December 8, 2024

Ethan Masiclat, Qian Ying Wong, Ramiyah Dougherty
Team 12

Cognitive Robotics

Neurorobots of the Deep - Deep Sea Skates

1 Introduction

Neurorobotics is a field that is inspired by living organisms' interactions with the actual world. Living organisms display intelligence and adaptive behavior. Many are able to learn and use the memory of what they learn to thrive in their environments (Krichmar and Hwu, 2022). These environment to organism interactions are a key area of interest and deep sea creatures are a good model to learn from.

Deep sea creatures are often not well studied because the depths of the ocean are difficult to traverse. However, from what has been studied on these creatures, they have complex behaviors that they utilize in order to survive the harsh environment that they live in. One such creature is the deep sea skate, a relative of sharks and rays.

The deep sea skate has a number of complex behaviors that have been documented. One of the most notable ones is how the deep sea skate lays its eggs in steam vents in order to hasten the hatching process (Keartes, 2018). This suggests that the skate is able to detect differences in temperature and learn about its surroundings in this way. Additionally, the deep sea skate uses its flat body to burrow on the seafloor in order to surprise its prey.

Taking note of these behaviors, our team decided to design a robot to mimic the deep sea skate. Using neurorobotic design principles, we wanted to test if our robot could learn to maintain itself with just these two simple behaviors. Additionally, we wanted to see if our robot behaved realistically while interacting with the environment based on just those two behaviors alone. The experimental design as well as design principles are all described in the methods.

2 Methods

2.1 Robot Design

The robot’s base design is kind of like a trike set-up. It has two motors to either side with wheels attached to them to navigate and turn. A huge sloped plow is attached in the front such that it makes contact with the floor in order to be able to cheaply create a “burrowing” action. A color sensor is attached to the bottom of the robot such that it faces the floor and clears the ground in order for the robot to detect sensory information about its environment. Finally, the

robot was covered in scratch paper in order to fully mimic the design of a deep sea skate as well as improve the mechanics of the burrowing.

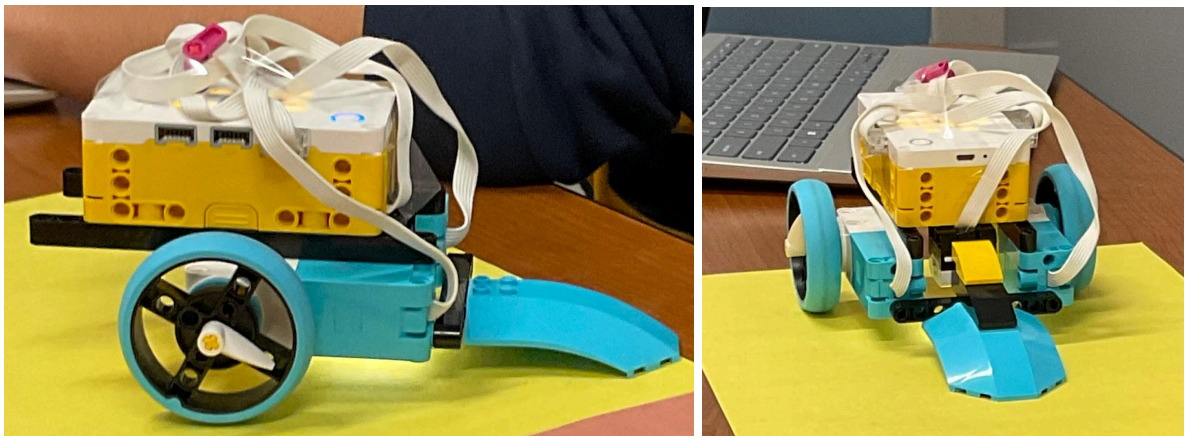


Figure 1. *The two pictures above depict the design of our robot before adding additional design work. In the photo on the left, you can see the light of the sensor showing through. Our sensor was placed at the bottom to ensure maximum efficiency when detecting color. In the photo on the right, you can see a clearer view of the additional piece in front of our robot that was used to create the "burrowing" effect our team was aiming for.*

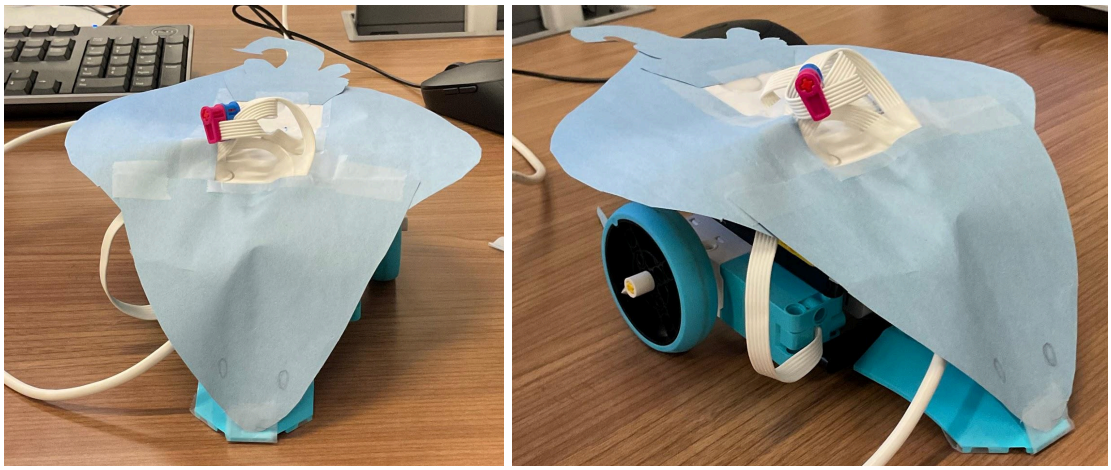


Figure 2. *These two photos showcase how our robot looked at added construction paper in order to make it appear like a deep sea skate.*

2.2 Code

The code is a series of functions which aims to follow neurorobotic design principles. The functions can be subdivided into movement, behavior and action selection/q-learning related code. The movement functions code how the robot moves when taking a certain action, such as moving forward, turning left or right. These functions allow the robot to execute basic motor commands in response to sensory input or decision-making processes. The robot also knows to back up, but the backing up movement is not a function by itself but is instead paired with the behavior functions.

The behavior functions code the behavior for when the robot achieves one of its goals (eat, lay eggs, or explore). We wrote an “eat” function, which makes the robot “eat” by stopping for a few seconds and then backing up once the feeding conditions are met (detection of green, and hunger values going over the threshold). The “layEggs” function has similar logic and behavior to the “eat” function, but has different conditions, which is the detection of white and having egg-laying values greater than the threshold. The “explore” function uses a for loop and checks for conditions for behaviors. The robot also checks for boundary colors (red, magenta, and blue), and will back up and turn around once it detects these colors to make sure it doesn’t go out of bounds. The “die” function stops all movements and behaviors, indicating that the neurorobot “died” if it becomes too hungry or has too high of an egg-laying value. This is checked through the variables hungerValue, eggsValue, hungerThreshold, and eggsThreshold.

```
# VARIABLES:
# Values influencing what set of actions the robot takes
hungerValue = 0
eggsValue = 0
hungerThreshold = 5 # was 20
eggsThreshold = 3 # was 10
```

The screenshot above shows our definition of the variables used in our code. “hungerValue” and “eggsValue” are initiated to 0, and will increment by 1 for every trial. They keep track of the hunger level and tendency to lay eggs of our deep sea skate neurorobot, and will be reset to 0 once the robot eats or lays eggs. It should be noted that the hungerValue has to be less than the eggsValue for the robot to lay eggs, so the robot would want to feed first before laying eggs. This logic is reflected in the get_reward() function in the action selection portion of our code. When either hungerValue or eggsValue exceeds their respective thresholds, the neurorobot will want to “eat” or “lay eggs”, or both. The threshold values also increment by 1 for every trial. We played around with the threshold values and did a few test runs before settling with 5 and 3.

Lastly, the action selection/q-learning functions utilizes a softmax function to select the robot's actions as well as sensory motor integration to calculate the reward that the robot should receive for completing a goal, moving closer to a goal, or going in the wrong direction. We initialized the parameters of the softmax function and Q learning function. The value for the beta (temperature) in the softmax function was tweaked to 0.3 because we wanted more exploration from the neurorobot, though this value can be changed depending on what type of behavior we want to observe (exploitation or exploration focused). There is also code for how the q-table is displayed. These functions are implemented into a while loop such that an action is selected based on the q-table, the reward (or punishment) is given to update q values and the

robot performs a behavior based on the outcome of that action. Bound checking behaviors (for red, magenta, and blue colors) are also implemented in the main while loop as a foolproof method to strictly contain the neurorobot in the environment.

Throughout the code, there are print statements to make the debugging processes easier. These sentences are printed out onto the console when the neurorobot runs, and they indicate what the robot is doing, and we're also able to keep track of its hunger level and tendency to lay eggs.

2.3 Neurorobotic Design Principles

When designing the robot a number of different neurorobotic design principles were used. They were implemented as follows:

Embodiment

The robot interacts with the environment through color detection. The environment motivates the robot's different behaviors.

Efficiency Through Cheap Design

The robot's design is inspired by the deep sea skate's triangular shape. The little edge at the front allows it to burrow into paper without explicitly having to program that behavior into the robot.

Sensory Motor Integration

The robot's color sensor input interacts with its motors to influence where it moves. The motor movement, in turn, influences what color is viewed by the robot.

Multitasking and Event Driven Processes

The robot bases its actions on the different events that occur in combination with a bit of randomization implemented. These events are based on the different colors that the robot can recognize.

Learning and Memory

The robot employs a q-learning algorithm in order to learn over time the best possible action for it to take while in a certain state. It stores q-values as memory for each state

Value Systems

The robot has a subsumption architecture that includes different values. The robot has values for hunger and for laying eggs as well as respective thresholds for each. The hierarchy is set up such that hunger is prioritized over anything else, followed by laying eggs and, lastly, exploration. If any value exceeds a threshold, the robot will perform appropriate behavior corresponding to the value. Otherwise, it will explore.

Exploration vs Exploitation

There is an inherent tradeoff between exploration and exploitation for the robot as it uses a q-learning algorithm. At first the robot chooses random actions, hence exploration. After learning for a bit, however, the robot is designed to eventually begin exploiting the environment based on the values it has learned.

Reward vs Punishment

There are rewards and punishments designed in the environment for the robot. When the robot is in a hunger state part of the environment is punishing for the robot while the other part is rewarding. The same applies for being in an egg laying state. There are reward and punishment tradeoffs depending on what state the robot is in.

Foraging vs Defending

The subsumption hierarchy prioritizes hunger over laying eggs. In the case of our robot, laying eggs is akin to defending. The tradeoff here is that the robot is unable to lay eggs if it is hungry.

2.4 Experimental Methods

In order to test the robot we designed an environment for it to explore. There are two sides to the environment: one that is considered hot temperature and one that is considered cold temperature. The robot's food source is on the cold temperature side while the robot's egg laying source is on the hot temperature side. The environment is surrounded by a boundary to keep the robot in bounds. To test what the robot does, we placed it in the center of the environment and let it explore on its own. We measured the amount of times it took each action, the progress of the robot's q-values, and the hunger and eggs laying values. Below is a diagram of its action selection possibilities.

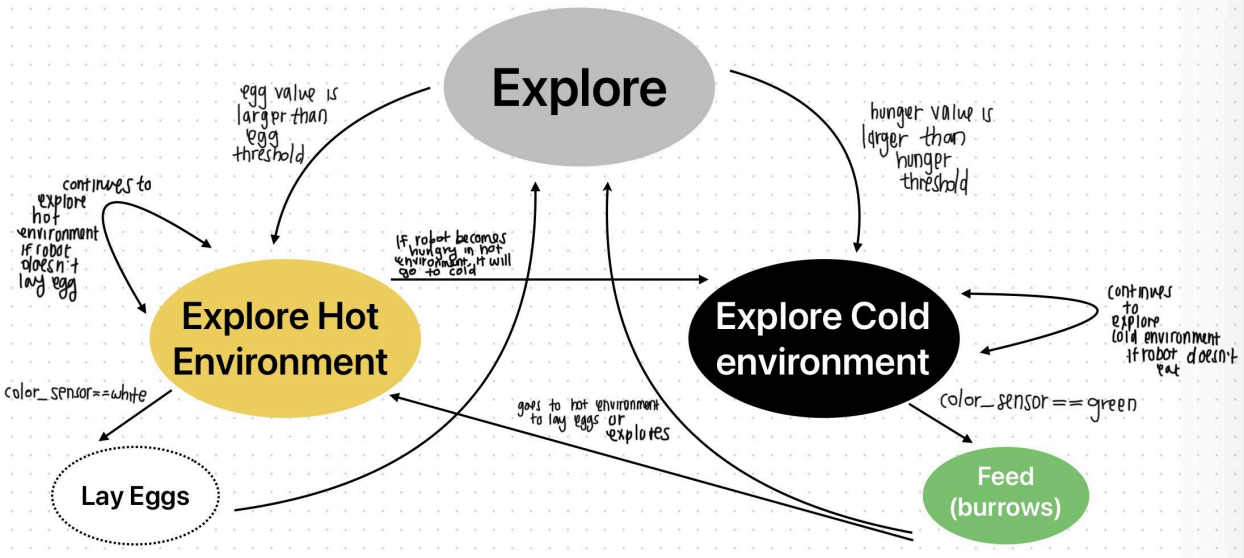


Figure 3. *Pictured above is our state event diagram. Our robot will first set out to explore its environment when the egg value or hungry value will choose its next steps: going into exploration of a hot or cold environment. From there, our robot will eat (and burrow), thus leading it to lay eggs after its hunger subsides. After all actions are completed ideally, the robot will go back to exploring.*

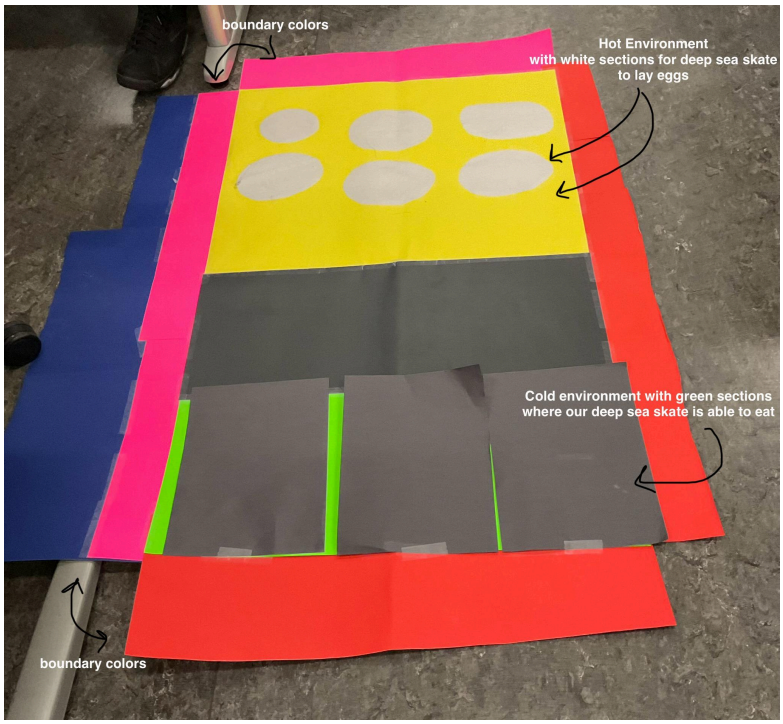


Figure 4*. In our environment, we used yellow for our hot environment with the addition of white in order to showcase our robot's ability to lay eggs. When our robot would sense the yellow environment, its egg value would increase, thus increasing its need to lay eggs. Then, when our robot would sense white, it would lay eggs. On the other hand, the black and green environment represented our robots ability to eat. Being in the hot environment would increase our robots hunger threshold, leading it to burrow, discover the green color, and eat when detecting the green. Lastly, we added boundary colors in order to keep our robot from leaving its environment.*

3 Results

After running the robot through 100 learning trials we obtained the following results. Looking purely at the robot's behaviors we observed that the robot ate more times than it laid eggs (see table 1). This tendency works in conjunction with the q values and will be further discussed later on.

3.1 Behavioral Measurements

	Eating	Laying Eggs
Number of Times	7	5

Table 1. *This table shows the amount of times the robot produced either an eating or laying eggs behavior. It can be seen that in the 100 trials that we ran, the robot ate 2 more times than it laid eggs. This could be due to the robot prioritizing eating over laying eggs*

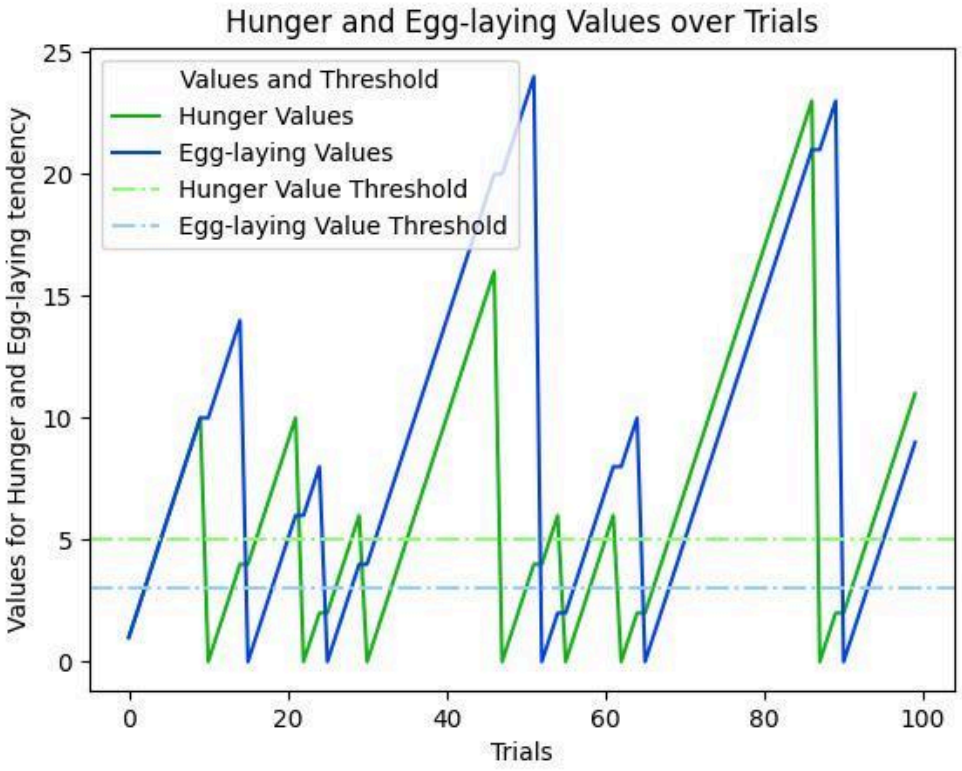


Figure 5. *Represented by green lines are hunger related values. Represented by blue lines are egg laying values. The robot is unable to perform either of its behaviors until the values exceed the threshold. This is seen by the peaks exceeding the horizontal lines before resetting. It is worth noting that after the 2nd time the robot laid eggs, it had to eat twice before it laid eggs again. This occurred for the 3rd egg laying as well.*

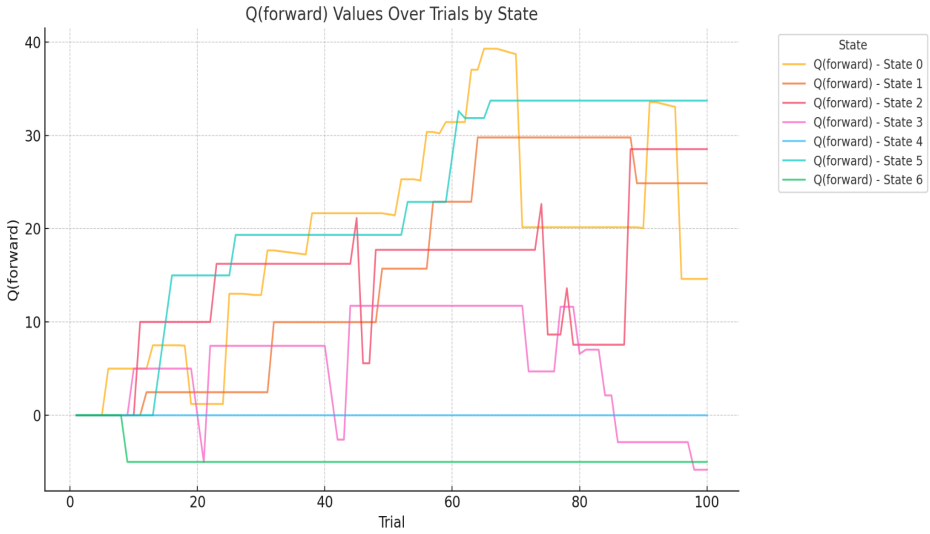
The behavior relationship between eating and laying eggs is best demonstrated by figure 5. With a hunger threshold of 5 and an egg-threshold of 3, the robot will explore when both values are below or equal to their respective thresholds. However, when either threshold is exceeded, the robot's behavior will change to match its respective needs. These needs are reflected by how each value reaches a peak and then resets once the need is satisfied. It is worth noting that egg laying only occurred after hunger occurred. Another thing to note is that if the robot waited too long to lay eggs, it would need to eat again before it could lay eggs. The robots choices that led to these behaviors will be further expanded upon by the q-values.

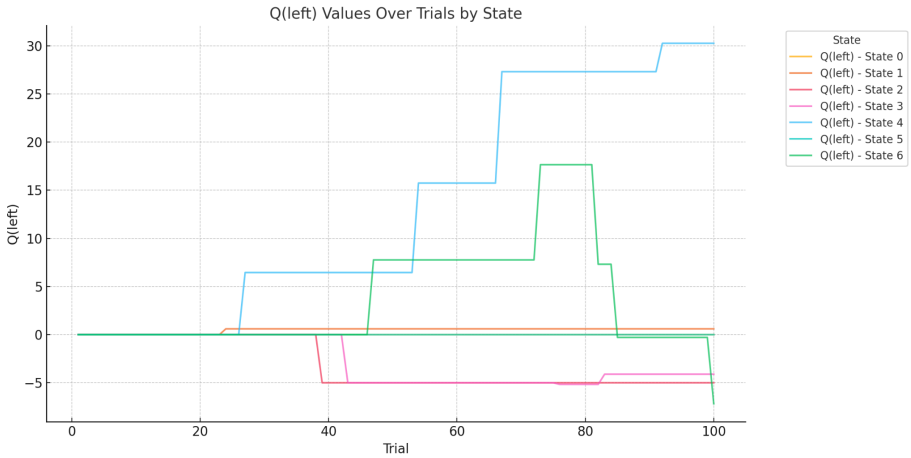
3.2 Q-table measurements

In order to understand the q-value chart we have to discuss the different states that we defined as seen in table 2. There was a need for seven different states due to the environment setup. Because there were 2 main zones, the hot and cold environment, the robot needed 4 states to represent its internal state for while it was in each of those environments. An additional 2 states were needed for when it ate or laid eggs. A final state was needed for all other conditions. Using this table as a reference we can now examine the graphs of the q-tables.

State Number	State Description
0	Exploration state
1	Hungry and in eating zone
2	Hungry and hot environment
3	Hungry and in cold environment
4	Need to lay eggs and in steam vent
5	Need to lay eggs and in hot environment
6	Need to lay eggs and in cold environment

Table 2. *The table here summarizes the different states that were created as a result of the experiment. The states are based on the robot's internal needs in conjunction with its location in the environment*





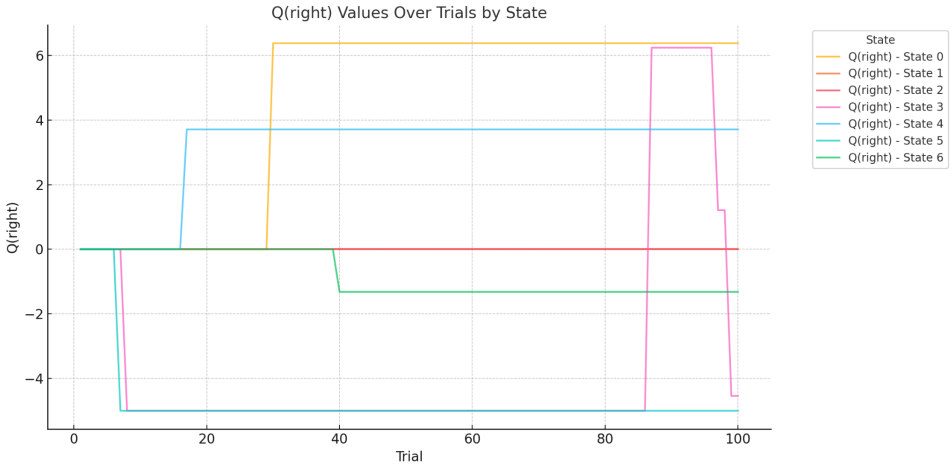


Figure 6. Picture above is the q-values over time for the forward action (top), left action (middle), and right action (bottom). Comparing between graphs, forward has the highest overall q-values suggesting that the robot's best choice would be to move forward and it would choose to move forward more often than not. There are some exceptions to this however as the highest q-values for state 3, 4 and 6 are not forward.

The q-values were graphed such that each possible action has its own graph and the q-values for each state are represented by 7 different lines. Comparing between the graphs, the forward action has the highest values, with a max q-value above 30. This indicates that the robot learned that forward was the preferable choice in most states. This is true for states 0, 2, and 5. Left was the preferred action for states 3 and 4, while right was the preferred action for state 6. It is also worth noting that the q-values for states 3 and 6 are negative for all actions. These trends are summarized in table 3 which has the q-table for the last trial.

State	Left	Right	Forward
0	0.00	6.38	14.61
1	0.60	0.00	24.86
2	-5.00	0.00	28.52
3	-4.11	-4.54	-5.84
4	30.26	3.71	0.00
5	0.00	-5.00	33.72
6	-7.18	-1.32	-5.00

Table 3. *Shown is the q-table for the last trial that was run before we terminated the program. The values demonstrate the final values that the robot learned and summarize what actions it was likely to take.*

4 Discussion

The results that we obtained from the robot's interaction with the environment influenced how we thought about the neurorobot design principles we implemented. Below we discuss how well the robot followed the design principles as well as our thoughts on the robot's overall performance.

4.1 Neurorobot Design Principles

Embodiment

The robot displayed a clear environment to body interaction. The stimuli from the environment influenced how the robot made decisions as the different locations informed the robot. This is displayed by how the robot had different preferences for the different possible actions. In turn, the robot physically interacted with the environment and influenced it as in the case with burrowing.

Efficiency through Cheap Design

The robot's burrowing mechanism worked well enough as it was able to lift the paper blocking its food source. When it worked correctly, the result looked fairly smooth and realistic. It definitely worked as the robot was able to perform its eating behavior.

Multitasking and Event Driven Processing

The robot was able to switch between its different states and choose the appropriate action based on these different states. This is shown by the fully updated q-table. The different states were based on how the robot needed to keep track of the different colors of the environment while also observing its own internal state.

Sensory-Motor Integration

The robot's sensors and motors worked together well for it to achieve its goals. Like predicted from the methods, the robot's sensors influenced its motor movement and the movement allowed the robot to get a new sensory input.

Learning and Memory

The q-learning algorithm was able to fill completely demonstrating the robot's learning. It was able to use this information as memory to influence its next action.

Value Systems

The robots value systems allowed it to choose certain behaviors over others. This was demonstrated well in figure 5 as hunger overrode the egg laying behavior.

Reward vs Punishment

There was a clear reward vs punishment tradeoff demonstrated by the robot as states 3 and 6 were updated in the q-table. These updates show that the robot had to consider the changing punishments and rewards related to the environment based on its internal state.

Foraging vs Defending

The robot displayed the foraging vs defending tradeoff as it needed to forage for food before it could lay eggs (defend). This trend is shown in figure 5. It is worth noting that the robot is at risk of death should it become hungry before it can lay its eggs.

Exploration vs Exploitation

The exploration vs exploitation tradeoff can be seen in figure 6. In the earlier trials, the robot has to explore greatly and the q-values fluctuate frequently. However, the robot should eventually reach a point where q-values do not fluctuate as much so that it can exploit the same action for the greatest amount of reward. This exploitation trend was not seen as clearly as later trials still fluctuated quite a bit; but some of the values remained pretty consistent over time. This may have been due to the low temperature value of 0.3.

4.2 Performance Evaluation

Overall, considering how the robot was able to fulfill many of the design principles quite well, the robot had pretty good performance comparing it to an actual organism. The slow movement (as seen in the corresponding video to this paper) mimics the deep sea skate realistically. The robot's behavior is consistent with an organism trying to achieve homeostasis and prevent itself from death. It was able to maintain itself over the course of the 100 trials we ran without dying. However there are a number of limitations to consider when thinking about the robot's performance.

4.3 Our Challenges and Limitations

While designing the robot's burrowing mechanism we found that it would get stuck on the loose tape and boundaries of the environment. This would interfere with the robots movement and possibly influence the best action that it could have taken. In the future, with greater resources, we would be able to build a cleaner environment for the robot to navigate or modify the burrowing mechanism to not cause interference.

Another limitation in our data comes from the actual design of the environment itself. Because the environment is a simple two sided area with one cold side and one hot side, the robot essentially only has to move back and forth between it. This could explain why the robot overwhelmingly chose forward as the best option as once it turned around, it could just drive

forward to the other side. These potential limitations signal the need to run more experiments as well as possibly modify the environment experimentally to test different setups.

4.4 Conclusion

From all of the data, we conclude that our robot is able to mimic the behavior of the deep sea skate quite well, but not perfectly. This indicates that the realistic behavior can be produced from simple design principles, but there are many factors that can influence how well the behaviors are performed. Future experiments should look at adjusting the robot's code and physical body could potentially better mimic behavior. Additionally, the robot's time scale should be adjusted to reflect the organism as hunger is a behavior on the order of days while laying eggs could potentially take weeks. These findings are rudimentary and preliminary, but open the doors for more avenues of research towards modeling realistic behavior.

5 Code

Code is uploaded to Canvas

Works Cited

Deep-Sea Skate • Animals of the deep • mbari. MBARI. (2024, June 27). https://www.mbari.org/animal/deep-sea-skate/

Natural World Facts. (n.d.). *Deep sea rays & skates*. https://www.naturalworldfacts.com/deep-sea-wonders-2/deep-sea-rays-%26-skates#google_vignette

Keartes, S. (2018). *These deep-sea skates use hydrothermal vents as egg incubators*. https://www.earthtouchnews.com/oceans/deep-ocean/these-deep-sea-skates-use-hydrothermal-vents-as-egg-incubators/