# A framework for comparing mobile robot navigation

Mohamed Osama Idries, Matthias Rolf, and Tjeerd V. olde Scheper

*Abstract*— **Exploration and navigation are one of the fundamental problems in mobile robotics. Efforts to address these range from reactive, map-based to learning-based approaches. With each method being developed and tested in an isolated environments, the precise improvements of these methods are unknown. This paper presents a framework to simulate, evaluate and compare these different algorithms. ******************* Our results demonstrate how methods compare over a range of attributes and environments. We anticipate that this framework and findings allow for development of more advanced approaches, but also serve as a good step towards navigating dynamic environments.**

## I. Introduction

In order for mobile robot systems to perform any task in their environment they must first be able to navigate it. The navigation of an environment generally involves the goal location and the approach to get there while avoiding any hazards along the way. Robotic navigation has been extensively studied for decades and there is extensive work on the navigation of environments. These efforts can generally be divided into reactive, that make only instantaneous decisions, map-based involving e.g. grid, metric or topological representations, and learning-based approaches that use neural networks and/or reinforcement learning.

Research in comparing these algorithms is mostly limited within a type of approach[][]. While there has been some research in exploring the limitations of navigational algorithms[][], a key issue that remains is that due to the fact that most approaches are developed in singular isolated environments there currently exists no method of comparing these algorithms. Furthermore, it's difficult to imagine fully autonomous exploration and navigation in a dynamic world where one approach would be continuously outperform. A more realistic method would be to dynamically switch between the different approaches in environments and conditions that they excel.

In this paper we are presenting a framework for comparing different navigation algorithms.

Introduce main contribution of this paper to aid in solving the problem above

## II. Related work and background

Present the main navigational algorithmic advancements justifying your choices too Present the comparison papers and how they have previously approached these comparisons

M.O. Idries, M. Rolf, and T. Scheper are with Faculty of Technology, Design and Environment, Oxford Brookes University, Headington Rd, Oxford OX3 0BP, The United Kingdom `midries,mrolf,tvolde-scheper@brookes.ac.uk`

This will contain a graphic showing all the different algorithm in a single environment. Each approach has it's own color. This serves to demonstrate in a simple way how the different algorithms behave in a simple and attracting way. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Lorem ipsum dolor sit amet, consectetur adipiscing elit.Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Fig. 1. Different navigational algorithms path towards a goal

## III. Methodology

### A. Experimental setup

The environments used to run these trials are on 3 different map sizes, obstacle types, and 3 complexities. The environments are in $300^2$, $600^2$, $1200^2$ arbitrary unit sizes surrounded by a wall to prevent the robot from leaving the area. The environment also has several added complexities in the form of obstacles, insurmountable rectangles of various sizes, hills, these can increase or decrease the cost of movement, and dungeons, which are a series of rooms and corridors simulating human like environments.

Combining the different obstacles types we have obstacle, hill, and dungeon maps. Furthermore, there are also obstacle and hill, obstacle and dungeon, hill and dungeon, and obstacle hill and dungeon maps. Depending on the size of the map the map types are calculated as following:

$$room_{count} = 15 * 3^{s-1} * c$$

$$corridor_{length} = 40 * 2^{s-1}$$

$$hills_{min} = (60 + (c-1) * 30) * 4^{s-1} - 10 * 4^{s-1}$$

$$hills_{max} = (60 + (c-1) * 30) * 4^{s-1} + 10 * 4^{s-1}$$

$$obstacle_{min} = (30 + (c-1) * 30) * 4^{s-1} - 10 * 4^{s-1}$$

$$obstacle_{max} = (30 + (c-1) * 30) * 4^{s-1} + 10 * 4^{s-1}$$

### B. environment type

3 different types of obstacles generating a total of 7 different types of maps, 3 different types

3 sizes 3 attributes

### C. Evaluation methods

## IV. Navigational algorithms

These algorithms represent the simplest implementation for each type of approach and were chosen for that reason.

### A. *random walk*

Random walk, a reactive strategy based on a Brownian motion, is an algorithm where the robot moves randomly till it gets to it's target location. The robot at every step evaluates if it can step forwards. If it can it will step forward otherwise it will randomly turn left or right.

### B. *wall follower*

Wall Follower, a reactive strategy based of a Braitenberg vehicle, is an algorithm where the robot moves forward until it finds a wall. It continues then to follow the wall counter-clockwise until it sees the goal at which point it moves towards it. Each step, after initially finding the wall, it will turn right and step forward if there is no wall on it's right, step forward if there is no collision ahead, or turn left if there is a collision ahead.

### C. *Pheromone potential field*

Pheromone potential field, a simple map-based strategy, is an extension of normal potential fields where the robot moves depending on the attractive and repulsive forces acting on it. The attractive force originates from the goal, and the repulsive fields originate from the walls within range, and the pheromone trail it leaves behind.

### D. *A* algorithm*

A*, a map-based strategy, is a widely used graph traversal algorithm that could be seen as an extension of Dijkstra's algorithm where the robot plans a path to the goal with it's current knowledge of the map. It proceeds to follow this plan until the point it can't (i.e. the plan suggest walking into a wall) at which points it plans a new path to the goal and sets out again.

**Algorithm 1** Random Walk strategy

```
while Robot Location != Goal Location do
    if Robot can step forward == true then
        Robot step forward
    else
        Orientation = Randomly left or right
        Robot turn to Orientation
    end if
end while
```

### E. *A* algorithm*

### F. *Q learning*

## V. RESULTS

For each trial (Robot, Goal, Map) we have the following measurements;

- $d_g$ distance to goal
- $\epsilon$ success
- $d_s$ distance to starting point
- $\sigma$ area explored
- $c$ path cost
- $\mu$ computational overhead
- $m_s$ map size
- $m_d$ map density

- How do we determine map complexity (needs a factor cross maps so that bigger maps are considered more complex)

$$map_{complexity} = \frac{d_{eu}}{d_{worst}} * \frac{map_{den}}{map_{size}} \qquad (*)$$

$$map_{complexity} = \frac{d_{eu}}{d_{worst}} * \frac{map_{den}}{map_{size}} \qquad (*)$$

- How does an algorithm perform on average

$$algorithm_{score} = (1 - \frac{d_g}{d_{eu}}) * \frac{d_b * c}{d_{eu}^2} \qquad (*)$$

| complexity / algorithm | 0-25% | 25-50% | 50-75% | 75-100% |
|---|---|---|---|---|
| random walk | 0 | 0 | 0 | 0 |
| wall follower | 0 | 0 | 0 | 0 |
| pheromone potential field | 0 | 0 | 0 | 0 |
| A* algorithm | 0 | 0 | 0 | 0 |
| Q learning | 0 | 0 | 0 | 0 |

- How does an algorithm's performance translate to hybrid maps.

| type / algorithm | O | D | H | OAD | OAH | DAH | ODH |
|---|---|---|---|---|---|---|---|
| random walk | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| wall follower | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pheromone potential field | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A* algorithm | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Q learning | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## VI. DISCUSSION AND CONCLUSION