A Framework for Reconfigurable Robot Swarm Simulation

Lluís Ribas-Xirgo, Daniel Alcantarilla, Ángel Diéguez, Pere Ll. Miribel-Català,
A. Josep Velasco-González and Manel Puig-Vidal

***Abstract*—Envisaged applications of mobile robots and microrobots are very promising but the development of these devices depends on a bunch of disciplines and is highly complex. In this paper we present a simulator of a swarm of robots with limited capabilities so that they require self-reconfiguration to adapt to their environment. A simple toy-problem is analyzed by using the simulator of such a type of robot swarms. The data is taken from real microrobots so the simulation outcome helps to foresee efficient robot behaviors in realistic scenarios.**

***Index Terms*— Cooperating robots, micro and nano robots, reconfigurable robots, robot swarms, simulation.**

I. Introduction

One of the main objectives in the field of Microrobotics is to develop microrobots with small dimensions, high power autonomy and accurate enough to face the stringent requirements of their possible applications, particularly those of biomedical applications. This objective is extremely ambitious and different research groups worldwide are working in this direction. Important technological challenges have to be solved in terms of miniaturisation and power autonomy.

The cooperation among microrobots to efficiently implement the proposed experiments within biomedical applications is an important challenge [1]. The degree of success depends on the accuracy of the autonomous and mobile robots, such as the I-SWARM robots [2], and on the biomedical application definition where the flexibility of these robots is needed.

In this work, we have focused on the problem of robot cooperation in a context in which microrobots have computational power limitations, basically due to small memory size and low power consumption requirements. The approach taken to study efficient cooperation strategies under the former conditions is by creating a sort of a "design framework" based on a customizable swarm robot simulator.

Given that the novelty of the field of biomedical applications of microrobot swarms and the incipient microrobot development technology do not enable to carry on experiments with actual robots, we propose to set up the framework by using a toy problem which is realistic enough to be representative for many such applications while preserving enough simplicity to be tackled by a simulator.

This simplicity stems from the fact that the behavior of the microrobots is highly abstracted both for their functionality and their non-functional characteristics. However, the computational limitations are still taken into account in the simulation model. Particularly, the microrobots follow a simple behavior and may require to be dynamically configured when faced with a context in which part of their current functionality becomes inoperative. The simulator enables the system designer to explore the search space formed by different behavior choices so to find an efficient solution in terms of time and energy consumption.

The rest of the paper is organized as follows. Section II is devoted to describe the toy problem that is going to be simulated and the abstract state machine (ASM) that is used to model the robots' behavior. Each ASM state is associated with non-functional parameters as well as with specific functionality. Particularly, one of the states is to account for the different configurations (program and sensor and actuator management) in which robots can operate. The next section describes the simulator and its use through an example that can be used in education [3]. In this case, Parallax Boebot robots, which have both computational and power limitations, are used. Afterwards, a futuristic scenario with realistic I-SWARM robots is set up and result simulations are presented. The last section concludes the paper by highlighting the main contributions of this paper and presenting the work to be done in the future.

II. The Red & Blue Dot Game

This "game" is about transforming red and blue dots of a given surface into black ones. The game is played by an autonomous robot swarm that is composed of two kinds of robots. The red ones are able to transform red dots into black ones and the blue ones do the same with blue dots.

Although all robots are homogeneous and have similar capabilities, the dot transformations imply different procedures that cannot be stored together in the program memory. Thus, red and blue robots can only work with dots of the same color unless they reconfigure themselves by reprogramming their memories with the other code, and discarding the previous one.

The outcome of this game depends on many factors as the number and distribution of dots and robots of each color, the reconfiguration policies, the time allowed, or the bounds on

Lluís Ribas-Xirgo is with Universitat Autònoma de Barcelona.
E-mail: Lluis.Ribas@uab.cat

Daniel Alcantarilla and A. Josep Velasco-González are with Universitat Autònoma de Barcelona.

Ángel Diéguez, Pere Ll. Miribel-Català and Manel Puig-Vidal are with Universitat de Barcelona.

energy consumption.

A. About the Players

The players of this toy problem are robots that share a common constraint: that of not being able to have enough memory to contain the code for working with red and blue dots. From this simple constraint, which might be applicable to microrobots, when they are widely available, a number of interesting problems can be observed and, obviously, the corresponding approaches developed.

For the sake of simplicity, we assume that robots are homogeneous both in hardware (sensors, actuators, motors, batteries or energy harvesting systems, *et cetera*) and rest of the controlling software. Therefore, they all have the same characteristics in terms of movement and energy consumption.

Particularly, we consider that robots share a basic behavior that is shown in Fig. 1. They start at searching colored dots in their sensor range and, when one is found, they go to it and look for their color. If it is the same, then the work of transforming it into a black dot begins. On the contrary, reconfiguration is considered so to either work with the found dot or disregard it and search for another one.

Search methods may range from a simple randomly selected series of goals (*i.e.* end positions to activate sensors to detect colored dots) to more elaborated strategies. Similarly, the reconfiguration policy can be as simple as to do the reconfiguration whenever is possible or to decide it in terms of the history of the robot.

In the *search* and *go&look* states, robots use a 360º sensor with a given radius range within they can detect colored dots and other obstacles. This sensor does not provide data about color, only about the presence of objects.

In the *go&look* state a contact sensor to detect the color is used. Depending on the result (the object can be a colored dot or any other thing) and on the reconfiguration policy, from the *go&look* state, it can start working, try to reconfigure itself, or go back into searching mode again.

During *reconfiguration* state, if the robot decides to do so, it is required that a robot of the opposite color is within reach of the data input terminal, which is also omni-directional and whose range can also be set. Only robots that

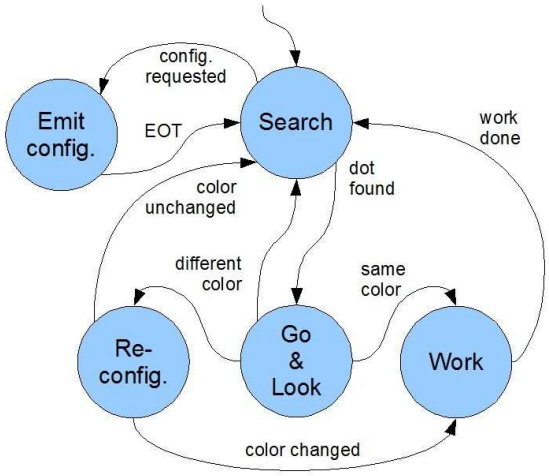

Fig. 1. High-level graph of a robot's behavior. Red and blue robots are only different at the *work* state.

TABLE I
NON-FUNCTIONAL PARAMETERS OF THE ROBOTS

Symbol	Quantity	Units
r_S	object detection sensor range (radius)	cm
r_R	communications' range (radius)	cm
v_S, v_G	average speed, usually $v_S = v_G = v$	$cm \cdot s^{-1}$
t_E, t_R, t_{WR}, t_{WB}	average time spent in a given state	s
P_S, P_G	power consumption	mW
E_E, E_R	energy consumption in reconfiguration	mWh
E_{WR}, E_{WB}	energy consumption when working with red and blue dots	mWh
E_B	total energy in battery, which can be ∞ (unlimited) in case of dynamic recharging	mWh

Subscripts S, G, WR, WB, E, and R are used for parameters that are related to states *search*, *go&look*, *work*, with red or blue dots, *emit reconfiguration*, and *reconfigure*, respectively. (Units are shown with default values that can be adapted to a given problem.)

are in the *search* mode may change it to a state in which emit their configuration to the requesting robot, *i.e.* this *emit configuration* state is only reachable from the *search* one. Robots enter this state when requested to do so and leave it on end-of-transmission (EOT).

As for the non-functional behavioral aspects that affect time, space and energy, we assume that a robot profile is obtained beforehand so the time and energy consumption to complete a color transformation work or to emit or make a reconfiguration is known. *Search* and *go&look* time and energy consumption are not constant because they depend on the environment. Thus, they are computed from average speed and maximum power consumption of the robots.

More accurate profiles can be built by thoroughly testing the robot under different cases. In this case, though, it is a formal model with no actual counterpart and does not require (nor has it) to be very detailed to prove the validity of the approach.

B. About the Scenario

The environment in which the robots play the red & blue dot game is a rectangular surface with an initial setup consisting on a number of red and blue dots as well as some obstacles, including the boundaries of the working area. The location of dots of every color can be defined or randomly set. The number of red and blue robots is also user decidable as well as their initial locations.

Combining different scenarios with different searching methods and reconfiguration policies it is possible to analyze the best setups for a robot swarm to finish the game in an efficient form or, at least, optimally with respect to some parameter.

Robot characteristics greatly influence the former decisions, as the time and energy spent to complete a given job depends on them.

Obviously, the total energy consumption cannot be greater than the total available energy or, more exactly, this restriction applies to every robot individually, *i.e.*:

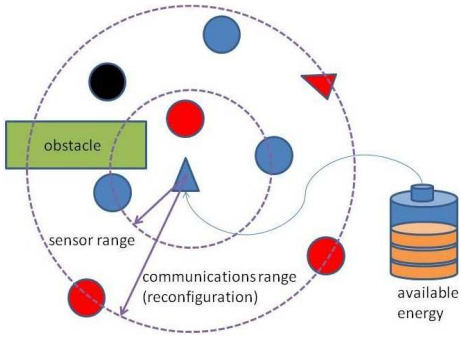

Fig. 2. Reconfiguration and object detection ranges, as well as the available energy affect the functional behavior of a player in the red & blue dot game.

$$E_{\mathrm{B}i} \geq \begin{array}{c} q_{\mathrm{S}i}t_{\mathrm{S}i}P_{\mathrm{S}i} + q_{\mathrm{G}i}t_{\mathrm{G}i}P_{\mathrm{G}i} + q_{\mathrm{WR}i}E_{\mathrm{WR}i} + \\ + q_{\mathrm{WB}i}E_{\mathrm{WB}i} + q_{\mathrm{R}i}E_{\mathrm{R}i} + q_{\mathrm{E}i}E_{\mathrm{E}i} \end{array}, \quad (1)$$

where $q_{\mathrm{X}i}$ stands for the times robot i has entered state X, for X = (S, G, WR, WB, R, E), and $t_{\mathrm{S}i}$ and $t_{\mathrm{G}i}$ can be defined as follows.

$$\begin{array}{c} t_{\mathrm{G}i} = f\left(r_{\mathrm{S}}/v_{\mathrm{G}},\ldots\right) \\ t_{\mathrm{S}i} = f\left(r_{\mathrm{S}}/v_{\mathrm{S}}, \left\lceil\left((A_N/A)\Pi r_{\mathrm{S}}^2\right)^{-1}\right\rceil - 1,\ldots\right) \end{array}, \quad (2)$$

where the second argument of the function for $t_{\mathrm{S}i}$ stands for the number of iterations over the *search* state before detecting an object. In this expression, term A_N/A refers to the object density in the working field, i.e. A_N is the total area taken by the N objects (robots, dots and obstacles) in the field, and A the area of the latter. Both functions can easily be approximated with linear expressions in terms of empirically obtained constants and given arguments.

Parameters $q_{\mathrm{X}i}$ depend on elapsed time, searching technique and reconfiguration policy, among others, and have interdependencies. For example, the number of remaining red and blue dots affects the quantity of possible reconfigurations, which depends on the ratio of red and blue robots, which, in its turn, varies with every successful reconfiguration. Following this very example, it is obvious that to compute $q_{\mathrm{X}i}(t)$ it is required to integrate the corresponding differential equations and, furthermore, that the solutions strongly depend on the initial conditions of the system.

Another approach to compute energy and time needs for a given scenario setup is to simulate the whole. By doing so, it is possible to take all the above-described factors into account and have reasonable estimates of the system performance, including individual data.

III. Simulator of the Red & Blue Dot Game

In order to obtain data on system performance by simulation, it is necessary to use a simulator that monitors not only time but other parameters, especially, the energy consumption, and state-related accounting.

Instead of developing a whole simulator for the red & blue dot game, we have adapted an existing one by modifying the robot behavior and by including performance data monitoring. The rest of this section is devoted to describe the original simulator and the adaptation, and to give a working example.

A. The MuRoS Simulator

MuRoS [4] is an object-oriented application that has been developed in Visual C++. It is based on running two main threads: one to solve the dynamic equations of the system and the other for the real-time screen display of simulation.

The hierarchy of classes stems from base class `CRobot`, which contains basic methods for robots that include those for managing sensors and actuators.

The basic robot is equipped with a sensor that detects the presence of obstacles and other elements. This kind of sensor returns a list of all objects in the sensor range with no shadows, *i.e.* closer objects do not hide other objects at their back along sensor view line.

Also it has an actuator consisting of a method to move it to an absolute destination point in the arena.

At the lowest level of the class hierarchy there are two different subclasses, `CRobotHolonomic` and `CRobotNonHolonomic`, each with its own characteristics. So subclasses can be derived robot-type holonomic or not holonomic.

This simulator does not take into account odometry errors: it is assumed that each robot knows exactly its position in the environment.

B. Adapting MuRoS to Red&Blue Dot Game Robots

To simulate robot swarms that are targeted to solve problems similar to the proposed toy problem it is necessary to organize the control program of each robot so its architecture reflects the abstract view of the high-level FSM presented in the previous section. On the other side, the non-functional parameters related to time and energy consumption must be taken into account.

In the following, the corresponding adaptation of MuRoS to these needs is discussed.

A derived class of `CRobotHolonomic` has been created so to easily simulate Brownian movement of cells and, especially, of random movement holonomic robots like the ones of the red & blue dot game. (Random movement can be changed to movement on local context.)

Robots of this new class have a behavior as described in the previous section, with reconfiguration and communication capabilities. Methods run at each state might be changed if required.

The simulator takes into account not only the time but also the energy spent of every robot in the simulated system. This enables to compare the functionality of the system with the expected one and also to know the cost in time and energy of a given experiment. Consequently, different functionally equivalent solutions can be simulated in order to choose the one with the less cost in terms of the target application and its constraints.

The time step of the simulator must be adjusted to the units of time used in the robot specification, i.e. the unit of the time step must be the maximum precision in which time-related data of robots is given. The amount of time that a robot spends in a given state depends on the parameters set by the user in accordance with the robot characteristics and control program. Particularly, the *work*, *reconfiguration* and *emission* states are defined as tasks that take a constant amount of time to complete, while *search* and *go&look*

depend on the average movement speed, which has to be set taken into account not only the movement but all the other things a robot must do to proceed with the action.

To take into account energy consumption and accounting data about the states, new variables have been added to the original simulator. Energy consumption depends on time, so its calculation is done similarly.

The working arena can also be configured, being rectangular by default. Obstacles can be added to suit any possible scenario, including that of arenas with other shapes.

The resulting simulator allows you to choose the number of robots and the number of points of each color, in addition to the distribution of robots and dots on the stage, which can be random or grouped. Robots on stage may have limited or unlimited battery, which drastically affects its behavior. Furthermore, robots can be reconfigurable or not, so easy comparison between reconfigurable or fixed robots can be done through simulation.

To show how the simulator works, a simple case of a small swarm of robots is presented below.

C. *Sample Case*

The red&blue dot game is played by a team of 10 robots that have to transform 20 coloured dots, 10 in red and 10 in blue, into black ones in a room of 5mx5m.

Initially, there are 5 red robots and 5 blue robots, whose non-functional parameters are set in accordance with simplified estimates for Parallax Boebots.

We assume that they work with batteries that deliver a total energy of $E_B = 6000$mWh, which is a very pessimistic estimate to account for the fact that Boebot's works efficiently with four 2000mAh AA batteries for about half an hour, even though they still have enough energy to move the servos and feed the rest of the robot's circuitry.

These robots have an average linear speed of 20cm/s. However, taking into account the twists and other movements are not made to the average speed linear, and with the idea of making the final figure more realistic, the rate is reduced to 40%, i.e. the average velocity is considered to approximately be v=12 cm·s^{-1}.

The reconfiguration and emission times are set to 22s, *i.e.* $t_E = t_R = 22$s. (It is the approximate time to send a 1Kb message by the infrared remote control communication protocol RC5.) And the corresponding energy consumptions are, approximately, E_E = 18,3mWh and E_R = 23,8mWh.

The power consumption at states *search* and *go&look* is about the same power that batteries can deliver, *i.e.* $P_S = P_G$ = 12000mW.

Time and energy consumptions in the working states depend on the task to be accomplished. We assume arbitrary values for t_{WR} = 30s and t_{WB} = 45s.

The energy consumption is also arbitrarily set, but with the difference that the work with red dots does not imply much movement while the blue dots require a constant movement of the robot over the dot. With this assumptions, E_{WR}= 65mWh and E_{WB}= 150mWh.

In practice, the effective communication range of the IR LED/sensors is of about 40cm, while when using them to detect obstacles, the range arrives perfectly to 1 meter.

In Fig. 3 there is a summary of all these data from the simulator program, and in Fig. 4 there is a screenshot of it when running a simulation, where robots are painted with a color that corresponds to their states at every moment.

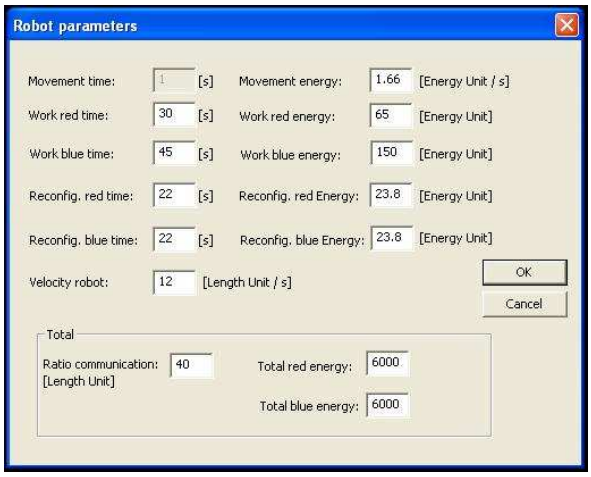

Fig. 3. Snapshot the parameters of the Boebot robot in the simulator.

A number of simulations have been done with this setup and randomly scattered dots and robots. While the number of working states reconfigurations does not significantly change among simulations, energy consumption and time to complete the job show big differences. Basically, because the most important part of energy and time consumption of Boebots is devoted to the movement, which is random in the search state.

Scenarios are completed between 300 and 600s 70% of times, 20% less than 350s, and 10% over 600, approximately. Average energy consumption per robot is shown in Fig. 5. Notice that they are similar though $E_{WR} < E_{WB}$ because most of the energy consumed by robots is on movement and reconfiguration.

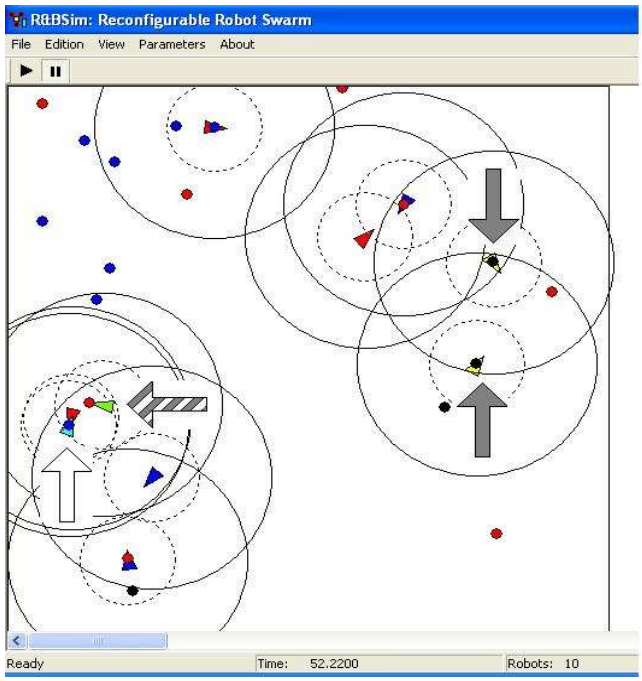

Fig. 4. Screenshot of simulator: Solid arrows signal robots at work, stripped arrow and white arrow are used to point to configuration emitting and reconfiguring robots, respectively.

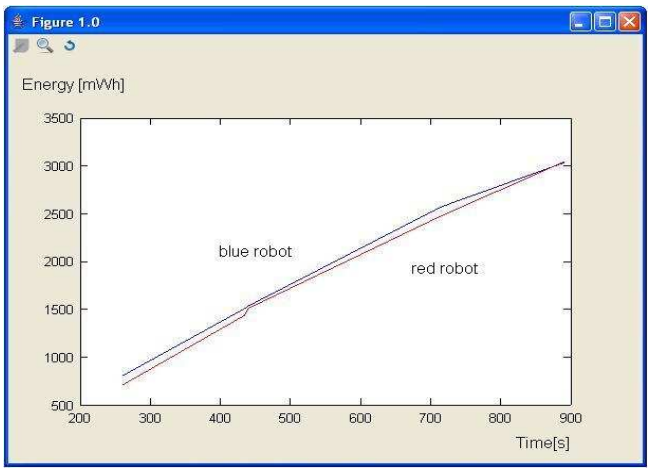

Fig. 5. Energy consumption along time.

IV. I-SWARM MICROROBOT

In classical microrobotics, highly integrated and specialized robots have been developed, which are able to perform micro-manipulations controlled by a central high-level control system. Technology is still far away from the first "artificial ant" which would integrate all capabilities of these simple, yet highly efficient swarm building insects.

These microrobotic insects are the main motivation focus on studying such swarm behavior and trying to transfer it to simulations and, further, to real physical robot agents.

Fortunately, even small robot groups of 10 to 20 robots are capable to mimic some aspects of such social insects. However, the used robots are usually huge compared to their natural counterparts, and very limited in terms of perception, manipulation and co-operation capabilities.

The I-SWARM project combines all: microrobotics, distributed and adaptive systems as well as self-organizing biological swarm systems. It is targeted to foster a technological advance to facilitate the mass-production of microrobots, which can then be employed as constituents of "real" swarms consisting of more than 100 microrobot clients. These clients will all be equipped with limited, so-called pre-rational on-board intelligence.

Devised swarms will consist of a huge number of heterogeneous robots, differing in the type of sensors, manipulators and computational power. Such robot swarms are expected to accomplish a variety of applications and to follow different strategies.

With the realization of different bio-inspired basic scenarios, swarms will be able to perform dispersion, aggregation and collective perception, for instance, as basic swarm operations to attain the goals of a given application.

In the following, the I-SWARM microrobot is described and further modeled with the required simulation parameters so to be used in a small swarm in a realistic application in the biomedical field.

A. *Robot Structure and Main Characteristics*

The I-SWARM microrobot is a high integration technological development. It is based on four modules: power, communication, electronics, and locomotion, assembled by means of a flexible membrane (FPC). The most critical module is the power autonomy.

The power autonomy is based on a full custom designed solar cell able to deliver a maximum power of 1.5mW continuously. Solar cells have been connected so to provide enough energy to the system trough two power supplies: an un-regulated 3.6 V to the I/Os and 1.6V to a regulated 1.2V to the core [5]. The robot is programmed by light pulses and is able to know his position in the arena by egopositioning. A storage capacitance is integrated in the robot to receive electrical energy from the solar cell and deliver the short high power pulses during the communication process.

The communication between robots is ensured by IR transceiver based on a LED and a photodiode integrated in each side of the robot. A specifically designed integrated micromirror ensures the horizontal communication. The maximum communication range is limited to 1cm to reduce the needed power for LED emission.

A full custom designed IC ensures the proper interface with all the robot modules, the control by means of an integrated microprocessor (8051 microcontroller) and the power optimization by means of a designed "power management" module.

The microrobot locomotion is ensured by a piezoelectric polymer (PVDF) material based structure on three legs with a bending angle around 45º. The integrated tool is indeed a fourth leg in horizontal position with specific geometry and different resonance frequency sensing the environment by contact.

The autonomous I-SWARM microrobot is based on a flexible membrane offering flexibility, electrical connections and representing the backbone of the robot.

B. *Robot Parameters for the Red & Blue Dot Game*

It is important to bear in mind that actual data is not known because at the time of writing this paper the first units of the I-SWARM robots are being delivered to the project participants. Therefore, data given in this section is based on estimates and tests of independently developed modules. Consequently, the simulations done with model parameters based on these data will give only a rough approximation of what may happen in reality, in the best of cases.

Sensor range for detecting objects is 7mm, as it is the

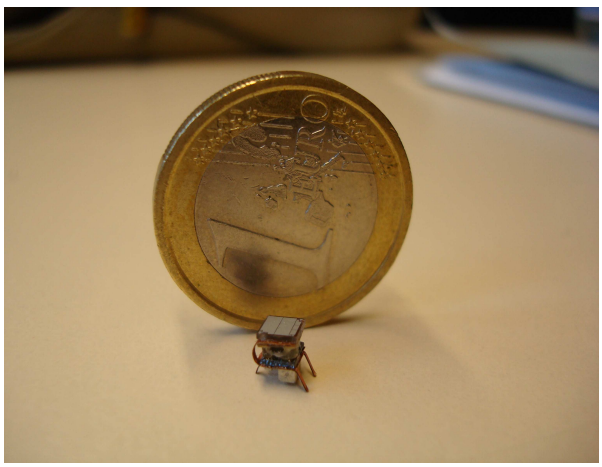

Fig. 6. Photograph of a I-SWARM robot dummy. (Courtesy of I-Swarm project.)

range of the IR sensors. The range for reprogramming can be considered infinity as the robots will not use inter-robot communication to get the data but the incoming data from solar-cells, which in their turn take them from pulsed light over the working field. Besides, this fact implies that these robots do never go to the configuration code emission state, which simplifies their functional behavior.

For the sake of simplicity, we assume that data fed to the robots through pulsed light (Fig. 7) contains is a continuous stream of red and blue codes interleaved with data for ego-positioning, *i.e.* robot identification and (*x*, *y*) position from a zenithal image of the working area.

During the *search* and *go&look* states, robot combines object detection stages with movement ones. We assume that the average speed is about half of the before given 0.4mm/s so to account for the time to detect objects and the variability in the speed as approaching to the goal. Power consumption is set up to the maximum 1500µW. Functional differences between the two states are neglected here. Notice that, in the *go&look* state, there is also a time to discover whether the dot is red, blue or black.

To estimate the reconfiguration time and energy a change of ¼ of the program memory, *i.e.* 2Kb, is considered. As reprogramming is limited by the incoming data rate, which is of about 1Kbit/s, it will take about 16 seconds. This would be in case the incoming data stream from pulsed light starts at the beginning of the right code at the very moment the robot is ready to get the code. Therefore, this waiting-time overhead must be taken into account in the time of reconfiguration.

With these data and a (small) swarm of 128 robots, the cycle of the data stream is of 38s, which gives an average of a 19s wait that should be added to the 16s to do the reprogramming, thus giving an average of 35s to reconfigure a robot. As for the energy consumption, the average is 1mW·35s = 0.00972mWh. It is quite a pessimistic assumption though it does not include the locomotion, nor does the IR I/O consumption.

The time and energy consumption during the working state cannot be estimated from actual data. We assume that the work consists of either touching a cell (red dot) with a sharp blow, *i.e.* applying a single-pulse force, or to measure the mechanically transmitted waveform from another point (blue dot) to look for possible changes on the elasticity of cytoplasmatic membrane. As a consequence, the work on a red dot involves waiting for another robot until given deadline occurs. If another robot comes to the same dot, then, it will go around the cell wall and measuring effects of force pulses in the membrane.

With this behavior, red and blue dots are distinguished by communication between the dot-contacting robots: no answer implies that it is a red dot, single answer that it is a blue one, and multiple answers that it is a black one. (This type of communication is done in the last stage of the procedure for completing the *go&look* state.)

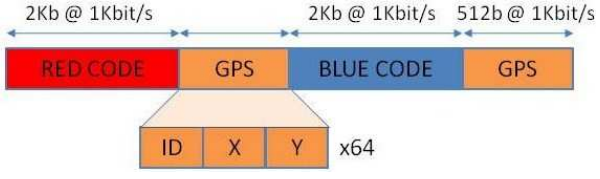

Fig. 7. Data frame loop from system supervisor to robot swarm.

Note that black dots are, in fact, non-testable cells because they have more than two contacting robots, and that a cell might be analyzed more than once. (There is no mechanism to ensure that each cell is only tested once, but it could be easily implemented by taking profit of the egopositioning data of the system that would mark some workplace locations as to contain black dots.)

Therefore, we assume that the time and energy spent in the *work* state depends on the type of dot. For a red one it consists of marking the dot in blue while waiting and, afterwards, of keeping applying force pulses into the cell under examination until a stop message is received. (Marking a dot in blue might consist of tapping the cell wall so any other robot contacting it can detect it. Similarly, the blue robot might tap the cell to send the red one a stop message.) For a blue dot, the work is to go around the cell detecting and measuring the mechanically transmitted waveform from another point on its wall, where the red robot is located.

The time spent for blue dots is basically the time spent in going round a cell and sensing transmitted forces in the membrane through the AFM tip. A rough estimate for going round a cell of a 30-40µm radius cell while doing alternative approaching-receding movements at a low speed to obtain maximum precision in point location, which is considered to be of about 5µm, gives an rounded value for t_{WB} of 4 seconds.

The average time that a red robot spends working with a red dot includes t_{WB} and the waiting time, which depends on the blue robot density and distribution. For a 200x200mm workspace with 128 robots, a robot may come across another after running about 10mm, which takes it 2.5s at 0.4mm/s. Therefore, a reasonable figure for t_{WR} is 4.5 seconds.

In Table II there is a summary of the discussion for the parameters of the I-SWARM robots to simulate a swarm that captures the cytoplasmatic membrane elasticities detected in a group of cells on a workspace of 200x200mm.

Note: The captured data is stored in the robots until

TABLE II
NON-FUNCTIONAL PARAMETERS OF I-SWARM ROBOTS

Symbol	Value	Units
r_S	7.00	mm
r_R	∞ (all the workspace)	mm
$v_S = v_G = v$	0.40	mm·s^{-1}
t_E	(not used) 0.00	s
t_R	35.00	s
t_{WR}	4.50	s
t_{WB}	4.00	s
$P_S = P_G$	1500.00	µW
E_E	(not used) 0.00	µWh
E_R	9.72	µWh
E_{WR}	1.88	µWh
E_{WB}	0.85	µWh
E_B	∞ (unlimited)	µWh

Parameters referred to the state of configuration emission are set to 0 because these robots do not go through it.

running out of data memory. At that moment, they must go into specific dots to download it. This is not considered in the presented scenario.

V. Results with a Realistic Scenario

In the previous section a prototype of a microrobot within the project I-SWARM is described and further modeled for the simulator in accordance with an application in which a colony of such robots try and find membrane elasticity of a group of cells.

Using the before-described simple model of the I-SWARM robot for the depicted work case the following scenarios have been simulated. The first one is to decide whether the robots should be reconfigurable or not, and the second one is about computing the best ratio of red and blue robots.

A. Reconfigurable vs. Non-reconfigurable Robots

Several simulations are done to see the advantages and drawbacks of the reconfiguration in terms of dot and robot density in the working area. In all cases, the initial ratio between red and blue robots is ½, and the number of robots is 64, 128, and 256, all working in a workspace in which there are 16, 64, 256, and 1024 cells. Robots and cells are randomly placed in the arena.

The search is done by random movement and, in case robots are reconfigurable, they do so if at the end of the *go&look* state they found the dot of a different color.

The results of these simulations are summarized in Table III. Data represent the average time and energy for every case, with a variable number of runs from a minimum of 5 so these figures are representative enough. Standard deviation of all given data is less than 15%.

Results show that the amount of time and energy to finish a job depends on the number of cells to be analyzed and also on the number of robots. However, notice that a higher robot density implies that they require less movement to get their goals (*i.e.* to go to the vicinity of a cell), which may have a strong impact on the time and energy consumption.

According to these results, for a scenario with low cell and robot densities, reconfiguration is not worth it, because the energy consumption and time are like the non-reconfigurable robots or even smallest. But, in the cases where those densities are significantly higher, reconfiguration are best at time and energy.

This conclusion is quite obvious, as the energy consumption to move around is much higher than the one spent to make a reconfiguration. However, the comparison may not be the same with non-random movements is the search state.

TABLE III
Simulation results of reconfigurable vs. no-reconfigurable robots

Robots / Cells	64 Red: 32 Blue: 32		128 Red: 64 Blue: 64		256 Red: 128 Blue: 128	
Recon-figurable	Time (s)	Energy consump. (μWH)	Time (s)	Energy consum. (μWH)	Time (s)	Energy consum. (μWH)
16	1408	644.8	219	99.5	155	70.7
64	1469	379.8	596	264.9	262	116.5
256	5959	2628.9	802	385.3	273	113.4
1024	70147	31872.9	1747	638.7	414	172.8
Non-recon-figurable	Time (s)	Energy consump. (μWH)	Time (s)	Energy consum. (μWH)	Time (s)	Energy consum. (μWH)
16	1882	827.5	454	206.9	155	70.3
64	3700	1589.8	1929	935.6	535	249.5
256	13368	5746.9	6282	2758.6	593	326.3
1024	~140K	51168.2	67670	30997.4	1437	1002.7

TABLE IV
Simulations results for different red & blue robot ratios

Robots / Cells	Red: 32 Blue: 96 1/4		Red: 64 Blue: 64 1/2		Red: 96 Blue: 32 3/4	
	Time (s)	Energy consum. (μWH)	Time (s)	Energy consum. (μWH)	Time (s)	Energy consum. (μWH)
16	300	136.66	219	99.48	230	105.55
64	305	136.04	596	264.87	562	251.20
256	762	320.82	602	385.27	1985	848.76
1024	1015	360.00	1747	638.65	3419	1342.78

This conclusion is quite obvious, as the energy consumption to move around is much higher than the one spent to make a reconfiguration. However, the comparison may not be the same with non-random movements is the search state.

B. Red & Blue Robot Ratios

In this case, robot swarms do not have the same number of red and blue robots, initially. Simulated scenarios include ratios of ¼, ½, and ¾ with a swarm of 128 microrobots. Again, the system is simulated for the same variety of cell quantities: 16, 64, 256, and 1024. Robots and cells are randomly placed in the arena.

Results of corresponding simulations are shown in Table IV and seem to reflect that the best initial distribution is half and half for red and blue robots.

In fact, notice that reconfigurability provides a mechanism of adaptation of the swarm to the environment and can compensate a bad initial guess.

VI. Conclusion

In this work, we have adapted an existent multi-robot system simulator, which worked basically as a particle simulator, so to add physical parameters to them. Particularly, they take up a given area and, in case of robot-particles, have a given battery charge.

Also, we have proposed a high-level individual robot behavior based on a toy problem that we created. Each state of the corresponding ASM is associated with power, energy consumption, and time parameters and can be easily re-programmed to change the functionality at each state.

The simulator may operate interactively or in batch mode, which is used to chain several runs so to help in analyzing time and energy consumption of a set of cases. It is the core of the simulation framework, which has been tested for a simple and direct solution of the toy problem with Parallax Boebot educational minirobots.

Finally, as the primary motivation of this work is to have a framework to analyze the efficiency of microrobot swarms with stringent computational limitations, we have devised its utility for realistic, biomedical applications, whenever microrobots are available with the required sensor and

actuator accuracies.

This case has been done with physical parameters estimated from the I-SWARM microrobot, which is a 3x3x3mm robot with a solar cell to obtain energy from the environment, a locomotion system based on three vibrating legs, and an 8051 microcontroller onboard. In our case, it is devised to carry an AFM tip to perform operations at that level.

We expect that this simulation framework is further used with accurate parameters to best design the behavioral algorithms of individual microrobots in swarms, particularly for biomedical applications.

ACKNOWLEDGMENT

This work is partially supported by the Spanish project COCOABOTS (MCYT TEC2005-08066-C02) and the European project ISWARM (FP6-2002-IST-507006).

REFERENCES

[1] J. Otero, M. Puig-Vidal, Ll. Ribas-Xirgo, "Multirobot Control System Design for Nanomanipulation Applications," *Int'l. J. of Factory Automation, Robotics and Soft Computing*, Italy, Dec. 2007.

[2] —, *I-Swarm: Intelligent Small World Autonomous Robots for Micro-manipulation*, IST FET-open EU Project 507006, 2005–2008. [Available on-line: www.i-swarm.org]

[3] Ll. Ribas, E. Ramon, "A minirobot architecture for edutainment and research," *Autonomous Minirobots for Research and Edutainment*. Heinz-Nixdorf Institute, Paderborn (Germany), Sept. 2007.

[4] Luiz Chaimowicz, Mario Campos, and Vijay Kumar. "Simulating Loosely and Tightly Coupled Multi-Robot Cooperation". *In Anais do V Simpóosio Brasileiro de Automaçao Inteligente.* (SBAI), Canela, RS, Brazil, September 2001.

[5] J. Colomer, A. Saiz, P. Miribel, J. Maña, J.Brufau, M. Puig, and J. Samitier. "Low-Voltage Low-Power Reference Circuits for an Autonomous Robot: I-SWARM", *Proc. of the 11th WSEAS Int'l. Conf. on Circuits*, Agios Nikolaos, Crete (Greece), July 23-25, 2007.