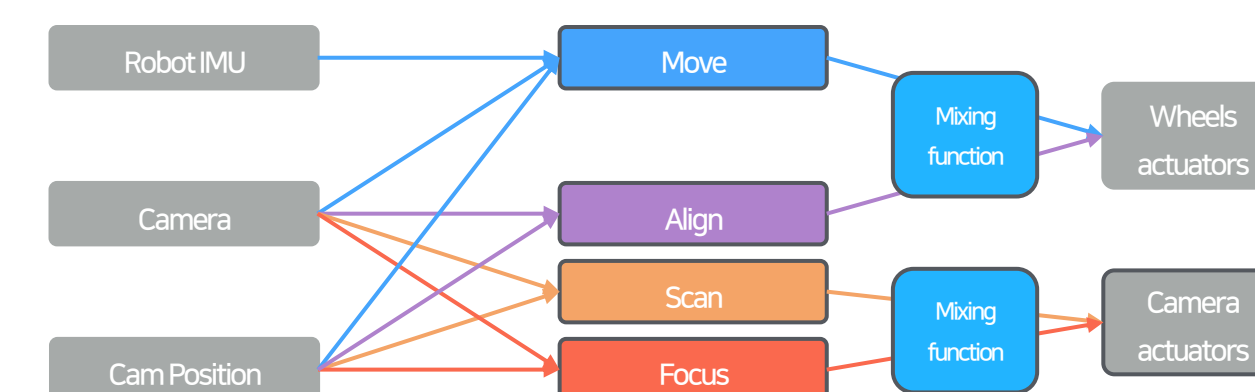
Agent we described, without the **Avoid** module



# 4.1 Visual Navigation in Simulator

## Three agents

- **Multi-agent modular RL**  
Agent we described, without the **Avoid** module
- **Single policy RL agent**  
“Classic” RL agent with same State and Action spaces, reward when reaching the goal
- **Single policy RL agent with reward shaping**  
“Classic” RL agent, but reward is hand-engineered weighted sum of the modular RL rewards





# 4.1 Visual Navigation in Simulator

Task 1: Find object in empty space and navigate to it

| Experiment                | N. Params | Training Budget | SR, single target                 |
|---------------------------|-----------|-----------------|-----------------------------------|
| Single agent w/o shaping  | 40        | 1000            | $0.47 \pm 0.20$                   |
| Single agent w/o shaping  | 40        | 5000            | $0.89 \pm 0.08$                   |
| Single agent with shaping | 40        | 5000            | $0.88 \pm 0.14$                   |
| Our multi-agent robot     | <b>22</b> | <b>1000</b>     | <b><math>0.96 \pm 0.05</math></b> |

Multi-agent robot performs better than single agent, with much lower training budget

# 4.1 Visual Navigation in Simulator

## Task 1: Find object in empty space and navigate to it

| Experiment                | N. Params | Training Budget | SR, single target                 |
|---------------------------|-----------|-----------------|-----------------------------------|
| Single agent w/o shaping  | 40        | 1000            | $0.47 \pm 0.20$                   |
| Single agent w/o shaping  | 40        | 5000            | $0.89 \pm 0.08$                   |
| Single agent with shaping | 40        | 5000            | $0.88 \pm 0.14$                   |
| Our multi-agent robot     | <b>22</b> | <b>1000</b>     | <b><math>0.96 \pm 0.05</math></b> |

Multi-agent robot performs better than single agent, with much lower training budget

## Task 2: Find sequence of objects and navigate to them

| Experiment                | N. Params | Training Budget | SR, target sequence               |
|---------------------------|-----------|-----------------|-----------------------------------|
| Single agent w/o shaping  | 40        | 1000            | $0.44 \pm 0.20$                   |
| Single agent w/o shaping  | 40        | 5000            | $0.72 \pm 0.08$                   |
| Single agent with shaping | 40        | 5000            | $0.77 \pm 0.11$                   |
| Our multi-agent robot     | <b>22</b> | <b>1000</b>     | <b><math>0.94 \pm 0.03</math></b> |

Multi-agent robot more robust, performances do not drop with multiple targets

# 4.1 Visual Navigation in Simulator

## Task 3: Find object and navigate to it in cluttered environment

| Experiment    | Training Budget | Success Rate    |
|---------------|-----------------|-----------------|
| Single policy | 5000            | $0.33 \pm 0.18$ |
|               |                 |                 |
| Multi-Agent   | <b>1000</b>     | $0.64 \pm 0.06$ |
|               |                 |                 |

Multi-agent robot more robust, successfully reaches target 64% of time.



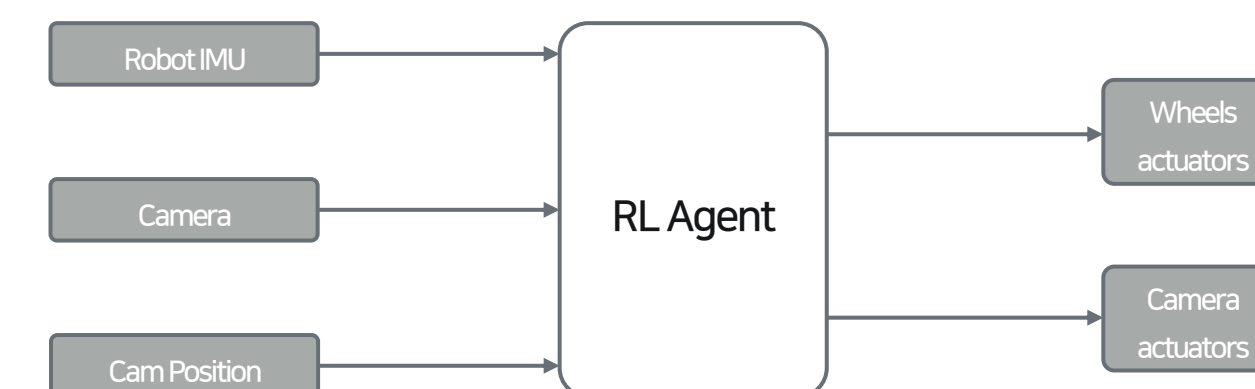
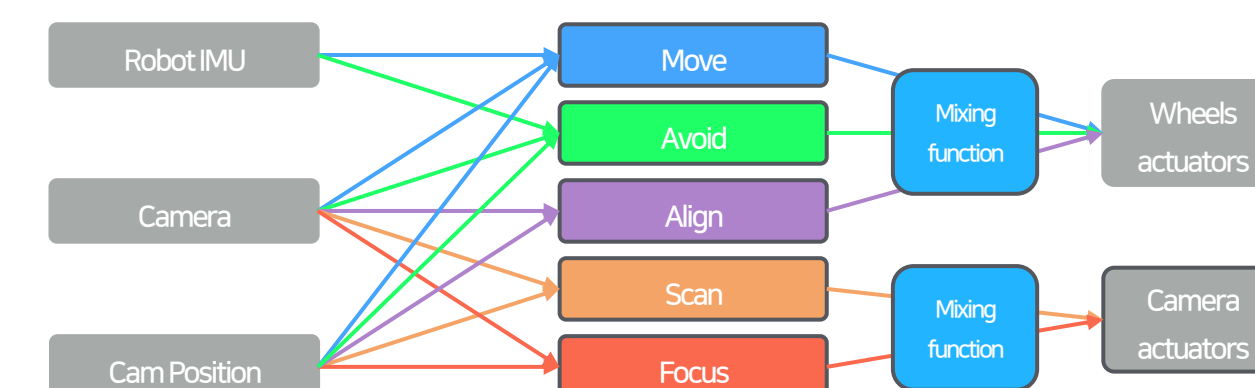
# 4.1 Visual Navigation in Simulator

## Task 3: Find object and navigate to it in cluttered environment

| Experiment    | Training Budget | Success Rate    |
|---------------|-----------------|-----------------|
| Single policy | 5000            | $0.33 \pm 0.18$ |
| Multi-Agent   | <b>1000</b>     | $0.64 \pm 0.06$ |

Multi-agent robot more robust, successfully reaches target 64% of time.

- Add collision avoidance (CA) capabilities
- Modular agent adds rule-based **avoid** module, re-uses policies learned earlier and only trains **mixing function**
- Single policy agent fully re-trained with shaped reward that includes penalty for collision



# 4.1 Visual Navigation in Simulator

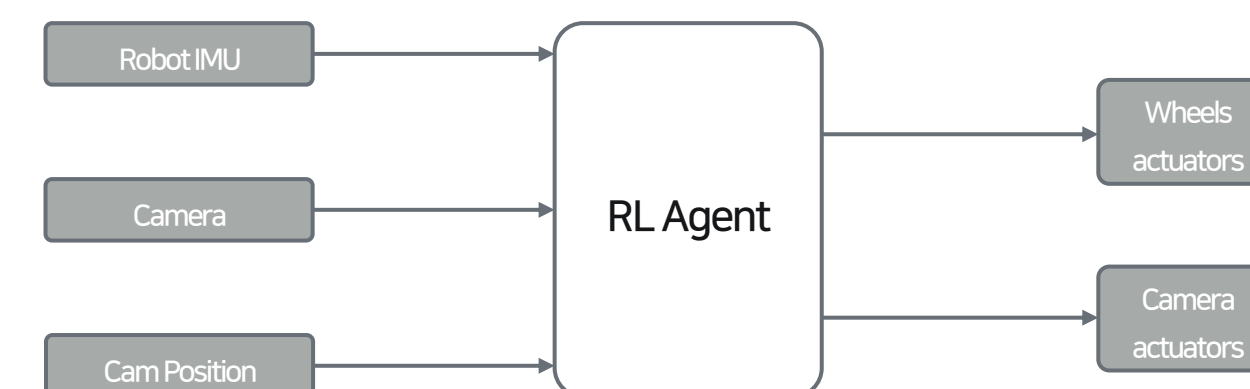
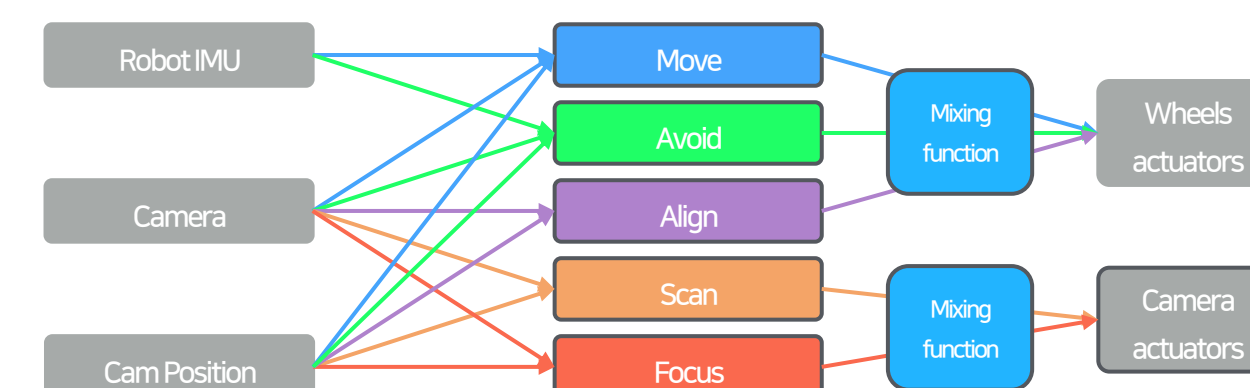
## Task 3: Find object and navigate to it in cluttered environment

| Experiment          | Training Budget | Success Rate                      |
|---------------------|-----------------|-----------------------------------|
| Single policy       | 5000            | $0.33 \pm 0.18$                   |
| Single policy w. CA | 10000           | $0.56 \pm 0.11$                   |
| Multi-Agent         | 1000            | $0.64 \pm 0.06$                   |
| Multi-Agent w. CA   | 2000            | <b><math>0.76 \pm 0.02</math></b> |

Multi-agent robot more robust, successfully reaches target 64% of time.

With little additional training budget, multi-agent performances increase significantly.

- Add collision avoidance (CA) capabilities
- Modular agent adds rule-based **avoid** module, re-uses policies learned earlier and only trains **mixing function**
- Single policy agent fully re-trained with shaped reward that includes penalty for collision

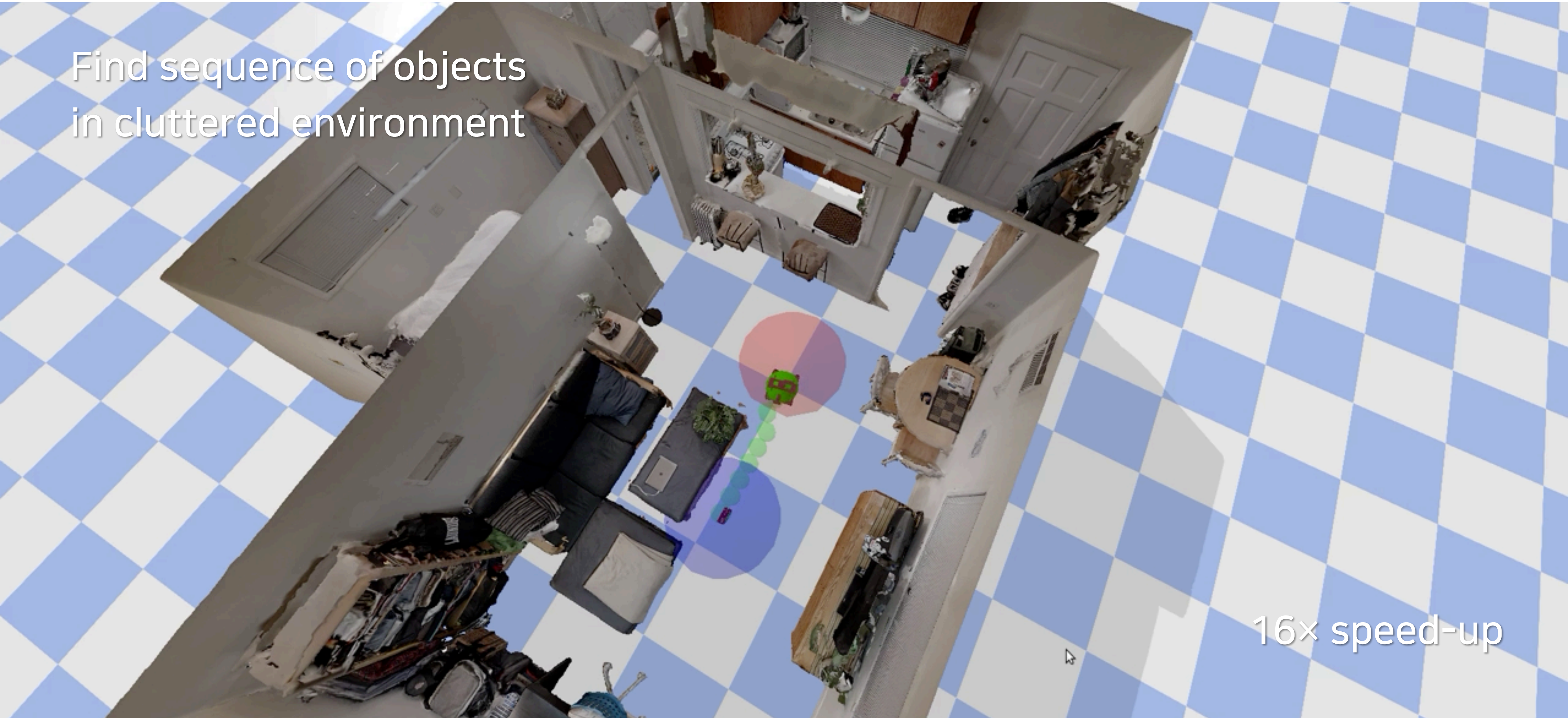




# 4.1 Visual Navigation in Simulator

Find sequence of objects  
in cluttered environment

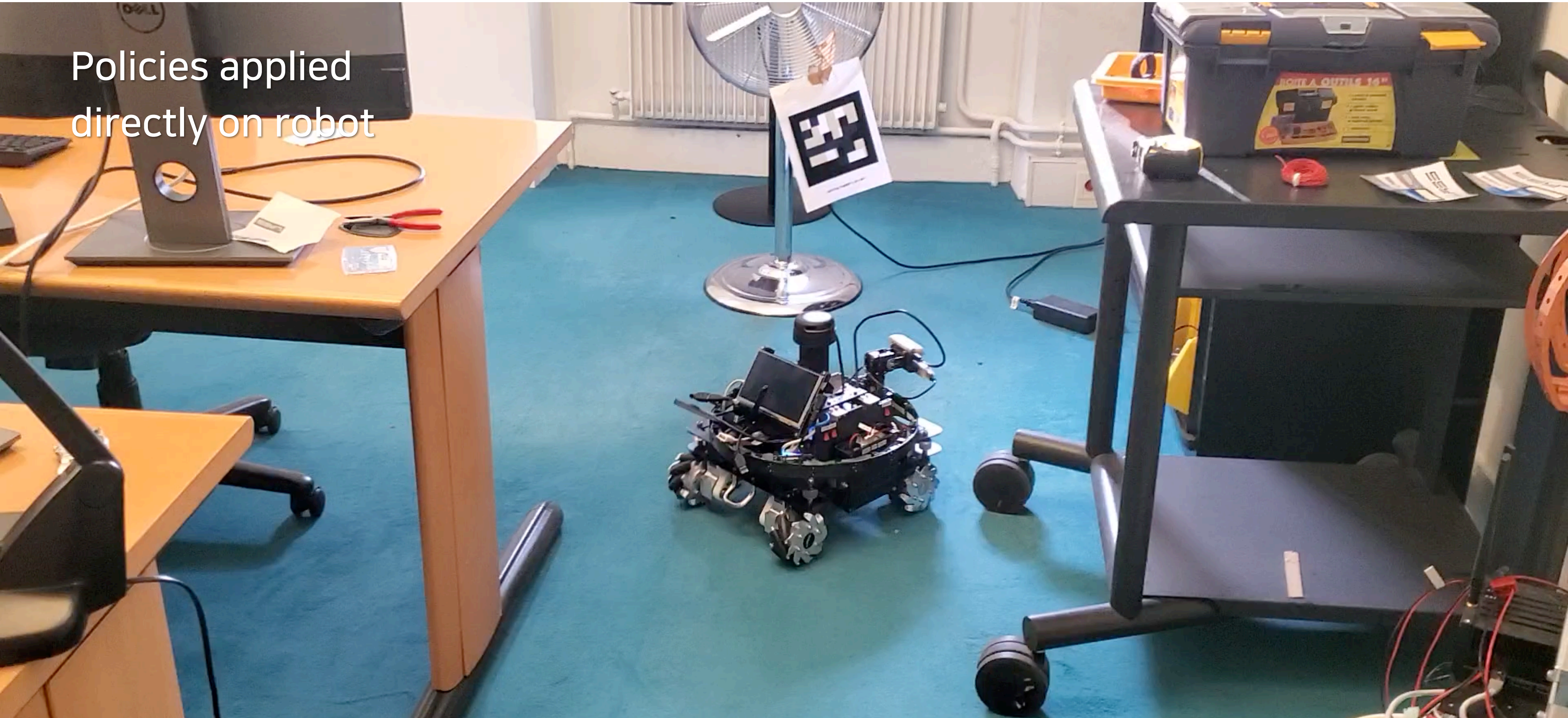
16x speed-up





# 4.2 Visual Navigation on Mobile Robot

Policies applied  
directly on robot





# 5. Concluding Remarks



# 5.1 Lessons Learned

- Promising direction, works better than single policy
- Transfer easily to robot
- Interpretability: easier to debug and fix
- Simulation very useful
  - Allows to make some progress in COVID times
  - Programmatic generation of environments important
  - But still, need to go back and forth between robot and sim

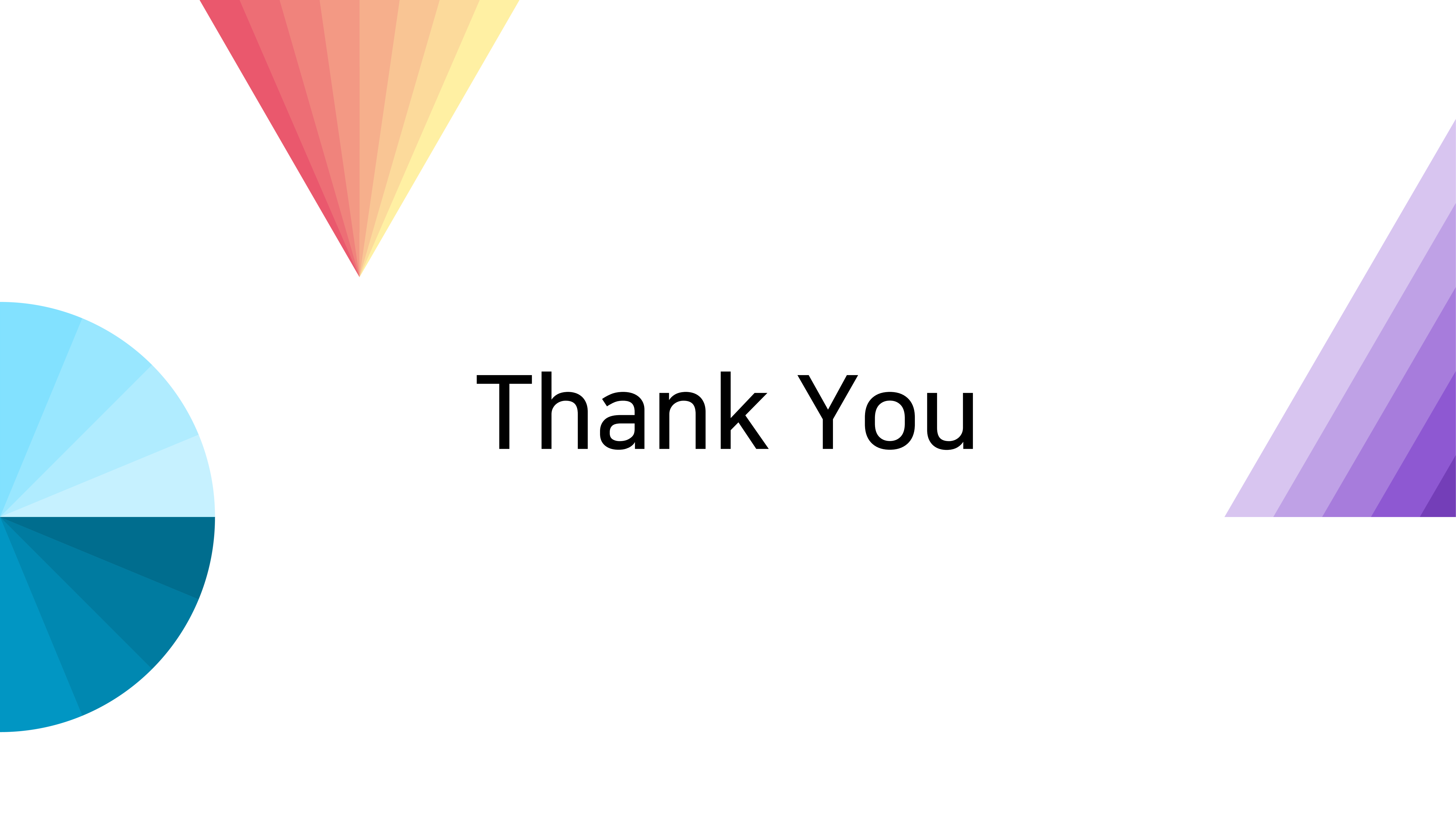


## 5.2 Next Steps

- More complex the task, bigger the advantage
  - Multiple tasks
  - More complex, realistic use cases
- Combine learn in sim and in robot
- Learn sensing functions (like mixing functions)
- Learn connections between modules



# Q & A



Thank You