Task 2

Report

David Cohen - Mat. 0001085730

19 giugno 2024

Indice

1 Behavioral Hierarchy and Implementation

The control program consists of three primary behaviors: **obstacle avoidance**, **phototaxis**, and **wandering**. These behaviors are organized in a hierarchical stack, each with an assigned priority. The control logic ensures that the highest-priority behavior meeting its activation condition takes control of the robot, with higher-priority behaviors subsuming lower-priority ones as needed.

1.1 Behavior Management

The goal was to manage and abstract behaviors as much as possible, following the subsumption architecture. Each behavior is managed through a **priority queu**e and contains a body that reflects the callbacks of ARGoS. Additionally, each behavior has two crucial functions: `execute()` and `condition()`. The `condition()` function acts as a guard to determine whether the behavior should be executed. If the condition is met, the behavior **is queued and executed based on its priority**, subsuming any lower-priority behaviors. This separation resulted in cleaner, more concise code, making it easier to manage constants and variables and allowing for a focus on fine-tuning parameters.

1.2 Execution Flow

Initialization: each behavior is initialized, setting up necessary parameters.

Condition Evaluation: the system evaluates conditions for all behaviors.

Behavior Execution: the highest-priority behavior with a true condition is executed.

Reset and Cleanup: behaviors are reset or cleaned up as needed.

```
function senseAndAct()
    behaviors:for_each(function(behavior)
        if behavior.condition() then
            behavior.execute()
            return true
        end
    end)
end
```

Code 1: `senseAndAct()` function

2 Behavior Details

2.1 Obstacle Avoidance

– Priority: 3 (Highest)

This behavior is crucial for preventing collisions. It activates when **the proximity sensors detect an obstacle within a specified threshold**. Once activated, the robot adjusts its speed to turn away from the obstacle, ensuring safe navigation. For obstacle avoidance, I used a proximity threshold and an emergency speed factor to determine the robot's corrective actions.

Unlike the previous exercise, i removed the two extreme lateral sensors, as the robot now has exclusively forward movement, eliminating issues caused by the lateral sensors.

2.2 Phototaxis

– Priority: 2

Phototaxis directs the robot towards light sources. It activates when **the detected light intensity exceeds a minimum threshold**. The robot calculates the direction to the light source and adjusts its wheel speeds accordingly, guiding it towards the light. For phototaxis, I used a minimum light intensity threshold to determine when to activate the behavior and how to steer the robot towards the light source.

2.3 Wandering

– Priority: 1 (Lowest)

This default behavior ensures the robot moves randomly when **no higher-priority behavior is active**. It introduces random changes in direction and speed, enabling exploration. For wandering, I focused on introducing **random changes** in the robot's direction and speed to ensure efficient **exploration of the environment**. This was challenging, as the lack of memory complicates intelligent behavior. Initially, I used a step-based logic, but I eventually incorporated **probabilistic decisions**, making the behavior more versatile and allowing better management of the robot's exploration.

3 Performance Analysis

During analysis, I adjusted various parameters, finding that many were closely correlated, while others heavily depended on the type of arena. For example, large directional changes were counterproductive when the robot always started on the opposite side of the light, making straight-line movements preferable. As in previous labs, I realized that **decision-making and behavior, like the concept of correctness, are closely related to the developer's perspective, given the few design constraints**.

Introducing noise affected the precision of sensor readings and actuator responses, leading to increased variability in the robot's path.

4 Conclusion

The subsumption architecture proved effective for implementing a robust robot control system in ARGoS. The hierarchical structuring of behaviors ensured that critical tasks were prioritized, resulting in reliable navigation. The figure 2 show the results covered in a simulation of *3000 steps* over *500 simulations*.

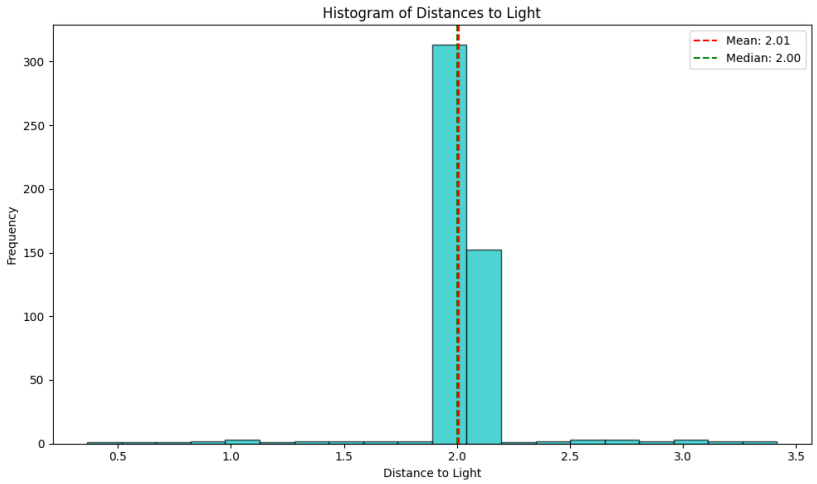

Figura 2: Histogram of Distances to Light