Autonomous Robot for Exploration, Search, Capture and Autonomous Robotic Navigation – Simulation in ROS and Gazebo

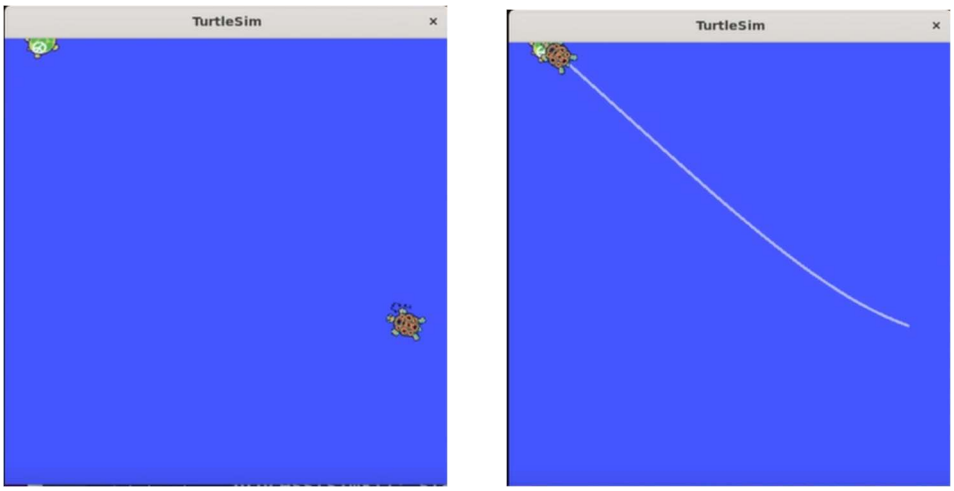
By: Zbigniew Barczyk

Contents

Part A - ROS

Task 1

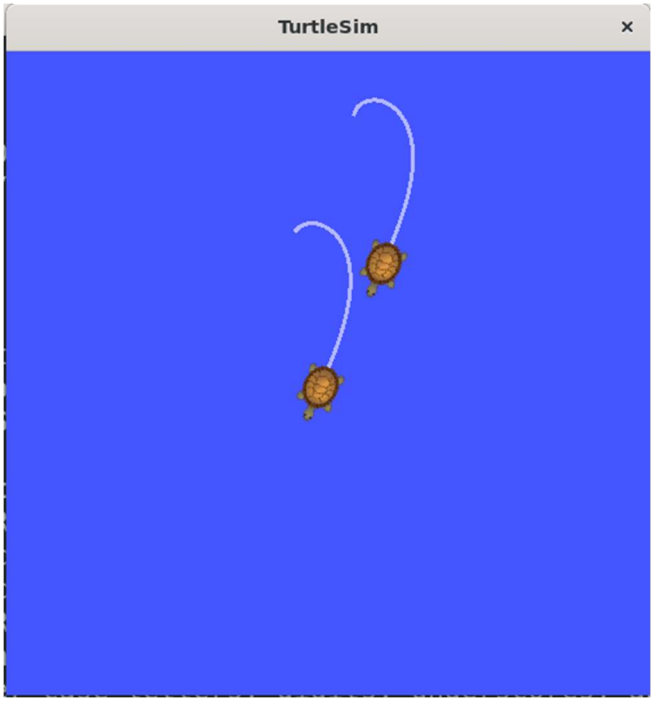
Task 1 was to spawn two turtles and make one of them capture the other. To achieve this the turtle being captured, called “mytarget” needs to share its coordinates with the other turtle “robot1” that is trying to capture it. To achieve this, we create a node and a topic called “target” where the coordinates of mytarget will be posted for robot1 to use. Then we need to write simple code for turtle movement since we know the position of the target.



Capturing the turtle – before and after

Task 2

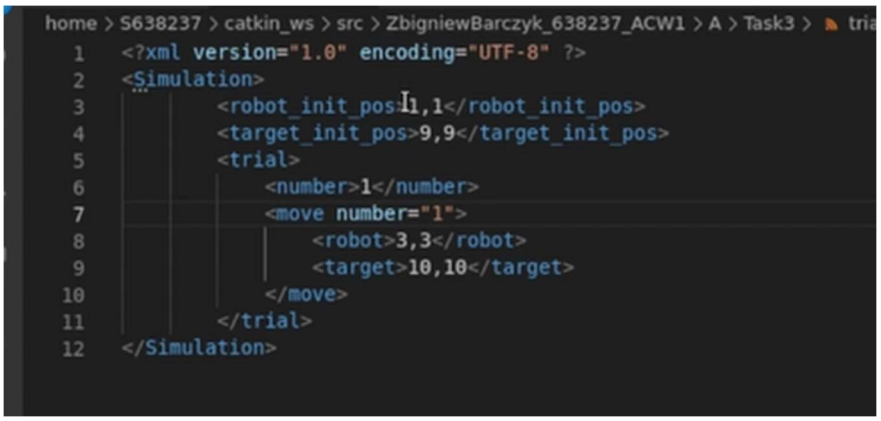
For task 2 we needed to add movement to mytarget turtle. The turtle must escape from robot1 turtle. To achieve this mytarget needs a subscriber to know where robot turtle is. This can be easily done by subscribing to the robot1/pose topic. For movement we can use the same code as for chasing the target, but reversing the angle, so that it is away from the turtle chasing it, instead of towards it.



Example of robot1 chasing mytarget

Task 3

Task 3 was more challenging as we needed to implement learning for robot1. To do this we needed to store previous trial information. For this we needed a parse and save function. The best way to store trial data was using XML files. I was able to save to a file and read from a file using the parse and save methods from the code. For movement we can easily use the code from previous tasks, however I was not able to implement the learning of robot1. I was able to save trial information to a file and load information from a file but implementing artificial intelligence was too difficult. The basic principle would be to see where mytarget was last trial and go towards that instead the current mytarget position.



Trial information data structure

Conclusion

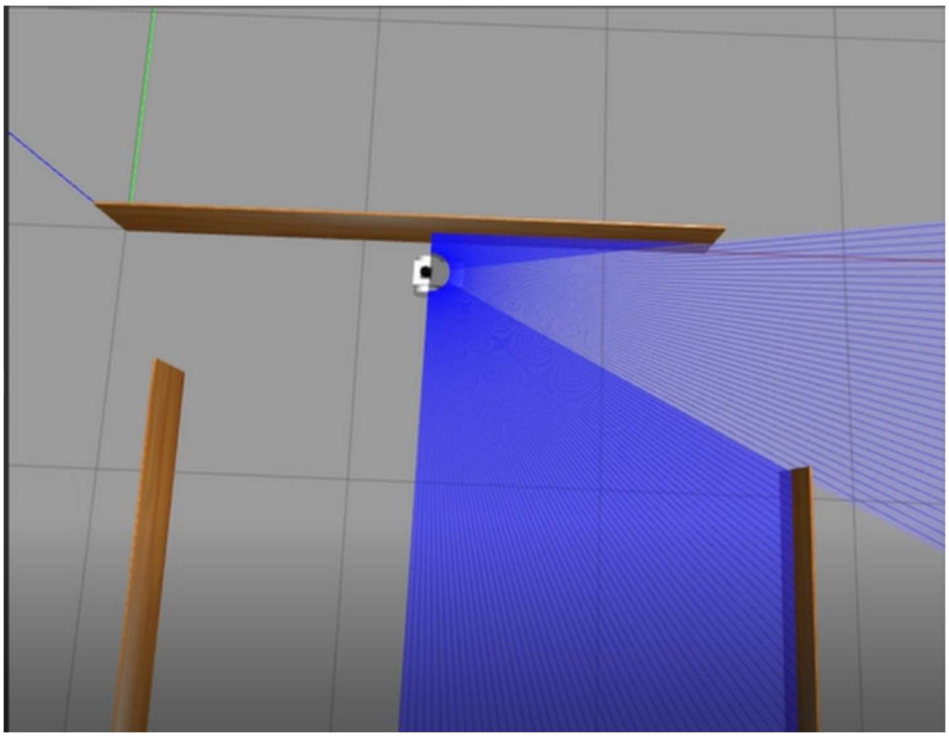
In these three tasks the algorithm for moving the turtles, especially mytarget could be greatly improved. They basically point towards the target location and move forward until they reach it. This way of solving the problem makes it hard to save the movement data to use it for teaching the robot1 to catch its target quicker. On top of that mytarget could greatly benefit from more effectively dodging the robot1. Currently it just moves away from it until it runs into the wall. It could escape for much longer if it could turn to the side before hitting a wall. I also wish I managed to use the trial data for teaching the turtle to catch its target quicker. It turned out to be much more challenging than I initially expected.

Part B - ROS and Gazebo

Task 4

In task 4 we combined ROS and Gazebo to simulate a robot. The idea was to program a robot with a laser scanner to solve a maze. First step was to construct a robot. This was done by editing a .sdf file which contained the details of the model. The robot uses the hokuyo laser scanner model. After that we needed a maze. For task 4 it was a simple maze, 4 walls. This was easily made with Gazebo build editor option. It allows to easily make a 3d room using a 2d drawing tool.

After that we needed to program the robot to solve the maze. I went with the wall follower approach. I programmed our robot to follow the left wall. The robot uses data from the hokuyo scanner. It was modified to have 180 sample areas in 180 degrees. This gives us full visibility of the obstacles ahead and next to the robot. Using the laser data, I was able to program the robot to follow a wall and avoid obstacles. It tries to go along a wall until the wall on the left is too close or too far, whish is when it turns into it or away from it. It also turns right when it finds a wall in front of it.



Robot following the wall going towards the goal

Task 5 and 6

In task 5 and 6 the maze was made more difficult first by including more walls, then by including obstacles. At first my robot control algorithm was not able to solve the maze, but after altering it a bit and changing constants for defining the turn speed I was able to make the robot solve each maze type with the same algorithm. This means the robot should be able to solve any maze with similar dimensions.

The problem with the robot is that from time to time it follow a certain path in which it manages to get itself stuck on a wall. This is not ideal. It doesn't happen often, but it is a big flaw. When it happens, the robot stops moving as it is not meant to get stuck, so it doesn't have any code for getting unstuck.

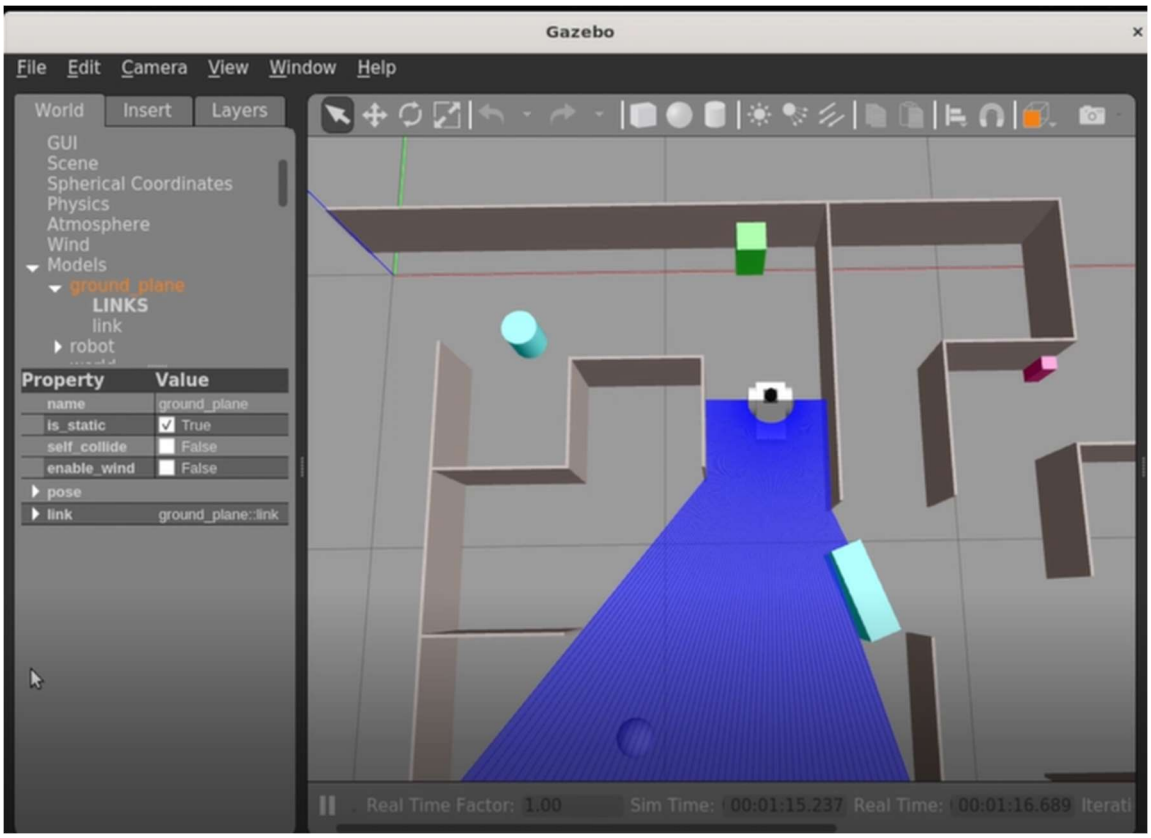
Another flaw is that it is using following the left wall approach. In this case it works great as it follows the shortest route, however in a different scenario it could follow the longest possible route to the end, which would take forever to solve. It would also allow for as many occasions for the robot to get stuck. Using a different algorithm for solving the maze would be better, since it would allow the robot to more efficiently solve different mazes.

The last flaw is the speed. The robot moves very slowly whish is because fast movement makes the scanner delay more noticeable. At low speeds we can

glance over imperfections in the control code, as a small error will not cause it to hit a wall or get stuck. Because it has enough time to correct itself. If the robot was going faster the algorithm would have to be more predicting to avoid walls and obstacles before, they are right in front of the robot.

Conclusion

Overall, the algorithm for controlling the robot is unideal, but it works perfectly fine in this scenario. Speed is not the defining factor. Since it's a simulation it can be left to run. The robot must solve the maze. Obviously faster would be better but the fact that it manages to solve the maze is enough. Even though following the left wall approach is not the most efficient in this case it is justifiable. The maze is easy enough for the approach to work just fine and the robot still almost takes the shortest path. Implementing following the left wall was much easier that other approaches and has a similar outcome.



robot solving the maze with obstacles from task 6