Wrocław University of Science and Technology
Faculty of Electronics

Image processing based line follower

Author: Kamil Goś

Class: Intermediate Project

Supervisor: Ph.D. Witold Paluszyński

Date: January 31, 2021

Abstract

The project involved developing a line tracking algorithm based on image processing algorithms. It was aimed to replace the commonly used reflection sensors in favour of a camera. In order to test the algorithms, the project also assumed the creation of a mobile platform. Project assumed to create the final system using Python language and Robot Operating System (ROS).

The project was a success. Image processing algorithms, regulator algorithms, as well as a test platform, were prepared and robot successfully followed the line. The whole system was implemented independently using both raw Python language and ROS. The system has been successfully tested. There was also no discrepancy between the two implementations (Python and ROS).

This work is licensed under a Attribution-NonCommercial 4.0 International license.

1 Introduction

Line followers are robots that follow the line that marks the track. Often there are competitions for which people can submit their robots and the constructor of the fastest vehicle wins. The route on which the races take place is usually marked by a black line (15 to 20 mm wide), which is placed on a white ground. The most common approach to build a line-following robot is to build construction with reflection sensors on the board. Such sensors can successfully detect the black line and send their output to some regulator, that process it to control signal. Example of such construction can be found in Pakdaman and Sanaatiyan [2009].

This project assumes that such sensors can be replaced by a camera. Having camera on board, frames can be read and using the image processing algorithm the black line can be detected similar to reflection sensors. Similar approach was introduced in Kondakor et al. [2018].

1.1 Assumptions

This project aimed to create the following parts:

- building a fully operational mobile platform, that can follow the line using developed algorithms,
- developing the image processing algorithms for line detection and error extraction,
- creating a closed-loop control system based on PD regulator,
- developing the drivers for PWM generators to control motors and servo,
- developing the whole system using raw Python code and Robot Operating System as two independent systems and compare them,
- building a custom map (track), that can be used to test the created platform and algorithms.

2 Summary

The following sections describe the most important components of the system. The software part of system was implemented using the following tools:

- Python (version: 3.9.1)
- ROS (version: melodic with Python3)
- OpenCV (version: 4.5.1)

2.1 Mobile platform

This part of the project involved creating a four-wheeled vehicle with a front steering axle and a rear driving axle. Figure 1 and figure 2 shows the created platform front and back side. The platform consist of the following elements:

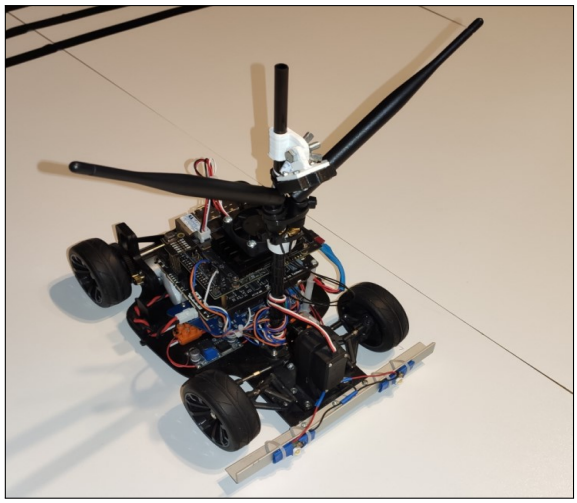

Figure 1 – Font side

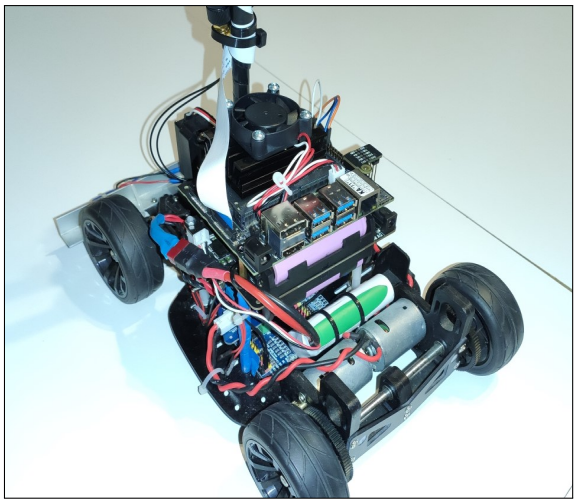

Figure 2 – Back side

- four-wheel chassis with rear wheels driven by two independent 5V DC motors and front axle controlled by one servo,
- motors controller,
- 2 PWM signal generators,
- Li-Po battery for powering both DC motors,
- DC-DC converter,
- Jetson Nano Developer Kit,
- Uninterruptible Power Supply UPS T-208 for powering Jetson Nano,
- Wi-Fi card with antennas,
- RGB camera.

Connections between those elements are presented in figure 3.

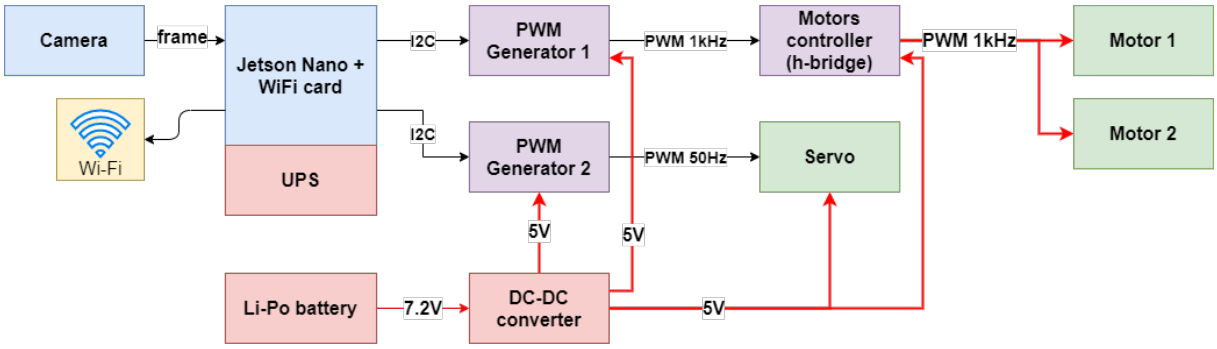

Figure 3 – Signals flow graph

The main processing unit is Jetson Nano Developer Kit. It is powering by an independent uninterruptible power supply that delivers enough power to run Jetson in high power mode.

It is connected with Jetson using a delivered coupler. Directly to Jetson the WiFi network card is connected so it is passable to establish wireless connection between Jetson and any other computer. To increase the signal strength, two additional antennas were attached to the network card. Next element is the camera. It is connected with Jetson nano through MIPI CSI-2 interface. This type of connection gives very fast data transmission which is extremely useful for image processing applications. To control engines the Pulse Width Modulation signal generator and motors controller (h-bridge) were used. Jetson Nano is connected with PWM generator (adjusted to 1kHz) using the I2C interface. The PWM signal then goes to the H-bridge, is processed (amplified) and goes to the motors' clamps, so they rotate in the expected direction and at the expected speed. To control servo motor another PWM signal generator is used. This one is adjusted to work wich 50Hz frequency which is needed to control servo. It is also connected to Jetson Nano using the I2C interface. The generated PWM signal goes directly to the servo, so one can control its rotation. The whole system (excluding Jetson Nano) is powering by Li-Po battery through DC/DC converter.

2.2 Structure of system

The whole system consist of several dependent parts that communicate together. It was divided into 5 modules. First one is the image processing module, whose job is to extract the line from picture and calculate the error (described in details in section 2.3). Next module is the regulator. It transform error signal to control signal (described in section 2.4). Two modules supporting PWM generators were also created. One is designed to control DC motors and the other to control the servo deflection. The final executor of all modules function is main loop, which performs the superior function in the system. It continuously executes the algorithm thanks to which the robot is able to follow the line. Figure 4 presents the general idea of working system.

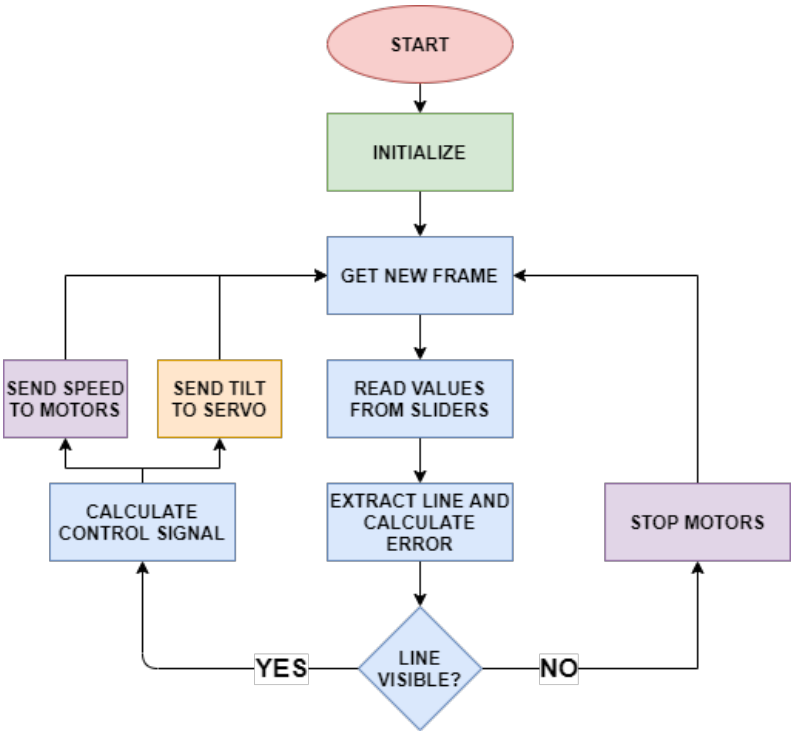

Figure 4 – System flow

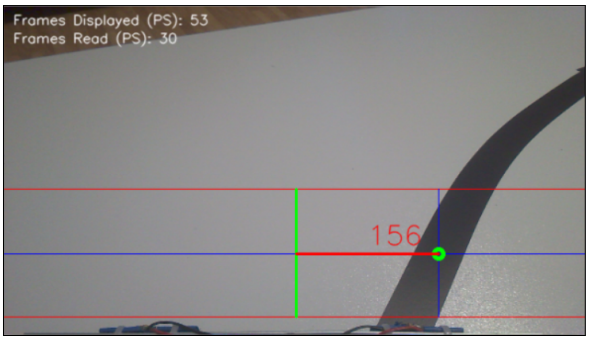

Figure 5 – Original frame

Figure 6 – Line extraction

Figure 7 – Feature extraction

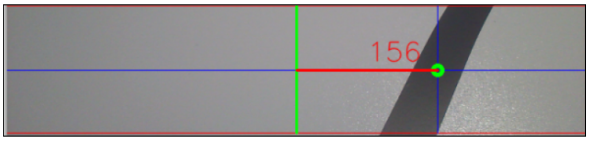

Figure 8 – Error calculation

2.3 Image processing module

The image processing module is responsible for extracting line form pictures. First of all, it reads the new frame from the camera. As the frames are quite large and no full area processing is required, each image is cropped. Figure 5 shows the original size frame. The red lines visible in the picture are related to a cropped area. In the next step, the colour space transformation from RGB to grayscale is performed. It reduces the image from 3 dimensions to only one, where all pixels are in grayscale. The next four steps are Gaussian blur, thresholding, erosion and dilation. The output of those algorithms is presented in figure 6. At that moment, the frame consists of a filtered white line and black background. The next phase is to extract the position of the line. For this purpose, the contours finding algorithm is used. Found contours are visible in the figure 7. Then, the idea is to find the centroid of the area defined by the contours. This goal can be achieved by using the method of moments. Figure 8 shows found centroid marked as a green dot. Also, it presents the methodology of error calculation. The error is calculated as the difference between the centre of contours area and the middle point of a considered frame. In this case, the error is +156. In the next phase, this value is sent to the controller module.

2.4 Control module

Control module is responsible for transforming error signal cumming from the image processing module to signal that control servo tilt. Developed control loop is closed-loop system (with feedback from camera) with PD controller. The structure of such system is shown in the figure 9.

Setpoint is related to the expected position of the robot. The error signal is higher than zero when a robot is on the right side of the line and lower than zero when a robot is on the left side, so the setpoint is zero. This is related to a robot position in the middle of the line. Then, the error signal is sent to the PD regulator. Term P is proportional to the current value of the error when term D is the best estimate of the future trend of error, based on its current rate of change. To achieve high-performance controller the balance of P and D part of the regulator is needed. To adjust the impact on the control signal of each part the K_p and K_d amplifiers are

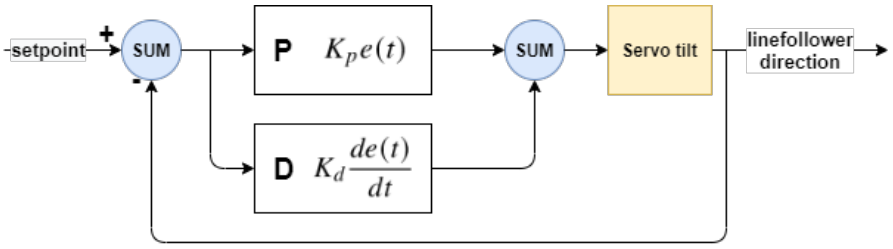

Figure 9 – Control system structure

used. As light (the random factor) has a large influence on the entire control system, the factors K_p and K_d can be selected manually by the user. Figure **??** shows the prepared, user-friendly Graphical User Interface for values adjustment. Besides the sliders for setting the values of K_p

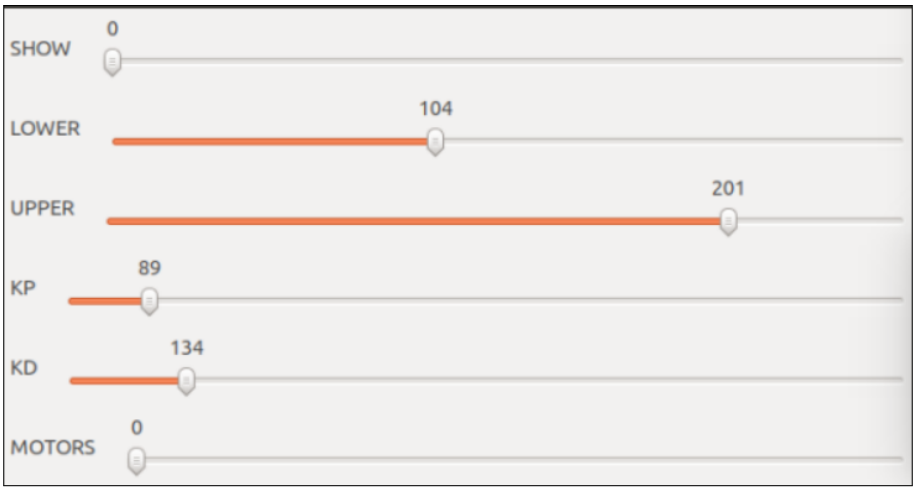

Figure 10 – Graphical User Interface

and K_d coefficients in the GUI, one can set whether the windows with the camera image should be displayed (*SHOW*), the motors speed (*MOTORS*) and the *LOWER* and *UPPER* threshold for image binarization.

2.5 Robot Operating System

This project also assumed the construction of the entire system using the Robot Operating System [1]. ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.
Figure 11 shows the structure of developed system [2]. The main elements of ROS system are nodes. Nodes are executive parts of system structure. They communicate together thorough topics using messages. This project consist of four nodes. First node is */camera* node, that run the image processing algorithms and reads values from GUI sliders. It publish message with error to */cam_error* topic and message with sliders readings to */trackback* topic. The next node

[1]The idea of using ROS in this project was to learn ROS. The system implemented using Python modules is fully functional and does not require ROS to run. Generally speaking, the system created with ROS is reconstruction of Pyhton system using tools and structures delivered by ROS.

[2]This graph was generated using *rqt_graph* ROS tool

is */controller*. It subscribe the */trackbars* and */cam_error* topics. It transform error signal to control signal and publish it to */tiltPub* topic. Node */tilt* subscribe */tiltPub* topic and send the proper (mapped) signal to PWM generator, that control the servo motor. There is also */motors* node that subscribe both */trackbars* and */cam_error* topics. */trackbars* topic deliver message with motors speed adjusted by user. */cam_error* topic deliver message with camera error. The system is created in such a way that if no line is found in the image, the error is infinite and */motors* node interpret is as "stop" signal.

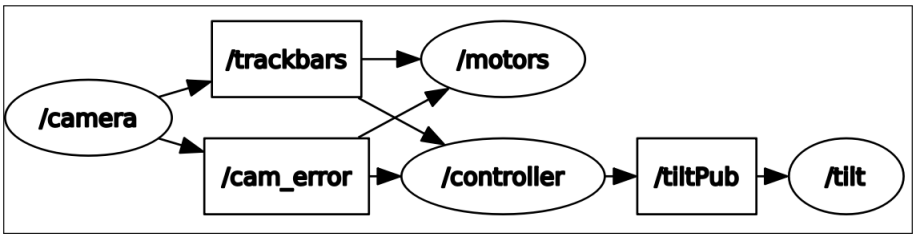

Figure 11 – ROS structure

2.6 Custom track

To test the robot the custom track was made. Picture of this track is shown in the figure 12. This is a close loop with some impediments. An "hdf" board with dimensions $2500x2070mm$ was used to build this track. The line was designated using 15mm black tape.

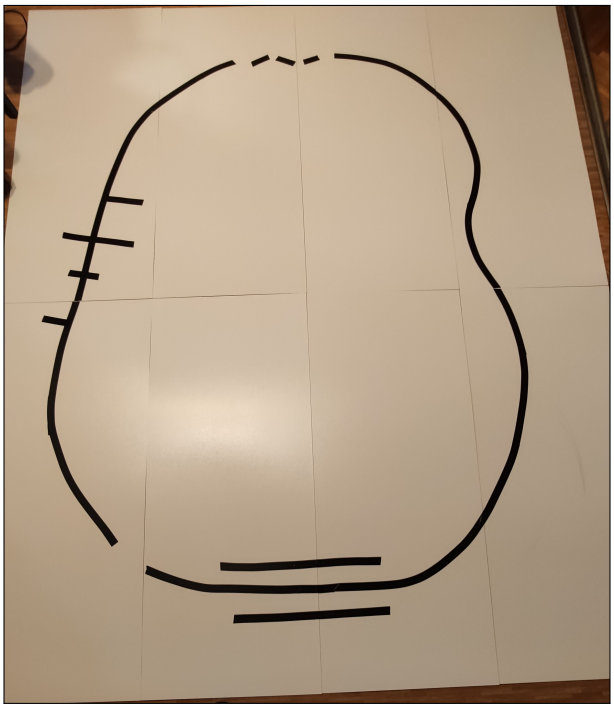

Figure 12 – ROS structure

3 Results

The designed and developed system met all assumptions. The project was a complete success. The created robot successfully followed the line with both algorithms implementation. It easily overcame all the difficulties on the track. Thanks to the use of Jetson Nano and GPU acceleration, it was possible to achieve a system that processes data 50 times per second, which is largely sufficient for line tracking. Movies presenting the working system are available on the YouTube platform:

- https://youtu.be/YFPQv9tbDRQ
- https://youtu.be/4GpwnwSXacw

3.1 Python and ROS systems comparison

There were no differences observed between the system implemented with Python and the system implemented with ROS. In both cases, an algorithm speed of about 50 loops per second was obtained.

References

Andras Kondakor, Zsombor Törcsvari, Akos Nagy, and István Vajk. A line tracking algorithm based on image processing. In *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 000039–000044. IEEE, 2018.

Mehran Pakdaman and M Mehdi Sanaatiyan. Design and implementation of line follower robot. In *2009 second international conference on computer and electrical engineering*, volume 2, pages 585–590. IEEE, 2009.

A Appendix

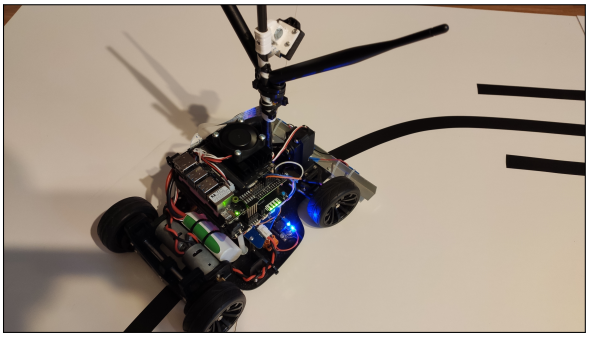

Figure 13 – Robot following line 1

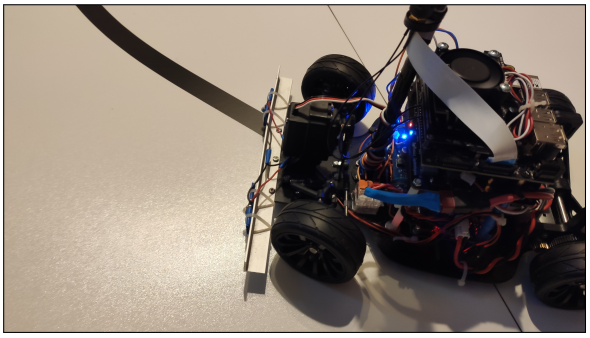

Figure 14 – Robot following line 2

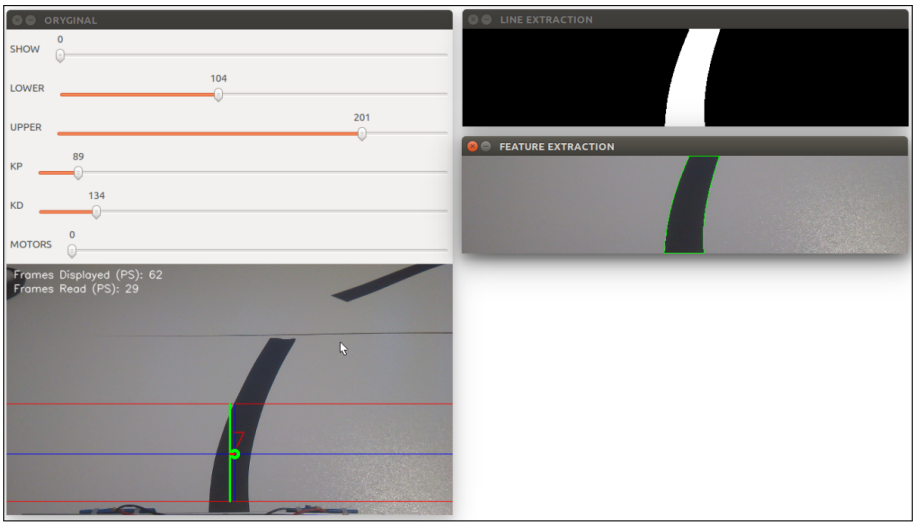

Figure 15 – Screenshot of entire GUI

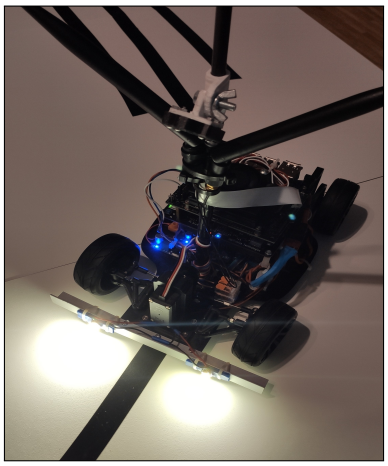

Figure 16 – Robot following line with lights turned on. The idea of mounting lights was to use them to illuminate the track in case of poor lighting conditions. This idea was not developed because the mounted lighting generated point light that interfered with the image processing algorithm.