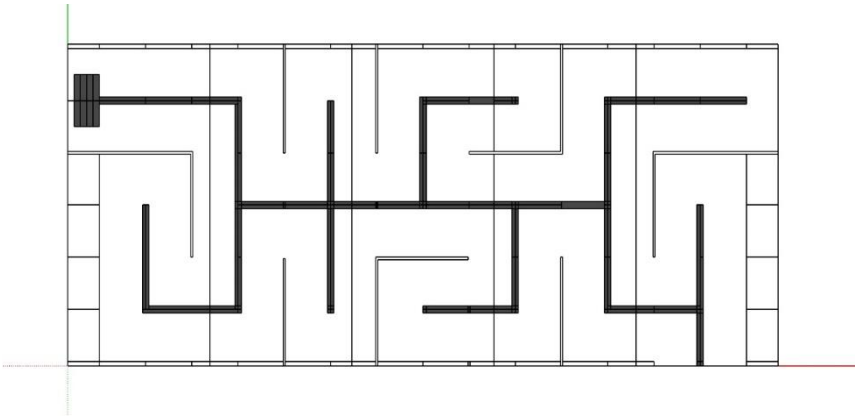
ROBOTICS 101: WORKSHOP 18.2 A MAZE-SOLVING ROBOTICS ALGORITHM

A “perfect” maze is a maze that does not have any loops. The illustration below shows a perfect maze which is suitable for demonstrating a very basic form of machine leaning.



Machine learning is pretty advanced, in fact at most universities machine learning [and artificial intelligence] is usually only studied from third year onwards because the first two years are spent learning more basic programming and control concepts to prepare oneself for greater challenges.

In the ROBOTICS 101 series of mechatronics training handbooks, machine learning is covered in Handbook II because Handbook I equips learners with the more basic skills required to understand the operation of the robot and prepare for more complex tasks.

The first step towards creating your own demonstration of basic machine learning, is to get your robot to follow a line with 45 degree turns. The next step is to program your robot to follow a line with 90 degree turns. To follow a line with 90 deg turns the robot must figure out which way to turn when it arrives at an intersection, in order to turn the correct way to continue following the line.

In the illustrated maze above there are some additional features, which were not present in the maze with 90 degree turns only, namely intersections with paths that lead into dead ends.

An intersection is here defined as a junction where there are one or more choices, like for example a four way stop where the robot could turn left or right, or continue straight. The T-junction is an intersection because it allows the possibility of the robot taking a turn to the left, or alternatively to the right. A left turn or right turn is not defined as an intersection, however, because at a left turn the robot must turn left, at a right turn the robot must turn right, and a U-turn requires the robot to turn around and head back.

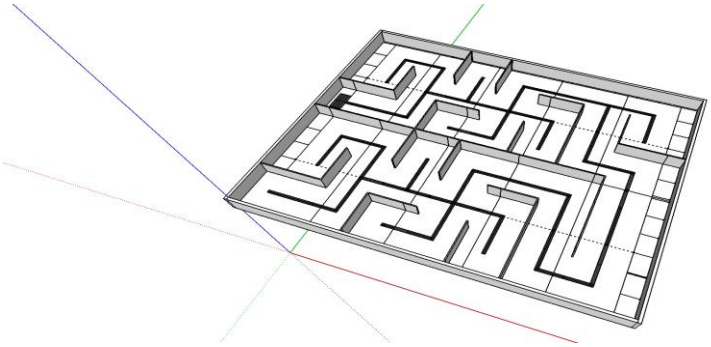
For the robot to get through the perfect maze above it needs to make some decisions, in fact the robot needs to use a logical system that records which turns it has taken, and where it has encountered a dead end, in order to “learn” where the dead ends are and avoid them. The ability of the robot to commit information to memory allows it to “learn” as a machine.

If you were to visit a person-sized maze, one of the first things you would realise is that you can't see the whole maze. While a bird flying overhead might be able to see the details of a maze, all you can see inside a maze is the part of the maze that is around you.

The robot can only see the part of the maze that’s right in front of it, being the table surface underneath where the line following sensor module is mounted. The next question is how does the robot find its way through the maze, even before it has recorded where the dead ends are?

To find its way through the maze the robot must implement an algorithm, which is a fancy name for a formula or set of rules. What someone discovered a long time ago was that if they kept their hand on the left hand side wall of a maze, and they did not lift their hand off the maze wall or re-enter or backtrack into a passage they traversed before, that person would eventually find their way to the end of the maze. This method is called the left-hand-on-the-maze-wall-algorithm.

To understand this better, keeping one's left hand on the wall forces the person in the maze to turn left when arriving at a passage to the left. If there is no left passage then keeping one's hand on the maze wall forces one to keep going straight. If there is no passage to the left, and there is no passage straight ahead, then the person would be forced to turn right to follow the left hand maze wall.



However, it is important to note the above method of navigating through a maze fails if there are loops in the maze. The left-hand-on-the-maze-wall-algorithm only works in a “perfect maze”, in other words a maze without joining passages or loops. That is one of the limitations of this particular algorithm. However, this is an excellent algorithm that produces a logical, repeatable outcome reliably every time, and can easily be used for teaching. Used in the line-following robot, this method is called the “left-hand-line-follower” [LHLF] algorithm which can be summarised as follows:

1. Always turn to the left if possible .. or make a U-turn if possible
2. If turning to the left is not possible, then go straight.
3. If it is not possible to turn to the left, or to go straight, then turn right.
4. If one arrives at a dead end, keeping one's left hand on the maze wall means you make a U-turn until you get back to the entry to the dead end and then turn left and continue through the maze until the next intersection, next dead end, or the end of the maze.

Applying the above will get a person [or robot] to the end of a perfect type maze, eventually.

The LHLF algorithm means the robot will go down every bad path, or dead end, which probably doesn’t sound very efficient … but it will get the robot to the end of the maze. If you observe the “behaviour” of the robot in the video notice how it does not go down dead ends that branch off the right hand side of the maze. If you swap everything around a right-handed algorithm also works.

You might want to do a Google search using these terms to find this excellent explanation:

“professor robert solis CS225 solving algorithms left hand maze algorithm”

In the next few parts of this workshop we take a look at how we can translate the above LHLF algorithm into actual working computer control code, building upon everything we have learnt about before in all the classes we attended before this class.