Line following using a Mighty Thymio

Elena Franchini

I. Introduction

The project consists in a Mighty Thymio which is able to automatically follow a line drawn on the ground. Different waved lines have been created to analyse the robustness of the code, starting from level 1 (smooth and large curves) to level 4 (tight curves). In addition two variations of the waved line of level 4 have been created (4.1, 4.2) to check also the behaviour of the robot with different line widths.
All these lines are used in 6 different scenes, and to make the project more interesting, an additional scene (5) with obstacles along the path has been created in order to check if the Mighty Thymio is able to deal with more complex scenarios.

II. Lines creation

Creating the lines which compose the path the Thymio has to follow has been quite complex, since there are different methods but I had to choose the one which worked better for my projects. The first idea has been to use build-in plane shapes and insert them in the scene, but there were too few modifiable parameters in order to get interesting and varied scenarios. The next idea which had more success has been to create a path made of multiple segments, where each segment started from the end of the last one, such that we get a continuous line. In addition, each segment had a bounded random orientation, such that we got a random zig-zag line at the starting of each simulation.

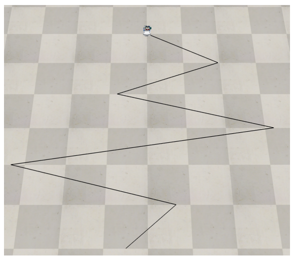


Fig. 1. Random zig-zag line

Unfortunately this idea created a lot of problems, and the main ones were:

- the sensors able to detect drawings on the ground are placed on the front-left and front-right of the Thymio, then if the robot is exactly above the line, this latter is not detected. To deal with this problem, I tried to increase the line size such that also if the thymio is exactly above the line the lateral sensors are able to detect it, but because of a fixed upper bound of the size, it has not been possible to pick an enough large size
- The Thymio struggled to follow zig-zag lines due to the tightness of the corners

The final idea has been to create waved lines and import them as textures:
I hand-created 4 waved lines of the same width using Draw.io, and each one has a specific level of complexity from the point of view of waves tightness such that we are able to analyse what are the limits of the Thymio. The robot is also able to stops when the path is finished because I used a different color to sign the end of the line (dark grey).

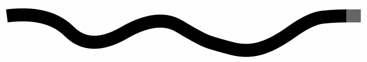


Fig. 2. Line level 1

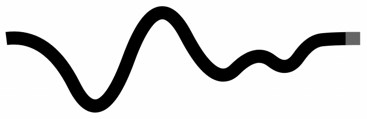


Fig. 3. Line level 2

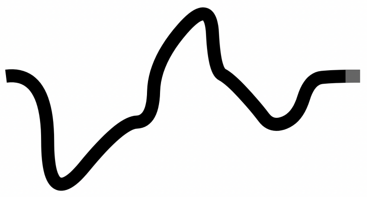


Fig. 4. Line level 3

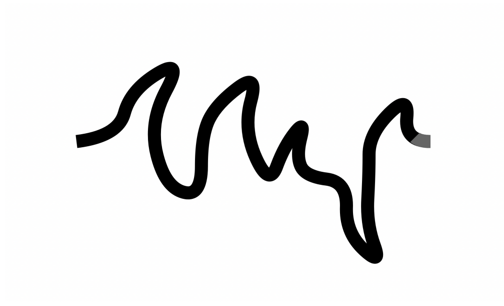


Fig. 5. Line level 4

To have a cleaner result, instead of inserting the textures manually using the GUI, I wrote a Lua child script where I created a plane shape placed above the standard plane and I attached the textures on it, setting correctly the needed parameters (for example the positions such that the lines start in front of the Thymio).
For the fifth scene, I also added three walls along the path. These walls are perpendicular to the line and they have double width (two joined walls), to make the process of avoiding the obstacles easier for the Thymio. To show the capabilities/limitations of the code, I used three walls of different length.

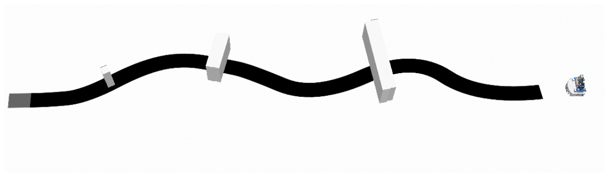


Fig. 6. Waved path with obstacles

III. Code

Finding the way in which the Thymio was able to follow a line drawn on the ground has been non-trivial. I had to enable the sensors to react to drawings and choose the correct way to collect data. It was possible to detect a line using the camera and the ground sensors. I used a single ground sensor because I found it easier than using the camera and because there was no publisher for the other two ground sensors, then it was not possible to retrieve data from them. This sensor detects drawings on the ground returning the light reflected (1023 is the returning light is very intense, otherwise another value).
At the beginning of the simulation the Thymio is always in front of the path to follow, then we simply move it forward. When one of the two sensors detect the line, I start to check the values of both of them: if the left sensor detects 1023 (no line), I add a counterclockwise rotation (negative angular velocity) to adjust the trajectory. On the other hand, if the right sensor detects 1023, I rotate clockwise the robot. If both sensors detect the line (value of ca. 120), we have only linear velocity without angular velocity. Where we have tight corners and the Thymio is right above them, both ground sensors detect a value of 1023 (because there is no line in front of the robot) and it creates problems in continuing following the path, then the solution is to stop the Thymio and enable only the angular velocity until its sensors detect again the line, and then we reactivate again the linear velocity.

The most complex part has been permitting the Thymio to avoid walls along the path. While the ground sensors are enabled permitting the Thymio to follow the line, the proximity sensors should continuously check for an obstacle, and if any, the Thymio should be able to avoid it without hitting it and then to continue following the path.
The main problem is that the left and right proximity sensors are not really on the left and on the right part of the thymio, but they're placed in the front part of the latter.

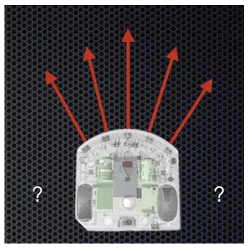
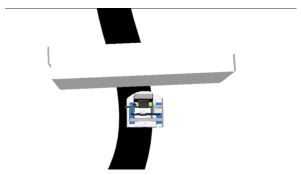


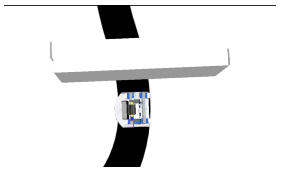
Fig. 7. Proximity sensors position

It took me a long time to find a solution since I implemented the code such that the Thymio bypasses the walls depending on the sensors measurements (such that the code doesn't work only with the specific shape of my walls but it should work in general for any type of squared obstacles). The solution is obtained following these steps:

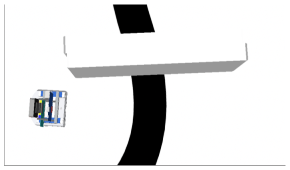
- When the front proximity sensors detect an obstacle, the Thymio stops
- The Thymio is rotated such that it is in front (orthogonal) to the obstacle



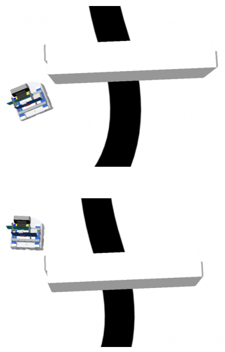
- Because the Thymio is always orthogonal to each wall, the time needed to oriented it (angular velocity) parallel to the wall is always the same. I used a timer such that when the given period is elapsed the Thymio is parallel to the wall (this is not a very clean implementation since it depends on its angular velocity, but it has been the only solution I found since using the proximity sensors resulted in a very unstable implementation)



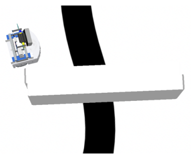
- The Thymio moves parallel along the first edge of the wall. When the right sensor doesn't detect the wall anymore, the robot continues to move for a short period of time. This is done because the right sensor is in a front-right position, then if we rotate the Thymio at the exact moment when the wall is not detected anymore and we move it straight to bypass the second edge, it hits the wall because it stops too early



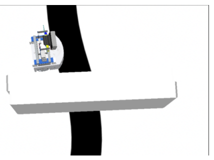
- The Thymio is rotated until the right sensor detects the wall, and it is moved along the second edge using two thresholds such that it doesn't go too close or too far away from the wall (because its trajectory is not as precise as for the first edge). When the wall is not detected anymore by the right sensor, the robot continues straight for a short period of time (same motivation of the previous point)



- The Thymio is rotated until the right sensor detects the wall



- The Thymio moves along the last edge of the wall and it stops when the one of the two ground sensors detect the line. Then it has to orientated in the correct direction following again the path: the robot is rotated left until the left ground sensor cannot detect the line anymore. At that point, the code cycle restart (the position is adjusted such that the Thymio advances and rotates right or left depending on which sensor detects the line)



- if the ground sensors detect the grey line segment, the Thymio stops because it reached the end of the line (in this case it cannot reach the end of the line since it is not able to bypass the last obstacle because of its small dimensions)

IV. Results and evaluation

The Thymio is able to follow all the four levels of lines with the same width, then the code is quite robust. To check even more its robustness, I used the scenes 4.1 and 4.2 which use the same path of the 4th scene but with smaller line width (13 and 1 pixels). This is done because as I explained above, the two ground sensors are placed on the front-left and front-right of the Thymio, then I want to test the limits of the line width.

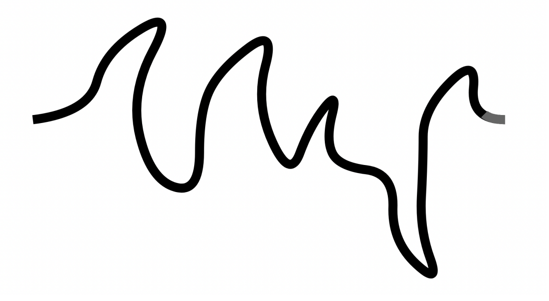


Fig. 8. Line level 4 of 13 pixels width

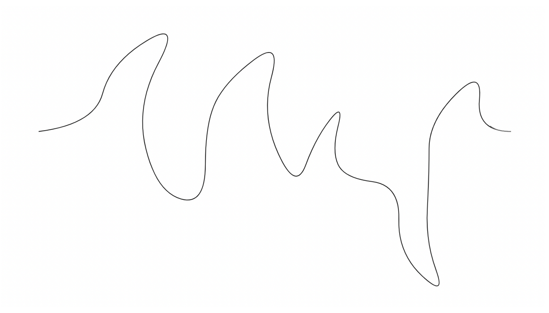


Fig. 9. Line level 4 of 1 pixel width

With a width of 13 pixels the Thymio is able to follow the line, but not with the one of 1 pixel. Such a thin line creates instability in the code without permitting the Thymio to complete the path: the Thymio movements aren't clean (it follow a zig-zag motion) and it returns back from where it comes from (because of the implementation for the detection of tight corners: the Thymio stops and rotates in the correct direction to continue following the line, but with a so thin line it cannot catch the correct direction).

Talking about the fifth scene (with the obstacles), the Thymio is able to follow the line, bypass the first two obstacles and catch again the line. It is not able however to bypass the third obstacle since it is too small, and because of this there are errors in the computations: The Thymio cannot place orthogonal to the wall since the right-front and left-front proximity sensors struggle to get the data.

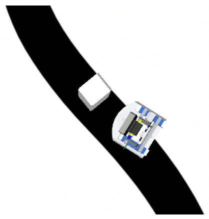


Fig. 10. Thymio unable to bypass small obstacles

V. Conclusion

With this project I have been able to use new tools of CoppeliaSim (importing images as textures, detect drawings on the ground, using the ground sensors). The Thymio has not been able to complete the path in all the scenes, but this permit to analyse the limits of the code and the sensors themself.
If the drawn lines are too thin, the Thymio is not able to follow them in the correct way. On the same way, If the obstacles are too small, the Thymio is not able to bypass them. Then as conclusion, to have a general more stable and precise result the idea is to use choose wisely the parameters sizes.