Spiralling Human-Inspired Exploration Algorithm with Doorway Detection

Rasmus Borrisholt Schmidt, Andreas Sebastian Sørensen, Thor Beregaard and Michele Albano[a]

Dept. of Computer Science, Aalborg Universitet, 9220 Aalborg Øst, Denmark

Keywords: Minotaur, TNF, Online Exploration, Signal Degradation.

Abstract: Exploration of unknown environments is an important task for autonomous robot swarm systems. The faster they can fully explore an area, the faster a coordinated plan can be made, or points of interest found, to support further tasks. Previous algorithms have often focused either on frontier based, or nature-inspired heuristics. We present a human-inspired exploration algorithm, Minotaur, that enables simple and efficient exploration of buildings. We studied how Minotaur and a state-of-the-art algorithm, namely The Next Frontier (TNF), perform. Minotaur follows walls to discover doorways, after which it coordinates with robots in the same room to extend the exploration to rooms accessible through the discovered doorways. Most algorithms assume either perfect communication, or line-of-sight (LOS) communication, which hinders the realism of the simulation results. We then modified an existing simulator to take into account realistic communication technologies that have limited penetration of materials through walls. Comparative experiments between Minotaur, TNF, and a simple greedy algorithm show the superiority of Minotaur when multiple robots are exploring buildings-like maps. However, when considering cave-like maps, Minotaur appears to have bad performance, but the greedy algorithm outperforms TNF, particularly when the algorithms are limited in their communication capabilities.

1 INTRODUCTION

Robot exploration is a common problem in swarm robotics, with applications in search & rescue, firefighting, emergency response, and space and undersea exploration (Queralta et al., 2020). Many exploration algorithms have been proposed, some inspired by animals (Nicola et al., 2018; Gul et al., 2021; Gul et al., 2023b; Gul et al., 2023a), some using frontier exploration (Burgard et al., 2005; Bautin et al., 2012; Colares and Chaimowicz, 2016), and some using various mathematical concepts (Mirjalili, 2016; Kennedy and Eberhart, 1995).

This paper proposes Minotaur, a human-inspired exploration algorithm. Each robot explores along the nearest wall, at a little less than their maximum vision range, detecting doorways on the way. If there are multiple robots in the current room, half of the robots will enter the discovered doorway, and explore from there, spreading throughout the map. When an explored area is seen, robots follow the explored area similarly to the initial wall-following strategy. From time to time, robots communicate their maps and marked doorways to limit redundant movements.

Inter-robot communication is useful to share the Simultaneous Localization and Mapping (SLAM) maps that the robots are building during exploration and robots current paths, and in general to strategize further exploration tasks. Most simulation solutions assume perfect communication between robots. This paper lays foundation for more realistic simulation by implementing a novel communication mechanism that takes into consideration signal loss through materials in an existing simulator, and uses it to analyze the performance of Minotaur against a state-of-the-art exploration algorithm, namely The Next Frontier (TNF) (Colares and Chaimowicz, 2016) algorithm.

The rest of this paper describes related works, namely simulators and exploration algorithms (Section 2), presents our proposed algorithm (Section 3) and discusses its implementation and the mechanisms used to implement realistic communication[1] (Section 4). Experimental results are described in Section 5, followed by our conclusions on the topic at hand and suggestions for future work (Section 6).

[a] https://orcid.org/0000-0002-3777-9981

[1]The developed code is released as open-source on https://github.com/DEIS-Tools/MAES. A demo video can be seen on https://youtu.be/gjnVm46Ezd0.

Schmidt, R. B., Sørensen, A. S., Beregaard, T. and Albano, M.
Spiralling Human-Inspired Exploration Algorithm with Doorway Detection.
DOI: 10.5220/0013237700003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 112-122
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

2 RELATED WORK

This section discusses simulators for exploration algorithms, and describes existing exploration algorithms relevant for this work.

2.1 Exploration Simulators

When simulating robots, realism is paid in terms of complexity and performance. In fact, on the one hand many works used simple simulators that consider robots moving on a grid, disregard collisions, and allow for unrestrained communication (see for example (Cheraghi et al., 2020; **?**)). On the other side of the spectrum there are high fidelity simulators that are very complex to set up and have bad performance (Pitonakova et al., 2018).

We opted for Multi Agent Exploration Simulator (MAES) (Andreasen et al., 2023), which provides a trade-off between realism and ease of usage. MAES is open source, it was used to compare exploration algorithms (Andreasen. et al., 2022) and provides the code to simulate them. Further, MAES supports the creation of transmission models dependent on distance and walls passed through, such as attenuation based models, and can generate maps that represent either cave-like or building-like environments.

2.2 The next Frontier (TNF)

TNF (Colares and Chaimowicz, 2016) is an exploration algorithm designed to use information gain and distance cost as a basis for autonomous swarm exploration. TNF uses the Near-Frontier Exploration (**?**), which identifies frontiers in the map, which are locations that are in-between explored and non-explored areas. In the fitness function that decides robots' waypoints, TNF considers the information gain that going to a given area would provide and the cost to get there.

Moreover, each robot stores the newest destination received from each other robot. When the robot decides upon the next frontier to explore, a wavefront will be created from each stored destination, reducing the score of each frontier near it. This means that frontiers close to the target frontiers of other robots will be lower in the score, causing the robots to spread out more in the map, exploring more disparate sections and reducing redundant work, speeding up the exploration.

2.3 Spiraling and Selective Backtracking (SSB)

SSB (Gautam et al., 2018) is a coverage algorithm that uses spiraling patterns and expects global communication and discrete grid-based movement. The robot spirals through known areas and updates other robots of the covered space and the nearby uncovered tiles it reserves for backtracking.

The backtracking is used to find new uncovered areas, and it uses an auction mechanism to decide which robot should go to each backtracking point. If no backtracking point is available, or if the robot did not win any auction for a backtracking point, an auction is instead done for the nearest unvisited point. As the robot spirals and creates backtracking points, it reserves the points of the spiral and backtracking, and broadcasts them to other robots, limiting their movement in an attempt to reduce redundant movement.

Our proposed solution uses concepts from SSB, while improving it in terms of robots' coordination and resilience to communication impairments.

2.4 Human Heuristics

Most previous algorithms are generally focused on mathematical formulas, or get inspired by animal behavior. An alternative approach is to explore in a way that makes sense to humans.

Human heuristics are used in various tasks in robotics (**?**; **?**). An example of human heuristics used for exploration is in (Liu et al., 2023), where the authors utilize computer vision with human heuristics to detect where doors are to separate rooms apart. Furthermore, it prioritizes finishing exploring the current room before proceeding to the next one.

3 THE MINOTAUR EXPLORATION ALGORITHM

The general idea behind our proposed exploration algorithm is to use two human-inspired heuristics: (i) robots will spiral inside a room to explore it; (ii) when a robot identifies a door in a room, half of the robots in the room will move out through the door to cover as many rooms as possible. Part of this approach is inspired by SSB (Gautam et al., 2018), particularly the spiraling movement and the auction mechanism to decide which robot goes to important locations, in our case doors.

To explain the algorithm, we will initially go over a constrained version for one single robot, then expand it to show the full multi robot behavior, with smaller black-box around doorways functionalities that are then described in their own sections, all together creating the full algorithm.

3.1 Single Robot

In the single robot algorithm, whose pseudocode is shown in Alg. 1, the robot starts by exploring along the wall nearest to its initial location, moving in a straight direction looking for a wall if it cannot see any. After this, the robot follows the wall, identifying (see Section 3.3) and marking the doorways it finds without entering them until current room is fully explored.

After covering all walls in a room, it will explore the center of the room in a spiral pattern, until the whole room is explored. After that, the robot will enter the nearest unexplored doorway in the room, and continue the exploration from there, returning to unexplored doorways in previous rooms if no unexplored door can be found in the new room.

```
Data: Maximum range of robot sensors
while No walls or explored areas are in range do
    Move forward ;
end
while Map is not explored do
    while Unexplored area is reachable without
      passing doorways do
        if Doorway found (Alg. 3) then
            Store the doorways, stay in the room
              and continue exploring along
              nearest wall ;
        end
        if Wall or explored area in visual range
          && unexplored area ahead then
            Follow wall or explored area at max
              visual range counter-clockwise ;
        else
            Go to the nearest unexplored area
              without passing a doorway ;
        end
    end
    if Unexplored doorway accessible without
      passing other doorways then
        Move through the nearest unexplored
          doorway within room ;
    else
        Move through the nearest unexplored
          doorway ;
    end
end
```
Algorithm 1: Minotaur Algorithm (Single Robot) .

```
Data: Maximum range of robot sensors
while No walls or explored areas are in range do
    Each robot moves forward ;
end
while Map not explored do
    Each robot broadcasts map, including walls,
      explored area, doorways and known robots
      location ;
    For half of robots in current room spin ←
      counter-clockwise, for the rest spin ←
      clockwise ;
    while Unexplored area is reachable without
      passing doorways do
        if Doorway found (Alg. 3) then
            Store & handle doorway (Alg.4) ;
        end
        if Wall or explored area in visual range
          && unexplored area ahead then
            Follow wall or explored area at max
              visual range in spin direction ;
        else
            Go to the nearest unexplored area
              without passing a doorway ;
        end
    end
    if Unexplored doorway accessible without
      passing other doorways then
        Move through the nearest unexplored
          doorway within room ;
    else
        Move through the nearest unexplored
          doorway ;
    end
end
```
Algorithm 2: Minotaur Algorithm.

3.2 Multiple Robots

In the strategy for multiple robots, see Alg. 2, half of the robots explore around the rooms in one direction, the rest proceed in the other direction, to identify all doorways quickly.

As they move, the robots broadcast their SLAM map and any found doorways for other robots to add to their own maps. On finding a doorway, the robot starts an auction with all the robots in the same room, which respond with a bid equal to their distance to that doorway, so long as they can get there without passing another doorway, i.e. they are in the same room. This ensures that no robot spends time moving to a doorway far away, and that all room that have started being explored also get finished. This is further described in Section 3.4.

3.3 Doorway Detection

When a robot has one or multiple walls in their line of sight, it will check if it is possible that it discovered a door, with a set of conditions that are different for one or multiple walls, as described in Alg. 3.

If there is a single wall, then when the robot finds a gap in the wall, it will continue along the wall as far as the door width parameter, and mark the gap as a door if it finds the continuation of the wall. In Fig. 1, in the first case there is a door in the middle of a wall; in the second one the door is located in the room's corner; in the third case there is no door in a corner; the fourth case represents a hallway, i.e. the distance between the interruptions of the wall is larger than the maximum door width.

If there are multiple walls, then the robot will create intersection points between the walls as infinite lines. If these intersection points have been seen, are non-solid, and close enough to make a doorway, then it is a door (last case in Fig. 1).

```
Data: Maximum range of robot sensors
Data: Maximum width of doors
if Wall ends/stops then
    move forward doorway width ;
    if Wall continues after doorway width then
        Mark doorway and direction it is seen from ;
    end
end
if Two walls in vision area then
    Store the wall points;
    Continue vision range forward;
    Store the new wall points;
    Create lines along wall direction between the points;
    if Lines intersect && distance between wall points greater than robot size then
        Find intersection point of lines;
        Move towards intersection point until it is explored;
        Create line segments between wall points and intersection point;
        if Ensemble of line segments is open then
            Mark doorway and direction doorway it is seen from;
        end
    end
end
if any doorways "seen" from both sides or doorway was passed through then
    Mark doorway as explored;
end
```

Algorithm 3: Doorway detection, as function of door maximum width.

```
Communicate doorway location to all robots (comprising itself);
if Receiving robot can reach the doorway on the same side without passing any doorway then
    Receiving robots give ACK and distance;
end
out ← half of received ACK (rounded down);
Command the out robots to exit the doorway;
The rest of the robots will stay in the room ;
```

Algorithm 4: Doorway auction.

3.4 Doorway Auction

When a robot, let us call it Alice, finds a door, it starts an auction by sending out a "doorway found" message. When a robot in communication range receives the message, comprising Alice, if it is able to reach that door without passing through another doorway, i.e. it is in the same room, it will respond with an ACK and a bid equal to its distance to that door. Alice too will respond to the auction with an ACK and its own bid. Afterward, Alice tells to the half of the respondents having the lowest bids to proceed through that door (Alice will probably be in this set). The rest of the robots will stay in the room to finalize its exploration. This is formalized in Alg. 4.

The approach is also visually explained on Fig. 2, where green areas are unexplored, the colored dots are robots, and the dashed lines are communication lines, with their color signifying whether they respond with an ACK (green) or not (red). On Fig. 2, Pink finds a door and communicates to the other robots in range. Cyan will be able to go to the room. Yellow cannot get to the door without passing another doorway, and therefore will not reply with an ACK. While Red technically can get to the door, it does not know this, as the area in-between is unexplored. From its perspective, it would have to backtrack through multiple doorways to reach Pink, and so it will not reply with an ACK.

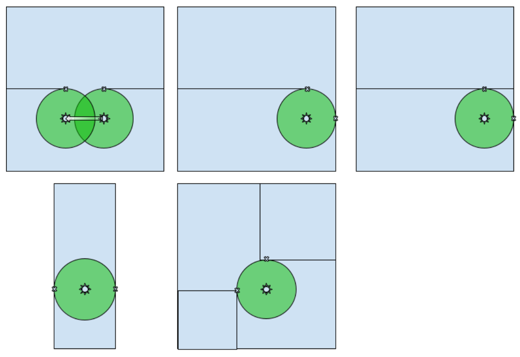


Figure 1: Base cases for the doorways.

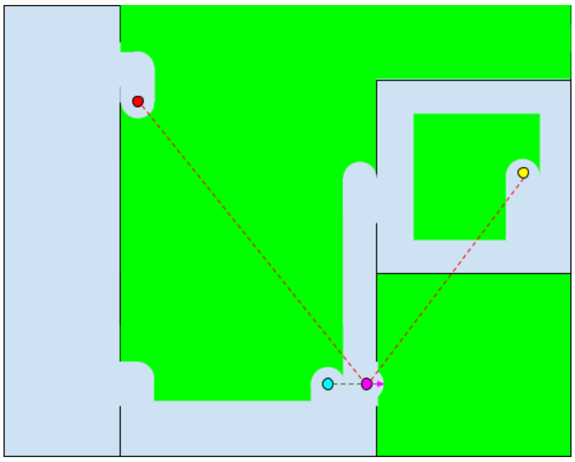


Figure 2: Communication scenario for Alg. 4.

3.5 Greed Algorithm

When the Minotaur has no nearby unseen areas, or unexplored doorways to go through, it will move to the nearest unexplored tile anywhere on the map and continue exploration from there.

This feature has been extracted into its own algorithm, creating a greedy algorithm called Greed, which always moves to the nearest unexplored tile, and still communicates and shares its map with other robots.

Compared to Minotaur, Greed has the advantage of being simpler and not needing to follow walls. On the other hand, Minotaur can better divide the robots between the rooms when it identifies doors.

4 DESIGN & IMPLEMENTATION

This section describes the support for realistic communication through walls in the selected simulator (MAES (Andreasen. et al., 2022)), and the spiraling and doorway detection mechanisms in Minotaur, since its implementation was surprisingly more complex than expected.

4.1 Materials in MAES

The MAES simulator allowed to simulate either global (unrestrained) and line-of-sight (LOS) only communication between robots. However, we deemed both of them to be not realistic enough and we decided to add to each section of the walls a *Material* property, and then use it to check if two robots are able to communicate with each other at a given time.

The physics of this problem has already been well studied, and multiple signal attenuation tables can be found for various materials at various frequencies, see for example (Ikpehai et al., 2019; Rappaport, 1996; **?**).

Signal loss can be described using the general path-loss algorithm, see for example Eq. 1, where α, β, and γ are empirically found ”magic” numbers that describe how much the signal is degraded for line of sight and non line of sight communication (**?**).

$$L_b(d, f) = 10\alpha \log_{10}(d) + \beta + 10\gamma \log_{10}(f) \quad (1)$$

In terms of implementation, we modified the mesh generation used to generate a map to add a *Material* property, which contains the α, β, and γ from Eq. 1, to each triangular section that makes up a wall in MAES. In terms of the Unity Engine, this was done using submeshes, which are subsets of a larger mesh and can contain separate materials, textures and animations. This was preferable in comparison to splitting the different meshes into separate game objects for performance sake, as it allows for the Unity renderer to perform draw call batching more efficiently.

Whenever MAES checks if two robots are able to communicate, it now draws a line between the robots, it splits it into segments each time the material between the robots change, and it calls a function with a list of tuples (Material, length), with one element in the list for each segment. The function will then apply Eq. 1 for each element in the list to compute how much the signal was attenuated. Equation 2 formalizes this process, where S_i is true when the transmission can get received successfully for robot i, R_i is the receiver sensitivity of robot i, T_j is the transmission strength of robot j, m is the number of walls (or open spaces) that have been crossed with a message, and $A_{n,f}$ are the attenuation values.

$$S_i = R_i \leq (T_j - \sum_{n=1}^{m} A_{n,f}) \quad (2)$$

The attenuation values for the different materials used in a map can be provided by the end user as a part of the configuration along with the transmission strength, frequency of the signal, and receiver sensitivity of the robots. MAES provides the default values in Table 1 and, if the end user does not specify their own, the transmission strength is set to 15 dBm, the receiver sensitivity is set to -82 dBm, and the frequency is set to 2.4 GHz. These values were chosen in accordance to 802.11ax also known as Wi-Fi 6 (IEEE, 2021).

4.2 Spiraling Implementation

This subsection focuses on the spiraling strategies that the robots follow while exploring a room. The imple-

Table 1: Default values for signal attenuation at different frequencies and materials (Ikpehai et al., 2019).

Frequency	1300 MHz	2400 MHz	5200MHz
Air	0 dB	0 dB	0 dB
Concrete	13 dB	15 dB	23 dB
Wood	5.1 dB	6.7 dB	14 dB
Brick	4.5 dB	5.5 dB	15 dB

mentation of spiraling relies on the overlaid grids on the SLAM map that the robot creates while navigating. Each tile on this grid can have one of three states, Open, Unseen and Solid, as can be seen in Fig. 3.

While following the wall on the initial exploration of a room, a robot looks for solid tiles within its vision range. The robot decides which wall to follow by checking if there are any unseen tiles in front of it. If there are none, the robot is done following that wall.

When the robot has multiple candidate walls, it prioritizes moving in the same direction that it is already moving. The robot computes the perpendicular vector from the furthest points it can see on the walls, to decide where to move to next, as illustrated on Fig. 4. If the wall ends or turns away from the robot without having a doorway, the robot will initially lose track of it, but then follow it around the bend. When reaching the end of a confined area, like a dead-end in a hallway, the robot will return to the last seen area with unexplored tiles and continue the spiraling pattern from there.

On completing the first lap around the room, the robot will start to spiral around the edge of the previously explored area. This is tricky, since opposite to walls, edges between explored and unexplored areas disappear when unexplored area in question gets explored. Therefore, the robot must use the fact that the explored area is continuously fed into its SLAM map, to look ahead beyond its own vision range and follow the edge there.

If the spiraling movement cannot reach unseen tiles, the robot will move greedily towards the nearest unseen tile in the room. When a room is fully explored, its robots enter the nearest unexplored doorways and continue the exploration.

4.3 Doorway Detection Implementation

While the robot is moving to any destination, it checks for doorways. For a doorway with a single wall, as seen on Fig. 5, the robot sees that the next tile of the wall is an open tile. When this happens, the robot will move forward for a distance equal to the doorway width parameter. The robot will then be standing at the end of the movement vector and will be able to see if the wall continues, and if so, creates an intersection point between the walls. Then it checks if that is a valid door, according to the parameter *maximum door width*.

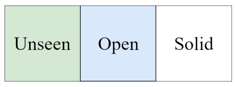


Figure 3: Representations of the different tile statuses.

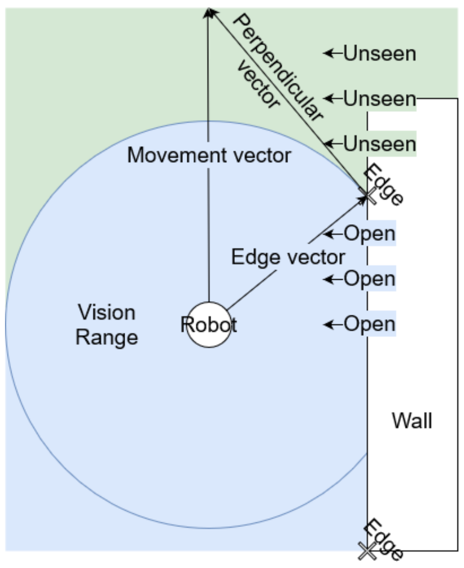


Figure 4: A valid wall due to having an unseen area where there presumably is a wall with edges and movement vector based upon those edges.

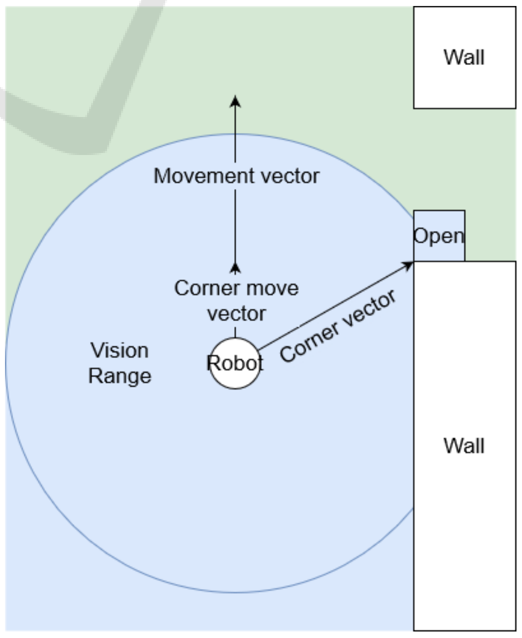


Figure 5: Doorway detection when seeing a single wall.

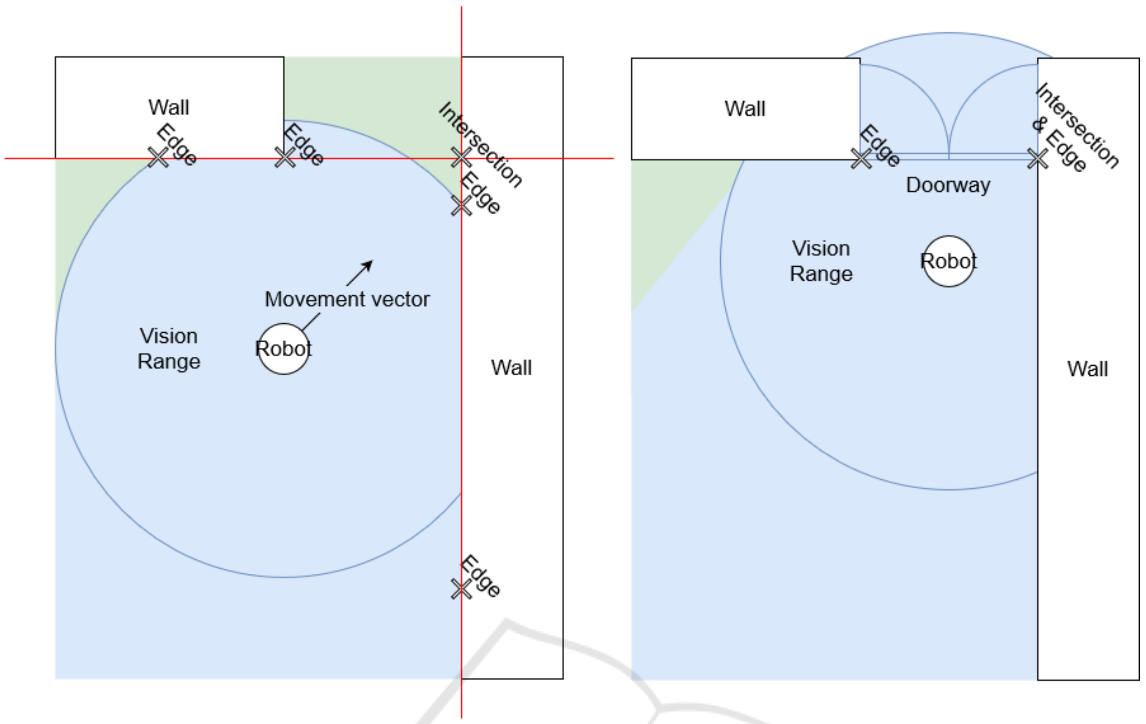


Figure 6: Doorway detection when seeing multiple walls. The left shows the state on detection of a potential door, and the right shows the state after moving to check intersection points.

If there are two or more walls, the robot looks for an intersection point between those, as can be seen on Fig. 6. When the intersection point is in an unseen area, as it is likely to happen, the robot will go to explore the intersection point. After the robot has moved to see the point, it takes the closest points to the intersection from the two walls that created it, as can be seen on the two edge points on the right state of Fig. 6. With those two points, the robot checks if they create a valid doorway, by once again considering the *doorway maximum width* parameter.

5 EXPERIMENTS AND RESULTS

In this section, we go through the experimental setup and the results of our experiments. These experiments are meant to understand the effect of more realistic communication on the exploration algorithms, and to compare the performance of Minotaur with TNF, as well as Greed (see Section 3.5).

5.1 Experimental Setup

To perform a fair comparison of the algorithms and the simulation settings, we initialized a random number generator with the seed 123456, then we generated 100 random building maps and 100 random cave maps, we selected random initial locations for the robots, and later we performed all simulations with those same initial conditions. An experiment was considered complete when 98% of the area was explored. On all the graphs in Section 5.2, we present the percentage of map that was explored at a given time, averaged over the 100 maps.

The simulation campaign we performed considered a map of 50x50, 100x100 or 150x150 tiles, 1, 2, 4, 8 or 16 robots, robots spawned either close together or at random over the whole map, maps were either representing caves or buildings (see Section 3.1 in (Andreasen et al., 2023)), the algorithm was either TNF, Minotaur or Greed, and the communication technology had either global coverage, line-of-sight limitation or we simulated communication through

Table 2: Settings for the exploration scenarios.

Robot random seed	123456
Maps size	50x50, 100x100, 150x150
Number of robots	1, 2, 4, 8, 16
Initial position of robots	Close together, random
Map type	cave, building
Exploration algorithm	TNF, Minotaur, Greed
Communication	Global, LOS, materials

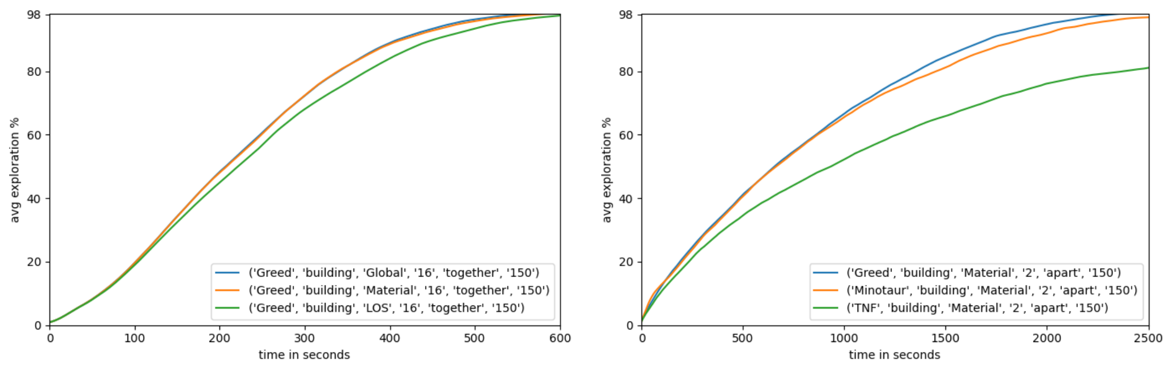


Figure 7: On the left, comparison of communication technology for the Greed algorithm, 16 robots, 150x150 building maps. On the right, comparison of algorithms for 2 robots in 150x150 tiles building maps.

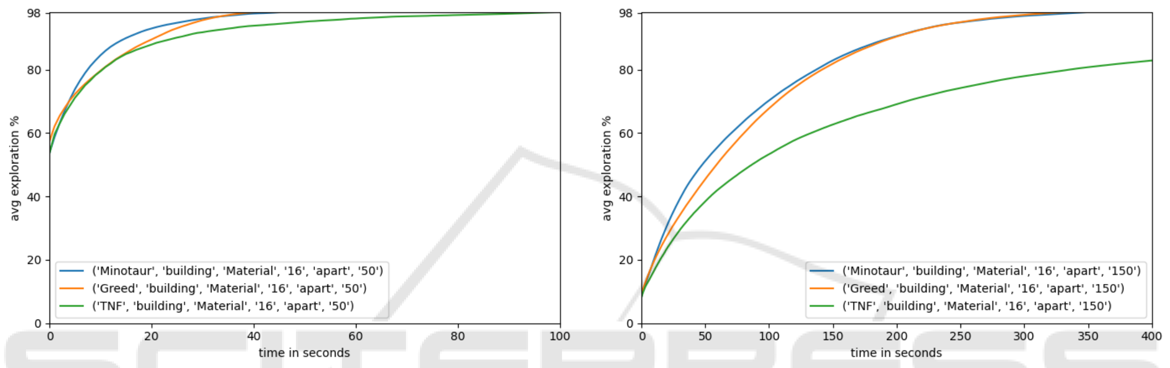


Figure 8: Comparison of algorithms with 16 robots on building maps (size is 50x50 tiles on the left, 150x150 tiles on the right).

materials as presented in Section 4.1. Table 2 provides a summary of the parameters used for the simulation campaign.

5.2 Results

We have compared the performance of the exploration algorithms when different communication capabilities were considered. In all cases, as expected, performance with LOS limitations was worse than with global communication, with communication through materials being in the middle. The performance gap was more pronounced with larger maps and when more robots were part of the picture. However, the effect was always quite limited. The left graph in Fig. 7 shows one of the most pronounced effects, for experiments with the Greed algorithm on large (150x150 tiles) building maps, 16 robots spawned close to each other. The rest of the results we are presenting will consider communication through materials.

One of our hypothesis was that Minotaur requires a sizable number of robots to benefit from its capacity to coordinate its robots efficiently. Results from 2 robots exploration confirmed it, but also hinted that in large building maps (150x150 tiles) both Greed and Minotaur outperform TNF, with Greed being slightly better than Minotaur. The right graph in Fig. 7 shows one example of this, for 2 robots over 150x150 tiles building maps.

When many robots are part of the picture, we expected that Minotaur had an edge over the other algorithms, since it pushes robots to cover the most rooms as fast as possible. The results confirmed it, even though the performance of the Greed algorithm was quite close to Minotaur's. Fig. 8 reports on a comparison of the algorithms using 16 robots on building maps of different sizes (50x50 tiles vs 150x150 tiles).

Finally, we decided to compare the algorithms on cave maps. The performance of Minotaur was very bad with respects to the other algorithms. This is explainable by the fact that Minotaur depends on its mechanism to recognize doors as opening on straight walls. However, cave maps do not have straight walls, invalidating the benefits of Minotaur, which still suffers from overheads. Fig. 9 reports on a comparison of the algorithms on cave maps, with 2 robots on 50x50 tiles maps on the left and 16 robots on 150x150 maps on the right, where TNF outperforms Greed on

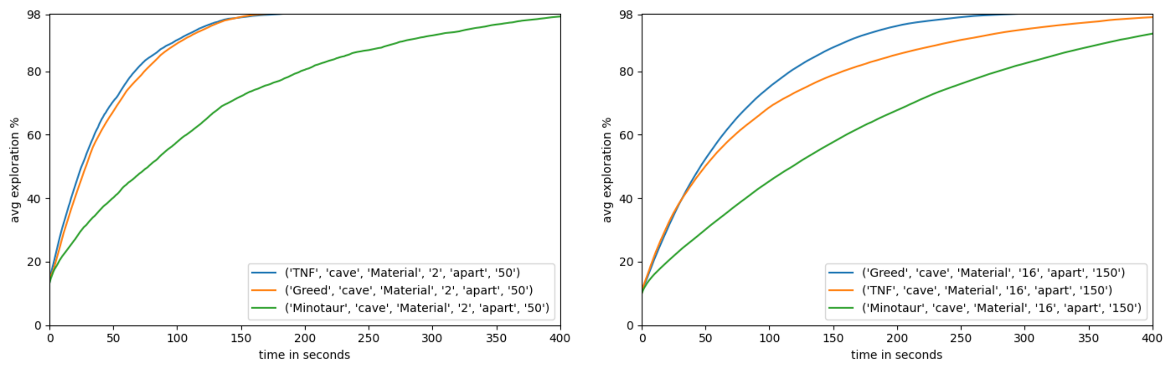


Figure 9: Comparison of algorithms with on cave maps (2 robots on 50x50 tiles on the left, 16 robots on 150x150 tiles on the right).

smaller maps with less robots, while Greed takes the lead on larger maps with more robots.

After the collection of the results, some thoughts were given to the causes of Minotaur slowdowns, for example on cave maps; and why on building maps Greed was often able to compete with Minotaur while both outperformed TNF. When a corner is encountered on a map, see Fig. 10, Greed approaches the corner, then it moves out looking for unexplored area. Minotaur instead moves closer to the corner and, when it sees the opposing wall of the corner, needs to check if there is a doorway. This causes it to move much closer to the wall than Greed.

A different situation is represented in Fig. 11, where Greed has a more direct movement through the door, while Minotaur spends time more to identify it, and then it decide to disregard the doorway and to continue the exploration of the room. These wasted movements all cause slowdown, providing Greed a relative advantage.

6 CONCLUSIONS AND FUTURE WORK

This paper presents an efficient exploration algorithm based on human-inspired heuristics. The algorithm uses the simple ideas of wall following, fully exploring rooms before leaving, looking for doorways, and splitting a group of robots as much as possible to continue the exploration of the current room while still crossing the newly found doors.

Comparisons have been made with The Next Frontier (TNF), and a greedy algorithm called Greed. When maps of buildings are explored, the more robots are added, the better Minotaur performes compared to Greed, and both algorithms perform far better than TNF. In case of maps of caves, Minotaur has bad performance since it loses its capability of door identification while still suffering from the overheads to execute its mechanisms.

To perform a more realistic comparison, we also enhanced the simulator MAES to support the associa-

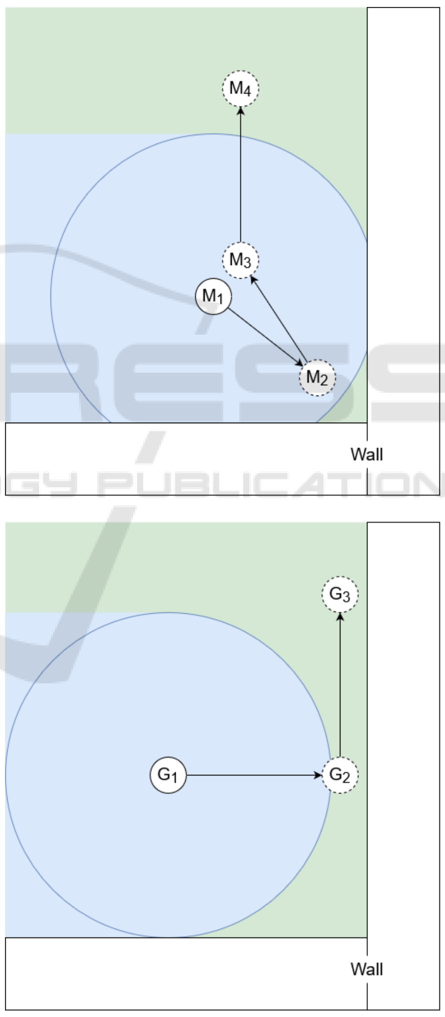
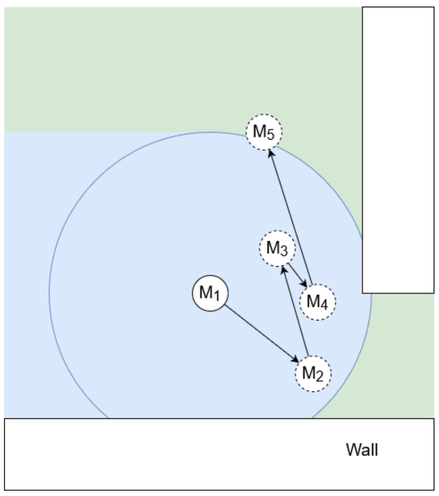


Figure 10: Minotaur (top) and Greed (bottom) exploring a corner without a doorway.



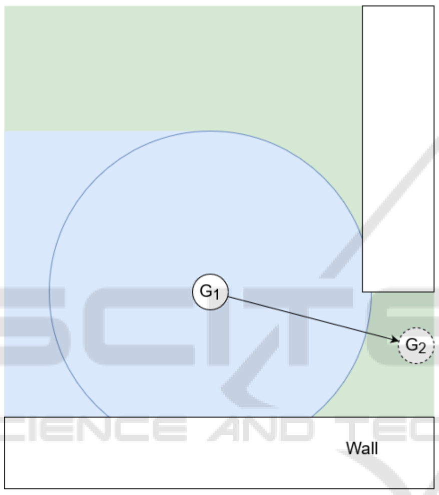


Figure 11: Minotaur (top) and Greed (bottom) exploring a corner featuring a doorway.

tion of materials to walls, which then are used to compute signal loss for communication between robots.

Future work on the topic can comprise the definition of an alternative door identification algorithm that can work also on cave maps.

More far fetched improvements can consider the extension of the Minotaur algorithm to the 3rd dimension. This would allow the algorithms to be used with aerial drones, mapping out far more complex spaces. This would likely require a different understanding of the SLAM map.

While algorithms can show promising results in the MAES simulator, implementing them on real robots, and testing them on areas created in the real world, would likely present unique challenges and different results.

ACKNOWLEDGEMENTS

This work was partially supported by the S4OS Villum Investigator Grant (37819) from Villum Fonden, and by the Validation WP of the ERC Advanced Grant LASSO (Learning, Analysis, SynthesiS and Optimization of Cyber-Physical Systems).

REFERENCES

Andreasen., M. Z., Holler., P. I., Jensen., M. K., and Albano., M. (2022). Comparison of online exploration and coverage algorithms in continuous space. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 1: SDMIS*, pages 527–537. INSTICC, SciTePress.

Andreasen, M. Z., Holler, P. I., Jensen, M. K., and Albano, M. (2023). MAES: a ROS 2-compatible simulation tool for exploration and coverage algorithms. *Artif. Life Robotics*, 28(4):757–770.

Bautin, A., Simonin, O., and Charpillet, F. (2012). Minpos: A novel frontier allocation algorithm for multi-robot exploration. In *Intelligent Robotics and Applications: 5th International Conference, ICIRA 2012, Montreal, Canada, October 3-5, 2012, Proceedings, Part II 5*, pages 496–508. Springer.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.

Cheraghi, A. R., Abdelgalil, A., and Graffi, K. (2020). Universal 2-dimensional terrain marking for autonomous robot swarms. In *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 24–32. IEEE.

Colares, R. G. and Chaimowicz, L. (2016). The next frontier: combining information gain and distance cost for decentralized multi-robot exploration. In Ossowski, S., editor, *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 268–274. ACM.

Gautam, A., Richhariya, A., Shekhawat, V. S., and Mohan, S. (2018). Experimental evaluation of multi-robot online terrain coverage approach. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1183–1189.

Gul, F., Mir, I., and Mir, S. (2023a). Aquila optimizer with parallel computing strategy for efficient environment exploration. *Journal of Ambient Intelligence and Humanized Computing*, 14(4):4175–4190.

Gul, F., Mir, I., Mir, S., and Abualigah, L. (2023b). Multi-agent robotics system with whale optimizer as a multi-objective problem. *Journal of Ambient Intelligence and Humanized Computing*, 14(7):9637–9649.

Gul, F., Mir, I., Rahiman, W., and Islam, T. U. (2021). Novel implementation of multi-robot space exploration utilizing coordinated multi-robot exploration and frequency modified whale optimization algorithm. *IEEE Access*, 9:22774–22787.

IEEE (2021). Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks. *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pages 1–767.

Ikpehai, A., Adebisi, B., Rabie, K. M., Anoh, K. O. O., Ande, R., Hammoudeh, M., Gacanin, H., and Mbanaso, U. M. (2019). Low-power wide area network technologies for internet-of-things: A comparative review. *IEEE Internet Things J.*, 6(2):2225–2240.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

Liu, J., Lv, Y., Yuan, Y., Chi, W., Chen, G., and Sun, L. (2023). An efficient robot exploration method based on heuristics biased sampling. *IEEE Transactions on Industrial Electronics*, 70(7):7102–7112.

Mirjalili, S. (2016). Sca: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, 96:120–133.

Nicola, G., Pedrocchi, N., Mutti, S., Magnoni, P., and Beschi, M. (2018). Optimal task positioning in multi-robot cells, using nested meta-heuristic swarm algorithms. *Procedia CIRP*, 72:386–391.

Pitonakova, L., Giuliani, M., Pipe, A., and Winfield, A. (2018). Feature and performance comparison of the v-rep, gazebo and argos robot simulators. In *Towards Autonomous Robotic Systems: 19th Annual Conference, TAROS 2018, Bristol, UK July 25-27, 2018, Proceedings 19*, pages 357–368. Springer.

Queralta, J. P., Taipalmaa, J., Can Pullinen, B., Sarker, V. K., Nguyen Gia, T., Tenhunen, H., Gabbouj, M., Raitoharju, J., and Westerlund, T. (2020). Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE access*, 8:191617–191643.

Rappaport, T. S. (1996). *Wireless communications - principles and practice.* Prentice Hall.