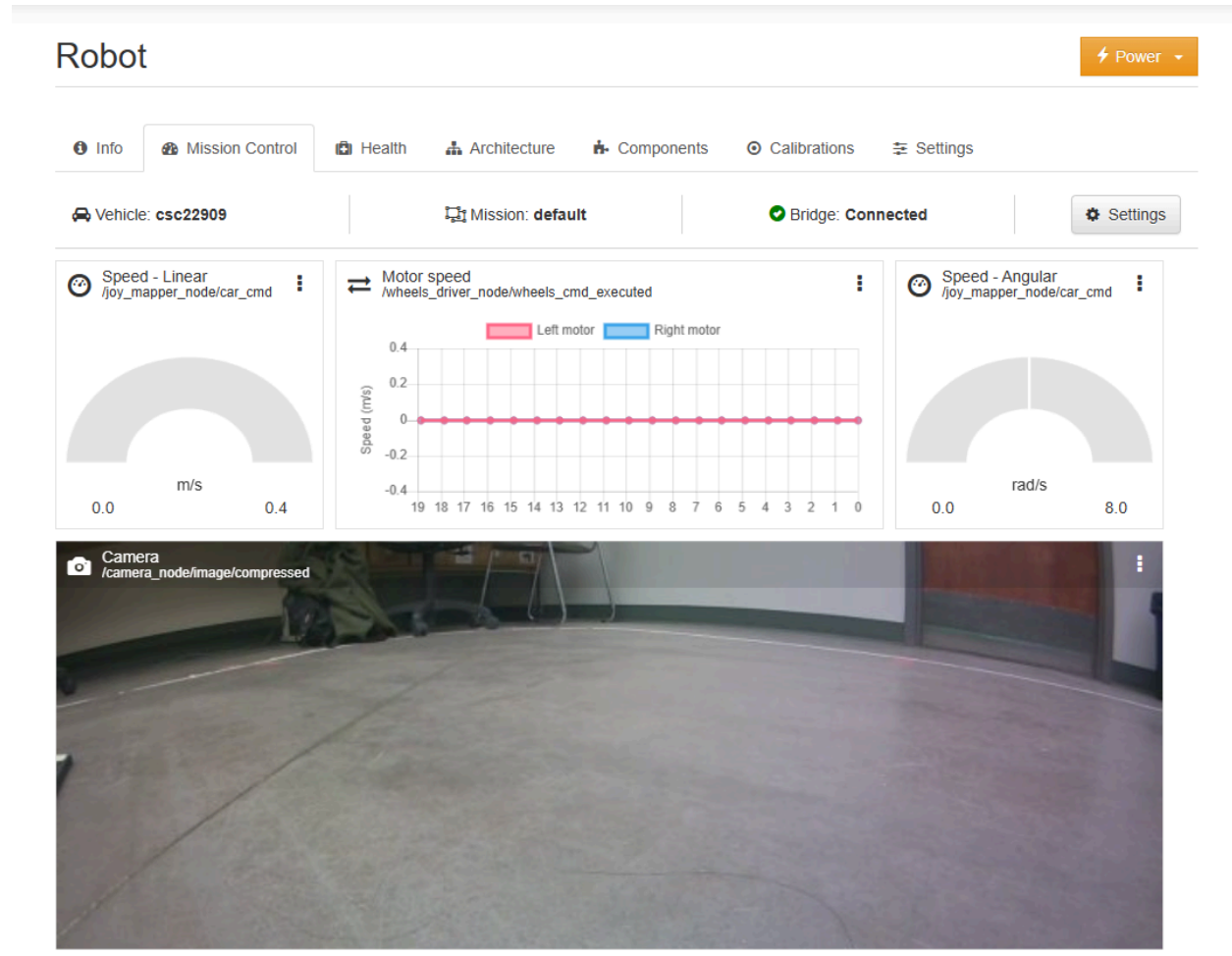


CMPUT 412/503 - Exercise 1 Report

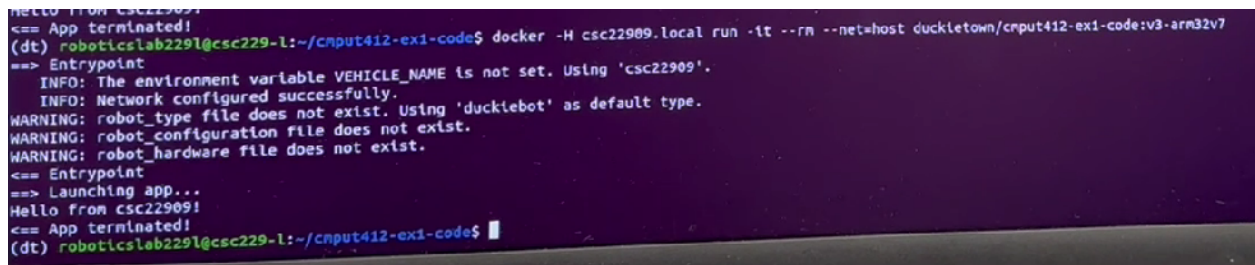
Abdullah Khadeli

1. Camera and Motor signals



The above screen capture of the dashboard shows the real-time camera and motor signals. From the mission control tab, we can verify that our machine receives motor signals to determine the motor, linear/angular speeds. Additionally, we can view the live camera feed from the camera mounted on the frontside of the Duckiebot.

2. Hello from csc22909!

A terminal window with a dark purple background. The prompt is `(dt) roboticslab2291@csc229-l:~/cnp412-ex1-code$`. The command entered is `docker -H csc22909.local run -it --rm --net=host duckietown/cnp412-ex1-code:v3-arm32v7`. The output shows the container's entrypoint, environment variable warnings, and the message "Hello from csc22909!".

```
HELLO FROM csc22909!  
<== App terminated!  
(dt) roboticslab2291@csc229-l:~/cnp412-ex1-code$ docker -H csc22909.local run -it --rm --net=host duckietown/cnp412-ex1-code:v3-arm32v7  
=> Entrypoint  
INFO: The environment variable VEHICLE_NAME is not set. Using 'csc22909'.  
INFO: Network configured successfully.  
WARNING: robot_type file does not exist. Using 'duckiebot' as default type.  
WARNING: robot_configuration file does not exist.  
WARNING: robot_hardware file does not exist.  
<== Entrypoint  
=> Launching app...  
Hello from csc22909!  
<== App terminated!  
(dt) roboticslab2291@csc229-l:~/cnp412-ex1-code$
```

This screenshot of the terminal shows the output of our Duckiebot saying “Hello from csc22909!”. Using the example codebase from the Duckiebot documentation, we were able to containerize our python code, which would print the designated name of the Duckiebot. Since Docker engine runs on the Duckiebot, we can run our container by specifying the target machine to run the container on (csc22909.local in this case). We specify the flags “-it” to start an interactive shell in the container, allowing us to execute a shell script and therefore running our code. We specify the network as “host” which uses the host network driver, allowing us to access the network accessed by the host machine, and therefore csc22909.local which is also connected to the same network. Lastly, the “--rm” flag will remove the container after the task is complete.

3. Calibration

Calibration of the camera is required to be able to correctly undistort frames from camera, have the correct parameters for localization and mapping, completing homography transformations between the surface and camera planes, mapping 3D world coordinates to 2D image coordinates, and path planning.

We calibrate the motors and wheels to account for the imbalances in the output and uniformity of the motors and wheels, allowing for consistent velocity, precision in localization and mapping and ensuring that the Duckiebot moves accurately according to the desired path.

3.1. Camera intrinsics

```
duckie@csc22909:~$ cat /data/config/calibrations/camera_intrinsic/csc22909.yaml
camera_matrix:
  cols: 3
  data:
    - 310.0149584021843
    - 0.0
    - 307.7177249518777
    - 0.0
    - 309.29643750324607
    - 229.191787718834
    - 0.0
    - 0.0
    - 1.0
  rows: 3
camera_name: csc22909
distortion_coefficients:
  cols: 5
  data:
    - -0.2683225140828933
    - 0.049595473114203516
    - 0.0003617920649662741
    - 0.006030049583437601
    - 0.0
  rows: 1
distortion_model: plumb_bob
image_height: 480
image_width: 640
projection_matrix:
  cols: 4
  data:
    - 213.63165283203125
    - 0.0
    - 313.718391657334
    - 0.0
    - 0.0
    - 242.34793090820312
    - 221.6794590338959
    - 0.0
    - 0.0
    - 0.0
    - 1.0
    - 0.0
  rows: 3
rectification_matrix:
  cols: 3
  data:
    - 1.0
    - 0.0
    - 0.0
    - 0.0
    - 1.0
    - 0.0
    - 0.0
    - 0.0
    - 0.0
    - 1.0
  rows: 3
```

This screenshot shows the camera, projection and rectification matrices, distortion coefficients, and image size, calculated from calibrating the camera using a sheet printed with a checkerboard pattern (7x5). Calibration was done by moving the Duckiebot in the horizontal and vertical axes, moving closer and farther for scale, and tilting for skew with the checkerboard in view. The camera matrix gives us the principal point and the focal lengths, which we use to build the projection matrix and thus helping us map 3D points to the 2D image space. The distortion coefficients along with the camera matrix, are used for correcting lens distortion in the camera frames, fixing straight lines that appear curved in each frame.

Explanation

3.2. Camera extrinsic matrix

```
duckie@csc22909:~$ cat /data/config/calibrations/camera_extrinsic/csc22909.yaml
homography:
- -0.00012804819298441165
- 0.0004481376276608629
- 0.34525185225242017
- -0.0016746663699615842
- -1.5998279042688944e-05
- 0.5274229627718546
- -0.00023336716207724326
- 0.01183509892677861
- -1.8162755258651098
```

This screenshot shows the homography matrix calculated from extrinsic calibration. We placed the checkerboard pattern (7x5) on a flat surface, with the Duckiebot viewing the pattern from above (bird's eye view). By running the extrinsic calibration pipeline, we got the homography matrix. This matrix describes the transformation between the two planes (camera plane and surface plane). We can use this matrix to transform coordinates from the surface plane to the camera plane and vice versa allowing us to localize and path plan. These extrinsics are needed for the lane following demonstration.

4. Kinematics

```
duckie@csc22909:~$ cat /data/config/calibrations/kinematics/csc22909.yaml
baseline: 0.1
calibration_time: 2025-01-23-00-15-33
gain: 2.0
k: 27.0
limit: 1.0
omega_max: 8.0
radius: 0.0318
trim: -0.055
v_max: 1.0
```

This screenshot shows the kinematics configuration of our robot. These parameters are used in the relevant calculations for a differential drive robot (velocity, angular velocity, dead reckoning). The only parameters we changed was the trim, which is the correction factor for imbalances in the wheels, and the gain, which is a scaling of the motor output. The trim parameter is negative as we observed the Duckiebot drifting to the left. Other parameters did not need to be modified since they are set based on the design of the Duckiebot.

5. Straight driving

[Link to video](#)

In this video, we demonstrate our Duckiebot driving straight for 2 meters. We had to re-run multiple demonstrations as we had to find a good value for our trim parameter. Additionally, towards the end of the demonstration, we see the Duckiebot turn slightly to the right. This is

likely due to the imbalances of the surface, and the lack of suspension on the wheel base, causing the Duckiebot to turn according to the surface.

6. Lane following

[Link to video](#)

In this video, we demonstrate our Duckiebot completing the lane-following demonstration given by the manufacturer. The path is a road connected in a loop, with two sets of edge road markings in white and a set of center road markings in yellow. The Duckiebot completed the demonstration well, given the imbalances on the road surface. We still saw that our Duckiebot slightly turned to the left, and would have to correct itself periodically, which leads us to believe that we still need to tweak the trim parameter to compensate for the drifting. The Duckiebot is able to conduct this demonstration, by using the calculated camera calibrations we did earlier. The homography matrix is used in the demonstration to transform frames of the surface to the camera plane, allowing for simultaneous localization and mapping, and therefore path planning around the lane. The Duckiebot would use the right outside edge lane markers in white and the center yellow lane markers to plan a path to traverse. By using PID, the Duckiebot can execute the correct motor outputs, along with the corrections we can see in the video to follow the planned path.

Takeaways:

Working on setting up the Duckiebot, and calibrating the different parts has got me to re-learn many concepts in computer vision for the camera calibration tasks, and recall my previous experience in working with mobile robotics in a previous class.

References

https://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf
http://docs.opencv.org/4.x/d9/dab/tutorial_homography.html
<https://docs.duckietown.com/daffy/opmanual-duckiebot/intro.html>