Chapter 3
Reactive Behavior

We are now ready to write our first algorithms for robots. These algorithms demonstrate *reactive behavior*: an event (such as the detection of a nearby object by the robot) causes the robot to react by performing an action that changes its behavior (such as stopping the motors). Purely reactive behavior occurs when the action is related only to the occurrence of an event and does not depend on data stored in memory (state).

The reactive behaviors are those of *Braitenberg vehicles* which are appropriate for introducing robotics because complex behavior arises from simple algorithms. Sections 3.1–3.3 describe Braitenberg vehicles that demonstrate reactive behavior; in Chap. 4 we present Braitenberg vehicles that have non-reactive behavior with states. Section 3.4 presents several algorithms for the classic reactive behavior of line following. Line following is an interesting task because the algorithms are sensitive to the characteristics of the sensors and the lines. Calibration is necessary to determine the optimum thresholds for fast robust motion of the robot. Section 3.5 gives a brief overview of Braitenberg's original formulation of the vehicles in a biological approach with sensors connected directly to the motors, not through a computer. Later in the book (Sect. 13.3) we discuss the implementation of Braitenberg vehicles using neural networks.

3.1 Braitenberg Vehicles

Valentino Braitenberg was a neuroscientist who described the design of virtual vehicles that exhibited surprisingly complex behavior. Researchers at the MIT Media Lab developed hardware implementations of the vehicles from *programmable bricks* that

M. Ben-Ari and F. Mondada, *Elements of Robotics*,
https://doi.org/10.1007/978-3-319-62533-1_3

were the forerunner of the LEGO® Mindstorms robotics kits.[1] This chapter describes an implementation of most of the Braitenberg vehicles from the MIT report. The MIT hardware used light and touch sensors, while our generic robot is based upon horizontal proximity sensors.

To facilitate comparison with the MIT report (and indirectly with Braitenberg's book), the names of their vehicles have been retained, even though it may be difficult to understand their meanings in our implementations.

Two vehicles are presented in detail by giving all of the following:

- The specification of the behavior of the robot;
- A formalized algorithm for the specified behavior;
- An activity that asks you to implement the algorithm on your robot.

The other vehicles are presented in activities that specify the behavior and ask you to develop an algorithm and to implement it on your robot.

3.2 Reacting to the Detection of an Object

Specification (Timid): When the robot does not detect an object, it moves forwards. When it detects an object, it stops.

Algorithm 3.1 implements this behavior.

Algorithm 3.1: Timid

```
1: when object not detected in front
2:     left-motor-power ← 100
3:     right-motor-power ← 100
4:
5: when object detected in front
6:     left-motor-power ← 0
7:     right-motor-power ← 0
```

The algorithm uses two event handlers, one for the event of detecting an object and one for the event of not detecting an object. The event handlers are written using the when statement whose meaning is:

when the event *first occurs*, perform the following actions.

Why do we use this construct and not the more familiar while statement (Algorithm 3.2)?

[1] The MIT report uses the term *Braitenberg creatures* but we retain the original term.

If we use the while statement, *as long as* the object is not detected the motors will be turned on, and *as long as* the object is detected the motors will be turned off. Since the sensor will detect the object over a range of distances, the motors will be repeatedly turned on or off. It is likely that no harm will be done if a motor that is already off is turned off and a motor that is already on is turned on, but these repeated commands are not necessary and may waste resources. Therefore, we prefer to turn the motors off only when the object is first detected and to turn them on only when the object is first not detected. The when statement gives the semantics that we want.

Algorithm 3.2: Timid with while

```
1: while object not detected in front
2:     left-motor-power ← 100
3:     right-motor-power ← 100
4:
5: while object detected in front
6:     left-motor-power ← 0
7:     right-motor-power ← 0
```

Activity 3.1: Timid

- Implement the Timid behavior.

Activity 3.2: Indecisive

- Implement the Indecisive behavior.

  Specification (Indecisive): When the robot does not detect an object, it moves forwards. When it detects an object, it moves backwards.

- At just the right distance, the robot will *oscillate*, that is, it will move forwards and backwards in quick succession. Measure this distance for your robot and for objects of different reflectivity.

Activity 3.3: Dogged

- Implement the Dogged behavior.

  Specification (Dogged): When the robot detects an object in front, it moves backwards. When the robot detects an object in back, it moves forwards.

Activity 3.4: Dogged (stop)

- Implement the Dogged (stop) behavior.

  Specification (Dogged (stop)): As in Activity 3.3, but when an object is not detected, the robot stops.

Activity 3.5: Attractive and repulsive

- Implement the Attractive and repulsive behavior.

  Specification (Attractive and repulsive): When an object approaches the robot from behind, the robot runs away until it is out of range.

3.3 Reacting and Turning

A car turns by changing the angle of its front wheels relative to the frame of the vehicle. The motor power is not changed. A robot with differential drive has no steering mechanism (like the steering wheel of a car or the handlebars of a bicycle). Instead, it turns by setting different levels of power to the left and right wheels. If one wheel turns faster than the other, the robot turns in the direction opposite that of the faster wheel (Fig. 3.1a). If one wheel turns backwards while the other turns forwards, the turn is much sharper (Fig. 3.1b). In the figures, the arrows denote the direction and speed of each wheel. The *turning radius* is the radius of the circle that is the path of the robot. We say that a turn is tighter if the radius is smaller. At the extreme, if one wheel turns forwards and the second turns backwards at the same speed, the robot will turn in place and the turning radius is zero.

Let us now implement a Braitenberg vehicle whose specification requires the robot to turn.

Specification (Paranoid): When the robot detects an object, it moves forwards (colliding with the object). When it does not detect an object, it turns to the left.

Algorithm 3.3 implements this behavior.

Algorithm 3.3: Paranoid

```
1: when object detected in front
2:     left-motor-power ← 100
3:     right-motor-power ← 100
4:
5: when object not detected in front
6:     left-motor-power ← −50        // Left motor backwards
7:     right-motor-power ← 50        // Right motor forwards
```

Activity 3.6: Paranoid

- Implement the Paranoid behavior.
- In the algorithm, the left and right motors are set to equal but opposite powers. Experiment with these power levels to see their influence on the turning radius of the robot.

Activity 3.7: Paranoid (right-left)

- Implement the Paranoid (right-left) behavior.

  Specification (Paranoid (right-left)): When an object is detected in front of the robot, the robot moves forwards. When an object is detected to the right of the robot, the robot turns right. When an object is detected to the left of the robot, the robot turns left. When no object is detected the robot does not move.

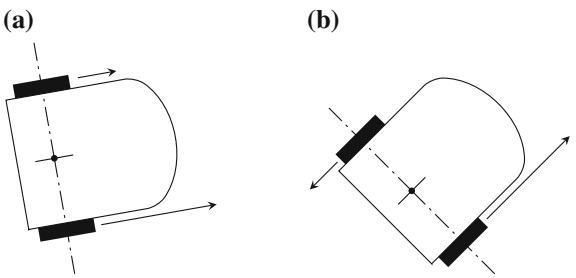

Fig. 3.1 **a** Gentle left turn. **b** Sharp left turn

Activity 3.8: Insecure

- Implement the Insecure behavior.

  Specification (Insecure): If an object is not detected to the left of the robot, set the right motor to rotate forwards and set the left motor off. If an object is detected to the left of the robot, set the right motor off and set the left motor to rotate forwards.

- Experiment with the motor settings until the robot follows a wall to its left.

Activity 3.9: Driven

- Implement the Driven behavior.

  Specification (Driven): If an object is detected to the left of the robot, set the right motor to rotate forwards and set the left motor off. If an object is detected to the right of the robot, set the right motor off and set the left motor to rotate forwards.

- Experiment with the motor settings until the robot approaches the object in a zigzag.

3.4 Line Following

Consider a warehouse with robotic carts that bring objects to a central dispatching area (Fig. 3.2). Lines are painted on the floor of the warehouse and the robot is instructed to follow the lines until it reaches the storage bin of the desired object.

Line following is a task that brings out all the uncertainty of constructing robots in the real world. The line might not be perfectly straight, dust may obscure part of the line, or dirt may cause one wheel to move more slowly than the other. To follow

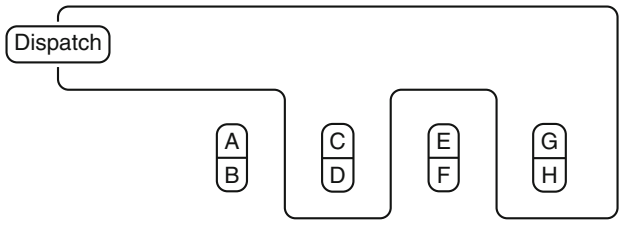

Fig. 3.2 A robotic warehouse

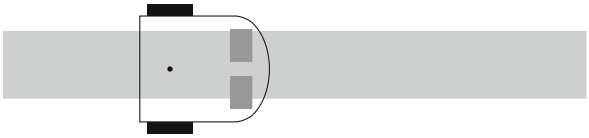

Fig. 3.3 A robot with two ground sensors over a line

Fig. 3.4 Leaving the line

a line, the robot must decide whether it is on the line or not, and if it starts to leave the line on one side, it must turn in the correct direction to regain the line.

3.4.1 Line Following with a Pair of Ground Sensors

To follow a line, a pair of ground sensors can be used (Fig. 3.3).[2] A ground sensor on a light-colored floor will detect a lot of reflected light. If a dark line is painted on the floor, the sensor will detect very little reflected light when it is over the line.[3] The line should be black for increased contrast with the white floor, but the figure displays the line in light gray so as not to obscure the robot and its sensors. Thresholds are used to determine when the event occurs of a sensor moving from detecting the line to detecting the floor or conversely.

The line must be wide enough so that both ground sensors will sense dark when the robot is directly above the line. The sensors do not have to be entirely over the line; it is sufficient that the amount of light reflected from line onto the sensor be below the threshold defined for black.

To implement line-following, the robot must move forward whenever both sensors detect a dark surface, indicating that it is on the line. If the robot starts to leave the line, either the left or the right ground sensor will leave the line first (Fig. 3.4):

- If the robot moves off the line to the *left*, the *left* sensor will not detect the line while the *right* sensor is still detecting it; the robot must turn to the *right*.
- If the robot moves off the line to the *right*, the *right* sensor will not detect the line while the *left* sensor is still detecting it; the robot must turn to the *left*.

[2]The figure shows a top view although the ground sensors are on the bottom of the robot.

[3]Our presentation of the line following algorithms assumes that the floor is light-colored. If your floor is dark-colored, a white line should be used.

For now, we specify that the robot stops whenever neither sensor detects the line.

Algorithm 3.4 formalizes the above informal description.

Algorithm 3.4: Line following with two sensors

```
when both sensors detect black
    left-motor-power ← 100
    right-motor-power ← 100

when neither sensor detects black
    left-motor-power ← 0
    right-motor-power ← 0

when only the left sensor detects black
    left-motor-power ← 0
    right-motor-power ← 50

when only the right sensor detects black
    left-motor-power ← 50
    right-motor-power ← 0
```

Activity 3.10: Line following with two sensors

- Implement Algorithm 3.4.
- Use black electrician's tape or gaffer tape to create a line on the floor. (Gaffer tape is used on stage and film sets to bind cables or fix them to the floor. Gaffer tape is usually less reflective than electrician's tape and thus a better choice for implementing line-following algorithms.)
- The line should have angles or curves which will cause the robot to start running off the line. Run the program and check that the robot can successfully follow the line with its angles and curves.
- Experiment with motor powers for turning back onto the line. If the turn is too gentle, the other sensor might also run off the line before the robot turns back. If the turn is too sharp, it might cause the robot to run off the other end of the line. In any case, sharp turns can be dangerous to the robot and cause whatever it is carrying to fall off.
- The forward speed of the robot on the line is also important. If it is too fast, the robot can run off the line before the turning actions can affect its direction. If the speed is too slow, no one will buy your robot to use in a warehouse. How fast can your robot follow the line without running off it?

Activity 3.11: Different line configurations

- What is the sharpest turn that your robot can follow? Can it follow a line that has a 90° turn?
- Experiment with different configurations of the line: gentle turns, sharp turns and zigzag lines.
- Experiment with the width of the line. What happens if the line is much wider or narrower than the distance between the sensors?

Activity 3.12: Regaining the line after losing it

- Modify the algorithm so that the robot returns to the line after it loses it, that is, when both sensors no longer detect black.
- Algorithm 1: Before both sensors no longer detect the line, one of them will be the first that no longer detects the line. Use a variable to remember the sensor that first lost the line; when the line is lost, turn in the opposite direction of the value of this variable.
- Algorithm 2: The previous algorithm might not work if the robot is moving too fast and runs off the line before it detects that only one sensor lost the line. Instead, if both sensors no longer detect black, cause the robot the search for the line for a short distance, first in one direction and then in the other direction.

Activity 3.13: Sensor configuration

- Discuss what effect the following modifications to the robot would have on its ability to follow a line:
  - Ground sensing events occur more often or less often.
  - The sensors are further apart or closer together.
  - There are more than two ground sensors on the bottom of the robot.
  - The sensors are on the back of the robot or the robot moves backwards.
- Experiment with these changes if they can be done on your robot.

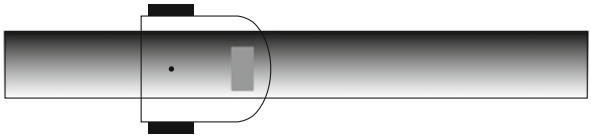

Fig. 3.5 A line with a grayscale gradient

3.4.2 Line Following with only One Ground Sensor

A robot can follow a line with only one ground sensor if the reflectivity of the line varies across its width. Figure 3.5 shows a grayscale line whose shade varies continuously from fully black to fully white across its width. The ground sensor will return values from 0 to 100 depending on which part of the line it is over.

When the robot is directly over the center of the line, the sensor will return the value 50 that is midway between black and white. Of course, we don't expect the value to be exactly 50, so there is no reason for the robot to turn left or right unless the value approaches 0 or 100. We define two thresholds:

- black_threshold: below this value, the robot is leaving the left side of the line.
- white_threshold: above this value, the robot is leaving the right side of the line.

Algorithm 3.5 changes the motor power settings when the value returned by the sensor crosses the thresholds.

Algorithm 3.5: Line following with one sensor

```
integer black-threshold ← 20
integer white-threshold ← 80
when black-threshold ≤ sensor value ≤ white-threshold
    left-motor-power ← 100
    right-motor-power ← 100

when sensor value > white-threshold
    left-motor-power ← −50
    right-motor-power ← 50

when black-threshold < sensor value
    left-motor-power ← 50
    right-motor-power ← −50
```

Activity 3.14: Line following with one sensor

- Implement Algorithm 3.5.
- Experiment with the thresholds until the robot can follow the line.
- Modify the algorithm so that the robot detects when it has run completely off the line. Hint: The only problem is when the robot runs off the black side of the line, because there the algorithm doesn't distinguish between that case and running off the white side of the line. One solution is to use a variable to remember the previous value of the sensor, so that the two cases can be distinguished.

Activity 3.15: Line following with proportional correction

- Since the sensor values are proportional to the grayscale of the line, the approximate distance of the sensor from the center of the line can be computed. Modify the algorithm to compute this distance.
- Modify the algorithm so that the motor setting is proportional to this distance.
- Run experiments for various constants of proportion and explain the results.

3.4.3 Line Following Without a Gradient

The receiver of a proximity sensor has an *aperture*, an opening through which the light is collected. Apertures are often wide in order to allow more light to fall on the sensor so that it will be responsive to low levels of light. Cameras have *f-stops*: the lower the value of the f-stop, the wider the aperture, so pictures can be taken in relatively dark environments. If the aperture of the ground proximity sensor is relatively wide, a gradient on the line is not needed. A single sensor can follow one *edge* of a line (Fig. 3.6). The figure assumes that the robot is supposed to follow the right edge of the line.

- Left image: If the sensor of the robot is over the line, little light will be sensed, so the robot is too far to the left of the right edge that it should be following. The robot must turn right.
- Center image: If the sensor is off the line, a lot of light will be sensed, so the robot is too far to the right of the right edge that it should be following. The robot must turn left.
- Right image: If the sensor is over the right edge of the line (as it should be), the amount of light sensed will be midway between the two extreme values. The robot can continue to move forwards.

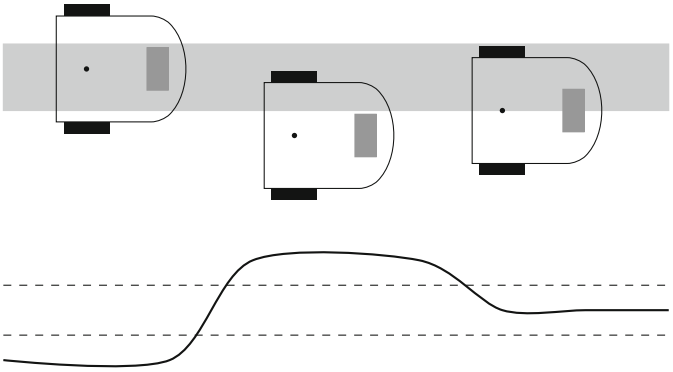

Fig. 3.6 Line following with a single sensor and without a gradient. *Above*: robot moving over the line, *below*: plot of sensor value vs. distance

At the bottom of the figure is a plot of the values returned by the sensors. The dashed lines are the thresholds between the three states: on the line, off the line, on the edge.

Activity 3.16: Line following without a gradient

- Write the algorithm in detail.
- Experiment to determine the thresholds.
- Implement the algorithm.
- Compare the performance of the algorithm with the line following algorithms using two sensors and one sensor with a gradient. Which algorithm is more *robust*, that is, which algorithm performs better at higher speeds and is able to follow sharper turns of the line?

Activity 3.17: Line following in practice

- Our presentation of line following is based on the use of sensors that detect a line painted or taped to the floor. What other technologies could be used to portray the line and to detect it?
- The algorithms cause the robot to follow the line but for a warehouse robot there needs to be a way of identifying when the robot has reached the required bin and a way of locating a specific item in the bin. How can these tasks be implemented?
- Suppose that the warehouse adds new bins. What changes need to be made to the robot? How could the algorithms be designed to facilitate change?

3.5 Braitenberg's Presentation of the Vehicles

Valentino Braitenberg's vehicles were constructed as thought experiments not intended for implementation with electronic components or in software. The vehicles had sensors directly connected to the motors as in the nervous system of living creatures. Some vehicles were designed with memory similar to a brain.

Figures 3.7a–b show robots that demonstrate Braitenberg's presentation. They have light sensors (the semicircles at the front of the robots) that are connected directly to the motors of the wheels. The more light detected, the faster each wheel will turn, as indicated by the + signs on the connections. If a strong light source is directly ahead of the robot, both sensors will return the same value and the robot will move rapidly forwards. Suppose now that the light source is off to the left. For the robot in Fig. 3.7a, the left wheel will turn rapidly and the right wheel will turn slowly. This results in the robot turning sharply right away from the light source. Braitenberg called this vehicle *coward*. For the robot in Fig. 3.7b, the right wheel turns rapidly and the left wheel turns slowly so the robot turns towards the light source, eventually crashing into it. This behavior is *aggressive*.

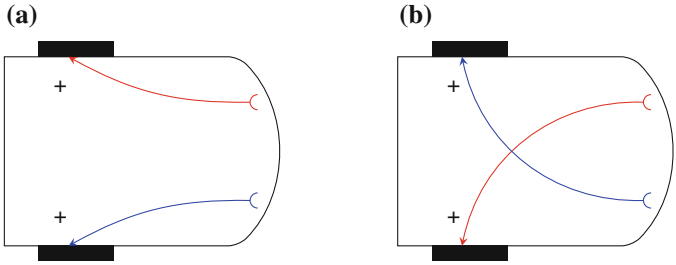

Fig. 3.7 **a** Coward vehicle **b** Aggressive vehicle

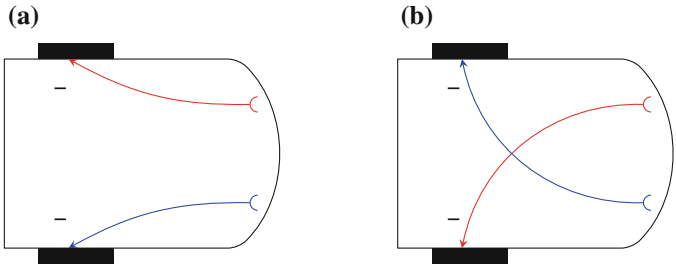

Fig. 3.8 **a** Loves vehicle **b** Explorer vehicle

Activity 3.18: Braitenberg's presentation of the vehicles

- Implement the *coward* and *aggressive* vehicles.
- Use proximity sensors in place of Braitenberg's light sensors and the detection or non-detection of an object in place of stronger or weaker light sources.
- The robots in Fig. 3.8a–b are the same as the robots in Fig. 3.7a–b, respectively, except that the values of the sensors are negated (the − signs on the connections): the more light detected, the slower the wheel will turn. Assume that there is a fixed bias applied to the motors so that the wheels turn forwards when no light source is detected.
- Implement the robot in Fig. 3.8a. Why is it called *loves*?
- Implement the robot in Fig. 3.8b. Why is it called *explorer*?

3.6 Summary

A robot exhibits reactive behavior when its actions depend only upon the current values returned by its sensors. This chapter presented two families of reactive behavior. Braitenberg vehicles implement reactive behavior by changing the setting of the motors in response to the events of proximity sensors detecting or not detecting an object at some position relative to the robot. The vehicles demonstrate that complex behavior can result from relatively simple reactive algorithms.

Line following is a fundamental task in robotics. Because of uncertainties of the robot's motion and its environment, robots use landmarks such as lines to ensure that they move to their intended destination. Line following is a reactive behavior because the robot modifies its behavior in response to values returned by the ground sensors. We have given three configurations for the robot and the line, and developed algorithms for each case. The performance of the algorithms depends on the sensor thresholds and the motor speeds, which must be determined by experimentation in order to ensure that the robot moves rapidly while remaining robust to changes in the environment.

3.7 Further Reading

Braitenberg's book [1] is interesting because he writes from the perspective of a neuroscientist. The book describes vehicles using imagined technology, but they are nevertheless thought-provoking. The Braitenberg vehicles described here are adapted from the hardware implementations described in [2]. An implementation of the Braitenberg vehicles in Scratch by the first author can be found at: https://scratch.mit.edu/studios/1452106.

References

1. Braitenberg, V.: Vehicles: Experiments in Synthetic Psychology. MIT Press, Cambridge (1984)
2. Hogg, D.W., Martin, F., Resnick, M.: Braitenberg creatures. Technical report E&L Memo No. 13, MIT Media Lab (1991). http://cosmo.nyu.edu/hogg/lego/braitenberg_vehicles.pdf