



Line Following Algorithms

Following lines is still an effective approach to implement autonomous robots in many applications, especially in manufacturing areas as shown in Figure 1.



Figure 1: Robots follow lines for autonomous operation in factories

In this tutorial, you will develop line following algorithms based on the line follower sensor BFD-1000 mounted on the robot last week. The algorithms include a simple left/right approach and a more adaptive approach based on the PID controller.

A. Simple algorithm

As shown in Figure 2, a simple algorithm would rely on the following actions:

- Go straight when the robot is on the line (Situation 1)
- Turn left when the robot deviates from the line to the right (Situation 2)
- Turn right when the robot deviates from the line to the left (Situation 3)

In case the line follower sensor has more than two line detectors (5 with BFD-1000), a better estimate of the robot position can be obtained, e.g., far left/far right of the line. In that case, the algorithm can be modified to add different level of left turn/right turn to improve the line tracking performance of the robot.

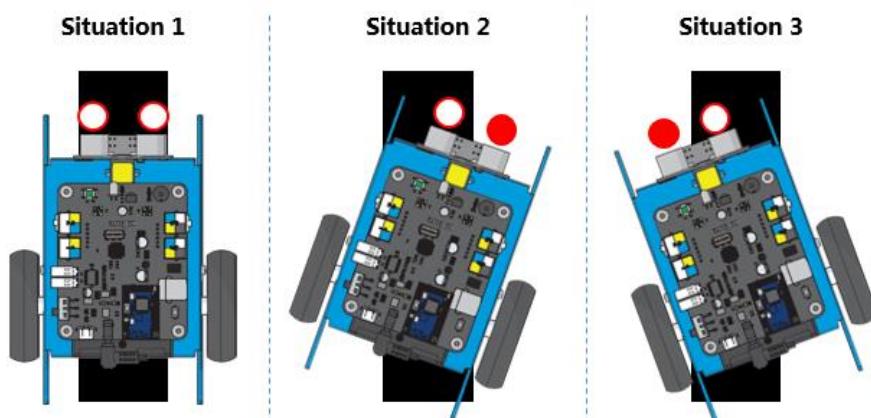


Figure 2: Robot positions during the line following process

Required task: Write a program for the robot to follow the line using the simple algorithm mentioned above.



B. Line following based on a proportional controller

In this algorithm, the turning level is chosen to be proportional to the deviation of the robot from the line. Let x^* be the line position and x be the current robot position. The error or the difference between the robot position and its expected position (which is the line position) is:

$$e = x - x^* \quad (1)$$

The turning level δ then can be computed based on the error as:

$$\delta = K_p * e \quad (2)$$

where K_p is a constant gain. The choice of K_p is dependant to the robot hardware and is often carried out based on trials. The calculation of error is based on the sensor values as illustrated in Figure 3.

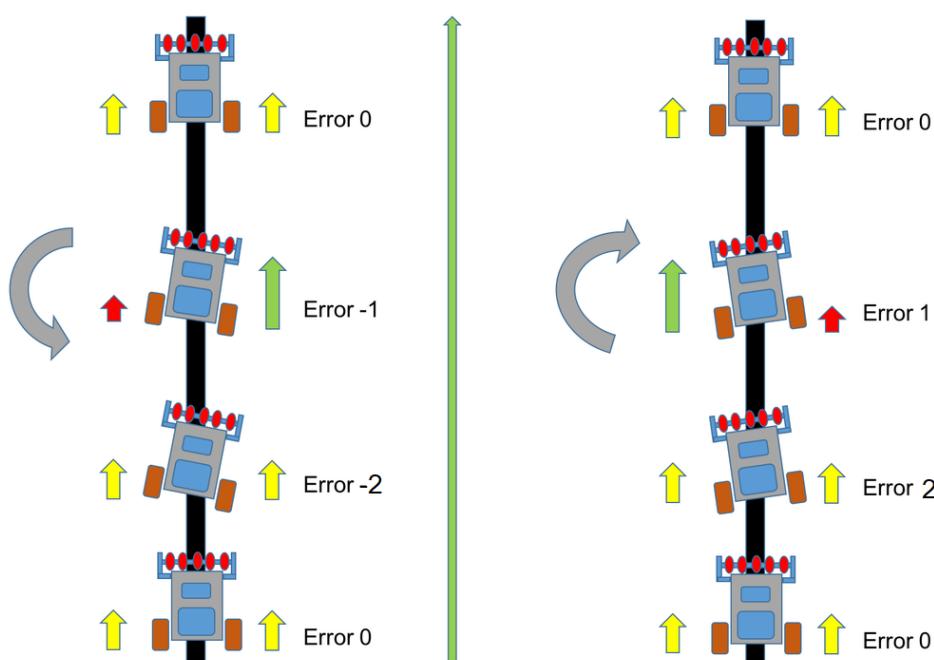


Figure 3: Calculation of error values

The value of δ is then used to adjust the turning speed. For the differential drive robots, δ can be used to adjust the speed of the left and right wheels as follows:

```
delta = Kp * error  
speedL = baseSpeedL + delta  
speedR = baseSpeedR + delta
```

Required task: Write a program for the robot to follow the line using the proportional algorithm mentioned above.

C. Line following based on the PID controller

This algorithm is based on a popular technique in control named Proportional-Integral-Derivative (PID) control. Despite advancement in control theory, PID controllers remain the most popular control technique being used in applications due to its effectiveness and simplicity. In PID control, the control



Intelligent Robot Studio

From Theory to Practice

signal is generated from three components, Proportional, Integral and Derivative. Specifically, let e be the error position defined in equation (1), K_p be the proportional gain, K_i be the integral gain, and K_d be the derivative gain. The turning level δ is computed as follows:

$$\delta = K_p e + K_i \int_0^t e d\tau + K_d \frac{de}{dt} \quad (3)$$

(Don't panic! I just write it down to give you the feeling that you are doing something cool). Since our system is digital (discrete time), the integral becomes the sum and the derivative becomes the difference. Equation (3) can be rewritten as:

$$\delta = K_p e + K_i \sum e + K_d (e - e_{pre}) \quad (4)$$

where $\sum e$ is the sum of the errors over time and e_{pre} is the previous error. Note that K_p, K_i, K_d are constant gains and should be tuned based on trials and errors for now (we will calculate them in another course on control). The meaning of the P, I, and D terms are as follows:

- The proportional term measures "how far" the robot is away from the line
- The integral term sums error to determine "how long" the robot has been away from the line
- The derivative term assesses "how fast" error in the process is changing

Properly combining those terms will allow the robot to better adapt to changes during the line following process. However, depending on the robot and sensor used, a certain term can be omitted, and the controller can be simplified to a PI or PD controller.

A sample code for this algorithm is as follows:

```
I = I + error
D = error - lastError
gain = Kp*error + Ki*I + Kd*D
speedL = baseSpeedL + gain
speedR = baseSpeedR + gain
if (speedL > maxSpeedL):
    speedL = maxSpeedL
if (speedL < minSpeedL):
    speedL = minSpeedL
if (speedR > maxSpeedR):
    speedR = maxSpeedR
if (speedR < minSpeedR):
    speedR = minSpeedR
lastError = error
```

Required task: Write a program for the robot to follow the line using the PID algorithm.

Want to read more about PID?



Intelligent Robot Studio

From Theory to Practice

- A tutorial on line following using the PID algorithm:
<https://create.arduino.cc/projecthub/anova9347/line-follower-robot-with-pid-controller-cdedbd>
- PID controller: https://en.wikipedia.org/wiki/PID_controller