Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Universal 2-Dimensional Terrain Marking for Autonomous Robot Swarms

Ahmad Reza Cheraghi, Abdelrahman Abdelgalil, Kalman György Graffi

2020

Preprint:

This is an accepted article published in 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2020). The final authenticated version is available online at: https://doi.org/10.1109/ACIRS49895.2020.9162613
Copyright 2020 IEEE

Universal 2-Dimensional Terrain Marking for Autonomous Robot Swarms

Ahmad Reza Cheraghi
Technology of Social Networks
Heinrich Heine University
Düsseldorf, Germany
ahmad.cheraghi@hhu.de
https://tsn.hhu.de

Abdelrahman Abdelgalil
Technology of Social Networks
Heinrich Heine University
Düsseldorf, Germany
abdelrahman.agalil@gmail.com
https://tsn.hhu.de

Kalman Graffi
Honda Research Institute Europe GmbH
Offenbach am Main, Germany
kalman.graffi@honda-ri.de
https://www.honda-ri.de

***Abstract*—This research introduces the design and implementation of a universal systematic 2-dimensional terrain marking and coverage solution. Real world applications such as lawn mowing, mine detection, chemical spill clean-up, and humanitarian search and rescue missions can be automated by employing swarms of autonomous mobile robots to complete the task. In most of these real world applications, efficiency is of utmost importance especially when human lives are involved. The solution proposed builds up on two graph traversal algorithms; Depth First Search (DFS) and Breadth First Search (BFS), where the algorithms are adapted and modified to be utilized for coverage of 2-dimensional isometric grid-like representations of terrains. The solution is developed so that each robot in the swarm would be fully capable of covering and marking any terrain by itself. The efficiency of the solution is optimized by increasing the swarm size, as robots benefit from data sharing in their path planning and self organization within the terrain. Communication between robots enable them to perform on a higher level by benefiting from the collaborative distributed behavior of the swarm as a whole. The communication between particles is decentralized and is carried out at a local level with no need for a central guidance mechanism. Robot abilities in this research are limited, where a robot can see and move only to locations that are adjacent to its current location. A simulation based evaluation is conducted in this research to assess the robots' area coverage and marking performance. The results show that the simulated robot swarm systems are suited for efficient flat area coverage, allowing for redundancy in data collection, and tolerating individual robot errors and shortcomings as the number of robots becomes more abundant.**

***Keywords*—Robot, Swarms, Marking, Searching**

I. Introduction

Autonomous robot swarms are rapidly becoming a key player in tackling some of the most complex and, until not so long ago, virtually unsolvable real world problems. Swarm robotics, as a field of research, is inspired and motivated by observing living species in nature, such as, ants, bees or even bacteria. Similarly, researchers have been exploring different ways to enable programmable robots to mimic the behavior of these observed living species. Naturally, the collective behavior of robots within a swarm enables the single robots to achieve ant colony like features utilizing the decentralized distributed actions of the numerous independent robots.

Swarm robotics can be described as the field of research concerned with exploring how multi-agent (-robot) systems can be customized and programmed to successfully perform and complete given tasks. The tasks involved are, in most cases, collective by nature, that is to say, the successful completion of a task is either not achievable by a single robot, or can be achieved more efficiently, time and cost wise, by the collective behavior of the swarm as a whole.

The term "swarm intelligence" has been widely adopted within the past two decades as a reference to the vastly superior capabilities that the swarm achieves collectively as a whole, as opposed to, the fairly limited capabilities at the individual robot level. Despite the incapability and ineffectiveness of the single robots, their collective behaviors shared locally with one another renders into a global behavior that, in most cases, compensates for the individual robot's incapability to perform. That said, a system designed matching these criteria would be robust, flexible and scalable.

Robustness is the ability of a system to continue performing its task despite the loss or incapacitation of a number of its individuals. Swarm robotics systems make good use of the redundancy of their individuals, however, fewer functioning individuals means lower performance level from the system as a whole. Flexibility is the ability of a system to utilize modularized solutions to be capable of handling a wide variety of working conditions and collective tasks. Scalability is the ability of a system to function under a wide range of agent group sizes.

The main objective of this paper is to introduce the design, implementation, and evaluation of a **Universal 2-Dimensional Terrain Coverage and Marking Algorithm** for virtual robot(s), based on the Depth- and Breadth-First-Search algorithms. The proposed solution complies with the aforementioned criteria for swarm intelligence.

Having defined the task that this study addresses, the next chapter is dedicated to giving a brief overview of the various algorithms and techniques developed by fellow researchers to solve terrain exploration and covering problems. The techniques discussed in the following chapter involve single, as well as, multiple robot solutions for terrain coverage problems.

II. Related Work

Single Robot Terrain Coverage Gabriely and Rimon [1] have developed a solution for single robot systems, named

Spanning Tree Covering (STC), which provides optimal shortest paths within a square grid representation of a terrain. The algorithm functions by dividing the terrain area to exclusively disjoints cells, and then connecting the cells by creating a spanning tree structure. The two researchers have published three versions of their STC solution. The first version; *offline*; involves the robot obtaining a complete overview of its environment, prior to starting the terrain coverage algorithm. The second version of STC is *online*, where the robot has no prior knowledge of its environment, but instead, it utilized its set of on-board sensors to navigate through the terrain, detect and avoid obstacles, and simultaneously, generate the spanning tree of the environment by constantly discovering new locations. The third and final version of STC is an *ant-like* solution, where the robot does not have any prior knowledge of its environment and the set of its on-board sensors are limited. In this third version, the robot can navigate through the environment and create the spanning tree by leaving itself trails of pheromone-like markers on the grid.

Collaborative Terrain Coverage Building up on STC, Hazon and Kaminka [2] have managed to improve the offline version of the algorithm by introducing multiple robots for spanning tree coverage; *MSTC*. In this modified version of the algorithm, robots obtain the full spanning tree model before starting their coverage task. The goal here is to maximize terrain coverage while optimizing the task completion time. The two researchers managed to achieve this by programming the robots to choose the spanning tree route that circumnavigates through the whole spanning tree in a clockwise motion. The robot would choose the route closest to itself and follow it in a clockwise manner, hence, spanning the tree. Since the robots have prior knowledge of their environment's spanning tree, the algorithm divides the longest routes evenly between two adjacent robots to distribute work loads. The two researchers have further expanded the MSTC algorithm, by implementing backtracking logic that guarantees optimum results.

One more STC oriented algorithm based on cellular decomposition, was proposed by Agmon, Hazon and Kaminka [3]. The researchers attempted an approach where they would create multiple environment spanning trees and then connect them to one another to form one larger central optimal tree. The purpose behind their approach was, again, optimizing task completion time. This version of the algorithm is once again offline, where the robots start their terrain coverage task with prior knowledge of their environment.

Batalin et al. [4] proposed an offline multi-robot algorithm that spreads the robots within a terrain by having the robots move away from one another when they come within sensing range. In their research, the researchers attempted to reach optimal terrain coverage but, failed to prove the optimality of task completion time for their algorithm.

Hazon et al. [5] have focused in a newer research paper on the robustness of their solution by ensuring that in a multi-robot system, the single robot should be fully capable of successfully solving the terrain coverage problem, though task completion time optimization was not the focus here. The solutions discussed in their work were online solutions were the robots do not have any prior knowledge of their environment. For that reason, each robot should be fully equipped to carry out the full task even when other robots within the system become incapacitated for various reasons.

Another algorithm that involves area coverage in forwards and backwards motion stages, was proposed by Kong et al. [6]. Their proposed algorithm is multi-robot compatible, in which the robots have real time knowledge of the discovered area, the edges of the undiscovered area, and their positions within the area. By utilizing these pieces of data, the particles can navigate through the work area accordingly to maximize the discovered portions. Under the assumption of global communication, the robots divide the terrain into cells of equal width and each particle gets assigned a cell to cover. The algorithm benefits from data sharing between the robots, however, the robots in their research rely on wireless communication for data sharing. This can prove to be very challenging in hazardous terrains with high structures as mountains or hills that can interfere with the wireless signals sent between the robots. This algorithm gives efficient results in less complex terrains.

Deterministic approaches have been explored to cover specialized environments [7], which in various can result in repeated coverage of already discovered portions of the terrain [8], [9], [10]. In addition, two more approaches experimenting with the effects of robot communication capabilities in [11], [12] have been proposed, that ensure total terrain coverage. On the other hand, non-deterministic coverage approaches involving the use of chemical traces [13], the use of neural networks [14], as well as the use of swarm intelligence [15], [4], [16], do not guarantee complete terrain coverage.

Zheng X. [17] has introduced an algorithm for terrain coverage that produces a performance boost of a maximum of eight times the optimal cost, provided that the terrain is previously known to the robots that are employed to cover it.

Several research papers were published, that based multi-robot terrain coverage strategies on market trading principles [18], [19]. The concept that these researchers were exploring revolves around the robots placing bids on sub-tasks of their main coverage task. The bids are usually based on useful values to the robots, such as the expected gain and travel cost overhead for navigating to a certain location within the terrain. The researches have adapted the bids to be assigned in a distributed manner [19], or using a central control mechanism [18]. The market based strategies aim to maximize gain while reducing cost overheads. In this approach, the robots make use of a goal scoring system of heuristics to generate a list of target locations to navigate to within the terrain. Before starting any trip, the robot uses the system of heuristics to generate a list of possible target locations to visit, and saves these targets in memory in a schedule/priority queue-like data structure. Each robot repeats this process until its memory is full. Subsequently, the robots iteratively participate in single-award multi-task auctions to filter their targets; robots remove the targets that are better suited for other robots. The auctions keep occurring until each robot is left with the most suitable targets in its memory. This approach produces efficient results even if some members of the robot swarms are out of the collective communication range. The author further developed the concept to include communication strength and reliability into the bidding system, which resulted in the robots avoiding targets that would take them out of the collective communication range of the swarm. This can be extremely beneficial

in swarm systems with strict communication constraints, as in such systems, communication is the key to task completion.

III. Methodology

A. Definitions

Particle (Mobile Agent): Let P be the set of particles in a swarm. Each $p \in P$ is a unique positive natural number that represents the *ID* of a particle.

$$P = \{p \mid p \in \mathbb{N}^+,\ p_i \neq p_j\}\ with\ 0 < |P| < \aleph_0$$

Coordinate: Let c be a coordinate pair.

$$c = (x, y)\ with\ x \in \mathbb{R}\ and\ y \in \mathbb{R}$$

Hop: Let h be a constant representing the distance measurement unit in this study. Particles in this study have the ability to discover, interact with, and move to coordinates that are at a maximum distance of $h = 1$ from their current coordinates.

Direction: Let DIR be a set of directions. Each $dir \in DIR$ represents a unique natural number in range of 0 to 5, that refers to a static x,y coordinate pair representing the x and y offsets needed to calculate the coordinates at a distance of $h = 1$ in that specific direction. For this study, each particle is equipped with a compass to calculate its bearing. Particles in this research are restricted to recognize only 6 distinct directions; East, Southeast, Southwest, West, Northwest, and Northeast. For each particle $p \in P$:

$$DIR_P(p) = \{E, SE, SW, W, NW, NE\}$$

Node: A node n is a virtual object that a particle can create and save in its own memory. A node can be created using any coordinate pair c as reference. Nodes act as place holders for coordinates discovered by particles within the terrain. In creating nodes, a particle can generate a map of all discovered coordinates within the terrain. A node can only be created on unobstructed coordinates that a particle has physical access to.

Map: A Map is the set of nodes saved in a particle's memory. The generated Map of nodes is the virtual image mirroring the floor structural composition of terrain. Let MAP be the set of all nodes created by a particle $p \in P$. Each $n \in MAP$ is a Node object created using some coordinate pair c. For each particle $p \in P$:

$$MAP_P(p) = \{n \mid n\ is\ a\ Node\ object,\ n_i \neq n_j\}$$

Due to the direction recognition restrictions in the particle's logic, particles can create nodes on their current coordinates, as well as, on six accessible adjacent coordinates in the 6 predefined directions. This results in a generation of an isometric grid like representation of the environment, with uniformly spaced nodes at a distance of $h = 1$ between any 2 adjacent nodes.

Current Node: A current node of a particle is the node where the particle is physically located on. For each particle $p \in P$, let $current_P(p)$ be the particle's current node.

Next Node: A next node is the node which a particle chooses to move to in its following iteration. For each particle $p \in P$, let $next_P(p)$ be the particle's next node.

Fig. 1: Isometric grid

Target Node: A target node is a node which a particle needs to reach but is not adjacent to its current node. Once a Target Node of a particle is assigned, it remains unchanged until the particle moves onto it. Only then does the target node get unassigned. For each particle $p \in P$, let $target_P(p)$ be the particle's target node.

Stuck Node: Stuck Node is the node at which a particle gets stuck behind a border or obstacle. For each particle $p \in P$, let $stuck_P(p)$ be the node at which the particle got obstructed by an obstacle.

Unmarked Nodes Set: The Unmarked Nodes Set contains a **sequence** of nodes in the order they were added to the set. The nodes in the unmarked nodes set are nodes that have been discovered and created but not yet physically visited by the particles. For each particle $p \in P$, let the set of Unmarked Nodes be:

$$UN_P(p) = \{\ un_i \mid un \in MAP_P(p), \tag{1}$$

$$i \in \mathbb{N}^+,\ 1 \leq i \leq |MAP_P(p)|\ \} \tag{2}$$

Note that the order with which the nodes are added to UN is crucial to the livelihood of the terrain marking algorithm.

Marked Nodes Set: The Marked Nodes Set contains the nodes that have been physically visited by a particle. For each particle $p \in P$, let the set of marked nodes be:

$$MN_P(p) = \{mn \mid mn \in MAP_P(p) \setminus UN_P(p)\}$$

Initial Particle Configuration: Each particle is activated with a set of objects in its memory. For each particle $p \in P$, let the initial configuration $CONFIG_P(p)$ be: a set of directions $DIR_P(p) = \{E_1, SE_2, SW_3, W_4, NW_5, NE_6\}$, an unassigned current node $current_P(p) = c$, an unassigned next node $next_P(p)$, an empty set of all nodes $MAP_P(p) = \{\}$, an empty set of unmarked nodes $UN_P(p) = \{\}$, and an empty set of marked nodes $MN_P(p) = \{\}$.

B. Discovering Adjacent Coordinates

Each particle can discover coordinates at the 6 predefined directions surrounding its current node. The particle can discover coordinates by creating nodes with said coordinates in a specific direction, at a distance of $h = 1$ from its current node. The steps are proposed in algorithm 1.

Algorithm 1 Discover Adjacent Coordinates

Remark. *The creation of a node by a particle implies the particle adding the created node to its $MAP_p(p)$.*

Require: Initial Particle Configuration $CONFIG_P(p)$

1) Create node 1 at direction (E) and add it with its coordinates to $UN_P(p)$ with index 1
2) Create node 2 at direction (SE) and add it with its coordinates to $UN_P(p)$ with index 2
3) Create node 3 at direction (SW) and add it with its coordinates to $UN_P(p)$ with index 3
4) Create node 4 at direction (W) and add it with its coordinates to $UN_P(p)$ with index 4
5) Create node 5 at direction (NW) and add it with its coordinates to $UN_P(p)$ with index 5
6) Create node 6 at direction (NE) and add it with its coordinates to $UN_P(p)$ with index 6

Algorithm 2 Breadth First Marking (BFM)

Remark. *The creation of a node by a particle implies the particle adding the created node to its $MAP_p(p)$.*

Require: Initial Particle Configuration $CONFIG_P(p)$

1) Create node c at current coordinates, and add it to $MN_P(p)$
2) Set $current_P(p)$ to c
3) Discover adjacent coordinates to $current_P(p)$ as demonstrated in Algorithm 1
4) Set $next_p(p)$ to the **First** node in $UN_P(p)$; node $n_{\min(i)} \in UN_P(p)$ (In this iteration it would be node 1 on the diagram)
5) Move to $next_p(p)$
6) Set $current_P(p)$ to $next_p(p)$
7) Mark $current_P(p)$ by removing it from $UN_P(p)$ and adding it to $MN_P(p)$
8) Repeat step 3, 4, 5, 6, and 7 until all $UN_P(p)$ is empty.

C. Breadth First Marking (BFM)

Breadth First Marking or BFM is, as the name implies, a modified version of the BFS algorithm's navigational pattern. When a particle is using BFM to traverse through the grid, the particle would navigate to all the adjacent nodes of its current node before moving on the next node. In doing so, a particle would completely explore all available nodes at a level of 1 hop before moving to one of the nodes in the first hop level and exploring nodes at the level of the second hop. In this solution, particles can only move to preexisting nodes in their memory. A particle's movement is determined by the particle's choice of its next node. In BFM, a particle $p \in P$ with a current node of $current_P(p)$ traverses across and expand through the environment by moving to nodes in its set of unmarked nodes $UN_P(p)$. The implementation of BFM is a combination of the *Breadth First Search (BFS)* algorithm and the *First In First Out (FIFO)* method where the particle chooses the first node in its set of unmarked nodes $UN_P(p)$ to navigate to. This ensures that the particle stays at one hop level away from its current node choosing another hop level to explore, hence, an adapted Breadth First movement is applied in the sense that each hop level is exhausted till its very end.

Breaking down the demonstration in figure 2, BFM proceeds as proposed in Algorithm 2

D. Depth First Marking (DFM)

Depth First Marking or DFM is, as the name implies, a modified version of the DFS algorithm's navigational pattern.

Algorithm 3 Depth First Marking (DFM)

Remark. *The creation of a node by a particle implies the particle adding the created node to its $MAP_p(p)$.*

Require: Initial Particle Configuration $CONFIG_P(p)$

1) Create node c at current coordinates, and add it to $MN_P(p)$
2) Set $current_P(p)$ to c
3) Discover adjacent coordinates to $current_P(p)$ as demonstrated in Algorithm 1
4) Set $next_p(p)$ to the **Last** node in $UN_P(p)$; node $n_{\max(i)} \in UN_P(p)$ (In this iteration it would be node 6 on the diagram)
5) Move to $next_p(p)$
6) Set $current_P(p)$ to $next_p(p)$
7) Mark $current_P(p)$ by removing it from $UN_P(p)$ and adding it to $MN_P(p)$
8) Repeat step 3, 4, 5, 6, and 7 until all $UN_P(p)$ is empty.

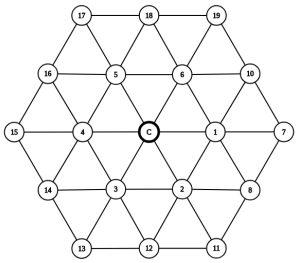


Fig. 2: A demonstration of the order in which nodes are explored using BFM

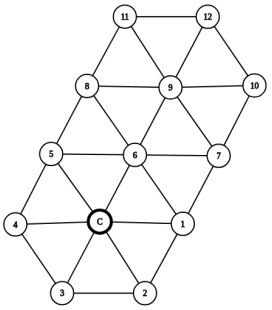


Fig. 3: A demonstration of the order in which nodes are explored using DFM

When a particle is using DFM to traverse through the map saved in its memory, the particle would navigate in one direction until it can no longer proceed in said direction, i.e., area border, or obstacle is reached. Once the particle can not go further in the direction it was in, it chooses a different direction, according to the sequence of directions stored in its memory, and follows it till the next border/obstacle and so on and so forth. In this solution, particles can only move to preexisting nodes in their memory. A particle's movement is determined by the particle's choice of its next node. In DFM, a particle $p \in P$ with a current node of $current_P(p)$ traverses across and expands through the environment by moving to nodes in its set of unmarked nodes $UN_P(p)$. The implementation of DFM is a combination of the *Depth First Search (DFS)* algorithm and the *Last In First Out (LIFO)* method where the particle chooses the last node in its set of unmarked nodes $UN_P(p)$ to navigate to. This ensures that the particle stays in one direction until it can not proceed any further, before choosing another direction to follow. Hence, an adapted Depth First movement is applied in the sense that each direction is exhausted till its very end.

Breaking down the demonstration in figure 3 , the DFM proceeds as proposed in Algorithm 3.

E. Optimized Next Node Selection

While in discovery mode, node selection achieved using DFM and BFM by particles, works with no redundancy when for some particle $p \in P$:

Algorithm 4 Optimize Next Node Selection

Remark. *This algorithm assumes that the particle is about to chose a new $next_P(p)$ to navigate to.*

Require: Pre-initialized Particle Configuration $CONFIG_P(p)$

function
  if Next node in the sequence provided by DFM or BFM, is more then 1 hop away from $current_P(p)$ **then**
    Set $next_P(p)$ to any of the adjacent unmarked node $un \in UN_P(p)$, to $current_P(p)$ ▷ Optimized return
  else
    Set $next_P(p)$ to the node provided by the DFM or BFM algorithms.
  end if
end function

- For DFM: The **Last** node $n_{\max(i)}$ in $UN_P(p)$ is adjacent to the $current_P(p)$
- For BFM: The **First** node $n_{\min(i)}$ in $UN_P(p)$ is adjacent to the $current_P(p)$

However, in the case where the chosen node of the particle is not directly accessible from the current node, the particle would have to navigate through already marked nodes in order to get to its desired unmarked node, hence, the default implementation causes redundancy and decreases particle efficiency. For this reason, the selection process is modified to increase efficiency by enabling the particle to navigate to the nearest node in its set of unmarked nodes $UN_P(p)$ instead of navigating to the node provided by the DFM or BFM algorithms. The steps are proposed in algorithm 4

F. Reaching Non-Adjacent Nodes

In some cases, particles may select destination nodes that are not directly accessible from their current nodes, i.e., non adjacent nodes to the current node. If such cases occur, the particle $p \in P$ would set its target node $target_P(p)$ to the desired node, and calculate the bearing from its current node to its target node using its compass. Once the particle has calculated the bearing of its target node from its current node, it uses this bearing to select the node from its adjacent nodes, that has the direction referenced by the bearing. If found, this node would take the particle one hop closer to its target. The particle keeps repeating this process in an iterative manner, until the target node is reached. The steps are proposed in algorithm 5

G. Obstacle Avoidance

Particles may encounter borders or obstacles in their way while they are trying to reach a remote node. The particle detects the obstacle using the bearings concept. If a particle is moving towards its target node, it calculates the bearing to that target node and chooses the node from its adjacent nodes that has the direction referenced by the bearing. However, if the bearing does not refer to anything in the set of adjacent nodes of the current node, this indicates that there is no node created in that direction; meaning that that direction is occupied by a border or obstacle. When the particle has detected that its path is blocked, it then proceeds with the obstacle avoidance algorithm.

Algorithm 5 Reach Non-Adjacent Node

Remark. *This algorithm assumes that the particle has already set its $next_P(p)$ to a node that is at a distance of $h > 1$ from its $current_P(p)$*

Require: Pre-initialized Particle Configuration $CONFIG_P(p)$

1) Set $target_P(p)$ to $next_P(p)$
2) Calculate the bearing of $target_P(p)$ from $current_P(p)$
3) Determine the appropriate directions corresponding to the calculated bearing
   - If bearing is North, then the appropriate directions would be NE and NW
   - If bearing is South, then the appropriate directions would be SE and SW
   - If bearing is East, then the appropriate directions would be E
   - If bearing is West, then the appropriate directions would be W
   - If bearing is Northeast, then the appropriate directions would be NE
   - If bearing is Northwest, then the appropriate directions would be NW
   - If bearing is Southeast, then the appropriate directions would be SE
   - If bearing is Southwest, then the appropriate directions would be SW
4) Set $next_P(p)$ to one of the adjacent nodes to $current_P(p)$, that are located in the direction corresponding the bearing calculated in step 3
5) Move to $next_P(p)$
6) Set $current_P(p)$ to $next_P(p)$
7) Repeat steps 2, 3, 4, 5, and 6 until $current_P(p) = target_P(p)$
8) Unset $target_P(p)$

Algorithm 6 Obstacle Avoidance

Remark. *This algorithm assumes that the particle is making navigating towards its $target_P(p)$, and has encountered an obstacle in its trajectory.*

Require: Pre-initialized Particle Configuration $CONFIG_P(p)$

1) If path is blocked, set $stuck_P(p)$ to $current_P(p)$
2) Calculate bearing from $stuck_P(p)$ to $target_P(p)$ (bearing 1)
3) Move to any adjacent node that is touching also the wall (following wall)
4) Calculate bearing from $current_P(p)$ to $stuck_P(p)$ (bearing 2)
5) Move to any adjacent node that is touching the wall, that the particle has not already moved to since setting $stuck_P(p)$ (This ensures movement in the same trajectory)
6) Repeat steps 3, 4, and 5 until bearing 2 = opposite of bearing 1 (180 degrees turn has been made)
7) Unset $stuck_P(p)$
8) Continue in original trajectory

The obstacle avoidance algorithm builds up on a simple wall following algorithm, where the particle follows and rotates around the wall (border/obstacle) until a 180 degrees rotation around the wall has been made, then continues on with its original trajectory. The steps are proposed in algorithm 6

H. Communication

Communication is the core element of swarm success. In this solution, communication is carried out locally in a decentralized manner between particles. Each particle has its local view and interpretation of its environment, and can share this local view with other particles as well as receive the local views of other particles. In doing so, particles can build a more complete map of their environment without the need for

a centralized control unit or synchronization mechanism.

Particles have the ability to communicate whenever they come within range of one another. In this solution, the communication range and frequency are both customizable and can be configured as desired for the swarm as a whole. The communication range is measured in Hops; meaning that particles can communicate if they come in range of n Hops, with $n \in \mathbb{N}^+$. The communication frequency is measured in time, meaning that the particles have the option to communicate, if they are within range, every m units of time, with $m \in \mathbb{N}^+$.

A particle $p \in P$ can share/extend its local view of the environment, by sending and receiving packets containing a copy of $MAP_P(p)$, a copy of $UN_P(p)$, and a copy of $MN_P(p)$.

In this study, the communication range and frequency of particles is not restricted; meaning that, particles have indefinite scanning range, and they are allowed to communicate at any given point in time, if they are within range of one another The steps are proposed in algorithm 7

I. Terrain Marking Solution

Following the descriptions of particle memory elements, abilities and functionalities in the previous sections, the modularized solutions can be implemented. For this study, the terrain coverage and marking solution; **Terrain Marking**, is a multi-particle compatible solution with local communication capabilities between the particles. The provided solution provides a fully functional terrain marking algorithm that enables each particle to achieve individual success, leading to swarm success. In this solution, each particle can choose between DFM and BFM at random at the initialization stage. The steps are proposed in algorithm 8. Particles can start communicating with one another only after a configurable amount of time units have passed. This ensures that the particles can spread out within the terrain before sharing their local views. Let t_{noComm} be a configurable value after which the particles are allowed to communicate.

IV. Evaluation

In this section, the collaborative performance of particles using the three marking algorithm possibilities (DFM, BFM, or both) are evaluated. The focus of this section of the evaluation is to assess the collective performance of the swarm, therefore, the simulations for this section are ran with **multiple particles** working under different environmental conditions.

Goals: Collaborative Performance Assessment: The main aim of the evaluation is to study the effect of increasing the number of particles on task completion time, in rounds. The simulation setup is run with particle group sizes of 1 to 6 particles. Four different terrain variants are involved in the collaborative performance evaluation.

A. Simulator

For the evaluation of the marking algorithm the round-based simulator **Swarm Sim** [20] has been used. This simulator is created by the "Technology of Social Networks

Algorithm 7 Communicate With Particles

Require: Pre-initialized Particle Config. $CONFIG_P(p)$, and Range r

1) Scan for other particles within range r
2) If other particles are found, then for each particle $p_{other} \in P$ found:
   a) Send Packet to p_{other}
   b) Receive Packet from p_{other}

Algorithm 8 Terrain Marking

Remark. *The creation of a node by a particle implies the particle adding the created node to its* $MAP_p(p)$.

1: Set of Particles P,
2: Initial Particle Configuration $CONFIG_P(p)$
3: Hop Value $h \in \mathbb{R}^+$,
4: $t_{noComm} \in \mathbb{N}^+$
5: Communication Range $r \in \mathbb{N}^+$,
6: Communication Frequency $f \in \mathbb{N}^+$
7: **For each particle** $p \in P$ **do:**

1) Initialize $CONFIG_P(p)$
2) Choose between DFM and BFM at random
3) Create node at current coordinates and set $current_P(p)$ to created node
4) Discover adjacent coordinates and create nodes representing them according to Algorithm 1
5) If time units passed since initialization of $CONFIG_P(p) \geq t_{noComm}$:
   - If time units passed since last communication $\geq f$, communicate with particles within range r according to Algorithm 7
6) Get next node:
   - If DFM was chosen, choose the last node in $UN_P(p)$ according to Algorithm 3
   - If BFM was chosen, choose the first node in $UN_P(p)$ according to Algorithm 2
7) Optimize next node selection according to Algorithm 4
8) Set $next_P(p)$ to the node selected in the previous step
   - If $next_P(p)$ is adjacent to $current_P(p)$:
     a) Move to $next_P(p)$
     b) Set $current_P(p)$ to $next_P(p)$
     c) Mark $current_P(p)$ by removing it from $UN_P(p)$ and adding it to $MN_P(p)$
     d) Discover adjacent coordinates and create nodes representing them according to Algorithm 1
   - If $next_P(p)$ is not adjacent to $current_P(p)$:
     a) Set $target_P(p)$ to $next_P(p)$
     b) Set $next_P(p)$ to the appropriate adjacent node according to Algorithm 5
     c) Move to $next_P(p)$
     d) Set $current_P(p)$ to $next_P(p)$
     e) Mark $current_P(p)$ by removing it from $UN_P(p)$ and adding it to $MN_P(p)$
     f) Discover adjacent coordinates and create nodes representing them according to Algorithm 1
     g) Repeat steps b, c, d, e, and f until $current_P(p) = target_P(p)$
   - If an obstacle is encountered:
     a) Set $stuck_P(p)$ to $current_P(p)$
     b) Set $next_P(p)$ to the appropriate adjacent node according to Algorithm 6
     c) Move to $next_P(p)$
     d) Set $current_P(p)$ to $next_P(p)$
     e) Mark $current_P(p)$ by removing it from $UN_P(p)$ and adding it to $MN_P(p)$
     f) Discover adjacent coordinates and create nodes representing them according to Algorithm 1
     g) Repeat steps b, c, d, e, and f until a 180 degree turn has been made around the obstacle
9) Repeat steps 4, 5, 6, 7, and 8 until $UN_P(p)$ is empty

Lab" of the Heinrich Heine University in Düsseldorf/Germany. Swarm Sim is a 2D and 3D simulation core that provides interfaces for creating individual scenarios and solutions with the main focus being the simulation of virtual swarms. This simulator is open source and can be downloaded from [21], the marking algorithm is available from [22]. For the simulation the following mathematical abstracts are used:

Rounds In swarm-sim the time T is modeled by the set of ordered rounds $\overline{R}$. Let R be a set of rounds. Each $r \in R$ is a natural number than can go to the maximum of defined round number called $\hat{r}$.

$$R = \{r \mid r \in \mathbb{N}_0, r \leq \hat{r}\}$$

Rounds is a totally ordered set of numbers which are monotonically increasing. Let $\overline{R}$ be the set of totally ordered rounds.

$$\overline{R} = (R, <)$$

Location: A location l is a point on the simulator grid, on which an agent is placed. As the coating algorithm is designed for two dimensional (2D) spaces, the location l is a 2D vector s.t. $\mathbb{R}^2$:

$$l = \begin{cases} (x, y) & \text{for y is even} \\ (x + 0.5, y) & \text{for y is odd} \end{cases}$$

We then define L to be the set of all locations that build up together the two-dimensional plane area, which enables the definition of proximity.

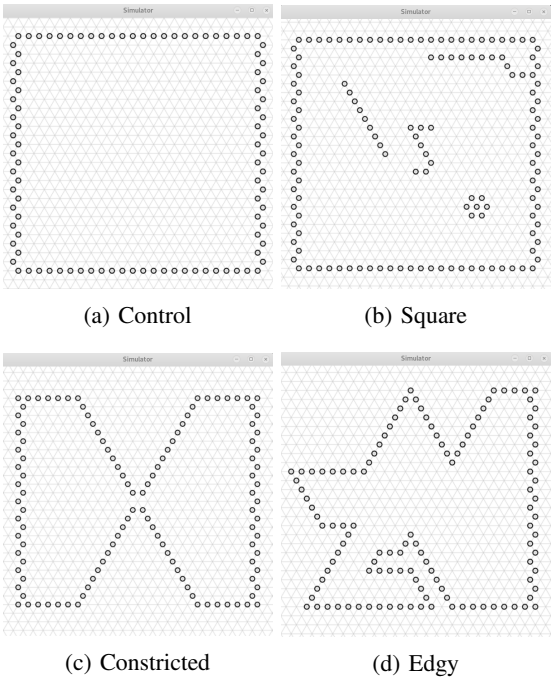
Neighbor: Each coordinate pair is connected to a maximum of $\hat{n}$ adjacent coordinates, termed Neighbors. Let N_l be an index set of neighbor locations for every coordinates l s.t. $N_l = \{n_j \mid l \in L, n_j \in L \setminus l, 1 \leq j \leq \hat{n}\}$. We assume a uni-distance model, in which the distance between each coordinates is constant (1) and inbetween the diameter of one and two agents.

B. Simulated Terrains

1) Control: The control terrain is a square shaped empty area with no constrictions, no obstacle, no sharp edges, and no enclosed areas. The square terrain is the largest in size in comparison to the other 3. This terrain has the following **features**. It contains 563 valid coordinates, is plain and completely empty inside.

2) Square: The square terrain is a square shaped area with scattered obstacles inside. The square terrain has the exact same dimensions as the control terrain described in the previous section, however, the square terrain contains fewer valid coordinates due to the presence of obstacles that use up some of the valid coordinates. This terrain has the following **features**. It contains 527 valid coordinates, obstacles exist and a very tight enclosed space is located in the top right corner.

3) Constricted: The constricted terrain is a medium sized closed area with the following **features**. It contains 363 valid coordinates and a very tight construction at the center splits the area in two halves.



(a) Control (b) Square

(c) Constricted (d) Edgy

Fig. 4: Terrain Shapes

4) Edgy: The edgy terrain is a medium sized closed area with the following **features**. It contains 321 valid coordinates, it has multiple sharp edges and an enclosed area at the bottom.

C. Simulation Setup

The simulation set up is as follows. We use 1 to 6 Particles in all terrains and 20 seeds for each setup. As marking algorithms we consider BFM, DFM and Mixed, where particles choose their algorithm at random.

D. Results

In total, 540 simulations were ran, with 180 simulations for each of marking algorithm possibilities; BFM, DFM and Mixed. The result can be seen in figures 5, 6, 7, and 8.

As demonstrated, all three marking algorithm possibilities showed a similar level of performance for the control terrain, with DFM scoring marginally worse results than the BFM and Mixed. From the data gathered for this section of the evaluation, it can be inferred that all three marking algorithm possibilities perform on a similar level in all terrains, efficiency wise. The rate of performance efficiency decreases logarithmically between particle group sizes of 1 to 6. Particle group sizes score better results but the efficiency growth rate levels off in to a linear growth pattern.

V. CONCLUSION

The solution developed for this research complies with the swarm robotics systems criteria defined in the introductory section. The solution is **robust**, as each robot/particle in the swarm system is fully capable of single handedly covering and marking any terrain as demonstrated in the demonstrated evaluation in the previous chapter. This implies that in a real world application, the proposed solution would not hinder task

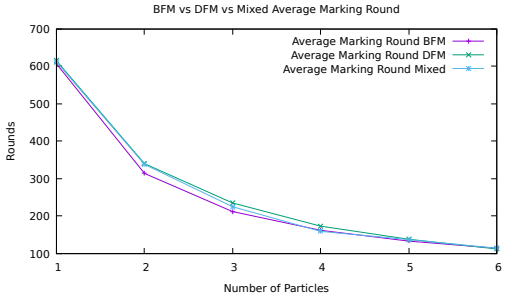


Fig. 5: Control - Marking Rounds

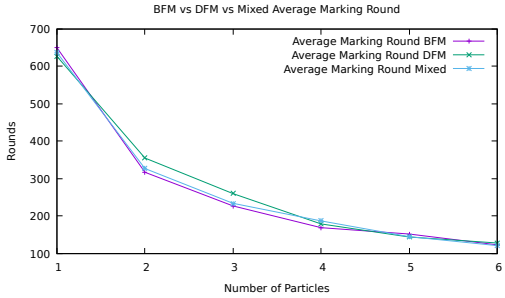


Fig. 6: Square - Marking Rounds

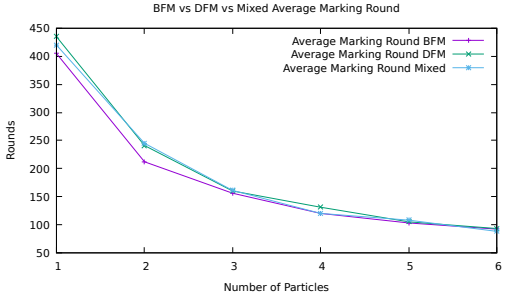


Fig. 7: Constricted - Marking Rounds

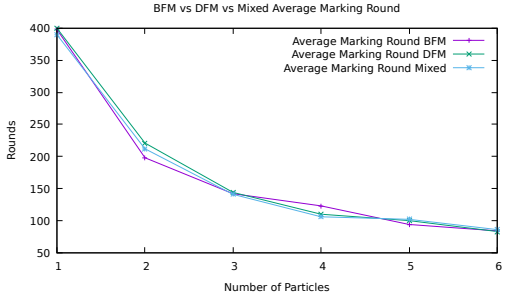


Fig. 8: Edgy - Marking Rounds

(a) 1 to 6 Particles - BFM vs DFM vs Mixed

completion if part, or most of the swarm members become incapacitated or damaged, instead, the rest of the swarm members carry on with completing the task at hand. The proposed solution is **flexible**, as the particles were able to successfully cover and mark every terrain they were assigned to, hence, the swarm system is capable of functioning under a wide range of working conditions. Finally, the solution is **scalable**, as the particles/robots are fully autonomous from start to end. The particles are capable of self organization and path planning. Communication is carried out on a local level between the particles, without the need for a central guidance system or remote control unit. In the previous chapter, the empirical data demonstrated that the solution is functional with various particle groups sizes, showing increased performance levels as the group sizes grow.

The proposed solution enables the particles to discover and spread through their environment using two distinct marking algorithms; Breadth First Marking (BFM) and Depth First Marking (DFM). BFM allows the particle to pivot around its current location, discovering all the coordinates in a rotational motion at a certain radius from its pivot, before increasing the radius length and moving on to a further depth level. DFM allows the particle to move as far away as possible from its current location by following one direction till the border of the terrain is reached, before choosing another direction and exhausting it till its end, and so on. Particles have the ability to make optimized choices for their next moves depending on their position within the terrain. Particles possess integrated obstacle avoidance logic that enables them to rotate around an obstacle by 180 degrees and then continuing on with their original trajectories. Particles in the proposed solution are capable of communicating with one another on a local level. Each particle discovers new nodes and creates a map of its discovered nodes, then shares this map with other particles if it comes within range. In doing so, each particle can build a complete map of their environment without having to physically visit every node within the terrain. The solution was developed so that no particle remains idle while there are still undiscovered locations in the terrain. Particles are always attracted to undiscovered areas and move as long as the task is not completed.

The effectiveness of the proposed solution is validated in the previous section, where the analysis of empirical data gathered from running a high number of simulations, showed how the solution enables particles to successfully complete the terrain coverage and marking task. The data evaluation demonstrated that the solution is compatible with single particle and multiple particle scenarios where the particles have successfully completed the terrain coverage task in over 500 simulations ran.

VI. Future Work

The proposed solution is evaluated with the particle communication range and frequency set to maximum values; meaning that the particles can communicate every round as they are constantly in range of one another. This is however not applicable to real world scenarios involving vast terrains or terrains containing tall structures that might hinder wireless communication. To account for loss of communication, the solution should be tested for using communication capability limitations. Communication range and frequency can be tested by defining a domain for each, and testing different combinations out of the two domains, such as high frequency low range or vice versa. The communication range and frequency combinations can be tested in a systematic fashion by running

very high numbers of simulations with different combination values. Through this we can find the optimum combination values for a given terrain. With more time, even a universal combination might be discovered that would generate optimal results in any terrain.

In the proposed solutions, particles share all data in their memory indiscriminately. As an optimization suggestion, the solving algorithms can be further developed to include a filtering system that would enable optimized data transfer where each particle would receive only the data that is best suited to the particle's position within the terrain, eliminating part or most of completion time redundancy. Here we could apply communication priority approaches as in [23], [24]. In Section II, the concept of market trading application to terrain coverage strategies is introduced. The approach involves the implementation of a bidding system that allows the particles to bid on target locations in synchronized auctions between the particles. Each particle makes an educated decision before placing a bid by using a goal scoring heuristics system. The particle decision is based on factors such as the expected gain or the travel cost overhead of navigating to a certain location within the terrain. During these synchronized auctions, the particles filter out the targets that would result in redundancy and only accept targets that would allow for optimal performance. This approach can be beneficial in reducing task execution time if integrated into the proposed solution.

The obstacle avoidance algorithm is simple and effective but, in some cases, produces sub-optimal results. In the proposed solution, when a particle detects an obstacle, it chooses a random direction to rotate around the obstacle without a clue which path around is fastest. As an optimized approach, particles could identify and remember critical points in the terrain such as corners or bottle neck entrances to enclosed areas. In doing so, the particle would have a reference point when detecting an obstacle, with that reference point being the critical point associated with the obstacle.

The solution may, in some cases, produce sub-optimal results when repeated coverage occurs. Assume particle a and particle b never had the chance to communicate and particle a already covered a certain area of the terrain and moved away; if particle b visits this already covered area, it has no possibility of knowing that this area has been already marked, thus resulting in repeated coverage and increased redundancy. A simple work-around this problem would be indirect communication between the particles, where the particles can leave pheromone-like markers on the terrain floor, so that they can learn if another particle has already been to that part of the terrain. This would naturally eliminate redundancy and result in increased performance efficiency.

ACKNOWLEDGMENTS

We would like to thank the Arab-German Young Academy of Sciences and Humanities for funding this research as well as our collaboration partner Dr. Ahmed Khalil for the valuable discussions.

REFERENCES

[1] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," in *Procof the IEEE Int. Conf. on Robotics and Automation*, vol. 2, May 2001, pp. 1927–1933.

[2] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proceedings of the 2005 IEEE Int. Conf. on Robotics and Automation*, April 2005, pp. 735–741.

[3] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2006, pp. 1698–1703.

[4] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Eds. Tokyo: Springer Japan, 2002, pp. 373–382.

[5] N. Hazon, F. Mieli, and G. A. Kaminka, "Towards robust on-line multi-robot coverage," in *Proceedings 2006 IEEE Int. Conf. on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1710–1715.

[6] Chan Sze Kong, New Ai Peng, and I. Rekleitis, "Distributed coverage with multi-robot system," in *Proceedings of the IEEE Int. Conf. on Robotics and Automation,.*, May 2006, pp. 2423–2429.

[7] Z. Butler, A. Rizzi, and R. Hollis, "Distributed coverage of rectilinear environments," vol. Proc. of the Workshop on the Algorithmic Foundations of Robotics, 2001.

[8] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "An algorithm of dividing a work area to multiple mobile robots," vol. 1995 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1995.

[9] Latimer, Srinivasa, Lee-Shue, Sonne, Choset, and Hurst, "Towards sensor based coverage with robot teams," vol. IROS, 2002.

[10] T. W. Min and H. K. Yin, "A decentralized approach for cooperative sweeping by multiple mobile robots," vol. 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1998.

[11] I. Rekleitis, A. P. New, and H. Choset, "Distributed coverage of unknown / unstructured environments by mobile sensor networks," vol. 3rd MRS Workshop, 2005.

[12] ——, "Limited communication, multi-robot team based coverage," 2004.

[13] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, Oct 1999.

[14] Chaomin Luo and S. X. Yang, "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, Oct 2002, pp. 660–665.

[15] D. Bruemmer, D. D Dudenhoeffer, M. D Mckay, and M. Anderson, "A robotic swarm for spill finding and perimeter formation," 06 2019.

[16] S. Ichikawa and F. Hara, "Characteristics of object-searching and object-fetching behaviors of multi-robot system using local communication," in *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics.*, vol. 4, Oct 1999, pp. 775–781.

[17] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Aug 2005, pp. 3852–3857.

[18] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945 – 955, 2006.

[19] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 3, May 2002, pp. 3016–3023.

[20] A. R. Cheraghi, K. Actun, S. Shahzad, and K. Graffi, "Swarm-sim: A 2d & 3d simulation core for swarm agents," in *3rd Int. Conf. of Intelligent Robotic and Control Engineering (IRCE'20)*, 2020.

[21] A. R. Cheraghi and U. Duesseldorf. (2020) Swarm simulator. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim

[22] A. R. Cheraghi and A. Abdelgalil. (2020) Marking algorithm. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim/-/tree/marking

[23] S. Sati, C. Probst, and K. Graffi, "Analysis of Buffer Management Policies for Opportunistic Networks," in *25th Int. Conf. on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–8.

[24] A. Ippisch and K. Graffi, "Infrastructure Mode Based Opportunistic Networks on Android Devices," in *IEEE Int. Conf. on Advanced Information Networking and Applications (AINA)*, 2017, pp. 454–461.