BPC-PRP Robot Project

1st Vilém Strachoň
Faculty of Electrical Engineering and Communication
Brno University of Technology
Brno, Czech Republic
247472@vutbr.cz

2nd Jakub Vacek
Faculty of Electrical Engineering and Communication
Brno University of Technology
Brno, Czech Republic
247487@vutbr.cz

***Abstract*—Paper is about a semesters worth of work on course Robotics and computer vision, the project is about working and programming small robot that can be seen on picture 1. This robot is equipped with several different sensors, our team mainly choose to work with LiDAR and IMU. Outcome of this course is code for maze escape exam, in which the robot has to succeed. Through the paper we discuss issues found on the way and the way we programmed our robot.**

***Index Terms*—Robotics, project, computer vision, LiDAR, Autonomous driving**

I. Introduction

Course Robotics and Computer Vision is centred around given robot, through the course we were tasked to accomplish milestones in shape of completing driving test of Line following and Corridor following, final exam had form of autonomous maze escape.

To accomplish these tasks, the robot platform1 is equipped with several key perception sensors as LiDAR (Light Detection and Ranging) that allow the robot to detect obstacles and walls of the maze, next essential sensor was a camera, to detect ArUco tags along the way in maze. For the ”Line following” task, we utilised reflective line detection sensors, which enabled the robot to interpret visual difference of ground, upon which the robot drove. Another important sensor is the Inertial Measurement Unit (IMU) by which we were able to exactly turn in corners (as the instructions promised 90° corners). The main decision-making part of the robot was a Raspberry PI, where we implemented the ROS2[1] framework for program navigation and data acquisition.

The course instructed us to develop the project in C++ (as it is one of the preferred languages for implementing the ROS2 framework. The programming process involved creating Nodes that subscribed data from sensors like IMU_node, Nodes that published data such as Motor_node for speed control and multiple connecting ”decision-making” Nodes where we fused and processed data from multiple sensors. The programme was written as a state machine[2], and is then further divided into multiple data filters, for extracting the right data, as we found out, that much noise is inducted into useful data.

This paper presents a overview of the system architecture, sensor integration, software design and our iteration through the semester. We describe the challenges encountered during hardware-software interfacing, discuss the methods used for sensor filtration and data fusion. The results demonstrate the functionality of our approach in achieving reliable autonomous navigation and highlight areas for future improvement, such as advanced computer vision by robots camera.

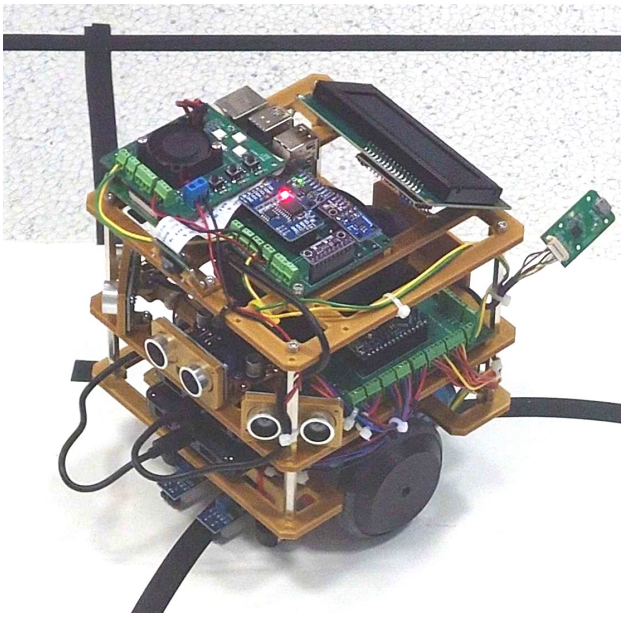

Fig. 1. Robot platform, following line

II. Main Text

To pass the Line following task of the robot platform evaluation tests we chose to work with data from IMU, rotational encoders, and two reflective optical sensors, in program we fused the data and then after evaluation algorithm sends appropriate response to two motors. The tracks robot needs to drive were known beforehand, it was straight line, Circle and shape of a number eight (8), these tracks were marked by black electrical tape, on semi-white surface (white linoleum with unevenly distributed dark grey spots. The tape had matt surface, while the floor has more reflective and glossier surface. In code, we established subscriber nodes to all sensors and one publisher, only to the motors. While testing we overcome few obstacles, mainly around physical construction of the robot, wheels were sometimes slipping on smooth surface of floor with dust, and two optical sensors were keen to the height from floor to sensor. In next Tasks we have forsaken their usage.

For algorithm to work we prepared data beforehand, as the data from optical sensors were unstable to external conditions like uneven surface, we found that once calibrated to black and white, they were stable enough, to distinguish where the tape ends. So we measured distance between sensors and in code calculated what values from both sensors correspond to offset from centre of the tape, both sensors can be seen in principal schematics 2 with green trapezoid representing the sensing angle of optical sensor. With the difference calculated, we send the value to PI regulator for which we used provided library of the course and the result value was then used in motor speed calculation. Base speed was set, and the code only subtracts or adds to it, where the highest impact of the regulator can have an effect of stopping the wheel, so the robot has fixed maximal curve sharpness, we did not investigate this issue furthermore as the exam's curves were not that sharp.

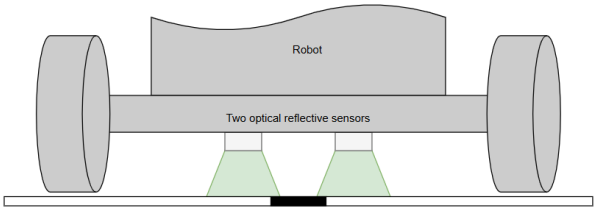

Fig. 2. Optical sensors, sensing line in the middle

Next more challenging task was corridor following, for this task we decided to use mainly the LiDAR module, which appeared as most accurate and useful data source, for robot centring and for wall (crash) avoidance. Other used sensors are the IMU and encoders, used in similar manner as in Line following task.

As the LiDAR provided for us data in 360°, we filtered it by radians, to get only data from 60° front, back, left and right. Data filtering was implemented to store data to vector (front, back...) if the angle condition is met, angles were transferred to natural radians, so data in "front" vector, had to pass to have lesser angle than $(\frac{\pi}{2} + \frac{\pi}{6})$, and a greater angle than $(\frac{\pi}{2} - \frac{\pi}{6})$. This was drawn into diagram of Data used by algorithm 3, where only blue filled cones were taken in consideration in Corridor following task, The red dot represents mean value of distances that robot considers to be in that place. Angles were added to the picture to better represent size of measured data.

For distance from walls, we calculated mean value of data in front, left and right vectors. This then gave us metric distance from walls. Here we encountered only minor difficulties, as the installed LiDAR is not capable to calculate distances shorter than 14 cm, if it does, he returns infinite values, so we decided to overwrite this with countable number zero, similarly when the LiDAR detects distances greater than 30 cm (when he detects corridor for example) we overwrite the value with 20 cm, this assures us that the robot wont get confused and go straight until in these situations. This method is not perfect, but we decided to keep it for Corridor following, and it was later changed for final exam.

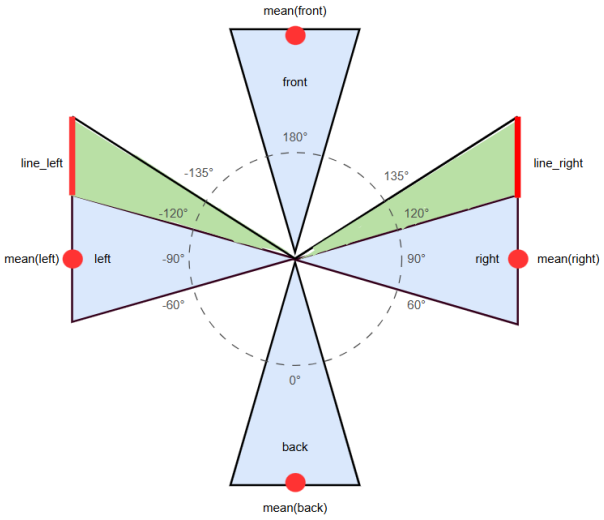

Fig. 3. Ranges of LiDAR's beams used in code

This distances from left and right wall allowed us to determine centre of corridor and use similar regulator as in Line following. After few tests rides we modified the regulator. We used two regulators, main one had proportional and derivative part and the second one only proportional part. This way, we were able to accomplish the second task, where we followed straight corridor, drove in corridor between two squares when looked from top, and corridor in shape of a number eight (8).

Last part of the course was the exam – maze escape, when we need to escape with best possible time (up to limit of 5 minutes). Maze consists of corridors with 90 degree and 180 degree turns.. Moreover in the maze are tiles that give us time penality if robot steps into them (minotaur which delays the escape) and also one tile that has positive effect of subtracting final time (treasure). Right way out of the maze is known and can be advised to the robot before each corner by reading a code by camera. ArUco tags can be found in corridors, showing always the right direction on the next intersection, therefore how to avoid minotaurs and how to find treasure.

Mentioned LiDAR was also craving for an modification. Our team decided to implement wall detection by an approximating line, this line enables the robot to detect existing corridor on left or right. Green cones on picture 3 represent the sensing angle for this role of detecting turn in corridor. We filter cone by angles as described above, here with modification of sensed angle to 15°, then we applied median filter for five nearby distance values in vector (by this we were able to eliminate random high numbers caused by disconnected maze walls). With this data, we calculated approximating line itself (represented by red line in picture 3), by method of Smallest-Squares. This gave us line, which we could test in real corridor.

This testing affirmed us in deciding to implement this method, robot was able to identify turning corridor by parameters of equation ($y = ax+b$), where the "a" parameter referred us to rotation of the robot compared to semi-straight corridor walls, and the parameter "b" that holds the information of

offset from origin of x-y axes (thus centre of the robot. At this phase we considered using parameter ”b” from left/right_line calculation to evaluate exactly centre of corridor, but we chose to keep working model as is for the moment, as the previously mentioned calculation of left and right mean values allows the robot to drive though 4-way corridor and when turning, allowed us to drive to centre of the square. Because at this moment robot is detecting corner turn few seconds before he is able to turn without colliding with wall.

Our code turned to be be ineffective on first official final exam. Where turns and crossroads were connected instantly to one and other, when on ”practise maze” in presentations turns had at least one square between them, on which we heavily depended for our decision making (robot was able to deal with each turn independently). This forced us to create new way of detecting crossroads. This way consists of calculating all distances shown on picture 3, detecting ”gaps” in walls, and deciding if crossroads is visible, if yes, artificial delay is called for about one second. After that time another measurement is taken and if corridor is still detected robot can turn.

This two ways of detecting corridor work in parallel, where each can decide to turn. We decided for this because when approximating lines, there is no need to add delays to the program, so it runs more smoothly.

At this point it is necessary to explain code more and how the robot makes decisions based on incoming data. Robot is programmed as simple state machine where the moving phases are shown on picture 4 by states (Corridor following has output of motor speed in centre of corridor) and reacts to sensor data by changing the state, thus when collision is being foretold, state ”Stop” is called and movement stops. By this description we implemented Moore type automat.

Implemenation of camera was also needed for successful escape from maze, once we were able to show compressed image, with number of detected ArUco tag, we set list of tags to read, this list can be changed, in default it is set to look for TREASURE mode tags, thus we read only 3 from 6 possible tags. Once robot founds tag, he stores it until he meets crossroads on his way, this was another challenge for us, because until this point we did not differentiate between crossroad and corridor turn. However now the code isa able to tell the difference.

III. Conclusion

The goal of the course was accomplished successfully, programmed robot was able to autonomously drive out from maze, without collision with walls.

Course enabled us to learn on physical robots with many different sensors, think about possible ways of controlling the robot, and implement any way of robot control we could think of or find on the internet. We had a chance to see robots reaction to our code, and implement real life regulators, which we previously only simulated in different courses. Moreover we had chance to get started with ROS2 framework which we are grateful for.

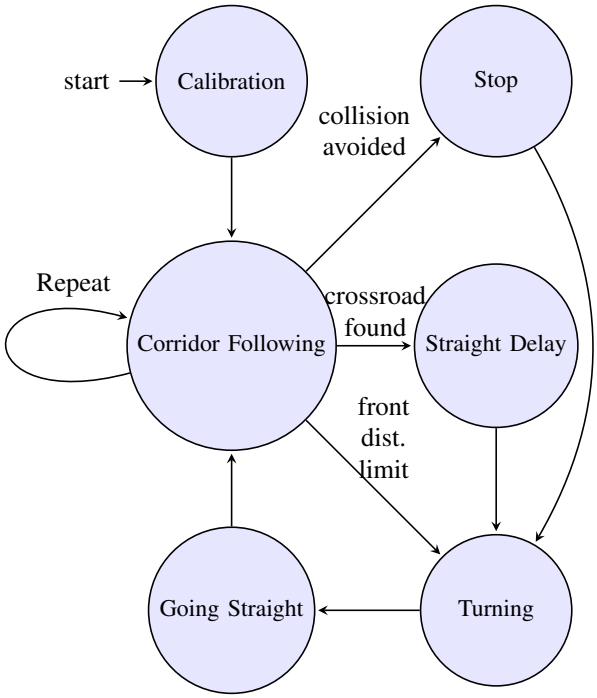

Fig. 4. Obrázek stavového automatu

On the negative points, we do think there is major difference in difficulty from ”driving in corridor” to ”maze escape”, the first and the second tests were possible to ace easily, implying the exam will not be different. Next we would really appreciate to gain ”known knowledge” in this really interesting field of mobile robot automation, as in first lecture was said ”not to reinvent the wheel”, sadly we felt pushed to this. In the end, We are both curious what is one of ”optimal ways” of controlling this robot. And weather our invented way of robot control (for example approximating position of the wall, or decision not to use Odometry or Kinematics) is optimal in any way.

In the end we both feel to have gained crucial knowledge in field of robotics and we are really happy to be able to work with robots in real life, seeing their reactions to prescribed code.

Acknowledgment

We would like to thank our mentors of BPC-PRP course for the insight to Robotics and for the opportunity to work with physical robots.

References

[1] “Framework for robotic systems, available for downolad at: https://github.com/ros/ros”. In: (2017).

[2] V.V Solov’ev. “Implementation of finite-state machines based on programmable logic ICs with the help of the merged model of Mealy and Moore machines”. In: (2013).