VRIJE UNIVERSITEIT AMSTERDAM

TASK 2 - FORAGING

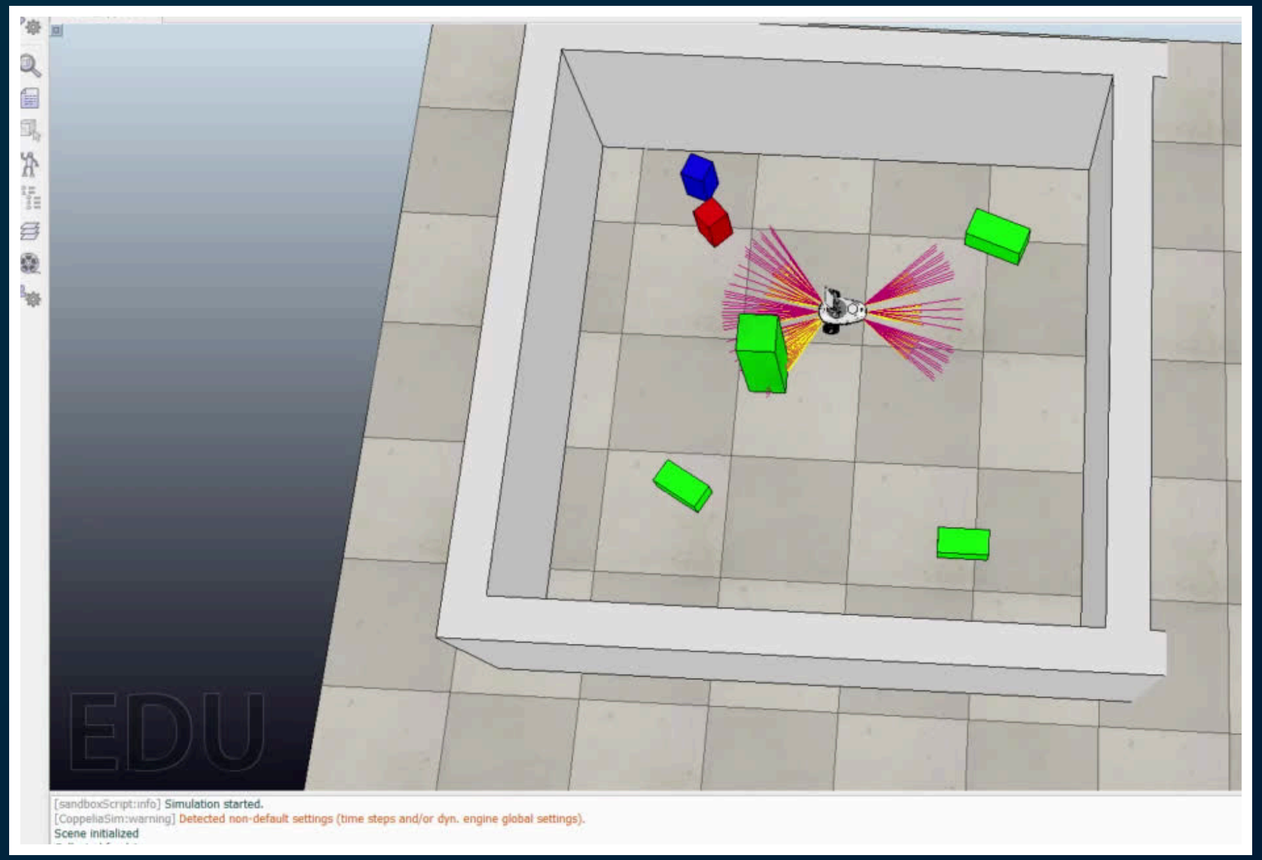
LEARNING MACHINES

Robobo

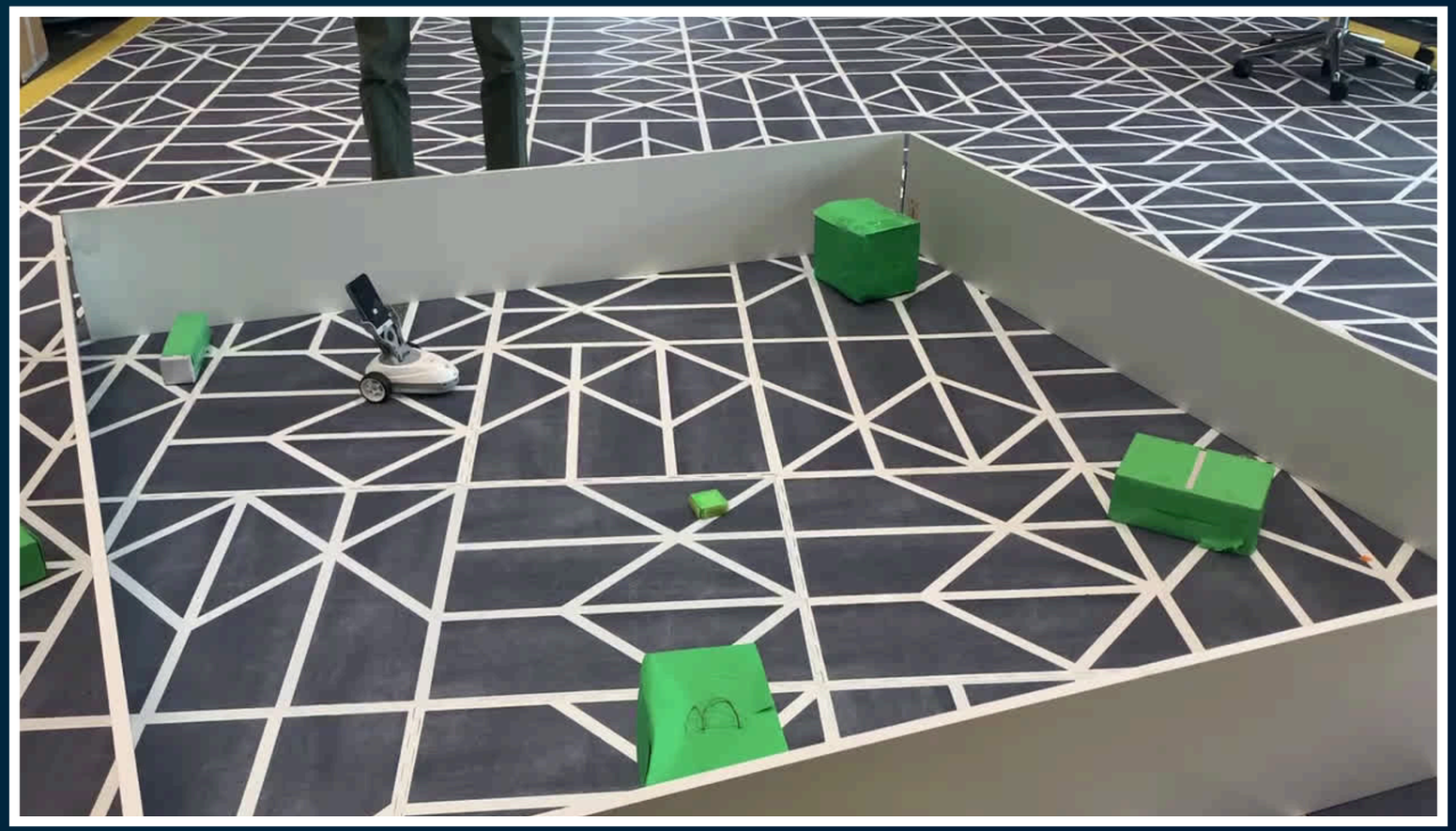
GROUP 14

Stergios Ntanavaras
Thijs Grootjans
Quirinus de Ruijter

SIMULATION DEMO



HARDWARE DEMO



PROBLEM & MOTIVATION

- **Objective:** Develop an AI-driven robot that learns to touch and collect the largest number of “food” items (packages) per minute within a given environment.

- **Challenges:**
  - Designing algorithms that can quickly and efficiently learn from interactions with the environment.
  - Requires advanced perception and adaptability.
  - Complex decision-making.
  - Solutions that work well in a virtual environment might not scale effectively to real-world scenarios.
  - Exogenous factors can significantly affect the performance of sensors like cameras.

PROBLEM & MOTIVATION

- **Utility and Importance to AI Community:**
  - Pushes the development of algorithms, particularly in perception and real-time decision making.
  - Leads to significant improvements in efficiency and cost reduction in various industries.
  - Creates the foundations for autonomous systems operating in the real-world.
  - Could revolutionize operations to industries like logistics, warehousing and manufacturing.

METHODOLOGY

- **Algorithm:** Q-learning, a model-free RL algorithm.
  - *Purpose:*
    - Develop a strategy for the robot to autonomously learn and optimize its actions in the environment.
  - *Goals:*
    - Enable the robot to learn from its own actions and experiences.
    - Improve the robot's actions to maximize the number of items touched and collected.
    - Refining decision-making over time.
  - *Learning Process:*
    - Initializes Q-values at zero.
    - Updates them based on rewards and future reward estimates:
      - $Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max Q'(s', a') - Q(s, a)]$

METHODOLOGY

- **Object Detection:** Analyzing the distribution of green pixels by capturing and processing the robot’s camera.
  - *Steps:*
    - Capture the current view from the robot's front-facing camera.
    - Clip the image to focus on the relevant part.
    - Calculate the percentage of green pixels : (#G/#T) × 100
    - Analyze greenness distribution:
      - Dividing the image into 4 vertical sections and focusing on the middle 2 (center view).
      - Identifying the section with the highest number of green pixels.
  - *Goal:* Create a state that provides the robot a location of the nearest green object.

METHODOLOGY

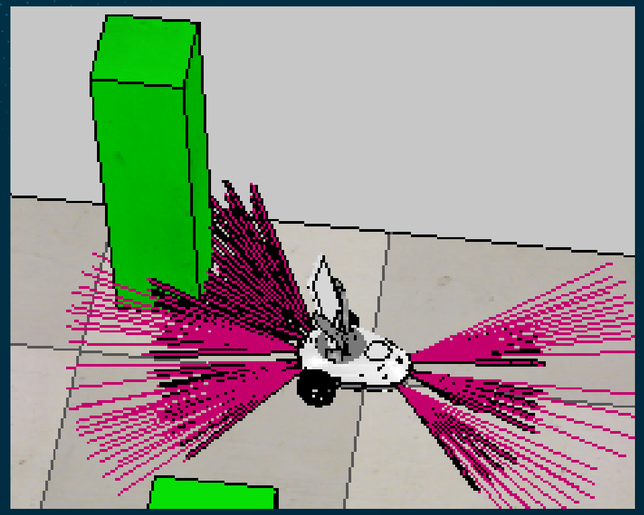
- **Sensing and Reality Gap:**
  - Sensors used:
    - Front Central IR sensor & Mobile Camera.
  - Infrared Sensor Binning Logic:
    - Converting raw infrared sensor values into discrete bins based on the defined thresholds.
  - Greenness Binning Logic:
    - Categorizing the greenness percentage into discrete bins.
  - Different thresholds for simulation and hardware, due to Reality Gap.

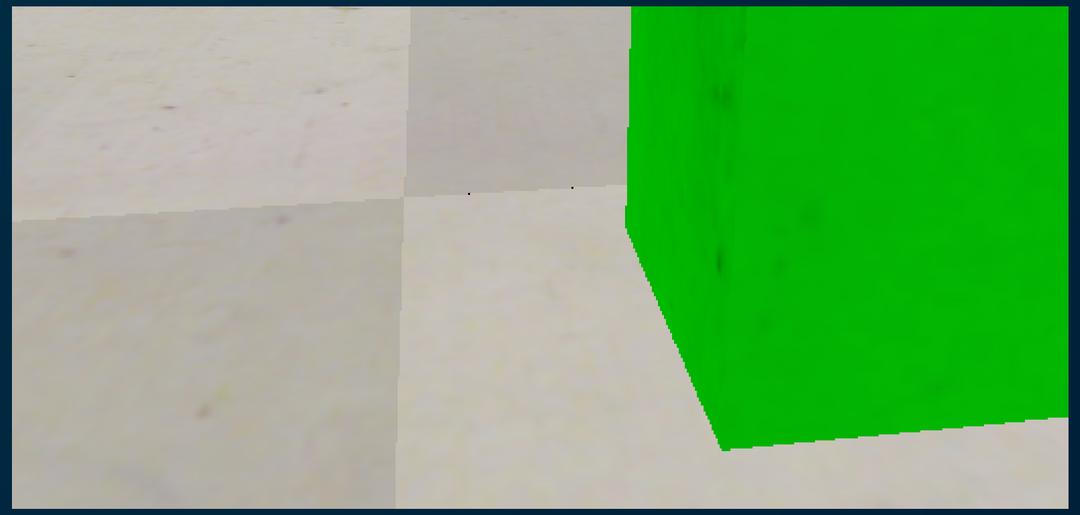
EXPERIMENTAL SETUP

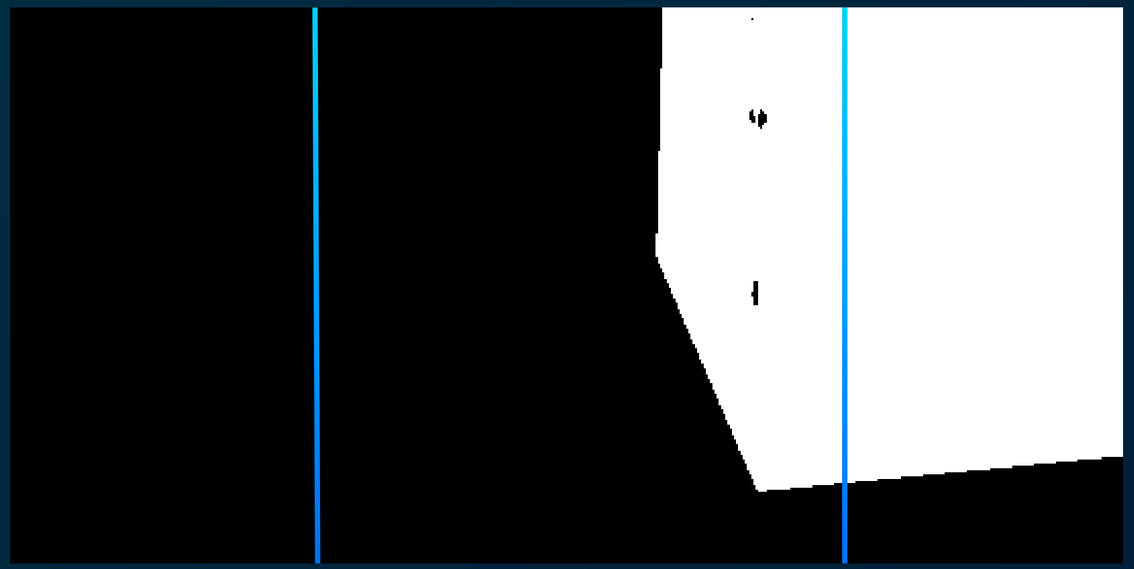
- Three actions *(‘forward’*, *‘left’*, and *‘right’)* were used determined by sensor-based state readings.
- The *‘Epsilon-Greedy Algorithm’* was used for exploration and exploitation.
- 2 sensors were used: Front Central IR sensor & Mobile Camera.
- 4 sensor-bins were used, and different bin-thresholds for simulation and hardware ([4, 7, 25] [-1, 20, 60]).
- 4 greenness-bins were used, and different bin-thresholds for simulation and hardware ([1, 20, 45] [1, 10, 40]).
- 4 green-direction-bins were used, which takes the max amount of pixels in a direction ([0, 1, 2, 3])
- 50 episodes were used for training, and up to 150 steps were used per episode.
- An *epsilon* of 0.3 was used, allowing for 30% random action selection.
- *Training:*
  - Conducted in simulation.
- *Evaluation:*
  - Total number of collected items per episode to measure the effectiveness of navigational decisions.

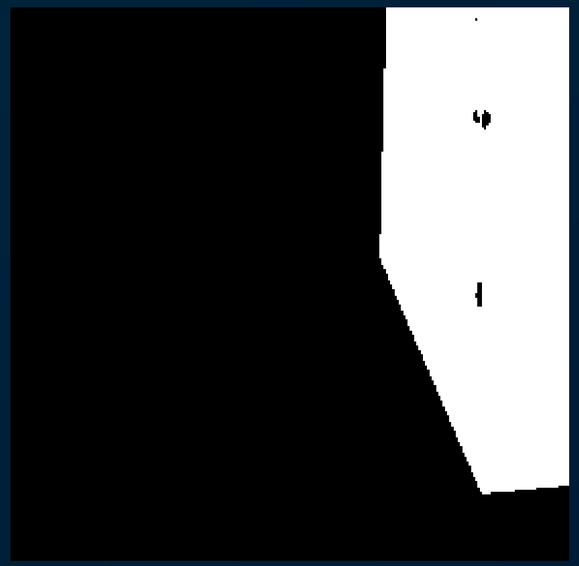
EXPERIMENTAL SETUP

VRIJE UNIVERSITEIT AMSTERDAM









IR value:
9

Green pixels:
[0, 16509, 27233]

25% green in center

State:
(2, 2, 3)

Q–values for this state:
[24.5..., 286.7..., 63.9...]

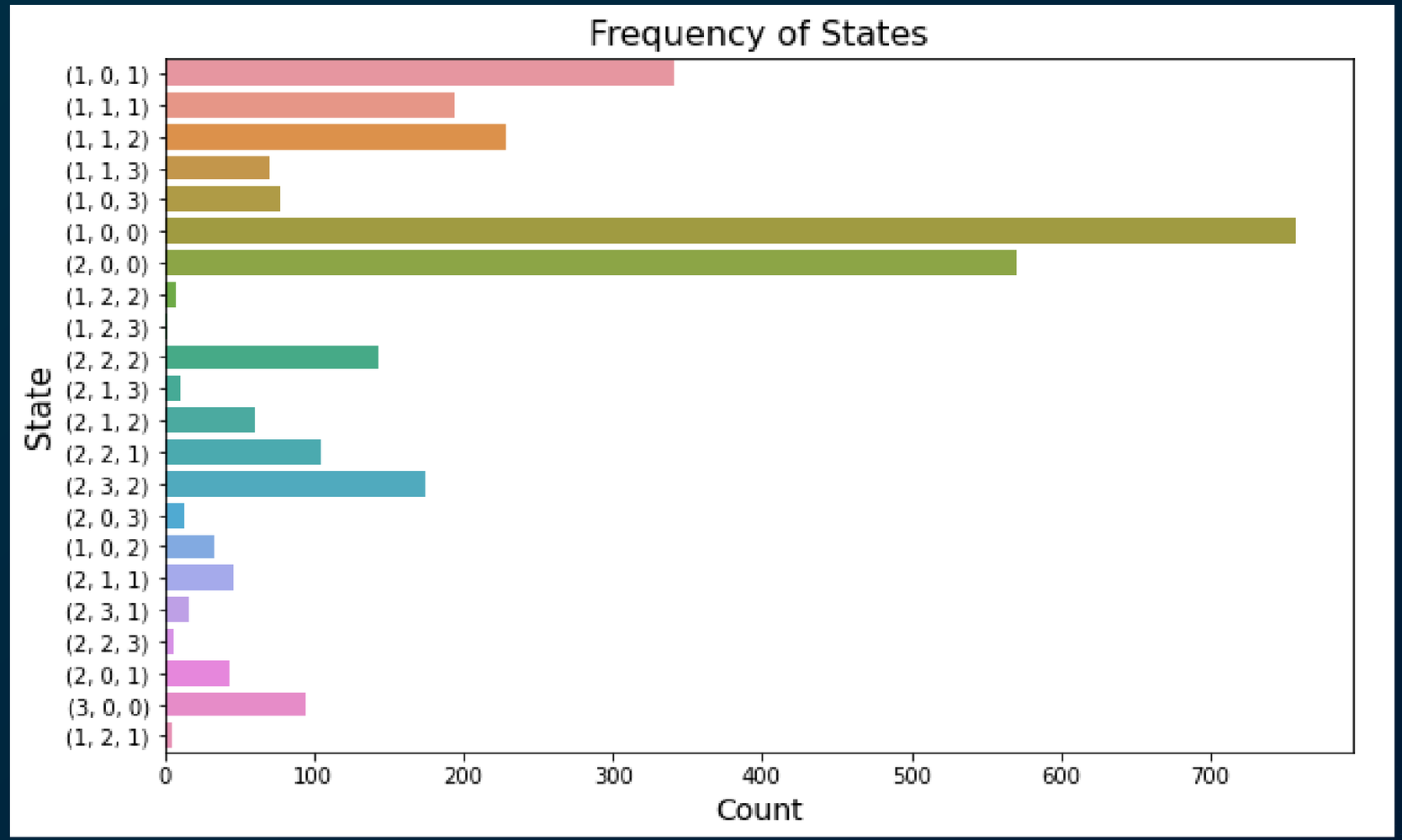
Go forward!

EXPERIMENTAL SETUP

- **Challenges Encountered:**
  - Determine an appropriate reward function.
  - Using only 1 sensor → robot gets stuck in wall often.
  - Reality gap:
    - Finding a green pixel filter is much harder in reality.
    - Robot leans slightly to the left when moving forward.

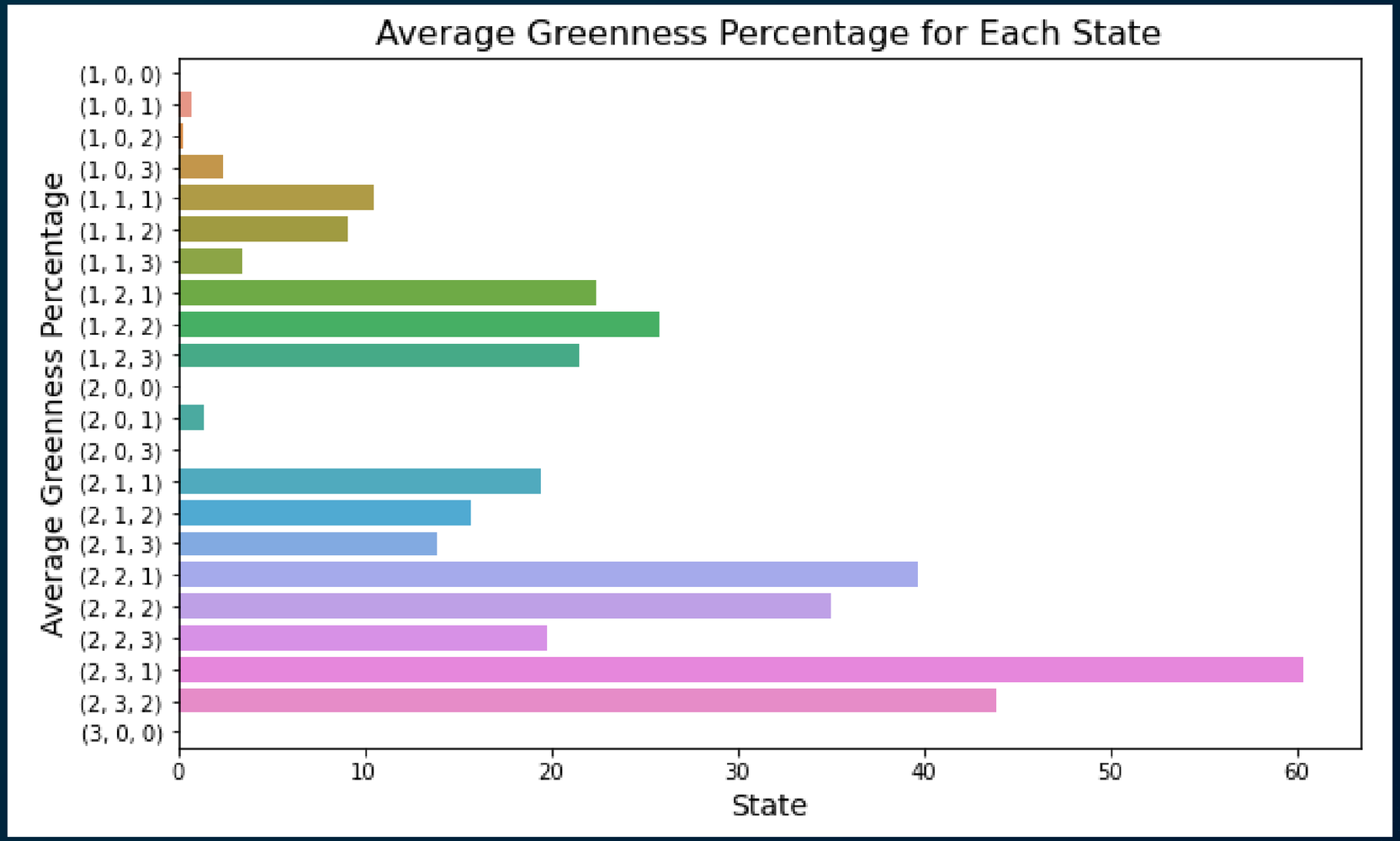
- **Solution Implemented:**
  - Trial and error.
  - Alter reward function to motivate robot to steer away when stuck.
  - Experiment with RGB and camera to see what the robot actually picks up.

RESULTS: WHAT STATES ?



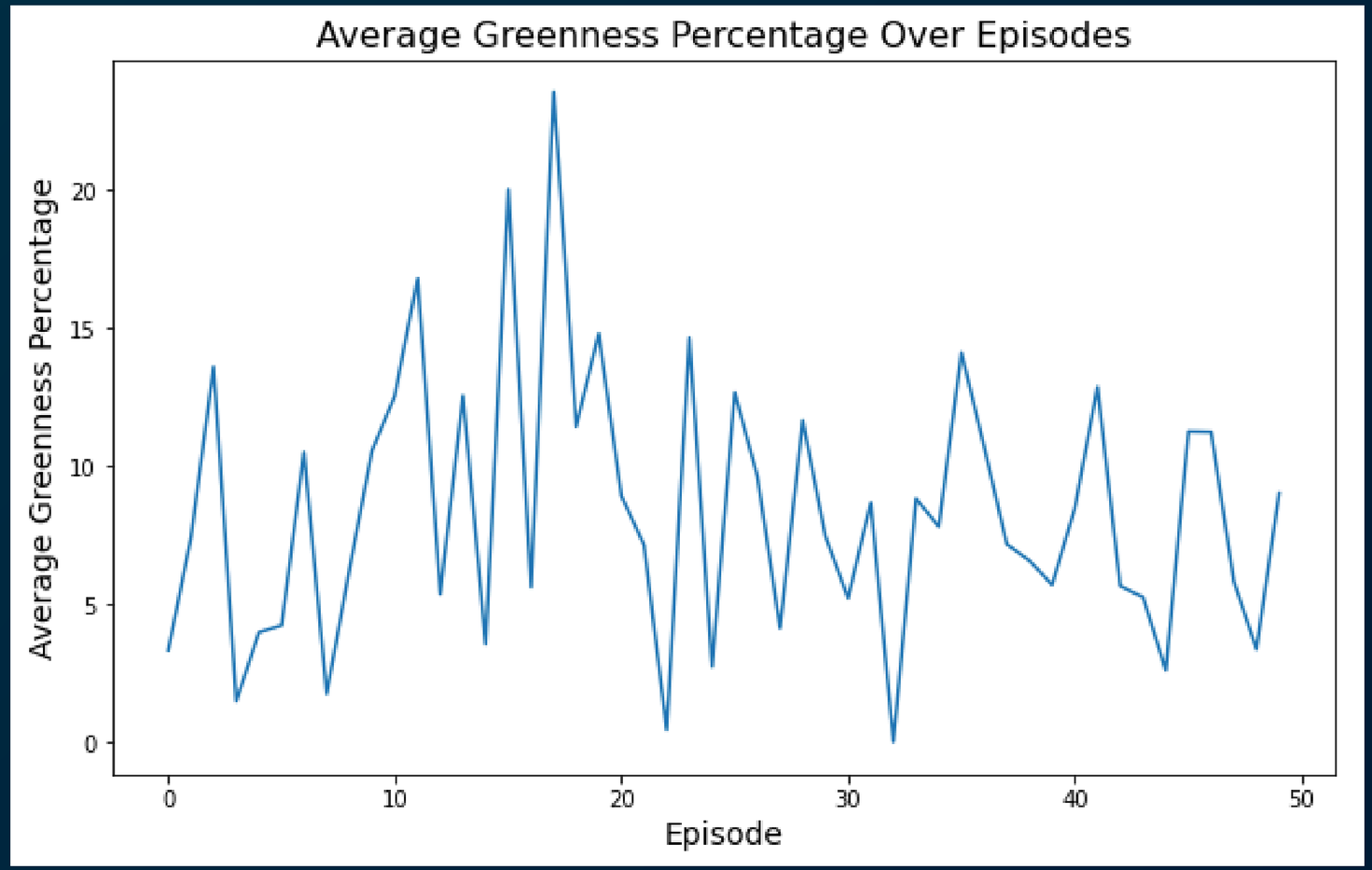
RESULTS: DO WE DETECT GREENNESS?

VRIJE UNIVERSITEIT AMSTERDAM



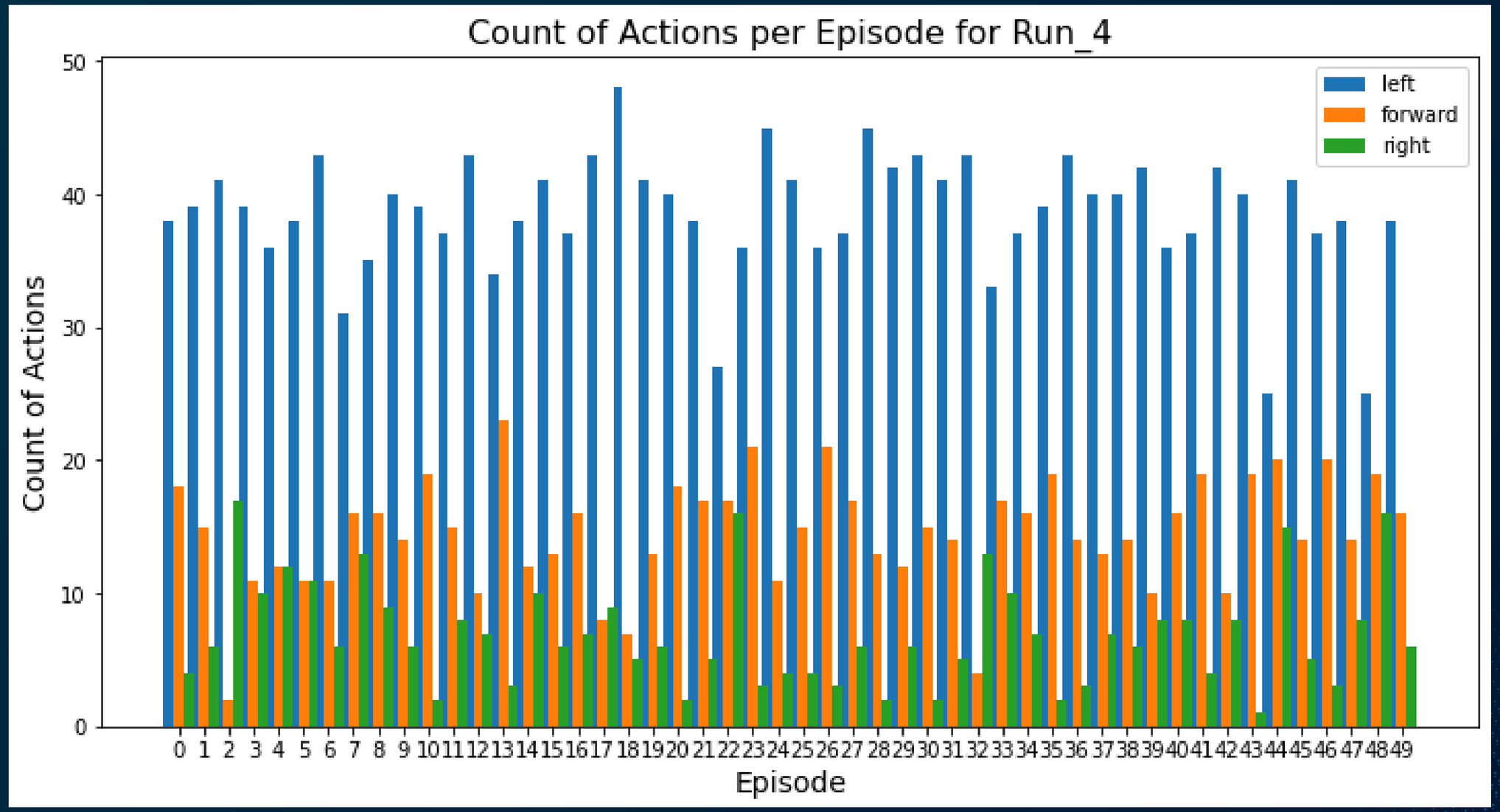
RESULTS: HOW FAST DO WE LEARN?

VRIJE UNIVERSITEIT AMSTERDAM



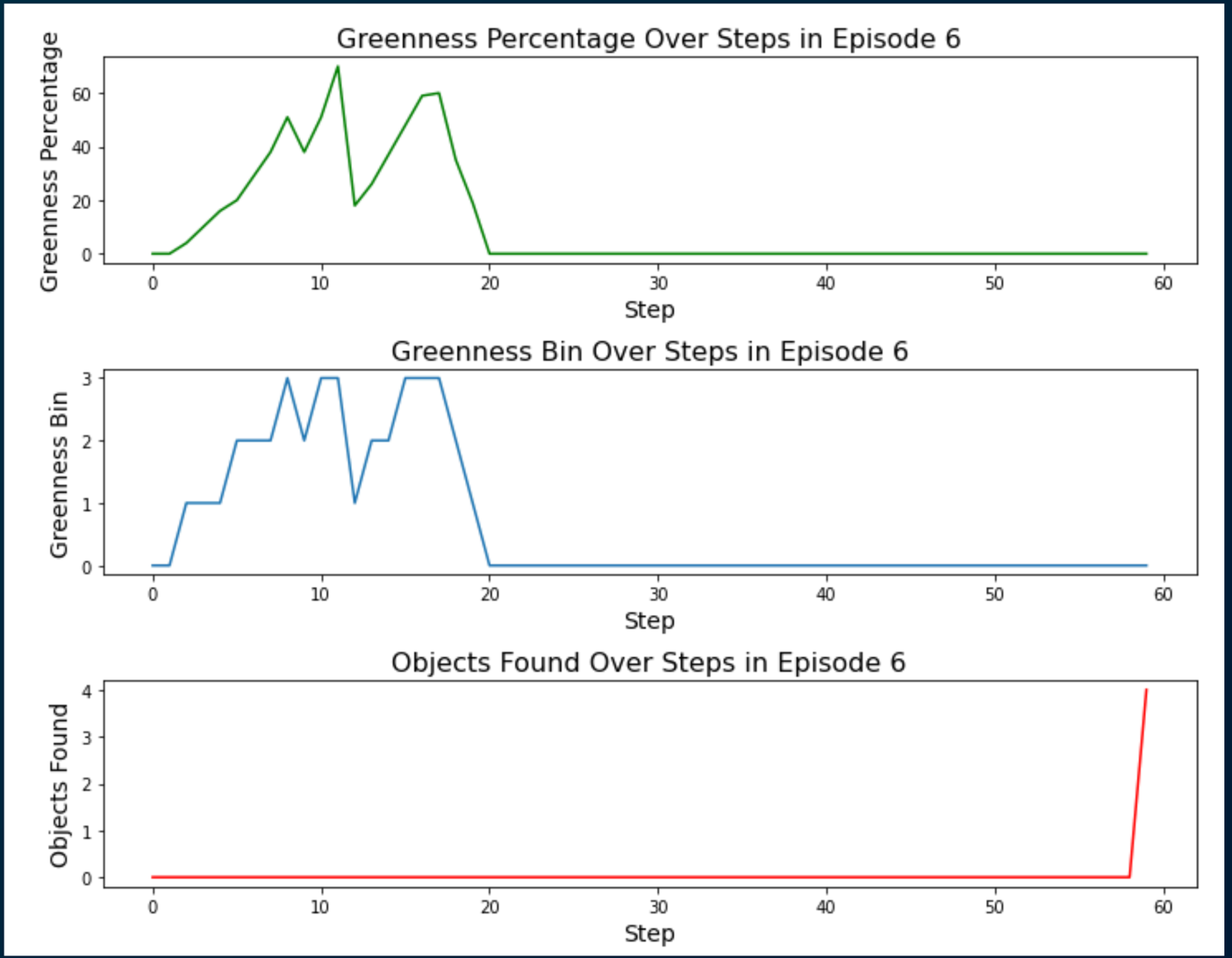
RESULTS: HOW FAST DO WE LEARN?

VRIJE UNIVERSITEIT AMSTERDAM



RESULTS: HOW FAST DO WE LEARN?

VRIJE UNIVERSITEIT AMSTERDAM



RESULTS: DO WE UNDERSTAND Q-TABLE?

State	Q-values (left, forward, right)
(1, 0, 0)	[89.12125342284683, 89.46763971700194, 79.54782758554464]
(1, 0, 1)	[189.9621171481047, 132.28667653201498, 73.31675587217163]
(1, 0, 2)	[67.54000128837055, 132.06308072521648, 36.38654302973711]
(1, 0, 3)	[86.86689297179747, 107.29163940066005, 186.0146593288292]
(1, 1, 0)	[0.0, 0.0, 0.0]
(1, 1, 1)	[257.5011812105793, 136.71256836524378, 128.81471541970532]
(1, 1, 2)	[106.3165210430226, 222.79536969759133, 143.14049038045465]
(1, 1, 3)	[37.09893972321737, 70.37274988961495, 180.73849954452137]
(1, 2, 0)	[0.0, 0.0, 0.0]
(1, 2, 1)	[273.10232825217696, 60.34970107310593, 40.01060276725058]
(1, 2, 2)	[54.43743412250306, 279.6672515946146, 95.54361230177116]
(1, 2, 3)	[10.407063501407588, 67.2478242616412, 0.0]
(1, 3, 0)	[0.0, 0.0, 0.0]
(1, 3, 1)	[51.53408731337623, 0.0, 0.0]
(1, 3, 2)	[0.0, 0.0, 0.0]
(1, 3, 3)	[0.0, 0.0, 0.0]
(2, 0, 0)	[61.63453784627899, -14.8447986810351, -2.210094243525016]
(2, 0, 1)	[156.33720459509175, 7.324134771738009, -8.290241499933833]
(2, 0, 2)	[0.0, 0.0, 0.0]
(2, 0, 3)	[35.01709326465945, 5.846646319521613, 284.9003000029957]
(2, 1, 0)	[0.0, 0.0, 0.0]
(2, 1, 1)	[192.79617837520652, 21.67534276772527, 0.8108334836484261]
(2, 1, 2)	[3.2754461715166485, 49.91624708536937, 0.0]
(2, 1, 3)	[25.487831672538796, 71.53630101240147, 260.6709883301907]

RESULTS: DO WE UNDERSTAND Q-TABLE?

```
(2, 2, 0)       | [0.0, 0.0, 0.0]
(2, 2, 1)       | [283.16683568799914, 87.91964295990556, 54.58849035032539]
(2, 2, 2)       | [70.44471036002687, 320.8118211950832, 136.9528844160416]
(2, 2, 3)       | [26.966628435003297, 294.33854188333146, 71.53737097227676]
(2, 3, 0)       | [0.0, 0.0, 0.0]
(2, 3, 1)       | [140.43315005753897, 209.50662239682515, 72.36459722085758]
(2, 3, 2)       | [182.0355949898773, 368.36090117464323, 142.2643422189268]
(2, 3, 3)       | [42.52074569317259, 285.7283503754035, 20.489132945699 19]
(3, 0, 0)       | [5.4379576766746505, -79.87224433089195, -73.57451164753292]
(3, 0, 1)       | [128.80397211011314, 0.0, -9.615251974570143]
(3, 0, 2)       | [0.0, 0.0, 0.0]
(3, 0, 3)       | [4.994406043921125, 4.133053275373953, 220.74413125015505]
(3, 1, 0)       | [0.0, 0.0, 0.0]
(3, 1, 1)       | [139.11203364554535, -25.18917009933058, 0.0]
(3, 1, 2)       | [0.0, 0.0, 0.0]
(3, 1, 3)       | [25.052324408838288, -3.0225648174040014, 260.8269640804169]
(3, 2, 0)       | [0.0, 0.0, 0.0]
(3, 2, 1)       | [208.73231477947195, -20.179334204373788, -3.3729855567160003]
(3, 2, 2)       | [0.0, 0.0, 0.0]
(3, 2, 3)       | [68.757836947404 56, 96.04778376421241, 324.2768778326491]
(3, 3, 0)       | [0.0, 0.0, 0.0]
(3, 3, 1)       | [390.18024984613885, 136.20158696582, 186.38894591677968]
(3, 3, 2)       | [302.10761222614343, 212.25947739248357, 150.01411672018912]
(3, 3, 3)       | [257.8468905958769, 198.72657463694443, 384.28452063352194]
```

CONCLUSIONS

- There is not really a difference between the number of bins chosen.
- Hardware works with selecting right Bin thresholds.
  - Could be done automatically by first calibrating sensors, then mapping to bin.
  - Too much time consuming.
- Future work:
  - Try to implement a Deep RL algorithm.

THANK YOU

QUESTIONS ???

VRIJE UNIVERSITEIT AMSTERDAM

CONTRIBUTION SHEET

- **Stergios:**
  - Created the structure and content for the presentation slides.
  - Experimented with the code for training the robot.
  - Experimented with different algorithms (CNNs).

- **Thijs:**
  - Improved on code.
  - Experimented with IR values.
  - Helped design and experiment with reward function.
  - Experimented with hardware.

- **Quirinus:**
  - Improved on code from T1: building new state, introducing separate functions.
  - Created code for storing data while running with a Training_Results class.
  - Created plots from saved data.

VRIJE UNIVERSITEIT AMSTERDAM

BUDDY-GROUP SUMMARY

- **Stergios:**
  - Better understanding of my team's capabilities.
  - Diverse perspectives improved my ability to see multiple sides of a situation before making decisions.
  - Learned more about the functionalities of the robot, including camera operation and object detection.

- **Thijs:**
  - Diverse Perspectives.
  - Learned practical techniques for minimizing the reality gap in simulations.
  - Learned how to use camera in RL algorithms.

- **Quirinus:**
  - Got better understanding of using Git.
  - Learned how to implement Training_Results class.
  - Learned to write better simple functions.
  - We started working together better with a plan and some task division.