Robust Electronic Hardware System Based on `Quorum Sensing`

Fredy Hernán Martínez Sarmiento

National University of Colombia
Faculty of Engineering
Department of Computer Science
Bogotá, D.C.
August 2017

Robust Electronic Hardware System Based on `Quorum Sensing`

Fredy Hernán Martínez Sarmiento

Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor en Ingeniería de Sistemas y Computación

Advisor
Jesús Alberto Delgado Rivera, Ph.D.

National University of Colombia
Faculty of Engineering
Department of Computer Science
Bogotá, D.C.
August 2017

Title in English

Robust Electronic Hardware System Based on `Quorum Sensing`

Título en español

Sistema Hardware Electrónico Robusto Basado en `Quorum Sensing`

Abstract: Electronic systems designed by man have a high level of complexity. This feature joined with its sequential operation has enabled the susceptibility to failures. In critical applications, this problem has been traditionally attacked duplicating the entire system (redundancy), and using a central control system. The term robust, word appearing in the formulation and in the title of this research, refers to the system's ability to withstand damage (usually physical) without losing its functionality or operation.

This research proposes a different design scheme inspired by a mechanism of gene expression control, mechanism which is dependent on cell density, and has been recently observed and characterized in the field of systemic biology, in medical research regarding the efficacy of antibiotics; gene expression responsible for social behaviors of independent cells (bacteria) using extracellular signals.

In the biological model, the cell-cell communication is performed through the exchange of chemical molecules called auto inducers. This process, called `Quorum Sensing` (QS), allows bacteria to monitor their environment for the presence of other bacteria, and thus, to respond to fluctuations in the number and/or species present. The great parallelism of this kind of systems involves a great robustness, since it avoids the sequential structure and dependence on a central control system. In this research, a mathematical model of this process (QS) is proposed, and then, the use of this model to design some initial electronic applications, specifically in the area of robotics is shown.

This model starts with the overall system behavior, and then, it focuses in an individual-level model. This design strategy allows, from the application criteria, to define the rules of behavior of each bacterium, which are the same throughout the community, in a similar way as occurs with cellular automata. In principle, the algorithm can be used in signal processing problems as originally formulated in the proposal, if these problems are formulated as a search problem. In this research the algorithm is implemented in some problems of navigation of robots establishing the navigation route as the search problem.

The research objective is not to create hardware with evolving capacities, but to propose a scheme of hardware design that reflects a redundant structure. This means formulating a behavioral algorithm that can be implemented functionally on a hardware (microprocessors, microcontrollers, CPLDs, FPGAs, robots, etc.). For this purpose, research proposes the use of different platforms on which different levels of collective bacterial structures are evaluated, while maintaining the premise of QS local communication.

Resumen: Los sistemas electrónicos diseñados por el hombre presentan un elevado nivel de complejidad. Esto unido con su característica de operación secuencial, ha permitido que en general se incremente la susceptibilidad a fallos. En aplicaciones críticas, este problema se ha atacado tradicionalmente con la duplicación de todo el sistema (redundancia), y el uso de un sistema de control central. El termino robusto, término que aparece en la formulación y en el título de ésta investigación, se refiere a la capacidad del sistema de

soportar daños (en general físicos) sin perder su funcionalidad u operación.

Esta investigación propone un esquema de diseño diferente inspirado en un mecanismo de control de expresiones genéticas dependiente de la densidad celular, observado y caracterizado recientemente en el campo de la biología sistémica, durante investigaciones médicas en relación con la eficacia de antibióticos. Este fenómeno es el responsable de que un conjunto de células independientes (bacterias), bajo la generación de señales extra celulares, desarrolle comportamientos sociales coordinados.

En el modelo biológico, la comunicación de célula a célula se realiza a través del intercambio de moléculas químicas llamadas auto inductores. Este proceso, denominado `quorum sensing` (QS), permite que las bacterias supervisen su ambiente para detectar la presencia de otras bacterias, y responder así a las fluctuaciones en el número y/o especies presentes. El gran paralelismo de éste tipo de sistemas conlleva una gran robustez, dado que evita la estructura secuencial y la dependencia a un sistema de control central. Aquí se plantea un modelo matemático de éste proceso (QS), y se utiliza luego éste modelo para el diseño de algunas primeras aplicaciones electrónicas, específicamente en el área de robótica.

El modelo que se propone parte del comportamiento global del sistema, y luego se enfoca a un modelo a nivel de individuo. Esta estrategia de diseño permite, a partir de los criterios de la aplicación, definir las reglas de comportamiento propias de cada bacteria, que serán las mismas de toda la comunidad, de forma similar a como ocurre con los autómatas celulares. En principio, el algoritmo puede utilizarse en problemas de procesamiento de señales como originalmente se formuló en la propuesta, si estos problemas se formulan como un problema de búsqueda. En esta investigación el algoritmo se implementa en algunos problemas de navegación de robots estableciendo la ruta de navegación como el problema de búsqueda.

El objetivo de la investigación no es crear hardware con capacidad evolutiva, sino proponer un esquema de diseño de hardware que refleje una estructura redundante. Esto significa formular un algoritmo de comportamiento que pueda ser implementado funcionalmente sobre un hardware (microprocesadores, microcontroladores, CPLDs, FPGAs, robots, etc.). Para ello, la investigación propone el uso de diferentes plataformas sobre las cuales se evalúan diferentes niveles de estructuras bacteriales colectivas, manteniendo siempre la premisa de comunicación local del QS.

Keywords: Biological behavior, electronic design, multiscale modeling, quorum sensing, robust systems.

Palabras clave: Comportamiento biológico, diseño electrónico, modelamiento multiescala, quorum sensing, sistemas robustos.

Claims: The main contribution of the research is the formulation of a new algorithm inspired by bacterial QS which is able to accelerate the convergence of any other search algorithm, informed or not. The algorithm mimics the aggressive behavior of the bacteria when the population size exceeds a certain threshold. When this aggressive behavior is manifested, biological bacteria attack their host. The algorithm in the artificial bacteria experiments a similar activation, and the effect is to increase the level or weight of the solution in proportion to the number of aggressive bacteria. This algorithm was verified experimentally.

The hardware design is implemented on two different platforms: embedded devices

and small mobile robots, on them the principles of local communication and virulent activation were implemented. Given the research stay (9 months) in the Motion Strategy Lab, Department of Computer Science at the University of Illinois (Urbana - USA), the proposed algorithm was evaluated in robotics, specifically in navigation tasks for autonomous robots in complex, dynamic and unknown environments. Different strategies were used for evaluation of the navigation algorithm. The robustness and performance evaluation was carried out on a simplified artificial potential field algorithm, known in robotic for its simplicity and its local minima problems.

The traditional algorithm of artificial potential field is used to design a potential field in accordance with the navigation path. Other features of the algorithm (repulsive force of the obstacles and the resulting calculation of motion) were replaced by an ergodic behavior, research area of the Motion Strategy Lab. In addition, the central control unit was removed to give movement autonomy to the robots, which identified the potential field strength through landmarks in the environment (local interaction of the robot).

The proposed algorithm based on QS was added to this navigation scheme, allowing assessments to task performance with and without QS. Throughout 300 simulations for a total of 20 robots in each case, and all on the same navigation environment, it was found that the addition of the QS to the navigation scheme reduced the navigation time by almost 60%. In this kind of applications, this reduction has a major impact on search and rescue tasks in environments that greatly limit the robots sensors (collapsed environments).

Contribuciones originales: El principal aporte de la investigación es la formulación de un nuevo algoritmo inspirado en el QS bacterial que es capaz de acelerar el proceso de convergencia de cualquier otro algoritmo de búsqueda, informada o no. El algoritmo imita el comportamiento agresivo de las bacterias cuando el tamaño de la población supera un determinado umbral. Cuando este comportamiento agresivo se manifiesta, las bacterias biológicas atacan a su anfitrión. El algoritmo en las bacterias artificiales experimenta una activación similar, y el efecto es aumentar el grado o peso de la solución en proporción al número de bacterias agresivas. Éste algoritmo fue verificado experimentalmente.

El diseño hardware se implementa sobre dos plataformas diferentes: dispositivos embebidos y pequeños robóts móviles, sobre ellas se implementaron los principios de comunicación local y activación virulenta. Dado la estancia de investigación (9 meses) en el *Motion Strategy Lab* del *Department of Computer Science* de la University of Illinois (Urbana - USA), el algoritmo propuesto se evaluó en robótica, específicamente en tareas de navegación para robots autónomos en ambientes complejos, dinámicos y desconocidos. Se utilizaron diferentes estrategias de navegación para la evaluación del algoritmo. La evaluación de robustez y desempeño se realizó sobre un algoritmo de campo de potencial artificial simplificado, conocido en robótica por su sencillez y sus problemas de mínimos locales.

El algoritmo tradicional de campo de potencial artificial se utilizó para diseñar un campo de potencial en concordancia con la ruta de navegación. Las demás características del algoritmo (fuerza de repulsión de los obstáculos y cálculo de la resultante de movimiento) fueron reemplazadas por un comportamiento ergódico, área de investigación del *Motion Strategy Lab*. Además, se eliminó la unidad de control central para dar autonomía de movimiento a los robots, los cuales identificaron la intensidad del campo de potencial a través de *landmarks* en el ambiente (interacción local del robot).

El algoritmo propuesto basado en QS se agregó a éste esquema de navegación, lo que

permitió hacer evaluaciones de desempeño de la tarea con y sin QS. A lo largo de 300 simulaciones con un total de 20 robots en cada caso, y todas sobre el mismo ambiente de navegación, se encontró que la adición del QS al esquema de navegación redujo el tiempo de navegación en casi un 60%. En éste tipo de aplicaciones, esta reducción tiene un gran impacto en tareas de búsqueda y rescate en ambientes que limitan considerablemente los sensores de los robots (ambientes colapsados).

Acceptation Note

Jury
Prof. Rafael Murrieta-Cid

Jury
Prof. Mario R. Arbulú

Jury
Prof. César A. Pedraza

Advisor
Prof. Alberto Delgado R.

Bogotá, D.C., August 2017

Personal declaration

I declare that I have worked on this thesis independently and with only the help of the means permitted and no different from those mentioned in the thesis itself. I recognize in this work (sources/resources listed in the bibliography) all texts taken of textual or figurative way from documents published and unpublished. No part of this work has been used in any other thesis, at this or any other university. The results, simulations and experiments are complete responsibility of the author.

Dedicated to

I would like to dedicate this work to all my family.

To my parents Alda and Johnny, for their understanding and help in times bad and less bad. They have taught me to face adversity without losing dignity will not fail in the attempt. They have given me everything I am as a person, my values, my principles, my perseverance and my efforts, all with a great deal of love and never asking anything in return.

To my children, Laura, Juan José, Juan Esteban, Ana María and Samuel. They are the best thing ever happened to me, and have come to this world to give me the final push. They are definitely my reference for the present and future, and the strength that lets me to get the balance that allows me not give up.

And finally, thanks to my students and friends who endured this long process with me, always offering support and patience... that angel who takes care of me... To those special people who have always been by my side, present and in thought. They who always supported me and expected the best of my... and that no matter how long they will always be in my heart.

To all of them... Wholeheartedly thank you very much.

Acknowledgements

First and most importantly, I would like to sincerely to thank to my advisor, professor Jesús Alberto Delgado, his effort and dedication, his knowledge, his guidance, his way of work, his persistence, patience and especially his confidence in me have been fundamental to my development as a researcher. He has instilled in me a sense of seriousness, responsibility and academic rigor without which there could be no a complete training as a researcher. In his own way, has been able to earn my loyalty and admiration, and I feel in debt to him for everything received during the time that has lasted this work.

I would also like to acknowledge the advice received over recent years by other professors of the Ph.D in Computer and Systems Engineering of the National University of Colombia (Bogotá), and professors from other universities who believe and support the doctoral program and that in one way or another, have contributed much more than a bit to my training. I highlight particularly the professor Luis Fernando Niño, the professor Steven M. LaValle (Department of Computer Science - University of Illinois) and professor Max Garzón (Department of Computer Science - University of Memphis). Likewise, I thank professors from other departments of the Faculty of Engineering for their humane treatment and critical view of many aspects of everyday life that help shape a person. I would name a lot, but highlight in particular the professors Rafael Murrieta, Mario Arbulú and César Pedraza, for the comments and suggestions made to this research, which certainly increased their level and rigor.

And last but not least, I am eternally grateful to my colleague accidentally, professor Leonardo Bobadilla (School of Computing and Information Sciences - Florida International University). A great person, with boundless enthusiasm that allowed me to work with him for several months. The working environment created was just perfect, and his vision, motivation and optimism have helped me in critical moments of the research. A great friend.

I am the sole author and responsible for this document, but would have been very difficult to develop it without them. To them... Thank you very much for everything.

Contents

List of Tables

List of Figures

Units, dimensions, and symbols

In general, the units and dimensions used in this document conform to the International Standard (SI) System.

Symbols

- $\neq$ Not equal to
- $\forall$ For all
- $\in$ Belongs to
- $\subset$ Subset of
- $\subseteq$ Subset of, or is included in
- $\setminus$ Relative complement. A complement of a set A refers to things not in (that is, things outside of) A. The relative complement of A with respect to a set B, is the set of elements in B but not in A $\left(B \setminus A = \overline{A} \cap B\right)$.

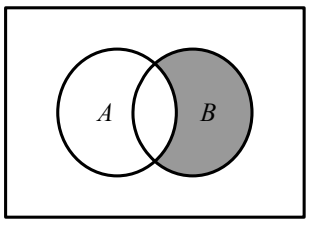


Figure 1: Relative complement $B \setminus A$.

- $\Rightarrow$ Implies.
- a The total number of behaviors that can have an agent within the model.
- α Heading. Angular position of the robot on the plane in the kinematic model.
- $\dot{\alpha}$ Rotational velocity of the robot.

- b Number of state variables (vectors) x required to describe $\boldsymbol{X}_p$. In the proposed algorithm, the value of b is equal to the number of grouping areas defined by the agents in the system.
- β Brownian particle density. Variable used to determine the motion of Brownian particles in time, considering their collective movement rather than their individual movement.
- C Set of colors. In the case of using marks on the environment as virtual gates, these marks are identified by robots because of their color.
- c A color of the color set C.
- Γ Set of virtual gates. A set of special elements located in the environment (ramps, color marks, paper gates, ...) that are identified as gates for the robots (agents). In the mathematical model, each gate $\gamma_i \in \Gamma$ reflects a 0 or a 1 if it is interpreted as an open or closed gate by robots.
- γ Virtual gate. It is a landmark in the environment that is identified as a gate by robots.
- D Discrete transition system. Action model for the hybrid system.
- D_C Diffusivity or diffusion coefficient is a proportionality constant between the molar flux due to molecular diffusion and the gradient in the concentration of the species (or the driving force for diffusion). Diffusivity is encountered in Fick's law and numerous other equations of physical chemistry.
- d_{ij} Physical distance between bacteria V_i and V_j, which is calculated by any appropriate norm. In general, this operator is used to indicate the distance between two points.
- E Environment, the surroundings of a physical system that may interact with the system by exchanging mass, energy, linear momentum, angular momentum, electric charge, information, or other conserved properties.
- f Nonnegative integer that indicates the number of bacteria in direct contact with the bacterium under analysis. It is a variable indicating the number of neighbors of the bacterium.
- $g : S_1 \longrightarrow S_2$ Information mapping, process by which an agent creates a model of its surroundings by performing appropriate sensor actions. The nomenclature indicates that a function g mapping a set S_1 into a set S_2.
- H Hybrid system. Hybrid systems are dynamical systems with interacting continuous-time dynamics (modeled by differential equations) and discrete-event dynamics.
- h Density threshold. Indicates the minimum distance between bacteria needed to define the bacterial population in order to implement the final task. It is a variable that indicates the size of the system and helps regulate their growth.
- h_{CS} Function that models the behavior of the contact sensor.

- h_{IS} Function that models the behavior of the intensity sensor.
- η Neighborhood threshold. It indicates the maximum amount of direct neighboring bacteria that a bacterium can have.
- $\dot{\theta}$ The speed of the wheel of the robot (differential wheeled robot).
- $\dot{\theta}_0$ The rated speed of the wheel of the robot (differential wheeled robot).
- $\Theta(\star)$ This function takes the value of 1 if $\star > 0$, and 0 otherwise. The function allows us to describe the activation of QS.
- I Variable used to determine the intensity of a signal transmitted by the robot.
- $\mathcal{I}$ Information space. For a system that stores an observation history, an information state, at a given time, is the set of this history plus the initial conditions. The set of all possible information states is called the information space.
- ι In the analysis of the agent as a hybrid system, where the genome is analyzed as a filter, corresponds to an encoding of a particular behavior.
- k_i The per capita rates at which the bacteria in the area i are activated, or become virulent.
- $\boldsymbol{L}_a$ Discrete control mode in the hybrid model at the agent level.
- $\boldsymbol{L}_p$ Discrete control mode in the hybrid model at the system level.
- l Possible behavior of an agent in the proposed model.
- l_a The set of all possible behaviors of the agents.
- λ_i The per capita rates at which recruiters bacteria take the bacteria to area i.
- m Intensity function of the signal identified by each of the agents.
- μ_i The per capita rates at which explorers bacteria discover the area i.
- n Number of agents, artificial bacteria or robots considered in the formulation of a problem.
- O Obstacle in $\mathcal{W}$. The obstacles are pairwise-disjoint and countably finite in number.
- $\mathcal{O}$ Set of obstacles.
- $\pi : S_1 \longrightarrow S_2$ Control policy. Control policy is formed by rules that allow to define control decisions according to the agent's local readings. The nomenclature indicates that a function π mapping a set S_1 into a set S_2.
- $P(x_1, x_2)|_{p_0 \to p_1}$ Function that defines the navigation path from the point p_0 to point p_1 in $\mathbb{R}^2$.

- p It is a dimensional coordinate indicating the exact physical position of the bacterium into the system. Two-dimensional coordinates for all systems analyzed in this research, in which agents move in a plane (a point in $\mathbb{R}^2$).
- $\mathbb{R}^q$ The q-dimensional Euclidean space.
- R Set of bacteria in the reproduction stage.
- $\mathcal{R}$ Collection of all regions r of $\mathcal{W}$. These regions are induced by the virtual gates in the environment.
- r Region which consists of the virtual gate and connecting cells.
- ρ_{ij} The per capita rate at which explorers bacteria and virulent bacteria of area i encounter site j and switch their allegiance by becoming explorers and virulent bacteria, respectively, of that site.
- s Fitness function for the adaptive linear combiner with recurrence.
- T Quorum threshold. It is a constant defined for the task, responsible for limiting the growth of the system (indirectly it defines the amount of bacteria in the system, limiting the reproductive stage).
- U Set of all possible control modes.
- u Control mode. It assigns a particular behavior to a particular local reading performed by the robot in the environment.
- V Mathematical representation of the artificial bacterium. It is a pair of numbers physically identifying the bacterium in the system.
- $\mathbf{v}(x_i)$ It is the value of the speed field sensed by the robot i at position x_i. It corresponds to the field designed for the environment, and contains the information of navigation path.
- v Translational velocity of the robot.
- W_0 Bacterial population. It is as a nonempty set of bacteria V_i, with non-zero distance between bacteria ($d \neq 0$).
- W Application Bacteria AB. It is a subset of the bacterial population comprised only by virulent or active bacteria. They are bacteria that develop the task.
- $\mathcal{W}$ Closure of a contractible open set that has a connected open interior with obstacles that represent inaccessible regions. Space of action of bacteria in the Euclidean space (agents implemented on robots in this research).
- $\partial\mathcal{W}$ The boundary of a set $\mathcal{W}$.
- $\varphi(\cdot)$ Symmetric hard-limit function defined as:

$$\varphi(n) = \begin{cases} \dot{\theta_0} & \text{if } n \geqslant 0 \\ -\dot{\theta_0} & \text{otherwise} \end{cases} \tag{1}$$

- $\phi : S_1 \longrightarrow S_2$ Filter. Structure used to perform the information feedback of the robots. The nomenclature indicates that a function ϕ mapping a set S_1 into a set S_2. In the hybrid model of the agent, is used to interpret the agent's local observations and control policies with the aim of defining the new behavior of the agent.

- $\boldsymbol{X}$ State space representation, mathematical model of a physical system as a set of state variables related by differential equations.

$$\dot{\boldsymbol{X}} = \mathbf{A}\,\boldsymbol{X} + \mathbf{B}\,\mathbf{u} \tag{2}$$

- $\boldsymbol{X}_a$ State space representation for the agent.

- $\boldsymbol{X}_p$ State space representation for the system.

- X Set of bacteria interacting locally with neighboring bacteria and the environment, but without activating its virulence.

- x State variables (vectors), are the smallest possible subset of system variables that can represent the entire state of the system at any given time.

- Y Observation space, is the space that contains all the variations of a vector field, and thus it allows to determine the observability of a nonlinear system (it is particularly used in nonlinear systems, but it is equally valid for linear systems).

- y Sensor observation event.

- $\tilde{y} : S_1 \longrightarrow S_2$ Observation history, is the ordered sequence of observations made by an agent into an environment. The nomenclature indicates that a function $\tilde{y}$ mapping a set S_1 into a set S_2.

- $\mathbb{Z}$ Symbol used to denote the set of integers. It comes from Zahlen, German for numbers.

- Z Set of bacteria in the activation or virulence stage.

- ω Tunable coefficients of adaptive linear combiner with recurrence.

Introduction

Artificial systems have been developed by man throughout history to support his daily activities. This development can be seen throughout history in cases like the wheel in transport solutions, or more recently, in the current space exploration systems. Today, these advanced artificial systems are composed of two subsystems: hardware and software. Traditionally the development of these two subsystems has been independent, but today there is a clear need and benefit of integrated design, which has been called Cyber-Physical Systems (CPS) (Sztipanovits et al., 2013; Lee, 2008).

There are two paradigms that have inspired the development of these artificial systems. From the point of view of hardware and software design, it should initially consider two fundamental theories which, although they are developed independently, in terms of their use are closely linked: The communication theory of Claude Shannon and the computational model of Alan Turing.

In the first case, Shannon showed that the information can be analyzed independently of the content, focusing only on abstract features which includes the transmission of information between a transmitter and a receiver, and analyzing the problems and limitations that such communication has on a single channel (Shannon, 1948).

In Shannon's model, the communication system comes from a source of information from which a transmitter sends a signal. This signal travels through a channel, with the possibility of being interfered by noise. The channel delivers the signal to the receiver who identifies the message.

On the other hand, and to complete the analogy, we have the computational model of Turing (Logical Computing Machine - LCM, (Turing, 1937)). The Turing model corresponds to an automaton that moves on a linear data sequence. At each instant, the machine can read only a data of the sequence, and perform certain actions according to a table that considers its current state and the last data read. Among the possible actions include the ability to write new data to the stream, through the sequence in both directions, and change state within a finite set of possible states.

Beyond these two paradigms, this research proposes a new model based on the principles of integrated design of the CPS. The model raises the hardware and software as part of the information processing problem, and therefore, assumes no initial existence of any of them. Furthermore, the structure is not restricted to a sequence of processes, but it seeks a structure of parallel sequences (Fig. 2).

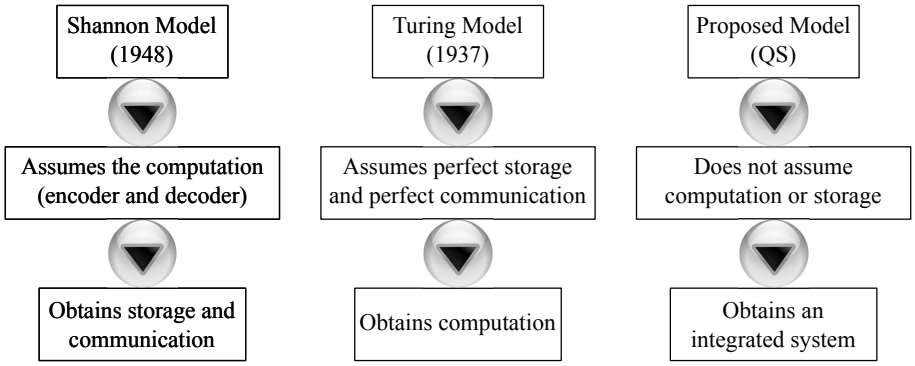


Figure 2: Comparison of classic models of information processing and proposed model

Inspired by Prof. Madhu Sudan talk (Microsoft, New England) entitled: *Communication & Computation: A need for a new unifying theory*, developed on September 19, 2011. University of Illinois Urbana-Champaign.

The design of artificial systems continues today, in general terms, these two classic models. However, today's applications are very different, much more complex. If it is wanted robustness in these artificial systems, then we need our systems to use more complex channels, composed of many smaller and clever sub-channels, and it is needed to store/communicate the information differently, without losing sight of the scalability of the system, and where it is able to merge computation and communication more tightly.

The systems today tend to be structurally sequential. These systems consist of a large number of elements that communicate sequentially. This communication may be at the software level (data and/or programs) between devices or between pieces of software; or hardware level, between devices or complete systems. Although this sequential structure has proven to work very well, it has also been shown to have the big problem of fail if any of its components fails. This is the problem of robustness that seeks to solve this research.

The solution to this problem requires a model that incorporates some level of intelligence. As a starting point of analysis is possible to consider the anthropological paradigm of intelligence, the human being. A key concept in psychology related to human behavior, states that this is addressed, regarding their actions in the world, in a general way not by reality but by the representation that the subject makes of it, which is influenced by prior knowledge, goals and purposes (Miller, Galanter, & Pribram, 1960). Thinking a bit on these ideas, we can conclude that this is also true for artificial agents.

The physical world, which we define as the state space $\boldsymbol{X}$, in which we can find an artificial agent in a state $x \in \boldsymbol{X}$, is not perceived as such by the agent. Instead, the agent knows the environment E where it moves from observations, using sensors that let it build an information space $\mathcal{I}$. x is estimated by sensor readings. An information mapping is of the form:

$$g: \ E \longrightarrow Y \tag{3}$$

wherein Y denotes an *observation space*, constructed from the sensor readings over time, i.e., through an observation history of the form:

$$\tilde{y} : [0, t] \longrightarrow Y \tag{4}$$

The interpretation of this information space, i.e., $\mathcal{I} \times Y \longrightarrow \mathcal{I}$, is that which enables the agent to make decisions (LaValle, 2006).

As seen, for the artificial agents, in a similar way to the biological, the interaction with the environment involves gathering and representing information (encoding), holding information (memory write), and getting at the information when needed (memory read). Besides, in building this information space, it is essential to consider the task that will develop the agent and the previous information that is stored in it. These concepts are fundamental to the design of artificial systems, and promote a sense of design which seeks to obtain the system required to solve a task (minimalist design) (Bobadilla, Sanchez, Czarnowski, & LaValle, 2011).

In addition to the problem of reliability in these communications, where sequential structure makes a device failure leading to system failure, these communication structures involve extra consumption of resources (primarily time), they become unreliable and insecure.

When the designer wishes robustness in the operation of an artificial system, he normally uses one of the following strategies:

- Fault avoidance, using formal specification and verification methods and a rigorous development process.
- He uses redundancy in the system.

Both strategies involve a high cost, reason why its use is restricted to safety critical applications or with a high degree of availability, because downtime is very expensive. In the first case, the designer takes high-assurance process, involving high development costs, which eventually transferred to the final system, without ignoring the sequential structure. In the second case, the designer opts for parallelism.

The systems can be viewed as hybrid structures, wherein a main system operates normally, but in case of failure is replaced by another (Sha, 2001), sometimes with simpler structure and higher reliability, which controls the system until the main system can take over again. Redundant elements are physically in parallel, but they never operate simultaneously. In addition, they require a decision logic to evaluate the performance and coordinate the switching (Fig. 3).

The problems of interaction, communication and processing have been solved by nature differently, and they are surprisingly robust. There are many examples of biological behavior that can be cited: the collective work of bees and ants, the functioning of an organ composed of cells, the communitary life of bacteria, or the way in which humans communicate (Juba, Kalai, Khanna, & Sudan, 2011).

Self-assembly processes are responsible for the generation of order in nature. They involve components at different scales, such as molecules, cells, organisms, communities,

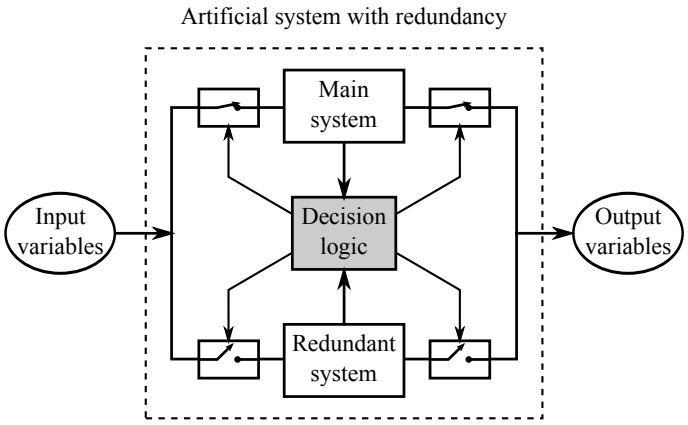


Figure 3: System with redundancy

ecosystems and weather systems. Scientists across many disciplines believe that the study of physical models of self-assembly can help in understanding nature and in advancing technology (Gross & Dorigo, 2008). These interactions between elements of relatively simple structure can be analyzed, modeled and replicated in the design of artificial systems.

Over the last few decades, research in bacterial cell-cell communication or `quorum sensing` (QS) has been quite intense (Karafyllidis, 2012; Wiedermann, 2011b). QS allows bacteria to coordinate their behavior and to act as one entity. QS controls microbiological functions of medical, agricultural and industrial importance and a better understanding of the underlying mechanisms and the conditions under which the signaling occurs; QS offers many possibilities for new applications.

In this research we sought to emulate the dynamic model of the interaction between bacteria. To do so, first we analyzed the simplified performance of QS on artificial hardware. This first emulation enabled us to understand the interaction in an artificial system, which allowed the development of descriptive mathematical models, one at system level and another at the agent level. Mathematical models were the tools used for the design of the proposed system. All this first part is presented in Chapter 2.

But these models are meaningless if they can not be applied to the design of artificial systems. In the original research proposal, we raised the possibility of using the QS scheme for processing signals, in particular for the recognition of the harmonic content of signals, however, the jury suggested a more impact approach. During the discussion of the proposal, they raised the need for a different application, one that exploded the possibilities of the architecture, and it was an open problem, unresolved, of the current engineering.

The answer to this quest came with the doctoral research internship. The doctoral research internship was performed in the Motion Strategy Lab of the Department of Computer Science at the University of Illinois (Urbana, USA), under the direction of Professor Steven LaValle. His work in robotics, and specifically in path planning, provided us with a real problem feasible to be attacked with our architecture. However, extensions of our basic ideas can be applied to the solution of other problems.

To solve problems with our algorithm based on QS, we analyze the problem from its most basic form to reach the general solution. First, we present the concept of wild motion,

and their dynamic properties, from which we can solve basic navigation problems. Then, we apply our model of behavior based on QS in conjunction with the wild motion to analyze a basic application of grouping using local communication. And finally, we apply all this theory to a general case of navigation on defined paths using as navigation strategy landmarks in the environment. All this application part is presented in Chapter 3.

However, before showing all this design and theoretical analysis, we present in Chapter 1, a short state of the art in relation to the study of bacterial QS. At the end of the document, we conclude the research.

CHAPTER 1

Background

1.1 Motivation

The operational and performance demands of artificial systems required today to facilitate the human activity, have acquired a high level of complexity; while aspects such as reliability and autonomy have been formed as important and inherent characteristics. The design of a system (automobile, building, home to the dog,...) requires the integration of different kinds of systems: electrical, electronic, mechanical, computational. The correct and efficient integration of these subsystems is what allows a great performance in the final system. Such systems are known today as Cyber-Physical System (CPS).

A CPS is a system in which the computational and physical elements of a system are tightly intertwined (Sztipanovits et al., 2013; Lee, 2008). It is often not possible to categorize a specific behavior of a CPS as being the result of computational elements or a unique physical law. Normally in a CPS the physical processes are in the feedback loops, affecting the calculations and the same physical processes. This structure is the reason why traditional engineering design techniques based in the development of a comprehensive model of the system can not be applied with ease. This, interestingly, also runs counter to the model of professional training curricula based on specific orientation (Electrical Engineering, Systems Engineering).

This holistic design concept, which first attempts to analyze the problem from the point of view of the multiple interactions that characterize it as a complex system, and then, from a holistic approach, produce a design adapted to the specific needs and conditions, not only solves the problem with a tailored solution, but this solution is characterized by an appropriate use of resources.

In engineering applications, one of the primary concerns in the design of artificial systems is the robustness of the system. This robustness is defined as the ability of the system to remain functional despite partial failures. A system with the ability to support damage, self-repair or self-organize its structure to maintain its functionality solves major problems of autonomy in many real applications. This was the initial motivation for research.

However, for the design of a complex system involving electrical, electronic, mechanical and computational elements, and which we expect a significant improvement in terms of robustness, it was necessary to look for a different design model to those used traditionally. Thus, the traditional sequential design scheme is replaced by the idea of a parallel model, and inspiration was taken from existing and successful parallel structures, i.e. a biological structure.

1.2 Theoretical background

The QS is one of the most important mechanisms in the cell to cell bacterial communication. It has been described as "*the most consequential molecular microbiology story of the last decade*" (Busby & Lorenzo, 2001; Winzer, Hardie, & Williams, 2002). This mechanism allows coordinate collective behavior in bacteria under conditions of the environment. The QS relies on the activation of a sensor kinase or response regulator protein by, in many cases, a diffusible, low molecular weight signal molecule (a pheromone or autoinducer, Fig. 1.1) (Camara, 2006).

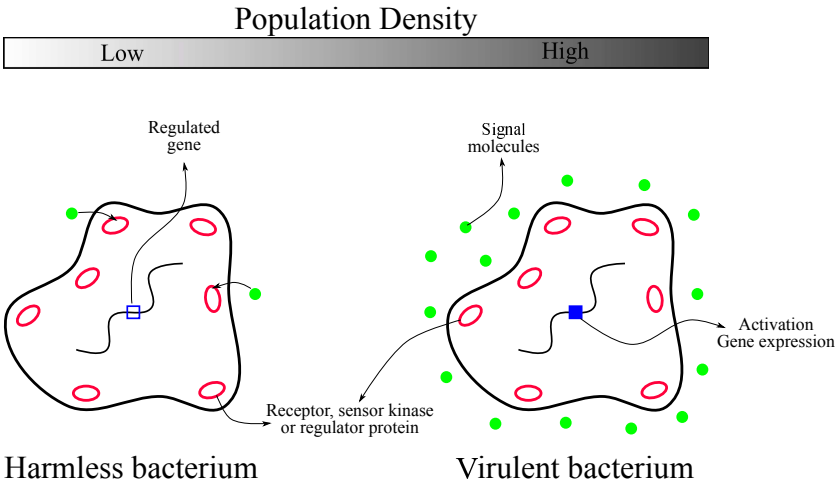


Figure 1.1: Bacterial `quorum sensing`

During operation of this mechanism, the concentration of the signal molecule reflects the number of bacterial cells in a particular area. The detection of a threshold concentration, a molecular signal, indicates that the population has reached the quorum, i.e., is ready to perform a specific collective behavior. This means that QS is a mechanism used by bacteria to activate phenotypic changes in the population, i.e., to coordinate gene-expression. This activation of the system (system made by bacteria) under specific conditions experienced by them, was the main motivation paradigm for the development and implementation of an electronic design model.

1.2.1 Models based on bacteria

Specifically in the development of computer models based on cell-to-cell communication in bacteria, there has been a large number of new publications since the doctoral research

proposal. For example, (Cho, Park, Jeong, & Chun, 2009) and (W. J. Tang, Wu, & Saunders, 2007) present two optimization algorithms based on bacterial foraging with QS. These algorithms seek to imitate the food-searching behavior of biological bacteria based on bacterial chemotaxis (which seeks to mimic the way in which bacteria react chemoattractants in concentration gradients), but with respect to the behaviors of reproduction and elimination-dispersal, and considering a fitness function. Another example is the model developed in (Dang, Brabazon, O'Neill, & Edelman, 2008), the researchers include a social swarming effect, very similar to the concept of QS, in order to provide faster convergence and more stable results. Other approaches to social interaction models are shown in (Zang, He, & Ye, 2010; Paton, Vlachos, Wu, & Saunders, 2006).

Other models of the dynamics of bacterial QS include descriptions of small RNAs that coordinate the process of collective organization (Shen & Zhou, 2010), evolving networks with reactive agents able to detect and update the system's structure (Yang, Liu, & Liu, 2010), and frequency synchronization of a set of cells coupled by QS (Taylor, Tinsley, Wang, Huang, & Showalter, 2009; J. Zhang, Liu, Li, & Chen, 2007). All these works are actually inspired investigations by the Bacterial Foraging Optimization (BFO) algorithm introduced by Passino in 2002 (Passino, 2002), which models the foraging behavior of *Escherichia coli* bacteria present in the human intestine.

Some authors propose an aging mechanism (Apolloni, Bassis, Clivio, Gaito, & Malchiodi, 2007) which develops artificial bacterial populations fighting against antibiotic molecules. Under this proposal, and using a very similar approach to this research, these authors do not interpret the strictly biological process, and the main element of the mechanism is based on individual reactions of agents. In addition, as in this research based on QS, the aging activation is triggered by a threshold.

Another strong area for the development of models is related to the P systems or membrane systems. In short, the idea is based on that in the membrane of cells, including bacteria, many reactions are developed related to normal cell activity, and that this membrane performs for the cell a work of modulation in the cell communication with the environment, and between the same cells. This membrane computing aims at defining computational models which abstract from the functioning and structure of the cell (Bernardini et al., 2006). There is a number of researches in which models have been developed based on these P systems that describe the behavior and virulence of bacterial QS. In (Bianco et al., 2006) for example, they work specifically in the communication mechanism between bacteria (the bacterium *Pseudomonas aeruginosa*), and focus on virulent behavioral activation. Similar research is presented in (Terrazas et al., 2005).

The proposed algorithm is bio-inspired in the sense that it takes important features of different bacteria. For example, the main feature is the robustness of the bacterial community as a system. A phylum with a lot of documentation about it is the *Proteobacteria*, which species such as *Methylobacteria*, *Asaia*, *Acinetobacter*, *Enterobacter*, and *Pantoea* have been the subject of study regarding population behavior against changes in diet and a variety of perturbations, including antibiotics and invading bacteria (Robinson, Schloss, Ramos, Raffa, & Handelsman, 2010; Wagner, 2005; Robinson, 2008). In these studies it shows how the structure of the community is essential to endure the attacks.

Another important feature identified in a large number of bacterial species is the modularity of its metabolic networks (Kreimer, Borenstein, Gophna, & Ruppin, 2007; Zhou &

Nakhleh, 2012). The metabolic network is the complete set of metabolic and physical processes that determine the physiological and biochemical properties of a cell. This feature has been identified in hundreds of bacterial species, including: obligate bacteria, specialized bacteria, aquatic bacteria, facultative bacteria, multiple bacteria (such as *Proteobacteria*), and terrestrial bacteria. Bacteria, like any other biological system, can be decomposed into nearly-independent structural units that perform specific functions (Parker, Kashtan, & Alon, 2007). In addition, each of these structural units exhibits a modular structure. Research shows that bacteria modularity is strongly dependent on the environment (for example, temperature and oxygen requirements), i.e., modularity can be modified when the environment changes over time (Parker et al., 2007).

These are the two main features taken from biological bacteria, and adapted to the proposed algorithm. They determine not only the behavior of artificial system, but its structural design. Both robustness (many individuals, where each one can perform any task in the system) and modularity (the same biological network is deployed at different times, in different places and in different organisms) are two biological characteristics that affect the ability of biological organisms to evolve (Wagner, 2005). In both cases these are biological strategies that provide adaptability, and therefore reduce the ability of genetic evolution. Consequently, the proposed algorithm does not consider evolution as it is understood in biology, but adaptation on different hardware platforms.

The architecture of our algorithm is based on the behavior between the agents of the system. For us, the system is comprised of a group of agents according to the definition given by (Russell & Norvig, 2002). Each agent corresponds to a simplified model of each bacterium of the real biological system. Under these ideas, the ability to function and interact with, a dynamic, changing environment is of great importance. The key element in this behavior based architecture is the design of controls for the agent to generate specific responses according to the local inputs of the sensors (Fig. 1.2).

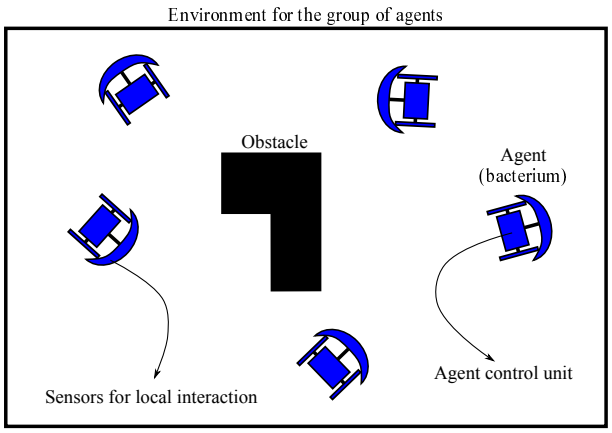


Figure 1.2: General architecture of the model based on QS

These characteristics are similar to those of the cellular automata (CA, cellular automaton, cellular spaces, tessellation automata, homogeneous structures, cellular structures, or iterative arrays). While there is no formal definition of traditional CA accepted, its description is characterized by the behavior of each agent, behavior that is determined by local interaction with its neighbors. Although this definition is closely related to the QS

model, in the CA the collective behavior is more related to emergent behaviors, while QS is related to pre-defined social behavior (the model mimics a real biological community with pre-defined responses to certain stimuli). Moreover, since it is a model for a biological process, this allows more complex interactions, and even different characteristics between bacteria (different species) which there is no in the definition of CA. However, it is clear that our model can be interpreted as a CA with QS rules.

Our idea of using a hybrid automaton (mathematical model for precisely describing systems in which digital computational processes interact with analog physical processes) to model a behavior based control system for a community of autonomous agents is nothing new, in robotics, it has been used widely to solve collective navigation problems (Egerstedt & Hu, 2002; Alur, Henzinger, Lafferriere, & Pappas, 2002; Egerstedt & Hu, 2001; Finucane, Jing, & Kress-Gazit, 2010; Lahijanian, Wasniewski, Andersson, & Belta, 2010; Smith, Tumova, Belta, & Rus, 2010; Kloetzer & Belta, 2010). Moreover, the layered structure modeling that we use to describe the interaction of bacteria, it has already been previously used by other authors to describe collective behavior in bees, ants, molecules and other collective biological models (Berman, Halász, Kumar, & Pratt, 2007a; Scheidler, Brutschy, Diwold, Merkle, & Middendorf, 2011; Wang, Zhao, & Liu, 2009; Whitesides & Boncheva, 2002), and even in other approaches to bacterial QS, as the case of (Lees, Logan, & King, 2007; W. Tang, Wu, & Saunders, 2007), where instead of using differential equations, they use a stochastic approach to the macro-scale model. On these same biological referents, have also developed mathematical models that assume hybrid structures, where agents switch between different system behaviors (Brutschy, Scheidler, Merkle, & Middendorf, 2008).

1.2.2 Applications

Perhaps the most logical application of a bio-inspired model is in search and optimization problems. For many researchers, all engineering problems can be formulated as a search problem. A first example of such applications using the bacterial foraging algorithm of (Cho et al., 2009) can be observed in (Cho & Kim, 2011). These Koreans researchers applied their algorithm inspired in bacterial communication to make an intelligent selection of features. A successful example of the use of the algorithm is in the solution of the classic pattern classification problem. The Bacterial Foraging Optimization (BFO) algorithm shown in (Dang et al., 2008) is used for harder higher-dimensional and dynamic optimization problems in finance, specifically for parameter estimation of a EGARCH-M model (Exponential General Autoregressive Conditional Heteroskedastic-in-Mean, model used in econometrics to characterize and model observed time series), whose results can be used to predict the volatility of price of a financial instrument over time.

Other examples of such applications include: optimal filter design for image processing (Ji, Li, Lu, & Wu, 2008), optimal estimation of power system harmonics in a dynamic environment (Ji, Li, Wu, & Jiang, 2011) (a very similar application to the problem originally formulated in this research in 2009), and to solve Optimal Power Flow (OPF) problems in power systems (Li, Tang, Tang, Wu, & Saunders, 2007), where the population size is included as a variable of the algorithm.

Another area of application of these bio-inspired models focuses precisely on one of the

new communication technologies, the Wireless-Sensor Networks (WSN). In this regard, there are several research works in which it is sought to increase the robustness and performance of these sensing and communication systems, adding capabilities of scalability, self-organization, self-adaptation, and survivability (Atakan, Akan, & Tugcu, 2009; Foley et al., 2008; Gabrielli & Mancini, 2008; Liu & Jia, 2008; Jiang, Tseng, Hsu, & Lai, 2005; Wokoma, Shum, Sacks, & Marshall, 2005). This is a field of research that we believe it will have a strong impact on the future design of hardware, since due to the low costs of ad-hoc wireless communication, it is increasingly closer the idea of pervasive computing; environments saturated with wireless computing devices collectively providing services any time and everywhere.

However, one of the most important application areas is robotics. Robotics, intimately linked to the control area, has been fueled by numerous ideas from these biological models. Such is the case of the self-assembly properties (the autonomous construction of a device by itself) and therefore the self-repair, that while it still remains as an open problem, the search in this area has enabled the development of modular robotics and swarm robotics (Bhalla & Bentley, 2006; Nagpal & Mamei, 2004; Murata, Yoshida, Kurokawa, Tomita, & Kokaji, 2001).

There are many variations of electronic hardware whose designs have been made using the same principles of self-assembly. In (Boncheva, Gracias, Jacobs, & Whitesides, 2002) for example, they show the design of a scale model of the folding of a chain of polypeptide structural motifs into a globular protein. Although it is not a formal design technique, in this way, it tests basic concepts that in the future could enable the self-assembly of complex electronic structures. Another interesting example is the design of synthetic biocircuits, in what has been called bacterial computing, such as that presented in (Goni, Redondo, Arroyo, & Castellanos, 2011); here they are proposing a model in which every bacterium is considered to be a single logic gate and chemical cell-to-cell connections are engineered to control circuit function, in particular, they demonstrate the validity of approach with the design of a XOR circuit. Another similar research is (Karafyllidis, 2012) where they develop a quantum gate circuit model to reproduce experimental results that quantify the response of the *Vibrio harveyi* (is a species of Gram-negative, bioluminescent, marine, rod-shaped, motile bacteria in the genus Vibrio), and the nanomachines investigated in (Wiedermann, 2011b). In the initial stage of this research, the author explores these ideas of synthetic biocircuits and amorphous computer (Petru & Wiedermann, 2011; Wiedermann, 2011a) with the intention of developing a simplified behavior of QS (Martinez S. & Delgado, 2010), and these hardware experiments were vital for the determination of our mathematical model (Chapter 2).

Teamwork is another desirable feature in robotic applications. In these tasks, the goal is to develop small, simple robots which can be coordinated to independently develop a complex task together, for example, the transport of large objects in environments with obstacles (Fink, Hsieh, & Kumar, 2008; Moon, Jang, & Baek, 2008).

Regarding teamwork, the author has concentrated his efforts on applications involving autonomous navigation problems. In this kind of tasks the cooperative behavior among different agents or robots is critical to the successful path planning, in particular distributed control strategies. This kind of problems has been investigated previously by many other researchers (Parhi, Pothal, & Singh, 2009; Sahin, 2005), and remains as an open problem

in robotics.

We should clarify that not all possible applications of these models are restricted to hardware, many of these models have been used to develop software, again in the field of fault tolerance and, in particular, with parallel architectures as a strategy to achieve it (software robustness) (Gutierrez & Huhns, 2008).

1.3 Research problem

While research has evolved considerably since its initial formulation in 2009, the objectives were unchanged. The motivation was always finding a bio-inspired model that could be used in the design of electronic hardware. With the advancement of research, it was found that this model really should be focused on the design of complete systems, due to integration into the structure required for development tasks. This involved a bigger picture and general formulation, without neglecting the possible real applications.

1.3.1 General objective

Here the general objective of the research is transcribed as originally presented in the research proposal, without adjustments suggested by reviewers:

> Develop a strategy model for robust electronic hardware design, based on the inter-cellular communication system characterized in bacteria, and whose function is to control gene expression in relation to cell density (`quorum sensing`). This model will be implemented by hardware as embedded system for laboratory evaluation, solving a test problem, specifically in the recognition of harmonic content of a continuous signal. This implementation will be carried on best available hardware/software technology, according to analysis done in parallel with the development of research.
>
> Such a model has to contemplate the concept of three (3) different bio-inspired functional algorithms of cell development:
>
> - Reproduction. Scheme whereby artificial cells are reproduced in the environment. This is seen primarily as a process of cell division, where the growth boundary should be defined by the amount of resources needed to implement the functionality of the system.
> - Cell differentiation. Each of the artificial bacterial of the system must have an identical copy of the genome. However, depending on the population density, position in the system, etc. each bacterium interprets the genome in a very particular way, extracting only the genes of interest for configuration.
> - Populational organization. The artificial system will be structured in a finite number of cells or artificial bacteria, where each performs a unique function determined by its genetic code (genome).

This general objective was modified by the academic peers during the evaluation of the research proposal. As it was argued, the harmonics identification application is a solved problem, so it was suggested to evaluate the algorithm based on QS in different problems. The problem selected was the path planning for robotics.

Details on compliance of the objective

This objective is documented throughout the document. The model for robust electronic hardware design, based on the inter-cellular communication system characterized in bacteria, is postulated in Chapter 2. The details regarding its approach based on QS are introduced in Chapter 1. We conducted several hardware implementations in embedded systems, some on small boards with CPLD (Chapter 2), and others on autonomous robots (Chapter 3). In each case the hardware selected was the most suitable in terms of availability, cost and performance. Behaviors defined in the model replicate the characteristics of cell development proposed: Reproduction, Cell differentiation and Populational organization.

1.3.2 Specific objectives

Again, the specific objectives are transcribed as originally presented in the research proposal, but this time they are organized in a table specifically to show the fulfillment of each of them (Table 1.1).

Except for the sixth objective that we modified according to the new application in robotics, all objectives were met. This is reflected in this thesis. For example, Chapter 1 reflects the development of objectives 1 and 3, Chapter 2 is directly related to objectives 1, 2, 4 and 5, Chapter 3 is related to objectives 3, 6 (with application in robotics) and 7, and the objective 8 throughout all the document.

1.4 Improvements suggested by the jury

The jury made a series of comments and observations to the initial draft of the dissertation. These suggested improvements are classified into three categories to facilitate their analysis: Background, Model and Task assessment. The organization is shown in Fig. 1.3.

In this dissertation the author has responded to each and every one of these recommendations. For example, in what has to do with background, as to the quality of the grouping distribution (dispersion of the agents in the group and radius of the group), on page 29 the text talks about the use of indexes to establish compatibility and separation of the group. Regarding the bacterium that inspired the algorithm, on page 3 explicitly specifies that features were adopted to the algorithm, and which bacteria (not of a single bacterium). As for the question that was presented on the evolution of hardware referred in the research, on page 4 it clarifies that the algorithm does not consider evolution as understood in biology, but seeks adaptability.

Concerning the observations of the model, its description and the detail of the behaviors involved were improved. Regarding the possibility of a heterogeneous system, on page 92

Table 1.1: Details on compliance of the objectives.

Specific objective	Details on compliance of the objective
Perform a detailed justification QS as an biological alternative approach to emulate, in the attempt to develop hardware architectures for electronic applications with inherent redundancy properties, and therefore a high level of reliability.	The reliability and robustness that research seeks to guarantee in electronic hardware is supported by redundancy (section 1.1). Therefore biological models that were considered are exclusively multi-agent systems with inherent redundancy (Introduction). The research considers in its beginnings to the ontogenetic systems in general, since the cell development processes have redundancy features. By studying bacteria within such systems (section 1.2), research identified the QS as a biological behavior capable of accelerating convergence processes, which is why the work focused on them (section 2.1).
To analyze the mechanism of intercellular communication in bacteria QS. Perform its functional characterization, and propose an operational scheme feasible for application to the design of electronic systems.	The mechanism of intercellular communication in bacteria is supported in the chemical information that bacteria take and/or place in the environment (section 1.2). That is why the proposed model is supported only on local communication schemes. Its application focuses on locally observable environments. Chapter 1 focuses on emulate this dynamic in hardware. For this characterizes a functional model of the bacterium that allows to define behavior programmable rules. This chapter characterizes the bacterium, their environment, their interaction rules, and functionality on an embedded system is demonstrated.
To evaluate electronic technologies available for implementation of the algorithms, topologies including (but not limited): FPGAs, CPLDs, multi-core processors, GPUs, and hybrid architectures and devices with Intel's Larrabee and AMD's Fusion. As a result of this evaluation, the best technology will be selected for the final design.	Regarding electronic technologies evaluated, the research selected the most suitable platforms according to availability, cost and performance. We use a XC2C256 CPLD group (section 2.1), wherein each CPLD contains the complete hardware of the agent. This platform allows to evaluate basic communication schemes and the feasibility of the algorithm, but restricts important features like motion. A second platform evaluated, which became the perfect agent for assessment (due to low processing requirements of the algorithm) is the differential SERB robot, controlled by microcontroller (ATmega328P, sections 2.4, 3.1, 3.3 and 3.4). Other platforms considered include the use of an FPGA on which individual cores (agents) are programmed, but this is not documented since corresponds to an evolution of the CPLD that moves away from applications in robotics.
To develop basic functional models on electronics technologies that support parallel processing of different cellular development algorithms projected from the QS model.	The functional models of intercellular communication, cell reproduction and virulence activated by QS were programmed and evaluated on electronic platforms based on CPLDs (section 2.1) and microcontrollers (section 2.4).
To formulate a model of robust electronic hardware design based on QS.	The model of robust electronic hardware design based on QS is formulated in Chapter 2. The model consists of a description of the multi-agent system according to population size (section 2.3), and a description of the agent for this system (section 2.4).
To analyze and delimit a specific application for the system based on the QS model, application related to a continuous signal processing, specifically with the recognition of its harmonic content. Such analysis involves a study of the state of the art and theory of harmonic distortion.	The specific application for QS-based model is in robot motion planning, and is fully described in Chapter 3. This chapter raises some possible environment configurations, which allow the movement planning. In the first case radiated signals by agents (section 3.3) is proposed, and in a second case, a more general strategy based on the design of gradients on the environment (section 3.4) is formulated. In the chapter ergodicity is also raised (section 3.1), as a dynamic feature necessary to ensure full coverage of the environment (search space), the use of the algorithm is detailed (section 3.2), and a comparison of performance is shown. (section 3.5).
To validate the design model both algorithms cell development level, and as a functional level of the system, by means of simulation and hardware prototype.	The model validation is performed at the cellular level for intercellular communication, reproduction and virulence QS activated processes. By software we perform in Matlab simulation of differential equations of the model to observe the population behavior (sections 2.3, 3.4 and 3.5), and in Player/Stage tool we observe the interaction of robots in real time (sections 2.4, 3.3 and 3.4). On hardware was conducted in CPLDs (section 2.1) and microcontrollers (section 2.4).
To develop the research documentation both the final document and published papers.	This is the final document that shows all the research. Throughout the development of the research, 11 articles were published, nine of them in journals category A1 and A2 (according to Colciencias). Also the author presented seven oral papers in high-impact conferences, five of them in recognized IEEE conferences. Section 1.5 shows the detail of these publications.

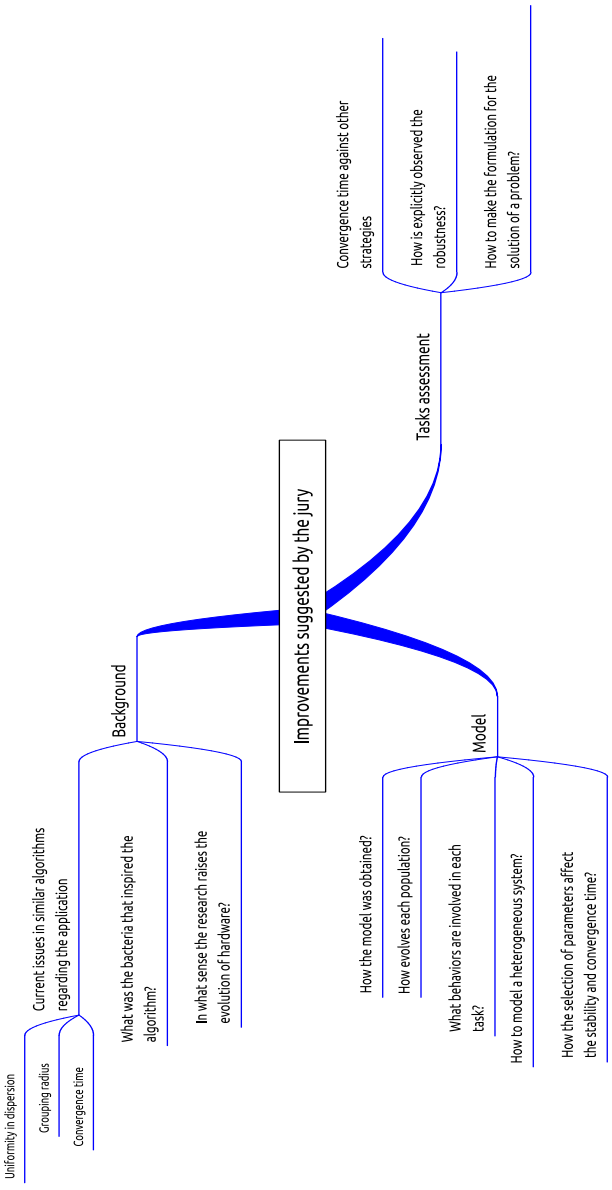


Figure 1.3: Organization of improvements suggested by the jury

two possible options (joint work of all agents or segregation of a group) are discussed, and the general characteristics of the system in each case is shown. Concerning to stability, convergence and parameters selection, on page 23 the author includes a section with a wide discussion about it.

Finally, with regard to the observations of task assessment, the broader discussion that is added to the document is regarding the convergence time comparison against other equivalent strategies. On page 97 the author includes a section in which a statistical test to the average times of six grouping strategies, half of them with QS, is applied. The test results demonstrate that the inclusion of QS in the original algorithm reduces the convergence times. On page 97 results per simulation for robustness tests are shown, and on page 65 the author includes a section showing the design process step by step.

1.5 Publications

The research results have been published in journal articles and international conferences. Tables 1.2 and 1.3 show the details of such publications. These publications are classified according to their approach. The first group corresponds to Theoretical foundation, and extend the content of the first two sections in Chapter 2. The second group is called Design, and corresponds to the application of the models described in Chapter 2 (last two sections). Finally, the third group corresponds to Application, reporting applications raised in Chapter 3.

1.6 Organization

This dissertation is organized as follows. In Chapter 2, the author explains in detail the design model based on QS. First, the thesis explains the simplified interaction between bacteria, and then shows a possible behavior hardware emulation. Then, the formal models are developed, one at system level and another at the level of bacterium. In Chapter 3, the experimental results for these models and systems are presented and analyzed, specifically in robots navigation applications. Finally, in Chapter 4, the author draws conclusions from these experimentations and discusses future work that could be done to further advance research.

Table 1.2: Publications regarding Theoretical foundation and Design.

Theoretical foundation

Reference	Kind	Character	Colciencias Category
Proyección de Diseño de Hardware Electrónico Robusto Basado en Quorum Sensing (2009), In: Sistemas, Cibernética e Informática Vol. 6 Num. 2, pp. 7-11, ISSN: 1690-8627	Journal	International	
Hardware Emulation of Bacterial Quorum Sensing (2010), In: Lecture Notes In Computer Science LNCS 6215, Springer, pp. 329-336, ISSN 0302-9743	Proceedings as special issue of journal	International	C (2010)
Seres Vivos: Fuente de Inspiración para el Diseño Artificial (2013), In: Tecnura Vol 17 No. 37, pp. 121-137, ISSN 0123-921X	Journal	National	A2 (2013)
A comparative study of geometric path planning methods for a mobile robot: Potential field and Voronoi diagrams (2013), In: II Congreso Internacional Ingeniería Mecatrónica y Automatización CIIMA 2013. ISBN 978-1-4799-2470-7	Conference	International (IEEE)	

Design

Reference	Kind	Character	Colciencias Category
Wild Robots, Ergodicity and Virtual Gates (2011), In: 2011 Symposium on Emerging Topics in Control and Modeling: Cyber-Physical Systems. USA.	Conference	National (USA)	
Wireless Visual Sensor Network Robots-Based for the Emulation of Collective Behavior (2012), In: Tecnura No. 31, pp. 10-18, ISSN 0123-921X	Journal	National	A2 (2012)
Controlling Wild Mobile Robots Using Virtual Gates and Discrete Transitions (2012), In: 2012 IEEE American Control Conference (ACC 2012)	Conference	International (IEEE)	
Image Processing System Based on Android Embedded Operating System (2013), In: Congreso Argentino de Sistemas Embebidos CASE 2013, ISBN 978-1-4799-1101-1	Conference	International (IEEE)	
Collective multi-agent navigation model based on bacterial quorum sensing (2016), In Tecnura Vol. 20 No. 47, pp. 29-38, ISSN 0123-921X	Journal	National	A2 (2016)
Study of collective robotic tasks based on the behavioral model of the agent (2015), In: Lecture Notes In Computer Science LNCS 9375, Springer, pp. 224-231, ISSN 0302-9743	Proceedings as special issue of journal	International	A2 (2015)

Table 1.3: Publications regarding Application.

Application

Reference	Kind	Character	Colciencias Category
Brownian motion as exploration strategy for autonomous swarm robots (2012), In: IEEE International Conference on Robotics and Biomimetics (ROBIO 2012), ISBN 978-1-4673-2125-9	Conference	International IEEE	
Particle Diffusion Model Applied to the Swarm Robots Navigation (2012), In: Tecnura Vol. 16 Edición Especial, pp. 34-43, ISSN 0123-921X	Journal	National	A2 (2012)
Geometric Hybrid Path Planning from the Artificial Potential Field Method (2013), In: Jornadas de Automática JA13. ISBN 978-953-307-098-5	Conference	International (IFAC)	
Autonomous Navigation Strategy for Swarm Robots using Local Communication (2014), In: Tecnura Vol. 18 No. 39, pp. 12-21, ISSN 0123-921X	Journal	National	A2 (2014)
Using the Delaunay Triangulation and Voronoi Diagrams for Navigation in Observable Environments (2014), In: Tecnura Vol. 18 Edición Especial, pp. 81-87, ISSN 0123-921X	Journal	National	A2 (2014)
Generación de Ruta Óptima para Robots Móviles a Partir de Segmentación de Imágenes (2015), In: Información Tecnológica, Centro de Información Tecnológica (CIT) Vol. 26 No. 2, pp. 145-152, ISSN 0718-0764	Journal	International	A1 (2015)
Performance evaluation of a geometric strategy for autonomous navigation in observable environment (2015), In: 2015 Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies CHILECON, ISBN 978-1-4673-8755-2	Conference	International (IEEE)	
Path planning for mobile robots based on visibility graphs and A* algorithm (2015), In: Proceedings of SPIE, The International Society for Optical Engineering Vol. 9631, ISSN 0277-786X	Proceedings as special issue of journal	International	A2 (2015)

CHAPTER 2

Multi-agent dynamic system

The multi-agent model adopted for artificial systems design is composed of a set of artificial bacteria or agents. This set of agents, and their interactions, reflect the dynamics that will solve the problems. All agents are identical in design. Nevertheless, to solve tasks, each of them undergoes certain behavior inside the system along the time.

The variables of this system are defined in a continuous space. However, the different agent behaviors are triggered by certain events (event-based), which implies that the appropriate analysis model is a hybrid system (the dynamic changes from one state to another when a condition is present).

In this chapter, we want to characterize the behavior of bacterial QS using three different approaches. First the interaction among bacteria is analyzed according to a model of behavior based on antibody-antigen recognition. The model allows to describe by rules the behaviors of reproduction and cell differentiation, which coordinate the self-organization of the system. Second, a macro-scale model is presented, which averages the performance of multiple agents. Using the model, it is possible to design the application as an average behavior. And third, a micro-scale model is presented to characterize each system agent. This model allows to describe the dynamics of the agents, and may even include their physical characteristics. This structure allows the use equations to describe the system state variables, and from them, following a scheme of top-down design, to define the behavior of individual agents. This is the design scheme that is used to develop the applications that are presented in the next chapter.

2.1 Bacterial Quorum Sensing

The proposed model of bacterial interaction involves two basic behaviors: reproduction and cell differentiation. This first section provides an approach to these two processes to characterize them and observe in detail their simplified operation. For that, a mathematical model of artificial bacterium and its behaviors reflecting the self-organization of the system is developed. And then, with the intention to closely observe this interaction, a digital architecture for the emulation of dynamics of bacterial QS is carried out.

The author analyzes the interaction among bacteria according to a model of behavior based on antibody-antigen recognition, and uses this model to emulate the dynamics in an artificial bacterium. The architecture is completely scalable, with all parameters stored in the artificial bacteria. It shows that self-organizing principles can be emulated in an electronic circuit, to build a parallel system capable of processing information, similar to cell-based structure of biological creatures.

The bacterium is implemented on an 8-bit microcontroller; and a framework with CPLDs to build the hardware platform where the bacterial population increases is developed. At the end of the section, simulation results show the ability of the system to keep working after physical damage, just as its biological counterpart. We have published part of this discussion in (Martinez S. & Delgado, 2010).

2.1.1 Background

The theory of self-organization is based on the assumption that the system functionality is not a result of the individual performance of its elements; but rather a result of the interaction among the elements.

Studying the characteristics of self-organizing dynamic systems (Prokopenko, 2008), it is possible to identify three key aspects of a self-organized system (Polani, 2003): (1) The existence in the system of a lot of units (agents) (Russell & Norvig, 2002) that interact among themselves, so that the system evolves from a less organized state to a more organized state dynamically, along time, and maintains some kind of exchange; (2) The organization is evident in the global system behavior as a result of the interaction of agents (Camazine et al., 2001), its functioning is not imposed on the system by any kind of external influence; and (3) the agents, all of them with identical characteristics, have only local information (Santini & Tyrrell, 2009; Polani, 2003), which implies that the process of self-organization involves some transfer of local information. In summary, a self-organizing system is characterized by:

- Many simple and identical agents.
- The continuous interaction of these agents.
- The use of local information by the agents.

Self-organization has been a subject of great theoretical research, but its practical applications have been somewhat neglected (Prokopenko, 2008). For example, Mange et al. in (Mange et al., 1996) and in (Freitas & Gilbreath, 1980) have worked long applications of FPGAs in bio-inspired hardware, the most important process that they involve is the cellular construction, therefore, their artificial hardware implements cellular processes of replication and regeneration.

This research aims at providing a new approach into the application of self-organization principles to one aspect of electronic systems, digital processing. Our approach has been to devise reconfigurable logic hardware and an architecture that permits self-organizing processes; and then to begin methodically in order to develop self-organization concepts and their translation into practice within this framework.

Specifically, in this first part the hardware emulation of bacterial QS is presented. A system composed of a set of simple processing units, all identical in structure and programming, which collectively self-organize and perform a complex task, with the broader aim of addressing engineering applications.

2.1.2 System structure

Approaches to self-organization can be grouped into at least two main categories: a statistical focus and an engineered focus (Prokopenko, 2008). Statistical approaches seek to manage complexity by discovering algorithms or techniques. Such ideas are not necessarily concerned with how the given task is accomplished, but how well it is actually accomplished. Examples of statistical approaches include phylogenetic[1] (genetic algorithms and systems based on theories of evolution (Greenwood & Tyrrell, 2006; Koza, Keane, & Streeter, 2003)), epigenetic[2] (neural networks and models of immune and endocrine systems (Arbib, 2003; Zeng & Li, 2009)) and ontogenetic[3] models (cell development, ontogenesis (Sipper, Mange, & Stauer, 1997)).

In contrast to statistical techniques, engineered approaches seek to achieve, more deliberately, some set of goals by following a predefined algorithm. Surprisingly, several engineered ideas also draw inspiration from biology, including the electronic embryology (embryonics) (Ortega-Sanchez, Mange, Smith, & Tyrrell, 2000; Prodan, Tempesti, Mange, & Stauffer, 2003). The discussion in this research falls into the domain of the engineered approach, taking as a design idea ontogenetic principles of living beings.

The simplified model of bacterial QS describes a set of independent cells, which under the generation of extra-cellular signals produce coordinated social behaviors (Otero, Munoz, Bernández, & Fábregas, 2004). The pathogenic bacteria that carry multicellular organisms are not virulent until they reach a sufficient majority to enforce a massive attack against the immune system. When bacteria determine that their population size is enough to trigger an attack, they transform and become virulent.

For our evaluation of behavior, we developed a specific reconfigurable platform called Bacterial Platform (BP). The BP, inspired by the Cell Matrix architecture presented in (Macias & Durbeck, 2005), is a regularly tiled collection of simple processing units called Bacterium. This platform is mainly an internally configured device, the configuration of each bacterium is written and read by the bacterium connected to it, i.e., its immediate adjacent neighbors, without the need for an external system (e.g., a PC). Only bacteria located on the perimeter of the platform can be accessible from other systems (communication with the platform). As CA, the BP is a dynamic system that evolves in discrete steps according to local interaction.

In this first prototype, bacteria are arranged in a fixed, identical topology throughout

[1] Is the study of evolutionary relationships among groups of organisms (e.g. species, populations). It is related to the adaptation and evolution of living populations due to genetic variation along generations.

[2] Is the study of heritable changes in gene activity which are not caused by changes in the DNA sequence. It relates to the adaptation of individual organisms by developing their internal systems along their life.

[3] Is the study of origin and development of an individual organism from embryo to adult. It relates to the ability of the cells to reproduce and specialize in supporting the development and operation of the individual.

the matrix, in a two-dimensional plane (Fig. 2.1). Each bacterium receives a dedicated input (called chemical signal input, CSI) from each of its neighbors, and generates a dedicated output (chemical signal output, CSO) to each of those neighbors. In this topology, each bacterium has four immediate neighbors, and there is no physical displacement.

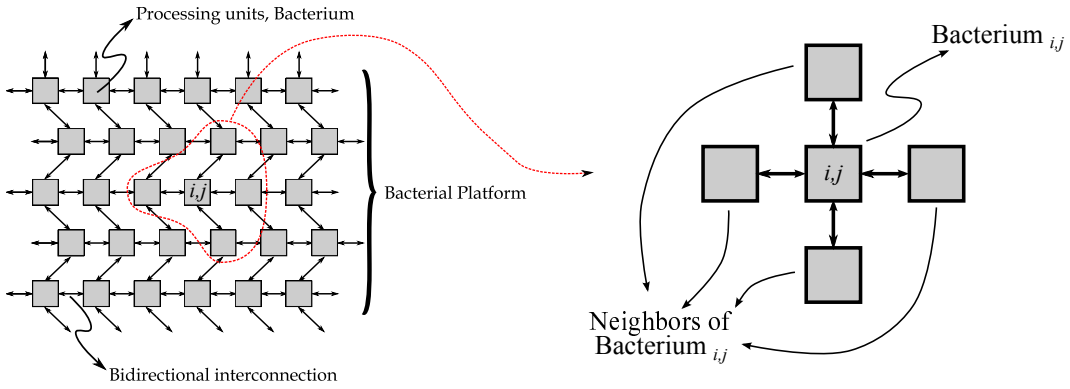


Figure 2.1: Details of the two-dimensional structure of the BP.

Each bacterium contains a small 8-bit microcontroller (Fig. 2.2) that stores the genetic code of the bacterium, and reacts according to environmental conditions (communication with its neighbors). This interaction allows the execution of two basic functions:

- Reproduction: Scheme by which an artificial bacterium reproduces in the system. It is basically a process of cell division, where the growth limit is defined by the amount of resources needed to implement the operation of the system.
- Cell differentiation: Each artificial bacterium in the system has an identical copy of the genome[4]. However, depending on population density and its physical position, each bacterium interprets it in a particular way.

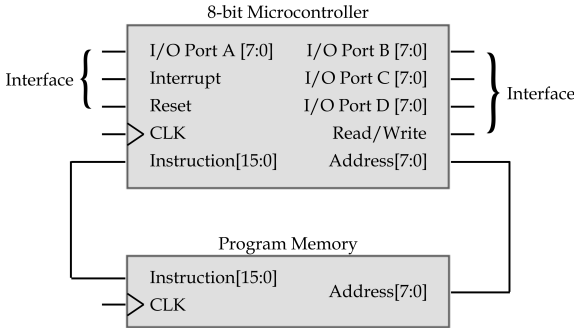


Figure 2.2: Microcontroller block diagram.

Loading the genome (program the microcontroller with the information about its functionality) from one bacterium to another, is called reproduction. Similarly, loading the

[4] In molecular biology and genetics, the genome is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of viruses, in RNA. For the artificial bacterium, the genome is the program code written in the microcontroller memory. This code contains the rules of behavior according to environment readings.

genome in several bacteria is called population growth. The system architecture is infinitely scalable; there is no architectural impediment to scale the system to any desired size.

2.1.3 QS behaviors and artificial bacteria

Definition 1. A bacterium is a pair

$$V = (f, p) \tag{2.1}$$

where f is a nonnegative integer ($f \in \mathbb{Z}^+$) that indicates the amount of neighboring bacteria in direct contact, and p is a point in q-dimensional space ($p \in \mathbb{R}^q$). For the particular case of the hardware prototype developed in this part of the research (Fig. 2.1), f takes values 0, 1, 2, 3 and 4 for each bacteria over time (the value changes when reproducing the bacterial population), and p is a point in two-dimensional plane $\big(q = 2, \Rightarrow p \in \mathbb{R}^2 \Rightarrow p = (x_1, x_2)\big)$.

The bacterial recognition occurs in a bacterium V_i when the bacterium defines its values f and p. This definition corresponds to an extension of the cell definition in the antibody-antigen mathematical model (Tarakanov & Dasgupata, 2000).

Definition 2. The population density is evaluated using the distance between bacteria. Let:

$$d_{ij} = d\,(V_i, V_j) \tag{2.2}$$

as the distance between bacteria V_i and V_j, which is calculated by any appropriate norm.

Definition 3. A bacterial population is defined as a nonempty set of bacteria.

$$W_0 = \{V_1, V_2, V_3, \cdots, V_m\} \tag{2.3}$$

with non-zero distance among bacteria:

$$d_{ij} \neq 0, \qquad \forall i, j \quad , \quad i \neq j \tag{2.4}$$

The bacteria that implement the final task are called here Application Bacteria, AB, and are a subset of the bacterial population (Fig. 2.3).

$$W \subseteq W_0 \tag{2.5}$$

That is, W_0 is the set of all bacteria in the system, and W is the set of virulent bacteria in the system.

Definition 4. The neighborhood threshold η, indicates the maximum amount of direct neighboring bacteria that a cell can have. For the prototype hardware (Fig. 2.1), $\eta = 4$ for all bacteria.

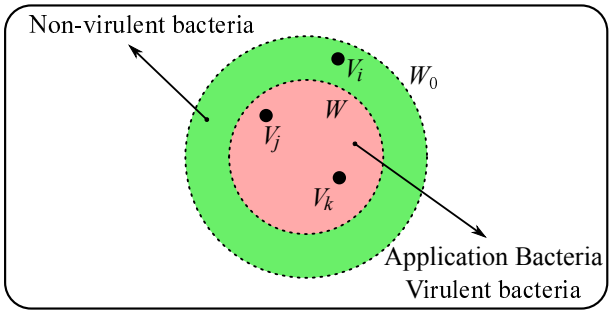


Figure 2.3: Bacterial population and application bacteria in the proposed system.

Throughout reproduction, the population size (number of agents) is always compared to T, the quorum threshold, it is the parameter defining whether or not it has reached the quorum. Regarding the physical dimensions of the bacterial population, the density threshold h indicates the minimum distance needed between bacteria; in other words, the minimum distance that must exist between any pair of bacteria, for example bacteria at opposite ends of the platform.

The behaviors of bacteria (self-organization) are coordinated by the following rules (the model does not include cell death; if there is cell death, it is caused by an external action):

Reproduction Rule: If the bacterium $V_i \in W_0 \setminus W$ can reproduce, i.e.:

$$f_i < \eta \quad \text{and} \quad d_{ij} < h, \qquad \forall V_j \in W \tag{2.6}$$

then V_i must reproduce by duplicating their DNA (code) in the available medium.

In other words, if the bacterial population is very small and the non-virulent bacterium V_i can reproduce, then V_i must reproduce (must copy its code to a neighboring inactive).

Virulence, Activation or Cell Differentiation Rule: If the bacterium $V_k \in W$ is the nearest to the bacterium $V_i \in W_0 \setminus W$ among all AB, i.e. (Fig. 2.3):

$$d_{ik} < d_{ij}, \qquad \forall V_j \in W,\ k \neq j \quad \text{and} \quad d_{kj} < h, \qquad \forall V_j \in W,\ k \neq j \tag{2.7}$$

and the number of bacteria in W is less than T, then V_i must be added to AB (the bacterium becomes virulent).

In other words, if the AB is very small and the non-virulent bacterium V_i is the nearest to a virulent bacterium, then V_i must become virulent (must solve the task with the other bacteria of AB).

2.1.4 Emulation and results

The system platform was constructed by assembling 12 identical blocks (Fig. 2.4), each block with total three Xilinx CoolRunner^{TM}-II CPLD (XC2C256, 256 macrocells, 144-pin TQFP). Each CPLD has the ability to communicate with its four neighbors (Fig. 2.1).

Each CPLD is programmed with a PicoBlaze soft processor of 8-bit RISC architecture. Initially, these processors do not have program code (they really are programmed, but inactive, as inert bodies), and therefore they are not considered part of the system. When the artificial bacterium program is loaded within the CPLD processor (with an external activation signal), it is said that the environment is contaminated and has a new bacterium, which can be virulent or not depending on bacterial density.

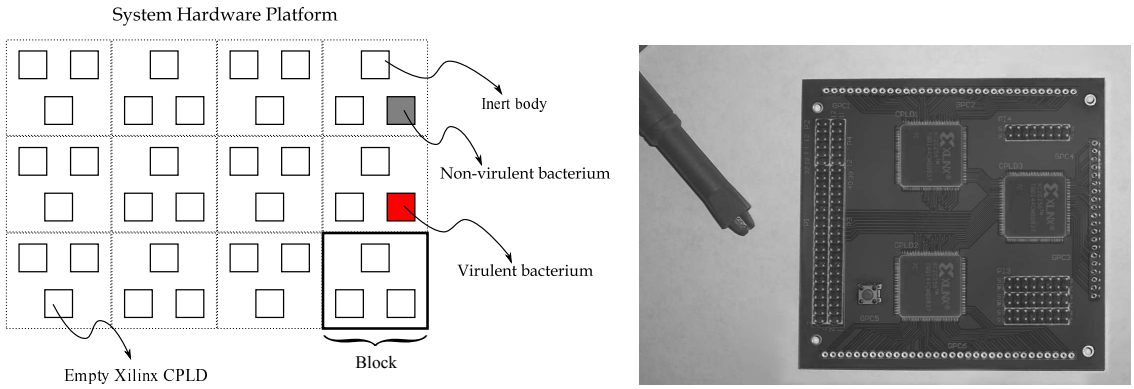


Figure 2.4: Hardware platform for microcontroller evaluation (36 Xilinx CPLDs).

(a) System hardware platform, (b) A block with three processors.

In order to analyze the behavior of bacterial algorithm on the platform, an emulation of bacterial growth based on the rules of reproduction and cell differentiation was done. An array that can change the status of each unit in each bacterial operation cycle (one complete cycle of the bacterium program) was designed.

Fig. 2.5 shows the growth of the bacterial population after an initial infection. The initial state ($t = 0$) is observed in Fig. 2.5(a) (all CPLDs are empty, are inert bodies). Fig. 2.5(b) shows the initial infection ($t = t_1$) caused by an external signal (bottom left corner). During cycles $t = t_2$, $t = t_3$, $t = t_4$ and $t = t_5$ is possible to observe the process of reproduction (Fig. 2.5(c) to Fig. 2.5(f)). Fig. 2.5(e) shows the appearance of the first virulent bacterium ($t = t_4$). In this emulation, a density threshold of five ($h = 5$) is used, therefore, the population growth ends in $t = t_5$ (Fig. 2.5(f)). However, the AB is still small, so the dynamics continues over several cycles. In subsequent cycles, bacteria close to AB become virulent until it meets the criterion of size in $t = t_7$ (Fig. 2.5(h)).

The QS in this emulation can be seen from cycle $t = t_4$. According to the activation rule, if the AB is small (the emulation used a size of $h = 5$) and the bacterium is surrounded by neighboring bacteria, whether virulent or in reproduction, the concentration of agents at this point will cause the bacterium become active (become virulent). The process is triggered, and ends with an AB size suitable for solving the task.

As noted, it requires a set of five agents to force the activation of a bacterium, unless the agents are present at the ends of the platform; the emulation cycle $t = t_4$ required only four bacteria (the non-virulent agent and three neighbors).

According to the rules of reproduction and activation, the behavior encoded in the genome of the artificial bacterium (algorithm implemented in each microcontroller) has the logical structure shown in Fig. 2.6.

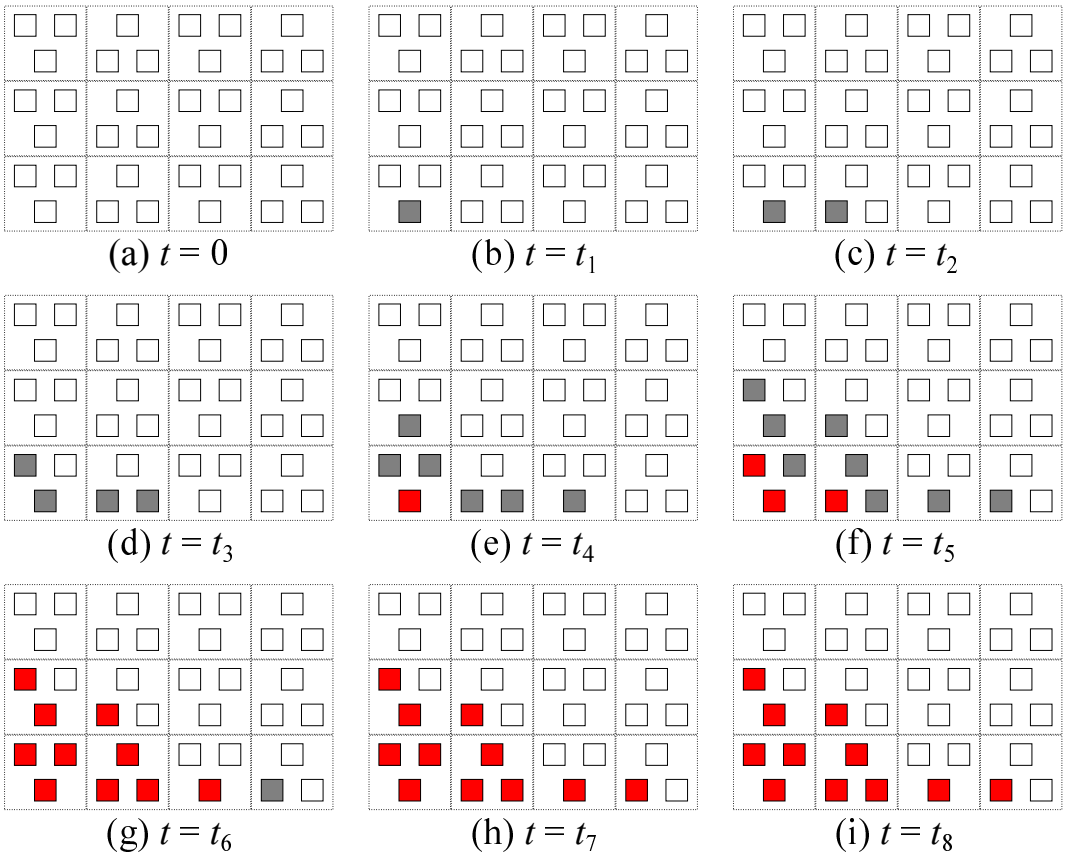


Figure 2.5: Bacterial growth on the platform.

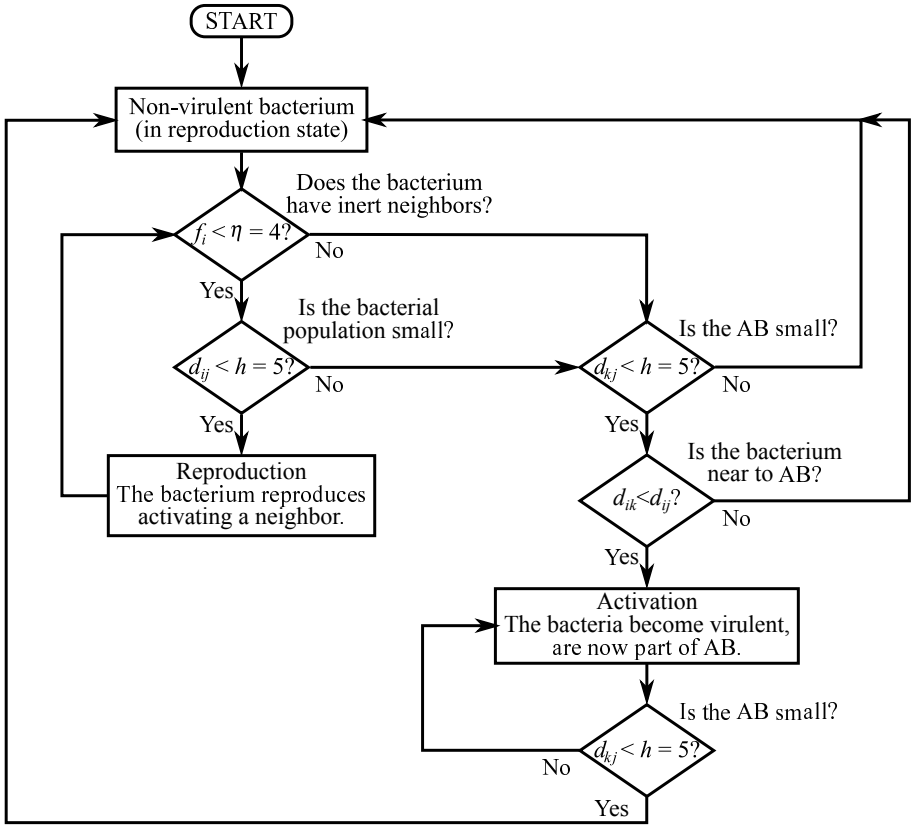


Figure 2.6: Behavior encoded in the genome of the artificial bacterium.

One of the important features of behavior included in the algorithm, corresponds to the ability of a virulent bacterium to return to its non-virulent state if damage occurs in the AB reducing the size of the community (final feedback on the left side of Fig. 2.6). In case of failure that disables some of the bacteria of AB, the system does not have the size needed to perform the task, and some of the agents should reproduce again to replace dead bacteria.

In order to evaluate the robustness and reliability of the system, one of the blocks is eliminated (lower right block) simulating the sudden death of three bacteria, and the response of the system is observed. Fig. 2.7 shows a new behavior where bacterial growth begins again, and again it meets the threshold density after two bacterium cycles (Fig. 2.7(c), $t = t_{11}$)). Now, although the bacterial population is large enough, it is not the case with AB, so the dynamics continues to reach again an equilibrium point in $t = t_{13}$ (Fig. 2.7(e)).

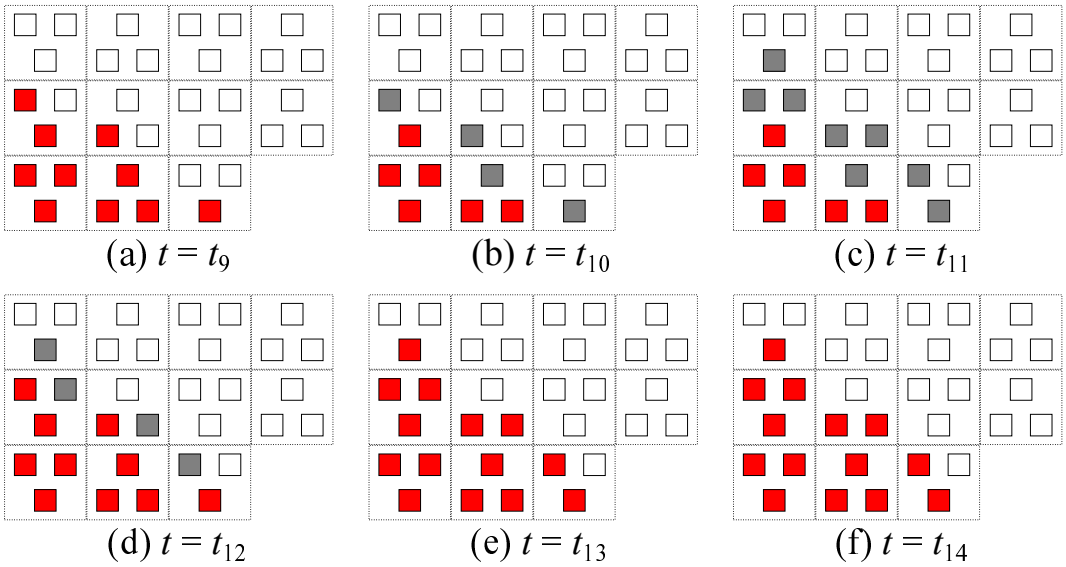


Figure 2.7: Adaptation of the system after physical damage.

This emulation demonstrates that with two simple behavioral rules the system can self-organize, and define its structure and size without external control. Once the system reaches an equilibrium point, the AB made up of active bacteria is ready to perform the task. In the following sections, models for the behavior of AB during the development of the task are developed (a bacterial group level model and an individual-level model).

2.1.5 Stability, convergence and parameters selection

The stability and convergence of the model is dependent on the parameters selected for the design of the navigation environment. Because the schema is supported in local readings, its use in real applications requires modification (design) of the navigation environment. The parameters selection is essential for this design and modification of the environment.

The design of the navigation environment consists of defining the areas of E to be the final destination for the agents, and then place in them and in their neighborhoods, special marks (landmarks) that indicate this condition. Generally these landmarks are not equal,

each value indicates the intensity corresponding to its position in E. The intensity value is designed for the entire navigation environment, so it grows near the final destination area.

Thus, during the search process (navigation), agents interpret local readings and determine the intensity value at each point of the environment. Climbing the slope of these values, each agent follows a heuristic that allows it to choose the optimal choice in each advance as does a greedy algorithm. Although such an approach can be inappropriate for some computational tasks (the strategy stagnates in local solutions without finding the global optimal solution), for the proposed algorithm based on QS this constitutes an exploration strategy that ultimately ensures convergence in the global optimal solution.

The effect of QS on the population is to accelerate the convergence in the areas of high performance. For this, the model takes into account the population size in the area, increasing the attraction when a threshold is exceeded. The value of the slope is designed as a function of the density threshold.

The following sections will show how to model the navigation field intensity as a gradient on the environment. This section focuses only in the vicinity of an area of high performance (area to be interpreted by the agents as of interest), and how to design a simple intensity field according to a slope, considering the issues of convergence.

The gradient in the area can be modeled by a system of two autonomous first-order ODEs (Ordinary Differential Equations), model in which the slope at each point represents the intensity of the gradient. To plot the slope field, the two autonomous ODEs must be written in the form (Eq. 2.8):

$$\begin{array}{l} \frac{dx}{dt} = \boldsymbol{G}(x, y) \\ \frac{dy}{dt} = \boldsymbol{F}(x, y) \end{array} \tag{2.8}$$

That is, the first expression is the derivative of the variable represented on the horizontal axis of the environment, and the second expression is the derivative of the variable represented on the vertical axis of the environment. The origin corresponds to the center of the area of high performance.

The two functions must confine the agents at the origin. Since the slope of the two curves are who push the agents, it is possible to think of a curve design for x and y as shown in Fig. 2.8.

These two curves are designed in such a way that with their growth there is no possibility of way between them. Curve y (green) is a reflection of the curve x (blue), and the slopes push towards the origin. To facilitate the calculation of derivatives, these curves can be replaced by exponential approximations (Eq. 2.9):

$$x = a \times e^{\frac{-y}{\tau}} - a, \text{ and } y = a - a \times e^{\frac{-y}{\tau}} \tag{2.9}$$

In these two curves is easy to define the behavior: a is the reference value and τ is the time constant, which directly affects the slope of the curve, and can be defined as a function of the density threshold.

To represent graphically the slope field is necessary to calculate the derivatives of the curves (Eq. 2.10):

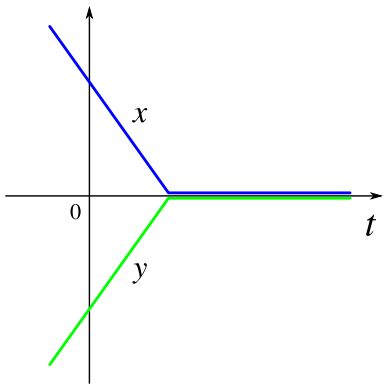


Figure 2.8: Linear approximation of intensity gradient behavior in an area of high performance.

$$\frac{dx}{dt} = -\frac{a}{\tau \times e^{\frac{y}{\tau}}}, \text{ and } \frac{dy}{dt} = \frac{a}{\tau \times e^{\frac{y}{\tau}}} \tag{2.10}$$

Since these curves are exponential, we subtract to each one the value of the other variable to prevent grow up indefinitely (y to x and x to y). The Fig. 2.9 shows the slope field for a time constant of: $5\tau = 5 \Rightarrow \tau = 1$.

The Fig. 2.10 shows the x and y curves for the resulting motion curve shown in Fig. 2.9 (red curve).

From the model and these behaviors is possible to conclude some general characteristics in terms of stability and convergence.

Lemma 1: Let $\boldsymbol{p} = \{p_1, p_2, \cdots, p_m\}$ be a set of points drawn randomly in E on a two-dimensional Euclidean plane $\mathbb{R}^2$, each with an associated slope value m according to some law of navigation. Let $\boldsymbol{G}(x, y)$ be a function on $\mathbb{R}^2$ that assigns a slope value to each point $p \in \boldsymbol{p}$, such that the maximum value is assigned to the central point of the area of high performance. Then there exists a point p_0 such that $\boldsymbol{G}(p_0) = m_0 \geq \boldsymbol{G}(p)\ \forall p \in \boldsymbol{p}$, where p_0 is the closest point to the center of the area of high performance.

Proof: By construction the maximum value of m is assigned to the central point of the area of high performance. If this construction is correct, p_0 always exists. p_0 is the $p \in \boldsymbol{p}$ closest point to the area of high performance. Furthermore, since the heuristic climbs always looking for the right next value, the agent eventually will find this point p_0 if there are no other areas of high performance. ■

Lemma 2: If there is at least one area solution in E, incorporation of QS in finding solutions does not produce stability problems in the search algorithm. By contrast, this reduces the convergence time.

Proof: The proposed algorithm based on QS alters the behavior of the function $\boldsymbol{G}(x, y)$ at points near to the high performance area. This alteration is conditioned to the density threshold, producing localized alterations if the population in the area exceeds that threshold. QS increases the weight of the solution, and therefore reducing the total search time. However, QS does not change the global behavior of $\boldsymbol{G}(x, y)$, therefore does not affect the

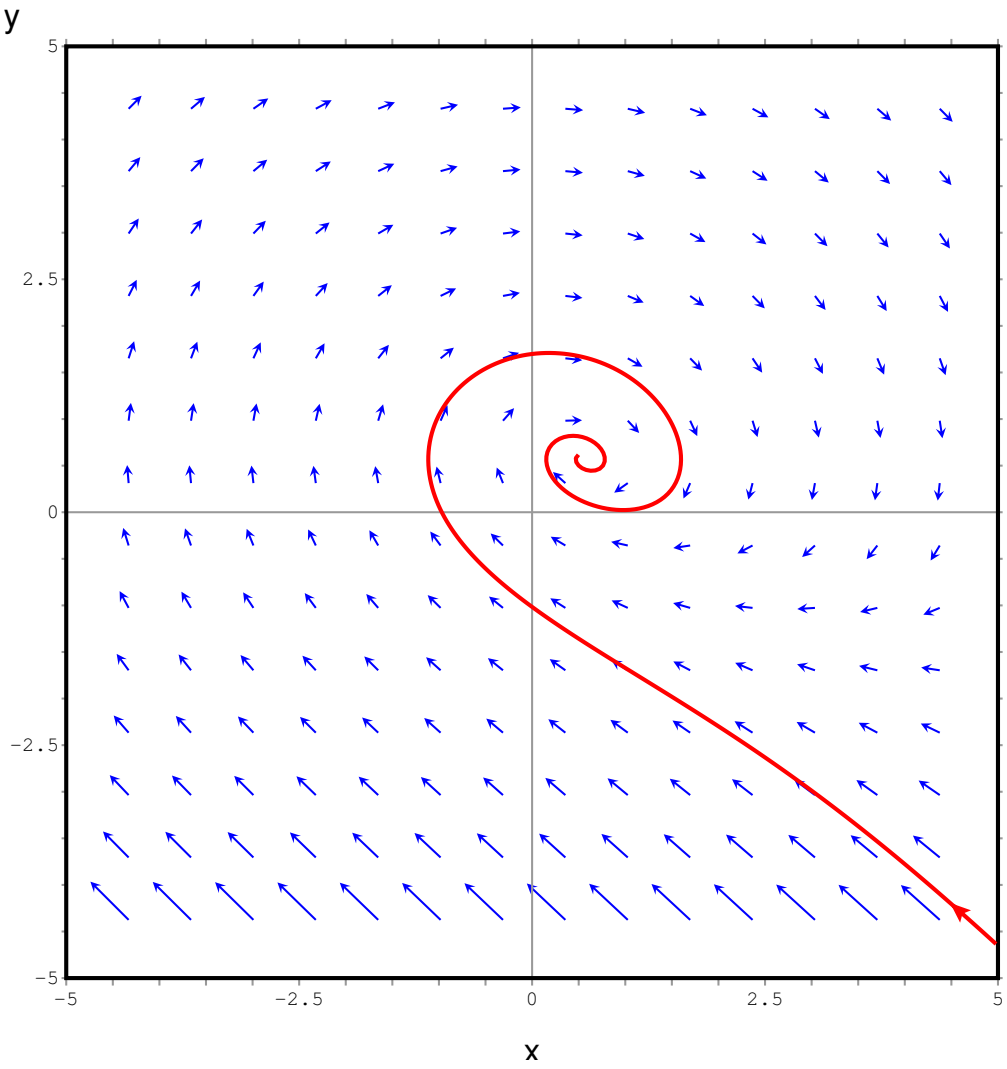


Figure 2.9: Designed slope field for an area of high performance.

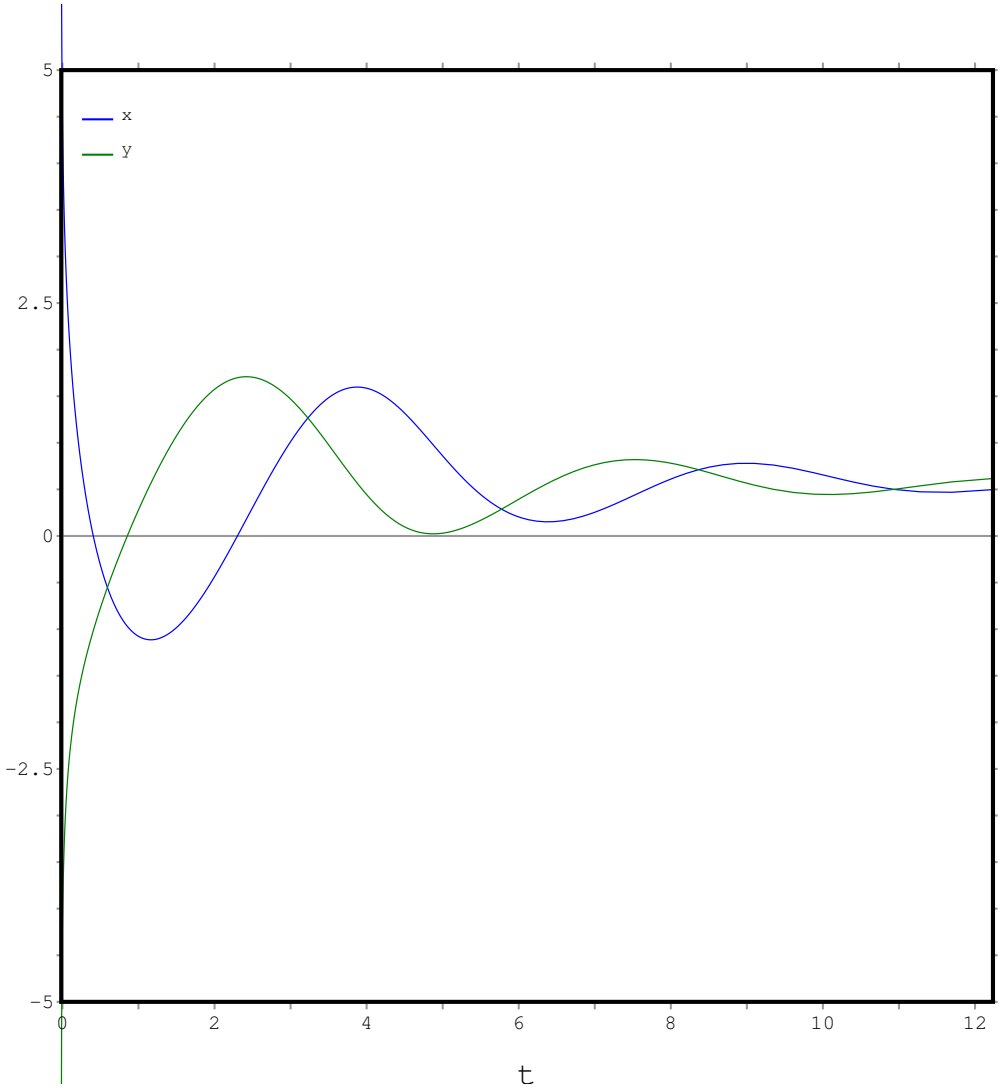


Figure 2.10: Exponential approximation of intensity gradient behavior in an area of high performance.

convergence of the search. ■

2.2 Cooperative navigation in robotic swarms: Features and unresolved problems

The aim of designing a system formed by a group of robots is to achieve their coordination and collaboration, in order for they complete a task that is difficult or impossible for a single robot. This is very interesting because these robots can be much simpler and economic. In addition, the parallel structure of the system provides a high level of robustness with respect to the development of the task. These systems, along with the swarm intelligence theory, have become in an area of great activity in robotics (Ping, Chao, Li, & Cuiming, 2010; Min & Wang, 2010, 2011). This kind of system was initially raised by (Reif & Wang, 1999) as a robotic economical solution for many real problems. There, in its strategy called *social potential fields*, they raise the basic characteristics of local interaction and self-organization.

To control the movement of individuals in the group, and the group as a system, different strategies exist. Some of them focuses the problem on the ability of each agent to obtain global, partial or even no environment information (Teturo, 2010). A common point in most research is the assumption of the capacity and ability of agents to sense the environment, and even communicate with other agents. This ability is essential to generate a collective behavior, particularly when working in dynamic environments.

Other strategies guide the study and design in the centralization of the control strategy (Savkin & Teimoori, 2010). While some research includes a central control unit, it is clear that such systems falls within the domain of decentralized control. However, it should be noted that such systems are dynamically decoupled, since the motion of one robot does not directly affect the motion of the others.

A first formal strategy was posed in (Vicsek, Czirók, Ben-Jacob, Cohen, & Shochet, 1995). They proposed a discrete-time model of a system consisting of several autonomous agents (particles), moving in the plane, with biologically motivated interaction. In their results, they indicate that a repulsion model in boundary conditions can be used to interpret observations of motions in bacterial colonies. Although the model is quite simple (each agent's motion is updated using a local rule based on its own state and the state of its neighbors), they were able to show that the emergent behavior of the system can be conditioned from simple rules of interaction, and biological systems such as bacteria colonies, are governed by these principles.

Many modifications of the Vicsek's model have been proposed since then, particularly adjusting and improving the equation of motion of each agent, considering the behavior of the agent group and the environment. However, many of such research consider simplest first- or second-order linear models for the motion of each agent, thereby, the obtained results are based on tools and methods from linear system theory. In (Savkin & Teimoori, 2010) they show examples of unrealistic physically embodied behavior that would be possible under such simplified models (movement restrictions in the real world).

In the QS model proposed in this research the motion of each agent is described by a

nonlinear model that allows to include restrictions inherent to the robot and the environment. Besides, the model as such is independent of the robot or agent used in the real world, it may be a differential wheeled robot, or a programmed code inside a large FPGA.

Approaches to group navigation problems also include formations, leader-follower schemes, abstractions, potential field methods, and Flocking. The first two schemes, formations and leader-follower, tend to restrict the relative position of the robots (Haghighi & Cheah, 2011; Grandi, Falconi, & Melchiorri, 2013; Sarkar & Kar, 2013; Yoshida, Fukushima, Kon, & Matsuno, 2014; H. Zhang, Meng, & Lin, 2012; H. Zhang, Zhao, & Lin, 2014). While it is true this reduces the complexity of the problem, facilitating its analysis, it is also true that many of these approaches have trouble keeping the formation in the presence of obstacles.

In the case of abstraction, the idea is to build abstractions to reduce the dimensionality of the problem (Filho & Pimenta, 2015; Ayanian & Kumar, 2010). This means making a mapping of the high-dimensional, multi-robot state space to a low-dimensional abstract space which characterizes the system dynamics. Here there is a group of algorithms that combines geometric tools which are common in observable and static environments. In these strategies the environment is divided into regions, and it is used the abstraction of the group of robots to reduce the dimensionality of the navigation problem (Ayanian & Kumar, 2010). The abstraction establishes a boundary for the group, allowing to reformulate the group navigation problem as a problem of planning and controlling the shape, position and orientation of this boundary. This strategy allows an explicit design of the navigation route without determining the position of each of the robots. These methods do not establish bounds on the positions of the robots which can be seen as a negative aspect, but in fact is positive in certain tasks on complex environments (for example, collapsed environments). The algorithm proposed in this research uses features of this category to design the navigation path.

A feature of these algorithms is that it does not allow to specify formations in the sense of exact shape and topology. However, it is precisely this characteristic that facilitates navigation in environments with obstacles, particularly with high dynamic.

Flocking or schooling strategies enable control of large groups of robots with relatively little computation (Sato, Kon, & Matsuno, 2011; Min & Wang, 2010, 2011). Usually, the group velocity is stabilized to a single velocity by each agent adjusting their velocity according to its neighbors. However, again in the presence of obstacles the schemes have many problems to ensure proper operation. Even, in some cases there are problems of local minimum, if there are not enough leaders (a problem widely known in potential field methods (Jacinto, Martínez, & Martínez, 2013)). The proposed QS algorithm also takes some features of these schemes to coordinate the collective movement. The scheme allows the inclusion of local communication among agents, and this information is included in the movement and behaviors activation policies.

With regard to the specific problems of coverage and grouping taken as an example in this research, there are parameters that allow to evaluate the degree of clustering and hence the performance of the algorithm used. This assessment can be done by calculating indexes that try to establish the quality of distribution, most of them centered on two properties: compatibility and separation. Compatibility corresponds to a measurement of the variation or dispersion of the agents in the group, while the separation considers the group structure (Brogan & Hodgins, 1997). Some valid parameters for measuring the

compatibility and separation include the uniformity in the dispersion and the radius of the group.

In contrast to the algorithms and strategies described above, the scheme based on QS proposed in this research it can hardly be classified into one of these categories. The scheme takes ideas from various approaches trying to find an optimal strategy for the particular type of problems of interest in terms of path planning: dynamic, unknown and observable environments. The algorithm does not intend to explicitly define the position of the robots. On the contrary, it looks to abstract the information from the environment to reduce the complexity in the design of the path. These features ensure navigation in environments with obstacles, even when they are in motion, one of the main problems of the cooperative navigation. It also eliminates the problem of impossible movements for the robot, since the dynamics of the robot is not part of the model. This allows the normal differences between robots, and even the use of radically different robots. Also, the problem of controller design is independent of the number of robots, which promises scalability to large groups and system functionality against the failure of a few of them (the characteristic of robustness sought in the research).

2.3 Macro-scale model

The motivation with this macro-scale model is to observe the set of agents as a single continuous behavior system. The final idea is to synthesize the desired behaviors at this level, which can then be executed by each agent individually. This structure of top-down design, defined as the starting point in the research, allows to define desired behaviors of the system which are then projected onto the behavior of each agent.

The tasks considered to be resolved in the research for this kind of system are related to robotics, specifically in problems of navigation and path planning. The reason for this decision is that the doctoral research internship was conducted in the Motion Strategy Lab of the Department of Computer Science at the University of Illinois (Urbana, USA), under the direction of Professor Steven LaValle (http://msl.cs.uiuc.edu/ lavalle/). However, extensions of the basic ideas can be applied to the solution of other problems.

The robot navigation problems considered allow to limit some of the features presented in the previous section, and extend other.

1. Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location.
2. The navigation problem only considers the movement of agents in a two dimensional plane. Besides, all agents are identical (a single species).
3. The behaviors of the agents are conditioned only by local events, interactions between agents and between agents and their environment.
4. Navigation tasks are related to real problems of search-and-rescue and surveillance.
5. The real problems to solve have multiple design criteria, this means greater number of behaviors for the agents. In particular, this implies great detail in the cell

differentiation (activation).

The representation that is used to describe and analyze the set of agents is consistent with that used in recent research on swarm robotic systems (bioinspired models) (Berman, Halász, Kumar, & Pratt, 2007b; Berman et al., 2007a; Lerman, Jones, Galstyan, & Matarić, 2006; Kazadi, Abdul-Khaliq, & Goodman, 2002). In particular, it is very important the work of (Belta & Kumar, 2004), where they build a map for the group of robots that results in a system of equations whose dimension is independent of the number of robots.

2.3.1 Model description

The macro-scale model seeks to formulate a set of equations that describes the average behavior of the bacterial population independently of the number of bacteria (agents). Then, this set of equations can be used to design the desired behavior of the AB (the solution to a problem formulated as a search). The principle of the algorithm based on QS is the grouping of agents. Agents are grouped into different areas of the environment according to local readings and their genome that follows some search criteria. The areas with the highest number of agents must show virulence and will correspond to the solutions of the problem.

Since the proposed model considers only local interactions, and that the goal of the design focuses on the coordination of the movement of the agents, the QS mechanism should facilitate the movement of agents in the environment towards areas with sufficient population, reflecting the decisions of many agents.

The system is composed of the n agents (n artificial bacteria) as a continuous system, whose dynamics can be described by differential equations. The agents in this system, all identical in design, experience at any instant a behavior l of a set l_a of possible behaviors of the agents (the micro-scale model is described in the next section). Furthermore, each agent is always performing a behavior at any given time. The variables in these differential equations indicate the size of the bacterial population that belongs to each of these behaviors.

Initially (Section 2.1) the author defined only two basic behaviors for the bacterial agents (two-stage proccess): Reproduction and virulence (activation or cell differentiation). However, the agents with the latter behavior may interpret their genome differently, depending on the conditions of population density and environment readings (information taken from sensors such as the position in the system, Section 2.1.2). This feature suggests the existence of other behaviors (in general a number a of behaviors) depending on the application to solve with the system.

In particular, considering the circumstances under which bacteria can be grouped, and that the intensity of this grouping (community size) is who allows to find a solution, the research proposes a three-stage process ($a = 3$). The additional intermediate stage should facilitate the population changes (Fig. 2.11).

Groups of agents, fractions of AB, at a given time experiencing the same behavior (any of the a behaviors). Therefore, the system can be analyzed as consisting of groups of agents characterized by a common behavior of a set of behaviors l_a.

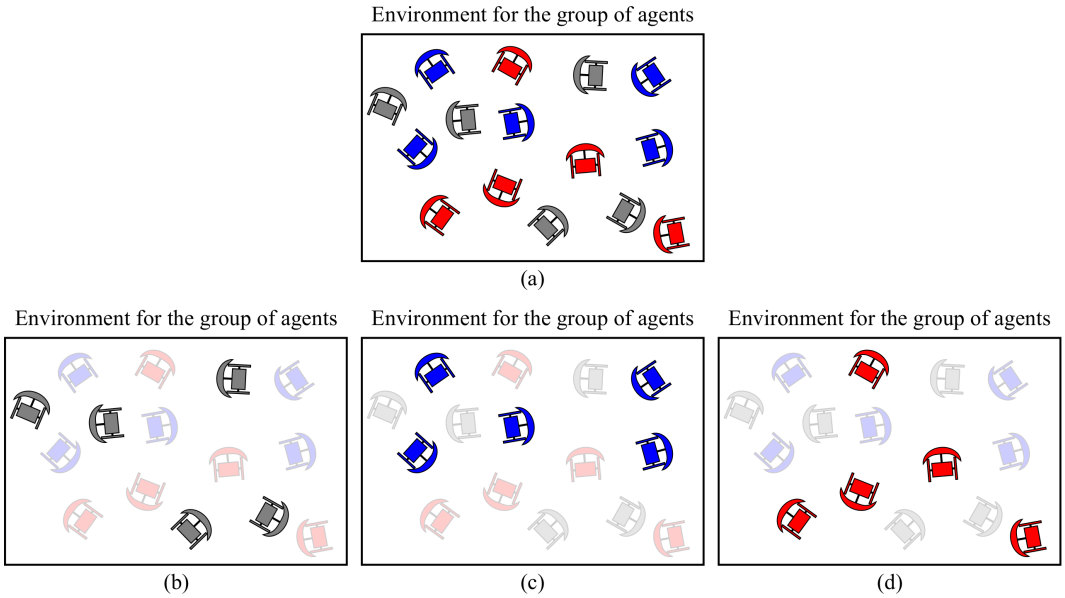


Figure 2.11: System and agents with three different behaviors.

(a) System composed of a group of agents (bacteria) in a three-stage process ($a = 3$), (b) system agents running behavior l_1, (c) system agents running behavior l_2, and (d) system agents running behavior l_3.

The model for this system is hybrid, both agent and system level. The set of agents, each with a continuous dynamic $\boldsymbol{X}_a$, switches between different behaviors during the development of the task. The group of agents simultaneously experiencing the same behavior, it also has a continuous collective behavior, so the system also has a continuous dynamic $\boldsymbol{X}_p$. When changing behavior one or more system agents (with discrete control mode determined by $\boldsymbol{L}_a$), changes the system as well (with discrete control mode determined by $\boldsymbol{L}_p$).

This means that the behavior space of the system is divided into a set $\boldsymbol{L}_p$ of l_a regions. The state of the system H_p is therefore defined by the values of the continuous variables of the system $\boldsymbol{X}_p$ and the discrete control mode determined by $\boldsymbol{L}_p$ (Fig. 2.12).

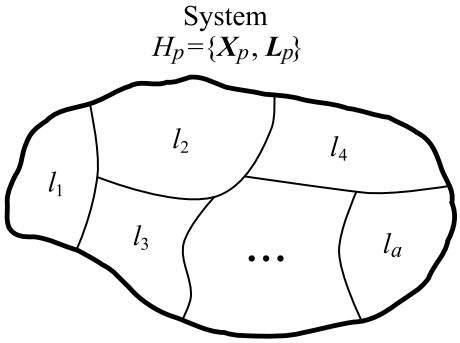


Figure 2.12: System and its fractions characterized by their behavior.

Each of these population fractions is characterized by a set of differential equations with state variables (vectors) denoted by:

$$x_i \quad / \quad i = 1, 2, 3, \cdots, b \tag{2.11}$$

A set of dimension b (the number of grouping areas defined by the agents in the system) that allows to describe the dynamics of the a different behaviors. The hybrid system then can be described as:

$$\begin{gathered} H_p = \{\boldsymbol{X}_p, \boldsymbol{L}_p\} \\ \boldsymbol{X}_p \subset \mathbb{R}^b \\ l_a \subset \boldsymbol{L}_p \end{gathered} \tag{2.12}$$

From Equation 2.12 it can be seen that the behavior defined for an agent affects the dynamics of the whole system. The next step is to define the model equations and limit them to a three-stage process.

Like other characterizations of collective behavior inspired by biological models (Franks, Pratt, Mallon, Britton, & Sumpter, 2002; Berman et al., 2007b), the research proposes the system as a three-stage process of individual behaviors. These three basic behaviors are:

- Reproduction
- Exploration
- Virulence

Where the new behavior, exploration, should help to find potential grouping areas. The principle of the algorithm based on QS is the grouping of agents. Agents are grouped into different areas of the environment (a total of b different areas) according to local readings and their genome (internal code) that follows some search criteria. The areas with the highest number of agents must show virulence and will correspond to the solutions of the problem. These population sizes are then the most important variables in the model.

The population size of each of these fractions of AB is characterized by a continuous variable in the following way:

- R Agents whose behavior is reproduction.
- X_i Agents whose behavior is exploration in the area i of a total of b areas in the environment.
- Z_i Agents whose behavior is virulence in the area i of a total of b areas in the environment.

It should be clarified that when agents begin to explore and activate their virulence, they begin to group in different areas of the environment (a total of b areas). This increases the number of system variables, where i indicates the grouping area of the agents. When the agents are reproducing there is no grouping.

The reproduction is the first behavior for all bacteria. After an artificial agent is enabled on the system (R), it begins to take local information but without activating its virulence (X_i), to finally activate its virulence when the quorum is fulfilled (Z_i). The activation or virulence of agents that are exploring the environment starts when $Z_i = T$, the quorum threshold. Both explorers and virulent can switch to new areas of the environment if the local information indicates that they are in areas of higher performance (according to the task). This is the most important part of the process, because it gives the opportunity to each agent individually to find the best solution (parallel processing and robustness characteristics that we seek in the system).

This parallel navigation structure allows that the explorers consider different areas of the environment before gathering around some of them. Thus, when a quorum is reached, the agents begin to become virulent according to the best option for each of them.

The flow chart of Fig. 2.13 shows the rules by which agents interact with each other and with the environment to meet the goal. In this case, the system is in equilibrium and able to solve the task. In Section 2.1.4 the bacterial population grows from an initial bacteria until reaching hardware limits (number of processors). Here, the population is constant. To consider in the model the incorporation of new agents, we must add an additional entry to the left block (Reproduction).

Fig. 2.13 shows the order in which the agents perform the behaviors, as well as the rate of change between them (μ, λ, k and ρ). As the model is supported by the population size, there must be the possibility of explorers and virulent bacteria change their behaviors according to its local readings. The ease with which these changes are made is modulated by these rates of change.

These rates are determined by the nature of the recruitment mechanism (defined as the way in which an agent communicates with its neighbors to try to form groups in its area, equivalent to the signal molecule), software and hardware, which means that they are strongly dependent on the final implementation of the system.

The recruitment is done through local communication, from agent to agent, that is, an agent at a time. This means that the rate of recruitment is not dependent on the number of explorers agents. Additionally, the explorers agents who have not recruited can discover areas of interest (the explorers agents discover the area i at per capita rate μ_i), which makes the exploration process as important as the recruitment.

The λ_i is the per capita rate at which explorers recruit agents to area i. The ρ_{ij} is the per capita rate at which explorers of area i encounter the area j and switch their allegiance by becoming explorers, of that area. The rate of recruitment λ_i does not encode any information regarding the area i, that is, recruitment is not faster in areas with better performance. Once committed to a particular area, an agent will attempt to recruit to that area with the same vigor as it would to any other suitable area. However, thanks to the activation threshold of QS, the waiting times before starting the recruitment to Z_i (described by k_i) depend on the quality of area i. Thus, the best performing areas more quickly activate the quorum consensus.

The explorers then have enough time to move to areas with better performance, that is, a migration from area i to the area j a rate of ρ_{ij}. The behavior of virulent bacteria is always defined by the fulfillment of the condition of quorum. If the population $Z_i < T$,

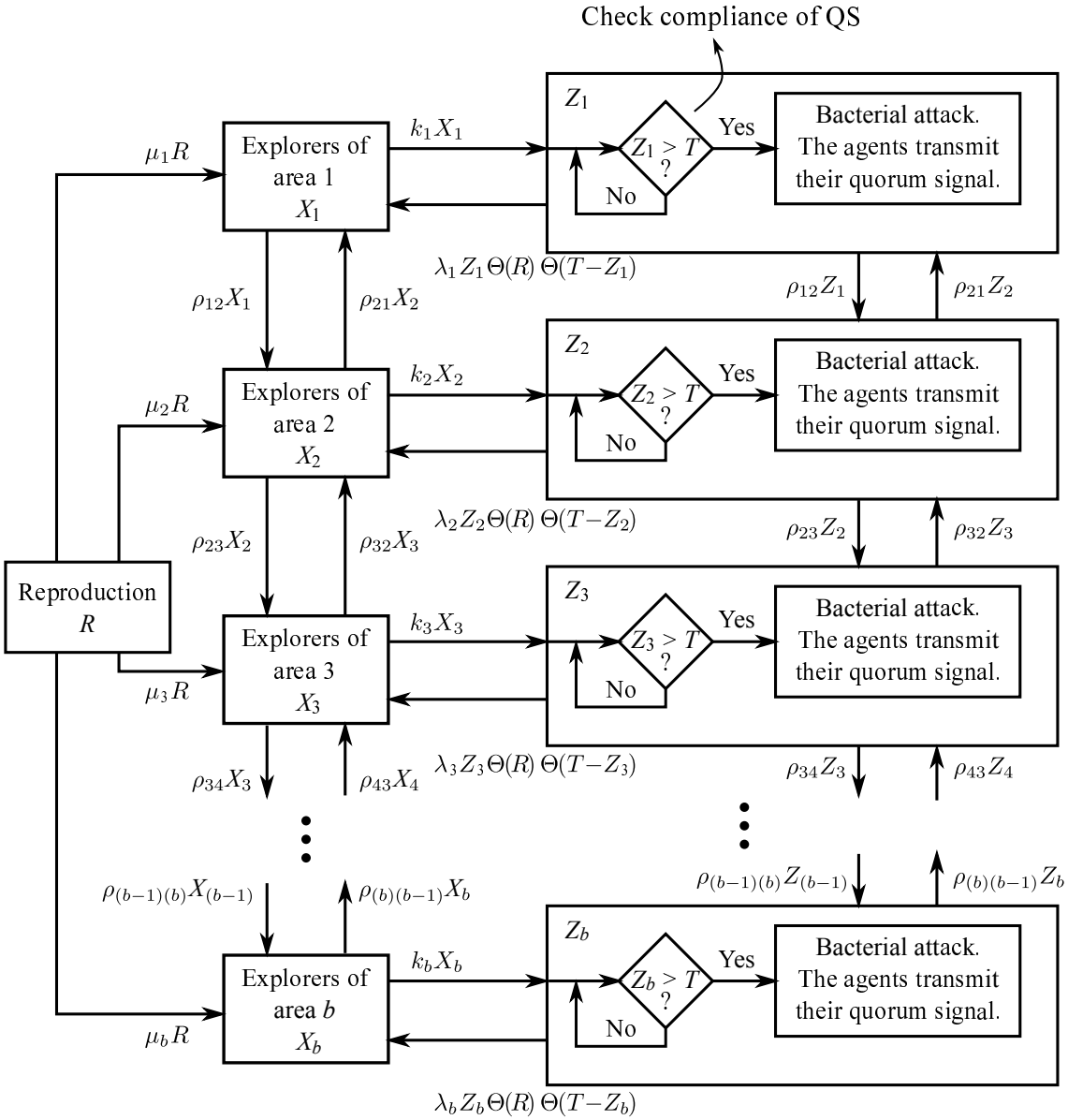


Figure 2.13: Flow chart for the proposed dynamic of bacterial QS.

the quorum threshold, then the quorum is not met. Finally, the function $\Theta(\cdot)$ takes the value of 1 if $\cdot > 0$, and 0 otherwise. This function allows to describe the activation of QS.

From the flow chart of Fig. 2.13, we can write the following equations for the system:

$$\begin{gathered}
\dot{R} = -(\mu_1 + \mu_2 + \cdots + \mu_b) R \\
\dot{X}_1 = \mu_1 R + \lambda_1 Z_1 \Theta(R)\,\Theta(T - Z_1) - \rho_{12} X_1 + \rho_{21} X_2 - k_1 X_1 \\
\dot{X}_2 = \mu_2 R + \lambda_2 Z_2 \Theta(R)\,\Theta(T - Z_2) - \rho_{23} X_2 + \rho_{32} X_3 + \rho_{12} X_1 - \rho_{21} X_2 - k_2 X_2 \\
\vdots \\
\dot{X}_{(b-1)} = \mu_{(b-1)} R + \lambda_{(b-1)} Z_{(b-1)} \Theta(R)\,\Theta\left(T - Z_{(b-1)}\right) - \rho_{(b-1)(b)} X_{(b-1)} + \rho_{(b)(b-1)} X_b + \\
+ \rho_{(b-2)(b-1)} X_{(b-2)} - \rho_{(b-1)(b-2)} X_{(b-1)} - k_{(b-1)} X_{(b-1)} \\
\dot{X}_b = \mu_b R + \lambda_b Z_b \Theta(R)\,\Theta(T - Z_b) + \rho_{(b-1)(b)} X_{(b-1)} - \rho_{(b)(b-1)} X_b - k_b X_b \\
\dot{Z}_1 = k_1 X_1 - \rho_{12} Z_1 + \rho_{21} Z_2 - \lambda_1 Z_1 \Theta(R)\,\Theta(T - Z_1) \\
\dot{Z}_2 = k_2 X_2 - \rho_{23} Z_2 + \rho_{32} Z_3 + \rho_{12} Z_1 - \rho_{21} Z_2 - \lambda_2 Z_2 \Theta(R)\,\Theta(T - Z_2) \\
\vdots \\
\dot{Z}_{(b-1)} = k_{(b-1)} X_{(b-1)} - \rho_{(b-1)(b)} Z_{(b-1)} + \rho_{(b)(b-1)} Z_b + \rho_{(b-2)(b-1)} Z_{(b-2)} - \\
- \rho_{(b-1)(b-2)} Z_{(b-1)} - \lambda_{(b-1)} Z_{(b-1)} \Theta(R)\,\Theta\left(T - Z_{(b-1)}\right) \\
\dot{Z}_b = k_b X_b + \rho_{(b-1)(b)} Z_{(b-1)} - \rho_{(b)(b-1)} Z_b - \lambda_b Z_b \Theta(R)\,\Theta(T - Z_b)
\end{gathered} \tag{2.13}$$

To observe the overall behavior of the system described by these equations, we consider below a first navigation task. Let us think in an environment with four high performance areas (for example, meeting areas with some interesting variable: temperature, noise, light, humidity,...) denoted by the points p_1, p_2, p_3 and p_4. Among these points, we will assume that p_1, p_2 and p_4 have a similar performance (local maxima, the variable is higher than the average values of the environment, but not the maximum), while point p_3 has a superior performance (global maximum, the variable has the maximum value in the environment). Under these conditions, we expect that the agents are distributed in the environment and navigate initially grouped around these four points. But then, they must migrate to the point of higher performance p_3. Fig. 2.14 shows the flow chart for this example of four areas.

We assume for simplicity that for the initial selection of the four areas do not exist favoritism criteria, i.e.:

$$\begin{gathered}
\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu \\
\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda
\end{gathered} \tag{2.14}$$

The migration process is based solely on the performance of the areas detected by the agents (ρ and k calculation) and the QS. Accordingly, the system of equations 2.13 can be written as 2.15.

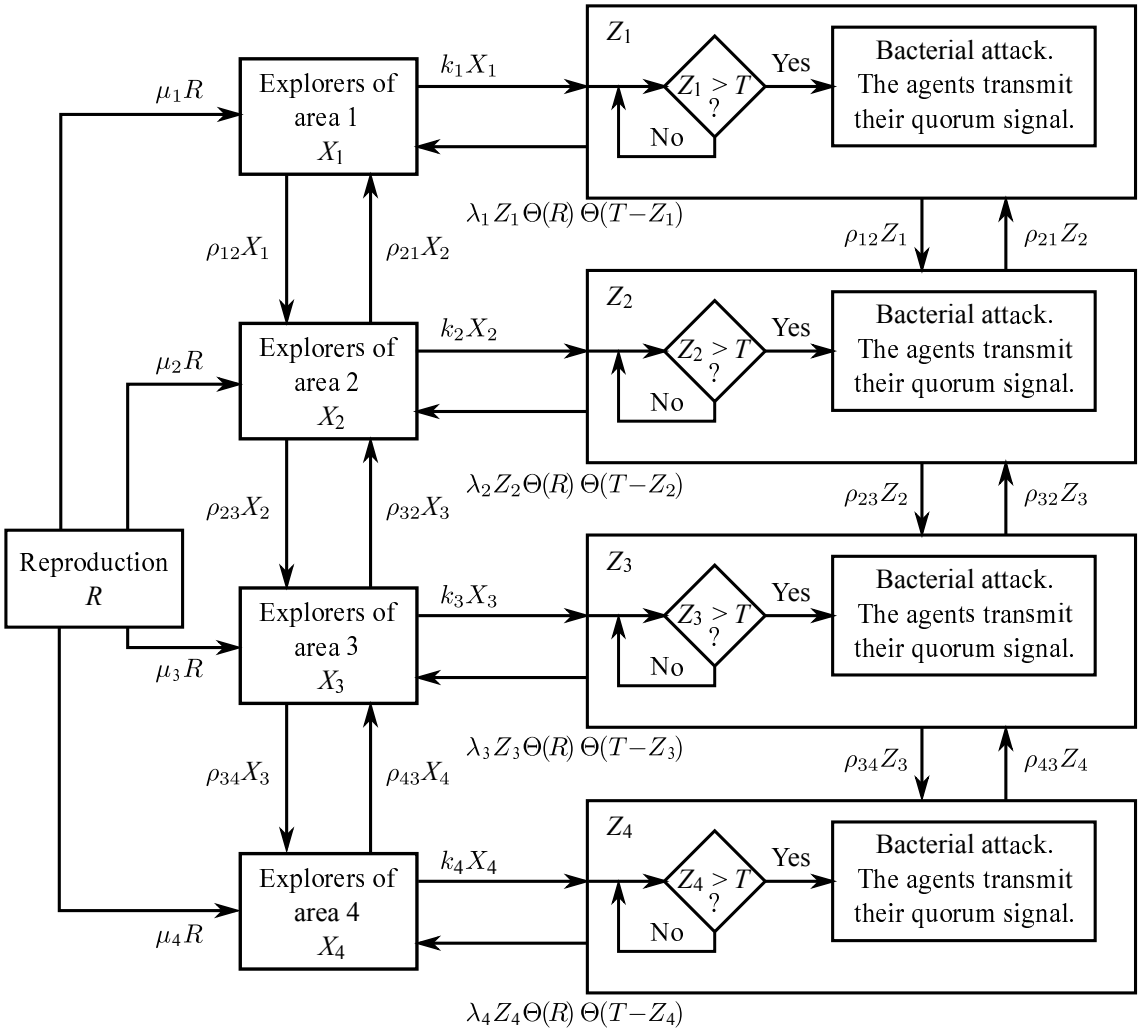


Figure 2.14: Flow chart for the example of four areas.

$$
\begin{aligned}
\dot{R} &= -4\mu R \\
\dot{X}_1 &= \mu R + \lambda Z_1 \Theta(R)\,\Theta(T - Z_1) - \rho_{12} X_1 + \rho_{21} X_2 - k_1 X_1 \\
\dot{X}_2 &= \mu R + \lambda Z_2 \Theta(R)\,\Theta(T - Z_2) - \rho_{23} X_2 + \rho_{32} X_3 + \rho_{12} X_1 - \rho_{21} X_2 - k_2 X_2 \\
\dot{X}_3 &= \mu R + \lambda Z_3 \Theta(R)\,\Theta(T - Z_3) - \rho_{34} X_3 + \rho_{43} X_4 + \rho_{23} X_2 - \rho_{32} X_3 - k_3 X_3 \\
\dot{X}_4 &= \mu R + \lambda Z_4 \Theta(R)\,\Theta(T - Z_4) + \rho_{34} X_3 - \rho_{43} X_4 - k_4 X_4 \\
\dot{Z}_1 &= k_1 X_1 - \rho_{12} Z_1 + \rho_{21} Z_2 - \lambda Z_1 \theta(R)\,\Theta(T - Z_1) \\
\dot{Z}_2 &= k_2 X_2 - \rho_{23} Z_2 + \rho_{32} Z_3 + \rho_{12} Z_1 - \rho_{21} Z_2 - \lambda Z_2 \theta(R)\,\Theta(T - Z_2) \\
\dot{Z}_3 &= k_3 X_3 - \rho_{34} Z_3 + \rho_{43} Z_4 + \rho_{23} Z_2 - \rho_{32} Z_3 - \lambda Z_3 \Theta(R)\,\theta(T - Z_3) \\
\dot{Z}_4 &= k_4 X_4 + \rho_{34} Z_3 - \rho_{43} Z_4 - \lambda Z_4 \Theta(R)\,\Theta(T - Z_4)
\end{aligned}
\tag{2.15}
$$

In the final applications, each agent must determine independently the value of the coefficients from local interactions between agents and with the environment. That is, agents can modify their environment to communicate locally, but it is also possible to use some special landmarks in the environment. For now, for this first test will be assumed the following values (the range of values is selected according to the displacement speed of the agents, and therefore, the system response time, in this case in the order of minutes):

$$
\begin{aligned}
\mu &= 0.01 & \rho_{34} &= 0.001 \\
\lambda &= 0.03 & \rho_{43} &= 0.01 \\
\rho_{12} &= 0.001 & k_1 &= 0.01 \\
\rho_{21} &= 0.001 & k_2 &= 0.01 \\
\rho_{23} &= 0.01 & k_3 &= 0.02 \\
\rho_{32} &= 0.001 & k_4 &= 0.01
\end{aligned}
\tag{2.16}
$$

These values agree with the characteristics described for this example (higher value for ρ_{23}, ρ_{43} and k_3). Fig. 2.15, 2.16 and 2.17 show the results of this first evaluation. We simulate the system with an initial population of 300 agents in reproduction and a quorum threshold of $T = 10$. Fig. 2.15 shows how this population is rapidly reduced by activating the other two behaviors (almost 10 percent over the first 50 min.). In Fig. 2.16 is shown as explorers agents increase in the four areas. While growth is similar in all four cases, we can appreciate more agents in area 3 (blue curve). Finally, in Fig. 2.17 is shown as increasing the virulent agents in the four regions. Initially, the agents increase in all four cases. However, in areas 1, 2 and 4 the number of agents decreases after about 200 to 300 minutes, while in area 3 (blue curve) the number of agents always increases. These curves clearly show the migration of agents to the area 3.

With a quorum threshold of $T = 10$ the four areas achieve to activate the QS at some point (area three at 25 min., area one at 43 min., area two at 68 min. and area four at 73 min.). Upon reaching a population of 10 agents, in four cases there was an increase in the rate of population growth. With a quorum threshold of $T = 60$, only area three activates the QS at 137 min. In the latter case, the population of virulent agents in two and four areas never exceeds 10 agents, and in the area one reaches up to 18 agents.

In the cases analyzed, it is clear how the size of the quorum threshold affects QS. The variation in the value of T directly affects the characteristics of the population growth in each of the areas. Small values of T allow virulent agents to reach the QS and increase

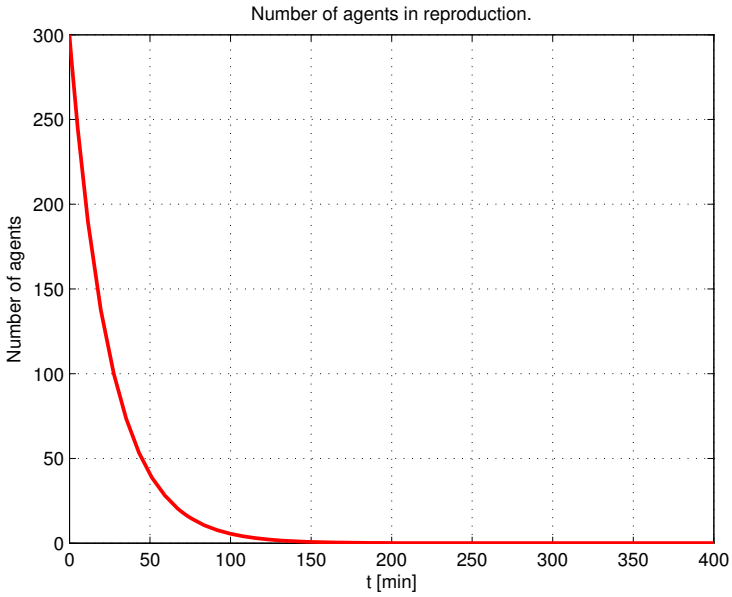


Figure 2.15: Macro model simulation results for a simple grouping task ($T = 10$) - Reproduction.

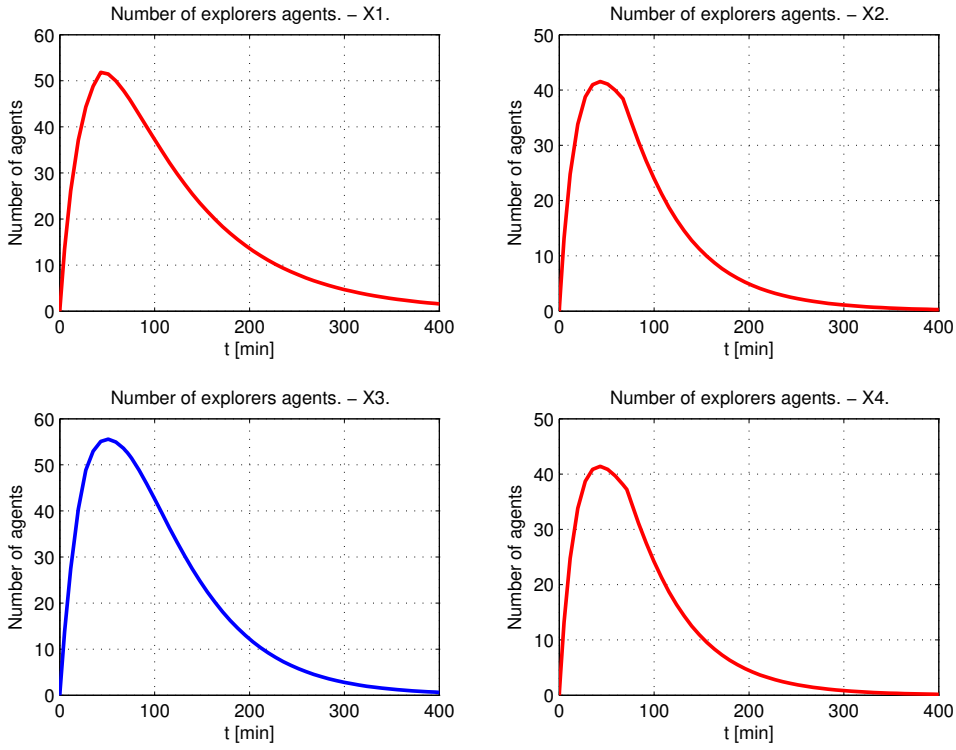


Figure 2.16: Macro model simulation results for a simple grouping task ($T = 10$) - Exploration.

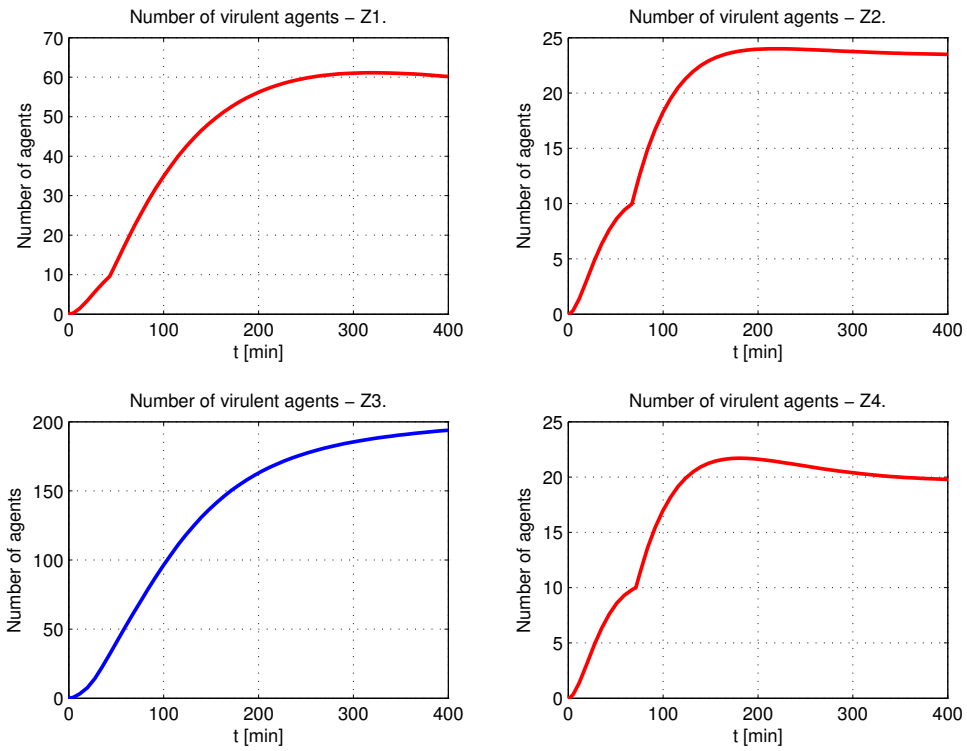


Figure 2.17: Macro model simulation results for a simple grouping task ($T = 10$) - Virulence.

their bacterial population in local minima, but they do not exceed the local maximum. In this case, the grouping is also faster. With large values of T, the QS tends to be expressed only in the global maximum, but the process takes a little longer. In both cases, the QS helps to find the global maximum (more virulent bacteria gather at the area of higher performance).

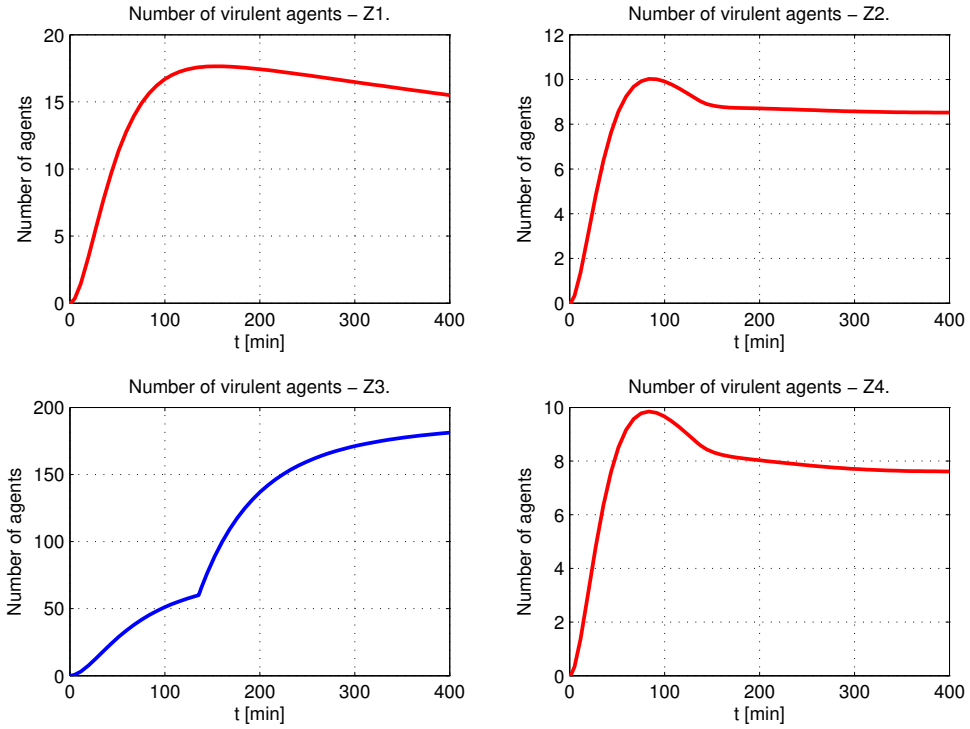


Figure 2.18: Macro model simulation results for a simple grouping task ($T = 60$) - Virulence.

2.4 Micro-scale model

After that we have a correct description of the behavior of the group of agents (macro-scale model), in this section we show a model of agent that allows to implement navigation tasks. In this case, our motivation, rather than compute the exact trajectories and sensory information of individual robots, we want our agents to be able to build an information space from the local information that they receive from the environment and its near neighbors, and from this information space, define its behavior required for the development of the task. This idea can be extended to search problems, area in which it is possible to use the proposed model.

2.4.1 Model description

We start from the assumption that an agent is capable of switching behavior, and develop together with other agents a task if it is able to make and remember local observations of the environment. These two properties are essential for the construction of the information space, which is used in the control hybrid structure for the feedback of each agent.

As described above, we consider our system as a set of n agents (differential wheeled robots in our experiments), each with a continuous state space $X_a \subset \mathbb{R}^2$, where the actions or behaviors of the agent belongs to a set L_a of l_a behaviors (Fig. 2.19). An agent may switch its state according to its control policy when it determines it is appropriate to do so. The activation of each of these behaviors should be done in a way that improves the overall performance of the system (the performance function should be encoded in its genome), in our case this means performing the task in the shortest time possible. Accordingly, each agent is a hybrid automaton whose continuous and discrete dynamics can be represented by:

$$H_a = \{\boldsymbol{X}_a, \boldsymbol{L}_a\} \tag{2.17}$$

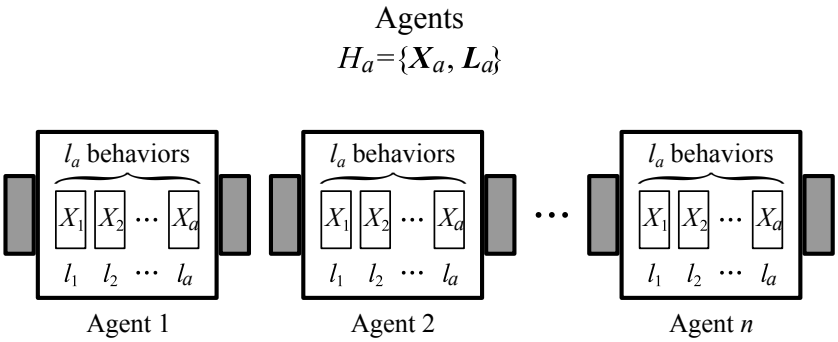


Figure 2.19: The agents in the system.

The agents have no global information from their environment or other agents (they do not know how many agents exist in the environment). Instead, the agents can sample the environment (assumed to be finite). They are able to move in the environment and make local observations of its neighbors and the environment. We assume that the agents are able to observe behaviors and discriminate their types (interpreting some actions, or by direct communication from their neighbors). Actually, this feature is part of the agent's local communication with neighbors.

Therefore, we conceive the agent as an individual of the system that stores information about its participation in the system (of their behaviors, and the conditions under which switches between them), local position, velocity, and is capable of reading local environment information, and communicate with their neighbors, using these two features to modify its navigation in the environment (to switch behavior).

Regarding its local position, the agent does not know the environment. However, using the local information that it senses, it is able to navigate using control policies defined for the application. The agent position, which is represented by a Cartesian coordinate in the

plane of environment, is numerically updated at each time step. The update equation is defined according to the navigation strategy designed for the task.

These are general characteristics that each agent must fulfill. The design of the code in each agent should implement these features conditioned to the development of the task.

2.4.2 Individual agent behaviors: A first approach

In this final section, the author will show how to generate basic navigation behaviors without requiring system identification, geometric map building, localization or state estimation. Instead, he presents an approach where the set of agents (robots) executes wild motions (the concept of wild motion, dynamic characteristic required by the system, it will be discussed in detail in the first section of the next chapter) along the boundary of their environment. This is an initial approach focused specifically on the problem of robotic navigation, and with a simple navigation task. The idea of this first approach is to show how to describe the model of each agent according to the problem to solve.

The prototypes used in these experiments are equipped with basic sensors to extract the minimum amount of information necessary of the interaction: an impact sensor and a visual sensor, which serves to establish a minimum local communication. The strategy, as shown in the model described in the previous sections, does not process or transmit state information (position, velocity, ...). The coordination of the robots to carry out the task is achieved by the visual information captured by the robot, and a set of behavioral rules. Using this scheme, we demonstrate that the system can solve basic navigation tasks such as coverage and grouping.

As mentioned, the tasks to be performed by the robotic system are related to collective navigation. In this sense, we wanted to duplicate the behavior of a bacterial community in a group of robots. In this simplified model, each bacterium contains a small microcontroller that stores the behavioral code of the bacterium, and reacts according to environmental conditions. In this second prototype system, each agent is implemented as a differential robot (Fig. 2.20). This interaction allows the execution of two basic functions:

- Reproduction: Scheme by which an artificial bacterium reproduces in the system. It is basically a process of cell division, where the growth limit is defined by the amount of resources needed to implement the operation of the system.
- Cell differentiation (activation): Each artificial bacterium in the system has an identical copy of the code. However, depending on population density and its physical position, each bacterium interprets it in a particular way.

The act of loading the code (program the microcontroller with the information about its functionality) from one bacterium to another neighbor, is called reproduction (as in the first platform, the code is in the inactive robot, in reproduction the robot is activated through a signal). Similarly, the act of loading the genome in a number of bacteria is called population growth. The system architecture is infinitely scalable (all agents are identical in hardware and software); there is no architectural impediment to scale the system to any desired size.

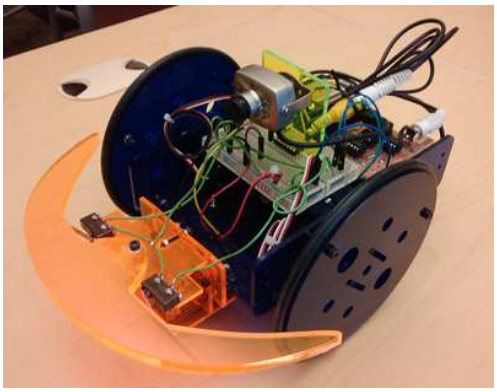


Figure 2.20: Differential robot with Atmel 8-bit AVR RISC-based microcontroller.

Reproduction is performed by an active robot on an inert robot in the environment. Initially, just one of the group is programed, all other robots are resources that can be used for population growth. The initial function of all robots is reproduction, and so on until the robot decides not to be able to be reproduced, or until the population is large enough for the task. At that time, the robot becomes virulent and autonomously changes to the process of cell differentiation. Cell differentiation makes all robots complete the task.

Until now, the most important functional difference with respect to the first platform is that robots can physically move in the environment (in the first platform with the CPLDs, the possibility of movement was discussed changing the label that identifies each agent, but this feature was not implemented). However, in this new case, the AB will implement a solution to a problem. The problem is grouping robots, reason why a scheme where the robot identifies movement and uses this information to learn how to group is proposed.

The learning ability is included to show the diversity of options available to implement the genome of each agent. Control policies can be very complex, or as simple as a table of data in ROM. The dynamic learning structure is implemented with a basic adaptive system: an adaptive linear combiner with recurrence (Fig. 2.22) (Delgado, Kambhampati, & Warwick, 1995). This filter performs a sum of products, it has a non-linear dynamics, and allows each output to take into account not only the input density h, but also the previous speed of the wheels (memory). This is just one possible example of updating equation, the model allows any structure to govern the movement of the agent.

The outputs are normalized with a hard limiter, whose saturation value corresponds to the rated speed of the wheels. This allows modeling with state diagrams (Fig. 2.21), which is very important when updating equation is supported in control policies.

Each wheel has only two options of movement: forward $\dot{\theta}_0$ and backward $-\dot{\theta}_0$. These two movement options correspond to the possible outputs of the linear combiner. In the diagram, there are four states characterized by a pair of speeds. Each data of the pair corresponds to the speed of a wheel: (left wheel speed, right wheel speed). Since each wheel has two possible states, combining them produces the four states of motion of each robot (Forward, Backward, Right and Left). As each wheel can change its state, the robot can go directly to any other state (or remain in the same).

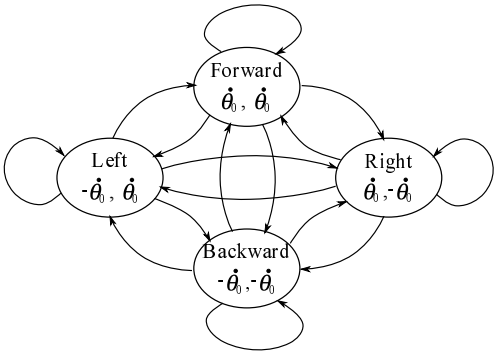


Figure 2.21: State diagram corresponding to the robot's movement.

The construction of the diagram is made from the point of view of movement of the motors.

These control outputs are defined as follows:

$$\begin{aligned} \dot{\theta}_{rk} &= \varphi\left(h_k\omega_{r0_k} + \dot{\theta}_{rk-1}\omega_{r_k} + \dot{\theta}_{lk-1}\omega_{lr_k}\right) \\ \dot{\theta}_{lk} &= \varphi\left(h_k\omega_{l0_k} + \dot{\theta}_{lk-1}\omega_{l_k} + \dot{\theta}_{rk-1}\omega_{rl_k}\right) \end{aligned} \tag{2.18}$$

where $\varphi(\cdot)$ is the symmetric hard-limit function defined as:

$$\varphi(n) = \begin{cases} \dot{\theta}_0 & \text{if } n \geqslant 0 \\ -\dot{\theta}_0 & \text{otherwise} \end{cases} \tag{2.19}$$

and $\dot{\theta}_0$ is the rated speed of the wheel of the robot.

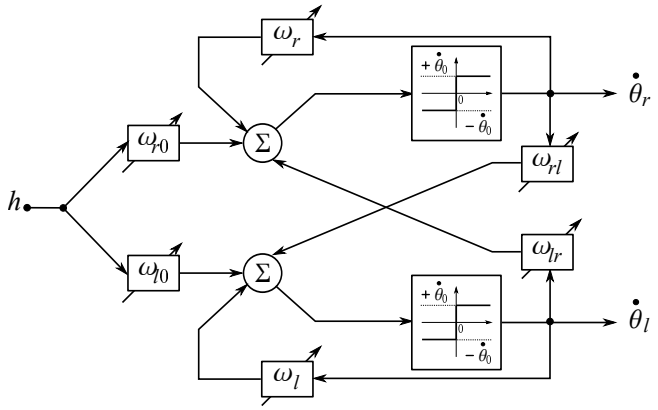


Figure 2.22: Adaptive linear combiner with recurrence.

Each agent does the adjustment of the adaptive coefficients, or filter weights, on-line dynamically and autonomously according to their local observations. The weight vector to be optimized has the following structure:

$$\begin{bmatrix} \omega_{r0_k} & \omega_{l0_k} & \omega_{r_k} & \omega_{l_k} & \omega_{rl_k} & \omega_{lr_k} \end{bmatrix} \tag{2.20}$$

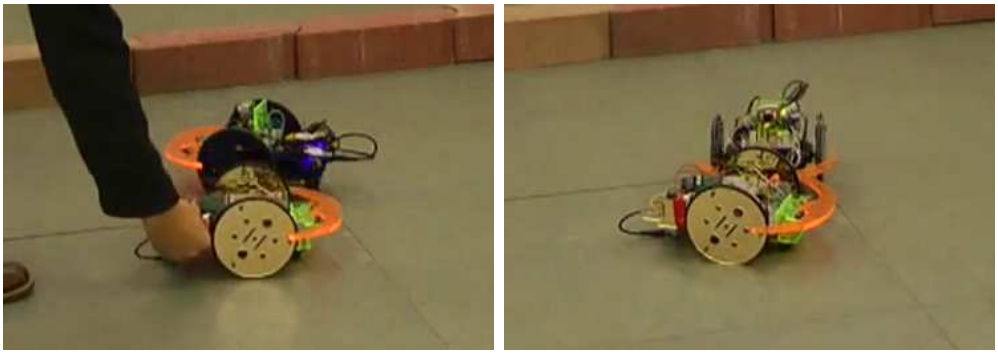
This decision structure in some way reproduces the sonar and radar systems. Systems like the bat ear, also perform some kind of spatial filtering (interferometry), which is then used by the auditory system to produce an action (Kössl & Russell, 1995). Our linear combiner also acts as a spatial filter with a direction set by the movement of the agent.

Robots perform initially a wild movement. This means that they do not follow any specific control rule, they only move in the environment and change their direction when hitting an obstacle (a wall or another robot). During operation, the movement of the robot responds to the observations of the camera, its current state, and the linear combiner. As the agents evaluate their performance and adjust their coefficients, the system begins to reflect the wanted collective behavior.

Coverage

The dynamic of each robot has two stages: an initial stage of reproduction and a final stage of cell differentiation (activation), which develops the task. The stage of reproduction is always the same in all cases: when a robot is activated (initial robot or robot activated by another robot), it begins its behavior at this stage. The robot looks randomly inactive robots around, and when one is found, it is headed in that way to activate it. Robots differ from the environment due to their physical characteristics. In the prototype, this special feature is its structure made of colored acrylic.

If after some time the robot has not been reproduced, the robot moves to the stage of cell differentiation, in this case, the coverage of the environment (Fig. 2.23). A video showing this behavior can be viewed at (Martinez S., 2011b) (*Quorum sensing-based navigation - Reproduction 1*).



(a) (b)

Figure 2.23: Basic behavior for autonomous navigation: Reproduction.

(a) An inactive robot (gold) is placed in the field of view of active robot (blue). (b) After a few seconds of searching, the blue robot moves toward the inactive robot and hits it.

In large environments, where the agents are widely separated from each other, the

reproduction with the inactive robots is performed using short-range wireless communication, specifically ZigBee IEEE 802.15.4 (2.4 GHz XBee S1 modules), this extends the activation range up to 100 m in open field.

The second stage, cell differentiation, corresponds to the adaptation and development of the task, in this case, the navigation of an unfamiliar environment. The objective of this application is that robots navigate around the environment. This behavior is achieved if the movement is coordinated with the proper function of performance, in this case, a function that is optimized looking for areas with low motion. According to this, we define the following fitness function:

$$s = \frac{1}{\|h_{actual} - 0.1\|} \tag{2.21}$$

This function, with a range from zero to 10, gives greater performance to weight vectors that direct the robot to areas with little movement. The value of the density h, integer defined from the data captured by the camera, has a range of zero to 15, zero for no movement detected and 15 for a high degree of movement.

Grouping

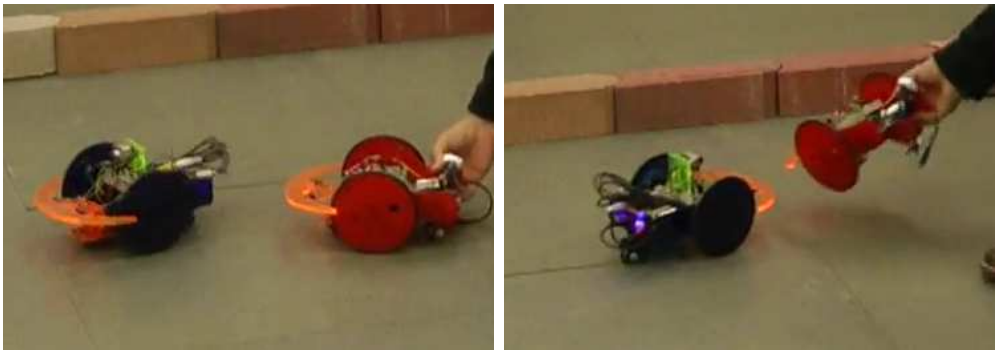
By way of comparison, it is also interesting to observe the opposite behavior, that all robots are grouped into a single point in the environment. The reproductive stage is identical to the case of the task of covering, in fact, it is possible to think that this task of grouping is necessary after a scan task.

In this case, the cell differentiation that leads to development of the task is achieved by optimizing the behavior to favor the robot movement toward to areas of the environment with a high degree of movement. According to this, the following performance function is defined:

$$s = \frac{1}{\|h_{actual} - 15.1\|} \tag{2.22}$$

This function gives greater performance to weight vectors that direct the robot to areas with other robots in motion. In our prototype, the robot performs a random search in the environment, and when it detects any movement, it goes to that point (Fig. 2.24). A video showing this behavior can be viewed at (Martinez S., 2011a) (*Quorum sensing-based navigation - Grouping of the robots*).

We have taken the task of covering and grouping as basic examples for observing the performance of the proposed control structure. Based on them, it is possible to think about more complex tasks such as carrying objects, adjusting the algorithm for robots to surround the object, and then moving together to a certain point in the environment. In the next chapter it will be shown formally how to design more complex tasks for the system using the macro and micro models.



(a) (b)

Figure 2.24: Basic behavior for autonomous navigation: Grouping.

(a) An inactive robot (red) is placed in the field of view of the active robot (blue). (b) After 1 minute of exploration without being able to identify robots in motion, the red robot is shaken in front of the blue robot to force its identification and movement.

Experimental Prototype and Simulations

For our experiments, we use the Oomlout Company's open-source design SERB Robot (Oomlout, 2013) (Appendix A). We modify the design to have a more robust bumper system using a similar geometry to the SERB robot chassis, and we add a camera as visual sensor. Our robot and all its peripherals, including the camera is handled by a single 8-bit microcontroller (Atmel ATmega328, 8-bit AVR RISC-based microcontroller).

We also use the idea of minimalist design in terms of information that the robot senses to perform tasks. While the camera can capture analog video with color, the microcontroller only captures monochrome frames of 8 x 8 pixels using the A/D converter. Several of these frames are used to build a video signal, but with very little visual information. We just take the video information related to the level of movement in front of the robot, in this way, we do not need a huge processing capacity.

We can control the translational and rotational speeds of the robot. However, we decide to design a software actuator on the displacement of the robot, a high-level control, which generates control states in order to facilitate the construction of state diagrams. In this way, each wheel has two choices of motion: clockwise or counterclockwise. Therefore, the robots can move forward, backward, or turn on their axis in any direction. An important aspect of the design of the robot, which facilitated its modeling, is that the robot's center of rotation coincides with its geometric center.

The system is homogeneous, that is, all agents are identical in design. If a task requires more agents, we only have to place them in the environment. Due to the impossibility of building tens of these robots, we only use the prototype robots to observe the basic behaviors of the algorithm. We have done the analysis of the dynamics of the system with a large number of robots through simulation.

The simulations are performed under Linux (kernel 2.6, 32 bits) on Player/Stage

(Gerkey, Vaughan, & Howard, 2003; Vaughan, 2008), this platform allows for the simulation to run exactly the same code developed in C for the real robot. The simulations consider not only the functionality of the algorithm, but also the physical characteristics of the robot, i.e. shape, size (0.26 m x 0.22 m) and weight (1.2 kg). The camera is installed just above its geometric center both in the real prototype as in the simulation.

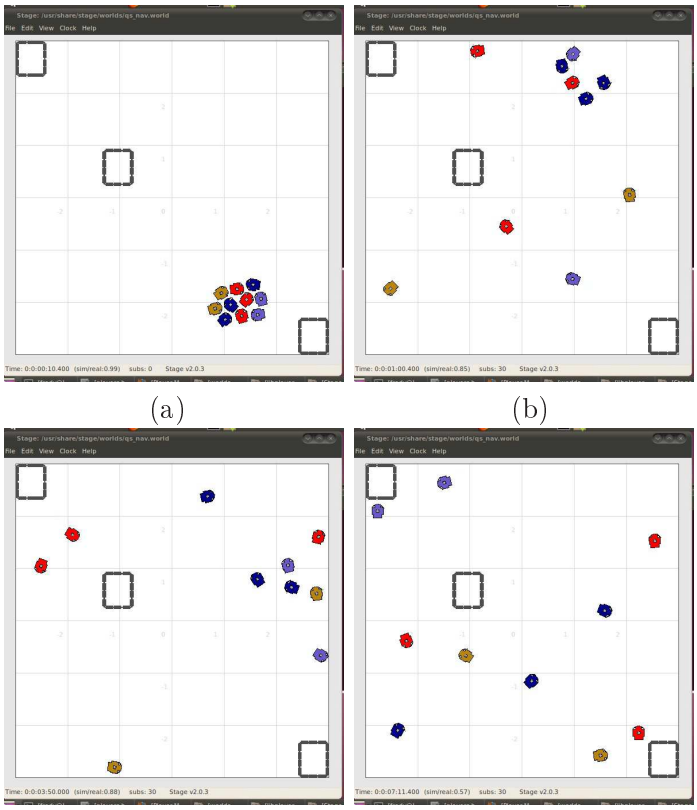
The simulation environment is designed in square shape with a total area of 6 m$\times$6 m = 36 m^2. Inside the environment, there are three holes inaccessible to the robots as an obstacle for navigation. In the simulations there are 10 agents, all completely independent of each other.

In the first case, for the covering task, we start the simulation with all the robots gathered in the lower right side of the environment. The simulation starts at t = 10 s, and the case documented here ends at 7 min. and 11 s (00:07:11, time in the simulation, Fig. 2.25). Throughout the simulation, is possible to observe the robots form small groups at certain times due to the dynamics of the system, but these groups are dispersed quickly and we do not observe later the same organization. In the end, all the robots navigate the environment avoiding the formation of groups. The simulation in Player/Stage can be observed in (Martinez S., 2012a) (*Quorum sensing-based navigation - Coverage simulation*).

From the viewpoint of the agent model, the observed behavior of the Fig. 2.25 shows two things. First, it shows that it is possible to define the behavior of the entire system from behavioral rules designed for each agent. Although this application is limited to the use of signal molecule without triggering the QS, the fact is that the control policies of each agent, coordinated by local interaction, let reflect a complex collective behavior. Second, it also shows that the rules of each agent can be as complex as the designer and the application required. It is possible to have reactive behavior, memory of previous events and even make the agent response evolves along the interaction.

In the second case, for the task of grouping, we start the simulation with the robots scattered in the environment. The simulation starts at t = 26 s, and the case documented here ends at 12 min. and 16 s (00:12:16, time in the simulation, Fig. 2.26). In this case, the collective behavior of the robots takes longer to be evident, i.e., the learning is more complex and time consuming. This simulation shows a behavior much more interesting than the previous one. For example, we see first signs of grouping of 2 or 3 agents throughout the simulation, and a behavior that resembles a dance between two robots at 6 min. and 42 s (00:06:42) that lasts a total of 38 s. At the end of this simulation we can observe a fairly stable group of robots in the lower right side of the environment. The simulation in Player/Stage can be observed in (Martinez S., 2012b) (*Quorum sensing-based navigation - Grouping simulation*).

The simulation again demonstrates the ability of the system to solve a complex task. It also shows that the performance is tied to the interpretation of the local information sensed by the agent (genome), rather than the platform. That is, the behavior can be implemented on other hardware, or emulated by software. In this second simulation the QS neither is implemented. However, through further analysis, as detailed in the next chapter, if we include the QS in the behavior of the agents, the time required is reduced by 60%.

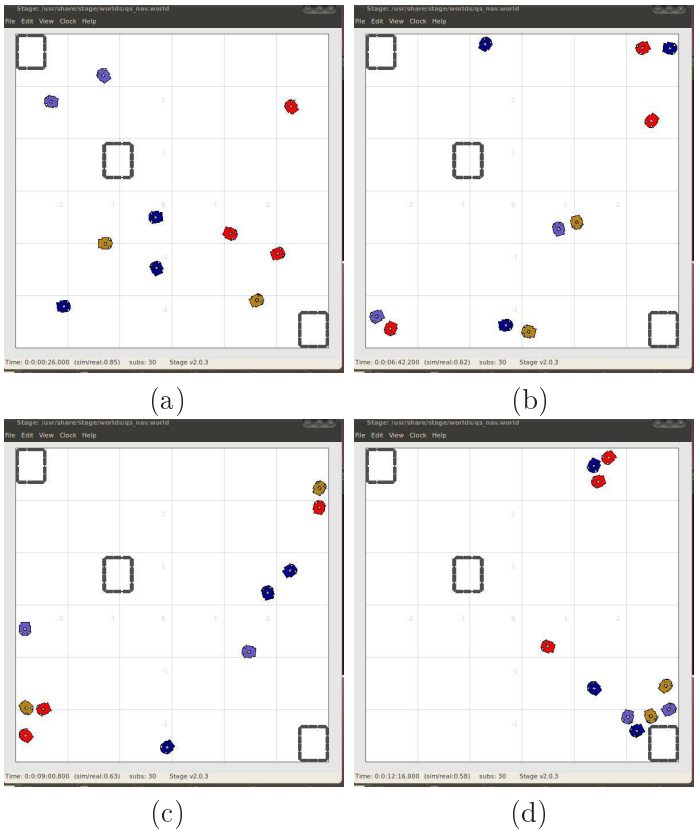


(a) (b)

(c) (d)

Figure 2.25: Simulation of the autonomous navigation: Coverage.

Robots learn to navigate autonomously and cover the whole environment without forming groups. (a) Initial state of the simulation (00:00:10). (b) and (c) Two states along the simulation (00:01:00, and 00:03:50), where we observe some momentary clusters of robots. (d) Final state of the simulation, the robots navigate the environment without forming groups.



(a) (b)

(c) (d)

Figure 2.26: Simulation of the autonomous navigation: Grouping.

Robots learn to navigate autonomously looking for the most densely populated areas, therefore, after some time, they learn to meet in groups. (a) Initial state of the simulation (00:00:26). (b) The two robots in the center (blue and golden) begin a dance that lasts for 38 seconds (00:06:42). (c) At 9 min. (00:09:00) we observe the formation of small groups of robots trying to stay together. (d) Final state of the simulation, the robots have established a stable group in the lower right side of the environment (00:12:16).

2.5 Summary

In this chapter, we show the mathematical model developed for the simplified dynamic of bacterial QS. This is the dynamic model adopted for the design, which simplifies the actual behavior of the bacteria.

First, we perform a theoretical analysis and a hardware emulation of interpreted behavior of bacterial QS. From this study, we develop our mathematical model as a two-level structure: a first system-level model, where general behavior of the bacterial system can be projected, and a model at the agent level, where we detail the agent's own actions which together reflect the collective behavior of the system.

Since both the system and each of the agents are modeled as a set of behaviors, and that the switching among these behaviors is triggered by specific actions, determined by the local communication among agents or by local environmental readings (event-driven or event-based), our models correspond to hybrid systems, more specifically, each agent is a hybrid automaton.

The most important contribution of this chapter is that, it shows how the biological QS is interpreted in the proposed model, and how this interpretation is included in a mathematical model to predict the effect of QS on the performance of a task, that is, in the behavior of the system. It is also important the relationship between the two mathematical models (system and agent), and the impact of behaviors of the agent on the system.

CHAPTER 3

A first application: Robot motion planning

In summary, in Chapter 1, the author shows the theoretical background of the research. The chapter begins studying the idea of robust design, and clearly reflects the need for an integrated system design, comprehensive concept of the CPS. After, the author presents a state of art of different models based on bacteria used in hardware design. This state of the art updates the research proposal, and illustrates the applications developed with these models. Finally, in the last part of the chapter, the author transcribes the proposed objectives.

In Chapter 2, the author presents the model of multi-agent dynamic system proposed. In the first part the author describes the behavior of biological QS. From this behavior, the author proposes an equivalent behavior on electronic hardware. In this Bacterial Platform, the author establishes the basic principles of the model, in particular the concept of bacterium, its basic behaviors, and the genome that interprets local interaction. Then, according to the basic principles of behavior, the author establishes two models to mathematically to describe the system: a system-level model and an agent-level model. Both models correspond to hybrid structures because they are supported in several behaviors that are activated according to local events. With the system-level model is possible to observe the evolution of the community over time and observe the effect of QS in response speed. With the model-level agent is possible to observe how behaviors of an agent influence the system response. In addition to the simulations, all individual behaviors are verified on hardware.

The intention in this Chapter 3 is to show how it is possible to design and solve real problems using the bacterial QS scheme proposed. The author shows how to design the solution of different navigational tasks, first considering only interaction between agents, and second, adding a few changes to the navigation environment (installation of landmarks in the environment). Bacteria respond not only to the signal molecule of other bacteria, they are also affected by other elements in the environment as food or temperature. Thinking about navigation, something similar happens to a person when looking to leave a room, or a mouse when looking for food. This idea can be implemented by using landmarks in the environment, special marks with which the agent can interact. But first, it is necessary to introduce a dynamic characteristic required by the agents, and that in robotics has been

called Wild Motion (Bobadilla, Sanchez, Czarnowski, Gossman, & LaValle, 2011).

The agents do not satisfy any equation of motion and do not use state feedback. They just move forward and rotate randomly when they crash. In this sense, we are motivated by the study of dynamical billiards. They are dynamical systems in which a particle, in our case the agents, alternates between motion in a straight line and specular reflections from a boundary. When the particle hits the boundary, it reflects from it without a loss of speed. An interesting feature of these systems is their ergodic motion[1] on surfaces of negative curvature (Fig. 3.1).

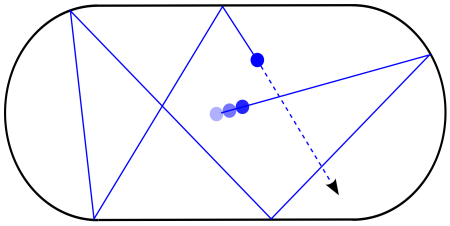


Figure 3.1: The Bunimovich stadium.

This is a well-known example of an ergodic system (Bunimovich, 1979). The "hockey puck" will strike every open set along the boundary as it travels forever (Stamatious, 2009).

But the system does not really need ergodicity. We need to ensure that the agents hit each open set of the region at some time. That is, we just need topological transitivity to ensure that the agents cross the gates at meet certain conditions. The topological transitivity means that after several iterations, the robots will visit each open set of the region. This is what is called Wild Motion.

This chapter begins studying navigation strategies based solely on wild motion. The study is done by analyzing the dynamic characteristics of wild motion through the design of basic navigation tasks that consider only two behaviors in the agents, behaviors induced by specific elements in the environment (virtual gates). This study focuses on the wild motion, and does not include QS.

The second section focuses on how to use the algorithm for solving a problem. Here a design process by which it seeks to characterize the task as a search problem is presented. After the search problem is formulated, the algorithm characteristics are defined.

Subsequently, in the final two sections, the author shows two examples of design of navigation tasks in which agents have more behaviors, and switch between them according to local readings. These design examples include both wild motion and design based on QS. In the first case, the author presents a strategy for grouping in areas where the agents navigate using only local communication. The second navigation strategy corresponds to a more general case, where there is interaction with the environment (landmarks), and which proposes a specific navigation path.

[1] The ergodic properties of a dynamic system are related to two basic features: 1) The particle (agent) does not lose speed (energy) throughout their interaction in the environment, and 2) given the negative curvature of the boundaries of the environment, the particle eventually will navigate all the environment (the system should run for a long time).

The performance analysis of the designed navigation tasks is done by simulation. Experiments with physical robots are very costly and time consuming. In addition, the performance analysis of a group of agents is complicated. The simulations are much faster and much more reliable than experiments, this facilitates the generalization of results, since few experiments (either real prototypes or by simulation) are not enough to generalize a behavior. The simulator Player/Stage introduced in the previous chapter, is a very powerful tool that allows modeling environments and robots very close to real versions. The author selects this tool for the analysis of results due to its characteristics and the reputation it has gained in scientific circles.

As in the previous chapter, the basic behaviors are always evaluated on real prototypes. Then, to analyze the performance of the system, the author performs simulations with different behaviors and a large number of agents. The analysis includes a large number of simulations for different cases with and without QS.

3.1 Wild motion, ergodicity and swarm robots

This first section shows a first application using only the dynamic characteristics of wild motion. Through the design of different basic navigation tasks, and their implementation on real robots, we seek to demonstrate the properties of this dynamic (wild motion) independently of QS.

The agents correspond to the simplest structure possible. The dynamic switching is performed by interpreting the environment "gates" encoding navigation information (the decision to move from one region to another depends on how the robot interprets this gate). This interpretation is related to policies designed to control the task.

The agents execute wild motions, which means each one will strike every open set infinitely often along the boundary of any connected region in which it is placed. The environment is divided into a discrete set of regions, with boundaries delineated with simple markers, such as colored tape (Fig. 3.2). Using simple sensor feedback (the robot controls its movement according to local readings), we show that complex tasks can be solved, such as patrolling, disentanglement, and basic navigation. The method is implemented on real robots and simulation. This work was developed with the Motion Strategy Lab at the Department of Computer Science of the University of Illinois at Urbana-Champaign, and presented in (Bobabilla & Martinez S., 2011; Bobadilla, Martinez, Gobst, Gossman, & LaValle, 2012).

3.1.1 Background

The robots work with very little information due to very limited sensing. There is no precise model of the equations of motion, and state feedback is impossible. We start with uncalibrated, "wildly behaving" robots that move more-or-less straight until a wall is contacted. They then pick a random direction to repel and continue moving straight (Fig. 3.3).

This motion primitive is inspired by *dynamical billiards* (Bunimovich, 1979; Tabachnikov, 2005). The only sensors required for navigation are simple contact sensors to detect

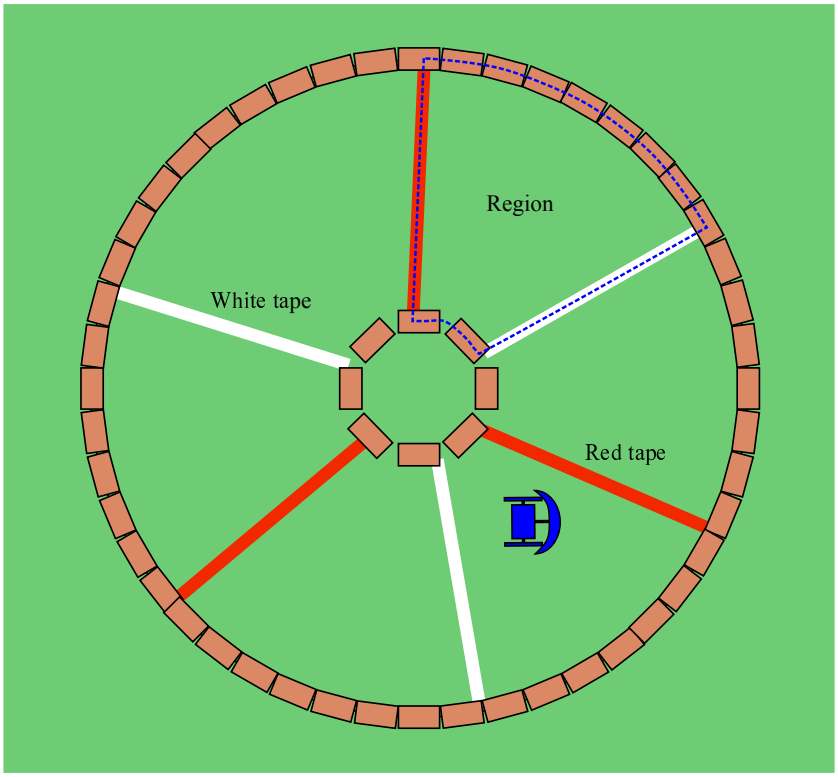


Figure 3.2: Navigation environment and regions.

Navigation environment designed to study the wild motion. The figure shows each of the regions in the environment (six regions), and the virtual gates implemented with different color tape (red and white). The small blocks are the boundaries of the environment.

obstacles and boundaries, and an inexpensive color sensor that can detect simple landmarks in the environment. Such a robot can be built with inexpensive parts for under $100 US (Fig. 3.4 (Bobadilla et al., 2012)). We can only *guide* the robot through its environment by designing appropriate responses to limited sensor feedback and sensing history. To achieve this, we formulate tasks in terms of a hybrid system (Branicky, Borkar, & Mitter, 1998; Egerstedt & Hu, 2001; Fierro, Das, Kumar, & Ostrowski, 2001; Frazzoli, Dahleh, & Feron, 1999; Haghverdi, Tabuada, & Pappas, 2005; Tomlin, Pappas, & Sastry, 1998).

As it is common in many approaches (Fainekos, 2011; Bhatia, Kavraki, & Vardi, 2010; Fainekos, Girard, Kress-Gazit, & Pappas, 2009; Finucane et al., 2010; Kloetzer & Belta, 2010; Kress-Gazit, Fainekos, & Pappas, 2008; Kress-Gazit, Fainekos, & Pappas, 2009; Kress-Gazit, Fainekos, & Pappas, 2005; Kress-Gazit, Fainekos, & Pappas, 2007; Lahijanian et al., 2010; Smith et al., 2010; Wu, Yan, Lin, & Lan, 2009), we partition the environment into a finite set of regions over which a discrete transition system is defined. Rather than develop state-feedback control laws within regions (Alur et al., 2002; Egerstedt & Hu, 2001; Finucane et al., 2010; Lahijanian et al., 2010; Smith et al., 2010; Kloetzer & Belta, 2010), we do not even attempt to stabilize the robots (to put the robot in a known initial state). We instead place *virtual gates* along the boundaries between regions that possibly

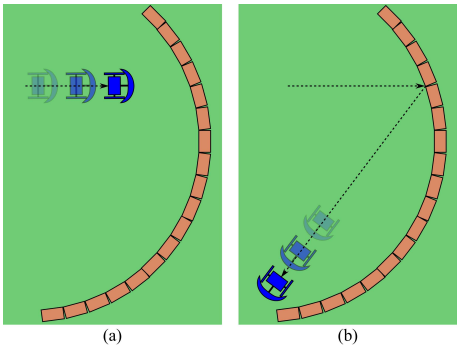


Figure 3.3: Robot interaction with environment boundary.

(a) The robot moves toward the environment boundary. (b) The robot impacts the environment boundary and randomly changes its direction.

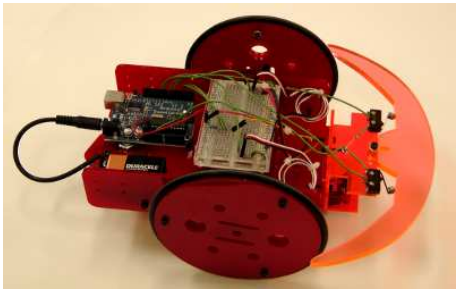


Figure 3.4: Differential robot with contact and color sensors.

enable discrete transitions, depending on information provided by a combinatorial filter (reactive system which makes the decision in accordance with sensed inputs) (LaValle, 2011; Tovar, Cohen, & LaValle, 2009) that maintains information states (because there is no state feedback, the state of the robot is described by local readings and its history) from weak sensor data.

3.1.2 Mathematical Model

A collection of n robots (numbered 1 to n) is placed into a compact, connected planar workspace $\mathcal{W} \subset \mathbb{R}^2$. Let $\partial\mathcal{W}$ denote the boundary of $\mathcal{W}$. Let Γ be a set of m *virtual gates*, for which each $\gamma_i \in \Gamma$ is the image of an injective, rectifiable curve $\tau_i : [0, 1] \to \mathcal{W}$ for which $\tau(0) \in \partial\mathcal{W}$ and $\tau(1) \in \partial\mathcal{W}$. Let C be a set of k *colors*, with $k \leq m$. Each virtual gate is labeled with a color by a given mapping $\kappa : \Gamma \to C$ (Bobadilla et al., 2012).

The gates induce a decomposition of $\mathcal{W}$ into connected *cells* (regions in Fig. 3.2 and Fig. 3.5). If the gates in Γ are pairwise disjoint (or mutually disjoint, if every two gates are disjoint), then each $\gamma_i \in \Gamma$ is a 1-cell and the 2-cells are maximal regions bounded by 1-cells and intervals of $\partial\mathcal{W}$. If gates intersect, then the 1-cells are maximal intervals between any gate intersection points or elements of $\partial\mathcal{W}$; the 2-cells follow accordingly. In either case, every 2-cell will be called a *region* and denoted as r. Let $\mathcal{R}$ denote the collection of all

regions (Bobadilla et al., 2012).

Figure 3.5: An annulus-shaped environment that has 6 regions.

There are three RED gates and three WHITE gates.

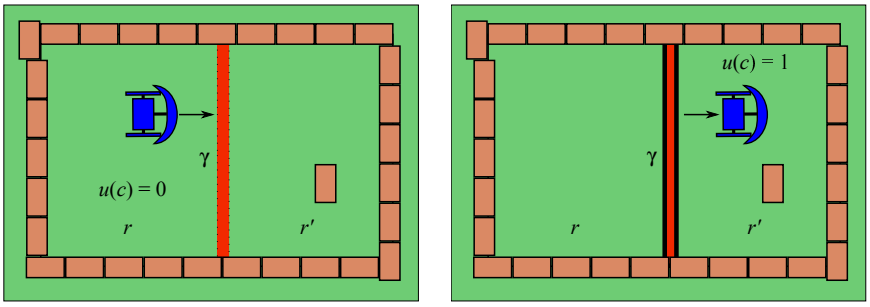
The robots are considered small with respect to $\mathcal{W}$ and the regions $r \in \mathcal{R}$. They are essentially modeled as points, but they may have specific kinematics, as in the common case of differential drive robots. More precisely, the assumption is that the collision-free subsets of $\mathcal{W}$ and every $r \in \mathcal{R}$ are homeomorphic (from a topological viewpoint they are the same) to those obtained for the case of a point robot, regardless of each robot's geometry. Furthermore, any $r \in \mathcal{R}$ is assumed to be able to fit all n robots without complicated geometric packing challenges (Schwartz & Sharir, 1983).

The particular equations of motion $\dot{x} = f(x)$ for each robot is unimportant in this approach. We do not explicitly control their motions and do not even attempt to measure their precise state. Instead, we rely on the fact that the robot moves in a wild, uncontrollable way, but the trajectory satisfies the following high-level property: For any region $r \in \mathcal{R}$, it is assumed that the robot moves on a trajectory that causes it to strike every open interval in ∂r (the boundary of r) infinitely often, with non-zero, non-tangential velocities. A body that satisfies this property is called *wild* (Bobadilla, Sanchez, Czarnowski, Gossman, & LaValle, 2011).

One set of systems that achieves this motion are the *ergodic* systems that arise in the study of dynamical billiards (Bunimovich, 1979). In this case, a Newtonian particle moves at constant velocity and then bounces with standard reflection laws from the boundary (Fig. 3.1). In most planar regions, ergodicity is achieved, which implies wildness. An alternative model, used in our experiments, is to bounce in a random direction from the boundary, rather than at a prescribed angle. This is preferable with the real robots because they cannot sense the angle of incidence. In the case of n robots, they may contact each other in a common region, a random bounce direction is used in this case as well (Bobadilla et al., 2012).

A *control mode* is a mapping $u : C \to \{0, 1\}$ that assigns one of two behaviors to every color gate. Let U denote the set of all possible control modes ($u \in U$). For a color $c \in C$, $u(c) = 0$ means that any virtual gate of color c does not obstruct the robot's motion. The control mode $u(c) = 1$ means that the robot must treat every virtual gate of color c as a wall that blocks its motion (Fig. 3.6).

For a single robot, we define a discrete transition system D_1 that simulates the original



(a) (b)

Figure 3.6: Virtual gate.

a) Gate open for red color from left to right. b) Gate closed for red color from right to left.

hybrid system. Let the state space of the discrete system be $\mathcal{R}$. The transition system is defined as:

$$D_1 = (\mathcal{R}, r_0, \rightarrow_1), \tag{3.1}$$

in which r_0 is the region that initially contains the robot. The transition relation $r \rightarrow_1 r'$ is true if and only if r and r' share a common border which corresponds to a virtual gate $\gamma_i \in \Gamma$. The environment with the regions defined by virtual gates may be analyzed by means of a labeled transition graph, in which the vertex set is the set of all regions $\mathcal{R}$, and every edge is a possible transition, labeled by the virtual gate color that allows it (Fig. 3.7).

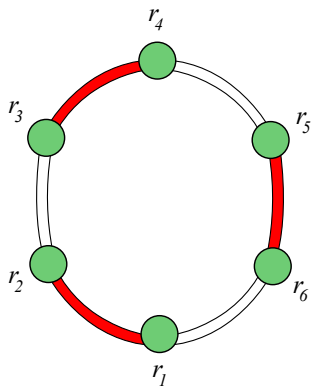


Figure 3.7: Edge-colored transition graph.

The region graph of Fig. 3.2 and Fig. 3.5. The vertex are labeled with the name of each region, and the edges are labeled with the virtual gate color.

It is straightforward to show that D_1 is a simulation of the original hybrid system. Therefore, we can design a solution plan over D_1, thereby inducing the correct behavior of the original hybrid system. This is the standard approach to hybrid system control using a

discrete abstraction. In the case of n robots, D_1 is extended in the obvious way by making an n-fold Cartesian product of the transition graph. This results in a discrete transition system D_n that simulates the motions of all robots.

We develop an event-based system (Åström, 2007). Each robot starts with an initial control mode. During execution, the control mode may change only when receiving an sensor observation event y.

Depending on the system, the events considered are:

1. **Gate crossing:** The robot detects that it crossed a virtual gate of color c. The observation is $y = c$.
2. **Timer expire:** The robot has been within a single region for at least t seconds. The observation is $y =$ TIMEOUT.
3. **Change in lighting:** The ambient room light changed, either as $y =$ LIGHTTODARK or $y =$ DARKTOLIGHT.
4. **Communication:** The robot receives a message from robot i that robot i crossed the gate of color c. The observation is $y = (c, i)$.

Let Y denote the set of all possible observation events for a robot in a particular system, then $y \in Y$.

The control modes of robot i are set during execution according to a policy. Since state feedback is not possible, *information feedback* is instead used. Let a *filter* be any mapping of the form $\phi : \mathcal{I} \times Y \rightarrow \mathcal{I}$, in which $\mathcal{I}$ denotes an *information space* (LaValle, 2006) that is designed appropriately for the task ($\mathcal{I}$ depends on the task, this should become clear in Section 3.1.3). The initial $\iota \in \mathcal{I}$ is given, and each time a new observation event occurs, an information-state transition occurs in the filter. A *control policy* is specified as an information-feedback mapping $\pi : \mathcal{I} \rightarrow U$, which enables the control mode to be set according to the sensor observation history (Fig. 3.8).

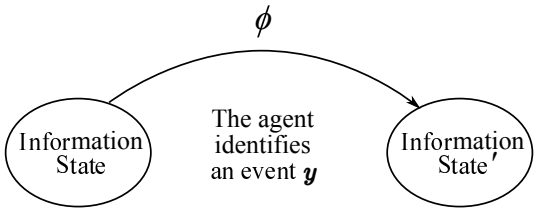


Figure 3.8: Information-state transition.

In other words, the genome of each agent encodes a control policy, which determines the behavior of the agent. In the mathematical description, this genome is interpreted as a filter whose input values are defined by the information captured by the agent (information feedback). The agent only has this information to determine its new state, i.e., there is no state feedback. The state of continuous variables is unknown, both the agent $\boldsymbol{X}_a$ and the system $\boldsymbol{X}_p$. The system evolves driven by observations of the agent.

3.1.3 Solving Tasks

Some examples of tasks that can be solved with the wild motion and different information spaces, filters and control polices are: Patrolling, separating into teams, navigation, reactive tasks, time-based policies and communication based strategies. Full videos for all tasks appear at (Bobadilla, Martinez, Gobst, Gossman, & LaValle, 2011a).

The SERB robot is used in performance evaluation of these tasks (Oomlout, 2013). Also, a Parallax ColorPAL sensor is added to the newly designed bumper system so that both physical and virtual walls can be detected with a simple attachment.

Patrolling

For this task, a set of robots (agents) must visit all of the regions in $\mathcal{R}$ repeatedly, in some specified order. In this case, we compute any cyclic path, and then assign a color to every edge, which corresponds to a virtual gate. Colors may be reused, provided that no two colors appear along the boundary of the same region. The minimum number of colors is the maximum degree of the region graph (Fig. 3.9).

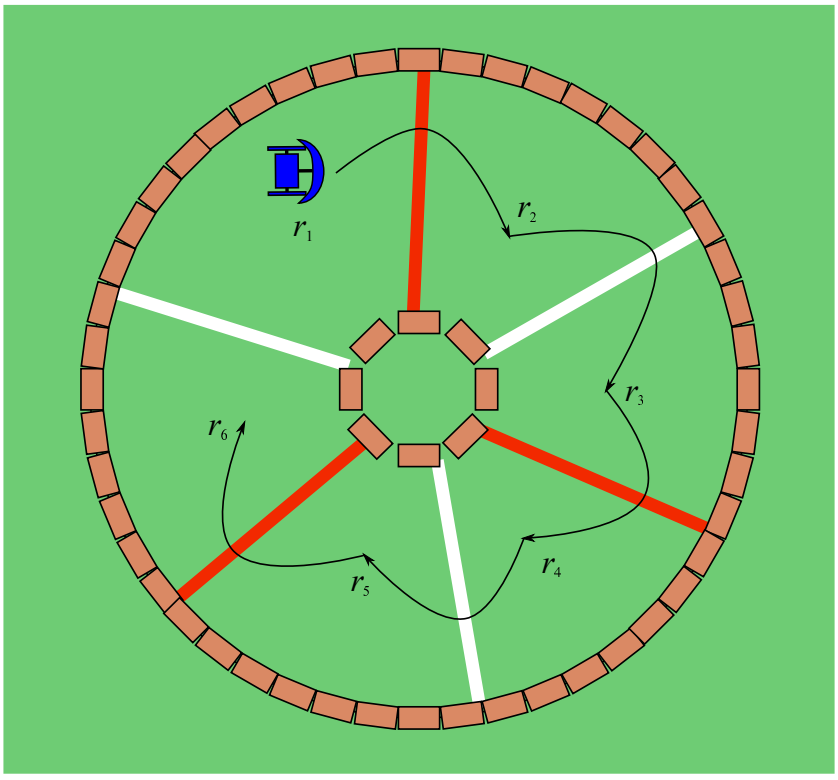


Figure 3.9: Cyclic path for patrolling task.

Robots must visit all regions in order: $r_1, r_2, r_3, r_4, r_5, r_6, r_1, \ldots$

Fig. 3.5 shows an example environment in which two colors are sufficient, resulting in $C = \{\text{RED}, \text{WHITE}\}$. These are the only observation events, leading to $Y = C$. There

are two control modes: $U = \{u_0, u_1\}$. The first, u_0, allows the robot to cross a white gate, $u_0(\text{WHITE}) = 0$ but it treats red as a wall $u_0(\text{RED}) = 1$. The second, u_1, has the opposite effect: $u_1(\text{WHITE}) = 1$ and $u_1(\text{RED}) = 0$. To achieve patrolling, we design a small information space $\mathcal{I} = \{\iota_0, \iota_1\}$. The control policy π is defined as $u_i = \pi(\iota_i)$ for $i \in \{0, 1\}$.

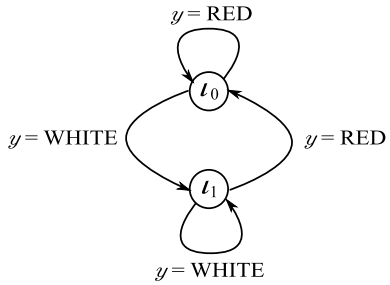


Figure 3.10: Filter used for patrolling

The filter ϕ switches information states when a color is crossed as follows: $\iota_1 = \phi(\iota_0, \text{WHITE})$ and $\iota_0 = \phi(\iota_1, \text{RED})$ (Fig. 3.10). When bouncing occurs from a virtual gate, it is assumed that no observation event occurs because the robot does not pass through the gate.

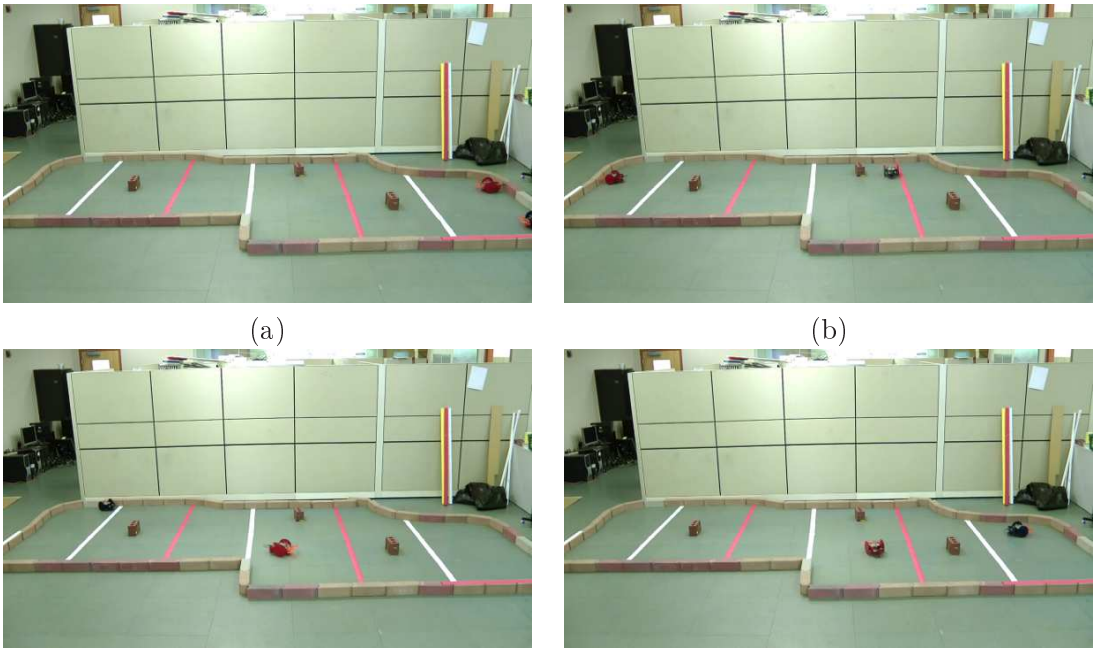
Depending on the initial region $r \in \mathcal{R}$ and initial information state $\iota \in \mathcal{I}$, a robot that executes π will indefinitely patrol the 6 regions in clockwise or counterclockwise order. Suppose that the initial state of the robot is ι_0 which makes it go through a white gate. When it crosses the white gate, it will transition to ι_1. This will make the white gate into a virtual wall forcing it to remain in the new region until the red gate is crossed. Watch video *Patrolling* in (Bobadilla, Martinez, et al., 2011a) in which 4 robots execute the same policy π, but with different initial regions and information states to induce various directions of patrolling.

Separating into teams

For another task, suppose we have two *teams* of robots that are initially in one region and we would like to separate them into one team per region. To solve this problem, we use the same filter and control law described in the previous subsection. It is required that members of the same team have the same initial information state. This task is implemented with four robots belonging to two different teams, blue and not blue. Watch video *Separating into teams* in (Bobadilla, Martinez, et al., 2011a).

Navigation

It is wanted a group of robots to navigate in an environment containing alternating colored gates. The goal is for the robots to move from one end region to another. The only additional information, with respect to the previous examples, is that the robot must choose an information state ι_0 or ι_1 depending on which virtual gate is crossed first. Watch video *Navigation* in (Bobadilla, Martinez, et al., 2011a).



(a) (b)

(c) (d)

Figure 3.11: Continuous navigation of the environment.

a) Two robots begin together in the right region; b) after 18 seconds, the blue robot has advanced two regions, while the red robot has advanced five regions reaching the left edge of the environment; c) after 54 seconds, the blue robot has reached the left edge of the environment, the robot red is back to the right; (d) after 100 seconds, the blue robot goes to the right and is about to ending a navigation cycle, the red robot is again approaching the left edge of the environment.

Continuous navigation

Another navigation task, a bit more complex than the previous, is moving back and forth between both end regions of the environment, as illustrated in the Fig. 3.11. The only modification that has to be made, with respect to the simple navigation, is to add information about the number of regions that the robot will have to visit.

An implementation video is shown in (Bobadilla, Martinez, Gobst, Gossman, & LaValle, 2011b).

Reactive tasks

It would also be liked to give the robots the ability to change their policies based on information collected from external environment conditions that appear during run time. This can be easily incorporated by adding simple sensors. For example, inexpensive photo diodes (for under $2 US each) onto the robots that can detect if a light is turned on in a given region. The observation event space includes LIGHTTODARK and DARKTOLIGHT.

Simple rules can be used, for example: If the light is on, go back to the previous region and if it is off, transition to a new region. Watch video *Reactive tasks* in (Bobadilla, Martinez, et al., 2011a) in which a robot makes decisions about its control modes based on lighting (Fig. 3.12).

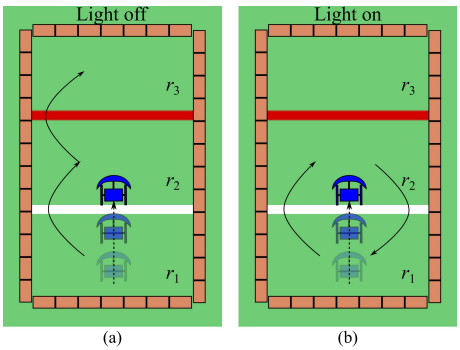


Figure 3.12: Reactive task: Light.

(a) No matter where the robot is, if the light is off then the robot must go to region 3. (b) No matter where the robot is, if the light is on then the robot must go to region 1.

Time-based policies

Suppose we are interested in visiting a region for at least t seconds. We can use this information along with gate crossing information. In this example, we use two more information states ι_2 and ι_3 and additional control modes u_2 and u_3 that make the robot treat both colors as a wall, $u_2(\text{WHITE}) = u_2(\text{RED}) = u_2(\text{WHITE}) = u_2(\text{RED}) = 1$. Also, $u_2 = \pi(\iota_2)$ and $u_3 = \pi(\iota_3)$

The filter is illustrated in Fig. 3.13. It is initialized to information state ι_2 with both gates closed. After a predetermined period of time $t > 0$ has passed, the filter switches to ι_0, which allows the robot to go through the WHITE gate. Once the robot crosses the white gate, receiving WHITE, the filter switches to ι_3, waiting until receiving TIMEOUT before transitioning to ι_1. Watch video *Time-based policies* in (Bobadilla, Martinez, et al., 2011a).

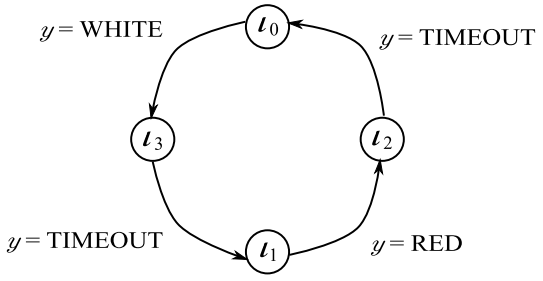


Figure 3.13: A simple filter used for time based patrolling.

Communication based strategies

Until now, the robots act independently to accomplish tasks using only local information of the environment. It is now used a approach that requires only local communication (only robots in the same region are allowed to transmit messages) and low bandwidth (operating with little information). Suppose that it is wanted a group of robots, initially located in the same region to visit a sequence of gates $\tilde{\gamma}$ with the constraint that no robot may advance to the next region until all have crossed the previous gate. The sequence of gates is a way to design a navigation path along different continuous regions (Fig. 3.9). This model uses the communication events $y = (c, i)$ mentioned in Section 3.1.2.

Initially all robots apply a control mode u_0 that guides them through the first virtual gate $\tilde{\gamma}[0]$. Once robot i crosses the gate, its filter will be in an information state for which all gates are blocked $u_0'(c) = 1$ for all $c \in C$, and it will broadcast the message $y = (\kappa(\tilde{\gamma}[0]), i)$ to all other robots in the region. When it receives $y = (\kappa(\tilde{\gamma}[0]), j)$ for $j \neq i$, the robot concludes that the whole group is in the same region and it will be allowed to move forward to the next region. This procedure continues for all gates in $\tilde{\gamma}$. Note that all robots run the same policy but their information states may differ at various points during execution.

The local communication is implemented on the robots by adding an inexpensive 2.4GHz XBee module (under $25 US) to communicate crossing information, encoded as integers. Watch video *Communication based strategies* in (Bobadilla, Martinez, et al., 2011a) where two robots navigate jointly through a sequence of five regions.

3.2 Using the algorithm

The use of the algorithm for solving a problem requires two processes:

- Design of the structure of the system.
- Adjustment process and performance evaluation.

For the system design, the structure should consider the following elements:

1. Formulation of the problem as a search task.
2. Agent definition, its characteristics (hardware, software, structure, mobility, type of interaction and communication).
3. Definition of the environment in which agents interact.
4. Definition of the set of behaviors and requirements for each activation.
5. Definition of the way these behaviors are represented in the agent artificial genotype. That is, how the genotype maps to the phenotype.
6. Definition of the population size, and the quorum threshold T.

7. Definition of a performance metric to evaluate the development of the assigned task to the group of agents.

Most engineering problems can be formulated as search tasks. The solution must meet certain performance criteria under certain operating conditions. The agents can be designed to interpret these conditions as part of its strategy of navigation.

For example, into a problem of design of pieces, the design of a heat sink, for example, the agents must be able to move according to readings of variables such as heat transfer coefficient, size, cost of the part, entropy, physical layout and even manufacturing limitations. For signal processing, for example, analysis of harmonic content of a voltage or current signal, the variables could be the value of periodic samples and maximum values or peaks.

The navigation environment is comprised of the different states of the search space. Each state is defined by a set of specific values of all variables of interest. The agent should be able to navigate this environment, that is, should be able to interpret the value of these variables at each point of the navigation environment.

Having identified the variables and the environment, it is possible to design the sensors and actuators of the agent.

The design of the system structure can be complex, but is considerably simplified by identifying the variables of interest. These variables are the same as allow to evaluate the system performance. The definition of behaviors is simpler, it is most convenient to think of a clustering process of three stages as shown in the following sections (exploration, recruitment and virulence).

3.3 Navigation using local communication between agents

The wild motion is a dynamic feature that allows to solve basic navigation problems, but tends to make "intelligent" the environment, which is responsible for guiding the "silly" agents. The scheme proposed by the author based on QS attempts to design navigation strategies on the agents according to different behaviors, behaviors that are activated in each agent according to its particular local interaction. To accomplish this, the author will in this section extend the theoretical framework of the previous section adding to the structure the interaction model based on QS.

The navigation task that we will design in this section corresponds to a particular case of grouping of agents, similar to that shown at the end of the previous chapter. Specifically, the author will analyze the problem of clustering by areas of agents under the same restrictive conditions of previous designs: without requiring system identification, geometric map building, localization, or state estimation. The movement of the agent in the environment, i.e., the position update equation for virulent bacteria, receives in this section the most interest. To define this movement, The author proposes the use of a gradient in the environment, constructed from signals radiated by the agent (here an audio signal). With this, The author attempts to generalize the case of the video signal taken as an example at the end of the previous chapter, to the general case of any kind of signal radiated by the agent (Fig. 3.14).

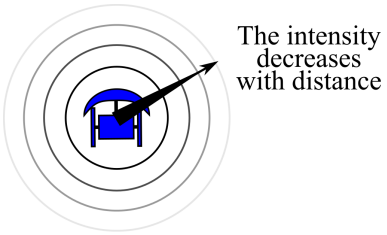


Figure 3.14: Signal radiated by the agent.

The signal is transmitted by each agent. The intensity of the signal at a point of the environment depends on the sum of contributions of each agent.

The transmission may use a video signal, audio, or any other feasible in each of the agents to build the gradient. In any case, it is clear that this gradient has a dynamic behavior along the time. That is, the gradient is continuously modified by all the robots in the form of local communication. This feature reinforces the system action strategy by consensus of its members.

The design scheme, both the algorithm and its implementation on the robot, looks for a minimalist approach, in which it minimizes the requirements of the robot (processing power, communication and type of sensors). Besides, the research seeks the autonomous navigation of each robot, and scaling the system to any number of agents.

The navigation algorithm is formulated for a task of grouping, where the robots form autonomously groups without any external interaction or prior information of the environment or other robots. At the end of the section, task performance is verified through simulation for the laboratory prototypes.

3.3.1 Background

In general, the interest focuses on analyzing the collective navigation problem in environments that limits the use of sensors, such as collapsed environments in a natural disaster. Guiding questions of our approach are: Can a group of robots autonomously navigate an unknown dynamic environment without collecting information about it? What is the minimum information necessary for an agent to solve the problem of navigation? Seeking answers to these questions, different navigation strategies are being investigated, in which we reduce as much as possible the collection of information by the agent, and consequently, the requirements of information processing and communication.

In this sense, the author proposes a collective navigation strategy for a swarm robots in which robots are oriented in the environment according to the signal that they emit. The intensity of this signal allows the robot to locate their mates, and therefore to group together.

3.3.2 Mathematical Model

Let $\mathcal{W} \subset \mathbb{R}^2$ be the closure of a contractible open set in the plane that has a connected open interior with obstacles that represent inaccessible regions. Let $\mathcal{O}$ be a set of obstacles, in which each $O \subset \mathcal{O}$ is closed with a connected piecewise-analytic boundary that is finite in length. Furthermore, the obstacles in $\mathcal{O}$ are pairwise-disjoint (their intersection is the empty set) and countably finite in number. Let $E \subset \mathcal{W}$ be the free space in the environment, which is the open subset of $\mathcal{W}$ with the obstacles removed (Fig. 3.15).

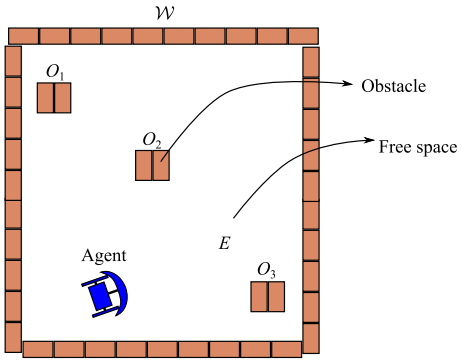


Figure 3.15: Environment description.

Let us assume a set of n agents in this free space. The agents know the environment E in which they move from observations, using sensors. These observations allow them to build an information space $\mathcal{I}$. An information mapping is of the form:

$$g: E \longrightarrow Y \tag{3.2}$$

where Y denotes an *observation space*, constructed from sensor readings over time, i.e., through an observation history of the form:

$$\tilde{y} : [0, t] \longrightarrow Y \tag{3.3}$$

The interpretation of this information space, i.e., $\mathcal{I} \times Y \longrightarrow \mathcal{I}$, is that which allows the agent to make decisions (LaValle, 2006).

We consider a group of n agents, differential wheeled robot, whose kinematics is defined by:

$$\begin{aligned} \dot{\boldsymbol{X}}_{ai} &= u_i \\ \boldsymbol{X}_{ai} &= (x_{1i}, x_{2i}, \alpha_i) \end{aligned} \tag{3.4}$$

with:

$$\begin{aligned} \dot{x}_{1i} &= v_i \, cos\, \alpha_i \\ \dot{x}_{2i} &= v_i \, sin\, \alpha_i \\ \dot{\alpha}_i & \end{aligned} \tag{3.5}$$

where $(x_{1i}, x_{2i}) \in \mathbb{R}^2$ is the position of the i^{th} robot, α_i is its heading, and v_i and $\dot{\alpha}_i$ are the controlled translational and rotational velocities, respectively. u_i denotes the agent's control input (Fig. 3.16).

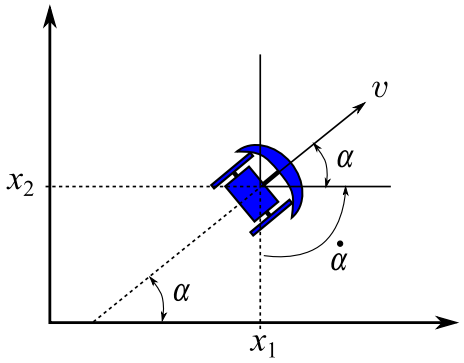


Figure 3.16: Continuous variables of the agent.

The interpretation of the information space is made by each agent according to control policies encoded in its genome. Each agent is modeled as a hybrid system, equations 3.4 and 3.5 characterize the dynamics of the continuous variables, and the control policies define the behavior of the agent according to local observations.

The discrete transition system D_1 simulates the original hybrid system. Let the state space of the discrete system be L_a. The transition system is defined as:

$$D_1 = (L_a, l_1, \rightarrow_2), \tag{3.6}$$

in which l_1 is the behavior initially running by the agent. The transition relation $l_1 \rightarrow_2$ is true if and only if the signal molecule (signals emitted by other agents or landmarks in the environment) is read and interpreted by the agent.

It is assumed the agents are able to sense the proximity of their team mates and/or obstacles within the environment, using minimal information. Therefore, the neighborhood of $\boldsymbol{X}_{ai}$ is given by the range and field of view of the sensing hardware.

All robots broadcast a signal, which is modeled as an intensity function over $\mathbb{R}^2$. Let m denote the signal mapping $m : \mathbb{R}^2 \longrightarrow [0, 1]$, in which $m(p)$ yields the intensity at $p \in E$. Since all the robots broadcast their own signal while moving in the environment, the spectrum or map navigation is always dynamic.

The environment E and even the signal mapping m are unknown to the robot. Furthermore, the robot does not even know its own position and orientation.

The goal is to design the control policies for the n robots in order to independently solve navigation tasks in a dynamic and unknown environment.

3.3.3 Navigation algorithm

Control strategy

The control strategy is supported by the intention to develop the simplest possible system, but able to collectively solve complex navigation tasks. The main feature that is wanted is robustness in real applications (the task must be met regardless of some of the agents fail).

With this approach, the strategy not requires determining the coordinates of the robots, their mathematical models or a central control unit. Instead, the strategy proposes that the robots navigate according to the intensity of a signal in the environment, an intensity gradient in the environment that they built themselves.

In principle, all the robots in the environment transmit a given signal. These signals, which alter the environment, establish a local communication between robots. The signals are radiated, centered on each of the robots, and lose intensity with distance from the robot. For analysis, it is assumed that all signals are identical, but the more general case allows differences.

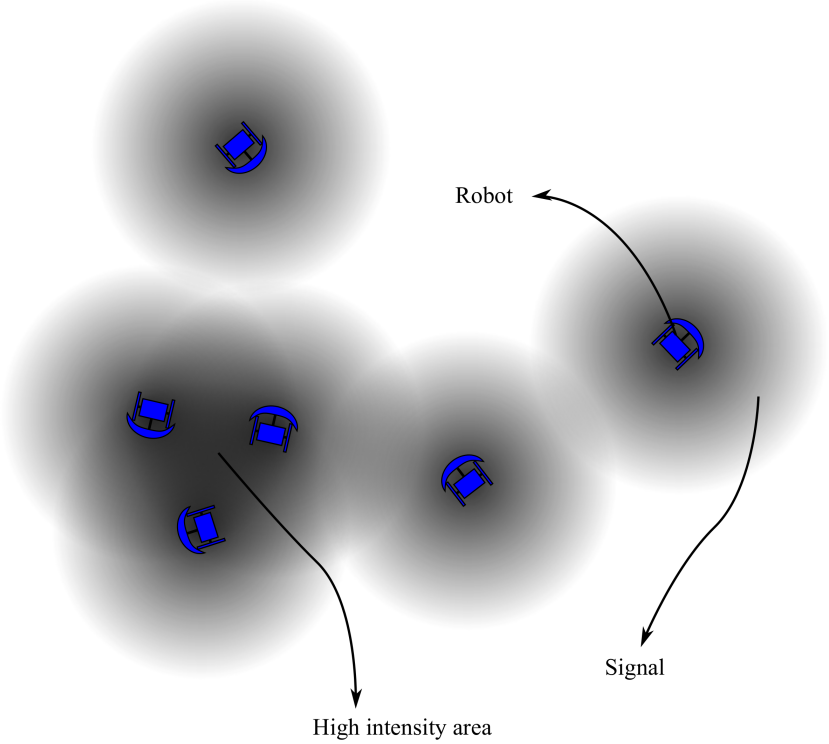


Figure 3.17: Robots and their signals.

The dark areas represent greater intensity.

It is considered two kinds of sensors. A first contact sensor (CS), which reports when the robot is in contact with the environment boundary ∂E:

$$h_{CS_i}(x_{1i},\, x_{2i}) = \begin{cases} 1 & \text{if } (x_{1i}, x_{2i}) \in \partial E \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

And the intensity sensor (IS), that indicates the strength of the signal:

$$h_{IS_i}(x_{1i},\, x_{2i}) = h(x_{1i},\, x_{2i},\, \alpha_i,\, E,\, m) \in [0, \quad 1] \tag{3.8}$$

The intensity values are normalized in the interval [0, 1], where 1 corresponds to the maximum intensity value. The robots must move to find the highs in the intensity map. Two intensity sensors in a robot are capable of providing information relating to the gradient of intensity map:

$$\nabla m_i\,(p_2 - p_1) \tag{3.9}$$

where p_1 and p_2 are the locations in $\mathbb{R}^2$ of the sensors on the robot i (Fig. 3.18).

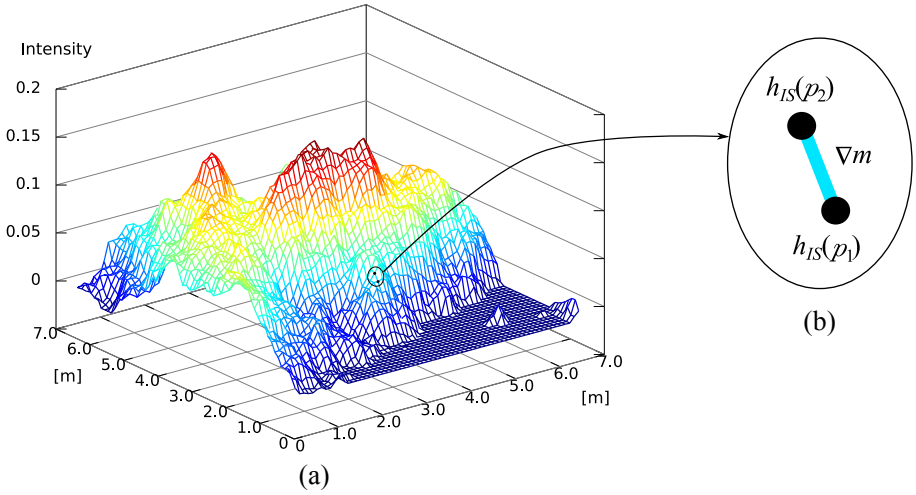


Figure 3.18: Intensity gradient in an environment.

Intensity gradient simulated for an environment of 7 m × 7 m. a) Three-dimensional view, the red color identifies the highest values. b) Intensity at points p_1 and p_2 corresponding to the intensity sensors of a robot.

The robots do not require other types of sensors, such as global positioning, odometry, or a compass. Therefore, it is unable to obtain precise position or angular coordinates.

The robots have no differential speed control on their wheels (On-Off control), this reduces the possible actions or motion primitives that are given to move the robot, simplifying the control and modeling. Furthermore, this allows some slack in the design of the robots; i.e., the scheme is robust regardless of variations in implementation (it is very difficult to build identical robots, and always to keep them in that way). The robots are allowed only four motion primitives (Fig. 2.21):

- $u_{fwd} \longrightarrow$ Forward: The robot goes straight forward in the direction it is facing. Both wheels rotate at their rated positive speed.
- $u_{bwd} \longrightarrow$ Backward: The robot goes backward in the opposite direction it is facing. Both wheels rotate at their rated negative speed.
- $u_{rgt} \longrightarrow$ Right: The robot rotates clockwise, turns right. The left wheel runs at rated positive speed, and the right wheel runs at rated negative speed.
- $u_{lft} \longrightarrow$ Left: The robot rotates counterclockwise, turns left. The left wheel runs at rated negative speed, and the right wheel runs at rated positive speed.

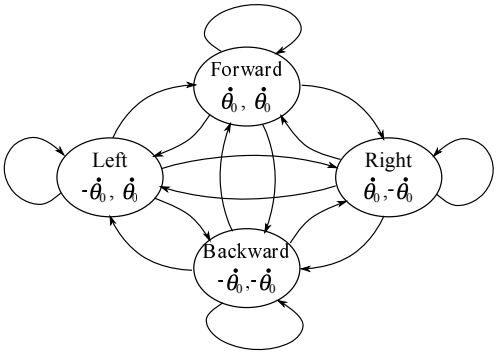


Figure 3.19: State diagram corresponding to the robot's movement.

The construction of the diagram is made from the point of view of movement of the motors.

Navigation plan

The intensity robot sensors provide information proportional to the gradient of intensity map. This information is sufficient to define the movement of the wheels of the robot, i.e., the motion primitive to be applied at any given time.

Let us consider our testing task: We want to group the robots at some arbitrary point in E. To solve this task, the robots must navigate with the steepest ascent of m. The equation 3.9 enables the robot to determine the intensity of m in the points p_1 and p_2 in which the sensors are located. Applying rotation primitives, the robot is able to align it self with the position that minimizes the difference. Then, using primitives for forward and backward, the robot can determine the direction in which the gradient of m increases. A possible navigation plan that solves this task is shown in Table 3.1.

The structure of the system, where each agent has a minimal hardware, makes difficult the accurate reading of the quorum threshold T by robots. The value of T for the activation of QS is estimated by each robot according to the sensed intensity. However, due to the limited hardware, it is not possible to discriminate with precision the number of robots. Neither is it possible to define high values of T (assign high value of T is equivalent to not include QS, because the threshold is never reached). However, it is possible to identify small groups of 3-5 robots.

Table 3.1: Plan for the grouping task

1. If $h_{CS} = 1, \Rightarrow$ apply u_{bwd}, and then apply u_{lft} a random angle. Do $I_{avg} = 0$ (average intensity).
2. Read sensors. Determine $I_{p1} = h_{IS}(p_1)$ and $I_{p2} = h_{IS}(p_2)$.
3. Determine the orientation of m by calculating the difference of intensities $I_{p2} - I_{p1}$.
4. If
   (a) $(I_{p2} - I_{p1}) > 0 \Rightarrow$ Apply u_{rgt}. Go to step 2.
   (b) $(I_{p2} - I_{p1}) < 0 \Rightarrow$ Apply u_{lft}. Go to step 2.
   (c) $(I_{p2} - I_{p1}) \approx 0 \Rightarrow$ Go to step 5.
5. Store the average intensity of the current position, I_{avg}.
6. If $T_{estimated}(I_{avg}) > T, \Rightarrow$ maximize the intensity of the signal emitted by the robot (activate QS).
7. Determine the direction of increased intensity calculating the difference of intensities stored $I_{avg}(t) - I_{avg}(t-1)$.
8. If
   (a) $(I_{pm}(t) - I_{pm}(t-1)) \geq 0 \Rightarrow$ Apply u_{fwd}. Go to step 1.
   (b) $(I_{pm}(t) - I_{pm}(t-1)) < 0 \Rightarrow$ Apply u_{rgt} to rotate 180 degrees, then apply u_{fwd}. Go to step 1.

Although the structure does not allow analysis for multiple values of T, it is possible to observe the effect of the QS on the performance of the task.

Convergence of the navigation plan

Before a performance evaluation, it is necessary to demonstrate that the proposed navigation plan is able to solve the task (grouping), i.e., that the algorithm converges. To establish the convergence, the author first shows that following the proposed navigation plan under the described conditions, any robot can navigate around obstacles (along the perimeter) in E following the gradient of intensity. And second, the author shows that if the robot is able to surround the obstacles, then the robot will find the other robots near it in a finite number of steps.

Lemma 3: If E is the free space where the agents can navigate, then all the points $p_j \in \mathbb{R}^2 - \mathcal{O}$ are possible meeting points. For every obstacle boundary ∂O of obstacles $O \subset \mathcal{O}$, and every possible meeting point p_j, there exist at least one intensity local maximum $p_0 \in \partial O$ in the range of the sensor of the agent for which the gradient of the signal constructed by each of the agents is disjoint from the interior of O (they have no element in common, Fig. 3.20).

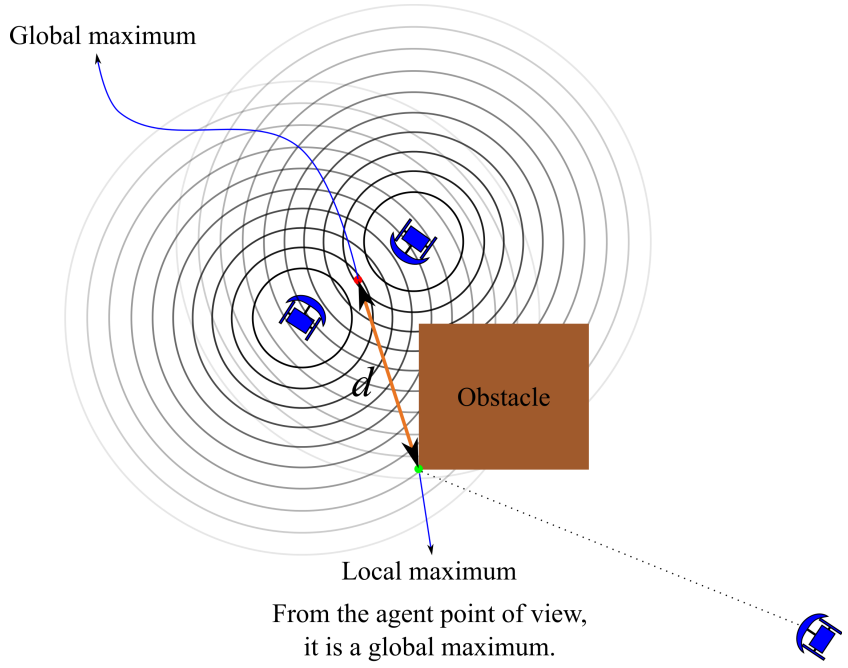


Figure 3.20: No observable global maximum by the agent due to obstacle.

The obstacle prevents that the agent observes the global maximum. However, there is a point in ∂O (local maximum) which is detected by the agent as a global maximum.

Proof: The navigation problem is that the agent can not directly sense the intensity of the point p_j (global maximum) due to the presence of an obstacle (Fig. 3.20). The

lemma states that under these conditions, the agent will always find a local maximum at ∂O. Navigating to this local maximum, over time, the agent will border the obstacle.

If it is assumed that under the conditions proposed in the navigation plan, there are a finite number of local maximums along ∂O, for a meeting point $p_{j,t}$ at time t, one or more of these points along ∂O can be a global maximum for the agent that searches the meeting point. Since the signal intensity increases with decreasing distance to $p_{j,t}$, the global maximums are points of ∂O that are close to $p_{j,t}$. If we call p to any of these points, and we call $d(p_{j,t}, p)$ the shortest distance from one of these points to $p_{j,t}$, then, by construction, no other points in O are closer to $p_{j,t}$ than p. This means that d and the interior of O are disjoint (d is in E). Therefore, if the agent navigates to the point p in ∂O, always achieves avoid the obstacle. ■

According to the above, if the agent follows the proposed navigation plan, the agent will always be able to find a maximum in the intensity gradient in ∂O, and therefore, it will be able to go around. This will be true all the time, regardless of the dynamics of change in the intensity gradient.

Lemma 4: For each global maximum $p_{j,t}$ at time t, if there is a path between the agent i and the meeting point $p_{j,t}$, then the agent i, following the navigation plan, will find the meeting point $p_{j,t}$ after a finite number of motion primitives.

Proof: The navigation plan of the agent moves each wheel to align it with the gradient of intensity. After to align it, the plan forces the agent to move to the point of highest intensity detected by the sensors. These points are local maxima of the intensity gradient. And, according to Lemma 3, it might be stated that the agent always successively come near until find the global maximum of the intensity gradient.

If the agent is able to navigate around obstacles, the navigation plan ensures that after a finite number of motion primitives the agent will be at a point of greatest intensity, i.e., agents who are close be grouped around $p_{j,t}$. ■

3.3.4 Experimental prototype and simulations

For the experiments, we use again the SERB robot (Fig. 3.4). Besides the bumper with impact sensor, we add two microphones as audio sensors. In the spirit of minimalist design, the robot and all its peripherals, including the microphones, is handled by a single 8-bit microcontroller (Atmel ATmega328, 8-bit AVR RISC-based microcontroller).

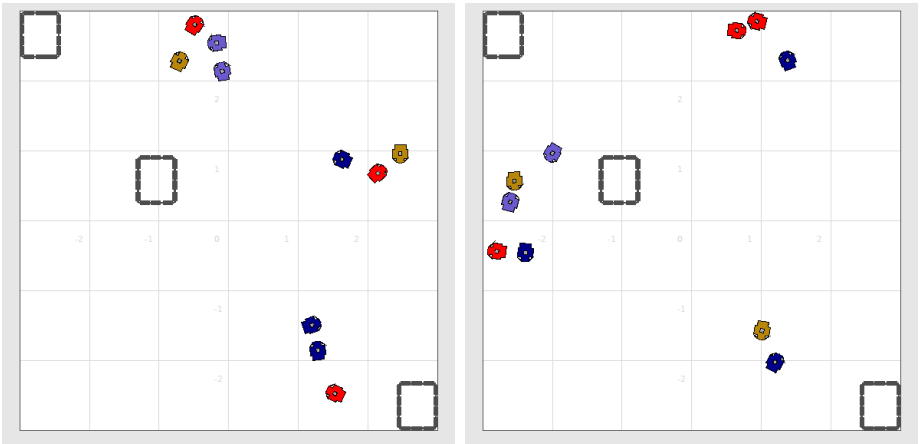
As detailed above, while it is possible to control the translational and rotational speeds of the robot, the scheme chooses to use a software actuator, a high-level control which generates control states in order to facilitate the construction of state diagrams. In this way, each wheel has two choices of motion: clockwise and counterclockwise. Therefore, the robots can move forward, backward, turn on their axis in any direction.

The system is homogeneous, that is, all agents are identical in design. If the task requires more agents, we only have to place them in the environment. Due to the impossibility of build tens of these robots, it is only used the prototype robots (three robots) to observe the basic behaviors of the algorithm. The analysis of the dynamics of the system is done with ten robots through simulation.

The simulations are performed on Player/Stage (Vaughan, 2008; Gerkey et al., 2003), this platform allows for the simulation run exactly the same code developed in C for the real robot. They consider not only the functionality of the algorithm, but also the physical characteristics of the robot, i.e. shape, size (0.20 m x 0.22 m) and weight (0.530 kg). The microphones are installed just above its geometric center.

The simulation environment is designed in square shape with a total area of 6 m $\times$ 6 m $=$ 36 m^2. Inside the environment, there are three holes inaccessible to the robots as obstacles for navigation and sensors. The simulations use a total of 10 agents, all completely independent of each other.

Fig. 3.21 and Fig. 3.22 shows the results of four simulations. The four simulations are performed under the same conditions (ten identical robots randomly distributed in the same environment, all them with the same navigation plan), the only difference among them is the initial position of robots (random initial location). In the first case (Fig. 3.21(a)), the robots form groups in 1 min. and 48 s (00:01:48, time in the simulation). In the second case (Fig. 3.21(b)), the robots form groups in 53 s (00:00:53). In the third case (Fig. 3.22(a)), the robots form groups in 1 min. and 26 s (00:01:26), and the fourth case (Fig. 3.22(b)), the robots form groups in 37 s (00:00:37).

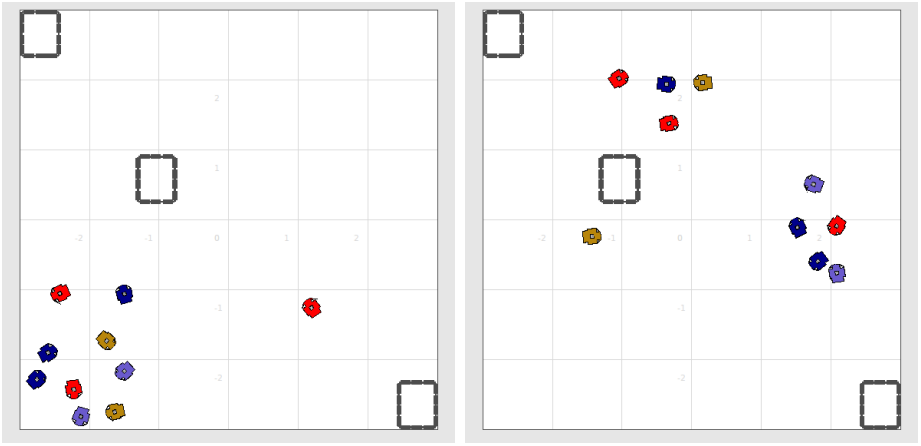


(a) (b)

Figure 3.21: Simulation of the autonomous navigation: Grouping 1.

Robots learn to navigate autonomously using local information, and after a while get together in small groups. (a) First case, the robots are grouped after 1 min. and 48 s (00:01:48). (b) Second case, the robots are grouped after 53 s ((00:00:53)).

In all these cases a value of $T = 20$ is assigned, i.e., the agents never activate their QS (grouping without QS). After 20 simulations under the same conditions, the simulations show an average expected time to perform the task of 1 min. and 10 s (00:01:10). To evaluate the impact of QS, other 20 simulations under the same conditions are performed, but with a value of $T = 3$. While the general behavior is the same, the simulation time is reduced to an average of 47 s (00:00:47), which means a time reduction of about 33.6%. Although it is true that times are dependent on the code (genome), robots and functioning



(a) (b)

Figure 3.22: Simulation of the autonomous navigation: Grouping 2.

Robots learn to navigate autonomously using local information, and after a while get together in small groups. (a) Third case, the robots are grouped after 1 min. and 26 s (00:01:26). (b) Fourth case, the robots are grouped after 37 s (00:00:37).

of the entire system (hardware), according to the simulations, it is clear the effect of QS on the performance of the algorithm. An additional effect of QS is that, the dispersion of the times is decreased.

These data are summarized and analyzed using box and whisker plots (Fig. 3.23). The summary statistics used to create a box and whisker plot are the median of the data, the lower and upper quartiles (25% and 75%) and the minimum and maximum values.

As in most of the cases observed in Player/Stage (62%), the macro-scale model is developed for two grouping areas. These two areas have similar performance, the meeting point depends on the position of the agents, not of the environment. According to the conditions of the system, for the initial choice of the areas there is no favoritism criteria, i.e.:

$$\mu_1 = \mu_2 = \mu \tag{3.10}$$

The process of migration between the two areas depends on the number of agents in each area. That is, the values of λ_1, λ_2, ρ_{12}, ρ_{21}, k_1 and k_2 depend (are functions) on X_1, X_2, Z_1 and Z_2. To simplify the analysis, these values are defined as:

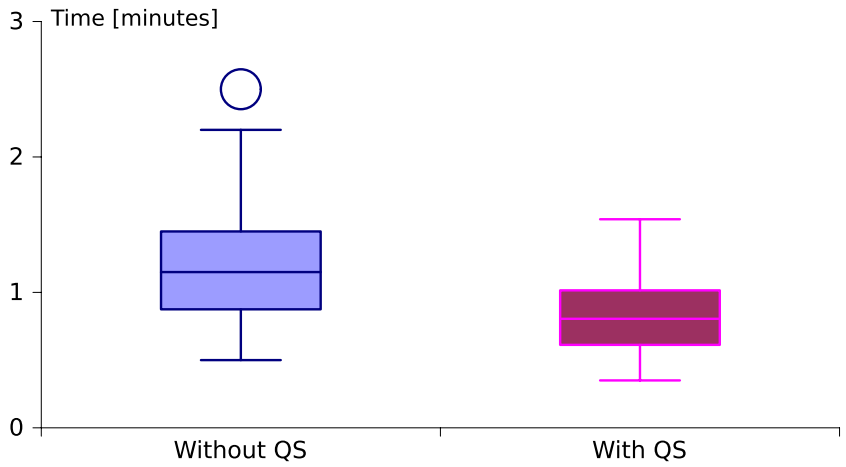


Figure 3.23: Effect of QS on the navigation total time.

Box and whisker plots of the navigation total time of 10 agents with random initial position. 20 simulations are performed for each of the two cases: Without QS (left in blue) and with QS (right in red). The central box represents the values from the lower to upper quartile (25th to 75th percentile). Excluded (and shown) outliers. The middle line represents the median. A line extends from the minimum to the maximum value.

$$\begin{aligned} \mu_1 &= \mu_2 = 1 \\ \lambda_1 &= X_1 \\ \lambda_2 &= X_2 \\ \rho_{21} &= X_1 + Z_1 \\ \rho_{12} &= X_2 + Z_2 \\ k_1 &= Z_1 + 1 \\ k_2 &= Z_2 + 1 \end{aligned} \tag{3.11}$$

Fig. 3.24 shows the flow chart for this system. The system of equations can be written as 3.12.

$$\begin{aligned} \dot{R} &= -2\mu R \\ \dot{X}_1 &= \mu R + \lambda_1 Z_1 \Theta(R)\,\Theta(T - Z_1) - \rho_{12} X_1 + \rho_{21} X_2 - k_1 X_1 \\ \dot{X}_2 &= \mu R + \lambda_2 Z_2 \Theta(R)\,\Theta(T - Z_2) + \rho_{12} X_1 - \rho_{21} X_2 - k_2 X_2 \\ \dot{Z}_1 &= k_1 X_1 - \rho_{12} Z_1 + \rho_{21} Z_2 - \lambda_1 Z_1 \theta(R)\,\Theta(T - Z_1) \\ \dot{Z}_2 &= k_2 X_2 + \rho_{12} Z_1 - \rho_{21} Z_2 - \lambda_2 Z_2 \theta(R)\,\Theta(T - Z_2) \end{aligned} \tag{3.12}$$

Fig. 3.25, 3.26 and 3.27 show the simulation of macro-scale model for different cases. Fig. 3.25 shows the case already discussed, where the system has 10 agents, a threshold of $T = 20$, and the agents are grouped into two areas of equal performance without triggering its QS. Fig. 3.26 shows the same case (10 agents and two areas of equal performance), but with a threshold of $T = 3$; in this case, the QS is activated ($t = 3$ min), which greatly increases the grouping speed (reduction in total time of 20%). The times in these curves

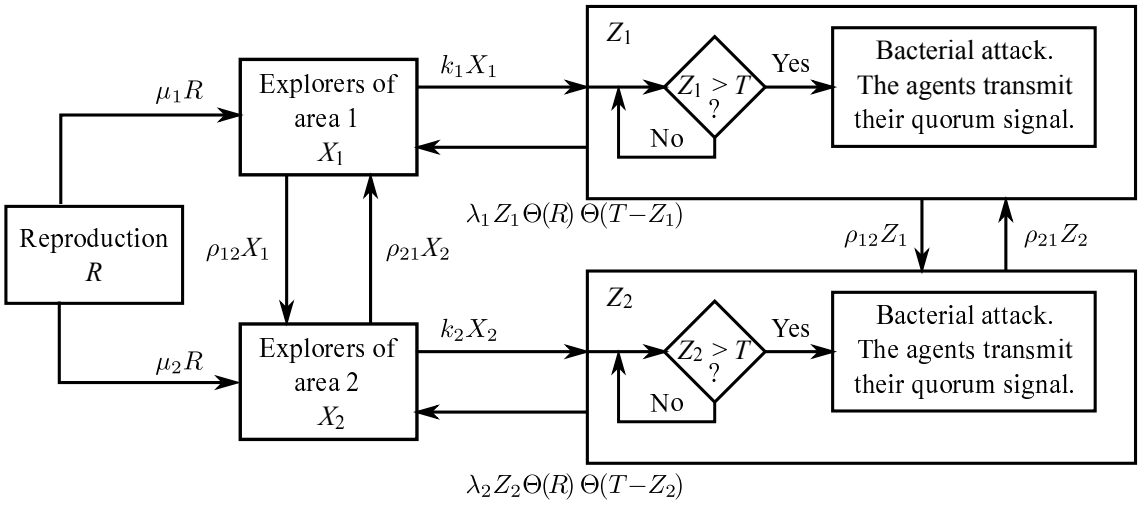


Figure 3.24: Flow chart for grouping in two areas.

do not match the times at Player/Stage due to how the constants ware defined in equation 3.11. These constants must also consider the response speed of the agents.

Fig. 3.27 shows a generalization of the behavior for 50 agents and a threshold $T = 10$. The figure shows again as the convergence process is accelerated when the system activates the QS ($t = 10$ min).

Finally, additional experiments allow to show that the algorithm converges even when agents are not completely uniform (Fig. 3.28). Different robots of similar size, transmitting a similar but different signal, also are able to group together when they follow the navigation plan. This means that the task is done regardless of whether some robots fail, or if its dynamics differs from the original model (robustness).

3.4 Designing navigation paths

The navigation strategy detailed in the previous section, allows bacteria with virulent behavior locate their neighbors, and remain close to them. However, these bacteria do not follow any path, and the final destination is unknown. In this section, the author generalizes the formulation with the intention to describe a formal navigation strategy for a desired path.

There are generally two approaches to solve navigation problems: On one hand the traditional choice in control, including system identification, geometric map building, localization, and state estimation. With these analytical tools, the location of the agent and the destination are know, and the control unit knows how the agent moves, and therefore it can exactly instruct the agent on how to move to the destination. On the other hand, there is the traditional choice in nature, in which the agents move according to information and interaction in the environment, for example motivated by food, light, temperature or other agents.

In the nature approach, which fits the proposed bacterial model, the information and

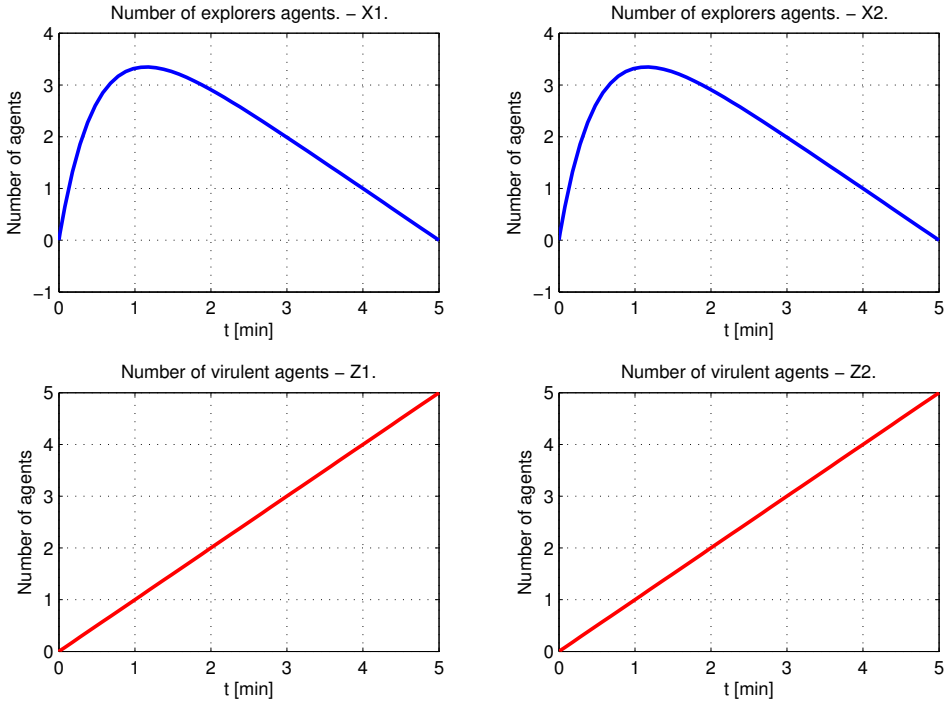


Figure 3.25: Macro model simulation results for a grouping task (10 agents, $T = 20$) - Without QS.

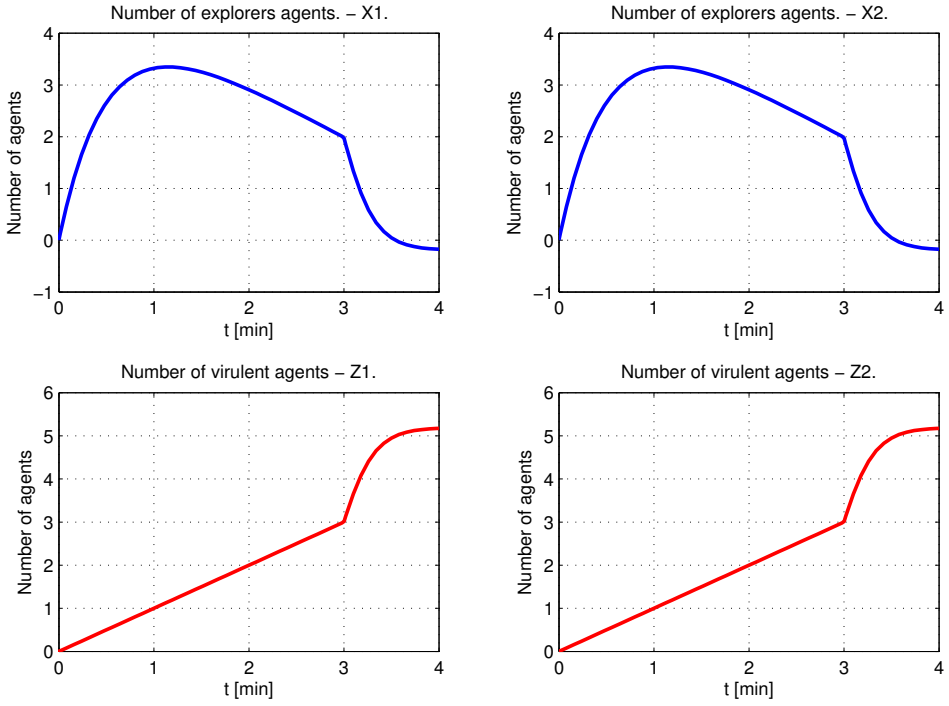


Figure 3.26: Macro model simulation results for a grouping task (10 agents, $T = 3$) - With QS.

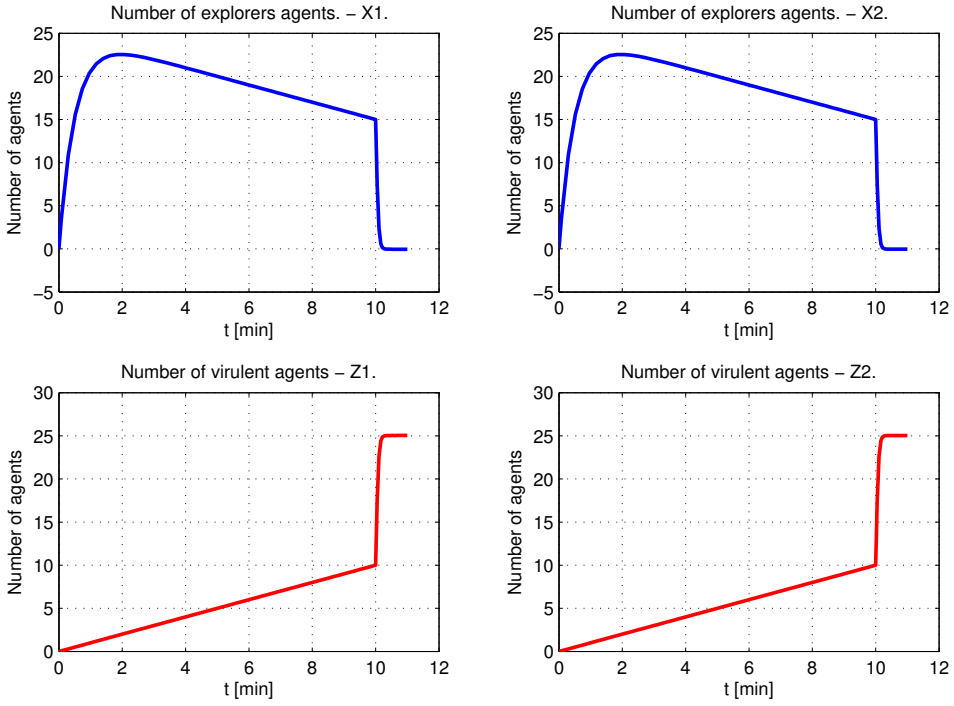


Figure 3.27: Macro model simulation results for a grouping task (50 agents, $T = 10$) - With QS.

Figure 3.28: Experiment with nonuniform agents.

Experiments with systems formed by agents with different shapes, sizes, speeds, sensors and signal intensity showed that the system converges. The robots were designed and constructed by students of Electrical Technology at the District University Francisco José de Caldas.

interaction in the environment can be modeled as landmarks. The biological bacteria, according to their species, move in a given environment towards areas of greater concentration of food (Guzmán, Delgado, & De Carvalho, 2010). Similarly, when a human being is in a mall, he seeks the things of interest examining display cases and looking access doors. In each of these cases, the items are recognizable landmarks for agents.

Returning to the bacterial model, the navigation problem is related with the motion of the agent in the environment. Therefore, what is the best model for updating the motion? The author has answered this question using a movement model that resembles the movement of particles suspended in a fluid (Toth, Banky, & Grolmusz, 2011).

3.4.1 Background

The proposed navigation strategy aims to ensure that the swarm of robots travel a route established according to the projection of the expected behavior of the robots on a speed field designed on the environment. Navigation is done according to Brownian motion (Einstein, 1905), but with the particles of the environment fixed as landmarks, and whose pushing force (signal strength) is designed according to the desired behavior.

This part of the research focuses on analyzing the navigation algorithm, the movement policies of the robots and the design of the speed field on the environment, maintaining the premises of minimalism, autonomy, and scalability. One question that guided us throughout the entire design process was: Does the robot can navigate the environment without collecting information?

3.4.2 Mathematical Model

As before, $\mathcal{W} \subset \mathbb{R}^2$ is the closure of a contractible open set in the plane that has a connected open interior with obstacles $O \subset \mathcal{O}$ that represent inaccessible regions. $\mathcal{O}$ is the set of obstacles. The obstacles in $\mathcal{O}$ are pairwise-disjoint and countably finite in number. $E \subset \mathcal{W}$ is the free space in the environment, which is the open subset of $\mathcal{W}$ with the obstacles removed (Fig. 3.15).

A set of n agents can navigate in the free space E. The agents know the environment E from observations, using sensors. These observations allow them to build an information space $\mathcal{I}$ (Equation 3.2, information mapping), where Y denotes an *observation space*, constructed from sensor readings over time, i.e., through an observation history (Equation 3.3). The interpretation of this information space, i.e., $\mathcal{I} \times Y \longrightarrow \mathcal{I}$, is what allows the agent to make decisions (LaValle, 2006).

Again, the mathematical model of these agents is not important for the control strategy. However, for analysis purposes, these agents are discussed as differential wheeled robots, whose kinematics is defined by Equation 3.4 and 3.5, where $(x_{1i}, x_{2i}) \in \mathbb{R}^2$ is the position of the i^{th} robot, α_i is its heading, and v_i and $\dot{\alpha}_i$ are the controlled translational and rotational velocities, respectively. u_i denote the agent's control input.

As expected, again the agents are modeled as hybrid systems. Equations 3.4 and 3.5 characterize the dynamics of the continuous variables, and the control policies define the behavior of the agent according to local observations. The discrete transition system D_1 simulates the original hybrid system. Let the state space of the discrete system be L_a. The transition system is defined as:

$$D_1 = (L_a, l_1, \rightarrow_2), \tag{3.13}$$

in which l_1 is the behavior initially running by the agent. The transition relation $l_1 \rightarrow_2$ is true if and only if the signal molecule (signals emitted by other agents or landmarks in the environment) is read and interpreted by the agent.

Now, for navigation, the environment has a total of k landmarks. All landmarks broadcast a signal. However, the value of the signal for each landmark is dependent on the navigational path designed for robots. The signal now is modeled as an intensity function over $\mathbb{R}^2$ (in the former case, the gradient map was dynamic and shaped by the contributions of each agent, now the map is static and designed for navigation of agents). Let m denote the signal mapping $m : \mathbb{R}^2 \longrightarrow [0,\ I_{max}]$, in which $m(p)$ yields the intensity at $p \in E$. We assume that the robot's sensing range is a small circle in $\mathbb{R}^2$ whose radius r is on the order of the robot's typical displacement.

The dynamics of the system is coordinated again by local interactions. The environment E and the signal mapping m are unknown to the agent. The robot also does not know its own position and orientation.

3.4.3 Navigation algorithm

Robot movement strategy

The navigation strategy that the author proposes intended to duplicate the movement of particles suspended in a fluid, due to the continuous bombardment of atoms and molecules. This phenomenon is known as *Brownian Motion*, a name also given to the mathematical models used to describe it (Toth et al., 2011; Sayed & Sayed, 2011).

In the mathematical description of Brownian motion developed by Albert Einstein (Einstein, 1905), the physicist made a diffusion equation for Brownian particles in which the diffusion coefficient, or mass diffusivity D_C, is related to the mean square displacement of the particle. If $\beta(x,t)$ is the Brownian particle density in the point x at time t, then β satisfies the diffusion equation:

$$\frac{\partial \beta}{\partial t} = D_C \frac{\partial^2 \beta}{\partial x^2} \tag{3.14}$$

Following the moments given by the solution of equation 3.14, and assuming this behavior model for the robots, the movement of the robots in small time intervals $\triangle t$, defined by its own dynamics (response speed), can be determined as:

$$x_i(t+\triangle t) = x_i(t) + \mathbf{v}(x_i)\triangle t + \sqrt{2D_C \triangle t} \tag{3.15}$$

where x_i is the kinematics of robot i, and $\mathbf{v}(x_i)$ is the value of the speed field sensed by the robot i at position x_i. As shown in equation 3.15, the displacement of the robot is determined by two terms, one which depends on the intensity of the speed field in the environment $\mathbf{v}$, and another which depends on the diffusion coefficient D_C. From the viewpoint of the model based on QS, this coefficient is a constant that represents the ease with which the robot moves in the environment (it relates to the speed of response of the robot).

The robots are basically event-driven agents. They move continuously changing direction randomly when hitting an obstacle. Their forward direction is adjusted according to the speed field, which is indicated by specially designed landmarks. These landmarks in our model emulates the behavior of the molecules in the fluid, but unlike many real systems, we do not want to fill the environment with them; the installation of a few is enough to coordinate the movement of the robots.

Each landmark encoding an intensity value $\mathbf{v}$ corresponding to the value of the speed field in the point p_x where the landmark is located. In this way, when an agent reaches the point p_x, using its sensors, reads the intensity value encoded in the landmark. This intensity value is a scalar (no encodes information related to the gradient of the speed field).

Once again, we consider two kinds of sensors. A first contact sensor (CS), which reports when the robot is in contact with the environment boundary ∂E:

$$h_{CS_i}(x_{1i},\, x_{2i}) = \begin{cases} 1 & \text{if } (x_{1i}, x_{2i}) \in \partial E \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

And the intensity sensor (IS), that indicates the strength of the signal in the landmark:

$$h_{IS_i}(x_{1i},\, x_{2i}) = h(x_{1i},\, x_{2i},\, E,\, m) \tag{3.17}$$

The intensity values are normalized in the interval [0, 0.2], where 0.2 m/s corresponds to the maximum speed value in the environment. We chose the maximum value of 0.2 m/s for convenience, since this is the speed at which move the laboratory prototypes. The robots do not require other types of sensors, such as global positioning, odometry, or a compass. Therefore, it is unable to obtain precise position or angular coordinates.

As we explained in the previous section, the robots have no differential speed control on its wheels, this reduces the possible actions or motion primitives that are given to move the robot, simplifying the control and modeling. Furthermore, this allows some slack in the design of the robots. The robots are allowed only four motion primitives, which we transcribe here again (Fig. 2.21).

- $u_{fwd} \longrightarrow$ Forward: The robot goes straight forward in the direction it is facing, both wheels rotate at their rated positive speed.
- $u_{bwd} \longrightarrow$ Backward: The robot goes backward in the opposite direction it is facing. Both wheels rotate at their rated negative speed.
- $u_{rgt} \longrightarrow$ Right: The robot rotates clockwise, turns right. The left wheel runs at rated positive speed, and the right wheel runs at rated negative speed.
- $u_{lft} \longrightarrow$ Left: The robot rotates counterclockwise, turns left. The left wheel runs at rated negative speed, and the right wheel runs at rated positive speed.

Let us consider the testing task: We want a group of robots to navigate the environment along a route, from some arbitrary point p_0 in E, according to the speed field in the environment. This speed field is specifically designed for the desired route, but for now is considered that exist in the environment, and is indicated to the robots through fixed landmarks in E (Fig. 3.29). When the landmark indicates a high intensity value, equation 3.15 establishes that the agent must greatly increase its speed. If the landmark indicates a low intensity value, the increase in agent speed is also low.

The intensity sensor on the robot allows to know the value of the speed field indicated by the landmark, but only when the robot locates the landmark. This feature reflects the robot's limited sensing capacity. The impact sensor completes the motion structure of the robot. Every time the robot detects an environmental limit or an obstacle on the way, the robot rotates on its axis a random angle, and advances again at constant speed. This random movement is what guarantees that the robot eventually finds the landmarks in the environment (Bobadilla, Gossman, & LaValle, 2011). Applying the primitives for forward, backward and rotation, the robot can follow the behavior described above. A possible navigation plan that solves this task is shown in Table 3.2.

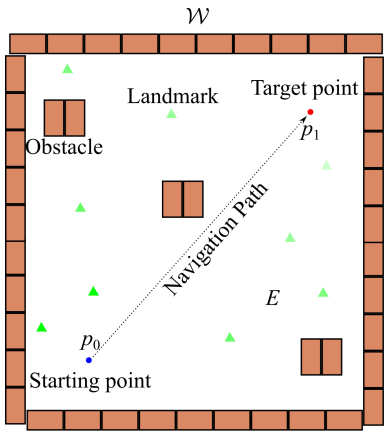


Figure 3.29: Environment for navigation on a designed route.

Environment with obstacles. The arbitrary starting point is p_0, and the destination point is p_1. Several landmarks indicate the value of the speed field at some points of E.

Table 3.2: Plan for the navigation task

1. If $h_{CS} = 1, \Rightarrow$ apply u_{bwd}, and then apply u_{lft} or u_{rgt} (random selection) a random angle. Go to step 1.
2. If $h_{CS} = 0, \Rightarrow$ apply u_{fwd}.
3. If $h_{IS}(x, y) \neq 0, \Rightarrow$ calculate the movement of the robot according to equation 3.15 and the value of $h_{IS}(x, y)$. Apply u_{fwd}, u_{bwd}, u_{rgt} or u_{lft} as required by the new position. Go to step 1.

In order to evaluate the behavior of this strategy, it is perform a first simulation of motion for a robot, under the following parameters:

1. A single robot in the environment. Constant robot speed of 0.2 m/s.
2. Environment of unlimited size and without obstacles.
3. Environment full of Brownian particles (landmarks), with random value of speed field. The range of values was taken between 0 and 0.2 m/s.
4. Robot's mass of 0.530 kg, according to real prototype. Mass assumed for Brownian particles of 0.1 kg.
5. Diffusion coefficient of the environment of 30×10^{-6} m^2/s.
6. Simulation time of one minute.

Starting from the position (0,0), and following the behavior policy described in Table 3.2, the robot navigated in the environment duplicating the behavior of a Brownian particle as shown in Fig. 3.30.

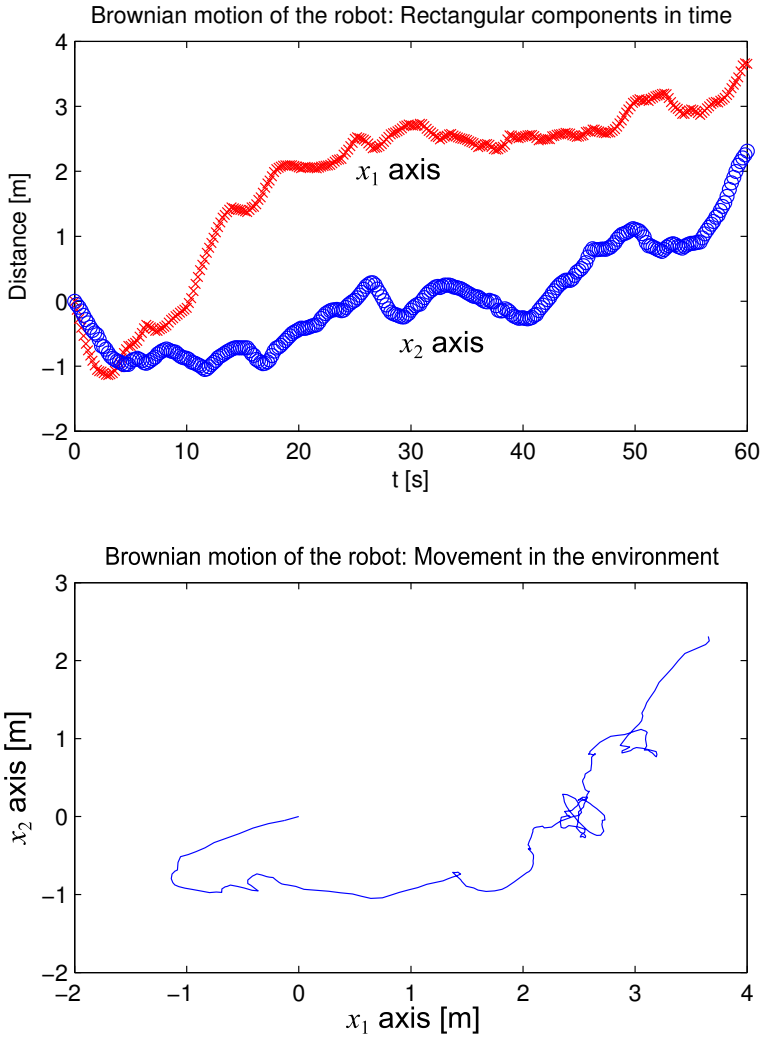


Figure 3.30: Brownian motion of a robot.

Brownian motion of a robot in an environment full of particles with random field intensity. (a) Evolution of the rectangular components of the displacement of the robot along the time. (b) Movement of the robot in the environment, starting from the position (0,0).

Route planning

This section presents a possible strategy for the design of the speed field in the environment of the robot, according to a desired path. Using this speed field to define the intensity of the signal in the landmarks, and according to the robot motion strategy proposed in the previous section, we expect the robots to achieve the goal, i.e., they will navigate the environment to their destination, avoiding obstacles, and no matter what were their starting point, after a finite number of motion primitives.

Let the function $P(x_1, x_2)|_{p_0 \to p_1}$ be the route designed for the robots in the environment E (Fig. 3.31). Let $\mathbf{v}(x_1, x_2)$ be the speed field on E which guides the movement of robots. Let D_C be the diffusion coefficient, that in the model is related to the concentration in the environment, and is therefore proportional to the mobility of the robots (constant value, all the robots are identical and with fixed structure).

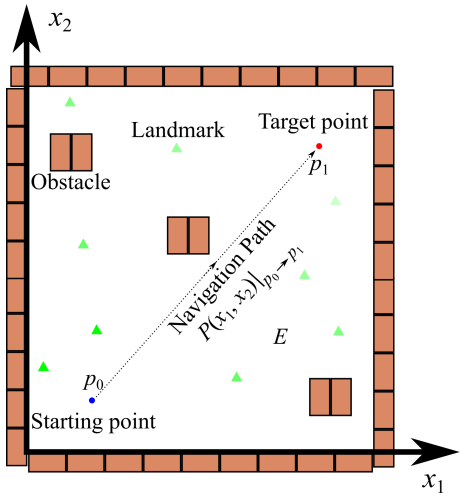


Figure 3.31: Planned route for navigation.

Planned route from p_0 to p_1 in two-dimensional environment with two obstacles.

In order that the robots are oriented along $P(x_1, x_2)|_{p_0 \to p_1}$, the intensity of the speed field should be minimal for the points belonging to $P(x_1, x_2)|_{p_0 \to p_1}$. In addition, the field intensity should increase as the points away from the route. This increase can be calculated along a normal line to the path traced. For any point $p(x_1, x_2)$ which does not belong to the path $P(x_1, x_2)|_{p_0 \to p_1}$, the value of the intensity of the speed field should be proportional to the perpendicular distance from the point to the path.

Finally, as the path has a direction, the intensity of the speed field must also decrease along the path towards the destination. As an example, consider again a two-dimensional environment without obstacles for the robot. Let us think now that we want to design a path to guide the robot to a certain position, for example, the position $(5, 4)$. The initial position of the robot is random, it can begin anywhere. For convenience of design, we selected as the starting point of the route an opposite extreme, for example the position $(1, 1)$. Following the given design parameters, we can construct a map of speed field as

shown in Fig. 3.32 (the maximum speed is adjusted according to the speed of the robot).

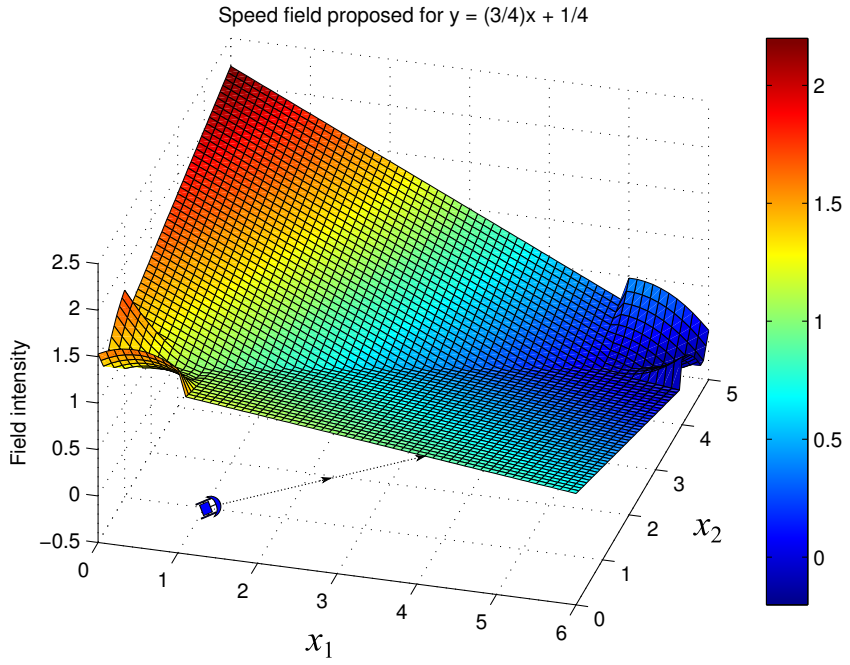


Figure 3.32: Speed field designed.

Speed field designed for the navigation path $x_2 = \frac{3}{4}x_1 + \frac{1}{4}$ between points $(1, 1)$ and $(5, 4)$.

This speed field does work as an attractive field for the particle, the robot. No matter what the initial position of the robot, the robot eventually will navigate guided by the path to its destination. To see this effect, we added this speed field to the previous simulation, and it is obtained a predictable behavior of the robot as shown in Fig. 3.33. All other parameters remained unchanged.

The initial position of the robot is selected arbitrarily in $(1, 3)$. Unlike the random behavior shown in Fig. 3.30, this time the robot turned to the path designed, and followed it as a reference until it reaches its destination, where it stayed around. However, the expected behavior in the prototypes is significantly different. We want to install only a few landmarks in the environment, which will make the robot explore the area before reaching its destination.

The landmarks always remain in its position, and encoding the value of the field intensity $\mathbf{v}$ at that point of E. Physically they can be implemented, for example, with colored markings on the floor. From this point of view, the landmarks can be interpreted as a generalization of the virtual gates.

When the agent finds one of these marks it reads the encoded intensity value (intensity of color, for example), and interprets it according to the control policy encoded in its genome (discrete transition to a new behavior). A high value of intensity in the landmark is interpreted by the agent as a high repulsive force, resulting in a high speed of movement in the agent in new random direction. In contrast, a low value of intensity in the landmark

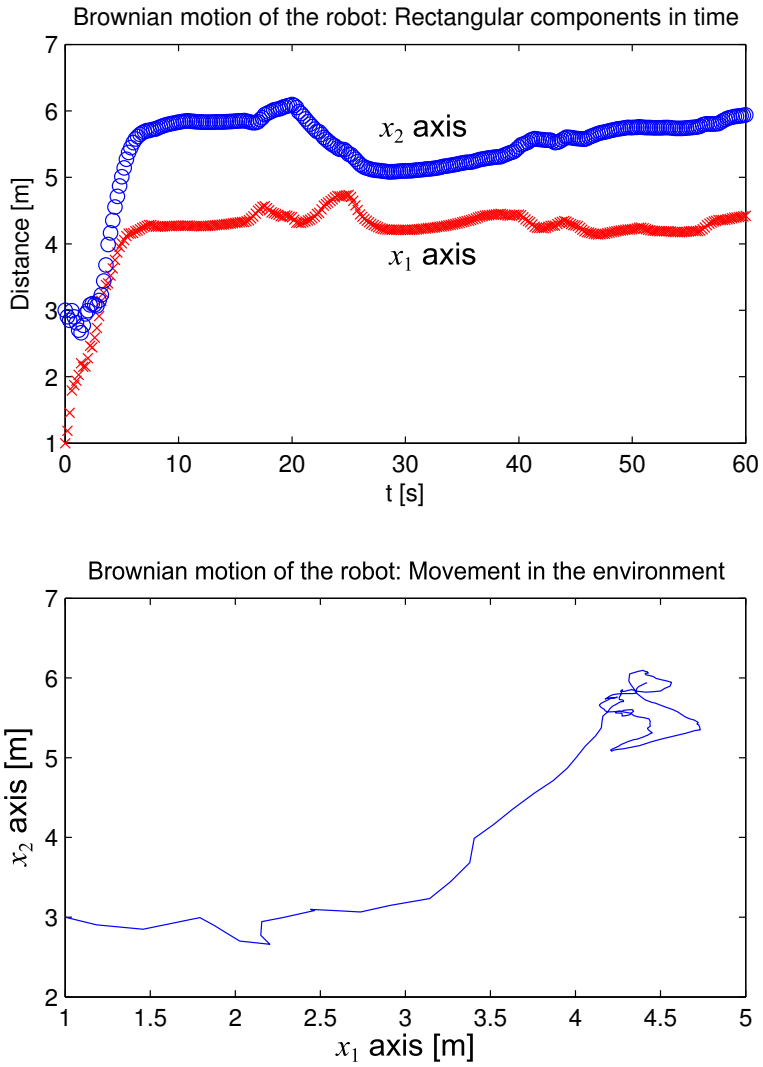


Figure 3.33: Brownian motion of a robot in the environment.

Brownian motion of a robot in an environment full of particles with designed field intensity for the navigation path $x_2 = \frac{3}{4}x_1 + \frac{1}{4}$ between points $(1, 1)$ and $(5, 4)$. (a) Evolution of the rectangular components of the displacement of the robot along the time. (b) Movement of the robot in the environment, starting from the position $(1, 3)$. The robot reads the local value of the speed field, and according to its intensity, the robot reacts and moves as if the environment pushed him toward his destination, in whose vicinity finally remains (top right of Fig. 3.33(b)).

is interpreted by the agent as a weak repulsive force, which keeps the agent relatively close to the landmark.

In the case of a hetereogeneos robot system, robots with different capacities of sensing and activation, the system model should focus on coordinate the sharing information (local communication between robots) and the information that each robot is able to read of the environment (Niemczyk & Geihs, 2015). The algorithm must be adjusted to allow that the system to process all the information required for the development of the task, which means that the robots read the environment according to the sensors that they possess and communicate this information to its nearest neighbors. Under this approach, all robots work together for the development of the task, regardless of their differences. However, it may be the case where it is desired that a group of robots, with certain characteristics, perform a task, while the rest perform another task. In the latter case it is necessary to change the model to allow segregation of a group of robots with certain characteristics (a particular species, for example). To do this, is necessary to create a representation of each group of robots, maintaining for the entire system (all robots) a single set of rules of interaction with the environment (for example, the same artificial potential function) (Filho & Pimenta, 2015).

3.4.4 Simulations

Simulations in Player/Stage are used to illustrate the performance. The configuration of the simulator fully respects the criteria of the current laboratory tests. Again, the simulations consider not only the functionality of the algorithm, but also having taken care of the physical characteristics of the robot (the SERB robot).

In relation to the diffusion model, the author has assumed as mass of the particles of the environment a value of 0.1 kg, this in order that this mass be comparable with the mass of the robot, and generates the desired motion effects.

The navigation task that are shown below is to guide a group of five robots to their destination in the top right of the environment. The environment has a few landmarks, so we hope that the robots explore the environment before reaching their destination. The robots are placed randomly in the environment, and they move independently according to their own evaluation of the model (discrete transition according to equation 3.15). The landmarks have been installed around the obstacles in the environment. This facilitated the characterization and implementation of the movement of the robots (in the wild motion, the agent changes direction when it finds an obstacle, the author uses this event to also change the speed of the agent). Fig. 3.34 shows the results of this simulation.

After about one minute, all the robots managed to reach the target area. The first one arrived in 35 s (00:00:35). As expected, the robots explored most of the environment during the navigation. When they reached the top right of the environment, the robots remained in that area navigating around it.

The analysis of the time required to complete the task, without and with QS, can be done statistically. The macro-scale model can also tune using this average time. It is possible to make a first approximation of statistical estimation through simulations. To do this, there were performed a total of 300 simulations, each with a total of 20 agents in

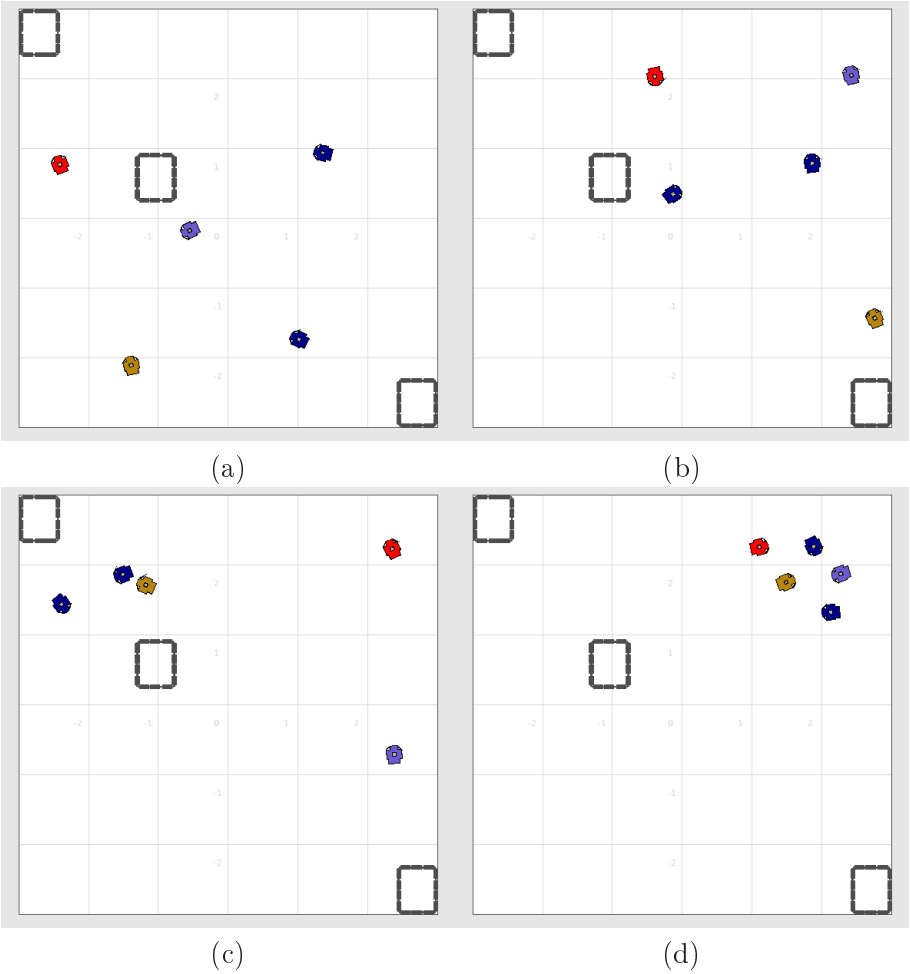


Figure 3.34: Simulation of five robots.

Simulation of five robots navigating from random positions in an environment of 6 m $\times$ 6 m. (a) Initial time of simulation ($t = 0$). The five robots start from random positions. (b) Sixteen seconds of simulation (00:00:16). The robots explore the environment and begin to proceed to their destination. (c) Thirty-five seconds of simulation (00:00:35). One robot has reached its destination. (d) Fifty-eight seconds of simulation (00:00:58). All robots have reached their destination.

the same environment.

Of these simulations, 150 were developed without including QS in the agents behavior, while the other half considered QS with a threshold of $T = 3$. All cases used a total of 20 agents in an environment of 6 m $\times$ 6 m. The simulations considered different numbers of landmarks to observe the effect on the required total time.

These data are summarized and analyzed using box and whisker plots. The summary statistics used to create a box and whisker plot are the median of the data, the lower and upper quartiles (25% and 75%) and the minimum and maximum values. Fig. 3.35 shows the results obtained when the system does not include QS, and Fig. 3.36 shows the results when the system includes QS.

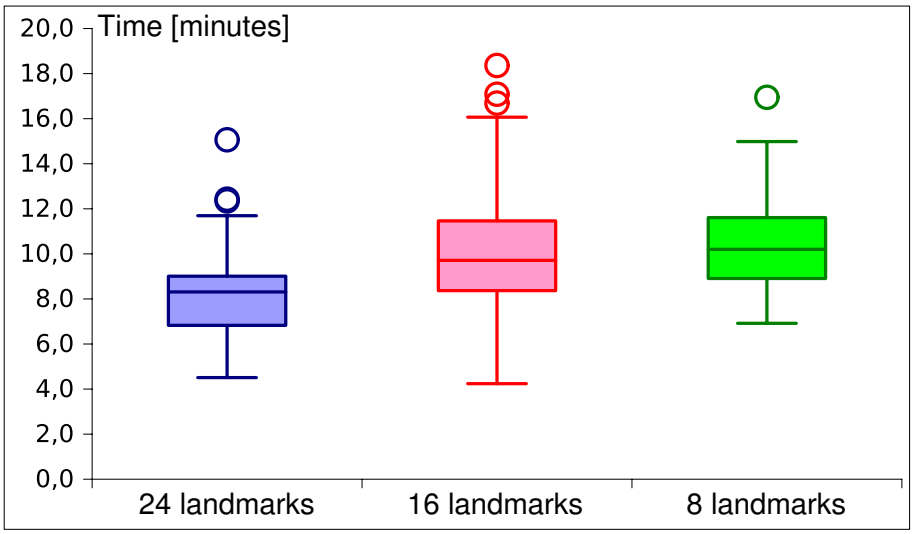


Figure 3.35: Effect of number of landmarks on the navigation total time (without QS).

Box and whisker plots of the navigation total time of 20 agents with random initial position. 50 simulations were performed for each of the three cases: 24 landmarks (left in blue), 16 landmarks (middle in red) and 8 landmarks (right in green). The central box represents the values from the lower to upper quartile (25th to 75th percentile). Excluded (and shown) outliers. The middle line represents the median. A line extends from the minimum to the maximum value.

Simulations show that in this navigation strategy, the number of landmarks affects the task development time, to a greater number of landmarks, lowest total time. This is particularly true when the control policies do not provide QS, where a reduction of 4% is observed when increasing from 8 to 16 landmarks, and 16% when increasing from 16 to 24 landmarks. When the system provides QS, the time is reduced by 19% when increasing from 8 to 16 landmarks, but the same effect is not observed when increasing over the number of landmarks.

The figures also show that the greatest impact in reducing the time it is due to QS. For the case of 8 landmarks, QS was able to reduce the time by 59%. In the case of 16 landmarks, QS was able to reduce the time by 65%. And in the case of 24 landmarks, QS

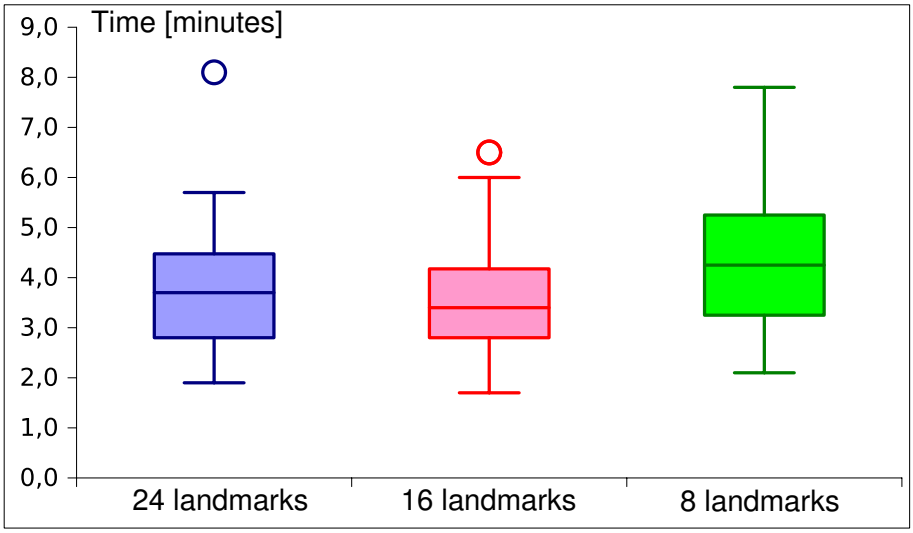


Figure 3.36: Effect of number of landmarks on the navigation total time (with QS).

Box and whisker plots of the navigation total time of 20 agents with random initial position. 50 simulations were performed for each of the three cases: 24 landmarks (left in blue), 16 landmarks (middle in red) and 8 landmarks (right in green). The central box represents the values from the lower to upper quartile (25th to 75th percentile). Excluded (and shown) outliers. The middle line represents the median. A line extends from the minimum to the maximum value.

was able to reduce the time by 55%.

For the macro-scale model, it is conceivable that, according to the design of the navigation strategy, each landmark induces a cell or region around, and that each of these regions may be a grouping area for agents. From this point of view, the model will have a number of areas equal to the number of landmarks. The following model considers the case of eight landmarks.

Again, according to the conditions of the system, for the initial choice of the areas there is no favoritism criteria, i.e.:

$$\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6 = \mu_7 = \mu_8 = \mu \tag{3.18}$$

The process of migration between the eight areas depends on the intensity of the speed field in the area. That is, the values of λ (eight variables), ρ (14 variables) and k (eight variables) depend (are functions) of $\mathbf{v}$. For simplicity, the landmarks are located along the designed route, from the farthest point (maximum intensity) to the destination point (minimum intensity). They are uniformly distributed to facilitate the calculation of the value of intensity that must encode each one (Fig. 3.37). The model is defined by the equations 3.19, 3.20 and 3.21.

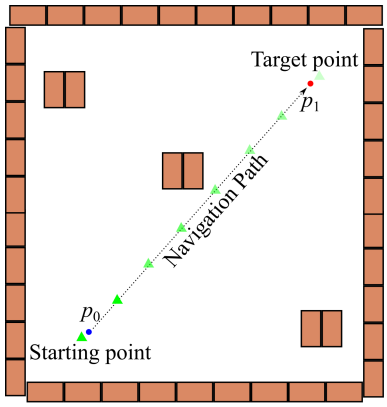


Figure 3.37: Environment and landmarks for navigation on a designed route.

The starting point is p_0, and the destination point is p_1. The landmark close to the starting point encodes the maximum intensity value. The landmark close the destination point encodes the lower intensity.

$$\dot{R} = -8\mu R \tag{3.19}$$

$$
\begin{aligned}
\dot{X}_1 &= \mu R + \lambda_1 Z_1 \Theta(R)\,\Theta(T - Z_1) - \rho_{12} X_1 + \rho_{21} X_2 - k_1 X_1 \\
\dot{X}_2 &= \mu R + \lambda_2 Z_2 \Theta(R)\,\Theta(T - Z_2) - \rho_{23} X_2 + \rho_{32} X_3 + \rho_{12} X_1 - \rho_{21} X_2 - k_2 X_2 \\
\dot{X}_3 &= \mu R + \lambda_3 Z_3 \Theta(R)\,\Theta(T - Z_3) - \rho_{34} X_3 + \rho_{43} X_4 + \rho_{23} X_2 - \rho_{32} X_3 - k_3 X_3 \\
\dot{X}_4 &= \mu R + \lambda_4 Z_4 \Theta(R)\,\Theta(T - Z_4) - \rho_{45} X_4 + \rho_{54} X_5 + \rho_{34} X_3 - \rho_{43} X_4 - k_4 X_4 \\
\dot{X}_5 &= \mu R + \lambda_5 Z_5 \Theta(R)\,\Theta(T - Z_5) - \rho_{56} X_5 + \rho_{65} X_6 + \rho_{45} X_4 - \rho_{54} X_5 - k_5 X_5 \\
\dot{X}_6 &= \mu R + \lambda_6 Z_6 \Theta(R)\,\Theta(T - Z_6) - \rho_{67} X_6 + \rho_{76} X_7 + \rho_{56} X_5 - \rho_{65} X_6 - k_6 X_6 \\
\dot{X}_7 &= \mu R + \lambda_7 Z_7 \Theta(R)\,\Theta(T - Z_7) - \rho_{78} X_7 + \rho_{87} X_8 + \rho_{67} X_6 - \rho_{76} X_7 - k_7 X_7 \\
\dot{X}_8 &= \mu R + \lambda_8 Z_8 \Theta(R)\,\Theta(T - Z_8) + \rho_{78} X_7 - \rho_{87} X_8 - k_8 X_8
\end{aligned}
\tag{3.20}
$$

$$
\begin{aligned}
\dot{Z}_1 &= k_1 X_1 - \rho_{12} Z_1 + \rho_{21} Z_2 - \lambda_1 Z_1 \Theta(R)\,\Theta(T - Z_1) \\
\dot{Z}_2 &= k_2 X_2 - \rho_{23} Z_2 + \rho_{32} Z_3 + \rho_{12} Z_1 - \rho_{21} Z_2 - \lambda_2 Z_2 \Theta(R)\,\Theta(T - Z_2) \\
\dot{Z}_3 &= k_3 X_3 - \rho_{34} Z_3 + \rho_{43} Z_4 + \rho_{23} Z_2 - \rho_{32} Z_3 - \lambda_3 Z_3 \Theta(R)\,\Theta(T - Z_3) \\
\dot{Z}_4 &= k_4 X_4 - \rho_{45} Z_4 + \rho_{54} Z_5 + \rho_{34} Z_3 - \rho_{43} Z_4 - \lambda_4 Z_4 \Theta(R)\,\Theta(T - Z_4) \\
\dot{Z}_5 &= k_5 X_5 - \rho_{56} Z_5 + \rho_{65} Z_6 + \rho_{45} Z_4 - \rho_{54} Z_5 - \lambda_5 Z_5 \Theta(R)\,\Theta(T - Z_5) \\
\dot{Z}_6 &= k_6 X_6 - \rho_{67} Z_6 + \rho_{76} Z_7 + \rho_{56} Z_5 - \rho_{65} Z_6 - \lambda_6 Z_6 \Theta(R)\,\Theta(T - Z_6) \\
\dot{Z}_7 &= k_7 X_7 - \rho_{78} Z_7 + \rho_{87} Z_8 + \rho_{67} Z_6 - \rho_{76} Z_7 - \lambda_7 Z_7 \Theta(R)\,\Theta(T - Z_7) \\
\dot{Z}_8 &= k_8 X_8 + \rho_{78} Z_7 - \rho_{87} Z_8 - \lambda_8 Z_8 \Theta(R)\,\Theta(T - Z_8)
\end{aligned}
\tag{3.21}
$$

Fig. 3.38, 3.39, 3.40 and 3.41 show the behavior of these equations. The curves show again how the QS reduces the task total time. When the system includes QS, the response speed is reduced, increasing the total time (from 300 s the population in region 8 grows very slowly).

When the system includes QS (threshold $T = 3$), the populations in the regions seven ($t = 350$ s) and eight ($t = 190$ s) reach QS, which increases the population growth in these areas.

An important aspect considered in the simulations was the robustness of the system, and the guarantee of success in the task. The Player/Stage simulations were duplicated applying during the navigation the death of some of the agents (5% to 20% of the population). As a result, since the scheme is not centralized, and each agent acts independently, the task was completed in 100% of cases. However, the total time required for developing the task changed. For populations without QS, the total time decreased between 2% and 5%, which makes evident some degree of correspondence between the time and population size. Furthermore, in cases with QS, total time increased nearly 8% (cases with more deaths), which also indicates some degree of correspondence between variables, but opposite. Dependence in cases with QS is because the activation of virulence is determined by a population threshold, which is affected by the death of agents.

3.5 Performance comparison

Looking to answer the doubts about the convergence time of a QS algorithm over another without QS, we developed a statistical comparison of results between different collective navigation strategies. For the comparison, three multiagent formation control algorithms

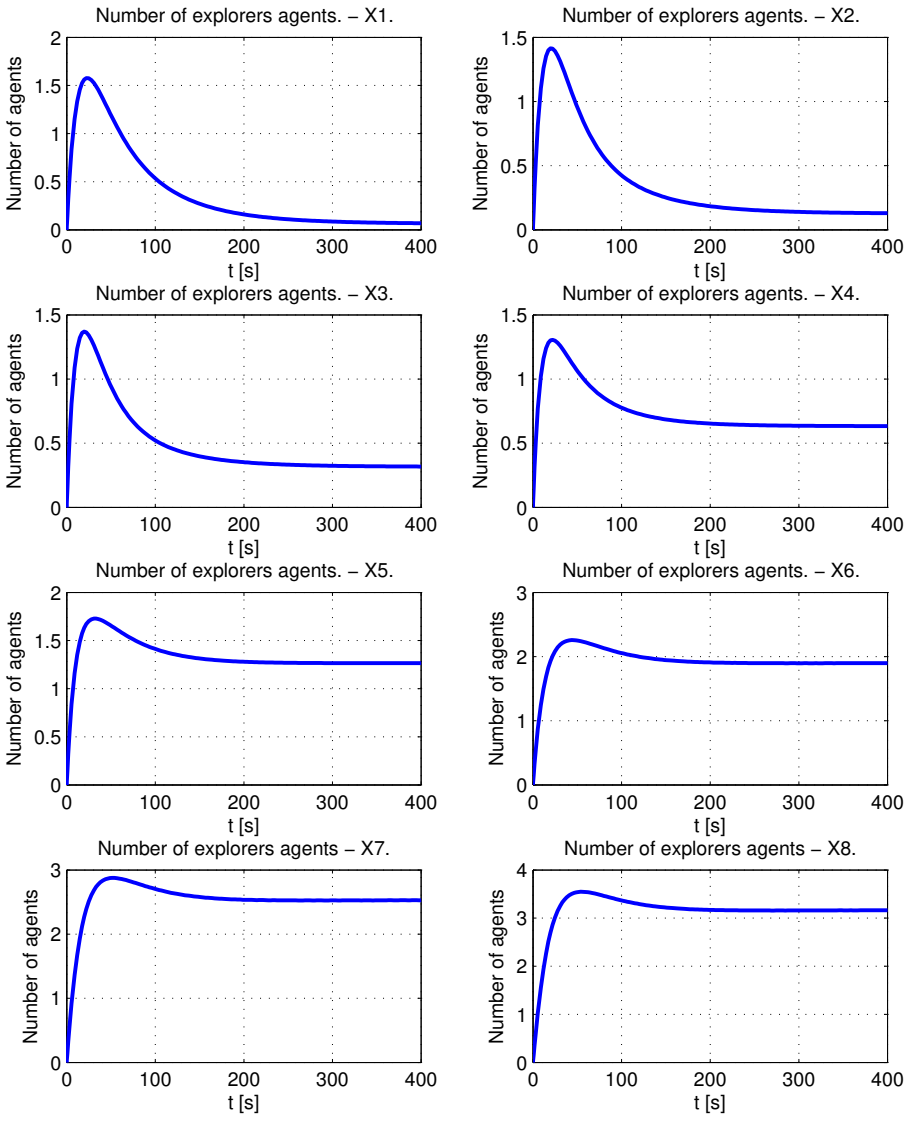


Figure 3.38: Macro model simulation for designed route (explorers, 20 agents, without QS).

Macro model simulation for navigation task with designed route. The figure shows the size of the population in areas close to the eight landmarks (explorer agents). The simulation does not include QS.

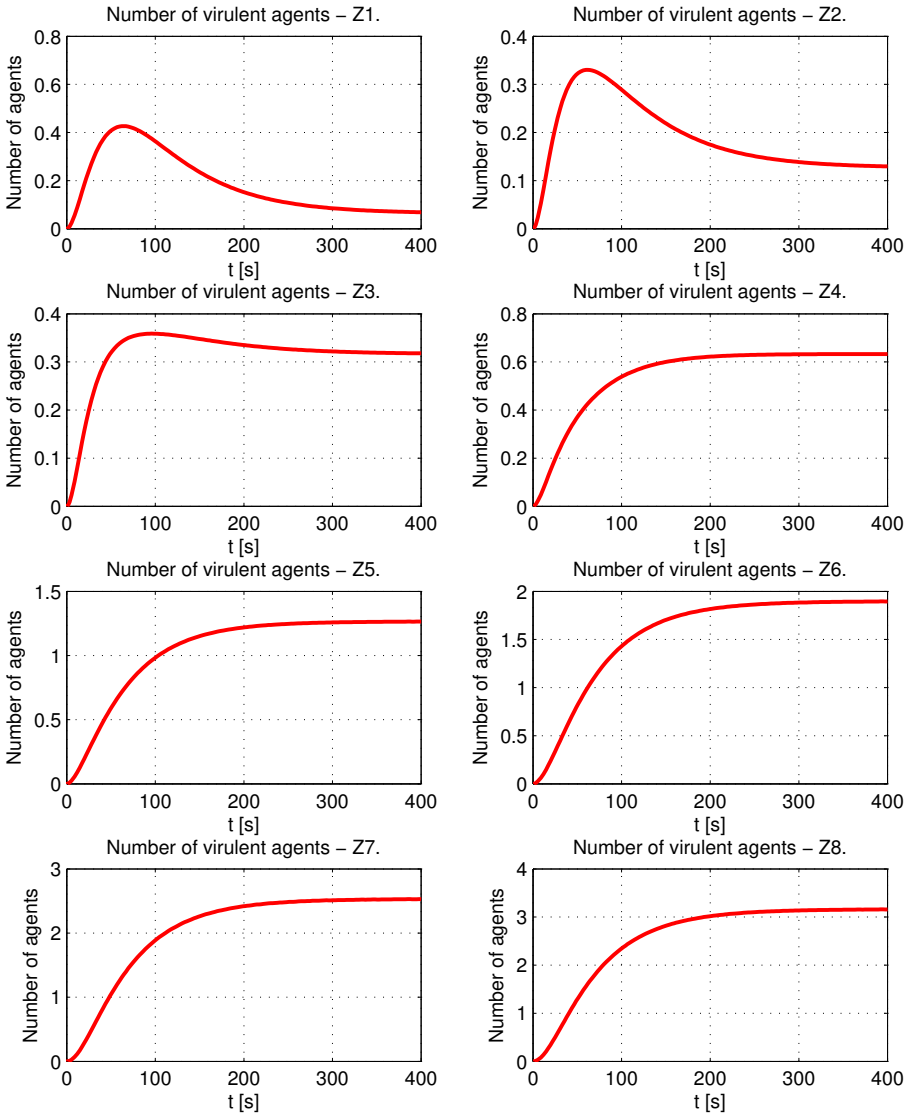


Figure 3.39: Macro model simulation for designed route (virulent, 20 agents, without QS).

Macro model simulation for navigation task with designed route. The figure shows the size of the population in areas close to the eight landmarks (virulent agents). The simulation does not include QS.

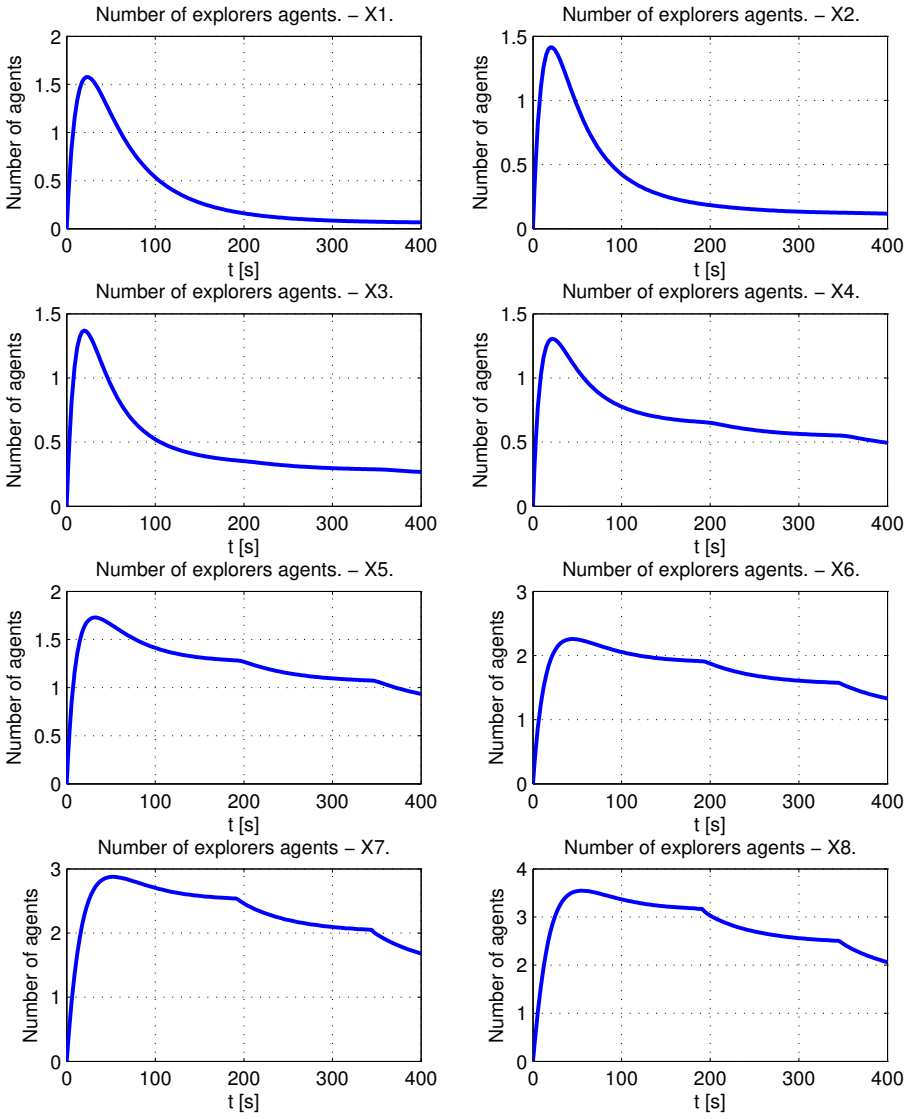


Figure 3.40: Macro model simulation for designed route (explorers, 20 agents, with QS).

Macro model simulation for navigation task with designed route. The figure shows the size of the population in areas close to the eight landmarks (explorer agents). The simulation includes QS ($T = 3$).

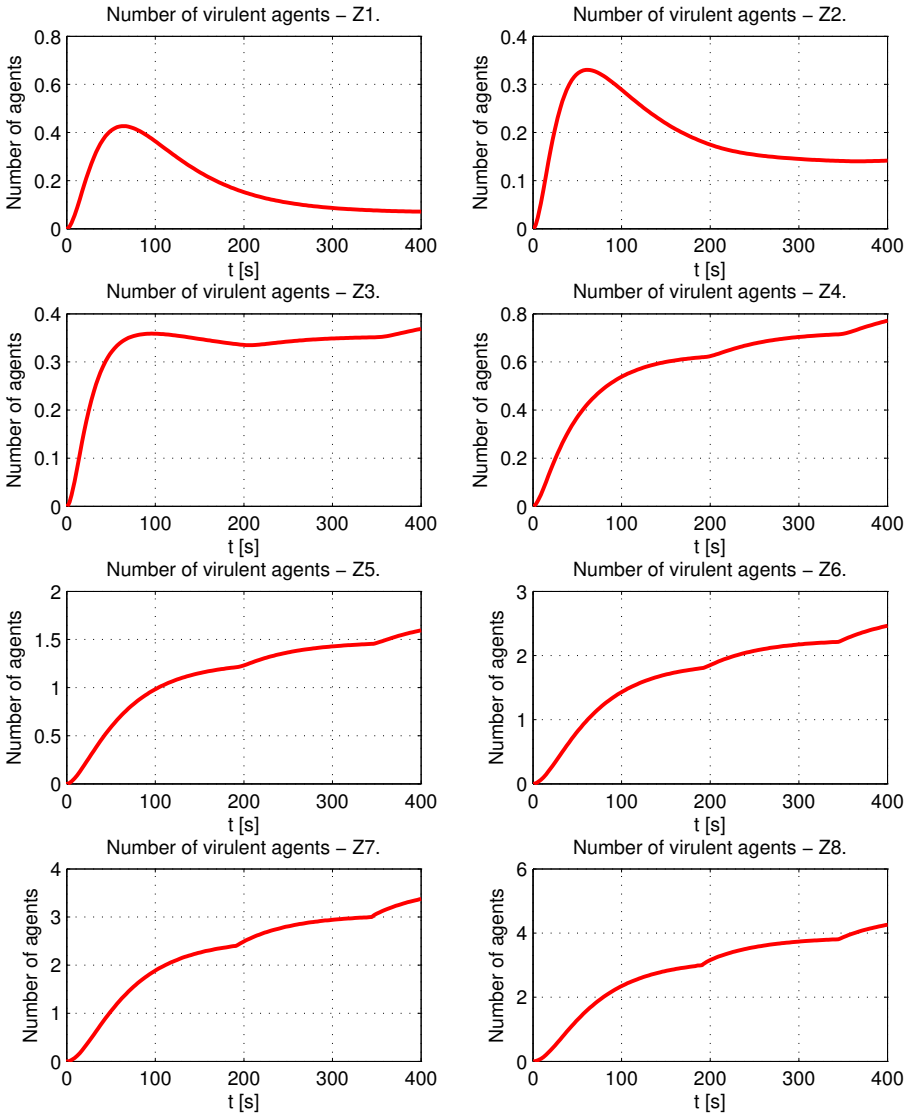


Figure 3.41: Macro model simulation for designed route (virulent, 20 agents, with QS).

Macro model simulation for navigation task with designed route. The figure shows the size of the population in areas close to the eight landmarks (virulent agents). The simulation includes QS ($T = 3$).

were selected. As a premise of selection of the algorithms, we decided to select fully documented algorithms in high-impact journals. The three selected strategies are:

- A sliding fuzzy scheme without delay (Sahraei, Nemati, Farshchi, & Meghdari, 2010)
- A sliding fuzzy scheme with delay (Khosravi, Jahangir, & Afkhami, 2012)
- A sliding fuzzy scheme and H^{∞} with delay (Ranjbar-Sahraei, Shabaninia, Nemati, & Stan, 2012)

These three researches were analyzed to achieve duplication of results. In all three cases, the solution of the problem is addressed by designing a potential field between the robots (attractive/repulsive potential field equivalent to the local communication between agents), and controlling the relative position of the robot using an approximate model (point at which differs each research). The three cases contemplate the formation of up to seven robots in a geometric structure as an equilibrium of the system. This is why it was included in all cases a block of random error in the sensor model (simulating severe limitations in the sensors), in order to make more difficult the task of control.

After duplicating the results of the three schemes, the problem conditions were adjusted to take it to a task of grouping of agents. This task defined a constant radius between agents of value equal to the radius of the circle which circumscribes it, i.e., $\frac{0.27 \text{ m}}{2}$ (for example, in star formation problem five different values are contemplated). The algorithms were implemented in ANSI C on Player/Stage.

The simulations used again the SERB robot in the 6 m $\times$ 6 m square navigation environment. Each test used a total of 10 robots, and a density threshold of $T = 5$. In order to obtain a correct estimate of the performance of each scheme, the evaluation of each is performed on a set and 50 different tests (300 simulations in total). In each test the initial position and orientation of each robot are changed by choosing them uniformly random with in the environment. This allows agents to experience many different initial conditions.

The population size has increased to facilitate QS. However, the value has remained close to that reported in (Ranjbar-Sahraei et al., 2012) in order to ensure consistency in the results. A comparison with a significantly greater population requires a scalability analysis of each strategy.

Table 3.3 shows the total task times obtained for 50 cases of each algorithm. The first three columns show the original algorithms adapted to the grouping problem, and the last three columns show the three algorithms modified with the inclusion of QS. Table 3.4 shows the results of an analysis of variance (ANOVA) used to analyze the differences between the groups of means of the 300 tests. Since the ANOVA test provides a statistical test of whether or not the means of several groups are equal, this analysis reveals whether really the inclusion of QS algorithm contributes to reducing the total time of the task (Fisher, 1925). Fig. 3.42, Fig. 3.43 and Fig. 3.44 show the details of behavior of the average time value for each algorithm (distribution of values), in each case comparing the time value without QS to the time value with QS.

Table 3.3: Total task time for six grouping algorithms.

Sahraei2010	Khosravi2012	Ranjbar2012	Sahraei2010+QS	Khosravi2012+QS	Ranjbar2012+QS
37.9	24.5	39.0	36.5	21.8	37.0
38.1	25.7	38.2	34.8	22.0	37.3
35.9	28.7	37.9	35.4	25.2	35.4
35.5	27.6	39.4	33.5	22.0	35.3
35.9	27.4	38.8	35.6	25.7	36.1
37.5	28.0	39.3	34.9	20.5	36.5
38.0	22.3	37.6	35.8	20.8	35.5
36.0	26.3	38.1	36.5	25.8	35.2
38.2	27.7	38.2	35.6	21.0	35.5
36.6	23.5	39.3	33.6	23.7	36.0
37.9	21.9	37.6	33.5	21.4	35.1
36.9	25.7	38.9	35.2	23.7	36.2
35.3	23.5	39.5	36.0	26.4	36.8
37.5	22.1	38.7	35.8	24.8	35.4
35.9	26.8	38.4	34.9	21.9	36.2
36.0	22.2	38.9	34.7	20.5	36.5
37.0	23.0	37.9	33.7	23.8	36.1
37.3	23.3	37.7	33.5	23.9	37.0
37.0	24.1	39.4	34.7	25.9	37.1
38.3	28.5	37.5	34.4	24.3	35.5
35.3	28.2	38.4	36.1	26.4	35.9
35.6	24.5	39.0	36.2	21.1	37.2
38.0	25.0	38.2	33.5	20.7	36.6
36.1	26.1	37.5	35.1	24.5	35.3
35.4	28.4	38.8	35.9	24.5	35.9
38.0	27.5	38.0	33.8	21.4	35.3
36.9	23.0	37.8	34.5	22.5	35.9
35.8	25.9	39.4	35.5	26.3	36.8
37.9	28.0	38.0	34.1	23.0	37.2
36.5	23.1	38.2	34.5	22.7	35.6
38.3	23.4	38.0	36.3	25.0	36.1
35.3	23.7	39.1	35.2	23.0	35.8
36.2	27.8	39.5	34.3	20.9	36.2
37.0	26.7	39.4	35.5	22.5	35.9
35.3	24.5	38.6	34.5	24.3	36.2
37.6	23.6	38.9	33.7	20.7	37.3
36.4	28.0	38.2	36.5	20.9	35.5
36.9	28.6	37.6	35.9	24.7	36.3
37.5	25.1	38.4	33.5	25.3	36.6
35.9	24.5	38.8	34.0	24.9	36.1
36.9	25.5	39.1	35.4	23.0	36.6
36.6	28.8	38.0	35.4	25.6	36.2
36.0	28.5	38.0	34.7	22.3	37.0
35.3	28.3	37.7	34.6	21.0	36.3
35.8	25.5	39.2	36.3	26.2	35.7
35.9	25.4	38.4	36.0	23.2	35.7
36.7	28.0	38.7	35.6	21.8	35.2
38.4	27.6	38.1	34.2	21.7	36.3
36.3	25.7	38.1	34.8	25.3	35.2
37.3	23.1	37.6	36.1	23.8	35.2

Each algorithm was run 50 times (300 simulations), altering in each case the initial conditions of the robots.

Table 3.4: ANOVA on the total task time for six grouping algorithms

Results of ANOVA test of average Time_[S] for groups from "Sahraei2010" to "Ranjbar2012+QS"

Analysis of variance table

Source	Sum of Squares	df	Mean Sum of Squares	F	p[1]
Between	10187.826	5	2037.565	1123.249	< 0.001
Within	533.314	294	1.814		

O'Brien's test for homogeneity of variance: 0.0 [2]

Group summary details

Group	N	Mean	CI 95% [3]	Standard Deviation[4]	Min	Max	Kurtosis[5]	Skew[6]	p abnormal[7]
Sahraei2010	50	36.72	36.447 - 36.993	0.985	35.2539857007	38.4360902759	-1.253	0.145	< 0.001
Khosravi2012	50	25.696	25.099 - 26.293	2.154	21.9466580357	28.7877084699	-1.328	-0.129	< 0.001
Ranjbar2012	50	38.465	38.291 - 38.639	0.628	37.4512564299	39.5453771598	-1.196	0.163	0.003
Sahraei2019+QS	50	34.998	34.736 - 35.261	0.947	33.4666895775	36.4925274578	-1.175	-0.125	0.004
Khosravi2012+QS	50	23.285	22.762 - 23.808	1.888	20.4753774557	26.4491906678	-1.304	0.100	< 0.001
Ranjbar2012+QS	50	36.095	35.915 - 36.274	0.648	35.1442187644	37.3343032694	-0.973	0.277	0.063

[1] If p is small, e.g. less than 0.01, or 0.001, you can assume the result is statistically significant ie. there is a difference between at least two groups. Note: a statistically significant difference may not necessarily be of any practical significance.

[2] If the value is small, e.g. less than 0.01, or 0.001, you can assume there is a difference in variance.

[3] We are 95% certain that the true value of the mean is within this interval. But it could still lie anywhere outside of those bounds.

[4] Standard Deviation measures the spread of values.

[5] Kurtosis measures the peakedness or flatness of values. Between -2 and 2 means kurtosis is unlikely to be a problem. Between -1 and 1 means kurtosis is quite unlikely to be a problem.

[6] Skew measures the lopsidedness of values. Between -2 and 2 means skew is unlikely to be a problem. Between -1 and 1 means skew is quite unlikely to be a problem.

[7] This provides a single measure of normality. If p is small, e.g. less than 0.01, or 0.001, you can assume the distribution is not strictly normal. Note - it may be normal enough though.

The ANOVA compares six (6) groups comprised of six algorithms on a navigation problem: A sliding fuzzy system without delay (Sahraei et al., 2010), a sliding fuzzy system with delay (Khosravi et al., 2012), a sliding fuzzy system and H^∞ with delay (Ranjbar-Sahraei et al., 2012), and variants each of them including QS ($T = 5$).

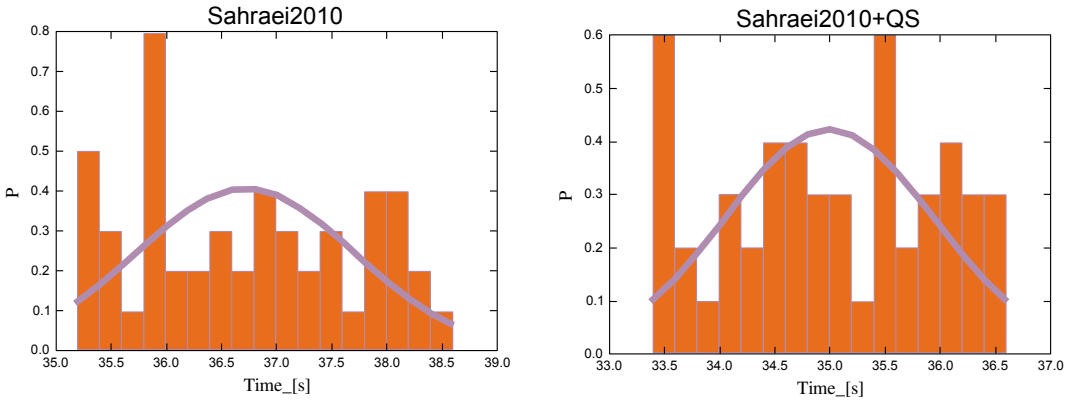


Figure 3.42: Sliding fuzzy system without delay (10 agents).

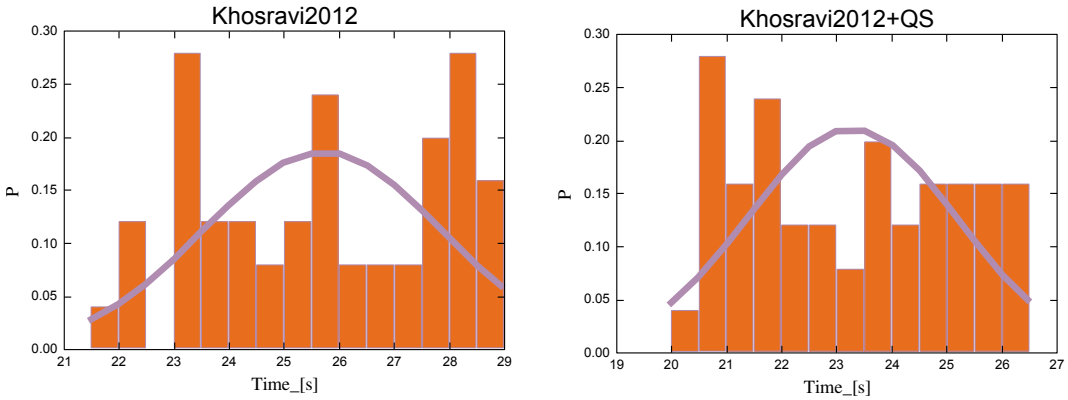


Figure 3.43: Sliding fuzzy system with delay (10 agents).

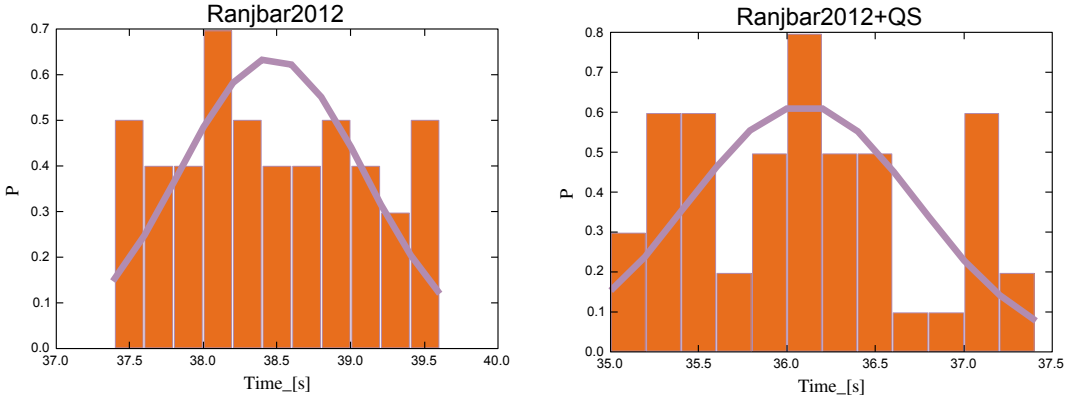


Figure 3.44: Sliding fuzzy system and H^{∞} with delay (10 agents).

3.6 Summary

In this final chapter, the author shows how to use the simplified model of bacterial QS to design electronic hardware systems. In particular, for the design of navigation routes in dynamic environments for swarms robots.

First, the author introduces and exemplifies the properties of, what has been called in robotics, wild motion. He presents a first approach to solve the navigation problem where the environment helps to guide the agents to their destination. This dynamic ensures complete exploration of the area of navigation, and thus ensures that each robot senses the local information needed to change its behavior.

In the second section, the author presents a local communication strategy for a group of robots that allows collective coordination. Using a basic task of grouping, the author shows the hybrid structures of the agent and the system, and shows how to define control policies to activate different behaviors. Analyzing different operating conditions, the author shows how the inclusion of QS considerably reduced the execution times.

Finally, in the last section, the author takes all these concepts to formulate a comprehensive motion strategy for robots based on fluids particle diffusion models. Using this updating movement, and local environmental readings (landmarks), the author manages to coordinate a group of robots along an unknown environment to the desired destination.

A key element throughout all designs, was the minimalist approach adopted. With advances in technology, mobile robots are increasingly equipped with rich sensors, powerful control boards, high-performance computers, and high-speed communication links. This has enabled the development of highly sophisticated systems for common tasks such as navigation, exploration, patrolling, and coverage. This usually leads to a significant modeling burden, which includes system identification and careful mapping of the robot's environment. Powerful sensors are used for mapping and localization. Filters are developed to obtain state estimates so that policies based on state feedback can be designed and implemented. Further complications arise in the case of multiple robots: Careful coordination and communication strategies are usually developed, sometimes involving a centralized controller. For us, all these features not only increases the complexity (cost) but also the risk of system failure, which is against our robust approach.

CHAPTER 4

Conclusions and future work

4.1 Conclusions

In this work, the author has presented a new design strategy for artificial systems based on a simplified model of cellular interactions in bacteria known as `quorum sensing` (QS). While the initial formulation of the research involved a design strategy specific to electronic hardware, along this investigation the author concludes two important aspects: first, the engineering design can not be restricted to a single application area, i.e. not be restricted exclusively to electronics. Current systems are an integration of different disciplines, and the design should include this kind of structure. This is why the current boom in what is known as Cyber-Physical System. And second, the design model that he proposes is not restricted only to electronic systems, this scheme can be applied to design any artificial system, any CPS, as the author has shown in robotic applications.

The proposed algorithm tries to duplicate genetic expressions of bacteria, activated by population sizes (QS). This feature of bacteria is simplified and implemented on several platforms, at different levels of complexity. Early experiments show the principles of interaction based on local environmental readings. Subsequent experiments show how to solve problems from this local interaction.

The QS is a behavioral model postulated in biology to analyze and understand social activity in bacteria. Thanks to it, today there is an understanding of the processes that can, for example, develop more efficient treatments for bacterial infections than traditional antibiotics. While the development of these models is quite recent, its use in engineering is no the novelty in this research. It is possible to find a lot of papers in engineering around these ideas, specifically as it relates to the development of bio-inspired models, and of course in its use for solving problems.

The different strategies that researchers try to copy into these models can be summarized as: bacterial foraging, bacterial chemotaxis, reproduction, coding in RNAs, adaptation, evolution, aging mechanism and membrane systems. All these schemes have a common element, they all seek to raise simple rules of each individual that addresses in the system its behavior, and that together allow the system reflects coordinated behaviors,

which respond to specific environmental conditions. This also was the idea in this research, and the author has achieved a quite simple differential model considering specifically the reproduction, cell differentiation and population organization processes.

Regarding the potential applications of these models in engineering, many researchers report work in search and optimization, which seems to be a natural niche of application of all bio-inspired models. Some use them in communication systems, others in pattern classification, in finance, for image processing and estimation of harmonics among others. These are all problems that can be formulated as harder higher-dimensional and dynamic optimization problems. Other examples of specific application include reconfigurable hardware, synthetic biocircuits, nanomachines and robotics, a field in which we decided to apply our design proposal. All these applications, with the correct formulation, can also be seen as search and optimization problems.

The author has divided the analysis and development of the design model based on QS in three parts. In the first part, he focuses on the interaction between cells and evaluated the general characteristics of a simplified model of QS. He provides a detailed explanation of adaptation of the process, and then he shows in laboratory the functioning emulating the bacterium model on an 8-bit microcontroller, and a framework with CPLDs to build the hardware platform where the bacterial population increases. Thanks to this first work, he manages to demonstrate the potential of the model for the design of artificial systems. Furthermore, the results of the system simulation help to understand the behavior of bacterial communities, and in particular they show that in the self-organization configuration investigated, the system's overall behavior is robust and scalable like the biological model.

In the second part, the author develops the macro-scale model. With the idea of top-down scheme design, he presents and explains the proposed system model. In this model, he synthesizes the desired behavior of the system from the processes of Reproduction, Exploration and Virulence. Through analysis and simulations, and using differential equations, he shows that the system is stable, and that the system converges to the desired final state.

Finally, in the third part, the author presents the micro-scale model. He shows a model of the agent that allows to implement tasks, in particular, robotics navigation tasks. Desired behaviors defined at macro-level were projected as individual bacterium behaviors. In this way, rather than compute the exact trajectories and sensory information of single individual, he defines local rules of behavior that allow each bacterium build its own information space, and from it develop the task. He shows the functionality of the design scheme through use of a few examples with real robots, particularly, basic navigation tasks.

Both the agent as the system are modeled as hybrid systems. From this point of view, the continuous dynamic that characterizes a certain behavior changes according to local conditions detected by the agent. These discrete changes are coordinated through control policies encoded in each agent (genome), which are modeled as filters. The hybrid behavior of agents also induces a hybrid behavior in the system.

After providing a discussion of design methodology, the author illustrates the functionality of the design scheme based on QS through the use of several examples. The results of the navigation experiments with small robots are presented, including experiments with wild bodies, experiments where robots perform the construction of a very simple informa-

tion space, and experiments with a formal application of navigation route design.

In the first experiments, the author presents an approach to control multiple robots without system identification, map building, localization or precise state estimation. The key ideas are to make wildly behaving robots and gently guide them through the use of virtual gates. He demonstrates the approach on simple, low-cost robots and in simulation. If the wildness condition is satisfied, then the discrete transitions occur during execution, thereby solving the desired task. The control modes are set to induce the desired paths through the transition graphs.

In subsequent experiments, the author allows robots build simple information spaces, and that they undertake collective coordination from them. First, he develops experiments where he wants collective coordination from radiated signals. He formulates the algorithm, the navigation plan to be followed by the agent, and analyzes the convergence of the strategy. The interaction between the robots responds to the simplified algorithm of local communication, which guarantees a minimum sensing (agent with limited sensing and/or environment that limits the use of sensors), what makes the strategy promissory for exploration in collapsed and unknown environments. In tests he demonstrates that the grouping task is performed by robots regardless if one or more of the agents are damaged.

Then, in the final tests, the author develops experiments to demonstrate how to design specific navigation paths for the robots. He shows that a group of robots can be used to solve a navigation task and exploration using a single minimalist design in hardware, software and algorithm operation. The system dynamics (interaction between the robots and the environment) responds to the simplified algorithm of Brownian motion. Again, the navigation task is performed by robots regardless if one or more of the agents are damaged. The robustness of the system, characterized by the multitude of agents developing the task, greatly increases the performance in real environments, unknown and dynamic.

In all developed experiments was observed as the inclusion of QS is able to reduce the time required for the task between 20% and 60%, depending on the particular conditions of the problem. The QS accelerates the convergence increasing the weight of the most promising solutions. If some agents find some interesting local maximum, the QS reduces the random scan, decanting faster the global maximum.

The experimental results have shown that the design methodology was successful in fulfilling the demands of performance and robustness in a new artificial system, and indeed, in a system consisting of multiple structures, both electronic as mechanical and computational.

4.2 Future work

There are several interesting ideas that could be explored in future work:

- Landmarks: Regarding the experiments conducted in this research, the observed results left some questions that should be thoroughly investigated, particularly with respect to the landmarks. For example, how many landmarks set in the environment? and more importantly, what are the optimal locations? Since the landmarks

can become simple color marks on the environment (with a color code), one of the strategies to investigate should involve analysis of graph coloring techniques.

- Evolution: A natural characteristic of bacteria that greatly influenced in their diversity and successful survival in a variety of environments and adverse conditions, is their high mutation rate. An interesting ingredient that deserves research, and could have a significant impact on the robustness of the model, and the versatility to development tasks, is the possibility of evolution and genetic adaptability of bacteria.

- Interaction with other bacterial species: A whole new field of research that opens from this work, is related to the interaction of different kinds of bacteria within the same system. That is, within a single system there may be two or more species of bacteria, each with its own genetic code (different behavioral rules) among which different kinds of relationships appearing. Interesting aspects to investigate include tasks in parallel, cooperativism between species, identification, classification and detection of intruders.

- Estimation of harmonics: This was initially the application proposed for the algorithm. Although at the suggestion of the reviewers of the research proposal, they propose to think about another kind of application, the interest in working in the area is maintained. In (Ji et al., 2008), for example, they propose an adaptive bacterial swarming algorithm to estimate the frequencies and phases of the fundamental frequency, integral harmonics and inter-harmonics, along with a least-square method to estimate the amplitudes. This is an optimization algorithm, such as the proposed algorithm based on QS. In this sense, a logical first work to be done is to optimize the search with our QS algorithm, and compare performance with the one obtained by (Ji et al., 2008).

APPENDIX A

SERB Robot

In this section, we include schematics and design information of the SERB (Arduino Controlled Servo Robot) robot (Oomlout, 2013) that document our navigation hardware.

A.1 Parts

Nuts and Bolts

- 3mm x 15mm bolt (x25)
- 3mm x 10mm bolt (x15)
- 3mm nut (x42)
- 3mm washer (x20)
- 8mm x 25mm bolt (x2)
- 8mm nut (x2)
- Skate Bearings (x2) These are standard skate bearings from roller-blades or skateboard.

Tires

- Large O-ring (4.5" ID 3/16" Bead Dash ♯349) (x2) (McMaster-Carr Product ♯9452K407)
- Small O-ring (3/4" ID 3/16" Bead Dash ♯314) (x1) (McMaster-Carr Product ♯9452K387)

Electronics

- Arduino Uno (x1) (Maker Shed)

- Continuous Rotation Servo (x2) (Parallax)
- 400 Contact Breadboard (x1)
- Quad AA Battery Box (x1)
- 9V Battery Clip (x1)
- 2.1 mm Plug (x1)
- 3 pin header (x2)
- A-B USB Cable - Printer Style (x1)

Batteries

- AA Battery (x4)
- 9v Battery (x1)

Wire

- 100 cm wire 22 AWG solid.

A.2 Cutting Pieces

The mechanical structure (chassis) of the SERB robot is formed by acrylic pieces. It is possible to purchase the pre-cut parts from Oomlout directly (Oomlout, 2013); however, since the design was modified and supplemented (Fig. A.1), the suggested choice for the reproduction of the robot is cutting an acrylic sheet. The design involves the use of a acrylic sheet of 3 mm thick. The cut should be made with a laser cutter, this is due to the fragility of the material, and the precision of the parts in the final assembly.

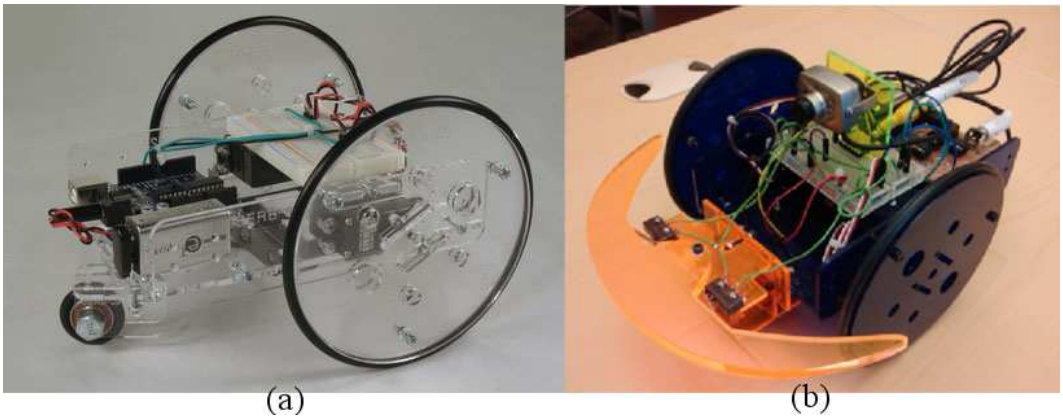


Figure A.1: SERB Robot. (a) Oomlout original design. (b) Modified design.

The laser cutter layout in PDF format is in the attachments to this document. Fig. A.2 shows the 3D layout for all acrylic pieces of the robot chassis, including the modifications made to the original design.

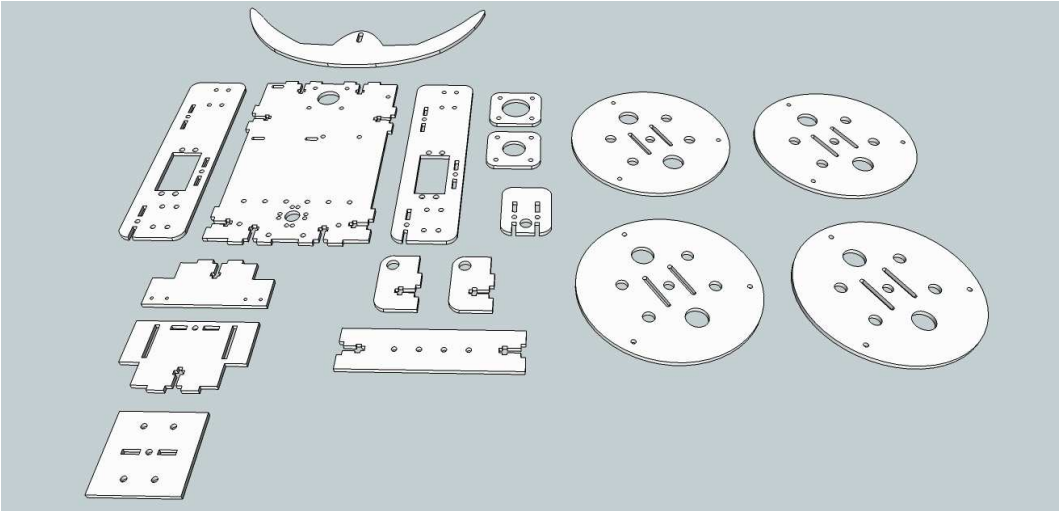


Figure A.2: 3D layout for SERB robot.

A.3 Assembly

The acrylic pieces are assembled together like a puzzle, and fixed with screws and nuts. Figs. A.3, A.4, A.5, A.6 and A.7 show different 3D views around of the robot chassis assembly scheme. Figures omitted screws, nuts and wires for the sake of clarity.

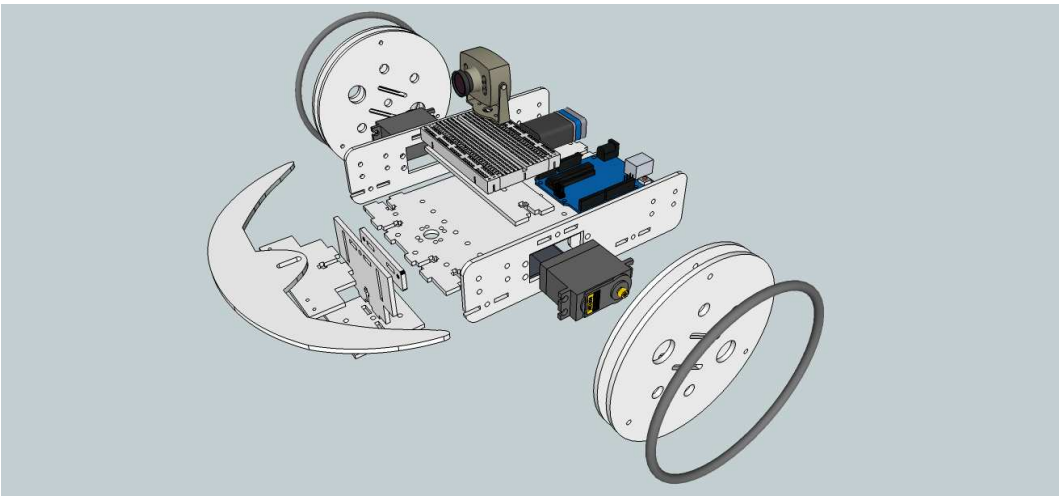


Figure A.3: Assembly SERB robot. 3D detail 1.

A animation of these 3D views can be seen at (Martinez S., 2013) (*Serb robot with camera*).

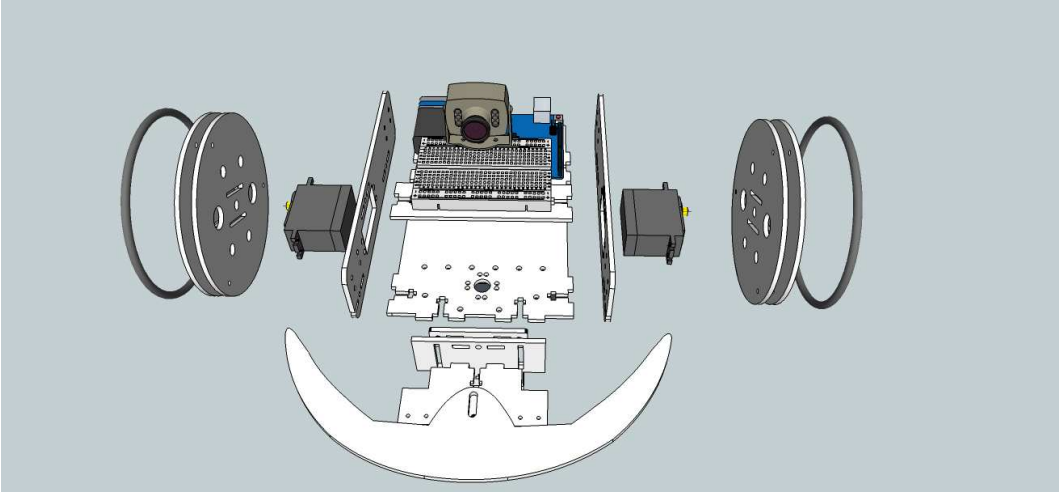


Figure A.4: Assembly SERB robot. 3D detail 2.

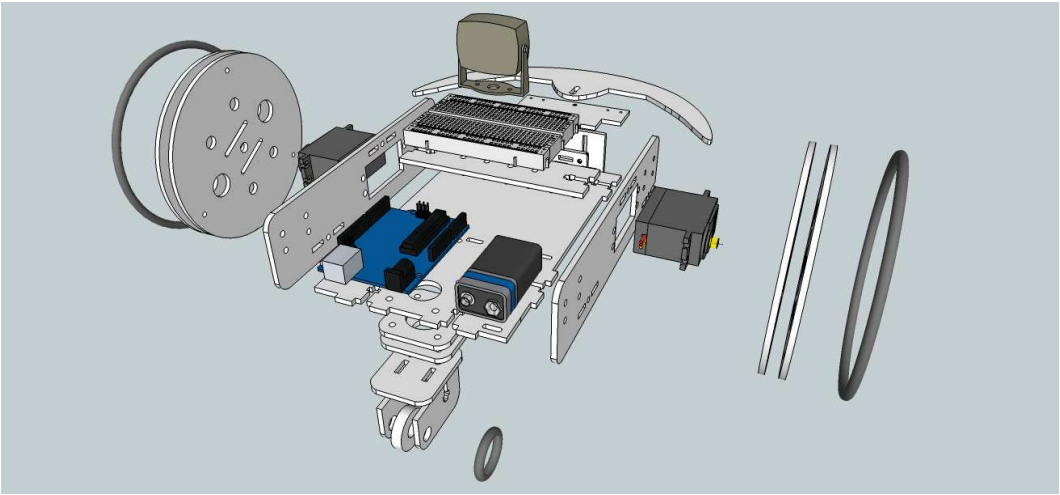


Figure A.5: Assembly SERB robot. 3D detail 3.

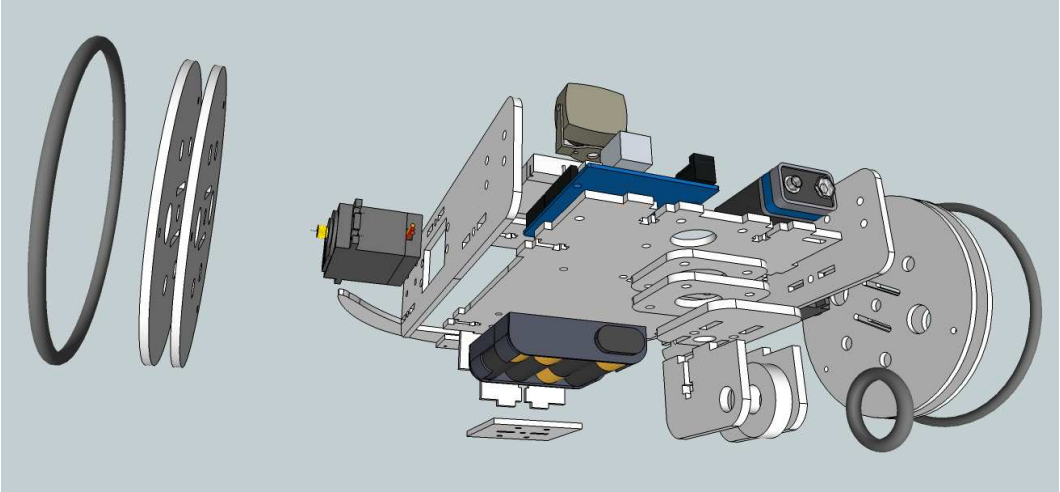


Figure A.6: Assembly SERB robot. 3D detail 4.

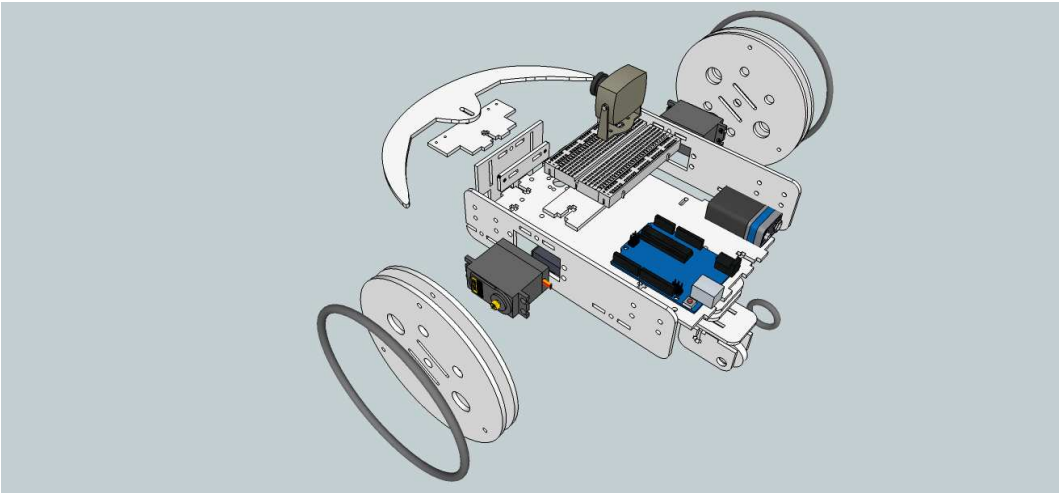


Figure A.7: Assembly SERB robot. 3D detail 5.

APPENDIX B

SERB Robot Software

In this appendix, we show the code used in the SERB robot implementing the ergodic navigation scheme with virtual gates (Chapter 3). This code was modified to obtain different simulation conditions.

```
 // Fredy H. Martínez S. - Leonardo Bobadilla
// September 1, 2011
// System:  Arduino UNO - SERB robot
// Description:  Navigation with ergodic motion and virtual gates.

#include <Servo.h> // Library for servos
#include <Wire.h>
#include <SoftwareSerial.h>
#include <NewSoftSerial.h> // Library for 2 wires comms
#include <TimerOne.h>

// A library for timer interrupts on Timer2 with ms resolution
#define RIGHTWHISKER 6 //the pin the right whisker is attached to
#define LEFTWHISKER 7 //the pin the left whisker is attached to
#define LEFTSERVOPIN 10 //The pin the left Servo is connected to
#define RIGHTSERVOPIN 9 //The pin the right Servo is connected to
#define MAXSPEED 30
//Due to the way continuous rotation servos, work maximum speed is
//reached at a much lower value than 90 (this value will change
//depending on the servos) (for Parallax servos)
//20 will give it full range, 10 makes it very controllable but a little slow

Servo leftServo;
Servo rightServo;

int leftSpeed; //sets the speed of the robot (left servos)
//a percentage between -MAXSPEED and MAXSPEED
```

```
int rightSpeed; //sets the speed of the robot (both servos)
//a percentage between -MAXSPEED and MAXSPEED
int speed = 40; //used for simple control (goForward etc.)
//a percentage between 0 and MAXSPEED

#define BLACK 2
#define RED 3
#define WHITE 4
#define WALL 1
#define NO_WALL 0

#define FOREVER 789234
#define ZIGBEE_PERIOD 2000
// The period (in ms) of the sensor checking function
#define ROBOT_ID 0

//Variables for ColorPAL
SoftwareSerial Color90(2, 3); // rx = 2, tx = 3

uint8_t pinRx = 11, pinTx = 12; // the pin on Arduino
long BaudRate = 9600 , sysTick = 0;
char GotChar;

// Initialize NewSoftSerial
NewSoftSerial mySerial( pinRx , pinTx );

int red;
int grn;
int blu;

int gotcolor = 0;
int letter;

int whiteCount=0;
int redCount=0;

//counters of walls
int whiteWalls=0;
int redWalls=0;
int wallCounter=0;
int wallCounterOther=0;
int path[5]={WHITE,RED,BLACK,RED,BLACK};
//int path[5]={RED,WHITE,BLACK,RED,BLACK};

// Constants and Data
//char buffer[BUFFER_SIZE]; a buffer for sensor data
int blueState = WALL,
```

```
   yellowState = WALL,
   whiteState=WALL,
   redState = WALL,
   blackState = WALL;
unsigned long startTime, d, time0,time1;
int totDist = 0;
int whiskerReads = 0;

boolean flag=true;

void setup()
{
   Serial.begin(BaudRate); // Start communication with serial port read value
   mySerial.begin(BaudRate);
   //pinMode(13, OUTPUT);
   //pinMode(12, OUTPUT);
   //pinMode(11, OUTPUT);

   Wire.begin();
   palSetup();
   serbSetup(); // Adds the servos and prepares all
   //SERB related variables
   whiskerSetup();
   nextAssigment();
   time0= millis();
}

void loop()
{
   time1= millis();

   goForward(); //sends the robot forward
   checkWhiskers(); //checks to see if a whisker is pressed
   checkColor();

   if((time1-time0)>2000){
      readZigBee();
      time0 = time1;
   }
   // terminateProgram();
}

void readZigBee()
{//call periodically to read zigbee

// Monitor data from XBee
   if ( mySerial.available() )
```

```
   {
      GotChar = mySerial.read();
      if(GotChar=='4')
      {
         Serial.println("White");
         wallCounterOther++;
      }
      if(GotChar=='3')
      {
         Serial.println("Red");
         wallCounterOther++;
      }
      if(GotChar=='2')
      {
         Serial.println("Black");
         wallCounterOther++;
      }
      if(wallCounterOther==wallCounter)
      { //we are in the same region
         nextAssigment();
      }
   }
}

void nextAssigment()
{
   Serial.println("Together");
   //Serial.println(wallCounter);
   //Serial.println(wallCounter);

   assignColor(path[wallCounter], NO_WALL);
   for(int i=2;i<=4;i++)
   {
   if(i!=path[wallCounter])
      {
         assignColor(i, WALL);
      }
   }
}

int getColor()
{
   int color=0;
   int cr, cg, cb;
   int wr, wg, wb; //Reference black
   int kr, kg, kb; //Reference black
   float rv, gv, bv;
```

```
wr=188;
kr = 17;

wg=214;
kg= 21;

wb=318;
kb= 30;

rv = (float)255/(wr-kr);
gv = (float)255/(wg-kg);
bv = (float)255/(wb-kb);

readcolor();
//White Reference R181 G243 B 287
//black reference R29 G32 B 37 Color:

cr = rv*(red - kr);
cg = gv*(grn - kg);
cb = bv*(blu - kb);

if(red==0 || grn ==0 || blu==0)
{
// Serial.println("Error");
   return -1;
}

int sum = cr+cg+cb;

if (cr < 20 && cg < 20 && cb < 20)
{
   color = BLACK;
   //Serial.print("Black");
}
else if (cr > 150 && cg > 150 && cb > 150)
{
   // Serial.print("White");
   color = WHITE;
   whiteCount++;
}
else if (cr >100 && red > blu && red > grn)
{
   //Serial.print("Red");
   color = RED;
   redCount++;
}
else
```

```
    {
       color =-1;
       whiteCount=0;
       redCount=0;
       //Serial.print("other");
    }

    gotcolor = 0;
    //delay(100);
    return color;
}

void palSetup()
{
    Color90.begin(4800); // Send signal to led to read value
    pinMode(2,INPUT); // serial pin out from color pal
    pinMode(3,INPUT); // from same serial pin, pulls up, sends, pulls down, reads
    digitalWrite(2,HIGH); // Enable the pull-up resistor
    digitalWrite(3,HIGH); // Enable the pull-up resistor

    pinMode(2,OUTPUT); // send signal out
    pinMode(3,OUTPUT);
    digitalWrite(2,LOW); // turn pin off so pin 3 can go high
    digitalWrite(3,LOW);

    pinMode(2,INPUT); // Input signal to print
    pinMode(3,INPUT);

    //Serial.println("Pass 1");

    while( digitalRead(2) != HIGH || digitalRead(3) != HIGH )
    {
    //Serial.println("In the loop");
    delay(50);
    }

    // Serial.println("Pass 2");

    pinMode(2,OUTPUT);
    pinMode(3,OUTPUT);
    digitalWrite(2,LOW);
    digitalWrite(3,LOW);
    delay(80);

    pinMode(2,INPUT);
    pinMode(3,OUTPUT);
    delay(100);
```

```
   Serial.print("Pass 3");
}

void readcolor()
{ // Reads ColorPAL, putting results in the red,grn,blu variables
   char rByte[9];
   char dummy[4];

   delay(20);
   Color90.begin(4800);
   Color90.print("= (00 $ m) !"); // set up loop to continuously send color data
   pinMode(3,INPUT);
   gotcolor = 0;
   while (gotcolor == 0)
   {
      rByte[0] = Color90.read();
      if( rByte[0] == '$' )
      {
         gotcolor = 1;
         for(int i=0; i<9; i++)
         {
            rByte[i] = Color90.read();
         }
         dummy[0] = rByte[0];
         dummy[1] = rByte[1];
         dummy[2] = rByte[2];
         dummy[3] = 0;

         red = strtol(dummy,NULL,16);

         dummy[0] = rByte[3];
         dummy[1] = rByte[4];
         dummy[2] = rByte[5];

         grn = strtol(dummy,NULL,16);

         dummy[0] = rByte[6];
         dummy[1] = rByte[7];
         dummy[2] = rByte[8];

         blu = strtol(dummy,NULL,16);
      }
   }
}

void terminateProgram()
{
```

```
   goStop();
   while(true){}
}

//------------------------------------------------------------------------
//WHISKER ROUTINES

void whiskerSetup()
{
   pinMode(RIGHTWHISKER, INPUT); //Sets the right whisker's pin to be an input
   pinMode(LEFTWHISKER, INPUT); //Sets the left whisker's pin to be an input
   digitalWrite(RIGHTWHISKER, HIGH);//Sets the right whisker pin's internal pullup
   digitalWrite(LEFTWHISKER, HIGH);//Sets the left whisker pin's internal pullup
}

void checkWhiskers()
{
   if(!digitalRead(RIGHTWHISKER)){reverseAndTurnWalls(); whiskerReads++;}
   //if the right whisker is pressed then reverse and turn
   if(!digitalRead(LEFTWHISKER)){reverseAndTurnWalls(); whiskerReads++;}
   //if the left whisker is pressed then reverse and turn
}

void checkColor()
{
   int colorLocal;
   colorLocal = getColor();
   // Serial.println(colorLocal);

   if((colorLocal==RED) && redState)
   {//red is a wall
      reverseAndTurnColors();
   }
   if((colorLocal==WHITE) && whiteState)
   {//white is a wall
      reverseAndTurnColors();
   }
   if((colorLocal==BLACK) && blackState)
   {//white is a wall
      reverseAndTurnColors();
   }
   if((colorLocal==RED) && !redState)
   {//red is one-way
      moveForward();
      mySerial.println(RED);
      wallCounter++;
      if(wallCounterOther==wallCounter)
```

```
         { //we are in the same region
            nextAssigment();
         }
   }
   if((colorLocal==WHITE) && !whiteState)
   { //white is one-way
      moveForward();
      mySerial.println(WHITE);
      wallCounter++;
      if(wallCounterOther==wallCounter)
      { //we are in the same region
         nextAssigment();
      }
   }
   if((colorLocal==BLACK) && !blackState)
   { //black is one-way
      moveForward();
      mySerial.println(BLACK);
      wallCounter++;
      if(wallCounterOther==wallCounter)
      { //we are in the same region
         nextAssigment();
      }
   }
}

void reverseAndTurnWalls(){
   goBackward(); //goes backward
   checkColor();
   delay(50); //for 700 miliseconds
   turnRandom(300,400); //turns randomly between 300 and 1500 ms
}

void reverseAndTurnColors()
{
   goBackward(); //goes backward
   delay(500); //for half a second
   turnRandom(300,1000); //turns randomly between 300 and 1500 ms
}

void moveForward()
{
   digitalWrite(13, HIGH);
   goForward();
   delay(500);
   digitalWrite(13, LOW);
}
//END WHISKER ROUTINES
```

```
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//START OF ARDUINO CONTROLLED SERVO ROBOT (SERB) ROUTINES
void serbSetup()
{
   setSpeed(speed);
   pinMode(LEFTSERVOPIN, OUTPUT); //Left servo signal pin to output
   pinMode(RIGHTSERVOPIN, OUTPUT); //Right servo signal pin to output
   leftServo.attach(LEFTSERVOPIN); //Attaches left servo
   rightServo.attach(RIGHTSERVOPIN); //Attaches right servo
   goStop();
}

/*
* Sets the speed of the robot between 0-(stopped) and 100-(full speed)
* NOTE: speed will not change the current speed, if you must change speed
* then call one of the go methods before changes occur.
*/
void setSpeed(int newSpeed)
{
   if(newSpeed >= 100)
   {
      newSpeed = 100;
   } //if speed is greater than 100 make it 100
   if(newSpeed <= 0)
   {
      newSpeed = 0;
   } //if speed is less than 0 make it 0
   speed = newSpeed * MAXSPEED / 100; //Scales speed [0 to MAXSPEED]
}

/*
* Sets the speed of the robots rightServo between -100-(reversed) and 100-(forward)
* NOTE: calls to this routine will take effect imediatly
*/
void setSpeedRight(int newSpeed)
{
   if(newSpeed >= 100)
   {
      newSpeed = 100;
   } //if speed is greater than 100 make it 100
   if(newSpeed <= -100)
   {
      newSpeed = -100;
   } //if speed is less than -100 make it -100
   rightSpeed = newSpeed * MAXSPEED / 100; //scales speed [-MAXSPEED to MAXSPEED]
   rightServo.write(90 - rightSpeed); //sends the new value to the servo
```

```
}

/*
* Sets the speed of the robots leftServo between -100-(reversed) and 100-(forward)
* NOTE: calls to this routine will take effect immediatly
*/
void setSpeedLeft(int newSpeed)
{
   if(newSpeed >= 100)
   {
      newSpeed = 100;
   } //if speed is greater than 100 make it 100
   if(newSpeed <= -100)
   {
      newSpeed = -100;
   } //if speed is less than -100 make it -100
   leftSpeed = newSpeed * MAXSPEED / 100; //scales speed [-MAXSPEED to MAXSPEED]
   leftServo.write(90 + leftSpeed); //sends the new value to the servo
}

/*
* Turns the robot randomly left or right for a random time period between
* minTime (milliseconds) and maxTime (milliseconds)
*/
void turnRandom(int minTime, int maxTime)
{
   int choice = random(2); //Random number between left (1) and right (0)
   int turnTime = random(minTime,maxTime); //Random number for the pause time
   if(choice == 1)
   {
      goLeft(); checkColor();
   } //If random number = 1 then turn left
   else
   {
      goRight(); checkColor();
   } //If random number = 0 then turn right
   delay(turnTime); //delay for random time
}

/*
* Sends the robot forwards
*/
void goForward()
{
   leftServo.write(90 + speed);
   rightServo.write(90 - speed);
}
```

```
/*
* Sends the robot backwards
*/
void goBackward()
{
   leftServo.write(90 - speed);
   rightServo.write(90 + speed);
}

/*
* Sends the robot right
*/
void goRight()
{
   leftServo.write(90 + speed);
   rightServo.write(90 + speed);
}

/*
* Sends the robot left
*/
void goLeft()
{
   leftServo.write(90 - speed);
   rightServo.write(90 - speed);
}

/*
* Stops the robot
*/
void goStop()
{
   leftServo.write(90);
   rightServo.write(90);
}

void assignColor(int tapeColor, int wallType)
{
   switch(tapeColor)
   {
      case WHITE:
         whiteState = wallType; break;
      case RED:// since RED and RED_TAPE map to same value, only one is included
         redState = wallType; break;
      case BLACK:// since RED and RED_TAPE map to same value, only one is included
         blackState = wallType;
   }
}
```

APPENDIX C

Player/Stage Simulations

In this appendix, we show the code used to simulate the interaction between robots in Chapters 2 and 3.

The Player/Stage Project includes the Player network server and the Stage robot platform (2D robot simulation environment). The Player is set of APIs (e.g. position2d, bumper, ir, speech, power) that can be implemented by a robot chassis (SERB, Roomba, Khephera etc.), possibly over serial line or network, or as we used it in our research by Stage.

The Stage simulator is a 2D multiple-robot simulation environment. Stage can be used alone to simulate robot behaviors via user-defined control programs. Stage can also interface with Player, allowing users of the Player to access simulated sensors and devices through the Player interfaces.

To use these tools is necessary to generate a set of configuration files. These files allow to define the drivers to be used with Player and the interfaces which will have access. A driver in Player is a piece of code that provides an interface to a sensor or motor device.

- *.cfg file: A *.cfg (configuration) file is what Player reads to get all the information about the robot that the user is going to use. This file tells Player which drivers it needs to use in order to interact with the robot (for simulation, the driver is always Stage). It also tells Player how to talk to the driver, and how to interpret any data from the driver so that it can be presented to the code.
- *.world file: A *.world file tells Player/Stage what things are available to put in the world. The user describes his robot, any items which populate the world and the layout of the world.
- *.inc file: A *.inc file, which follows the same syntax and format of a *.world file but it can be included, which is easily reusable.

Player uses a Server/Client structure in order to pass data and instructions between the code (the user code, the Player client program) and the robot's hardware. Player is a

server, and a hardware device on the robot is subscribed as a client to the server via proxy. The code can be written in C or C++.

This code was modified to obtain different simulation conditions.

C.1 qs_nav.cfg file

```
Description:  Navigation of robots - Quorum Sensing
 # Author:  Fredy H. Martinez S.
 # Date:  14 Feb 2012
 # CVS: v 1.0 2012/02/14

 # load the Stage plugin simulation driver
 driver
 (
    name "stage"
    provides ["simulation:0" ] # key:host:robot:interfce:index
    plugin "libstageplugin"

    # load the named file into the simulator
    worldfile "qs_nav.world"
 )

 driver
 (
    name "stage"
    provides ["map:0"]
    model "lab"
 )

 # Driver and attach position2d, laser and blobfinder interfaces to "robot1"
 driver
 (
    name "stage"
    provides ["6665:position2d:0" "6665:laser:0" "6665:ptz:0"]
    model "robot1"
 )

 # Driver and attach position2d, laser and blobfinder interfaces to "robot2"
 driver
 (
    name "stage"
    provides ["6665:position2d:1" "6665:laser:1" "6665:ptz:1"]
    model "robot2"
 )

 # Driver and attach position2d, laser and blobfinder interfaces to "robot3"
```

```
driver
(
   name "stage"
   provides ["6665:position2d:2" "6665:laser:2" "6665:ptz:2"]
   model "robot3"
)

# Driver and attach position2d, laser and blobfinder interfaces to "robot4"
driver
(
   name "stage"
   provides ["6665:position2d:3" "6665:laser:3" "6665:ptz:3"]
   model "robot4"
)

# Driver and attach position2d, laser and blobfinder interfaces to "robot5"
driver
(
   name "stage"
   provides ["6665:position2d:4" "6665:laser:4" "6665:ptz:4"]
   model "robot5"
)

# Driver and attach position2d, laser and blobfinder interfaces to "robot6"
#driver
#(
# name "stage"
# provides ["6665:position2d:5" "6665:laser:5" "6665:ptz:5"]
# model "robot6"
#)

# Driver and attach position2d, laser and blobfinder interfaces to "robot7"
#driver
#(
# name "stage"
# provides ["6665:position2d:6" "6665:laser:6" "6665:ptz:6"]
# model "robot7"
#)

# Driver and attach position2d, laser and blobfinder interfaces to "robot8"
#driver
#(
# name "stage"
# provides ["6665:position2d:7" "6665:laser:7" "6665:ptz:7"]
# model "robot8"
#)
```

```
# Driver and attach position2d, laser and blobfinder interfaces to "robot9"
#driver
#(
# name "stage"
# provides ["6665:position2d:8" "6665:laser:8" "6665:ptz:8"]
# model "robot9"
#)

# Driver and attach position2d, laser and blobfinder interfaces to "robot10"
#driver
#(
# name "stage"
# provides ["6665:position2d:9" "6665:laser:9" "6665:ptz:9"]
# model "robot10"
#)
```

C.2 qs_nav.world file

```
# Description:  Differential robot with black and white analog camera
# CVS: v 1.00 2012/02/14
# Fredy H. Martinez S.

# Defines differential robot SERB
include "serb.inc"

# defines 'map' object used for floorplans
include "mapa_uiuc.inc"

# defines sensor for the analog camera as laser with 120 deg and 0.4 m
include "cam_bw.inc"

# size of the world in meters
size [6 6]

# set the resolution of the underlying raytrace model in meters
resolution 0.02

# update the screen every 10ms (we need fast update for the stest demo)
gui_interval 200

# configure the GUI window
window
(
   size [ 591.000 612.000 ]
   center [-0.010 -0.040]
```

```
   scale 0.011
)

# load an environment bitmap
map
(
   bitmap "bitmaps/lab_uiuc.png"
   size [6 6]
   name "lab"
)

# create a SERB robot
SERB_CAM
(
   name "robot1"
   color "DarkBlue"
   pose [1.627 0.879 -204.413]
   sick_laser( samples 361 laser_sample_skip 4 )
)

# create a SERB robot
SERB_CAM
(
   name "robot2"
   color "red"
   pose [-2.289 0.506 417.420]
   sick_laser( samples 361 laser_sample_skip 4 )
)

# create a SERB robot
SERB_CAM
(
   name "robot3"
   color "SlateBlue"
   pose [-0.302 -0.045 546.146]
   sick_laser( samples 361 laser_sample_skip 4 )
)

# create a SERB robot
SERB_CAM
(
   name "robot4"
   color "DarkGoldenrod"
   pose [-1.308 -2.385 64.475]
   sick_laser( samples 361 laser_sample_skip 4 )
)
```

```
# create a SERB robot
SERB_CAM
(
   name "robot5"
   color "DarkBlue"
   pose [1.277 -1.859 96.801]
   sick_laser( samples 361 laser_sample_skip 4 )
)

# create a SERB robot
#SERB_CAM
#(
# name "robot6"
# color "red"
# pose [2.139 0.682 401.806]
# sick_laser( samples 361 laser_sample_skip 4 )
#)

# create a SERB robot
#SERB_CAM
#(
# name "robot7"
# color "SlateBlue"
# pose [-0.092 2.142 459.462]
# sick_laser( samples 361 laser_sample_skip 4 )
#)

# create a SERB robot
#SERB_CAM
#(
# name "robot8"
# color "DarkGoldenrod"
# pose [2.460 0.959 269.721]
# sick_laser( samples 361 laser_sample_skip 4 )
#)

# create a SERB robot
#SERB_CAM
#(
# name "robot9"
# color "DarkBlue"
# pose [1.194 -1.496 378.183]
# sick_laser( samples 361 laser_sample_skip 4 )
#)

# create a SERB robot
#SERB_CAM
```

```
#(
# name "robot10"
# color "red"
# pose [1.525 -2.473 154.768]
# sick_laser( samples 361 laser_sample_skip 4 )
#)
```

C.3 serb.inc file

```
# Desc:  Device definitions for SERB robots.
# Author:  Fredy H. Martinez S.
# Date:  14 Feb 2012
# CVS: serb.inc,v 1.0 2012/02/14

# The SERB microswitch like laser sonar.
define serb_switch ranger
(
   scount 16

   # define the pose of each transducer [xpos ypos heading]
   spose[0] [ 0.075 0.130 90 ]
   spose[1] [ 0.115 0.115 50 ]
   spose[2] [ 0.150 0.080 30 ]
   spose[3] [ 0.170 0.025 10 ]
   spose[4] [ 0.170 -0.025 -10 ]
   spose[5] [ 0.150 -0.080 -30 ]
   spose[6] [ 0.115 -0.115 -50 ]
   spose[7] [ 0.075 -0.130 -90 ]
   spose[8] [ -0.155 -0.130 -90 ]
   spose[9] [ -0.195 -0.115 -130 ]
   spose[10] [ -0.230 -0.080 -150 ]
   spose[11] [ -0.250 -0.025 -170 ]
   spose[12] [ -0.250 0.025 170 ]
   spose[13] [ -0.230 0.080 150 ]
   spose[14] [ -0.195 0.115 130 ]
   spose[15] [ -0.155 0.130 90 ]

   # define the field of view of each transducer [range_min range_max view_angle]
   sview [0 5.0 15]

   # define the size of each transducer [xsize ysize] in meters
   ssize [0.01 0.05]
)

# The SERB camera like blobfinder
```

```
define serb_camera ptz
(
   # number of colours to look for
   channels_count 4
   # which colours to look for
   channels ["red" "DarkBlue" "SlateBlue" "DarkGoldenrod"]
   # camera parameters
   image [160 120] # resolution of the image in pixels
   range_max 1.00 # range in meters
   ptz [0 0 0] # [Pan Tilt Zoom]
   size [0.04 0.04]
)

# a SERB in standard configuration
define SERB_CAM position
(
   # actual size
   size [0.26 0.22]

   # the SERB's center of rotation is in its center of area
   origin [-0.00 0.0 0]

   # draw a nose on the robot so we can see which way it points
   gui_nose 1

   # estimated mass in KG
   mass 1.2

   # this polygon approximates the shape of a SERB
   polygons 1
   polygon[0].points 17
   polygon[0].point[0] [ 0.25 0.00 ]
   polygon[0].point[1] [ 0.75 0.00 ]
   polygon[0].point[2] [ 0.75 0.25 ]
   polygon[0].point[3] [ 0.90 0.25 ]
   polygon[0].point[4] [ 0.775 0.00 ]
   polygon[0].point[5] [ 0.975 0.25 ]
   polygon[0].point[6] [ 1.00 0.50 ]
   polygon[0].point[7] [ 0.975 0.75 ]
   polygon[0].point[8] [ 0.775 1.00 ]
   polygon[0].point[9] [ 0.90 0.75 ]
   polygon[0].point[10] [ 0.75 0.75 ]
   polygon[0].point[11] [ 0.75 1.00 ]
   polygon[0].point[12] [ 0.25 1.00 ]
   polygon[0].point[13] [ 0.25 0.875 ]
   polygon[0].point[14] [ 0.00 0.875 ]
   polygon[0].point[15] [ 0.00 0.125 ]
```

```
    polygon[0].point[16] [ 0.25 0.125 ]

    # differential steering model
    drive "diff"

    # use the sonar array for microswitch and blobfinder for camera
    serb_switch()
    serb_camera()
)
```

C.4 mapa_uiuc.inc file

```
define map model
(
    # sombre, sensible, artistic
    # color "black"
    color "gray30"

    # most maps will need a bounding box
    boundary 1

    gui_nose 0
    gui_grid 1
    gui_movemask 1
    gui_outline 0

    gripper_return 0
)
```

C.5 cam_bw.inc file

```
define sick_laser laser
(
    range_min 0.0
    range_max 0.5
    fov 120.0
    samples 361

    color "gray"
    size [ 0.05 0.05 ]
)
```

C.6 grouping.c file

```
// Description:  Grouping robots using local interaction
 // 2011/09/21
 // Fredy H. Martinez S.

 #include <stdio.h>
 #include <stdlib.h>
 #include <time.h>
 #include <libplayerc/playerc.h>

 #define num_robots 10
 #define num_pasos_simulacion 20000

 int angulo_giro();
 float pos_inicial();
 void espera(int);
 float pos2d[num_robots];

 int main(int argc, const char **argv)
 {
    int i;
    int j;
    int k;
    int l;
    int m;
    int n;
    int o;
    int p;
    int q;
    int r;
    int bigote[num_robots];
    int j_in;
    int j_ini;
    float h[num_robots]; //Robot density threshold
    float teta_punto_r[num_robots][num_pasos_simulacion]; //Speed right wheel, robot j step i
    float teta_punto_l[num_robots][num_pasos_simulacion]; //Speed left wheel, robot j step i

    float wr0[num_robots][num_pasos_simulacion]; //Weights for robot and iteration
    float wr[num_robots][num_pasos_simulacion];
    float wlr[num_robots][num_pasos_simulacion];
    float wl0[num_robots][num_pasos_simulacion];
    float wl[num_robots][num_pasos_simulacion];
    float wrl[num_robots][num_pasos_simulacion];
    float sum_h[num_pasos_simulacion];

    playerc_client_t *client;
```

```
   playerc_position2d_t *pos2d[num_robots];
   playerc_laser_t *laser[num_robots];
   playerc_ptz_t *ptz[num_robots];

   // Create a client and connects it to the server, all in 6665.
   client = playerc_client_create(NULL, "localhost", 6665);
   if (0 != playerc_client_connect(client))
      return -1;

   // Create and subscribe (m) agents to position2d (Robot1, Robot2,..).
   for (m = 0; m < num_robots; m++)
   {
      pos2d[m] = playerc_position2d_create(client, m);
      if (playerc_position2d_subscribe(pos2d[m], PLAYER_OPEN_MODE))
         return -1;
   }

   // Create and subscribe (m) laser to the (m) robots.
   for (n = 0; n < num_robots; n++)
   {
      laser[n] = playerc_laser_create(client, n);
      if (playerc_laser_subscribe(laser[n], PLAYER_OPEN_MODE))
         return -1;
}

   // Create and subscribe (m) blobfinders to the (m) robots.
   for (o = 0; o < num_robots; o++)
   {
      ptz[o] = playerc_ptz_create(client, o);
      if (playerc_ptz_subscribe(ptz[o], PLAYER_OPEN_MODE))
         return -1;
   }
   printf(" All Connected!!!\n");

   // Create random initial values ??for the weights
   for (o = 0; o < num_robots; o++)
   {
      wr0[o][0] = pos_inicial();
      wr[o][0] = pos_inicial();
      wlr[o][0] = pos_inicial();
      wl0[o][0] = pos_inicial();
      wl[o][0] = pos_inicial();
      wrl[o][0] = pos_inicial();
      //printf("Agent %i wr0= %f, wr= %f, wrl= %f\n", o, wr0[o][0], wr[o][0], wlr[o][0]);
   }

   // Robot positions!!!!!
```

```
//playerc_position2d_set_cmd_vel(device, vx, vy, va, state);
// vx:  forward speed (m/s).  vy:  sideways speed (m/s).  va:  rotational speed (rad/s).
// All speeds are defined in the robot coordinate system.

// Learning
for (i = 0; i < num_pasos_simulacion; i++)
{
   // Read new data from the server
   playerc_client_read(client);

   // Movement of robots, I examine each robot j
   sum_h[i] = 0.0;
   for (j = 0; j < num_robots; j++)
   {
      teta_punto_r[j][i] = 0.2;
      teta_punto_l[j][i] = 0.2;
      //First, I check the whiskers of all robots:  If 1 it crashed!!!
      bigote[j] = 0;
      for (p = 1; p < 361; p++)
      {
         if (laser[j]->ranges[laser[j]->scan_count-p] < 0.18)
         {
            bigote[j] = 1;
         }
         else
         {
            bigote[j] = bigote[j] + 0;
         }
      }
      //printf("Bigote de %i esta en %i\n", (j+1), bigote[j]);

      //Second, I check if the robot sees someone:  determines density threshold h (0 to 1)!!!!!
      h[j] = 0;
      for (p = 1; p < 361; p++)
      {
         if (laser[j]->ranges[laser[j]->scan_count-p] < 0.5)
         {
            h[j] = h[j] + ((1.0)/360); //Threshold normalized from 0 to 1
         }
         else
         {
            h[j] = h[j] + 0;
         }
      }
      //printf("Agente %i Densidad %f\n", (j+1), h[j]);

      //Third, I dynamically adjust the weights
      sum_h[i] = sum_h[i] + h[j]; // Accumulated density per iteration
```

```
            if (j == num_robots) // If I'm in the last robot of the cycle...
            {
               if ((sum_h[i] - sum_h[i-1]) < -1.0) // If the accumulated density decreased...
               {
                  for (j_in = 0; j_in < num_robots; j_in++)
                  {
               if (h[j_in] > 0.2)
               {
                  wr0[j_in][i] = pos_inicial();
                  wr[j_in][i] = pos_inicial();
                  wlr[j_in][i] = pos_inicial();
                  wl0[j_in][i] = pos_inicial();
                  wl[j_in][i] = pos_inicial();
                  wrl[j_in][i] = pos_inicial();
               }
            }
         }
         else
         {
            for (j_ini = 0; j_ini < num_robots; j_ini++)
            {
               if (h[j_ini] > 0.2 & h[j_ini] < 0.6)
               {
                  wr0[j_ini][i] = wr0 [j_ini][i] * 0.9;
                  wr[j_ini][i] = wr[j_ini][i] * 0.9;
                  wlr[j_ini][i] = wlr[j_ini][i] * 0.9;
                  wl0[j_ini][i] = wl0[j_ini][i] * 0.9;
                  wl[j_ini][i] = wl[j_ini][i] * 0.9;
                  wrl[j_ini][i] = wrl[j_ini][i] * 0.9;
               }
            }
         }
      }


      //Fourth, I calculate the new wheel speed
      if (h[j] > 0 & h[j] < 1 & bigote[j] != 1 & i > 0 & sum_h[i-1] < 3)
      {
         if ( (2*(h[j]-0.5)) * wr0[j][i] + (1.0*teta_punto_r[j][i-1]) * wr[j][i] + 1.0 * teta_punto_l[j][i-1]
* wlr[j][i] < 0)
         {
         teta_punto_r[j][i] = -0.2;
         }
         else
         {
         teta_punto_r[j][i] = 0.2;
         }
         if ( (2*(h[j]-0.5)) * wl0[j][i] + (1.0*teta_punto_l[j][i-1]) * wl[j][i] + 1.0 * teta_punto_r[j][i-1]
* wrl[j][i] < 0)
         {
```

```
          teta_punto_l[j][i] = -0.2;
          }
          else
          {
          teta_punto_l[j][i] = 0.2;
          }
          //printf("Agente %i, paso %i, h= %f, teta_r %f, teta_l= %f\n", j, i, (2*(h[j]-0.5)), teta_punto_r[j][i],
teta_punto_l[j][i] );
          }
          else
          {
          }
          if (h[j] > 0.2 & h[j] < 0.5 & bigote[j] != 1 & i > 0 & sum_h[i-1] > 3)
          {
             teta_punto_r[j][i] = 0.2;
             teta_punto_l[j][i] = 0.2;
          }
          if (h[j] == 0 & bigote[j] != 1 & i > 0 & sum_h[i-1] > 3)
          {
             teta_punto_r[j][i] = -0.2;
             teta_punto_l[j][i] = 0.2;
          }
          //printf("Agente %i, bigote %i, teta_r %f, teta_l= %f\n", j, bigote[j], teta_punto_r[j][i],
teta_punto_l[j][i] );


          //Fourth, I turn randomly if the whiskers found obstacle or according training
          if (bigote[j] == 1)
          {
          if (playerc_position2d_set_cmd_vel(pos2d[j], -0.2, 0, DTOR(angulo_giro()), 1) != 0) return -1;
          //if (playerc_position2d_set_cmd_pose(pos2d[j], 0, 0, DTOR(angulo_giro()), 1)!= 0) return -1;
          }
          else
          {
             //if (teta_punto_r[j][i] == 0.200000 & teta_punto_l[j][i] == 0.200000)
             if (teta_punto_r[j][i] + teta_punto_l[j][i] > 0.35)
                {
                playerc_position2d_set_cmd_vel(pos2d[j], 0.2, 0, 0, 1);
                //printf("Agente %i, bigote %i, teta_r %f, teta_l= %f\n", j, bigote[j], teta_punto_r[j][i],
teta_punto_l[j][i] );
             }
             //else if (teta_punto_r[j][i] == -0.200000 & teta_punto_l[j][i] == -0.200000)
             else if (teta_punto_r[j][i] + teta_punto_l[j][i] < -0.35)
             {
                playerc_position2d_set_cmd_vel(pos2d[j], -0.2, 0, 0, 1);
                //printf("Agente %i, bigote %i, teta_r %f, teta_l= %f\n", j, bigote[j], teta_punto_r[j][i],
teta_punto_l[j][i] );
             }
             //else if (teta_punto_r[j][i] == -0.200000 & teta_punto_l[j][i] == 0.200000)
             else if (teta_punto_r[j][i] < 0)
```

```
            {
               playerc_position2d_set_cmd_vel(pos2d[j], 0, 0, -1, 1);
               //printf("Agente %i, bigote %i, teta_r %f, teta_l= %f\n", j, bigote[j], teta_punto_r[j][i],
teta_punto_l[j][i] );
            }
            //else if (teta_punto_r[j][i] == 0.200000 & teta_punto_l[j][i] == -0.200000)
            else if (teta_punto_l[j][i] < 0)
            {
               playerc_position2d_set_cmd_vel(pos2d[j], 0, 0, 1, 1);
               //printf("Agente %i, bigote %i, teta_r %f, teta_l= %f\n", j, bigote[j], teta_punto_r[j][i],
teta_punto_l[j][i] );
            }
            else
            {
               playerc_position2d_set_cmd_vel(pos2d[j], 0, 0, 0, 1);
            }
         }

      } // End of movement of the robots!!!!!!!!!!!!
      printf("PASO %i Densidad Acumulada %f\n", (i), sum_h[i]);
      if (i == 10000)
      {
         i = 0;
      }

   } // End of simulation!!!!!!!!!!!!!!!
   printf(" Terminado aprendizaje!!!\n");

   // Shutdown
   for (l = 0; l < num_robots; l++)
   {
      playerc_laser_unsubscribe(laser[l]);
      playerc_laser_destroy(laser[l]);
   }
   for (k = 0; k < num_robots; k++)
   {
      playerc_position2d_unsubscribe(pos2d[k]);
      playerc_position2d_destroy(pos2d[k]);
   }
   for (r = 0; r < num_robots; r++)
   {
      playerc_ptz_unsubscribe(ptz[r]);
      playerc_ptz_destroy(ptz[r]);
   }
   playerc_client_disconnect(client);
   playerc_client_destroy(client);
   printf(" Todo desconectado!!!\n");
   return 0;
```

```
}

// Generation of random angle between -360 and 360 to rotation
int angulo_giro()
{
   int aleatorio;
   int M;
   int N;
   M = -360;
   N = 360;
   aleatorio = (rand () % (N-M+1) + M); // Generates random number between M and N
   return aleatorio;
}

// Generation of random initial values ??between 1 and -1 for weights
float pos_inicial()
{
   float pos_x;
   int M;
   int N;
   M = -100;
   N = 100;
   pos_x = ((rand () % (N-M+1) + M) * (1.0))/100; // Generates random number between M and N
   return pos_x;
}

// Delay in milliseconds
void espera(int mseg)
{
   clock_t t = mseg + clock();
   while (t > clock());
}
```

APPENDIX D

NetLogo Simulations

In this appendix, we show the code used in implementing the navigation scheme without and with our bacterial Quorum Sensing algorithm used in Chapter 3 in the performance analysis. This code was modified to obtain different simulation conditions.

```
;; Fredy H. Martínez S.
;; March 2013, Bogotá D.C., Colombia
;; Path navigation with landmarks with Quorum Sensing

breed [reproductive]; Agents that move and activate inactive agents (R).
breed [explorer]; Agents that monitor local information, but are not virulent (X).
breed [virulent]; Virulent agents, they apply the quorum sensing (Z).
breed [landmark]; Landmark in the environment

globals
[
   total-reproductive ; Total number of agents that are reproductives
   total-explorer ; Total number of agents that are explorers
   total-virulent ; Total number of agents that are virulents
   turn-check ; Holds the random variable for the wander sub-routine
   wall-turn-check ; Holds the random variable for the wall sub-routine
   explorer-vision; Range vision of explorers
   Inten; Intensity of the speed field in a landmark
   Dc; Mass diffusivity
]

turtles-own
[
   seek-virul; Agent-subset consisting of virulent agents within vision radius of a single explorer
   seek-explo; Agent-subset consisting of explorer agents within vision radius of a single explorer
   seek-landk; Agent-subset consisting of landmarks within vision radius of a single explorer
   nearest-virul; Variable that holds the target virulent for a single explorer
```

```
   nearest-landk; Variable that holds the target landmark for a single explorer
   intensity; Intensity of the speed field in the landmark.
   T; Quorum Sensing threshold
]

to Setup
   ; Environment size 6m x 6m, each block is 0.2m
   ; Origin in the lower left corner
   set Dc 0.4
   ask patches [ set pcolor white ]; Free space in white
   update-globals
end

to Obstacles
   ; Creates obstacles in the environment
   ask patch 1 30 [ set pcolor black ]; Obstacle 1
   ask patch 1 29 [ set pcolor black ]
   ask patch 1 28 [ set pcolor black ]
   ask patch 1 27 [ set pcolor black ]
   ask patch 2 30 [ set pcolor black ]
   ask patch 2 29 [ set pcolor black ]
   ask patch 2 28 [ set pcolor black ]
   ask patch 2 27 [ set pcolor black ]
   ask patch 3 30 [ set pcolor black ]
   ask patch 3 29 [ set pcolor black ]
   ask patch 3 28 [ set pcolor black ]
   ask patch 3 27 [ set pcolor black ]

   ask patch 10 20 [ set pcolor black ]; Obstacle 2
   ask patch 10 19 [ set pcolor black ]
   ask patch 10 18 [ set pcolor black ]
   ask patch 10 17 [ set pcolor black ]
   ask patch 11 20 [ set pcolor black ]
   ask patch 11 19 [ set pcolor black ]
   ask patch 11 18 [ set pcolor black ]
   ask patch 11 17 [ set pcolor black ]
   ask patch 12 20 [ set pcolor black ]
   ask patch 12 19 [ set pcolor black ]
   ask patch 12 18 [ set pcolor black ]
   ask patch 12 17 [ set pcolor black ]

   ask patch 28 4 [ set pcolor black ]; Obstacle 3
   ask patch 28 3 [ set pcolor black ]
   ask patch 28 2 [ set pcolor black ]
   ask patch 28 1 [ set pcolor black ]
   ask patch 29 4 [ set pcolor black ]
   ask patch 29 3 [ set pcolor black ]
   ask patch 29 2 [ set pcolor black ]
```

```
   ask patch 29 1 [ set pcolor black ]
   ask patch 30 4 [ set pcolor black ]
   ask patch 30 3 [ set pcolor black ]
   ask patch 30 2 [ set pcolor black ]
   ask patch 30 1 [ set pcolor black ]

   let n0 0; Lower limit of the environment
   while [n0 < 32]
   [
      ask patch n0 0 [ set pcolor black ]
      set n0 (n0 + 1)
   ]

   let n1 0; Upper limit of the environment
   while [n1 < 32]
   [
      ask patch n1 31 [ set pcolor black ]
      set n1 (n1 + 1)
   ]

   let n2 0; Right limit of the environment
   while [n2 < 32]
   [
      ask patch 31 n2 [ set pcolor black ]
      set n2 (n2 + 1)
   ]

   let n3 0; Left limit of the environment
   while [n3 < 32]
   [
      ask patch 0 n3 [ set pcolor black ]
      set n3 (n3 + 1)
   ]
end

to Agents
   ; Creates agents in the free space of the environment
   clear-turtles
   ;let NumAgents 20; Number of agents to create (replaces slider)
   create-reproductive NumAgents
   ask turtles [setxy ((random 30) + 1) ((random 30) + 1)]; Random starting position

   ; Moves the agent if it is over other agent or an obstacle
   ask turtles
      [
         while [ any?  other turtles-here ]
         [
```

```
        rt random 360
        fd 1
      ]
   ]
ask turtles
   [
      while [ [pcolor] of patch xcor ycor = black ]
      [
        rt random 360
        fd 1
      ]
   ]
ask turtles; Initially all agents are reproductive and blue
   [
      set color blue
   ]

set-default-shape landmark "circle"; Lanmarks in the environment
create-landmark 1 [setxy (5 * 5) (4 * 5) set color orange set size 0.5 set intensity 0.00]
; Position (5,4), cada tile is 0.2m.  Target!!!
create-landmark 1 [setxy (0.8 * 5) (5.2 * 5) set color 27 set size 0.5 set intensity 1.72]; Obst1
create-landmark 1 [setxy (0.4 * 5) (5.2 * 5) set color 27 set size 0.5 set intensity 1.98]; Obst1
create-landmark 1 [setxy (0.8 * 5) (6.0 * 5) set color 27 set size 0.5 set intensity 2.04]; Obst1

create-landmark 1 [setxy (1.8 * 5) (4.2 * 5) set color 27 set size 0.5 set intensity 1.28]; Obst2
create-landmark 1 [setxy (2.6 * 5) (4.2 * 5) set color 27 set size 0.5 set intensity 0.96]; Obst2
create-landmark 1 [setxy (1.8 * 5) (3.2 * 5) set color 27 set size 0.5 set intensity 1.08]; Obst2
create-landmark 1 [setxy (2.6 * 5) (3.2 * 5) set color 27 set size 0.5 set intensity 0.68]; Obst2

create-landmark 1 [setxy (5.4 * 5) (1.0 * 5) set color 27 set size 0.5 set intensity 0.50]; Obst3
create-landmark 1 [setxy (5.8 * 5) (1.0 * 5) set color 27 set size 0.5 set intensity 0.46]; Obst3
create-landmark 1 [setxy (5.4 * 5) (0.4 * 5) set color 27 set size 0.5 set intensity 0.62]; Obst3

create-landmark 1 [setxy (3.5 * 5) (0.8 * 5) set color 27 set size 0.5 set intensity 0.79]; Random
create-landmark 1 [setxy (3.6 * 5) (3.9 * 5) set color 27 set size 0.5 set intensity 0.54]
create-landmark 1 [setxy (0.3 * 5) (0.4 * 5) set color 27 set size 0.5 set intensity 1.23]
create-landmark 1 [setxy (5.8 * 5) (4.8 * 5) set color 27 set size 0.5 set intensity 0.10]
create-landmark 1 [setxy (4.6 * 5) (3.7 * 5) set color 27 set size 0.5 set intensity 0.10]
create-landmark 1 [setxy (5.2 * 5) (2.0 * 5) set color 27 set size 0.5 set intensity 0.38]
create-landmark 1 [setxy (0.5 * 5) (3.4 * 5) set color 27 set size 0.5 set intensity 1.68]
create-landmark 1 [setxy (3.5 * 5) (4.8 * 5) set color 27 set size 0.5 set intensity 0.76]
create-landmark 1 [setxy (2.8 * 5) (0.3 * 5) set color 27 set size 0.5 set intensity 0.96]
create-landmark 1 [setxy (0.6 * 5) (1.6 * 5) set color 27 set size 0.5 set intensity 1.28]
create-landmark 1 [setxy (4.2 * 5) (2.2 * 5) set color 27 set size 0.5 set intensity 0.44]
create-landmark 1 [setxy (2.8 * 5) (5.8 * 5) set color 27 set size 0.5 set intensity 1.22]
create-landmark 1 [setxy (4.6 * 5) (6.0 * 5) set color 27 set size 0.5 set intensity 0.56]
```

```
   set explorer-vision 5
   ;ask turtles [ pd ]; Pen down so we can see the path
end

to Navigate
   ; Navigation algorithm
   exploration-check; Reproductive to explorers
   virulence-check; Explorers to virulents
   bacterial-attack-check; Virulent agents transmitting quorum signal
   update-globals
end

to bacterial-attack-check; Virulents
   ask explorer
   [
      set seek-explo explorer in-radius explorer-vision
      ; Adds all explorer in vision radius of explorer to an agent subset for that agent
      set T 6; Quorum Sensing threshold
      if any?  seek-explo
      [
         if any?  other turtles-here with [color = orange]; Target
         [
            if count seek-explo > (T - 1) [become-virul]; Quorum Sensing threshold T = 6
         ]
      ]
      set seek-virul virulent in-radius explorer-vision
      ; Adds all virulent in vision radius of explorer to an agent subset for that agent
      if any?  seek-virul [run-toward]
   ]
   ask virulent
   [
      set seek-virul virulent in-radius explorer-vision
      ; Adds all virulent in vision radius of explorer to an agent subset for that agent
      if any?  seek-virul [run-toward]
      if any?  seek-virul [local-path-planning]
   ]
end

to run-toward
   set nearest-virul min-one-of seek-virul [distance myself]
   face nearest-virul
   if [pcolor] of patch-ahead 1 != white [wall]
end

to virulence-check ; Test if explorers are near a virulent agent and must become virulent
   ask explorer
   [
      if any?  other turtles-here with [color = red]
```

```
      [ become-virul ]

      ifelse any?  other turtles-here with [color = green]
      [local-path-planning]
      [local-path-planning]
   ]
end

to become-virul ; The agent becomes virulent
   ask explorer-on patch-here[set breed virulent]
   ask virulent-on patch-here [set color red]
end

to local-path-planning; Navigation plan (Brownian motion)
   ask explorer
      [
         while [ [pcolor] of patch xcor ycor = black ]; Obstacle
         [
            rt random 360
            fd 0.1
         ]
      ]
   ask explorer
   [
      if any?  other turtles-here with [color = 27]; Landmark
      [
         rt 180
         ask landmark-on patch-here [set Inten intensity]
         ask explorer-on patch-here [set intensity Inten]
         rt random 360
         fd ( (intensity + 0.5) + Dc)
      ]
   ]
   ask explorer
   [
      ifelse any?  other turtles-here with [color = orange]; Target
         [
            rt random 360
            fd 0
         ]
         [fd 0.0002]
   ]
   ask virulent
   [
      ifelse any?  other turtles-here with [color = orange]; Target
         [
            rt random 360
            fd 0
```

```
        ]
        [fd 0.0002]
   ]
end

to run-mark
   set nearest-landk min-one-of seek-landk [distance myself]
   face nearest-landk
   if [pcolor] of patch-ahead 1 != white [wall]
end

to exploration-check ; Test if reproductive agents are ready to become explorer agents
   tick

   ask explorer
   [
      set seek-landk landmark in-radius (explorer-vision - (explorer-vision - 1) )
      ; Adds all landmark in vision radius of explorer to an agent subset for that agent
      if any?  seek-landk [run-mark]
      if any?  seek-landk [local-path-planning]
   ]
   ask reproductive
   [
      if ticks > 20 [ become-explor ]
   ]
   ask reproductive
   [
      if ticks < 20 [ wander ]
   ]
end

to become-explor ; The agent becomes explorer
   ask reproductive-on patch-here[set breed explorer]
   ask explorer-on patch-here [set color green]
end

to wander; If an explorer agent is not activating agents, have it wander around aimlessly
   set turn-check random 15
   if turn-check > 10 [right-turn]
   if turn-check < 5 [left-turn]
   ask reproductive
   [
      if [pcolor] of patch-ahead 1 != white [wall]; Obstacle
   ]
   ;ask reproductive ; The bacterium reproduces
   ; [
   ; if any?  other turtles-here with [color = gray]
```

```
   ; [
   ; activate-agent
   ; ]
   ; ]
end

to right-turn; Generate a random degree of turn for the wander sub-routine
   right random-float 10
end

to left-turn;G enerate a random degree of turn for the wander sub-routine
   left random-float 10
end

to wall; Turn the agent away from wall
   set wall-turn-check random 10
   if wall-turn-check >= 6 [wall-right-turn]
   if wall-turn-check <= 5 [wall-left-turn]
end

to wall-right-turn; Generate a random degree of turn for the wall sub-routine
   right 150
end

to wall-left-turn; Generate a random degree of turn for the wall sub-routine
   left 150
end

to update-globals; Set globals to current values for reporters.
   set total-reproductive (count reproductive)
   set total-explorer (count explorer)
   set total-virulent (count virulent)
end
```

References

Alur, R., Henzinger, T. A., Lafferriere, G., & Pappas, G. J. (2002). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, *88*(7), 971-984.

Apolloni, B., Bassis, S., Clivio, A., Gaito, S., & Malchiodi, D. (2007). Modeling individual′s aging within a bacterial population using a pi calculus paradigm. *Natural Computing*, *6*, 33-53.

Arbib, M. A. (2003). *The handbook of brain theory and neural networks* (S. Edition, Ed.). MIT Press.

Åström, K. J. (2007). Event based control. In *Analysis and design of nonlinear control systems: In honor of alberto isidori.* Springer Verlag.

Atakan, B., Akan, O. B., & Tugcu, T. (2009). Bio-inspired communications in wireless sensor networks. In *Guide to wireless sensor networks* (p. 659-685). Springer London.

Ayanian, N., & Kumar, V. (2010). Abstractions and controllers for groups of robots in environments with obstacles. In *2010 ieee international conference on robotics and automation (icra)* (p. 3537-3542).

Belta, C., & Kumar, V. (2004, Oct.). Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, *20*(5), 865-875. (ISSN 1552-3098)

Berman, S., Halász, A., Kumar, V., & Pratt, S. (2007a). Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. In E. Sahin, W. Spears, & A. Winfield (Eds.), *Swarm robotics* (Vol. 4433, p. 56-70). Springer Berlin Heidelberg.

Berman, S., Halász, A., Kumar, V., & Pratt, S. (2007b, April). Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In *Ieee international conference on robotics and automation icra 2007* (p. 2318-2323). (ISBN 1-4244-0601-3)

Bernardini, F., Gheorghe, M., Krasnogor, N., Muniyandi, R., Pérez, M., & Romero, F. (2006). On p systems as a modelling tool for biological systems. In R. Freund, G. Paun, G. Rozenberg, & A. Salomaa (Eds.), *Membrane computing* (Vol. 3850, p. 114-133). Springer Berlin Heidelberg.

Bhalla, N., & Bentley, P. J. (2006). Working towards self-assembling robots at all scales. In *Proc. of the 3rd int. conf. on autonomous robots and agents* (p. 617-622).

Bhatia, A., Kavraki, L. E., & Vardi, M. Y. (2010). Sampling-based motion planning with temporal goals. In *Proceedings ieee international conference on robotics & automation* (p. 2689-2696).

Bianco, L., Pescini, D., Siepmann, P., Krasnogor, N., Romero, F., & Gheorghe, M. (2006). Towards a p systems pseudomonas quorum sensing model. In H. Hoogeboom, G. Paun, G. Rozenberg, & A. Salomaa (Eds.), *Membrane computing* (Vol. 4361, p. 197-214). Springer Berlin / Heidelberg.

Bobabilla, L., & Martinez S., F. H. (2011, October). Wild robots and virtual gates. In C. S. L. U. of Illinois Urbana-Champaign (Ed.), *2011 symposium on control & modeling cyber-physical systems.*

Bobadilla, L., Gossman, K., & LaValle, S. M. (2011). Manipulating ergodic bodies through gentle guidance. In *Ieee conference on robot motion and control.*

Bobadilla, L., Martinez, F., Gobst, E., Gossman, K., & LaValle, S. (2011a, September 24). *Controlling wild mobile robots using virtual gates and discrete transitions.* On line. Retrieved from `http://msl.cs.uiuc.edu/virtualgates/`

Bobadilla, L., Martinez, F., Gobst, E., Gossman, K., & LaValle, S. (2011b, September 24). *Navigation.* On line. Retrieved from http://www.youtube.com/watch?v=0IHPcLSV1GE&feature=channel&list=UL

Bobadilla, L., Martinez, F. H., Gobst, E., Gossman, K., & LaValle, S. (2012, June). Controlling wild mobile robots using virtual gates and discrete transitions. In *Proc. 2012 ieee american control conference (acc).*

Bobadilla, L., Sanchez, O., Czarnowski, J., Gossman, K., & LaValle, S. M. (2011). Controlling wild bodies using linear temporal logic. In *Proceedings robotics: Science and systems.*

Bobadilla, L., Sanchez, O., Czarnowski, J., & LaValle, S. M. (2011). Minimalist multiple target tracking using directional sensor beams. In *Proc. ieee/rsj int intelligent robots and systems (iros) conf* (pp. 3101–3107).

Boncheva, M., Gracias, D. H., Jacobs, H. O., & Whitesides, G. M. (2002). Biomimetic self-assembly of a functional asymmetrical electronic device. *Proc Natl Acad Sci U S A*, *99*(8), 4937-40.

Branicky, M. S., Borkar, V. S., & Mitter, S. K. (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, *43*(1), 31-45.

Brogan, D., & Hodgins, J. (1997). Group behaviors for systems with significant dynamics. *Autonomous Robots*, *4* , 137-153.

Brutschy, A., Scheidler, A., Merkle, D., & Middendorf, M. (2008). Learning from house-hunting ants: Collective decision-making in organic computing systems. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. StÃ$\frac{1}{4}$tzle, & A. Winfield (Eds.), *Ant colony optimization and swarm intelligence* (Vol. 5217, p. 96-107). Springer Berlin Heidelberg.

Bunimovich, L. A. (1979). On the ergodic properties of nowhere dispersing billiards. *Communcations in Mathematical Physics*, *65*, 295-312.

Busby, S. J. W., & Lorenzo, V. de. (2001). Cell regulation: putting together pieces of the big puzzle. *Current Opinion in Microbiology*, *4* , 117-118. (ISSN 1369-5274)

Camara, M. (2006). Quorum sensing: A cell-cell signalling mechanism used to coordinate behavioral changes in bacterial populations. In H. Hoogeboom, G. Paun, G. Rozenberg, & A. Salomaa (Eds.), *Membrane computing* (Vol. 4361, p. 42-48). Springer Berlin Heidelberg.

Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-organization in biological systems.* Princeton University Press. (ISBN 978-0691012117)

Cho, J. H., & Kim, D. H. (2011). Intelligent feature selection by bacterial foraging algorithm and information theory. In T.-h. Kim, H. Adeli, R. J. Robles, & M. Balitanas (Eds.), *Advanced communication and networking* (Vol. 199, p. 238-244). Springer Berlin Heidelberg.

Cho, J. H., Park, J. I., Jeong, J. S., & Chun, M. G. (2009, August). Bacterial foraging with quorum sensing based optimization algorithm. In *Proc. ieee international conference on fuzzy systems fuzz-ieee 2009* (p. 29-34). (ISBN 978-1-4244-3596-8)

Dang, J., Brabazon, A., O'Neill, M., & Edelman, D. (2008). Option model calibration using a bacterial foraging optimization algorithm. In M. Giacobini & A. Brabazon (Eds.), *Applications of evolutionary computing* (Vol. 4974, p. 113-122). Springer Berlin Heidelberg.

Delgado, A., Kambhampati, C., & Warwick, K. (1995). Dynamic recurrent neural net-

work for system identification and control. *IEE Proceedings -Control Theory and Applications*, *142*(4), 307–314.

Egerstedt, M., & Hu, X. (2001, January). A hybrid control approach to action coordination for mobile robots. *Automatica*, *38*(1), 125-130.

Egerstedt, M., & Hu, X. (2002). A hybrid control approach to action coordination for mobile robots. *Automatica*, *38*(1), 125-130.

Einstein, A. (1905, May). über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der Physik*, *17*, 549-560.

Fainekos, G. E. (2011). Revising temporal logic specifications for motion planning. In *Proceedings ieee international conference on robotics & automation.*

Fainekos, G. E., Girard, A., Kress-Gazit, H., & Pappas, G. J. (2009, February). Temporal logic motion planning for dynamic mobile robots. *Automatica*, *45*(2), 343-352.

Fierro, R., Das, A., Kumar, V., & Ostrowski, J. P. (2001). Hybrid control of formations of robots. In *Proceedings ieee international conference on robotics & automation* (p. 157-162).

Filho, E., & Pimenta, L. (2015). Segregating multiple groups of heterogeneous units in robot swarms using abstractions. In *Ieee/rsj international conference on intelligent robots and systems (iros).*

Fink, J., Hsieh, M. A., & Kumar, V. (2008). Multi-robot manipulation via caging in environments with obstacles. In *Proc. ieee int. conf. robotics and automation icra 2008* (p. 1471-1476).

Finucane, C., Jing, G., & Kress-Gazit, H. (2010). LTLMoP: Experimenting with language, temporal logic and robot control. In *Proceedings ieee/rsj international conference on intelligent robots and systems* (p. 1988-1993).

Fisher, R. (1925). *Statistical methods for research workers* (1st ed.). Macmillan Pub Co. (ISBN 0-05-002170-2)

Foley, C., Balasubramaniam, S., Power, E., Leon, M. de, Botvich, D., Dudkowski, D., et al. (2008). A framework for in-network management in heterogeneous future communication networks. In S. van der Meer, M. Burgess, & S. Denazis (Eds.), *Modelling autonomic communications environments* (Vol. 5276, p. 14-25). Springer Berlin Heidelberg.

Franks, N., Pratt, S., Mallon, E., Britton, N., & Sumpter, D. (2002, November). Information flow, opinion polling and collective intelligence in house-hunting social insects. *Philosophical Transactions of The Royal Society B Biological Sciences*, *357*(1427), 1567-1583.

Frazzoli, E., Dahleh, M. A., & Feron, E. (1999). *Robust hybrid control for autonomous vehicles motion planning* (Tech. Rep. No. LIDS-P-2468). Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

Freitas, R., & Gilbreath, W. (1980). *Advanced automation for space missions.* 1980 NASA/ASEE Summer Study. (Washington D.C.)

Gabrielli, A., & Mancini, L. (2008). Bio-inspired topology maintenance protocols for secure wireless sensor networks. In P. Lio, E. Yoneki, J. Crowcroft, & D. Verma (Eds.), *Bio-inspired computing and communication* (Vol. 5151, p. 399-410). Springer Berlin Heidelberg.

Gerkey, B., Vaughan, R. T., & Howard, A. (2003, June). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings ieee 11th international*

conference on advanced robotics icar'03 (pp. 317–323). Coimbra (Portugal).

Goni, A., Redondo, M., Arroyo, F., & Castellanos, J. (2011). Biocircuit design through engineering bacterial logic gates. *Natural Computing*, *10*, 119-127.

Grandi, R., Falconi, R., & Melchiorri, C. (2013). Coordination and control of autonomous mobile robot groups using a hybrid technique based on particle swarm optimization and consensus. In *2013 ieee international conference on robotics and biomimetics (robio)* (p. 1514-1519).

Greenwood, G., & Tyrrell, A. (2006). *Introduction to evolvable hardware: A practical guide for designing self-adaptive systems* (I. P. S. on Computational Intelligence, Ed.). John Wiley and Sons. (ISBN 978-0-471-71977-9)

Gross, R., & Dorigo, M. (2008). Self-assembly at the macroscopic scale. , *96*(9), 1490–1508.

Gutierrez, R. L., & Huhns, M. (2008). Multiagent based fault tolerance management for robustness. In A. Schuster (Ed.), *Robust intelligent systems* (p. 23-41). Springer London.

Guzmán, M. A., Delgado, A., & De Carvalho, J. (2010, April). A novel multiobjective optimization algorithm based on bacterial chemotaxis. *Eng. Appl. Artif. Intell.*, *23*(3), 292-301. Retrieved from `http://dx.doi.org/10.1016/j.engappai.2009.09.010`

Haghighi, R., & Cheah, C. (2011). Asynchronous dynamic multi-group formation for swarm robots. In *2011 50th ieee conference on decision and control and european control conference (cdc-ecc)* (p. 2744-2749).

Haghverdi, E., Tabuada, P., & Pappas, G. J. (2005, September). Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, *342*(2-3), 229-261.

Jacinto, E., Martínez, F., & Martínez, F. (2013). A comparative study of geometric path planning methods for a mobile robot: Potential field and voronoi diagrams. In *2013 ii international congress of engineering mechatronics and automation (ciima)* (p. 1-6).

Ji, T. Y., Li, M. S., Lu, Z., & Wu, Q. H. (2008). Optimal morphological filter design using a bacterial swarming algorithm. In *Proc. (ieee world congress computational intelligence). ieee congress evolutionary computation cec 2008* (pp. 452–458).

Ji, T. Y., Li, M. S., Wu, Q. H., & Jiang, L. (2011). Optimal estimation of harmonics in a dynamic environment using an adaptive bacterial swarming algorithm. *IET Generation, Transmission & Distribution*, *5*(6), 609–620.

Jiang, J.-R., Tseng, Y.-C., Hsu, C.-S., & Lai, T.-H. (2005). Quorum-based asynchronous power-saving protocols for ieee 802.11 ad hoc networks. *Mobile Networks and Applications*, *10*, 169-181.

Juba, B., Kalai, A. T., Khanna, S., & Sudan, M. (2011). Compression without a common prior: an information-theoretic justification for ambiguity in language. In *Ics* (p. 79-86).

Karafyllidis, I. (2012, March). Quantum gate circuit model of signal integration in bacterial quorum sensing. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(2), 571-579. (Early Access)

Kazadi, S., Abdul-Khaliq, A., & Goodman, R. (2002, February). On the convergence of puck clustering systems. *Robotics and Autonomous Systems*, *38*(2), 93-117. (ISSN 0921-8890)

Khosravi, S., Jahangir, M., & Afkhami, H. (2012). Adaptive fuzzy SMC-Based formation design for swarm of unknown time-delayed robots. *Nonlinear Dynamics*, *69*(4), 1825-1835.

Kloetzer, M., & Belta, C. (2010). Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics and Automation*, *26*(1), 48-61.

Kössl, M., & Russell, I. J. (1995). Basilar membrane resonance in the cochlea of the mustached bat. *Proceedings of the National Academy of Sciences*, *92*(1), 276-279. Retrieved from `http://www.pnas.org/content/92/1/276.abstract`

Koza, J., Keane, M., & Streeter, M. (2003, July). The importance of reuse and development in evolvable hardware. In *Conference nasa of evolution hardware* (p. 33-42). (ISBN 0-7695-1977-6)

Kreimer, A., Borenstein, E., Gophna, U., & Ruppin, E. (2007). The evolution of modularity in bacterial metabolic networks. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(19), 6976-6981. (doi: 10.1073/pnas.0712149105)

Kress-Gazit, H., Fainekos, G., & Pappas, G. (2007). Where's Waldo? sensor-based temporal logic motion planning. In *Proceedings ieee international conference on robotics and automation.*

Kress-Gazit, H., Fainekos, G. E., & Pappas, G. J. (2009, December). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics and Automation*, *25*(6), 1370-1381.

Kress-Gazit, H., Fainekos, G., & Pappas, G. (2005). Temporal logic motion planning for mobile robots. In *Proceedings ieee international conference on robotics and automation.*

Kress-Gazit, H., Fainekos, G. E., & Pappas, G. J. (2008). Translating structured english to robot controllers. *Advanced Robotics*, *22*(12), 1343-1359.

Lahijanian, M., Wasniewski, J., Andersson, S. B., & Belta, C. (2010). Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proceedings ieee international conference on robotics & automation* (p. 3227-3232).

LaValle, S. M. (2006). *Planning algorithms.* Cambridge, U.K.: Cambridge University Press. (Available at http://planning.cs.uiuc.edu/)

LaValle, S. M. (2011). Sensing and filtering: A tutorial based on preimages and information spaces. *Foundations and Trends in Robotics.* (To appear)

Lee, E. (2008). Cyber physical systems: Design challenges. In *2008 11th ieee international symposium on object oriented real-time distributed computing (isorc)* (p. 363-369).

Lees, M., Logan, B., & King, J. (2007). Multiscale models of bacterial populations. In *Proc. winter simulation conference* (p. 881-890).

Lerman, K., Jones, C., Galstyan, A., & Matarić, M. (2006, March). Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, *25*(3), 225-241. (ISSN 0278-3649)

Li, M., Tang, W., Tang, W., Wu, Q., & Saunders, J. (2007). Bacterial foraging algorithm with varying population for optimal power flow. In M. Giacobini (Ed.), *Applications of evolutionary computing* (Vol. 4448, p. 32-41). Springer Berlin Heidelberg.

Liu, D.-D., & Jia, X.-H. (2008). Location service, information dissemination and object tracking in wireless sensor networks by using quorum methods. In Y. Li, M. T. Thai, & W. Wu (Eds.), *Wireless sensor networks and applications* (p. 235-258). Springer US.

Macias, N. J., & Durbeck, L. J. K. (2005, July). A hardware implementation of the cell matrix self-configurable architecture: the cell matrix mod 88. In *2005 nasa/dod conference on evolvable hardware* (p. 103-106). (ISBN 0-7695-2399-4)

Mange, D., Goeke, M., Madon, D., Stauer, A., Tempesti, G., & Durand, S. (1996). *Embryonics: A new family of coarse-grained field programmable gate array with self-repair and self-reproducing properties* (Vol. 1062; L. N. in Computer Science, Ed.). Springer.

Martinez S., F. H. (2011a, December 11). *Quorum sensing-based navigation - grouping of the robots.* On line. Retrieved from http://www.youtube.com/watch?v=BPy9Izvt-CA

Martinez S., F. H. (2011b, December 11). *Quorum sensing-based navigation - reproduction 1.* On line. Retrieved from http://www.youtube.com/watch?v=TabO1unzONS

Martinez S., F. H. (2012a, April 16). *Quorum sensing-based navigation - coverage simulation.* On line. Retrieved from http://youtu.be/-G2REr8-2Fk

Martinez S., F. H. (2012b, April 18). *Quorum sensing-based navigation - grouping simulation.* On line. Retrieved from http://youtu.be/Lc4DUSW-BFo

Martinez S., F. H. (2013, March 6). *Serb robot with camera.* On line. Retrieved from http://youtu.be/GqW9YcIttRM

Martinez S., F. H., & Delgado, J. (2010, August). Hardware emulation of bacterial quorum sensing. In D.-S. Huang, Z. Zhao, V. Bevilacqua, & J. Figueroa (Eds.), *Lecture notes in computer science 6215. advanced intelligent computing theories and applications* (Vol. 6215, p. 329-336). Springer Berlin Heidelberg.

Miller, G., Galanter, E., & Pribram, K. (1960). *Plans and the structure of behavior.* Holt. Retrieved from http://books.google.com/books?id=jMxJAAAAMAAJ

Min, H., & Wang, Z. (2010). Group escape behavior of multiple mobile robot system by mimicking fish schools. In *2010 ieee international conference on robotics and biomimetics (robio)* (p. 320-326).

Min, H., & Wang, Z. (2011). Design and analysis of group escape behavior for distributed autonomous mobile robots. In *2011 ieee international conference on robotics and automation (icra)* (p. 6128-6135).

Moon, W.-S., Jang, J. W., & Baek, K. R. (2008). Evolutional interactivity in a swarm of robots. In *Proc. int. conf. control, automation and systems iccas 2008* (p. 118-122).

Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., & Kokaji, S. (2001). Self-repairing mechanical systems. *Autonomous Robots*, *10*, 7-21.

Nagpal, R., & Mamei, M. (2004). Engineering amorphous computing systems. In F. Bergenti, M.-P. Gleizes, F. Zambonelli, & G. Weiss (Eds.), *Methodologies and software engineering for agent systems* (Vol. 11, p. 303-320). Springer US.

Niemczyk, S., & Geihs, K. (2015). Adaptive run-time models for groups of autonomous robots. In *10th international symposium on software engineering for adaptive and self-managing systems.*

Oomlout. (2013, March 1). *Arduino controlled servo robot.* On line. Vancouver, British Columbia. Retrieved from http://oomlout.com/a/products/serb/

Ortega-Sanchez, C., Mange, D., Smith, S., & Tyrrell, A. (2000, July). Embryonics: A bio-inspired cellular architecture with fault-tolerant properties. *Genetic Programming and Evolvable Machines*, *1*(3), 187-215.

Otero, A. M., Munoz, A., Bernández, M. I., & Fábregas, J. (2004). *Quorum sensing el lenguaje de las bacterias* (First Edition ed.). Spain: Acribia. (ISBN 9788420010465)

Parhi, D. R., Pothal, J. K., & Singh, M. K. (2009). Navigation of multiple mobile robots using swarm intelligence. In *Proc. world congress nature & biologically inspired computing nabic 2009* (p. 1145-1149).

Parker, M., Kashtan, N., & Alon, U. (2007). Environmental variability and modularity of bacterial metabolic networks. *BMC Evolutionary Biology*, *7*(169), 1-8. (doi:10.1186/1471-2148-7-169)

Passino, K. M. (2002, August). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems*, *22*, 52-67. Retrieved from

Paton, R., Vlachos, C., Wu, Q., & Saunders, J. (2006). Simulated bacterially-inspired problem solving. the behavioural domain. *Natural Computing*, *5*, 43-65.

Petru, L., & Wiedermann, J. (2011). A universal flying amorphous computer. In C. Calude, J. Kari, I. Petre, & G. Rozenberg (Eds.), *Unconventional computation* (Vol. 6714, p. 189-200). Springer Berlin Heidelberg.

Ping, Y., Chao, Y., Li, Z., & Cuiming, L. (2010). Based on game theory and ant colony algorithm's research on group robot cooperative system control. In *2010 international conference on electrical and control engineering (icece)* (p. 532-535).

Polani, D. (2003). Measuring self-organization via observers. In *7th european conference on artificial life* (Vol. 7, p. 667-675). Springer. (ISSN 0302-9743)

Prodan, L., Tempesti, G., Mange, D., & Stauffer, A. (2003). Embryonics: Electronic stem cells. In *8th international conference on artificial life* (p. 101-105). MIT Press.

Prokopenko, M. (2008). *Advances in applied self-organizing systems* (1st ed.). Springer. (ISBN 978-1-84628-981-1)

Ranjbar-Sahraei, B., Shabaninia, F., Nemati, A., & Stan, S. (2012). A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems. *IEEE Transactions on Industrial Electronics*, *59*(8), 3124-3134.

Reif, J., & Wang, H. (1999). Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, *27*(3), 171-194.

Robinson, C. (2008). *Robustness of lepidopteran midgut microbial communities*. Unpublished doctoral dissertation, University of Wisconsin-Madison.

Robinson, C., Schloss, P., Ramos, Y., Raffa, K., & Handelsman, J. (2010). Robustness of the bacterial community in the cabbage white butterfly larval midgut. *Microbial Ecology*, *59*(2), 199-211. (DOI: 10.1007/s00248-009-9595-8)

Russell, S., & Norvig, P. (2002). *Artificial intelligence: A modern approach* (Second Edition ed.). Pearson Prentice Hall. (ISBN 0137903952)

Sahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In E. Sahin & W. Spears (Eds.), *Swarm robotics* (Vol. 3342, p. 10-20). Springer Berlin Heidelberg.

Sahraei, B. R., Nemati, A., Farshchi, M., & Meghdari, A. (2010). Adaptive fuzzy sliding mode control approach for swarm formation control of multi-agent systems. In *ASME conference proceedings* (p. 485-490).

Santini, C., & Tyrrell, A. (2009, Sept.). Investigating the properties of self-organization and synchronization in electronic systems. *IEEE Transactions on NanoBioscience*, *8*(3), 237-251. (ISSN 1536-1241)

Sarkar, S., & Kar, I. (2013). Formation control of multiple groups of robots. In *2013 ieee 52nd annual conference on decision and control (cdc)* (p. 1466-1471).

Sato, N., Kon, K., & Matsuno, F. (2011). Navigation interface for multiple autonomous mobile robots with grouping function. In *2011 ieee international symposium on safety, security, and rescue robotics (ssrr)* (p. 32-37).

Savkin, A., & Teimoori, H. (2010). Decentralized navigation of groups of wheeled mobile

robots with limited communication. *IEEE transactions on robotics*, *26*(6), 1099-1104.

Sayed, A. H., & Sayed, F. A. (2011, Nov.). Diffusion adaptation over networks of particles subject to brownian fluctuations. In *2011 conference record of the forty fifth asilomar conference on signals, systems and computers asilomar* (p. 685-690). (ISBN 978-1-4673-0321-7)

Scheidler, A., Brutschy, A., Diwold, K., Merkle, D., & Middendorf, M. (2011). Ant inspired methods for organic computing. In C. MÃ$\frac{1}{4}$ller-Schloer, H. Schmeck, & T. Ungerer (Eds.), *Organic computing - a paradigm shift for complex systems* (Vol. 1, p. 95-109). Springer Basel.

Schwartz, J. T., & Sharir, M. (1983). On the Piano Movers' Problem: III. Coordinating the motion of several independent bodies. *International Journal of Robotics Research*, *2*(3), 97-140.

Sha, L. (2001). Using simplicity to control complexity. *IEEE Software*, *18*(4), 20-28.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379-656.

Shen, J., & Zhou, H. (2010). The dynamics of quorum sensing mediated by small rnas in vibrio harveyi. In K. Li, X. Li, S. Ma, & G. Irwin (Eds.), *Life system modeling and intelligent computing* (Vol. 97, p. 177-184). Springer Berlin Heidelberg.

Sipper, M., Mange, D., & Stauer, A. (1997). *Biosystems* (Vol. 44) (No. 3). Elsevier.

Smith, S. L., Tumova, J., Belta, C., & Rus, D. (2010). Optimal path planning under temporal logic constraints. In *Proceedings ieee/rsj international conference on intelligent robots and systems* (p. 3288-3293).

Stamatious, G. (2009, July 14). *Motion of a particle in the bunimovich billiard.* On line. Retrieved from `http://upload.wikimedia.org/wikipedia/commons/8/83/BunimovichStadium.png`

Sztipanovits, J., Ying, S., Cohen, I., Corman, D., Davis, J., Khurana, H., et al. (2013). *Foundations for innovation: Strategic r&d opportunities for 21st century cyber-physical systems* (Tech. Rep.). Steering Committee for Foundations in Innovation for Cyber-Physical Systems.

Tabachnikov, S. (2005). *Geometry and billiards.* Providence, Rhode Island: American Mathematical Society.

Tang, W., Wu, Q., & Saunders, J. (2007). Individual-based modeling of bacterial foraging with quorum sensing in a time-varying environment. In E. Marchiori, J. Moore, & J. Rajapakse (Eds.), *Evolutionary computation, machine learning and data mining in bioinformatics* (Vol. 4447, p. 280-290). Springer Berlin Heidelberg.

Tang, W. J., Wu, Q. H., & Saunders, J. R. (2007). A bacterial swarming algorithm for global optimization. In *Proc. ieee congress evolutionary computation cec 2007* (p. 1207-1212).

Tarakanov, A. O., & Dasgupata, D. (2000, Feb.). A formal model of an artificial immune system. *Biosystems*, *55*(3), 151-158. (ISSN 0303-2647)

Taylor, A. F., Tinsley, M. R., Wang, F., Huang, Z., & Showalter, K. (2009). Dynamical quorum sensing and synchronization in large populations of chemical oscillators. *Science*, *323*(5914), 614-617.

Terrazas, G., Krasnogor, N., Gheorghe, M., Bernardini, F., Diggle, S., & Cámara, M. (2005). An environment aware p-system model of quorum sensing. In S. Cooper, B. Löwe, & L. Torenvliet (Eds.), *New computational paradigms* (Vol. 3526, p. 7-27).

Springer Berlin Heidelberg.

Teturo, I. (2010). Group dynamics of robots without sensing devices. In *2010 world automation congress (wac)* (p. 1-6).

Tomlin, C., Pappas, G. J., & Sastry, S. (1998, April). Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, *43*(4), 508-521.

Toth, A., Banky, D., & Grolmusz, V. (2011, Dec.). 3-d brownian motion simulator for high-sensitivity nanobiotechnological applications. *IEEE Transactions on NanoBioscience*, *10*(4), 248-249. (ISSN 1536-1241)

Tovar, B., Cohen, F., & LaValle, S. M. (2009). Sensor beams, obstacles, and possible paths. In G. Chirikjian, H. Choset, M. Morales, & T. Murphey (Eds.), *Algorithmic foundations of robotics, viii.* Berlin: Springer-Verlag.

Turing, A. (1937). On computable numbers, with an application to the entscheidungproblem. *Proceedings of the London Mathematical Society*, *1*, 230-265.

Vaughan, R. (2008, December 1). Massively multi-robot simulation in stage. *Swarm Intelligence*, *2*(2), 189-208.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., & Shochet, O. (1995). Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, *75*(6), 1226-1229.

Wagner, A. (2005). *Robustness and evolvability in living systems.* Princeton University Press.

Wang, R., Zhao, X., & Liu, Z. (2009). Modeling and dynamical analysis of molecular networks. In J. Zhou et al. (Eds.), *Complex sciences* (Vol. 5, p. 2139-2148). Springer Berlin Heidelberg.

Whitesides, G. M., & Boncheva, M. (2002, April 16). Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proceedings of the National Academy of Sciences*, *99*(8), 4769-4774.

Wiedermann, J. (2011a). Amorphous computing: a research agenda for the near future. *Natural Computing*, 1-5.

Wiedermann, J. (2011b). Nanomachine computing by quorum sensing. In J. Kelemen & A. Kelemenova (Eds.), *Computation, cooperation, and life* (Vol. 6610, p. 203-215). Springer Berlin Heidelberg.

Winzer, K., Hardie, K. R., & Williams, P. (2002, April). Bacterial cell-to-cell communication: sorry, can't talk now - gone to lunch! *Current Opinion in Microbiology*, *5*, 216-222.

Wokoma, I., Shum, L. L., Sacks, L., & Marshall, I. (2005). A biologically-inspired clustering algorithm dependent on spatial data in sensor networks. In *Proc. proceeedings of the second european workshop wireless sensor networks* (p. 386-390).

Wu, M., Yan, G., Lin, Z., & Lan, Y. (2009). Synthesis of output feedback control for motion planning based on LTL specifications. In *Proceedings ieee/rsj international conference on intelligent robots and systems* (p. 5071-5075).

Yang, B., Liu, J., & Liu, D. (2010). An autonomy-oriented computing approach to community mining in distributed and dynamic networks. *Autonomous Agents and Multi-Agent Systems*, *20*, 123-157.

Yoshida, K., Fukushima, H., Kon, K., & Matsuno, F. (2014). Control of a group of mobile robots based on formation abstraction and decentralized locational optimization. *IEEE Transactions on Robotics*, *30*(3), 550-565.

Zang, T., He, Z., & Ye, D. (2010). Bacterial foraging optimization algorithm with particle swarm optimization strategy for distribution network reconfiguration. In Y. Tan, Y. Shi, & K. Tan (Eds.), *Advances in swarm intelligence* (Vol. 6145, p. 365-372). Springer Berlin / Heidelberg.

Zeng, J., & Li, T. (2009, Nov.). A novel computer virus detection method from ideas of immunology. In *International conference on multimedia information networking and security mines '09* (p. 412-416). (ISBN 978-0-7695-3843-3)

Zhang, H., Meng, Z., & Lin, Z. (2012). Experimental verification of a multi-robot distributed control algorithm with containment and group dispersion behaviors. In *2012 31st chinese control conference (ccc)* (p. 6159-6164).

Zhang, H., Zhao, Z., & Lin, Z. (2014). Experimental verification of a multi-robot distributed control algorithm with containment and group dispersion behaviors: the case of dynamic leaders. *IEEE/CAA JOURNAL OF AUTOMATICA SINICA*, *1*(1), 54-60.

Zhang, J., Liu, Z., Li, Y., & Chen, L. (2007). Frequency synchronization of a set of cells coupled by quorum sensing. In K. Li, X. Li, G. Irwin, & G. He (Eds.), *Life system modeling and simulation* (Vol. 4689, p. 21-27). Springer Berlin Heidelberg.

Zhou, W., & Nakhleh, L. (2012). Convergent evolution of modularity in metabolic networks through different community structures. *BMC Evolutionary Biology*, *12*(181), 1-14. (DOI: 10.1186/1471-2148-12-181)