Duckiebot Exercise 1 Report

Created By: Basia Ofovwe
Student ID: 1667223

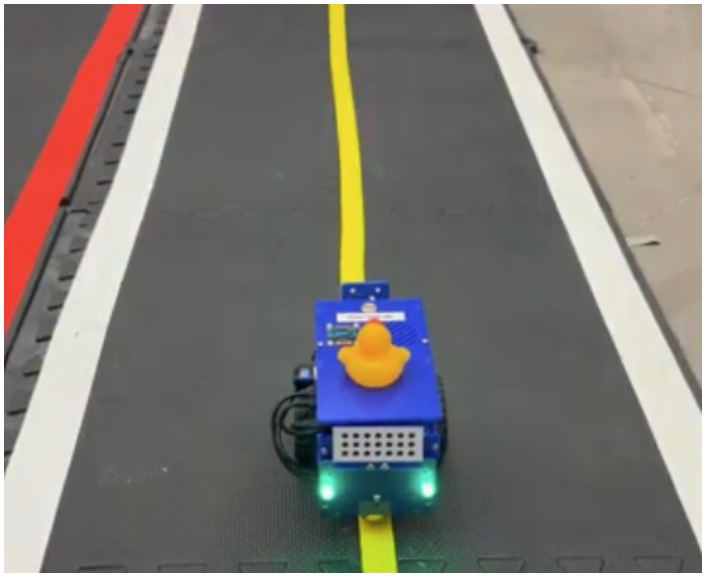
1. Introduction

The Duckiebot Lab aims to explore autonomous robotics and system integration using the Duckiebot platform. The primary objective of this exercise is to implement and test functionalities such as driving in a straight line, lane following, and utilizing calibration techniques to optimize performance. Additionally, this lab introduces the basics of Duckiebot assembly, networking, and Docker usage. Through this lab, I begin to understand the interplay between software and hardware components and learn troubleshooting techniques for autonomous systems.

2. Deliverables

2.1 Driving in a Straight Line

Video:



Description:
To ensure the Duckiebot could drive in a straight line for 2 meters, we performed wheel calibration focusing on the `trim` parameter. The `trim` parameter compensates for minor discrepancies between the left and right motors. By adjusting this parameter, we balanced the motor outputs to prevent the bot from veering[1]. The calibration process involved:

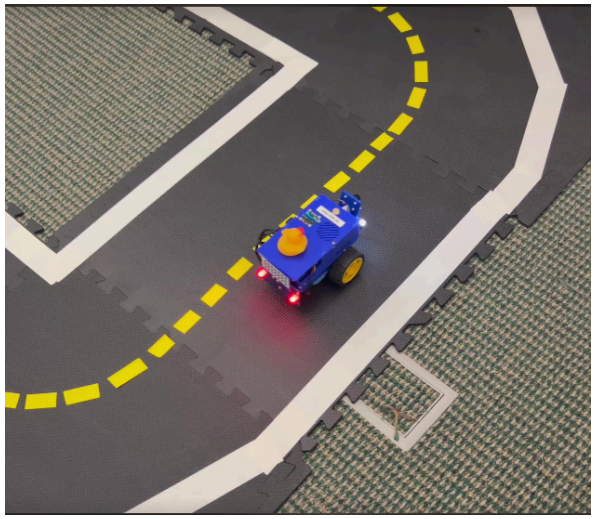
1. **Preparation**:

    - Placed a straight tape line on the floor mat as a reference (already prepared by the TAs).
    - Positioned the Duckiebot at one end, centered on the line.
2. **Testing**:
    - Commanded the Duckiebot to move forward for approximately 2 meters using the duckiebot keyboard_control command.
    - Observed the direction and extent of any drift from the reference line.
3. **Adjustment**:
    - If the Duckiebot drifted left, we decreased the `trim` value; if it drifted right, we increased it.
    - Repeated the test and adjustment until the drift was less than 10 centimeters over 2 meters, which is acceptable for Duckiebot operations.

This methodical approach ensured the Duckiebot maintained a straight path during operation. Our trim value ended up being 0.01.

2.2 Lane Following Demo

Video:



Description:
The lane-following demo showcased the Duckiebot's autonomous navigation within a predefined lane using its onboard camera and image processing algorithms[1]. The implementation steps were:

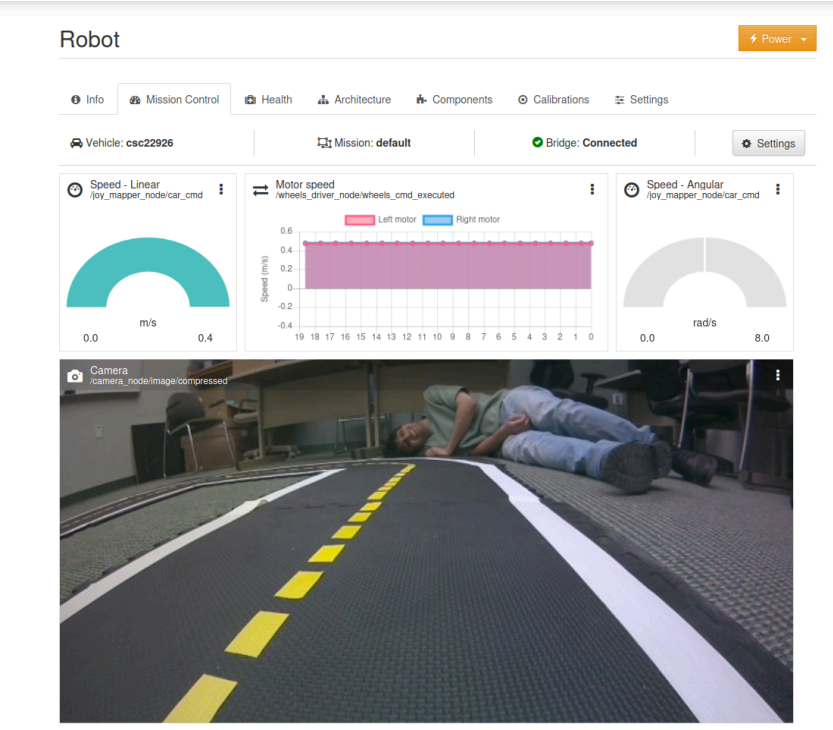
1. **Prerequisites**:
    - Completed wheel and camera calibration to ensure accurate movement and image capture.
    - Set up a Duckietown loop with clear lane markings under adequate lighting conditions (already setup by TAs).

2. **Demo Execution**:
    - Launched the lane-following demo using the keyboard_control command. Initiated autonomous driving by pressing the appropriate key in the keyboard control interface (`a` to start, `s` to stop).
3. **Visualization and Debugging** (Optional):
    - Used `rqt_image_view` to visualize detected line segments and ensure the image processing pipeline accurately identified lane markings.

This process enabled the Duckiebot to navigate autonomously by detecting and following lane lines.

2.3 Dashboard Outputs

Screen Capture:



Description:
The Duckiebot dashboard provided real-time insights into the robot's operations, including camera feeds and motor commands[1]. To utilize the dashboard:

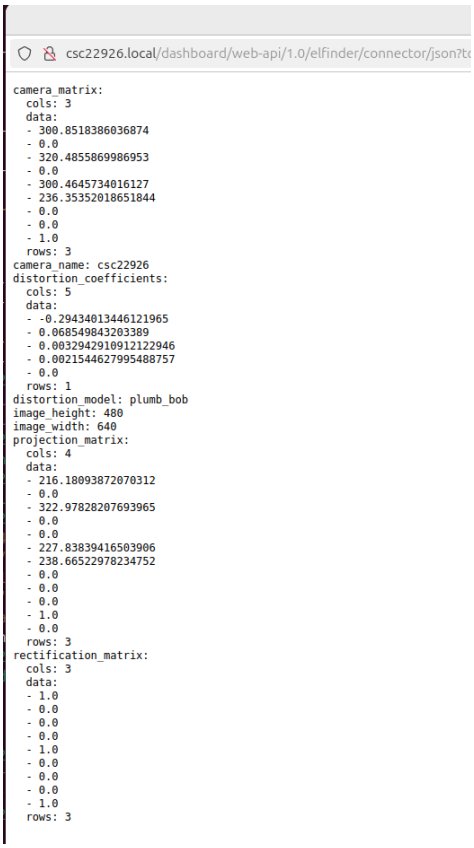
1. **Access**:
    - Connected to the Duckiebot's dashboard via a web browser using the robot's hostname.
2. **Monitoring**:
    - Observed the live camera feed to verify the robot's visual input.

- Monitored motor command outputs to assess the robot's movement responses.

3. **Debugging**:
   - Identified discrepancies between expected and actual behavior by analyzing dashboard data.
   - Made necessary adjustments to calibration parameters or control algorithms based on observations.

2.4 Calibration Files

Intrinsic Calibration (calibration_intrinsics.yaml):
Screen Capture:



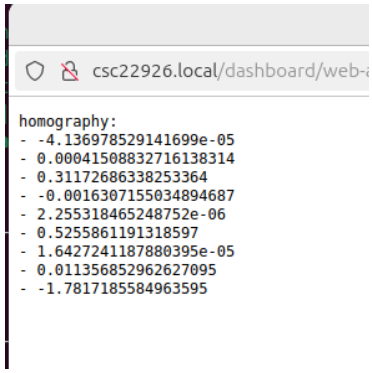
Explanation:

Intrinsic calibration involves determining the camera's internal parameters, such as focal length and optical center, to correct image distortions[1]. The process included:

1. **Setup**:
   - Displayed a checkerboard pattern in the Duckiebot's field of view.
   - Captured multiple images from different angles and positions.
2. **Calibration**:

- Used the captured images to compute the camera's intrinsic parameters.
- Saved the parameters in the `calibration_intrinsics.yaml` file.

Extrinsic Calibration (calibration_extrinsics.yaml):
Screen Capture:

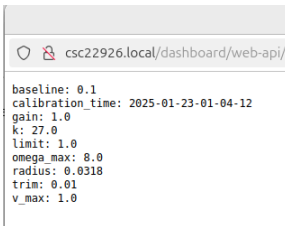


Explanation:

Extrinsic calibration defines the spatial relationship between the camera and the Duckiebot's chassis. Accurate extrinsic calibration ensures that the camera's perspective aligns with the robot's physical orientation, which is important for tasks like obstacle avoidance and lane tracking [1].

1. **Setup**:
    - Positioned the Duckiebot in a known reference frame within the Duckietown environment.
2. **Calibration**:
    - Aligned the camera's coordinate frame with the Duckiebot's physical frame using a predefined calibration sequence.
    - Saved the parameters in the `calibration_extrinsics.yaml` file.

Kinematics Calibration (kinematics_calibration.yaml):
Screen Capture:



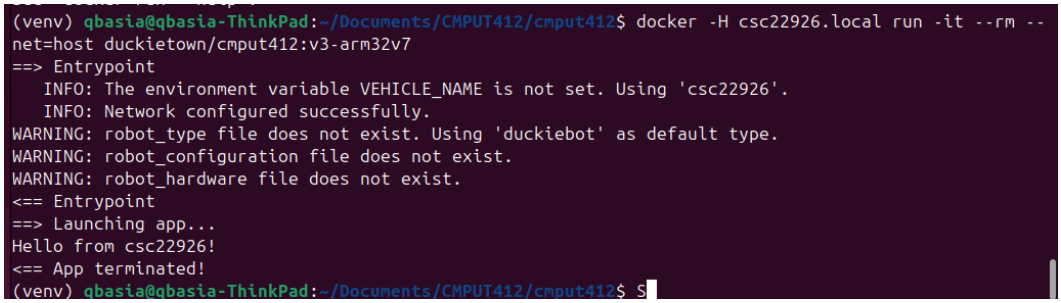
Explanation:

Kinematics calibration determines the Duckiebot's wheelbase (distance between the wheels) and wheel radius, ensuring accurate motion control. This is achieved through the following steps:

1. **Setup**:
    - Run the kinematics calibration tool within the Docker container, which initiates predefined movement commands for the Duckiebot.
    - Ensure the Duckiebot is on a flat, smooth surface free of obstructions (including footprints/debris).
2. **Parameter Adjustment**:
    - Here the only test that was done to calibrate this was the driving in a straight.
    - The calibration tool calculates corrections to the wheel radius and wheelbase parameters, optimizing them based on observed behavior.
    - These adjustments are saved in the `kinematics.yaml` file.

2.5 Duckiebot Greeting

Screen Capture:



Description:
The greeting message "Hello from csc22926" was generated using a pre-configured Python script executed within a Docker container. This script tested the Duckiebot's software setup and confirmed successful communication between its hardware components and the control software. The execution of this command also served as a basic diagnostic tool to verify the readiness of the Duckiebot for further tasks [1].

3. Reflection and Key Takeaways

3.1 Calibration and Motors

Calibrating the camera and motors are crucial for the Duckiebot's accurate performance. Motor calibration ensures consistent speed and direction, while camera calibration eliminates image distortions that could affect lane detection. These steps were essential for aligning hardware behavior with software expectations.

3.2 Docker Utility

Docker is instrumental in managing dependencies and ensuring reproducibility of the software environment. Its containerization capabilities simplified the process of setting up the Duckiebot's software on different machines and eliminated compatibility issues.

3.3 Troubleshooting and Observations

A free and declustered network is needed for optimum and fast connection which aids in the performance as well as calibration steps. It took multiple tries to get the camera calibration up and running.

When the Duckiebot veered right while driving straight, the issue was traced to an imbalance in motor output. Adjusting the motor trim parameter in the Docker container corrected this drift by a lot. During the lane-following demo, the Duckiebot struggled in areas with uneven lighting, which led to inconsistent lane detection. Improving the image preprocessing pipeline and fine-tuning the control algorithm resolved this challenge.

In addition, when doing the lane following, it is crucial to have the orientation of the bot correct with on respective sides of the road (in America, we drive on the left side of the road) as the bot has been programmed that way. We encountered issues with the bot leaving the lanes when the orientation was reversed as the bot was trying to have the white line on its right side.

3.4 Key Takeaway

The lab emphasized the importance of calibration, iterative testing, and understanding the interplay between hardware and software in robotics systems. It also highlighted the value of tools like Docker in managing complex software environments. Additionally, setting up the course website and GitHub repository emphasized the importance of documentation and collaboration in robotics projects.

4. Conclusion

The Duckiebot Lab provided a comprehensive introduction to autonomous robotics through hands-on experience with assembly, calibration, and operational testing. Each deliverable reinforced critical concepts such as motor calibration, image processing, and control algorithms. By leveraging tools like Docker and the Duckiebot dashboard, we streamlined the development process and resolved challenges such as drift and inconsistent lane detection. This lab highlighted the value of iterative testing and calibration in robotics, and it underscored the importance of robust documentation and collaboration for effective project execution. Ultimately, the lab provided valuable insights into the complexities of autonomous system design and prepared us for future explorations in robotics and automation [3].

5. References

[1] Duckietown Documentation Team, "Duckiebot Operations Manual," [Online]. Available: https://docs.duckietown.com/daffy/opmanual-duckiebot/operations/make_it_move/index.html. [Accessed: Jan. 15, 2025].

[2] OpenAI ChatGPT, "Assistance with Duckiebot Lab Report," ChatGPT-based guidance for robotics documentation, [Online]. Available: https://chat.openai.com/. [Accessed: Jan. 27, 2025].