Stigmergic navigation on an RFID floor with a multi-robot team

Ali Abdul Khaliq, Maurizio Di Rocco, Alessandro Saffiotti,

Abstract—Stigmergy is a mechanism that allows the coordination between agents through traces left in the environment. In this paper we discuss the use of stigmergy in minimalist multi-robotic systems, where robots have minimal sensing and processing capabilities. We present three case studies: building a globally optimal navigation map, building an obstacle map, and performing safe goal-directed navigation using these maps. Through stigmergy, multiple robots can seamlessly build a shared map in a coordinated way without any explicit communication. We show empirical validations of our results on a team of ePuck robots as well as in simulation.

Keywords—Stigmergy, RFID floor, Multi-robot systems, Decentralized coordination, Minimalistic robotics

1 Introduction

One of the many fascinating phenomena which can be observed in nature is stigmergy. The concept of stigmergy was introduced by Pierre-Paul Grassé [**?**] to describe a mechanism of coordination and indirect communication used by insects. Agents organize their individual or collective work through the bias of traces left in the environment. These traces are made of volatile chemicals called pheromones [**?**] that are released and detected by agents of the same species. These pheromones can be seen as an external memory whose contents are written by an agent and then read by a different agent, or by the same one at a later time, who uses this information to decide its behavior. Stigmergy allows a simple agent, or a collection of simple agents, to exhibit complex behavior without the need for internal memory. The mediation of environment ensures that tasks are executed in the right order, without any need for planning or for direct interaction between the agents [**?**].

Researchers have designed a number of successful algorithms using a stigmergic approach, addressing problems as diverse as combinatorial optimization, routing in communication networks, graph drawing and partitioning, exploratory data analysis, task allocation in a multi-robot system [**?**], [**?**], [**?**], [**?**]. Several work used stigmergy-based approach to address path planning problems in robotics. In [**?**] a multi-agent model is proposed that defines a distributed and asynchronous version of a BFS-like wavefront algorithm. In simulation, agents collectively forage in an unknown environment in search of resource and build paths between home and resources. In [**?**], an improved artificial potential field is proposed based on a genetic algorithm for path planning of mobile robots. An additional force is introduced for overcoming local minima. A pheromone-based potential field approach using RFID tags is also proposed by [**?**].

In this paper, we build upon the stigmergic approach proposed in [**?**], in which one or multiple robots can compute the optimal path to a target position even when the robots have no sense of global location and the target is outside from the range of the robot's sensor. The approach relies on an RFID floor, i.e., a floor with a dense grid of passive RFID tags buried under it, which the robots can use as a stigmergic medium to store information and communicate indirectly. This paper extends the previous work in three ways: first, we show how the paths in the RFID floor can be computed by multiple robots in a *coordinated* way;

• *A. Khaliq, M. Di Rocco and A. Saffiotti are with the Department of Science and Technology, AASS Cognitive Robotic Systems Lab, Örebro University, Sweden*
E-mail: ali-abdul.khaliq@oru.se

second, we show how to compute a second type of information on the RFID floor, namely *closeness to obstacles*; third, we show how a robot can effectively *navigate* on the RFID floor by combining the two information above.

The rest of this paper is organized as follows. Section 2 describes the above extensions. Section 3 presents the experimental setup, and Section 4 reports the experimental results. Finally, Section 5 concludes the paper.

2 Three Stigmergic Algorithms in Robotics

Our starting point is the stigmergic algorithm proposed in [**?**]. The algorithm builds a *distance map*, i.e., a navigation map which gives at each position the distance to a fixed goal point in the environment. The map is build directly into the read/write RFID tags embedded in an RFID floor, using only local sensing. This map can be used for navigating to that fixed goal by performing local gradient descent. Since the map is stored in the RFID tags, the robots do not need to keep it in their internal memory, nor to rely on self-localization to register their map with the environment. Therefore, the method is suitable for navigation of robots with only minimal sensing and computational resources. Moreover, multiple robots can seamlessly contribute to map building, and any robot can be added or removed at anytime during the process without the need of saving or restoring its internal state. Finally, since RFID tags are cheap and do not require power, the method is suitable to scale up to large environments. Approaches that use active devices have been proposed [**?**], [**?**] but their scalability remains unexplored.

In this section, we present three extensions of the above algorithm:

1) Building the distance map in a *coordinated* way, by letting each robot mark its own "territory" (algorithm 1);
2) Building an *obstacle map*, also in a coordinated way: the value at each tag of RFID floor is proportional to closeness to the nearest obstacle. This map can be used for obstacle avoidance and safe navigation (algorithm 2); and
3) *Safe navigation* on the built map towards the target position, by using a combination of the distance map and the obstacle map (algorithm 3).

In the first and second case (algorithm 1 and 2), robots write their individual "traces" into a dedicated block of the RFID tags during the map building.[1] These traces are used by the same or other robots to coordinate in map building process. By writing these traces into the RFID tags, robots mark their own territories in the environment. Robots avoid going into the territories of other robots by detecting their traces; this reduces the redundancy of robots in an area of environment. In algorithm 1 and 2 $Explore()$ realizes the exploration strategy: move straight until an obstacle is met, or the trace of another robot is detected; when this happens, rotate by a random angle. $ReadTag()$ reads the RFID tag in the cell currently under the robot. $NextTag(x)$ is true if the ID of the read tag x is different from the one read at the previous iteration. $IsEmpty(trace)$ checks if the current tag is in the territory of another robot or not. $WriteTag(x, k, trace)$ writes value k and robot's $trace$ in memory blocks of tag x.

```
Require: All tags initialized to ∞, robot starts at goal
distance_count ← 0
while Explore do
  x ← ReadTag()
  if NextTag(x) then
    distance_count ← distance_count + 1
    if distance_count > val(x) then
      distance_count ← val(x)
    else
      if IsEmpty(trace) then
        trace ← robot_id
      end if
      WriteTag(x, distance_count, trace)
    end if
  end if
end while
```

Algorithm 1: BuildDistanceMap()

Our third algorithm (algorithm 3) allows a robot to navigate safely to the target position: it reads the distance values in the RFID tags that were stored during map building by algorithm 1 and moves towards the tags that have the lower value; at the same time, it keeps

1. Typically, RFID tags have several blocks of storage that can be addressed individually.

Require: All tags initialized with $val = 0$
1: obstacle_count ← 0
2: **while** *Explore* **do**
3: x ← ReadTag()
4: **if** Obstacle() **then**
5: obstacle_count ← ∞
6: **end if**
7: **if** NextTag(x) **then**
8: obstacle_count ← obstacle_count - 1
9: **if** obstacle_count < $val(x)$ **then**
10: obstacle_count ← $val(x)$
11: **else**
12: **if** IsEmpty(trace) **then**
13: trace ← robot_id
14: **end if**
15: WriteTag(x, obstacle_count, trace)
16: **end if**
17: **end if**
18: **end while**

Algorithm 2: BuildObstacleMap()

a safe distance from obstacles by exploiting the obstacle distance values stored by algorithm 2. To simplify gradient descent, we assume that the robot may be equipped with multiple RFID readers – here, three. In algorithm 3, $Goal()$ checks if the goal tag has been reached. $ReadTag(reader_i, distance\_val)$ reads the memory block that contains the $distance\_val$, from the RFID tag that is in the range of $reader_i$. $LocalMinimum()$ detects if the robot is at a local minimum: if the current readings of all three readers are greater then the last readings. $PController(error)$ computes the rotation angle of the robot. $SetVel(speed, \theta)$ sets the speed and turning angle of the robots. The $threshold$ value gives the minimum desired clearance from obstacles; setting this value to zero will allow the robot to navigate alongside obstacles, while increasing it will make the robot stay away from the obstacles. When the threshold value is set to the maximum, the robot will navigate as far from the obstacles as possible, i.e., it will move on the edges of the Voronoi diagram of the free space.

Combining the distance map and the obstacle map is somehow similar to combining the goal attractive field and the obstacle repulsive fields in artificial potential field approaches, and it can similarly lead to the production of local minima. To cope with this situation, the robot's navigation is switched from using both obstacle and distance map to use only distance map when a local minimum is detected (line 11 in the algorithm).

Ensure: $distance_i$ eventually reaches to 0, i.e goal tag
1: $\theta \leftarrow 0$
2: **while** !$Goal()$ **do**
3: error ← 0
4: **for** $i = 0$ to 2 **do**
5: $distance_i$ ← ReadTag($reader_i$, $distance\_val$)
6: $obstacle_i$ ← ReadTag($reader_i$, $obstacle\_val$)
7: $\rho_i \leftarrow distance_i$
8: **if** $obstacle_i >=$ threshold **then**
9: $\rho_i \leftarrow obstacle_i \times distance_i$
10: **end if**
11: **if** LocalMinimum() **then**
12: $\rho_i \leftarrow distance_i$
13: **end if**
14: **end for**
15: **if** $(\rho_0 <= \rho_1)\&\&(\rho_0 <= \rho_2)$ **then**
16: error ← 60°
17: **end if**
18: **if** $(\rho_2 <= \rho_1)\&\&(\rho_2 <= \rho_0)$ **then**
19: error ← -60°
20: **end if**
21: θ ←PController($error$)
22: SetVel($speed$, θ)
23: **end while**

Algorithm 3: Navigate($threshold$)

3 PHYSICAL TESTBED

All cases mentioned above have been implemented in a physical test-bed that consists of a large apartment-like environment called "PEIS-Home2" and a set of ePuck[2] robots — see Figure 1. PEIS-Home2 is equipped with a 6 m×9 m parquet floor that hides a hexagonal grid of RFID tags. The grid contains about 1500 read-write HF Texas Instruments Tag-it® tags, spaced at 20 cm. Each ePuck is equipped with an M1 RFID tag reader and RFID antenna. The dsPIC microcontroller of ePuck has two UARTs. The first one is used for communication between the ePuck's microcontroller and an M1 RFID tag reader. The second UART is connected to a bluetooth chip which is used for communication to a PC. The tag reader is set to the maximum allowed baud rate (57600), and takes power from ePuck's battery in order to reduce the weight load. A circular RFID antenna is mounted around the robot. The radius of the antenna is selected in such a way that ePuck can avoid obstacles and other robots on its way

2. http://www.e-puck.org/

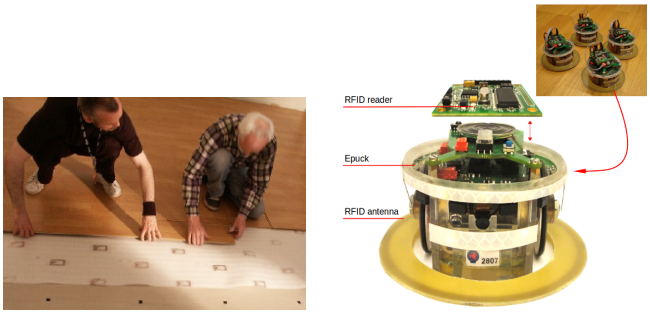


Fig. 1. The physical test-bed. Left: "PeisHome2" where the RFID tags placed under the floor. Right: the ePuck robot equipped with the RFID reader and antenna.

using the built-in IR proximity sensors. Each robot can read and write bytes of information from/to the RFID tags while it moves over them. This mechanism mimics the pheromone trails left by ants in one of the most famous example of stigmergy found in nature.

For experiments related to third case i.e, navigation on the built map, three RFID readers are installed on the robot to read the value of three neighbor tags. In addition to physical testbed, a simulator has been developed in order to perform quick and extensive tests.

4 Experiments and Results

The first experiment was performed with a set of four ePuck robots, and it was aimed at evaluating the distance-map building algorithm. Figure 2 shows a snapshot of the map building process after 18 hours. We calculate the error at time t between the built distance-map and the true distance map (ground truth) by equation 1:

$$Err(t) = \sqrt{\frac{1}{|X_t|} \sum_{x \in X_t} (v_t(x) - d(x))^2} \qquad (1)$$

where $v_t(x)$ denotes the value stored in tag x at time t and $d(x)$ denotes the true (geodesic) distance of tag x from the goal tag. Figure 3 is the plot of error function (1) over t during the building process. Full convergence required 18 hours, which is not surprising considering that the environment is about 12,000 times bigger than the size of a robot, and that the average speed of each robot was 5 cm/sec, so that even with an optimal strategy it would take about 2 hours for a robot to fully cover the environment — recall that the robots do not have any long-range sensors, so they need to physically visit each cell. In practice, it should be noted that the map obtained after 2 hours is sufficiently correct to perform nearly optimal navigation [**?**].

In order to complement the map building experiment performed in the real environment, four experiments were performed in simulation with four different goal positions. The screenshots of the built maps are shown in Figure 4. For visualizing the territories of robots, each robot's trajectory is drawn with different color during the map building process — see Figure 5.

The second experiment was performed in simulation, and it aimed at visualizing the results of building an obstacle map. The algorithm was run on 10 robots for about 7 hours in a static environment. The resulting map is shown in Figure 6.

The final experiment, also shown in simulation, was to navigate towards the target position taking both the distance and the obstacle map into account. The robot navigated according to algorithm 3 above. Figure 7 shows the results of the navigation using clearance threshold of one tag, of two tags, and maximum clearance, respectively.

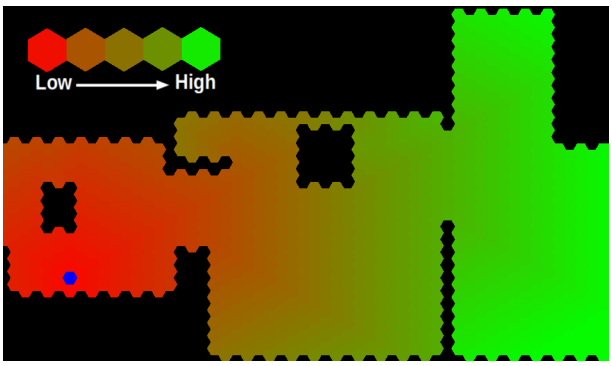


Fig. 2. Snapshots of distance map building after 18 hours. The blue tag is the goal position.

5 Conclusions

In this paper we have discussed stigmergic algorithms for mobile robots, focusing on three

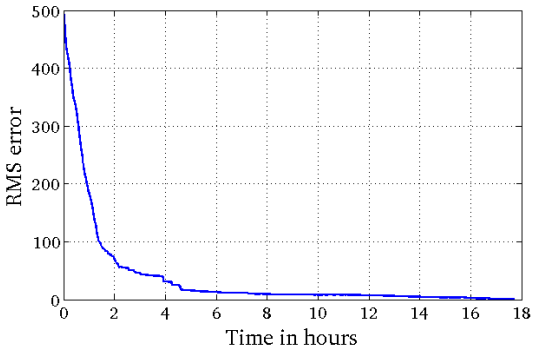


Fig. 3. RMS error during the building of the distance map.

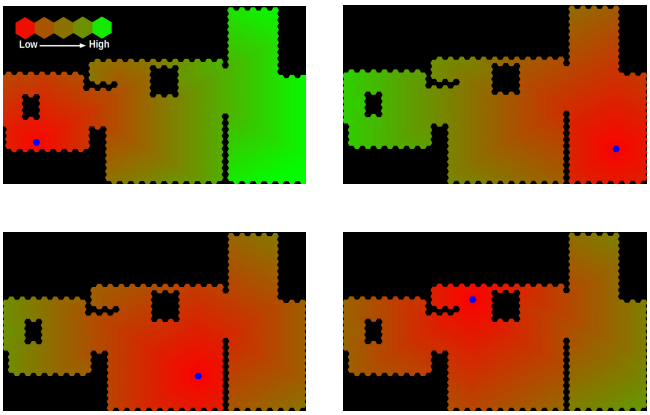


Fig. 4. Distance maps built for four different goal locations. The blue tags are the goal positions.

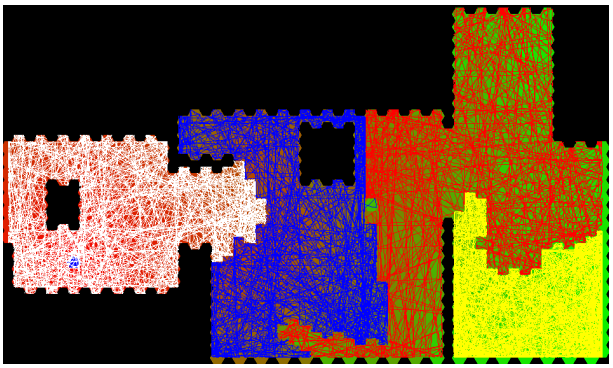


Fig. 5. Exploration trajectories and territories of four robots when concurrently building the above four distance maps.

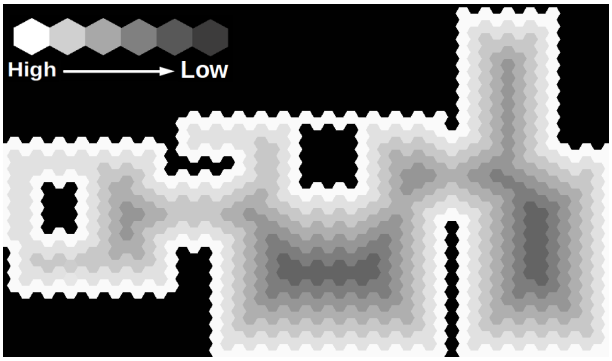
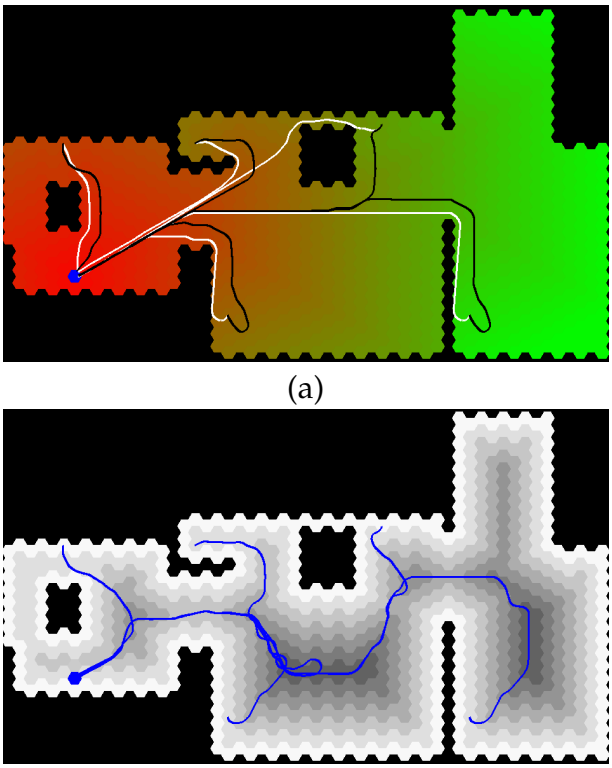


Fig. 6. Snapshot of the obstacle map.



(a)

(b)

Fig. 7. Snapshots of robot navigation considering the obstacle distance value. The blue tag is the goal position. a) White trajectories: clearance set to one tag; Black trajectories: clearance set to two tags. b) Safest possible navigation.

case studies. In the first case, coordinated multiple robots build a distance map leading toward some specific goal location. In the second case, coordinated multiple robots built an obstacle map for safe navigation. And in third case a robot navigates towards the goal using distance and obstacle maps. All three cases have been empirically evaluated through experiments on real platforms as well as experiments in simulations.

The stigmergic approach is well suited for minimalist robots with very basic sensor and processing capabilities. The standard methods for navigation involve the use of an internal

map, of path-planning, and of self-localization. To store, construct and maintain a map requires significant storage and computational resources [**?**]. Path planning is computationally expensive both in time and in space [**?**]. And accurate self-localization requires complex sensory equipment [**?**]. Our stigmergic algorithms, by contrast, only require local sensing (namely, the RFID tag reader and a collision detection sensor), constant time computation, and a few bytes of memory. Robots do not require any global localization for map building nor for navigation, since they simply rely on the local information sensed in the environment. The robots act as the moving computing elements of a spatially distributed memory co-extensive with the environment that it represents [**?**].

We are currently performing experiments in real apartments equipped with RFID floors, in order to assess the practical viability of these algorithms to help the navigation of domestic service robots in real situations.

ACKNOWLEDGMENTS

This work was supported by a strategic grant from Örebro University, Sweden. We thank the anonymous reviewers for their insightful comments and suggestions.