**Group Project Report**

**GROUP 13**

**Members**

Tran Phuoc Tuan (s3866850)

Nguyen The Minh (s3878434)

Nguyen Viet Quan (s39)

Vo Phuc Duy Nhat (s3868763)

**Lecturer:** Dr. Tri Dang

**Due date:** December 18, 2022

**Table of Contents**

[**I.** **Introduction** 3](#_Toc122288329)

[**II.** **Description of the Algorithm** 3](#_Toc122288330)

[Figure 1: Trémaux’s Algorithm [3] 4](#_Toc122288331)

[**1. Subproblem 1: update\_move\_to\_be\_made** 4](#_Toc122288332)

[**2. Subproblem 2: validate\_direction** 5](#_Toc122288333)

[**3. Subproblem 3: traceback** 5](#_Toc122288334)

[**4. Subproblem 4: Navigate** 5](#_Toc122288335)

[**III.** **Complexity and Pseudocode** 5](#_Toc122288336)

[**Compexity of the algorithm** 5](#_Toc122288337)

[**Pseudocode** 6](#_Toc122288338)

[**IV.** **Evaluation and comparison** 7](#_Toc122288339)

[Figure 2: Left and Right Wall following routine [1] 8](#_Toc122288340)

[Figure 3: Wall following Algorithm [4] 8](#_Toc122288341)

[Figure 4: Result for bottom right conner exit of Tremaux's algorithm (TA) in comparison to the other method [4] 9](#_Toc122288342)

[Figure 5: Result for Center finish point of Tremaux's algorithm (TA) in comparison to the other method [4] 9](#_Toc122288343)

[**V.** **Conclusion** 10](#_Toc122288344)

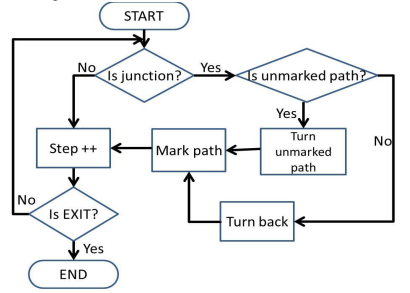
[**VI.** **Reference** 10](#_Toc122288345)

# **Introduction**

The advancement of navigational technology has resulted in the necessary to employ automated systems in the process of mapping areas that are too hazardous to investigate by hand. A small-scale autonomous robot is used in this study effort to investigate a maze by employing a variety of different experimental mapping and navigating methods. Specifically, we are designing data structures and algorithms for a robot which can automatically explore a labyrinth. At most 1000 rows and 1000 columns can be found throughout the rectangle-shaped maze. Each row is a String, and the length of each row is the same as the size of the other rows. A matrix of known size that contains one of the following in each cell: " " for blank areas, "." as walls, "X" for the exit gate, and so on. The robot is initially placed in a seemingly random location. The robot has the capability of moving in any of the following four orientations during every stage: up, down, left, or right. The robot cannot see anything in the maze since it is completely dark.

# **Description of the Algorithm**

Starting with three sorts of values: a dot (.) symbolises a wall or obstacle in the maze, a space () represents a feasible course of travel for the robot to take to reach the ending point, and an uppercase letter (X) represents the goal that the robot must find. Trémaux 's algorithm, often known as the DFS, serves as the foundation of our application (Depth First Search). We programmed the robot to move in the following directions: UP, RIGHT, LEFT, DOWN. This means that as soon as we locate a feasible path, we will check to see if it is a genuine path (not a wall or an obstacle or a path that has been gone before). Following that, we would commit to continuing in a single route until we encountered an invalid path, at which point we would try the next direction in the order. As a result, as long as there is one plausible way (valid path) in every location in the maze, the direction will continue until we reach the goal (X). This is also used for numerous possible paths, implying that there is more than one valid path. We shall continue to follow the above-mentioned direction until it either encounters a dead end or leads to the destination (X). Having said that, there will be times when the robot comes to a dead end (a point where there is no legitimate path to take next) and we will need to employ a backtracking method. We use two attributes in the Robot class (move to be made and origin) to maintain track of the robot and save data for every location it has visited. The move to be made is a matrix similar to the original maze, but with double the size, which will be handed by the lecturer and filled with 0. We are looking forwards to storing the number of times the robot has visited each location in the maze for this property. As a result, we will be able to prevent the robot from returning to its previous course, so avoiding a loop in the robot's journey. The detailed algorithm mechanism shown as the figure below:



## Figure 1: Trémaux’s Algorithm [3]

Similarly, the origin is coded to be a matrix that matches the original maze but is larger in size. This feature serves the goal of tracking the robot's previous journey, assisting the robot in backtracking if it eventually hits a dead end. To solve the maze with the size of M\*N, we have the following subproblem below:

## **1. Subproblem 1: update\_move\_to\_be\_made**

This subproblem is used to update the move\_to\_be\_made matrix based on the direction that the robot chose. We take this situation to 15 different cases as follows: 1-4 represents 4 basic directions (UP, RIGHT, LEFT, DOWN); 5-10 represents 6 combinations of 2 distinguished directions; 11-14 represents 4 combinations and 15 represents a dead-end route (no valid path). Based on this numeric system, we write for each spot in the move\_to\_be\_made matrix number to represent its case.

## **2. Subproblem 2: validate\_direction**

This subproblem is used to check whether the next spot to go in the maze is a valid path or not.

## **3. Subproblem 3: traceback**

This subproblem is used to find the direction to backtrack the route if the next path is a dead-end based on the origin matrix.

## **4. Subproblem 4: Navigate**

This subproblem is used to decide which direction to go as well as when the robot should backtrack or not. More importantly, this subproblem will present the result after the decision if it is a winnable maze or it is an unsolvable maze.

# **Complexity and Pseudocode**

## **Compexity of the algorithm**

//1 for up, 2 for right, 3 for down, 4 for left, 5 for 1 + 2, 6 for 1 + 3, 7 for 1 + 4, 8 for 2 + 3, 9 for 2 + 4,

//10 for 3 + 4, 11 for 1 + 2 + 3, 12 for 1 + 2 + 4, 13 for 1 + 3 + 4, 14 for 2 + 3 + 4, 15 for the end no route left

// return 1 => we can go that direction

// return 2 => we can't go that direction since there is 1. obstacle, 2. we have go there already

//3 condition for each case:

//1. we have not made that move yet

//2. there is no obstacle there (Check in navigate)

//3. we have not been here yet since if we have been here, we can just call a traceback if there is no step to explore anymore

* Every vertex in the graph is visited once, and every edge is examined once, during a depth-first search. DFS complexity is therefore O(V + E)O(V+E). Assumed here is the representation of the graph as an adjacency list.
* In conclusion, depth-first search, which has an O(n/2) time complexity, enables the algorithm to visit each node once or more times depending on the size of the map and gather all feasible pathways, including the optimal one.

## **Pseudocode**



# **Evaluation and comparison**

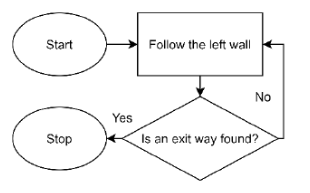
The first stage in any maze-solving system is a maze-solving algorithm. Various maze-solving algorithms attempt to identify a path from the source to the objective. Two maze-solving algorithms are compared in this portion of the paper. The most fundamental way for solving mazes is the wall following Algorithm, and the Trémaux Algorithm, commonly known as DFS, is an effective strategy that needs drawing lines on the floor to establish a path, which is the most important foundation for our project's aims.

To begin with, the Wall following algorithm is the most popular type of algorithm used by maze-solving robots. The robot's path will be determined by the wall to its left or right. This strategy is sometimes referred to as the left-right hand rules. When the robot comes to a fork in the road, it will look for openings in the walls and choose a path that prioritises the wall it has already chosen [1]. This strategy can get the robot to the end of the maze without needing to solve it since it uses the walls as guides. This solution, however, is ineffective for solving maze issues. As a result, solving some maza layouts, such as a maze with a closed loop section, will be impossible using the wall follower strategy [2]. Thus, this behavior can be clearly observed through the figures below:

Ảnh có chứa bàn

Mô tả được tạo tự động

## Figure 2: Left and Right Wall following routine [1]



## Figure 3: Wall following Algorithm [4]

Because of this, the wall following algorithm will not work properly in the majority of maze, especially those with loops. Tremaux's algorithm, on the other hand, can determine whether or not there is a direct route to the target point and always finds a route to the goal point in any case [5]. To be more specific, Trémaux 's algorithm operates as follows: if a junction is unmarked (unvisited), a random unmarked path is chosen, followed, and registered as visited. Turn around and return along the same path if a junction contains a single mark, identifying it as reached a second time. This situation may occur if the robot comes to a halt. If a junction has a lot of marks, pick one of the remaining paths with the fewest markings, follow it, and mark it as visited [5]. It is guaranteed will always function with mazes that have clearly defined passageways. Furthermore, some of the collected research will be shown to support the team’s decision: First of all is the case of bottom right conner Exit (X).

Ảnh có chứa bàn

Mô tả được tạo tự động

## Figure 4: Result for bottom right conner exit of Tremaux's algorithm (TA) in comparison to the other method [4]

Similarly, with the case of Center finish point.

Ảnh có chứa bàn

Mô tả được tạo tự động

## Figure 5: Result for Center finish point of Tremaux's algorithm (TA) in comparison to the other method [4]

In a nutshell, the experiment's results revealed that Tremaux's algorithm was effective in completing the mazes employed in the experiment. At this critical point, our team determined that the requirement outlined in this report needed to be developed and implemented.

# **Conclusion**

Finally, using the Trémaux Maze Solving Algorithm, we were able to successfully execute project solutions. Mapping and navigation algorithms that successfully compiled were developed. The algorithm performed exceptionally well in every maze. In theory, the function performed as intended and was able to provide support for the robot. As a consequence of this, the prototype would have accurately navigated the maze and performed it in accordance with its requirements.

This project aims to expand a variety of critical research facts as well as comprehension of a big number of decision-making algorithms. Furthermore, learning about the digital execution and orientation of this project was beneficial. The fresh knowledge gathered by our team will have a significant impact on the work we undertake with actual prototypes in the future.

# **Reference**

[1] Autonomous Maze Solving Robot - ResearchGate. [Online]. Available: https://www.researchgate.net/publication/316664613\_AUTONOMOUS\_MAZE\_SOLVING\_ROBOT. [Accessed: December 13, 2022].

[2] Min Raj Nepali, “A Novel Wall Following Algorithm for Mobile Robots”, Cscjournals.org, 2017. [Online]. Available: http://www.cscjournals.org/library/manuscriptinfo.php?mc=IJRA-126. [Accessed: 10- December- 2022].

[3] N. Zheng, Yew, K. Ming, S. Ting, and Yong, “Recursive Path-finding in a Dynamic Maze with Modified Tremaux’s Algorithm,” World Academy of Science, Engineering and Technology International Journal of Mathematical and Computational Sciences, vol. 5, 2011.

[4] I. R. Fahmi and D. J. Suroso, “A Simulation-Based Study of Maze-Solving-Robot Navigation for Educational Purposes,” Journal of Robotics and Control (JRC), vol. 3, no. 1, Jan. 2022.

[5] S. Alamri, S. Alshehri, W. Alshehri, H. Alamri, A. Alaklabi, and T. Alhmiedat, “Autonomous Maze Solving Robotics: Algorithms and Systems,” International Journal of Mechanical Engineering and Robotics Research, vol. 10, Dec. 2021.