CMPUT 412 – Lab Exercise #1

Antonio Lech Martin-Ozimek

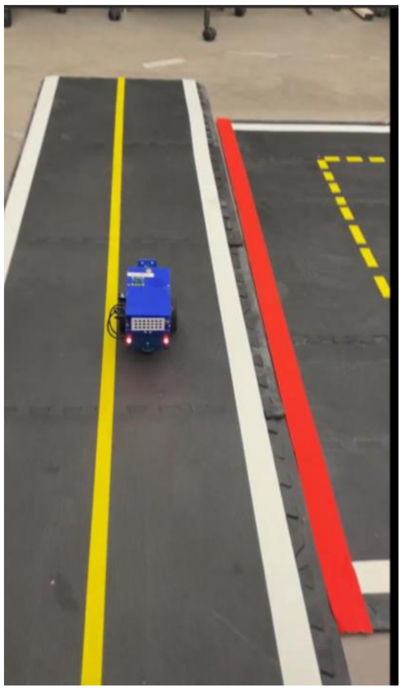
1658820

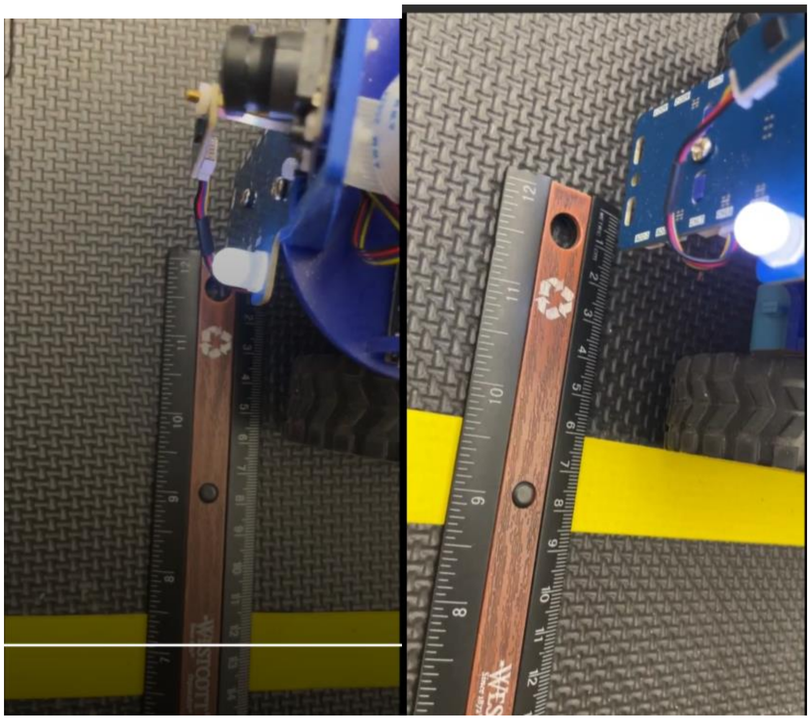
Section H02

“Calibrating the camera and motors on a Duckiebot ensures accurate navigation and task performance, such as lane following. Camera calibration aligns the robot's visual data with its internal reference frame, while motor calibration ensures proper response to control signals, preventing issues like drifting. Docker is useful because it allows for containerization of the Duckiebot’s software, making it easier to deploy and run the code consistently across different machines without worrying about dependency conflicts. If the robot veers off course, like drifting right, it may be due to motor discrepancies or uneven wheel calibration. To adjust for this, the robot could use its camera and sensors to detect drift and correct its trajectory in real-time. We also implement the trim factor which I assumed reduced or added more power to the right motor than the left one to compensate for the drift.

During the lane-following demo, my Duckiebot struggled with sharp corners, likely due to limited camera resolution and the mat's inconsistent traction, which could disrupt calibration and cause control issues. The robot likely follows the lane by using vision-based algorithms that detect the lane and adjust the steering accordingly. This lab highlighted the importance of calibration and testing, showing that even small errors can lead to significant performance issues. My key takeaway is how sensor feedback, calibration, and tools like Docker are essential to creating reliable autonomous systems." [ChatGPT]

● A video of your Duckiebot driving in a straight line for a distance of 2 meters



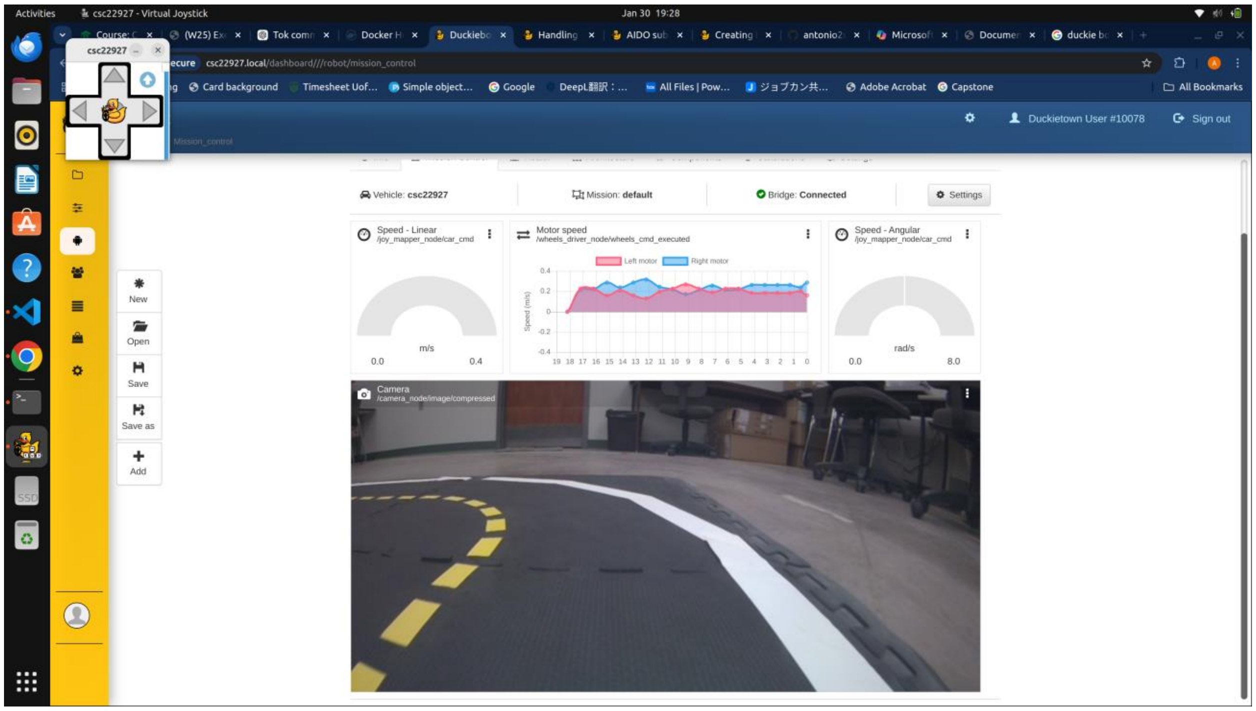


This task was the hardest in the lab for me, because the setup of the experiment changed the results. I found that placing the Duckiebot directly on the yellow line would sometimes cause slippage of the back end of the Duckiebot and it would veer off in one direction. I had to test multiple different setups to find one where the Duckiebot would not be affected by the condition of the mat or slip on the tape. The most consistent results occurred when I lined up the left wheel of the Duckiebot with the side of the yellow tape. This was something I could repeat and gave me an accurate idea of the effect of the trim on the path of the Duckiebot.This led to me fine tuning the car enough to stay within 6cm of the line. I measured this by measuring from the center of the yellow tape to the center of the camera, which lies about in the center of the two wheels. At the start that distance was about 8cm and at the end it had grown to about 14 cm. Since the trim that had gotten that result had been consistently showing good results, I decided to stop calibrating there.

- A video of your Duckiebot running the lane following demo

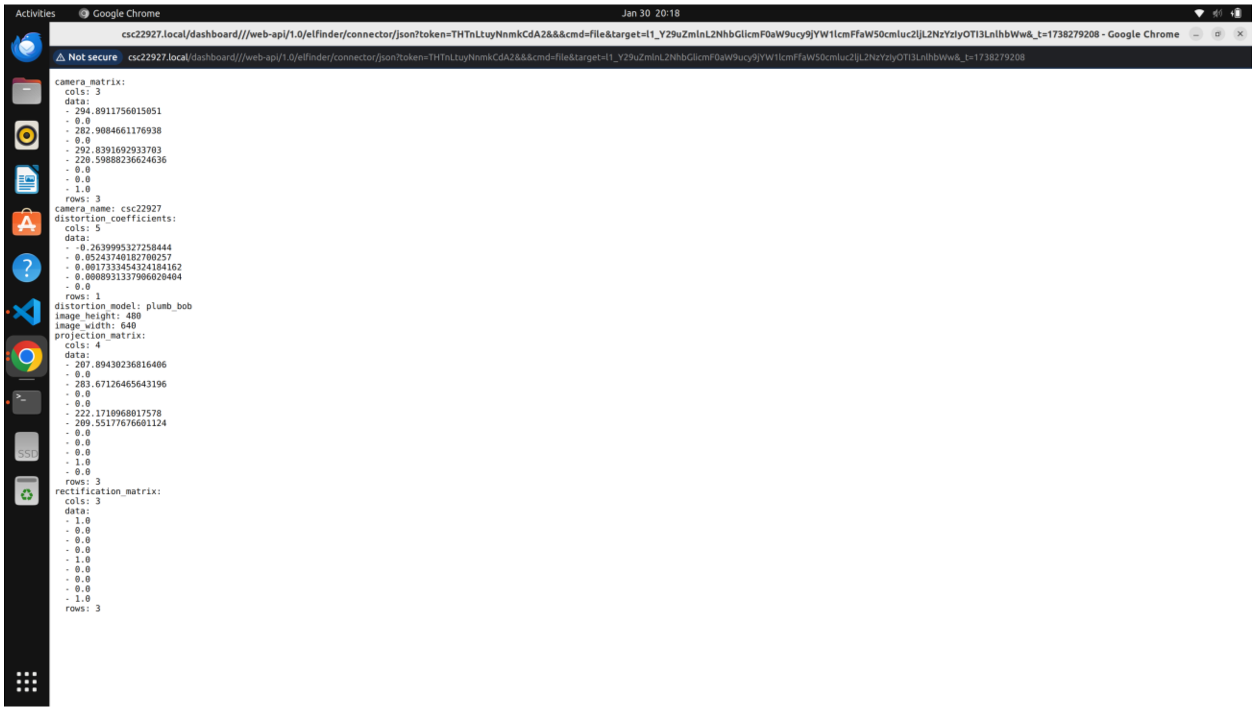
The lane following demo was quite simple to run after all the calibration was done. I did notice that my Duckiebot had a lot of trouble on the corners. It would turn shallowly and overshoot the white line. It would then course correct and follow just on top of it maybe a bit to the inside of the line. Even though the calibration was done, I doubt that the sensors themselves were completely free from error. I intuit that the overshoot is related to an accumulation of errors between the wheels and the camera. Furthermore, the surface of the mats is not consistent and the softness of it may affect not only the calibration but also the live demo.

- A screen capture of the camera output and motor signals as seen from the Dashboard



I took this shot during the line following demo because I couldn't drive the car and screen shot at the same time. This task was very easy to implement but provided a lot of insight and fun to the lab. I found that being able to see the motor outputs for each wheel and being able to see through the camera made it feel like I was playing a video game. Aside from that, I noticed that there were spikes in the graphs when I was turning the car by hand, where both engines would suddenly turn in the same direction, and the car would slow down. Having the dashboard gave me an idea of the odd behavior the car has when you turn it for too long.

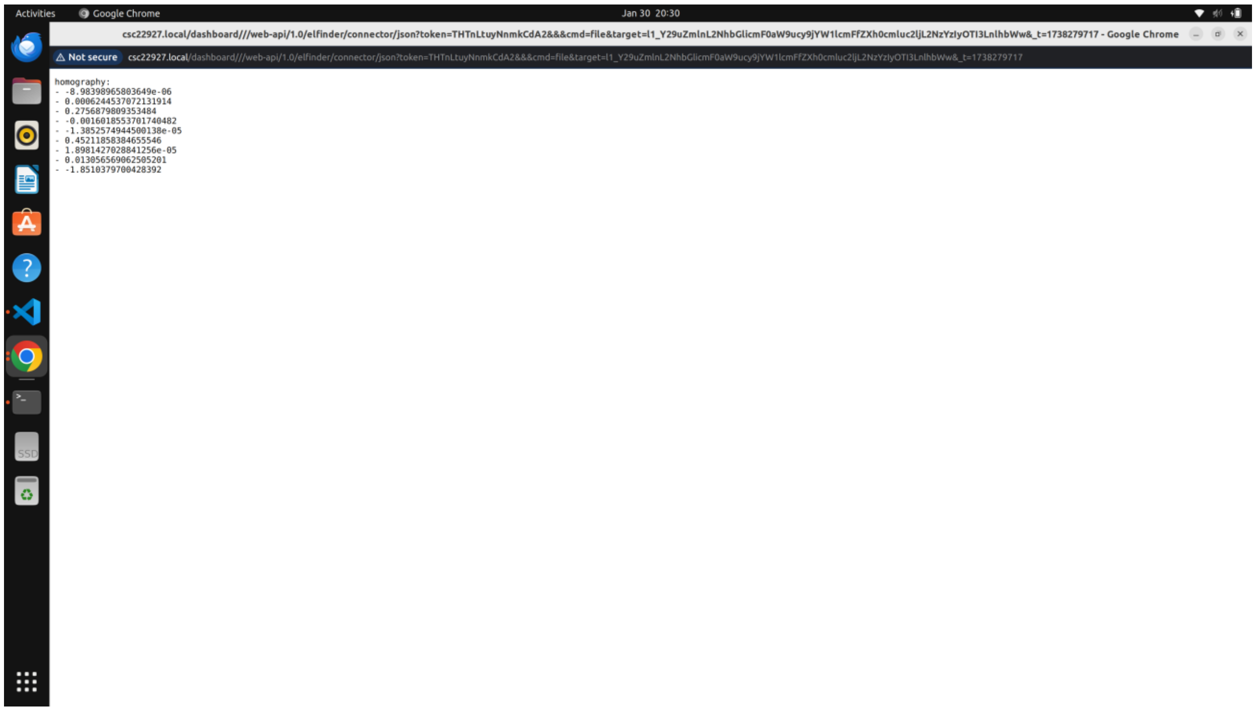
● A screen capture of your intrinsics calibration.yaml le



```
camera_matrix:
  cols: 3
  data:
  - 294.8911756015051
  - 0.0
  - 282.9084661176938
  - 0.0
  - 292.8391692933703
  - 220.59888236624636
  - 0.0
  - 0.0
  - 1.0
  rows: 3
camera_name: csc22927
distortion_coefficients:
  cols: 5
  data:
  - -0.2639995327258444
  - 0.05243740182700257
  - 0.0017333454324184162
  - 0.0008931337906020404
  - 0.0
  rows: 1
distortion_model: plumb_bob
image_height: 480
image_width: 640
projection_matrix:
  cols: 4
  data:
  - 207.89430236816406
  - 0.0
  - 283.67126465643196
  - 0.0
  - 0.0
  - 222.1710968017578
  - 209.55177676601124
  - 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
  - 1.0
  rows: 3
```

This one was a bit of a pain to implement. Making sure the box was not moving too fast while also moving it enough to satisfy all the constraints took me a while to understand how to do it properly. The previous camera calibration I had done in a different lab required the chessboard pattern to remain at the same depth from the camera, so I had made that assumption initially. However, after a while of things not working, I started moving it all over the place and the calibration happened much faster.

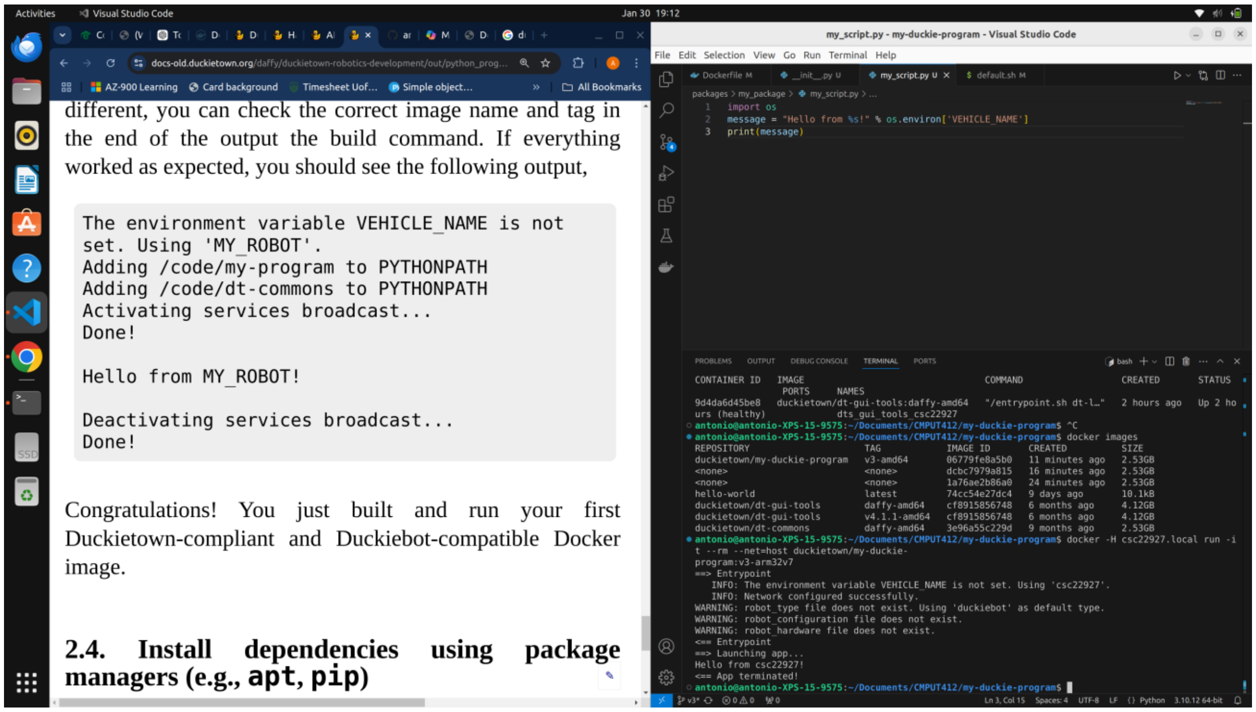
- A screen capture of your extrinsics calibration.yaml



```
homography:
-  -8.98398965803649e-06
-  0.0006244537072131914
-  0.2756879809353484
-  -0.0016018553701740482
-  -1.3852574944500138e-05
-  0.45211858384655546
-  1.8981427028841256e-05
-  0.013056569062505201
-  -1.8510379700428392
```

Thankfully this calibration was simple. I did not understand the point of doing this calibration if we had already moved the chessboard throughout the whole range of the camera's lens. However, I suspect that it is a check to see if the intrinsic calibration correctly identified the lens properties and could generate a rectified image of the chessboard.

● A screen capture of your Duckiebot saying "Hello from MY_ROBOT"



different, you can check the correct image name and tag in the end of the output the build command. If everything worked as expected, you should see the following output,

```
The environment variable VEHICLE_NAME is not
set. Using 'MY_ROBOT'.
Adding /code/my-program to PYTHONPATH
Adding /code/dt-commons to PYTHONPATH
Activating services broadcast...
Done!

Hello from MY_ROBOT!

Deactivating services broadcast...
Done!
```

Congratulations! You just built and run your first Duckietown-compliant and Duckiebot-compatible Docker image.

2.4. Install dependencies using package managers (e.g., **apt**, **pip**)

```python
import os
message = "Hello from %s!" % os.environ['VEHICLE_NAME']
print(message)
```

```
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS
                 PORTS     NAMES
9d4da6d45be8   duckietown/dt-gui-tools:daffy-amd64   "/entrypoint.sh dt-l…"   2 hours ago   Up 2 ho
urs (healthy)            dts_gui_tools_csc22927
antonio@antonio-XPS-15-9575:~/Documents/CMPUT412/my-duckie-program$ ^C
antonio@antonio-XPS-15-9575:~/Documents/CMPUT412/my-duckie-program$ docker images
REPOSITORY                    TAG            IMAGE ID       CREATED          SIZE
duckietown/my-duckie-program  v3-amd64       06779fe8a5b0   11 minutes ago   2.53GB
<none>                        <none>         dcbc7979a815   16 minutes ago   2.53GB
<none>                        <none>         1a76ae2b86a0   24 minutes ago   2.53GB
hello-world                   latest         74cc54e27dc4   9 days ago       10.1kB
duckietown/dt-gui-tools       daffy-amd64    cf8915856748   6 months ago     4.12GB
duckietown/dt-gui-tools       v4.1.1-amd64   cf8915856748   6 months ago     4.12GB
duckietown/dt-commons         daffy-amd64    3e96a55c229d   9 months ago     2.53GB
antonio@antonio-XPS-15-9575:~/Documents/CMPUT412/my-duckie-program$ docker -H csc22927.local run -i
t --rm --net=host duckietown/my-duckie-
program:v3-arm32v7
==> Entrypoint
   INFO: The environment variable VEHICLE_NAME is not set. Using 'csc22927'.
   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 csc22927!
<== App terminated!
antonio@antonio-XPS-15-9575:~/Documents/CMPUT412/my-duckie-program$
```

This was a pretty interesting view into how Duckiebots run containerized instructions. I found it interesting how they combine python packages with containerization, but I am not sure why one requires the other. The other interesting point is that I could write a container that could run on the Duckiebot which would explain why everything needs to be containerized so that a program can run on either the host machine or the duckiebot machine. I did run into a problem because of this implementation. I initially tried to run the lab from my own docker container instead of dual booting. However, I could not detect any of the Duckiebots using dfs fleet discover, so I had to scrap that whole idea. However, I believe that part of the problem is the network bridging becomes complicated when you nest docker containers. Overall, this was a very informative section of the exercise.

Colour Detector

This task seemed so simple at first, but I ran into so many errors with it that I could not seem to get around. I tried creating the same Dockerfile described in the instructions, but I got a ping error were trying to connect to the Duckiebot caused a TCP error. I then tried using the DB21M version of the command, but it did not work. So, I decided to try using the tutorial for creating Duckiebot Dockerfiles from section B-2, but then opencv-python would not compile correctly and my Duckiebot crashed. I did all this after disabling the dt-interface, so I know it is not a camera connectivity issue. I then tried to ssh into the Duckiebot to install opencv locally but could not access it as root to install pip. I then decided to give up on getting it to work because I wasn't sure how else to go about solving the problem.

A short overall write up about what you implemented, how it works, what you learned,

what challenges you came across, and how you overcame the challeng