Using an Alternative Model in a Complex Environment for Nanorobotics Navigation

C. A. Piña-García† E. J. Rechy-Ramírez‡ V. A. García-Vega†

† Facultad de Física e Inteligencia Artificial, Universidad Veracruzana, México
capiga21@gmail.com, angegarcia@uv.mx

‡ Laboratorio Nacional de Informática Avanzada A.C.
erechy@lania.edu.mx

Abstract

Advances in nanotechnology have opened the space: exploration of unaccessible environments, microsurgery, and nanorobotics, with these advances come a growing need to develop control strategies suitable for a collective behavior.

Nanorobotics becomes an important field of study to face complex environments. Collective behavior and communication models have motivated our approach to study a multiple robot system within complex environment. In which, robots have to perform specific tasks without centralized control or explicit communication. Our approach focus on some known failures of communication models. This summarizes distinct aspects of communication models, and illustrates our synthetic model for mobile robots inside of a complex environment by means of our simulation tool. This paper essentially advocates the integration of communication models to front important challenges in exploration.

1 Introduction

Our present work describes a tool and a model to simulate an operative autonomous multi-robot team (nanorobots) to achieve collective tasks. In this collective work does not exist explicit communication. Our proposed nanorobots are displayed within a "complex environment", where mobile targets must be collected. By "complex environment" we mean places affected by a dynamics partially unforeseeable.

Multi-robot systems have showed to be more efficient and fault tolerant than single robot systems by its number of displayed robots in action (two robots at least). They are showed to be more cost-effective by its individual simplicity, their working configuration made them more flexible by its redundancy.

Our approach is non-cooperative because we use a parallel divide and conquer search to gain efficiency in execution. Emphasis on ease of design and complexity reduction have lead us to chose feasible solutions instead of optimal solutions.

The remainder of this paper is organized as follows. Section 2 we discuss previous work about nanorobotics. In section 3 based on these observations, we describe an alternative communication model, the complex environment design, and the behavior of our proposed robots . In section 4 we present our tool description. Section 5 we discuss our simulation results obtained in our simulator *Cellbot-Simulator*. Finally, section 6 concludes the paper.

2 Previous Work

This section, briefly describes previous work about distributed computing applied to nanorobotics. A team of robots can be considered as a kind of mobile distributed system, therefore, coordination of teams of robots faces same distributed system difficulties. However, nanorobotics considers robots built at a molecular-level, where interactions of nanorobots are different in comparison with high level robots.

According to researchers in molecular nanotechnology [3], it would be theoretically possible to build robots from sizes $\approx 2\mu m$ (animal cells $\approx 50\mu m$). The extremely small size of nanorobots, offers a vast number of potential applications interacting in complex environments. So, almost every application of nanorobotics imagined so far, involves large teams of robots rather than a single entity. Communication and navigation problems represent some important chal-

lenges for nanorobotics field.

Adriano Cavalcanti and Robert A. Freitas Jr, present an advanced three-dimensional graphic environment that works as a simulation tool [2].

This research suggests that, emulating methods from social insects with pre-programmed actions, enable agents to achieve specific goals. Cavalcanti shows that, coherent behavior displayed in their simulation, can also be attributed to the common shared goal.

3 The Alternative Communication Model

Creation of a communication model means that we have to study *minimum environmental conditions* under which particular problems are solvable. For our propose, not explicit communication is allowed, this is because our model is based in a biological cell and according to [3] wireless communication might be lethal in cells. Besides, there are electrochemical elements that can be used to establish communication between nanorobots as a kind of *breadcrumbs*. Nevertheless, this practice could leave remainders inside the organism, therefore we propose a not explicit communication model.

The achievement of our robots is to search mobile targets inside a complex environment; to accomplish this, our robots use an alternative communication model. A communication model defines how entities can interact [3]. Defago in [3] defines four communication models:

1. **No communication model.** Mobile hosts do not communicate directly. Instead, they act according to their own perception of the environment.

2. **Cellular network model.** The search area is divided into cells, each of which is managed by a base station. A backbone network interconnects the base station. Two mobile hosts cannot establish a direct connection. They must rely on a base station in order to communicate.

3. **Ad hoc network model.** Mobile hosts use wireless communication, but do not rely on any specific infrastructure (i.e., a network of base stations).

4. **Alternate network model.** In some applications, none of the existing techniques might be acceptable. This consideration is important, because the communication medium cannot always be abstracted out of the network model.

Our communication model combines two of them: *cellular network model* and *no communication model*. Since our search area is divided and the robots do not have communication between them. It is important to remark that our communication model does not have base stations, in contrast with the cellular network model.

3.1 Virtual Complex Environment

We have designed a tool that was used to simulate the behavior of our autonomous multi-robot team and to model the virtual complex environment. We have developed a tool to design of *cellbots* [1] (see table 1). This should be robust enough to operate in the virtual complex environment, which presents a dynamics explained below. The design of nanorobots adopts concepts provided from robotics [1], but keeping in mind kinetics assumptions about nanorobotics, where friction, adhesion, and viscous forces are paramount and gravitational forces are neglected because they can be seen as emergent properties.

Table 1. Representation of our agents according to our computational simulator.

Shapes	Names
	Mobile Obstacles
	Cellbot
	Mobile Targets
	Wall

3.1.1 The importance of noise and disturbances.

Exploration of unaccessible environments (e.g., space, deep sea, cells, human body), represents new challenges for nanorobotics. Also, many classical objections to the feasibility of nanotechnology, such as quantum mechanics, thermal motions and friction, have already been considered and resolved. This motive us to model a chaotic environment, adding noise and disturbances to test our proposed agents (cellbots). We have visualized the chaotic environment, considering a set of cells where each one has open/close gates to access a contiguous cell, as shown in figure 1. In some cases do not exist direct access to a contiguous cell, therefore, the

[1] a cellbot is a simulated agent in our tool.

complex environment presents diverse paths to achieve a cell.

To represent a complex and dynamic environment we have adding a method. This is achieved by adding some disturbances handled by the algorithm 1. This algorithm enables us to model a chaotic environment. This is used as a benchmark to experiment with our autonomous multi-robot team.

Closing gates algorithm opens and closes gates randomly. Experimental results have lead three gates as maximum threshold, because this enables to conserve an acceptable agent behavior, without presents stagnation. Closing gates algorithm is described next:

Algorithm 1: Closing gates algorithm

```
1  begin
2      gate1 ⟵ 0
3      gate2 ⟵ 0
4      gate3 ⟵ 0
5      closing − frequency ⟵ 500
6      if timer / closing-frequency = 1 then
           /* random values to close three
              gates.                          */
7          set gate1 ⟵ random [0..n-gate]
8          set gate2 ⟵ random [0..n-gate]
9          set gate3 ⟵ random [0..n-gate]
           /* close gates.                    */
10         if gate1 = n-gate or gate2 = n-gate or
           gate3 = n-gate then
11             close n-gate
12         else
               /* otherwise we open gates     */
13             open n-gate
           /* start the cycle again           */
14         reset-timer
15 end
```

3.1.2 Division of the Environment (Interaction Zones).

Kube [5] has pointed out that a careful decomposition of the main problem task into subtasks with action based on local sensor-based perception could generate multi-robot coherent behaviors without explicit communication.

First, our environment (search area) has nine cells. We can see it in figure 1. The environment has nine cells and is divided in *interaction zones*. The number of interaction zones depends of the number of robots. Each robot has a designated zone composed by cells, in which the robot can move and search for mobile targets.

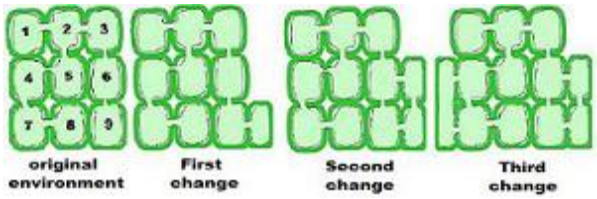

Figure 1. Environments presented along the time.

Being SA the original environment (search area), c a cell, ncb the number of cellbots and z an interaction zone. We have: $SA = \{c_k | k \geq 1 \wedge k \leq 9\}$.

- if ncb = 5, SA = $\{z_1, \ldots, z_5\}$, where $z_1 = \{c_1 \cup c_2\}$, $z_2 = \{c_4 \cup c_5\}$, $z_3 = \{c_7 \cup c_8\}$, $z_4 = \{c_6 \cup c_9\}$, $z_5 = \{c_3\}$.
- if ncb = 6, SA = $\{z_1, \ldots, z_6\}$, where $z_1 = \{c_6 \cup c_9\}$, $z_2 = \{c_7 \cup c_8\}$, $z_3 = \{c_4 \cup c_5\}$, $z_4 = \{c_2\}$, $z_5 = \{c_1\}$, $z_6 = \{c_3\}$.
- if ncb = 7, SA = $\{z_1, \ldots, z_7\}$, where $z_1 = \{c_1\}$, $z_2 = \{c_2\}$, $z_3 = \{c_3\}$, $z_4 = \{c_4\}$, $z_5 = \{c_5\}$, $z_6 = \{c_7 \cup c_8\}$, $z_7 = \{c_6 \cup c_9\}$.
- if ncb = 8, SA = $\{z_1, \ldots, z_8\}$, where $z_1 = \{c_1\}$, $z_2 = \{c_2\}$, $z_3 = \{c_3\}$, $z_4 = \{c_4\}$, $z_5 = \{c_5\}$, $z_6 = \{c_7\}$, $z_7 = \{c_8\}$, $z_8 = \{c_6 \cup c_9\}$.
- if ncb = 9, SA = $\{z_1, \ldots, z_9\}$, where $z_1 = \{c_1\}$, $z_2 = \{c_2\}$, $z_3 = \{c_3\}$, $z_4 = \{c_4\}$, $z_5 = \{c_5\}$, $z_6 = \{c_6\}$, $z_7 = \{c_7\}$, $z_8 = \{c_8\}$, $z_9 = \{c_9\}$.

It is important to say, that our environment has three "environmental changes" (cell deformation) as we can see in figure 1. In each change is generated one extra cell. These cells are examined by the robot or robots that are in a contiguous cell.

3.1.3 Behavior of Cellbots.

Our cellbots are designed as reactive agents with two specific behaviors. First, they use a simple evasive behavior to avoid obstacles within complex environment. Second, they use a collector forage behavior to identify mobile targets and collect them. We must note that nanorobots differ from robots by means of an important feature, self-replicate [2]. We have not include their kind of self-replicating behavior.

Our cellbots have sensors to report collisions with *walls*, we mean by *walls* as borders in our environment.

[2] to make a *physical* copy of themselves

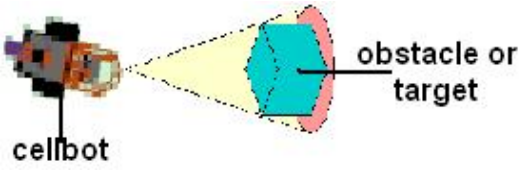

Figure 2. A cellbot using the in-cone function

Figure 3. Cellbot-simulator

Also, they have sensors for identifying when an encountered object is an obstacle to be avoided or a target to be caught. Finally, if that is the case, they have a predefined knowledge about "zone boundaries". All tasks mentioned above are done in a concurrent way.

The predefined actions that we implement in our agents are defined next:

- **Evasive behavior.** When cellbots find a wall, another cellbot or a mobile obstacle, they backward and turn a random value in a range between 0..360° to the left. But if they find zone boundaries, they just turn in the same range to the left.
- **Identify the behavior of mobile targets.** In order to search the mobile targets, the cellbots look for them by using an "in-cone" function (see figure 2) that is presented in equation 1, with a perception of $\angle 120°$. Then, when they locate a mobile target, they fix a trajectory toward it and they go directly for it. The exploration of cellbots presents an acceleration in speed when they perceive a mobile target in order to catch it, the mobility of targets can cause a failed try for cellbots; on the other hand, cellbots keep trying until they catch it.

$$V = \frac{\Pi \cdot r^2 \cdot h}{3} \quad (1)$$

where V is the volume of perception, r is the radius and h is the height.

3.1.4 Mobile Obstacles and Mobile Targets.

The behavior of mobile obstacles avoids walls and reports collisions with another objects (cellbots, mobile targets and other obstacles), we have three kinds of obstacles distinguished by their speed.

Mobile targets inherit same behaviors from mobile obstacles but they could be caught by cellbots and are created spontaneously along simulation (these behaviors are concurrent). Distribution of mobile obstacles and mobile targets is according to a position matrix, that we setup when simulation initiates. This position matrix has random coordinates within the environment, these coordinates are the angles of each cell, except the fifth cell (see figure 1). This is because in experimental results, all the cellbots can initiate the search in that cell. So, it does not make sense to explore the fifth cell.

Being α an angle of a cell, c a cell, C the set of cells, A the set of angles and PM the position matrix. We have:

$A = \{\alpha_1, \ldots, \alpha_n\}$, $C = \{c_1, \ldots, c_9\}$, $PM = \{\{C\} - \{c_5\}\}$.

$\forall c_i \in PM \exists AC$, where AC is a cuadruplet of angles $a_1, a_2, a_3, a_4 \in A$.

4 Tool Description

Cellbot-Simulator (see figure 3) is an interactive computational tool for simulating the navigation of cellbots to seek targets, using an alternative communication model that consists in the use of the cellular network model and no communication model. The tool is written in NetLogo 3.1.3. [9], that is an agent-based parallel modeling and simulation environment produced by the Center for Connected Learning and Computer-Based Modeling at Northwestern University.

The user can select the number of *cellbots* using the slider of num-cellbots and if he wants to see the identifier of every cellbot he has to set the "show-who? switch" to on.

Our tool offers the possibility of dividing the environment into zones when the "by-zones? switch" is on or handling the environment without divisions when the "by-zones? switch" is off. For the other hand, if the "place-bots? switch" is on, the cellbots are placed in their corresponding zones according to our distribution function based on experimental results; but if the "place-bots? switch" is off, all the cellbots are located in the center (that is because cellbots are placed by default in the fifth cell).

The user can observe the number of current targets, collisions, the target log and the number of recollected

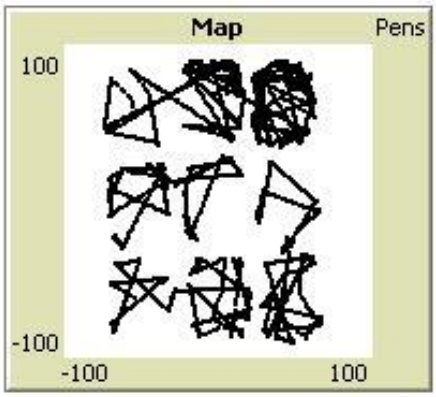

Figure 4. Map of the terrain coverage by the cellbots in our tool

targets in their respective monitors. Also, he can observe terrain coverage by the cellbots in the "map" (see figure 4) and at what time and which cellbot found a target in the "time of recollected targets" output.

5 Results

In this section, we describe how we did the tests on our tool and we present our results.

We used as maximum nine cellbots and as minimum five cellbots, having five tests with the switches "by-zones? " and "place-bots? " in on and five tests with the switches "by-zones?" = off and "place-bots?" = on, resulting ten tests (see table 2). The cases with "by-zones? switch" = on are to establish the zone boundaries and the cases with "by-zones? switch" = off are to allow the cellbots to move freely in the whole search area. Each test was running around 1000 time steps (time unit in NetLogo), following the guides explained below. Then we did the ten tests eight times (in total 80,000 time steps), we got the number of recollected targets from each test. Finally, we obtained the averages of the recollected targets of the eight groups of the ten tests and we plotted them, we can see the ten tests and their averages of the recollected targets in the table 2.

We did each test, following the next guides:

- We set the number of cellbots.
- For the first five tests, we set the "by-zones? switch" = on and for the last five tests, we set the "by-zones? switch" = off.
- We set the "place-bots? switch" = on.

Table 2. Tests done

#Test	*#cellbots*	*By− zones?*	*Place− bots?*	*Avg_of recollected targets*
1	5	on	on	219.38
2	6	on	on	246.38
3	7	on	on	282.38
4	8	on	on	304.50
5	9	on	on	468.50
6	5	off	on	157.50
7	6	off	on	169.13
8	7	off	on	238.63
9	8	off	on	298.63
10	9	off	on	334.13

- We clicked the set-up button.
- We clicked the go button.
- We stopped the go button when the number of time steps was 1000.
- We exported from the output the number of recollected targets.

In the figure 5, we can see the averages of the 10 tests, that we did eight times each one. The line with squares represents the averages of the recollected targets of the five tests with the switches "by-zone? " and "place-bots? " in on (Tests 1, 2, 3, 4 and 5 of the table 2); while the line with triangles is the averages of the recollected targets of the five tests with the switches "by-zone?" = off and "place-bots?" = on (Tests 6, 7, 8, 9 and 10 of the table 2). As we can see, the averages of the tests 1, 2, 3, 4 and 5 are greater than the averages of the tests 6, 7, 8, 9 and 10.

Nevertheless, the differences between them, are not quite similar. When the number of cellbots is seven, the average of recollected targets of the test 3 has a difference of 43.75 targets with respect to the average of recollected targets of the test 8; but if the number of cellbots is five (Tests 1 and 6), the difference is of 61.88 targets and if the number of cellbots is six (Tests 2 and 7), the difference is of 77.25 targets. However, when the number of cellbots is eight (Tests 4 and 9), the difference is minimum, it is just of 5.87 targets. For this reason, we need to do more tests with this number of cellbots, in order to find problems in the distribution function that we are proposing in our model. It is important to remark that our distribution is feasible,

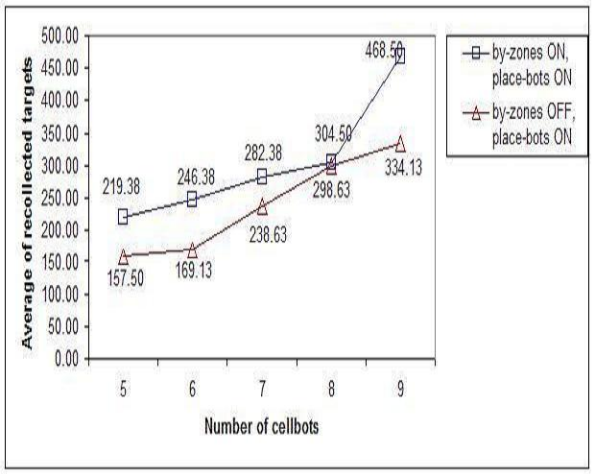

Figure 5. Average of the recollected targets from the tests

but not optimal. For the other hand, when the number of cellbots is nine (Tests 5 and 10), the difference is 134.37, a big difference with respect of the others, almost the double of the difference with the tests done with six cellbots (Tests 2 and 7).

6 Conclusions

In spite of some recent work in a similar direction [2, 3], the field is still largely unexplored and promises for many interesting challenges, especially in modeling of complex environments. Many of the most difficult problems are modeling environments where do not exist explicit communication.

The present work considers the importance of communication models to face new challenges in nanorobotics. We have presented a graphic environment that models nanorobotics motion and physically based simulation applied to seeking tasks. We have introduced the notion of combining models, that allows agents to reach their tasks, taking into account the complex environment influence.

The experiments to improve the distribution function are in progress. Future work will focus to develop a self-replicating behavior and will consider the implementation of reinforcement mechanism (learning) in order to enable the nanorobots to identify priority of tasks.

Acknowledgements.

The authors thank the reviewers of this paper for their useful comments. Mr. Piña-García and Ms. Rechy-Ramírez have been partially supported by the Mexican National Council of Science and Technology (CONACYT), through the program "Becas para Estudios de Posgrado" (no. 197883) and (no. 183592), respectively.

References

[1] Arkin, R. C.: Cooperation without communication: Multiagent schemes-based robot navigation. Journal of Robotic Systems, 9(3): 351-364.

[2] Cavalcanti A. and Robert A. Freitas Jr.: Nanosystem Design with Dynamic Collision Detection for Autonomous Nanorobot Motion Control using Neural Networks. Computer Graphics and Geometry Journal, Vol. 5, no. 1, pp. 50-74, May 2003.

[3] Defago X.: Distributed computing on the move: From mobile computing to cooperative robotics and nanorobotics. In Proc. 1st ACM Int'l Workshop on Principles of Mobile Computing (POMC'01), pages 49–55, Newport, RI, USA, Aug. 2001.

[4] Gilbert, G. N., Troitzsch, K. G.: Simulation for the Social Scientist. J. Artificial Societies and Social Simulation 3(3): (2000).

[5] Kube, C. R., Zhang, H.: Collective robotics: from social insects to robots. Adaptive Behavior 2, 189218 (1994).

[6] Muñoz, A. and Drogoul, A.: What Kind of Cooperation is Required by Situated Agents?. The Principle of Situated Cooperation. Active Media Technology, LNCS 2252, pp. 165-170. Liu J. et al. (eds). Springer-Verlag, Berlin.

[7] Piña-García, Carlos Adolfo and Garcia-Vega, V. Angelica: A Hybrid Methodology for Robotic Architectures with a Cellular Approach. E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on (ICELIE), 156-160, Dec. 2006.

[8] Reich, J. and Sklar, E.: Robot-Sensor Networks for Search and Rescue. IEEE International Workshop on Safety, Security and Rescue Robotics, Gaithersburg, MD., August 2006.

[9] U. Wilensky.: Netlogo, center for connected learning and computer-based modeling, 1999. http://ccl.northwestern.edu/netlogo.