Development of an agent for a robot that follows black/white transition

Pedro Santos (76532), Beatriz Coronha (92210), and Ruben Kins (92078)

Universidade de Aveiro, Aveiro, Portugal

Abstract. This document summarizes the development of a software project that allows a robot to navigate a course consisting of simple black and white shapes on a table. Additional challenges that the robot also needs to master for a successful completion of the course have been included in the given task. The project was a success and the robot could master all the challenges given in the example course.

Keywords: Agent · robots · c ·

1 Introduction

This report will explain the reasoning and the overall architecture used in the development of an agent for a robot that follows black/white transitions. Besides the development of the navigation that consists in following black/white transitions there are other challenges that were given. In the next section each one of the challenges will be explained. In the end the results will be presented as well as some of the things that could be improved.

2 Project Development

In this section our solution for each one of the challenges will be presented. The developed code and solution was designed to be the as scalable and elegant as possible. The list of the challenges that the robot does are:

1. Simple navigation- The robot is positioned in front of the start line, it does two laps and in the last one it rotates 180 degrees and performs another lap.
2. Navigation challenge 1 with sharp edge- This challenge consists in the robot be able to detect a sharp edge and will continue to follow the black/white transition.
3. Navigation challenge 2 with obstacle- The robot will detected the object and avoid the object and then it will continue to do the circuit.
4. Navigation in challenge 3- The robot will detect the circle in the ground and between the two circle it will turn on the leds and also will reduce to a slower and constant speed. Another thing that the robot will do is that during the last lap the start line is taken and then the robot stops as close as possible where the start line was located.

One thing that must be addressed before any more in depth explanation is that there is a main loop and in each cycle of the loop the robot has to make a decision based on the reading of the sensors. So in every cycle - in accordance with the given challenges - the robot will:

1. read the value of the sensors;
2. check for the start line;
3. check for the circles on the ground;
4. check for any obstacle;
5. make the decision of direction if there is no obstacle;
6. deal with the obstacle if it found an obstacle;

Another thing which is implemented is that each time values from the sensors are read, the values are read multiple times in a loop. If the value will be used in a threshold this loop serves to count the number of times that the sensor read above or below the threshold depending on the case. Then, for example if the cycle repeats 10 times, the value of the counter has to be bigger than 7. This way, the error produced by unstable sensor readings is diminished.

2.1 Navigation

For the navigation, it was used the ground sensors to detect the position of the robot with regards to the transition. For that, it was used five bits (one for each sensor) to know if the robot is on the black or white part of the path and, since the sensors detect reflection of light, if the bit is equal to 1 then it is on the black, if it is equal to 0 it is on the white. We also used the sensors to correct the position of the robot in case it starts to leave the transition. If the three bits from the left are 0 (white) and the other two are 1 (black), then the robot needs to turn a little to the side that is black, so it becomes more centered and, if only the bit from the far left is 1 (black) and the others are right, that means that the robot must take a more significant turn to the side that is black. Both cases work the same way for navigating to the other direction as well. If all the bits are the same, it means that the robot found a sharp edge. This case will be explained in another section.

In other words, there are six unique options for taking an action. The first option is what happens in figure 1. When the sensor on the left of the arrow is black and the other ones are white it is necessary to turn left. When the sensors on the left of the arrow as shown in figure 2 detect black and the others one are detecting a white surface it is necessary to turn only slightly right. The situations shown in figure 3 and figure 4 are analogous to the previous two, but in the other direction. The second last situation happens when all the sensors are detecting the same white (or black) surface which means that the robot is detecting a sharp edge. The last situation is the default situation in which he robot goes forward in a straight manner. The default situation is always chosen if the sensor readings don't match one of the previous situations. The algorithm is implemented in a way that it also works with inverted colours so it is independent of the direction the robot is going to (black edge on the left or the right side: does not matter).

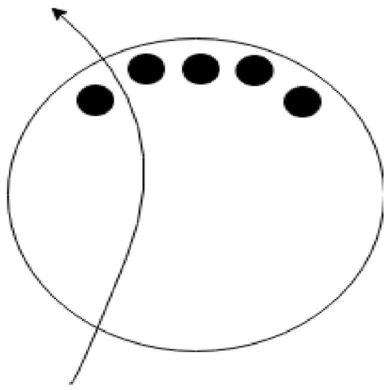

Fig. 1. One sensor detecting black surface - turn left

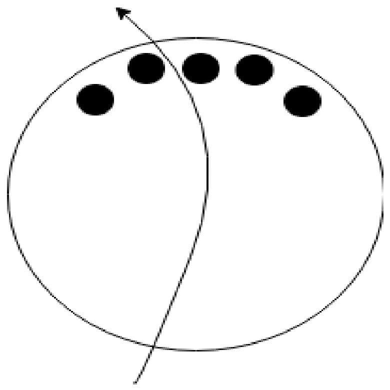

Fig. 2. Two sensor detecting black surface - turn slightly left

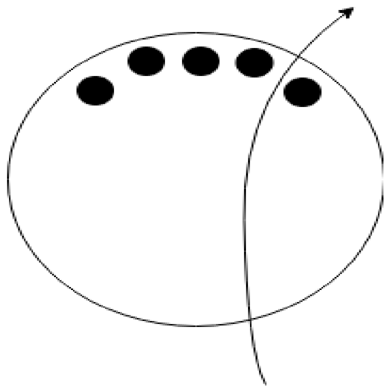

Fig. 3. One sensor detecting black surface - turn right

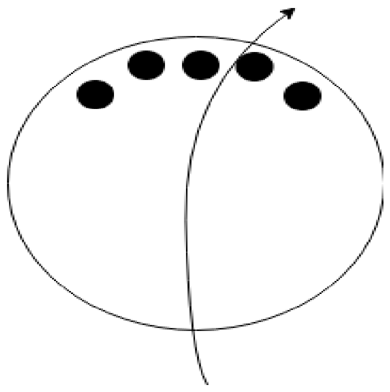

Fig. 4. Two sensor detecting black surface - turn slightly right

2.2 Simple navigation

The simple navigation is the basic task without any additional challenges. The robot simply follows the rules established in the paragraph above so it does not get off the track and completes the round. The simple navigation mode can be overwritten by all the other program parts which means that all the other challenges have priority over it. Only if no special case is currently being detected the robot can execute its basic navigation. This makes sense because the challenges are unique situations that require special actions. Following the black/white transition is the default state.

2.3 Navigation challenge 1: Transition with a sharp edge

In this navigation challenge the robot must be able to detect a sharp edge but continue to follow the black/white transition. The algorithm described in the section above can only deal with moderate changes of direction. When it loses track of the transition it does not know where to go. To solve this problem the robot remembers the last sensor value before it lost track. If the sharp edge turns right, the right sensor will be the last sensor to pick up the surface (see figure 5). If the right sensor triggers last - as shown in the figure - the robot will turn right until the sensor values change, which means that it found the transition again. After that the robot can start its basic navigation.

2.4 Navigation challenge 2: Avoiding an obstacle

In this challenge the robot must detect an object placed on the track, go around it and then, after passing the object, continue following the transition. To do this the function *detectFreeRoad* was developed. This function gets the current ground sensor value passed to it as an argument. This argument is used later. In order to detect the presence of an object the information from the center sensor is

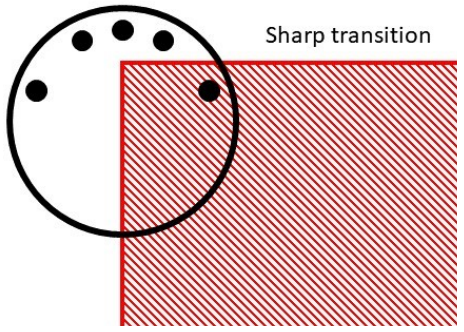

Fig. 5. Detecting a sharp edge

enough, but the other sensors can provide additional information. The function will check if there is an object under 20 cm by using the center sensor, but it will also check the left or the right sensor depending on the given argument. The direction the robot uses to avoid the obstacle depends on the direction in which the white surface is at the moment of the function call. If there is an object the function will return false. By returning false the robot will not follow the transition any further and it will start avoiding the object.

The main idea behind the solution created is that when the robot finds an object it will rotate approximately 90 degrees determined by the given ground sensor value in the argument. Then it will check the corresponding distance sensor (left or right depending on the direction it will go to). After doing this, the robot will move forward as long as the corresponding sensor detects an object. When the sensor is not detecting the object any more the robot will move forward an amount equal to the radius of the robot. This is necessary because in order to overcome an object the robot needs to also consider its own body. After this the robot will rotate again 90 degrees, but this time in the opposite direction. It will also check if there is any additional obstacle in front. If there is it will repeat the process again, however now in the next iterations the robot will not travel the radius distance, but a smaller one. The next step is to go forward an amount equal to the diameter of the robot, because in order to to overcome the object the robot has to travel at least the distance its own body takes. Then the robot will move forward until the corresponding sensor that reads the side of the object no longer detects it. The last steps are to turn 45 degrees in the direction of the white surface, to go forward until it finds the white surface and lastly to rotate in the direction that it must go. The robot will stop when the corresponding sensor is detecting the black surface so the robot is no longer detecting an object and it can proceed doing the normal navigation.

2.5 Navigation challenge 3

This challenge is divided in two parts. The first part is to detect the circles and between the two circle the robot must go at a slower constant speed and turn on the LEDs. The second part is that during the last lap the start line is taken and then the robot stops as close as possible where the start line was located. There will be a section for each part.

Circle detection To deal with this challenge a function called *checkCircle* was created. This function receives as argument that corresponds to the value read of the ground sensor and it will check if the sensor is picking up a circle. To do that the function will check the value of the ground sensor. If the value of the ground sensor is 11001, 11101, 10011 or 10111 it will return true. Both directions are being considered in these values. This happens because the robots sensor that is further away from the color black, when following the transition black/white, should detect always the color white, so when that sensor is not white it can be asserted that the robot detected a circle. After this the return value is checked and if it is true, it will create a Boolean variable that is inverted every time a circle gets detected. Then, if the robot was not inside a slow zone, the LEDs will be turned on with the appropriate given function and the speed is set to the controlled slow speed. If the robot was inside a slower zone the LEDs are turned off and the speed is set to the normal. The remaining problem is that the robot will detect a circle multiple times on every function call inside the loop because all the times that the robot would read the value from the sensors and check if the most far away sensor is black it would enter the if-condition and change the behave of the robot. To resolve this problem a variable was created that will save the last returned value before calling the function to detect the circle. And then, the if condition that verifies the current value of the function has one more condition so that it verifies whether the value of this variable is false. If this variable is true it means that the robot is detecting the same circle all the time, so only if the old reading value is false and the new one is true will enter in the if condition and change the behavior of the robot. Basically, the behavior of the robot will change in the first transition of the white to the black of the circle.

Start line detection Another sub-problem that needs to be solved is finding the location of the start line and detecting it. This is relevant because the robot needs to complete two laps, turn 180 degrees and then complete another lap in a different direction before stopping. The basic program guiding the robot through these laps is simple: Every time the robot detects that he is on the start line it has to do one of these steps:

1. First time: Continue forward for the second lap;
2. Second time: Turn around and go to the opposite direction;
3. Third time: Stop, the course has been completed;

The most difficult part is to detect the start line. In order to make the detection as reliable as possible, redundant information from different sources is used.

The resulting information about the start line makes the detection much more reliable. There are two different sources of information.

The first is the odometry of the robot. This given function provides an estimate of the robots location based on the movement of the wheels. It can give the cartesian coordinates and the rotation angle of the robot relative to its initial state. The problem with this function is that errors accumulate over time so that the estimate becomes worse. The second source is the ground sensor of the robot that detects the sudden inversion of colours on the ground. This function is very precise, but it is hard to differentiate between other transitions that are not the start line and erroneous sensor readings.

In the code there are three Boolean conditions for a successful start line detection. The first condition is that a transition gets picked up by the ground sensors. The general principle behind that has been discussed in the previous chapters. The second condition is that the position of the robot estimated by the odometry is inside a tolerance radius around the start line. This way the robot knows which transition belongs to the start line. Over time the odometry error would still accumulate so that the real start line location differs from the estimated one. That is why the start line location gets updated every time it gets detected, causing no drift over time. The third condition is a locking variable that gets set to false every time the start line gets detected. This prevents multiple detection every cycle the function gets called so the detection does not trigger every cycle while the robot is on the start line. The only way for the robot to unlock the variable is to reach a certain distance to the start line. Only after that another detection is possible. The general principle is illustrated in 6.

On the last lap the start line is removed so that the only source of information is the odometry input. That causes the detection to be more inaccurate which is acceptable because it is the last round.

3 Results

After a lot of tests and versions the results are quite good. It is possible to say that all the challenges were successfully mastered. In some tests the circle detection fails, however the environment of tests was not the best. There were a lot of people also trying to test and the circles were in different, not always ideal places all the time. Nevertheless, the circle detection worked most of the time. About the object detection, the main idea was to do a simple and efficient approach. For the obstacle detection we chose a method that made the robot able to overtake them fast but that also relies on precise distance sensor readings. Sometimes the robot collided when we used different obstacle shapes other than box shaped ones. All the tests that were done for the other challenges had a 100% success rate.

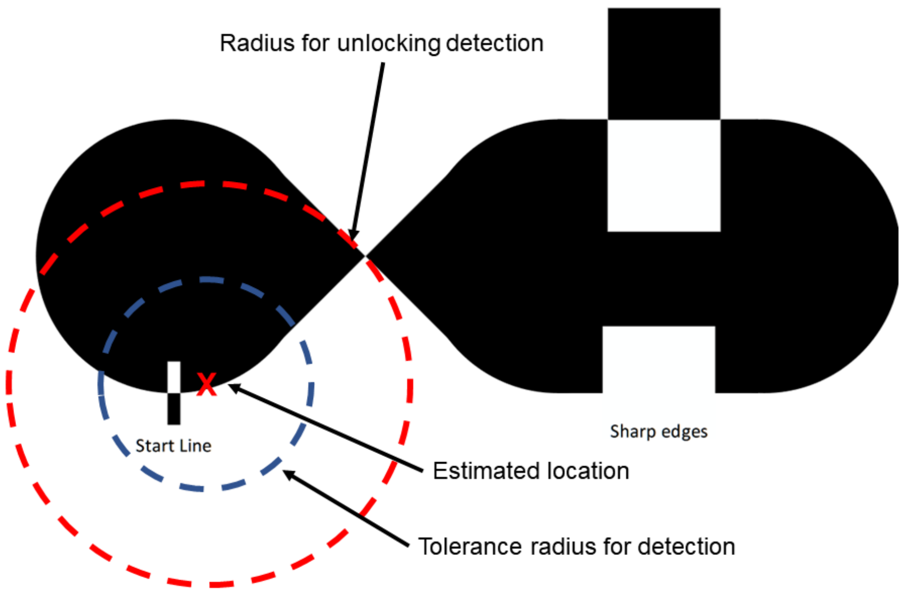
Radius for unlocking detection
Sharp edges
Start Line
Estimated location
Tolerance radius for detection

Fig. 6. Principle of the start line detection

4 Conclusion

The objectives of the project were accomplished and we learned the development mechanisms of microcontroller-based robots.