Task 1

Report

David Cohen - Mat. 0001085730

17 giugno 2024

Indice

1 Methodology

Initially, I analyzed the constraints to understand the design possibilities. I developed and tested each task individually before combining them into a single system.

1.1 Phototaxis

The phototaxis task guides the robot towards the most intense light source using the 24 light sensors. The function `utilities.getMaxSensorReading('light')` detects the index and intensity of the highest reading. The rotation angle and wheel speeds are calculated based on the distance to the light and the footbot's maximum speed, as implemented in the `phototaxyTask` function.

1.1.1 Calculating Wheel Speeds

Helped by ChatGPT, I was able to find a suitable formula for adjusting the wheel speeds based on the angular difference

```
baseSpeed + sign * difference * baseSpeed
```

For example, with `baseSpeed` of 10 and `difference` of 0.5:

- right wheel (`sign` = 1): 15;
- left wheel (`sign` = -1): 5.

This allows the robot to orient towards the light by appropriately adjusting wheel speeds.

1.2 Obstacle Avoidance

To avoid obstacles, I used the 12 front sensors and developed an 'emergency' behavior. The `obstacleAvoidanceTask` function includes two behaviors:

- reducing speed and steering when the obstacle is within a threshold.
- rotating on the spot if the obstacle is within 0.5 times the threshold.

This approach made obstacle avoidance smoother and more robust.

1.3 Random Walk

The `randomWalkTask` function assigns random speeds to the footbot's wheels for a predetermined number of steps, facilitating environmental exploration.

1.4 Task Combination

In the `step()` function, I prioritized obstacle avoidance, followed by phototaxis, and finally random walk. This structure ensures effective priority management.

2 Results and Discussion

The composite behavior effectively combines phototaxis, obstacle avoidance, and random walk. The logic was encapsulated in individual tasks to make the `step()` function streamlined and self-explanatory.

```
function step()
if not obstacleAvoidanceTask() and not phototaxyTask() then
randomWalkTask()
end
end
```

Code 1: `step()` function

The implementation effectively uses sensor data and adaptive wheel speed control to navigate the environment. Adding noise (0.1-0.5) showed **significant behavior variations**, demonstrating the system's adaptability. Using multiple robots (up to 5) showed they all reached the light center, **aggregating** with occasional detachment and reaggregation.

3 Conclusion

The developed system effectively combines the required tasks, showing good adaptability and robustness. I aimed to make the `step()` function **streamlined** and **self-explanatory** by encapsulating logic within individual tasks.

- **Behavior combination strategy**: Developing basic behaviors separately and then combining them proved more effective. The "divide and conquer" strategy simplified code development and maintenance, allowing for a modular and scalable implementation.
- **Memory utilization**: Currently, the controller does not use memory, but integrating memory could significantly enhance efficiency and robustness by tracking obstacles, storing light source positions, planning optimized paths, improving multi-robot coordination, and dynamically adapting to the environment.

- **Alternative approaches**: To further optimize performance, the code should remain simple and readable. Implementing artificial intelligence models could simplify task combination and enhance system efficiency.