ELEC330 ASSIGNMENT 3

Jetbot Autonomous Navigation & Object Detection

Author:
Minghong Xu (201601082)

Module Co-ordinator:
Dr Heba Lakany

Abstract

This report details the development of autonomous navigation and object detection on a robotic platform called JetBot.

Declaration of academic integrity

13th May 2023

Contents

List of Figures

1 Executive Summary

1.1 Introduction

The original aims are to deploy simultaneous localisation and mapping (SLAM) algorithm, enable map-based autonomous navigation, and implement object detection on JetBot, an entry-level differential drive robot with a dedicated NVIDIA GPU for running deep learning algorithms [1]. After the Easter break, the co-ordinator devolved the aims to lane marking following-based navigation with autonomous emergency braking (AEB) when a rubber duck is detected [1].

Despite the modification of the assignment, all the features requested before and after have been attempted to be completed. Attempts have been made to improve the robustness of lane marking following and the performance of object detection. The map-based autonomous navigation is not yet functional, and efforts to resolve this are ongoing. The code is available on `https://github.com/MinghongAlexXu/JetBot`.

1.2 Methodology

The robot was equipped with a Jetson Nano. Sensors included an VL53L5CX for low-resolution distance ranging, a RPi camera module v2 for image processing, a 6DOF IMU, ICM20600, for position and orientation estimation, two encoders for feedback control of the motor speed, and a LDS-01 LiDAR for SLAM and navigation.

We began our process with the development environment set-up. We first connected the robot to the eduroam, then deployed a reverse proxy client to bypass the eduroam's intranet access restrictions. This was followed by the configuration of the SSH password-less login for remote development convenience. We used Docker container for getting OpenCV with GStreamer support and ROS2.

In order to enable the robot to move at the desired speed, a controller has been implemented. This controller receives a Twist message from a ”cmd_vel” topic and interprets it based on the mathematical model of differential drive kinematics to obtain the angular velocities of the two wheels. It then sends these velocities to the motor driver to move the motors.

Two line following algorithms were implemented. The first one is a simple proportional control. It takes an image and binaries the image by Otsu's method. To reduce the influence of clutter in the image, only one row of pixels is selected, and the assumption is made that the longest consecutive white pixels represent the line. The midpoint of the line is then calculated. By comparing the midpoint of the line with the midpoint of the image, an error can be obtained. Based on this error, the robot can be adjusted to move either to the left or to the right. The improved algorithm is a PD controller. It fits a parabola to a bird-view transformed binarised image to model the line. The parabola is then utilised for calculating the trajectory the robot should follow: $\kappa' = -K_p(y_e + K_d \sin\psi_e) + \kappa$, where y_e is the distance error from the robot to the nearest point on the parabola, ψ_e is the heading angle error of the robot with respect to the parabola, κ is the curvature of the parabola, and κ' is the curvature of the trajectory that the robot should follow [2].

The duck detection feature was implemented by using You Only Look Once (YOLO), a family of object detection models based on the deep learning technique. For convenience, YOLOv8, which is supported by an extensive ecosystem, was chosen. Considering the need to detect only one object and the limited memory capacity of Jetson Nano, the smallest architecture, YOLOv8n, was selected. The model was trained on a dataset comprising one hundred manual captured and labelled rubber duck images. The training program of YOLOv8 comes with built-in data augmentation, therefore, even with a small amount of data, it is still possible to train a model that has sufficient robustness and accuracy. For best inference performance on the NVIDIA's Tegra, the trained YOLO model was run with NVIDIA TensorRT, which gave the lowest memory usage and highest frame per second.

The line following and duck detection were integrated using the communication mechanism of ROS2. Firstly, there was a camera node responsible for capturing video frames from the camera and publishing them to a camera topic. Both the line following node and the duck detection node subscribe to this topic and start their respective processing. The duck detection node published a ”duck_detected” topic. The underlying controller node, which is responsible for converting Twist messages from the line following node into motor control signals, subscribes to it. When the controller receives a ”True” message from the ”duck_detected” topic, it unsubscribes from the line following node's ”cmd_vel” topic and stops the motors. When the controller receives a ”False” message from the ”duck_detected” topic, it resubscribes to the line following node's topic to resume operation.

The implementation of SLAM heavily relies on the Google Cartographer system. Running the Cartographer 2D SLAM algorithm requires 2D depth data and data from the IMU's accelerometer and gyroscope. The LDS-01 sensor has a ready-to-use driver that can publish LaserScan messages as a node. As for the IMU, a driver and a node for publishing the data were implemented based on the sensor's data sheet. For SLAM visualisation, Foxglove was used. By leveraging Foxglove's teleop panel along with the differential drive controller, the robot can be remotely controlled to perform mapping within the Foxglove interface.

Autonomous navigation relies on ROS2's Nav2 package. This is a sophisticated system that requires parameter tuning to work on specific robot platforms. Its visualisation was also handled by Foxglove. The SLAM and autonomous navigation can work simultaneously. The SLAM algorithm provides localisation information, replacing methods like AMCL for estimating the current robot position. Additionally, instead of using the teleop panel to manually control the robot's movements, the navigation goal can be specified in a 3D panel with current map displayed, and let the navigation system handles the motion planning.

1.3 Results

The results were demonstrated during the bench-inspection day. Overall, nearly all aims were achieved, except the autonomous navigation system was not producing any motion control outputs despite the absence of error messages, and the imporved line following algorithm was not functional. Further investigation were required.

1.4 Discussion

The current SLAM system exhibits a severe map drift issue, which may be attributed to the low accuracy and data rate of the utilised LiDAR sensor. In addition, the rough data from the IMU could also be a contributing factor, and this issue could be mitigated by incorporating a Kalman filter.

Thresholding is a simple method for image segmentation. The improved line following algorithm utilises the Otsu's thresholding technique to extract line markings. However, it is susceptible to interference from clutter in the image, which can lead to inaccuracies in fitting the parabolic representation of the line. Therefore, it is necessary to employ a more reliable method for segmenting the pixels of the line marking.

The official motor driver only provides simple PWM control, which results in the inability to control the robot to reach the desired speed. The limitation of the motor driver is one potential cause for the failure of the improved line-following algorithm, which outputs the linear and angular velocity that robot should follow. To address this issue, encoder data can be integrated into the controller to establish feedback control and mitigate this limitation.

The VL53L5CX and LSM303D are not suitable sensors for SLAM. The VL53L5CX sensor can only sample up to 64 points, which is far from sufficient for 3D SLAM, and its sensing range is extremely limited. Moreover, the manufacturer, ST, does not provide information on how to compute the coordinates of each sample point relative to the sensor, which poses a significant challenge. The LSM303D sensor lacks a gyroscope, which is essential for most SLAM algorithm.

Image data is a large volume data type, and transmitting it requires a high bandwidth. The inherent latency in image data transmission can result in delays that make it challenging to perform inference and corresponding actions within an acceptable timeframe. Image transport addresses this issue by compressing image information and transmitting data over a lower bandwidth, to some extent alleviating the problem. If image transmission occurs on the same machine, the optimal approach is to use zero-copy, which not only saves on system resources but also provides the fastest transfer speeds.

The ROS communication mechanism serves as an interface between different processes, effectively separating a complex system into independent components. These components can even be implemented using different programming languages and language versions. However, ROS, as a communication middleware, is not without its flaws. It is built on top of Linux, which does not inherently possess real-time capabilities. There have been attempts to migrate ROS to some real-time operating systems (RTOS), but it is currently not mature enough. Furthermore, the implementation of ROS communication itself is not ideal. It is not only challenging to use but also lacks the necessary speed to support systems with high-speed communication requirements. Industrial-grade robot systems may require alternatives to ROS, such as eCAL, IceOryx, Fast DDS, and Cyclone DDS, to fulfill their needs.

1.5 Reflection

Firstly, I acknowledge that relying heavily on one person to complete a group task is not ideal. It's true that I've been trying to shoulder most of the responsibility, and it appears that my actions might have unintentionally discouraged others in the group from participating. I should have been more mindful of the importance of everyone's involvement and how my actions could impact the team's dynamics.

Secondly, it's clear from the feedback that the group feels disengaged and uninvolved. This isn't a situation I want to perpetuate. It's important for everyone to have a voice and feel like they are part of the process. I understand that everyone is attending classes and wants to contribute to our group work, and I need to do better at fostering that environment.

Lastly, I need to appreciate the effort others are putting into their work, even if it doesn't meet my exact expectations. Criticism should be constructive and serve as a means of improvement rather than discouragement.

References

[1] H. Lakany, *ELEC 330 assignment 3 brief*, University of Liverpool Canvas, Jan. 2023.

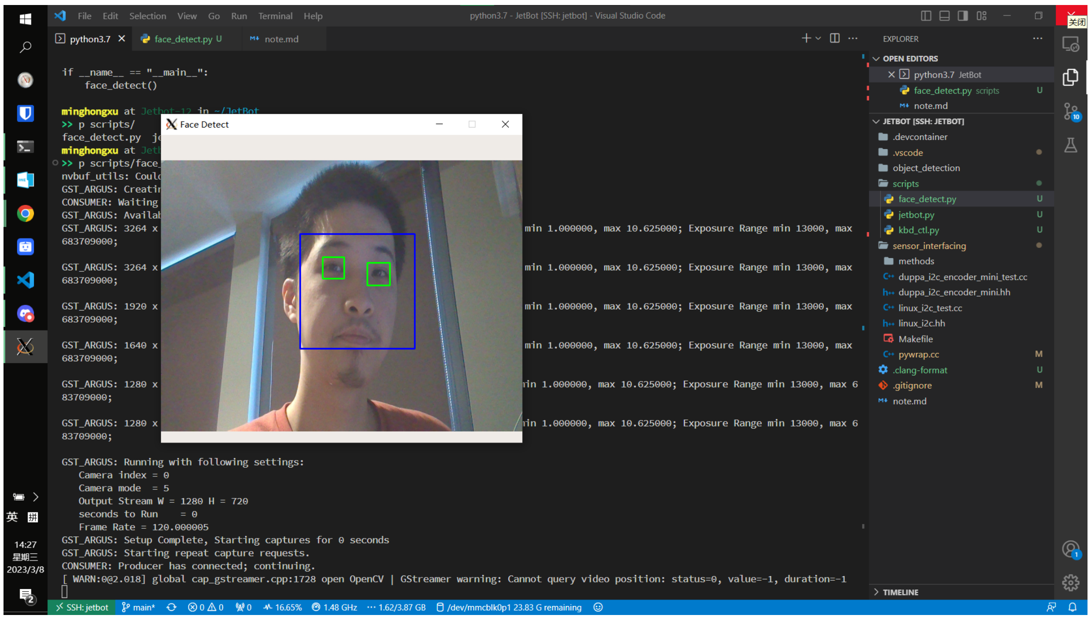
[2] A. Sloane, *Fast line-following robots*, Nov. 2018. [Online]. Available: `https://a1k0n.net/2018/11/13/fast-line-following.html` (visited on 20/04/2023).

Appendices

Important details such as the steps gone through were documented in `https://github.com/MinghongAlexXu/JetBot/tree/16de560ec3843861ca7abe4e7b711e4e5e5ece9d/note`.

A Interfacing sensors

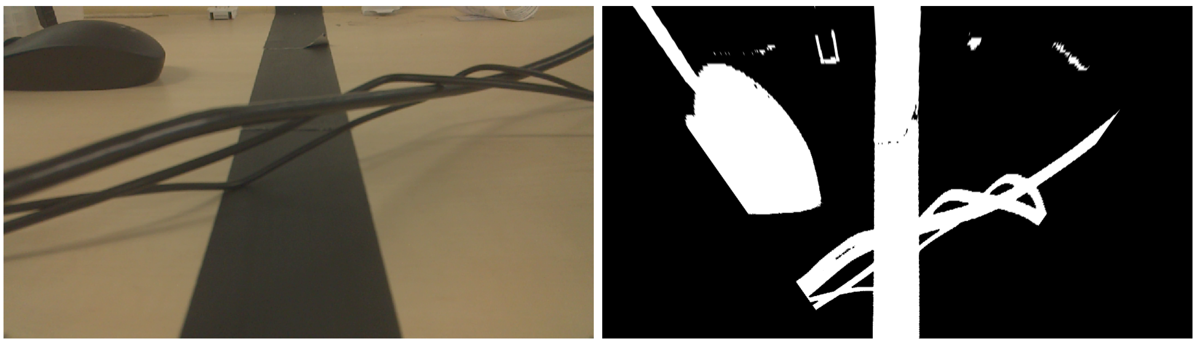
I2C Linux driver and drivers for DuPPa I2C Encoder Mini, LSM303D, and ICM20600 were implemented in C++. Unit test for each driver were also written.



```
if __name__ == "__main__":
    face_detect()

minghongxu at Jetbot-12 in ~/JetBot
>> p scripts/
face_detect.py  je
minghongxu at Jet
>> p scripts/face_
nvbuf_utils: Coul
GST_ARGUS: Creati
CONSUMER: Waiting
GST_ARGUS: Availa
GST_ARGUS: 3264 x                                    min 1.000000, max 10.625000; Exposure Range min 13000, max
683709000;

GST_ARGUS: 3264 x                                    min 1.000000, max 10.625000; Exposure Range min 13000, max
683709000;

GST_ARGUS: 1920 x                                    min 1.000000, max 10.625000; Exposure Range min 13000, max
683709000;

GST_ARGUS: 1640 x                                    min 1.000000, max 10.625000; Exposure Range min 13000, max
683709000;

GST_ARGUS: 1280 x                                    in 1.000000, max 10.625000; Exposure Range min 13000, max 6
83709000;

GST_ARGUS: 1280 x                                    in 1.000000, max 10.625000; Exposure Range min 13000, max 6
83709000;

GST_ARGUS: Running with following settings:
   Camera index = 0
   Camera mode  = 5
   Output Stream W = 1280 H = 720
   seconds to Run    = 0
   Frame Rate = 120.000005
GST_ARGUS: Setup Complete, Starting captures for 0 seconds
GST_ARGUS: Starting repeat capture requests.
CONSUMER: Producer has connected; continuing.
[ WARN:0@2.018] global cap_gstreamer.cpp:1728 open OpenCV | GStreamer warning: Cannot query video position: status=0, value=-1, duration=-1
```

Figure 1: Testing the CSI camera

B Lane marking following

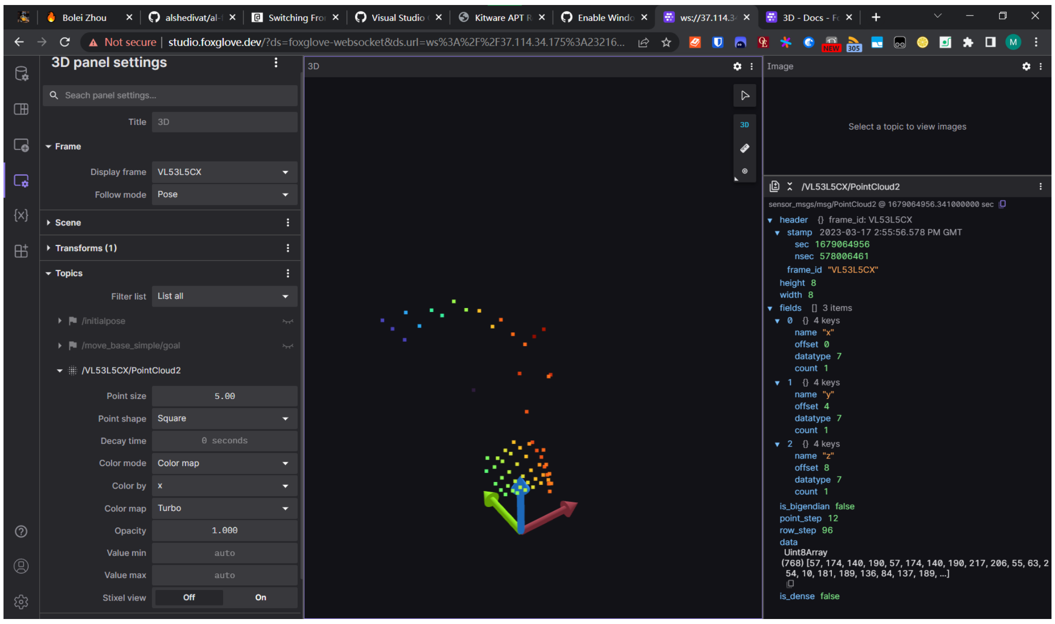


(a) Original perspective

(b) Segmented and bird-view transformed

Figure 2: Image processing before fitting a parabola to the lane mark

C SLAM with Google Cartographer

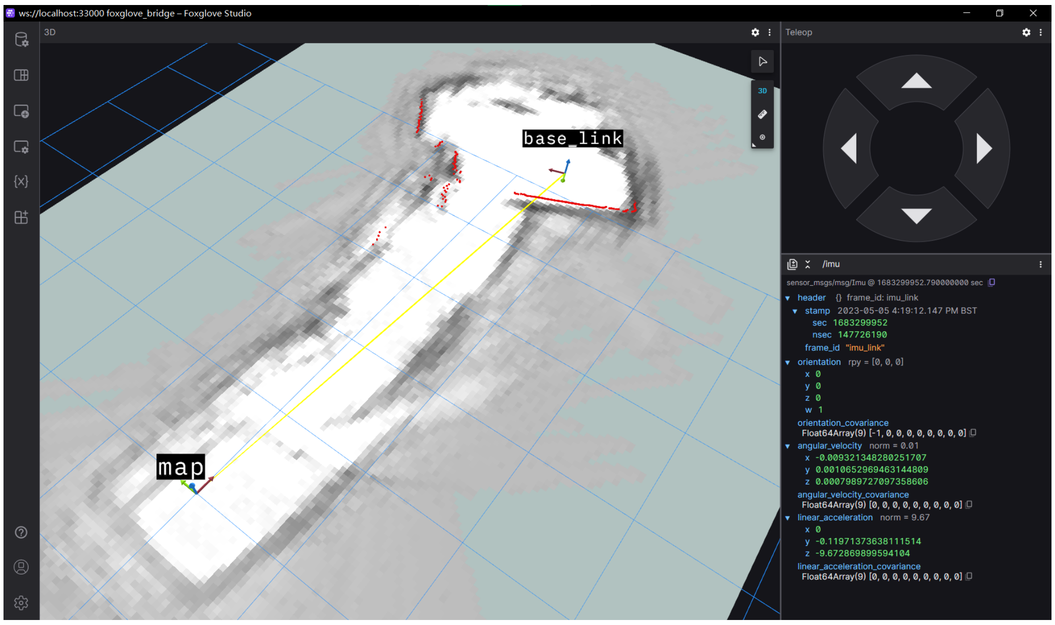


3D panel settings

```
Title          3D
▾ Frame
  Display frame  VL53L5CX
  Follow mode    Pose
▸ Scene
▸ Transforms (1)
▾ Topics
  Filter list    List all
  ▸ /initialpose
  ▸ /move_base_simple/goal
  ▾ /VL53L5CX/PointCloud2
    Point size     5.00
    Point shape    Square
    Decay time     0 seconds
    Color mode     Color map
    Color by       x
    Color map      Turbo
    Opacity        1.000
    Value min      auto
    Value max      auto
    Stixel view    Off  On
```

```
/VL53L5CX/PointCloud2
sensor_msgs/msg/PointCloud2 @ 1679064956.341000000 sec
▾ header  {} frame_id: VL53L5CX
  ▾ stamp  2023-03-17 2:55:56.578 PM GMT
      sec  1679064956
      nsec  578006461
    frame_id  "VL53L5CX"
  height  8
  width  8
▾ fields  [] 3 items
  ▾ 0  {} 4 keys
      name  "x"
      offset  0
      datatype  7
      count  1
  ▾ 1  {} 4 keys
      name  "y"
      offset  4
      datatype  7
      count  1
  ▾ 2  {} 4 keys
      name  "z"
      offset  8
      datatype  7
      count  1
  is_bigendian  false
  point_step  12
  row_step  96
  data
   Uint8Array
  (768) [57, 174, 140, 190, 57, 174, 140, 190, 217, 206, 55, 63, 2
   54, 10, 181, 189, 136, 84, 137, 189, ...]
  is_dense  false
```

Figure 3: VL53L5CX data visualised in Foxglove



```
/imu
sensor_msgs/msg/Imu @ 1683299952.790000000 sec
▾ header  {} frame_id: imu_link
  ▾ stamp  2023-05-05 4:19:12.147 PM BST
      sec  1683299952
      nsec  147726190
    frame_id  "imu_link"
▾ orientation  rpy = [0, 0, 0]
    x  0
    y  0
    z  0
    w  1
  orientation_covariance
   Float64Array(9) [-1, 0, 0, 0, 0, 0, 0, 0, 0]
▾ angular_velocity  norm = 0.01
    x  -0.009321348280251707
    y  0.0010652969463144809
    z  0.0007989727097358606
  angular_velocity_covariance
   Float64Array(9) [0, 0, 0, 0, 0, 0, 0, 0, 0]
▾ linear_acceleration  norm = 9.67
    x  0
    y  -0.11971373638111514
    z  -9.672869899594104
  linear_acceleration_covariance
   Float64Array(9) [0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Figure 4: Map of a corridor with a 2D LiDAR