Pontifícia Universidade Católica do Rio de Janeiro

Pedro Bittencourt e Silva

Collective Behavior on Multi-Agent Robotic Systems using Virtual Sensors

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica da PUC-Rio.

Orientador: Marco Antonio Meggiolaro

Rio de Janeiro, 26 de abril de 2012

Pedro Bittencourt e Silva

Collective Behavior on Multi-Agent Robotic Systems Using Virtual Sensors

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Marco Antonio Meggiolaro
Orientador
Pontifícia Universidade Católica do Rio de Janeiro

Dan Sali Reznik
University of California - Berkeley

Raul Almeida Nunes
Pontifícia Universidade Católica do Rio de Janeiro

José Eugenio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 26 de abril de 2012

Resumo

Bittencourt, Pedro. **Comportamento Coletivo em Sistemas Robóticos Multi-Agentes usando Sensores Virtuais.** Rio de Janeiro, 2012. Dissertação de Mestrado - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Robótica coletiva de enxame é uma abordagem para o controle de sistemas robóticos multi-agentes baseada em insetos sociais e outros sistemas naturais que apresentam características de auto-organização e emergência, com aplicações disruptivas em robótica e inúmeras possibilidades de expansão em outras áreas. Porém, sendo um campo relativamente novo existem poucas plataformas experimentais para seu estudo, e as existentes são, em sua maioria, especialmente desenvolvidas para tarefas e algoritmos específicos. Uma plataforma de estudos genérica para o estudo de sistemas robóticos coletivos é, por si só, uma tarefa tecnológica não trivial além de ser um recurso valioso para um centro de pesquisas interessado em realizar experimentos no assunto.

Neste trabalho dois importantes algoritmos de controle colaborativo multi-robôs foram estudados: busca do melhor caminho e transporte coletivo. Uma análise completa dos mecanismos biológicos, dos modelos lógicos e do desenvolvimento dos algoritmos é apresentada.

Para realizar os experimentos uma plataforma genérica foi desenvolvida baseada nos robôs móveis “iRobot Create”. Sensores virtuais são implementados em através de um sistema de visão computacional combinado com um simulador em tempo real. O sistema de sensores virtuais permite a incorporação de sensores ideais no sistema experimental, incluindo modelos mais complexos de sensores reais, incluindo a possibilidade da adição de ruído simulador nas leituras. Esta abordagem permite também a utilização de sensores para detecção de objetos virtuais, criados pelo simulador, como paredes virtuais e feromônios virtuais.

Cada robô possui um sistema eletrônico embarcado especialmente desenvolvido baseado em micro controlador ARM. A eletrônica adicionada é responsável por receber as leituras dos sensores virtuais através de um link de radio em um protocolo customizado e calcular, localmente, o comportamento do robô. Os algoritmos são implementados na linguagem de alto nível Lua. Mesmo com as leituras dos sensores virtuais sendo transmitidas de um sistema

centralizado é importante ressaltar que todo o algoritmo de inteligencia é executado localmente por cada agente.

As versões modificadas e adaptadas dos algoritmos estudados na plataforma com sensores virtuais foram analisadas, juntamente com suas limitações, e se mostraram compatíveis com os resultados esperados e acessíveis na literatura que utiliza sistemas experimentais mais específicos e mais dispendiosos. Portanto a plataforma desenvolvida se mostra capaz como ferramenta para experimentos em controle de sistemas robóticos multi-agentes com baixo custo de implementação, além da inclusão, através do mecanismo de sensores virtuais, de sensores ainda em desenvolvimento ou comercialmente indisponíveis.

Palavras Chave

Swarm Robotics; Inteligência Artificial; Robôs Móveis; Visão Computacional; Sistemas Embarcados.

Abstract

Bittencourt, Pedro. **Collective Behavior on Multi-Agent Robotic Systems using Virtual Sensors.** Rio de Janeiro, 2012. Dissertação de Mestrado - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Swarm robotics is an approach to multi-robot control based on social insects and other natural systems, which shows self-organization and emergent characteristics, with disruptive applications on robotics and possibilities in a variety of areas. But, being a relatively new field of research, there are few experimental platforms to its study, and most of them are crafted for very specific tasks and algorithms. A general study platform of swarm robotics, by itself, is a non-trivial technological deed and also a very valuable asset to a research center willing to run experiments on the topic.

In this work, two important algorithms in multi-robot collaborative control strategies are studied: path finding and collective transport. A complete analysis of the biological mechanisms, models and computer abstractions that resulted in the development of those algorithms is shown.

To perform the multi-robot experiments, several "iRobot Create" mobile robots are employed. Virtual sensors and virtual walls are implemented in real time in the experimental system through cameras and especially developed computer vision software. Virtual sensors allow the incorporation of ideal sensors in the experimental system, including complete models of real sensors, with the possibility of adding virtual noise to the measurements. This approach also allows the use of sensors to detect virtually created objects, such as virtual walls or virtual pheromones.

Each physical robot has a customized embedded system, based on the ARM microprocessor, which receives the virtual sensors readings through a radio link in an also customized protocol. The behavior of each autonomous agent is locally calculated using the high-level programming language Lua. Even though the virtual sensor readings are transmitted from an external centralized computer system, all behaviors are locally and independently calculated by each agent.

The adaptations of the studied algorithms to the platform with virtual sensors are analyzed, along with its limitations. It is shown that the experimental results using virtual sensors are coherent with results from the literature using very specialized and expensive robot/sensor setups. Therefore, the developed platform is able to experimentally study new control strategies and swarm algorithms with a low setup cost, including the possibility of virtually incorporating sensors that are still under development or not yet commercially available.

Keywords

Swarm Robotics; Artificial Intelligence; Mobile Robots; Computer Vision; Embedded Systems

Summary

List of Figures

1 Introduction

Robots are defined as autonomous agents capable of interacting with the physical world. For instance a mechanical manipulator on a production line or an extraplanetarian exploratory rover can be considered Robots, if they have certain level of autonomy. Robots have been heavily used in industry for decades, particularly in the automobile manufacturing plants [1] where robotic manipulators are used for repetitive tasks such as welding and painting. This brings benefits like costs reduction, lead time reductions, quality improvements and many others.

A great trend in modern robotics are the mobile robots, autonomous agents with movement. Their popularity rose mostly because of the cost reduction in the robot hardware production and advances in microprocessors technology, allowing more powerful computer systems to be embedded. Novel applications of mobile robots are appearing in every sector of the economy, from primary industry ones, like the CAT Minestar [2] system of mine automation (including autonomous heavy-duty trucks), commercial and distribution applications like the kiva warehouse [3] system that relies on a group of robots that handle and organize shelves, highly improving operation speed on distribution hubs. And finally and most dramatic change in robot market: the rise of personal robotics, robots that are able to do basic household tasks such as the iRobot Roomba autonomous vacuum cleaner.

An evidence of this growth has been stated by the IFR (International Federation of Robotics) on the annual report of robot market [4]:

"In 2010, about 2.2 million service robots for personal and domestic use were sold, 35% more than in 2009. The value of sales increased by 39% to US$538 Million.".

This rapid growth on mobile robots market created a demand for more developments on the area, mainly in the artificial intelligence and control theory fields, the ones responsible for the development of the core component of these robots, their autonomy.

In this work the problem of multi-robot systems will be studied. Multi-robot coordination and control is proved to be an affordable path to increase performance of robotic systems, in special the multi-robot collaborative behavior. In this context swarm robotics appeared as a biologically inspired solution. As stated by Garnier, Gautrais et Theraulaz in 2007 [5]

“The roots of swarm intelligence are deeply embedded in the biological study of self-organized behaviors in social insects. From the routing of traffic in telecommunication networks to the design of control algorithms for groups of autonomous robots, the collective behaviors of these animals have inspired many of the fundamental works in this emerging research field”.

Şahin [6] proposed the motivations of swarm robotics

“[...] there exists no centralized coordination mechanisms behind the synchronized operation of social insects, yet their system-level functioning is robust, flexible and scalable. Such properties are acknowledged to be desirable for also multi-robot systems, and can be stated as motivations for the swarm robotics approach.”

The author described the three properties as:

- **Robustness**
  Robustness is the capacity of the system to resist endogenous failures and exogenous disturbances and yet accomplishes its objectives, with loss of performance or not. Endogenous failures can be seen, in a multi-robot system, as the complete or partial failure of some agents and exogenous disturbances are the unexpected changes in the environment. An example of robustness in natural system are the domestic ant raids that are very hard to eliminate and resilient.

- **Flexibility**

  Flexibility is described as the capability to solve different tasks in modularized ways. The author exemplify this ability with the task allocations in an ant colony, where ants perform a variety of tasks and can be reallocated dynamically from one task to another , without central control, as the demands of the colony. In robots that can be seen as the ability to perform different tasks just by reorganizing their behaviors.

- **Scalability**

  Scalability is the capacity of work with different group sizes and to be able to improve performance with addition of more agents, or naturally adapts to keep the task when the number of agents diminishes.

Social insects, as referred above, are insects that have colonies self-organized in highly collaborative structure, also known as eusocial societies or super organisms. A common example and source of inspiration are ant colonies, entities that perform a myriad of tasks impossible to single ants to achieve, like carrying heavy weights or transpose gaps wider than their own bodies, cited by the author above and illustrated in Figure 1.

Figure 1: *Eciton burchelli* ants form living bridges with their bodies. (With the permission of the author) © Alexander Wild.

Swarm robotics, a term created by Gerardo Beni and Jing Wang in 1989 to cellular robotic systems (robotics over cellular automatons), is based on three hypotheses:

- **Local knowledge**

Each agent has a limited perception of its environment through local sensors. This hypothesis come from the biological notion that senses have a limited range and capabilities, for example the antennae of ants that can detect chemicals at very short lengths.

- **Simplicity of agents**

Agents are simple in relation to the desired task to be executed, this means that a single robot of the swarm is not capable of execute the task, or can realize it very poorly relative to the desired performance.

- **No centralized control**

There is no central control for the system, but is possible to use a central observation system that would not inflict the local knowledge hypothesis.

Swarm intelligence is a promising field in robotics, especially if combined with other trends like Nano robotics, self-assembled robots and others. Other evidence of the importance of the field is the amount of investments that national agencies and other research foundations are lending to swarm related projects. A great example of this is the couple of sequential projects funded by the Future and Emerging Technologies program of the European Commission:

- **Swarm-bots project (from 2001 to 2005)**

A 42 month, 2.17 million euro project coordinated by Professor Marco Dorigo and describe its goals as stated:

"The objective of the SWARM-BOTS project is to study a novel approach to the design, hardware implementation, test and use of self-assembling, self-organizing, metamorphic robotic systems called *swarm-bots*. This novel approach finds its theoretical roots in recent studies in swarm intelligence, that is, in studies of the self-organizing and self-assembling capabilities shown by social insects and other animal societies."

- **Swarmanoid (from 2006 to 2011)**

Also a 42 month, 2.17 million euro project. Considered a second part of the swarm-bot project, but with more advanced goals:

"The Swarmanoid project proposes a highly innovative way to build robots that can successfully and adaptively live in human-made environments. The main scientific objective of the proposed research is the design, implementation and control of a novel distributed robotic system comprising heterogeneous, dynamically connected small autonomous robots so as to form what we call a

Swarmanoid. The Swarmanoid that we intend to build will be comprised of numerous (60) autonomous robots of three types: eye-bots, hand-bots and foot-bots."

1.1.1.General Applications for Swarm Robotics

Swarm robotics are applicable in a myriad of fields, researchers consider the solution more valuable for:

- **Inspection**

Swarm robotics can greatly improve inspections, in special for large or complex sites. Since the scan can be parallelized attributing different regions for subgroups of agents achieving scalability. Other advantage is the robustness, since the malfunction of one agent does not compromise the whole mission.

- **Rescue and searching missions**

By its distributed nature swarm robotics can achieve faster results in locating elements in a map and retrieving it, robustly. The two algorithms implemented in this work can be used concomitantly to create a team of robots that can locate and rescue victims in danger zones.

- **Site construction**

Some studies analyze bees, wasps and termites nest construction behaviors to deploy teams of robots capable of constructing structures autonomously. This ability has been considered a viable way to construct extraplanetarian bases on distant moons and planets without human intervention, or even remote areas of the planet.

- **Asset protection**

By imitating some ant's behavior, robots can detect areas and valuable assets and, through a coordinated strategy, surround the area or asset to protection or inhibition. This kind of behavior can be used to create swarms of oil-spill cleaner robots that are capable of detecting the boundaries of the spill, surround it and eventually absorb it.

- **Flight and ground traffic coordination**

Another important application can be the coordination of ground and air traffic for groups of autonomous vehicles. *Nissan´*s realized experiments in the area with the it´s robot platform *Eporo* [7], in this experiments the imitation of fish in a school is used to create collision avoidance in common traffic situations.

The same principle can be applied for the coordination of UAVs (Unmanned Aerial Vehicles), as an experiment from *EPFL*, in Switzerland showed. [8].

Beside the cited possible applications, swarm robotics can be a low cost, robust, and scalable solution to many other common industrial, scientific and even medical problems.

1.2.Objectives

In this work is presented the project and development of a complete platform for multi-agent robotic systems study. The platform is based on the *iRobot's* corporation *Create* robot, as seen in Figure 2.

Figure 2: *iRobot Create* hardware with the *iRobot Control Module*.

Create robots are very limited in terms of sensing and communication, so a computer vision system was used to detect robot's position and orientation. With this information a simulator software computes each robot simulated sensor reading; those readings are sent to each agent by a radio frequency link and finally each agent take the decision of what action to take based on the algorithm running on their core processor. In this case an external electronics was necessary to run the language that permitted the dynamical reprogramming of the agent, eLua [9].

Virtual sensors, as will be further called theses sensors created at the simulator, act exactly as a real sensor mounted on each robot, extending to the possibility of creating sensors to virtual artifacts, such as virtual pheromones and obstacles. This strategy also allows the platform user to test a variety of different sensors just programming the sensor response, or even test models of real sensor (models containing noise, miscalibration etc).

In a higher level, the system described above is a closed loop as illustrated on the Figure 3

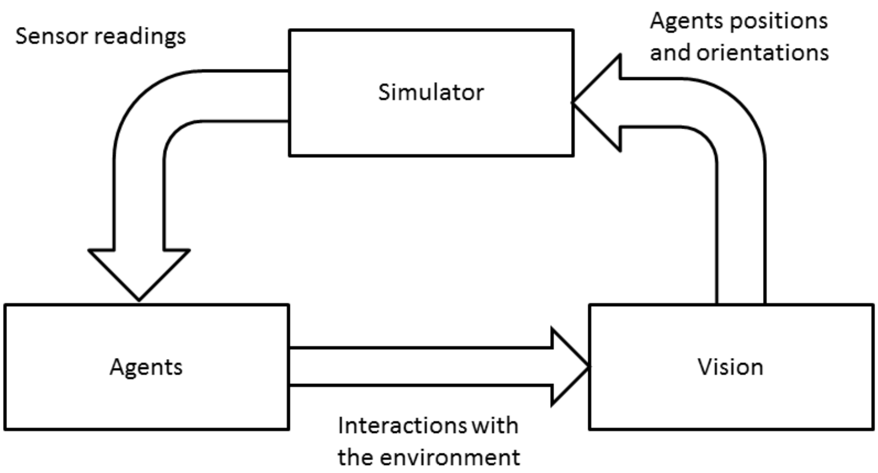

Figure 3: Diagram with system overview.

Within this context two algorithms were implemented: the path finding algorithm based on the pheromone trail biological model and loosely in the graph searching algorithm described in chapter 2, and the collective transport behavioral algorithm described by roboticists with inspiration from cooperative prey retrieval habits from ants, also described in chapter 2.

The two main goals of this work are: first develop a complete, physical, study platform for swarm robotics experiments, a second, to implement basic algorithms within the platform.

1.3.Dissertation Structure

This report is organized in six chapters, as described below:

Chapter one introduces the motivations of the collective robotics and swarm intelligence, and present the objectives and basic practical aspects of developed work.

Chapter Two gives and overview of the theoretical foundations of swarm intelligence, the biological models and the analogies made from nature to computer algorithms and control strategies for collective robotics.

Chapter three describes the technical aspects of the developed platform, project definitions, part descriptions and implementation details.

Chapter four discuss two algorithms of swarm control applied to the developed platform: the collective transport with suppressive hierarchical behavior based control and the self-organized path forming.

Chapter five presents the results of experiments on the platform, a quantitative and qualitative analysis of the results.

Chapter six concludes the work critically reviewing the results and objectives and possible future developments.

2 Theoretical Framework

As stated in the first chapter, the term “Swarm Intelligence” was coined in the end of the 80’s in the context of cellular robotics, but the theory behind it first appeared, in an organized and scientific way, in the early 60’s with the study of honeybees and termites by biologists. A major contribution at this time was the discovery of Stigmergy by Pierre-Paul Grassé, a termite specialist, in 1959 [10]. Stigmergy is the indirect form that social insects communicate, it’s a communication based on the changes on the environment caused by agents that stimulates other agents. It is one of the most important concepts of Swarm Intelligence. A sample of applied Stigmergy in algorithms is described later in this section on the Ant System routing algorithm description where the pheromone trails is the only form of communication.

The systematic analysis of collaborative biological behaviors started to gain strength in the end of 70’s and beginning of the 80’s because of technological advances, such as portable video cameras and computers. The focus at the time was given in the study of fish and krill schools formation and behavior.

Soon this studies start to interest computer scientists and mathematicians who developed theoretical abstractions and models, what allowed computer simulations of those behaviors to be created. The first documented simulator was called *Boids*, developed by computer scientist Craig Reynolds in 1987 [11] and it was a bird flock simulator based on three simple rules:

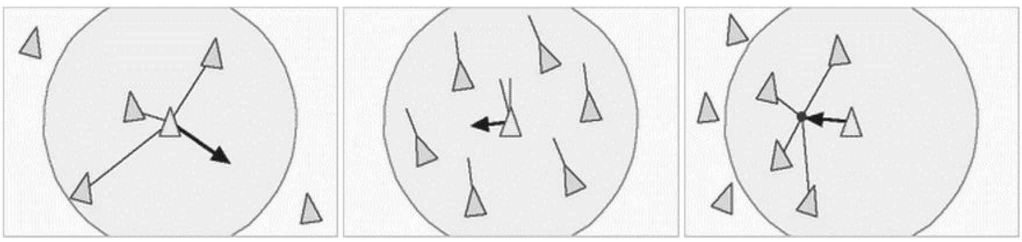

Figure 4: *Boids* original steering rules: separation, alignment and cohesion.

- **Separation**

Steer to avoid crowding local flock mates;

- **Alignment**

Steer towards the average heading of local flock mates;

- **Cohesion**

Steer to move toward the average position (center of mass) of local flock mates.

Those rules were activated based only on the local neighborhood, in a tridimensional space, of a given agent. The result is an emergent phenomenon of synchronized global group movements, called flocking. A very common phenomenon observed in bird flocks and large fish schools.

2.1. Routing and Graph Traversing

With time, deeper analysis of social behaviors led to more complete models that could be applied on useful algorithms, a major contributor to the fundamental advances in the area is the biologist Jean-Louis Deneubourg. One of the most useful models applied to the creation of algorithms was described by Deneubourg in 1989 [12], it was based on the Argentine ant food foraging behavior.

Using a simple experimental setup, the researcher shown that path selection to a food source in the Argentine ant *Linepithema humile* is based on self-organization. In this experiment, a food source is separated from the nest by a bridge with two equally long branches A and B (Figure 5). Initially there is no pheromone on the two branches. One ant has the probability P_a of choosing path A and probability $P_b = (1 - P_a)$ of choosing path B, when there are no pheromones on the paths random fluctuations will cause a few more ants to select one branch, for instance A over B. Because ants deposit pheromones while walking, the greater number of ants on branch A determines a greater amount of pheromones on A, which in turn stimulates more ants to choose A, creating a positive feedback loop.

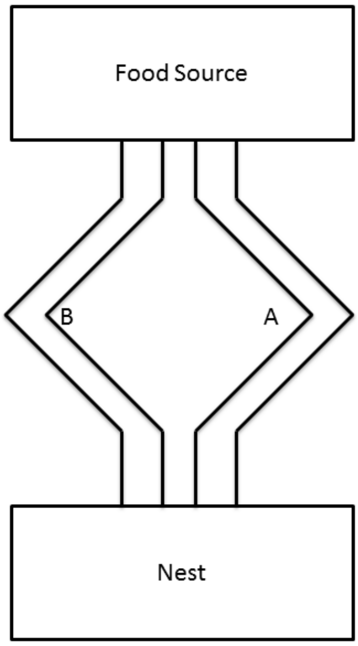

Figure 5: Deneubourg experiments with equally long branches from the nest to food source.

The model developed by Deneubourg et al. [12] explained the phenomenon mathematically. In the model, the probability of choosing a branch at a certain time depends on the total number of ants that used the branch until that time. Let A_i and B_i be the numbers of ants that have used the branches A and B after i ants have used the bridge. The probability P_a that the $(i+1)th$ ant choses branch A is

$$P_a = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} \qquad (2.1)$$

Where:

P_a Is the probability of the $(i+1)th$ ant to choose branch A;

k Is the degree of attraction of an unmarked branch;

n Is the parameter to determine the degree of nonlinearity of the choice function.

The values found to be the best fit to the real experiment where $n \approx 2$ and $k \approx 20$. A second experiment showed how the pheromone trail mechanism was used on choosing the shortest path [13], in this second experiment branches with different lengths were used, as shown in Figure 6.

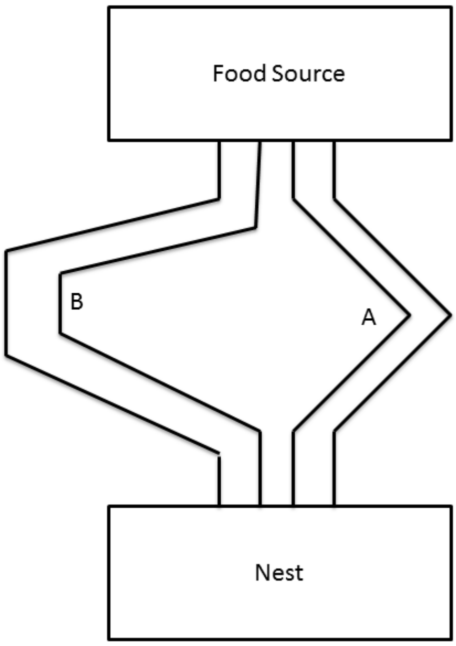

Figure 6: The second experiment. Branch B is longer than branch A.

Adding the pheromone evaporation to the model, the researchers showed the ant food foraging behavior have an emergent property of choosing the shortest path between two points. Because of the decaying amount caused by the evaporation, the longer trail naturally have less pheromone, due longer travel time, causing the shortest path to become more prone to be selected.

Dorigo et al. [14] developed a computational abstraction of this model and applied to the problem of graph path-finding, most notably benchmarked on the travelling salesman problem (TSP). In the TSP the goal is to find a closed tour of minimal length connecting n given nodes. Each node must be visited once and only once. Let d_{ij} be the distance, or weight, between nodes i and j in a graph given by(N, E). The Ant System (AS), how it was originally called, build solutions for the TSP by moving on the graph one node to another until a group of virtual ants complete a tour. On an AS iteration each ant $k, k = 1, \cdots, m$ executes $n = |N|$ steps with a probabilistic transition rule. Iterations are indexed by $t, 1 \leq t \leq t_{max}$ where t_{max} is a parameter that limits the number of iterations. For each ant, the transition rule from node i to node j at iteration t depends on three items:

- **Node list**

For each ant a memory is maintained. Each tour keeps a list, which is cleared at the end of a tour. The memory defines, for each ant k, the set J_{ik} of the nodes that the ant has to visit when it is in node i. This mechanism is used to avoid the ant k of visiting a node more than once.

- **Nearness**

The inverse of the weight $\eta_{ij} = {}^{1}/_{d_{ij}}$, which is a strictly local attribute that represents the heuristic desirability of choosing node j when in node i. This is the heuristics of a "greedy algorithm", which alone gives very low quality solutions.

- **Pheromones**

The amount of virtual pheromone trail $\tau_{ij}(t)$ on the edge that connects the node i to node j. Pheromone trails are updated online and is intended to represent the learned desirability of an edge. Opposed to the distance, the pheromone trail gives global information about the problem.

The probability for an ant k to go from node i to node j in the $t - th$ tour is called R*andom Proportional Transition Rule*:

$$p_{ij}^k(t) = \begin{cases} 0 & if\ j \notin J_i^k \\ \dfrac{\left[\tau_{ij}(t)\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum_{l \in J_i^k}[\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & if\ j \in J_i^k \end{cases} \qquad (2.2)$$

Where:

$\tau_{ij}(t)$ is the intensity of the pheromone trail at edge E_{ij} at iteration t;

η_{ij} is the inverse of the weight of the edge E_{ij};

α and β are adjustable parameters that control the relative weight of trail intensity.

At the end of each tour, each ant k lays a quantity of pheromone $\Delta\tau_{ij}^k(t)$ on each edge E_{ij} that it has passed by. The value of $\Delta\tau_{ij}^k(t)$ depends on how well the ant has performed. The value is given by

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L_k(t)} & if\ (i,j) \in T_k(t) \\ 0 & if\ (i,j) \notin T_k(t) \end{cases} \qquad (2.3)$$

Where:

$T_k(t)$ is the tour done by the ant k at iteration t;

L_k is its length;

Q is a parameter to correct the order of magnitude based on the optimal tour length.

Based on the original phenomenon that presented pheromone evaporation, the algorithm implements as a decay rule for the value of $\tau_{ij}(t)$ the following expression

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (2.4)$$

Where:

ρ is the coefficient of decay
$\Delta\tau_{ij}(t)$ is the sum of all pheromone contributions for the edge E_{ij} by all ants.

The initial pheromone amount τ_0 at each edge is assumed to be constant at t_0 and homogeneous, that means all edges have the same amount of pheromone at the beginning. With this algorithm, Dorigo et al [15] showed that pheromone based multi-agent algorithm like the one described above can solve the TSP, and with some changes [16] can be extrapolated to solve any graph traversing problem. The AntSystem algorithm show a practical example of how Stigmergy can be applied on algorithm design, and further on, real engineering problems. This solution has been used on dynamical network routing with best performance than other classical solutions, as stated by Schoonderwoerd et al. [17]. Other applications with success were on vehicle routing problem, Bullnheimer et al. [18] and some similar technique was previously applied to the job scheduling problem by Graham et al. [19].

This family of algorithms can be modified to be implemented on robotic agents performing a search in a graph-mapped environment without much theoretical extrapolation, but several practical adaptations is needed as shown in chapter four.

2.2.Collective Transport

Some species of ants can carry heavy preys, much heavier than the capacity of a single ant, by aggregating around the burden and collectively pushing and pulling until the group can retrieve the object to the nest. This is one of the more emblematic images of social insect’s efficiency and is often used to illustrate teamwork.

This is a behavior observed on several species of ants [20]. Collective transport is considered a remarkable deed of insects, but no formal description of the biological phenomenon has been developed. Surprisingly, roboticists achieved better results to model the phenomenon than biologists. The most complete model has been described by Kube and Zhang on a series of works about collective transport on robots based on ants [21] [22] [23]. Not intending to make it biologically plausible, but using the inspiration given by the ants aiming at autonomous robots applications.

Even without the development of accurate biological models, some biologists made extensive researches about the topic. One of the most insightful analyses was made by Moffett [24] when studying collective transport on the Asian ants *Pheidologeton diversus.* Results of this study show the efficiency of collective transport in terms of speed and energy.

The Kube and Zhang approach for robots is based on the reactive paradigm introduced by Valentino Braitenberg in his classical work *Vehicles* [25] and in the subsumption architecture introduce by Brooks [26].

It is a three sensor – five behaviors based implementation, as seen in Figure 7 bellow.

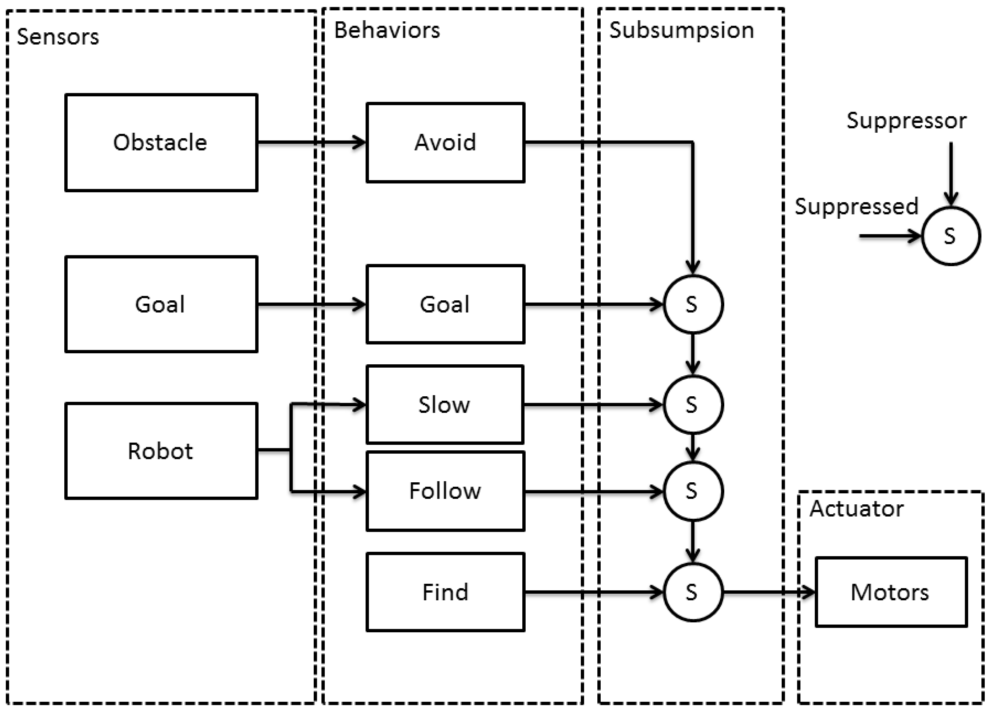

Figure 7: Kube and Zhang architecture to box-pushing behavior, with legend to explain the subsumption operator

Each sensor creates a stimulus that activates one or more behaviors, behaviors are predefined routines that can encapsulate arbitrary levels of complexity, and creates signals to the actuators. Each behavior has a priority, which means, one behavior can subsume a less important one. This behavioral modularity gives the robot software a remarkable flexibility, allowing changes on one behavior without any interference on another and limitless, theoretical, expansion of robot´s capabilities.

With this implementation, the researchers built five collaborative box-pushing robots and gave several insights to the biological behavior of coordinated prey retrieval on insects. The robots performed the first phase of the collaborative box-pushing, the aggregation phase, with success as stated by Kube and Bonabeau at [23]. This first step, where ants need to find the prey and aggregate to start the transport, shows the collaboration through local behavior only but a global resulting effect.

Their experiment relied on a group of five robots, and its physical implementation is quite similar to the presented in the following sections of this work. On Figure 8 a sequence of photos shows the experiment in progress.

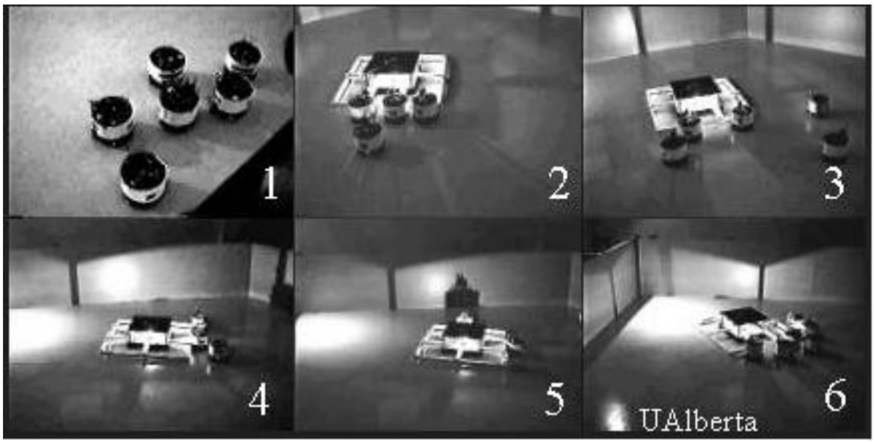

Figure 8: The Kube and Zhang´s experiments on collective transport, aggregation and pushing.

3 Experimental Platform

To execute experiments in swarm robotics a robotic system was necessary. Initial prototype was developed to work with the *Merlin Miabot* robot, but further research led to the more reliable and cost effective *iRobot Create*. *Create* is a robot based on a popular robotic vacuum cleaner (Figure 2), the *Roomba*. Although *Create* robots have just basic sensory capacities, no integrated communication hardware and lack the processing power and programmability necessary to algorithm development. Their capability of expansion compensate those limitations, those drawbacks shaped the requirements for the full platform to be functional.

The final conceptual and practical result is a complete experimental system, fully customizable to new applications and experiments in collective robotics (even to single mobile robot studies). Each main part is illustrated in Figure 9, and detailed in the following sections. Finally there is a section addressing the integration of parts, protocols and interfaces.

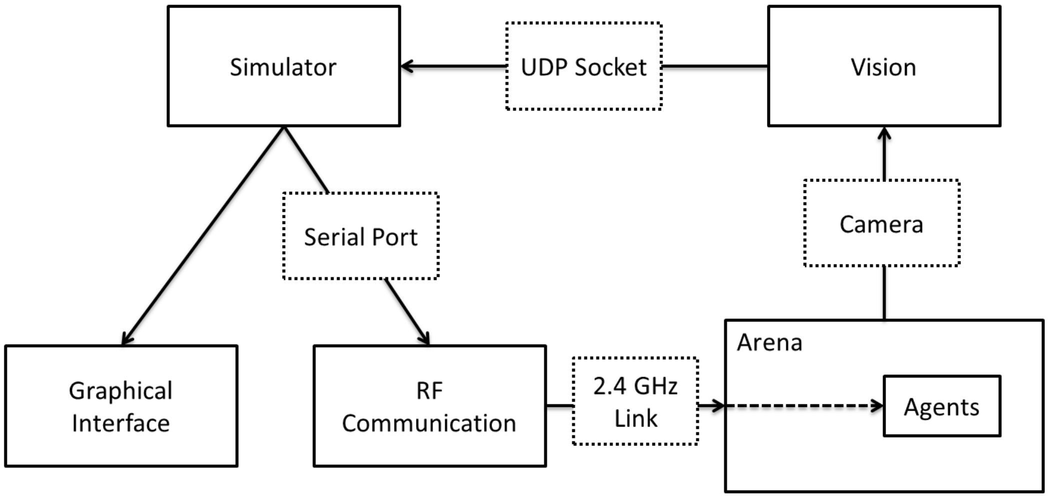

Figure 9: Diagram of the platform components, links and interfaces

3.1.Arena

The arena is the area within the sight of the camera, or, more precisely, the area that is used by the computer vision system to compute agents' position and orientation.

For clarity some elements on the arena have special definitions, listed below:

- **Arena**

The area covered by the camera vision and processed by the computer vision system, except in case of lack of information experiments.

- **Object**

Any artifact identified by the computer vision system, in general have a proper ID and a unique marker.

- **Agent**

Agents are special objects that are capable of movement and have sensors to be stimulated by the arena environment.

- **Virtual**

Any artifact without physical representation that stimulates agents, most used for virtual sensors and virtual pheromones.

The physical setup of the Arena is schematized in Figure 10 and the physical setup on Figure 11.

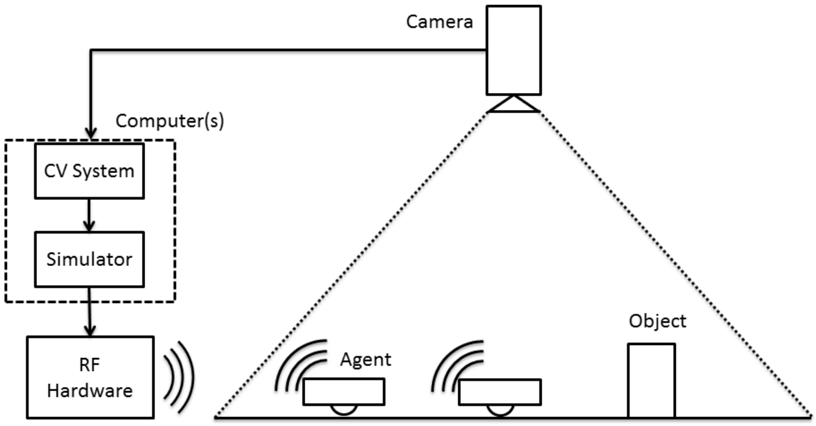

Figure 10: Schematics of the experimental setup arena.

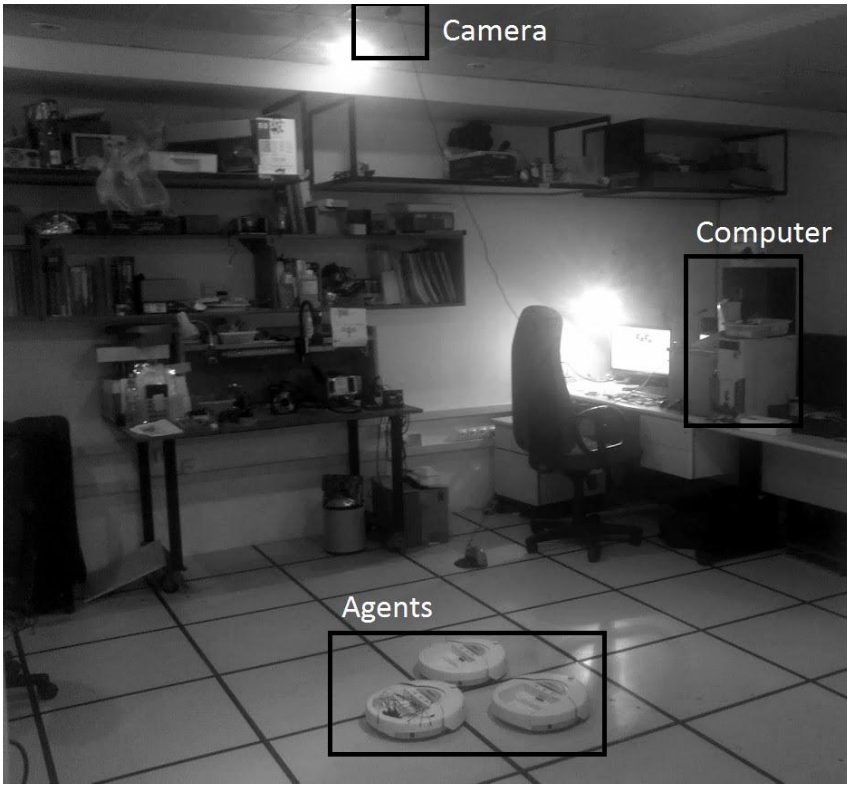

Figure 11: Real experimental setup with main components highlighted

3.2.Simulator

The simulator is a software written in the Lua programming language, it's the module responsible for storing the arena state and calculating the virtual sensors readings for all registered agents. It receives data from the computer vision system by the UDP socket in the TUIO protocol, a special protocol for tangible interfaces similar to the MIDI protocol, where each data packet contains the agent ID, absolute position and absolute orientation.

The simulator converts the coordinates to the arena coordinate system and calculates the sensors reading according to the sensor script file, a mechanism of sensors customization that describes each sensor of each agent. Finally it sends to the RF communication hardware each agent sensor packets and what's going to be rendered by the graphical interface.

It´s architecture is a finite state machine with four main states:

Read UDP Socket

At this state, the simulator reads the datagram broadcasted from the computer vision system that returns the IDs of the agents and objects (each marker has a unique ID), their position and orientation. This is a blocking operation and the simulator passes to the next state as soon as all expected data packets are read or if a timeout occurs. In case of timeout the program follow with the last valid packet and send a warning message to the user.

Calculate virtual sensors readings

Once all positions and orientations are converted to the arena coordinate system, the simulator calculates the virtual sensor readings according to the sensors script, which possesses the calculation method of each sensor, including detection range.

This calculus can vary according to the type of sensor, for example a pheromone sensor reads a virtual amount of pheromone that exists only at the simulator's arena stored internal state. This kind of sensor doesn't need any external information, but an object sensor for instance needs to know the position and orientation of an object marked and detected by the computer vision system.

Refresh the Graphical interface

At this point that all positions, orientations and sensor readings are known, the simulator software refreshes the graphical interface with all available data. In this work the user have no interaction with the simulator, but this can be extended in the future.

Send Virtual Sensor data to RF hardware

The final state sends data to the RF hardware via a serial COM port, the protocol is the same as the RF link: four bytes payload with the first byte as the agent ID, the second is the sensor ID and the last two are the sensor readings.

There is a last phase that occurs at the RF hardware board that verifies the validity of received serial packets before sending it through the 2.4GHz link.

3.2.1.Virtual Sensors

Introduced as a sub product of the developed platform, virtual sensors are a mechanism to simulate any kind of sensor that the experimental agent may have. This resource is extremely useful for three main applications: to test new theoretical models of real sensors, to use ideal sensor that isolates any possible issue associated with physical ones, and to create sensors infeasible in the physical sense, an example used in this work is a sensor to a virtual pheromone – a simulated chemical trail that the agent lays over the arena floor.

- **Test models of real sensors**

More complicated experiments demand a complete knowledge about sensors capabilities and possible flaws, with the virtual sensor mechanism is possible do study theoretical models of real sensors prior to their real utilization and investigate any idiosyncrasy, all that already on the real robotic hardware. An example is to model different levels signal-to-noise ratios to define more robust specifications to the final sensor hardware.

- **Ideal Sensors**

Other advantage of this mechanism is to decouple the experiment from the sensor hardware, because is possible to program an ideal virtual sensor that will be limited only by the computer vision capabilities and the simulator software (both can be customized according to the needs). The other advantage of the utilization of ideal sensors over real ones is the cost, since even basic sensor hardware can be expensive depending of the application.

- **Sensors to virtual stimuli**

Finally, the main reason why this mechanism has been implemented is to measure unpractical or even physically unfeasible evidences. The main example is the pheromone trail, originally a chemical product that insects lay on their paths to communicate, what in a robotic implementation suffers from a series of practical issues such as inaccuracy of sensors, the need to keep a reservoir of a chemical solution and many others. Virtual stimuli can be anything the user can implement as an arena state, any variable visible to the simulator software or even a result from a composition from various sources of data.

3.2.2.Interface with the RF Communication Module

For the simulator to send a packet by the RF link to the agents an electronic board converts serial output from the computer to the nRF24L01 communication format. The hardware is an *Atmel AVR* microcontroller board (Figure 12) and it creates a serial communication port for the computer and converts received data to the SPI interface framing correctly to the radio protocol.

The radio module expects to receive four bytes packets that it transmits after as validity checking, as described in the simulator main description.

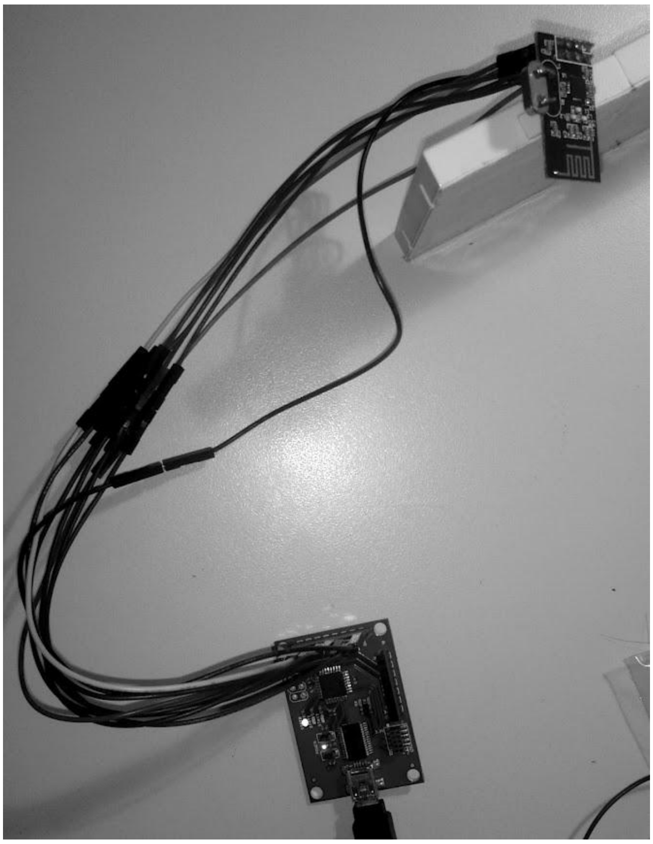

Figure 12: RF communication hardware on the computer side, the converter board is connected to the USB on one side and to the nRF24L01 on the other

3.3.Processing power and programmability

The native controller of the *iRobot Create* is quite simple and inflexible, related to the goals of this work, so a more powerful and programmable hardware was necessary to manage the several tasks that must be autonomously performed by the agent. A second requirement was the interoperability between code and hardware to guarantee that the same firmware developed to a controller will continue to run on a newer processor in the future, allowing the continuous independent (software from hardware and vice-versa) evolution of the platform.

To fulfill those two requirements the chosen hardware is the STM32 controller, in special, the low-cost development board of the processor designed by the *Futurlec* Company, Figure 13.

To the firmware the most adequate solution found was the dynamic language eLua [9], an embedded version of the Lua programming language [27] mainly by its dynamical programming capability (reprogram agents on runtime, without software recompilation) and an advanced hardware abstraction layer, allowing the same code to be executed on different processors and architectures.

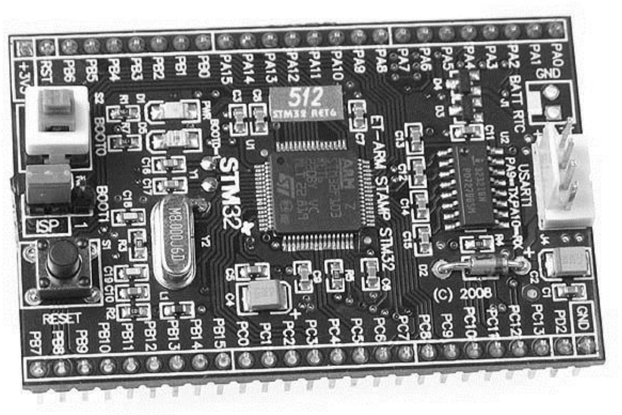

Figure 13: STM32 development board by *Futurlec*, the core processor electronic board of agents.

3.3.1.Agents electronic setup

An electronic setup was necessary to integrate the STM32 processor board and a *Nordic* nRF24L01 radio link to the *Create´s* docking bay, as shown in Figure 14

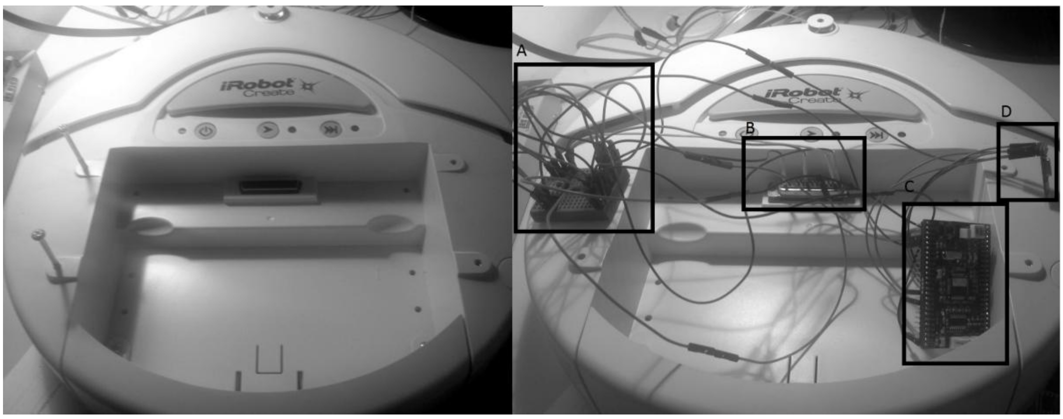

Figure 14: Create before the electronic setup (left) and after (right) with a) Voltage regulator and level converter board b) Custom DB-25 connector to docking bay c) STM32 Board d) nRF24L01 module

Each agent is equipped with a processor board (item c in Figure 14) and a communication module (item d). To connect to the Creates proprietary *Open Interface* connector a custom connector (item b) was built from a DB-25 connector. Finally, due the incompatibility between the robot 5 Volts output source and the 3.3 Volts STM32 processor, a voltage regulator and a level converter scheme was developed and assembled in a prototype board (item a).

3.3.2.Firmware

Each agent computer system runs a customized Lua code that has three main responsibilities:

Run the algorithm

Executing the code of the algorithm implementation, this code is dynamically loaded by the firmware and executed. So it is independent of the main firmware architecture, easing the use and fast prototyping for new experiments.

Interface with RF communication

Virtual sensor readings are transmitted to the agents by the RF link, consequently the main program must execute routines to interface with the SPI of the RF module and decode the transmissions into the virtual readings that will be available to the algorithm code.

Interface with the agent

iRobot Create has an special command language called *Open Interface* that is physically accessed by an UART port on the docking bay connector. To facilitate the algorithm programming, special functions that encapsulate most of the Open Interface commands have been implemented, giving to the algorithm programmer an abstraction layer to control the agent.

The basic architecture of the firmware is also a state machine with three main states, corresponding to the phases described above. First, the algorithm reads the sensor data queued by the RF communication interface phase and then generates events that will be sent to the agent at the Open interface phase.

3.4.Communication

The communication between the simulator and the agents is a radio frequency link of 2.4GHz with the nRF24L01 transceiver module, as shown in Figure 15.

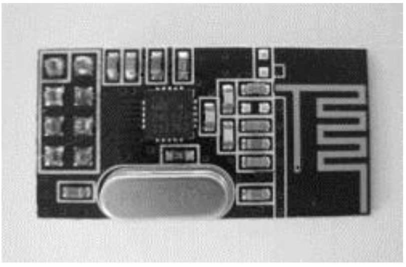

Figure 15: *Nordic* nRF24L01 transceiver, the RF link hardware component between agents and the simulator.

It operates on 3.3 Volts and has a standard SPI serial interface. In this implementation the transceiver only operates in one mode (transmitter for the simulator side and receiver for agent's side), even if it can perform both roles by switching its configuration.

3.4.1.Protocol

Radio link communications use a customized protocol based on a four byte packet. Because of the nature of the data flow – from the simulator to all agents, from one to many – there is no need of anti-collision schemes or complex acknowledgment mechanisms. There is a simple package checksum for data integrity check and lost package automatic resend mechanism.

The packets are arranged in the following order, where each cell is a byte:

Agent Address	Sensor ID	Reading 1	Reading 2

Where readings can be used as single value, where reading 1 is the higher byte and reading 2 is the lower or as two separated values depending on the sensor.

Since the agents implementation can´t rely on hardware interruptions for detecting incoming sensor readings two special packs are sent to each agent: The beginning of sensor array readings (BSR) and end of sensor array readings (ESR). Those packets have an special sensor ID of 255 (or 0xFF in hexadecimal basis)

For example an agent with ID equals to 1 with three sensors with IDs 1, 2 and 3 will receive a chain of packets of the form:

1	255 (special code)	0	0 (BSR)

1	1	Reading 1	Reading 2

1	2	Reading 1	Reading 2

1	3	Reading 1	Reading 2

1	255 (special code)	0	255 (ESR)

Other special case is about more complex sensors that need more than two bytes to represent its readings. In this case packets can be chained repeating the sensor ID and turning one of the readings byte into an address byte for the order of the sensor reading.

3.5.Computer Vision

The computer vision system main goal is to detect the objects on the arena, returning to the simulator their unique ID, position and orientation.

Its main software library is the reacTIVision framework, developed at the University Pompeu Fabra in Barcelona, Spain by Becina et al [28]. Originally developed for robust camera based multi-touch interfaces, the system can be adapted to track marker symbols over the arena. Attaching the markers over the robots, exemplified at Figure 16, the setup becomes equivalent of a multi touch surface (the original environment for the framework).

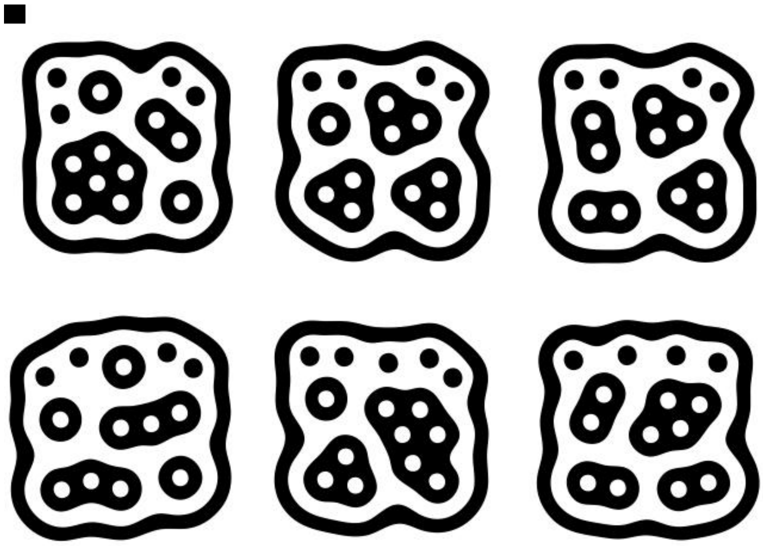

Figure 16: Some samples of the identification and tracking optimized markers created by genetic algorithms

The system is capable of identifying each marker by its assigned ID and detects its position and orientation. Markers were created using a genetic algorithm to optimize identification and tracking [29].

The reacTIVision framework implements a tile-based variation of Bernsen's adaptive method [30] as a thresholding (or binarisation) phase, as illustrated on Figure 17 and the variations of the gradient filter parameter, that can be adjusted to better performance in different backgrounds and image qualities, is illustrated on Figure 18.

Figure 17: Raw greyscale image and the binary image by the Bernsen's binarisation method.

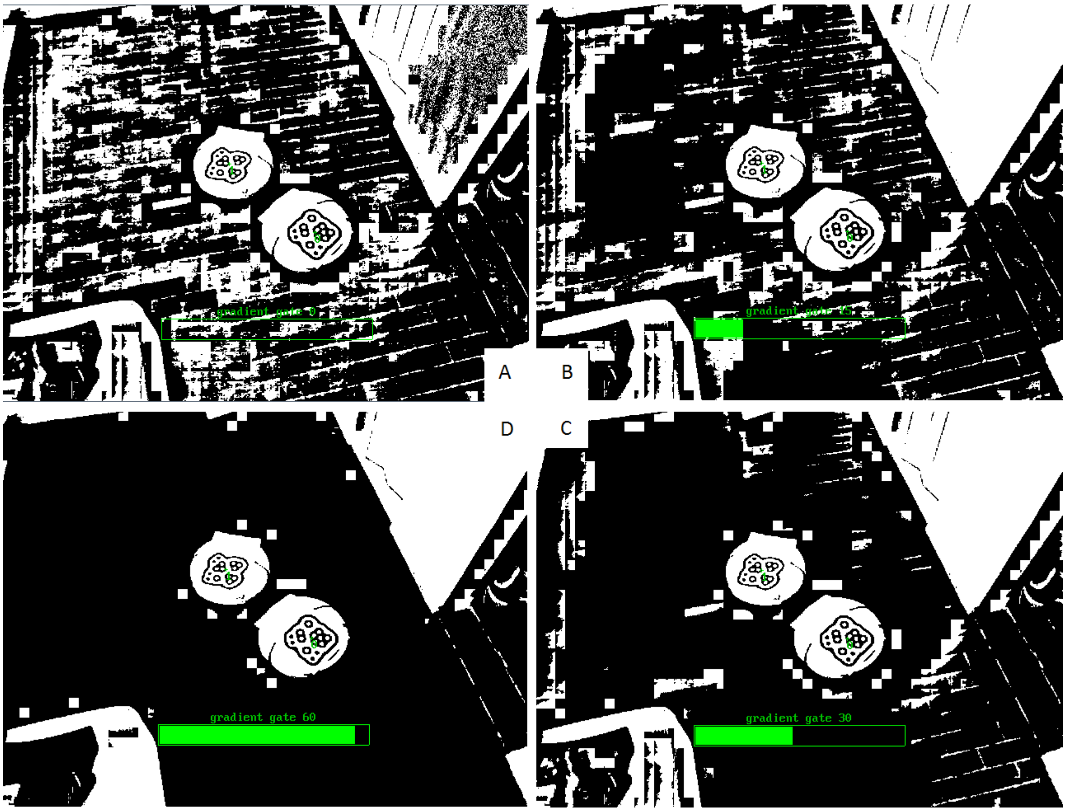

Figure 18: Comparison between four levels of gradient filter gate, clockwise from top left: a) G_S= 0; b) G_S= 15; c) G_S = 30 and d) G_S= 60.

After the binarisation phase, the system creates a feature graph from the scene and find the mark patterns sub-trees as described by Becina et al. [28]. Once the markers are identified their geometric center is assumed to be the agent position and it´s angle in relation to the camera (based on the original marker orientation) is assumed to be the agent´s orientation. That information is sent through the network broadcasted over a UDP socket.

Camera calibration and luminance equalization are automatically done by the reacTIVision system and the device driver of the camera.

3.6.Graphical Interface

A system interface was created for visualization and debugging reasons. It utilizes the IUP and the CD libraries to control windows and the canvas, both developed at TecGraf laboratory, in Rio de Janeiro [31] [32]. The window shows the whole arena, agents and virtual objects and it´s updated by the simulator main program right after positions and orientations are received via the UDP socket.

In Figure 19 is shown a screenshot of the graphical interface with four agents and one immobile object on screen.

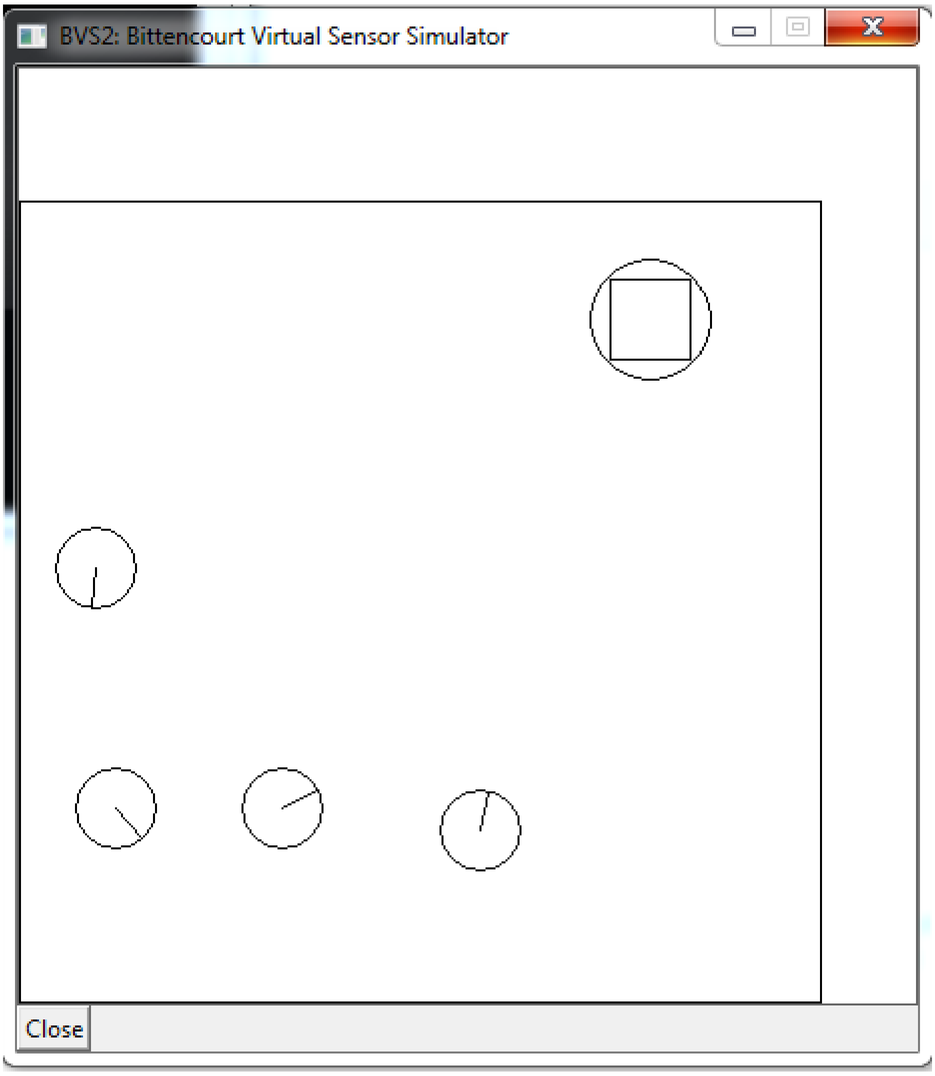

Figure 19: Screenshot of the graphical interface of the simulator with four agents and one object visible.

3.7.System Integration

All systems work integrated to form the full control cycle described at the beginning of the session, since the communication between the components relies on different protocols (illustrated at Figure 9) issues about synchronicity and timing must be addressed.

By the other side there are certain advantages in the selected communications channels, on of then is ease for parallelization of the system giving scalability to the platform. This scheme is described in next subsections.

3.7.1.UDP

The vision system communicates to the simulator via a UDP Socket, what may introduce latency and eventually lags due packet losses, but this solution also permits the parallelization of the simulator process. Parallelization can be achieved by broadcasting the vision system packets and running distributed simulators (each simulator instance calculates a set of agents' sensor readings and transmits just to this set of agents in a multicast scheme).

3.7.2.RF Link

Following the distributed mechanism described above the RF communication will be naturally distributed, if each simulator instance has its own RF Hardware. The topology of the distributed scheme is on Figure 20. This setup turns the whole system highly parallelizable.

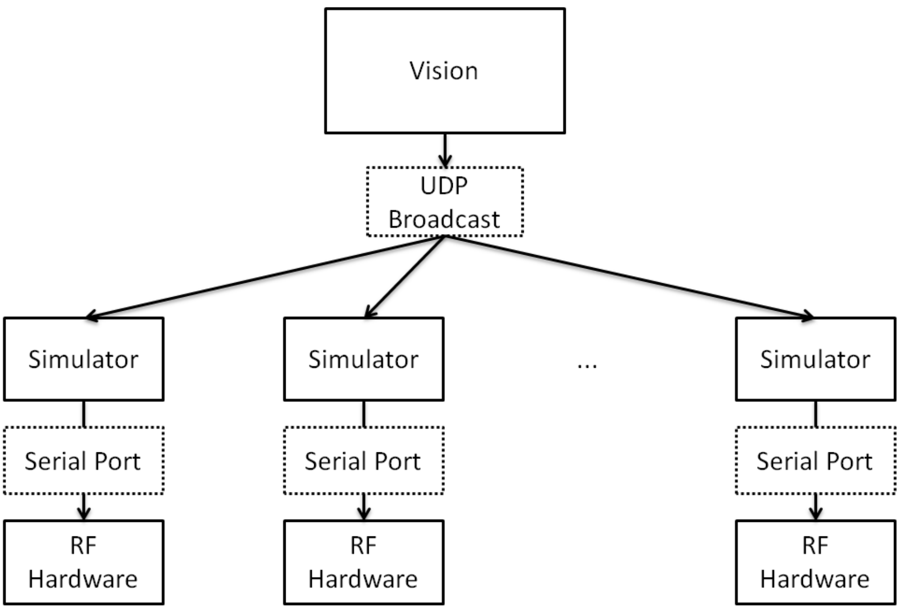

Figure 20: Topology of the distributed simulator scheme, relying in the UDP broadcast capability.

4 Algorithms Implementation

Two algorithms were implemented on the platform, the path finding algorithm based on artificial pheromones and the collective transport aggregation phase, where agents position themselves for the active transportation.

Both algorithms share a common virtual sensor that is an ideal magnetometer that is capable of giving the global orientation of the robot, this sensor is used as a feedback sensor to the turning movement, together with physical encoder on robot´s wheels, in a simple control loop.

4.1.Path Finding

Inspired on the computer algorithm created by Dorigo et al. and the model of path finding by ants pheromones presented before, an adaptation for the robots was conceived and implemented on the platform.

The original algorithm is crafted for graphs, a discrete space. To accommodate that requirement the robotic version tessellates the arena space in quadrants of equal sizes and creates a neighborhood graph of quadrants, as illustrated at Figure 21.

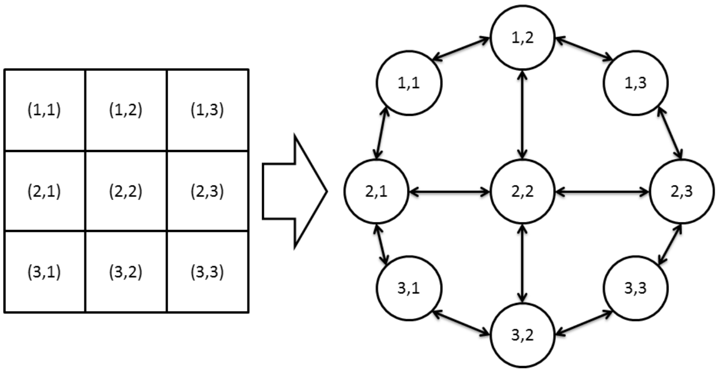

Figure 21: example of arena tessellation and graph representation of quadrant neighborhood.

The problem is to find a certain node, that represents the food source, and find the shortest path from the nest to it.

Each node is initialized with a constant amount of pheromone τ_0, the original metaphor consider the pheromone on the edges of the graph, but for the practical implementation the nodes (or quadrants) that contain the amount of pheromone. The algorithm also relies on a probabilistic transition rule similar to the one presented on chapter two, but considering that all surrounding nodes have the same weight (the distance is the same to all four neighbors); hence the factor η on equation (2.2) is constant for all cases. And differently from the original algorithm, the pheromone evaporation occurs in real time, since there are no tours to count as iterations. Agents moves by the arena according to the probability transition rule, once the agent reaches the target it comes back to the nest by the same way releasing a dose of pheromone on each node that is incorporated to the original amount of the node. The way back is reconstructed by storing the past nodes in a list, further on called memory.

With those alterations the robotic version developed resembles more the biological model than its graph traversing version. With the presented simplifications the general transition rule is written by the equation:

$$P_d = \frac{(\tau_d)^n}{\sum_{k \in K} (\tau_k)^n} \qquad (4.1)$$

Where:

P_d is the probability of choosing the direction d;

K is the set of all possible directions;

τ_i is the amount of pheromone in the node at direction i;

n is an adjustable nonlinearity factor.

An example of a typical situation that an agent may face is illustrated bellow on Figure 22.

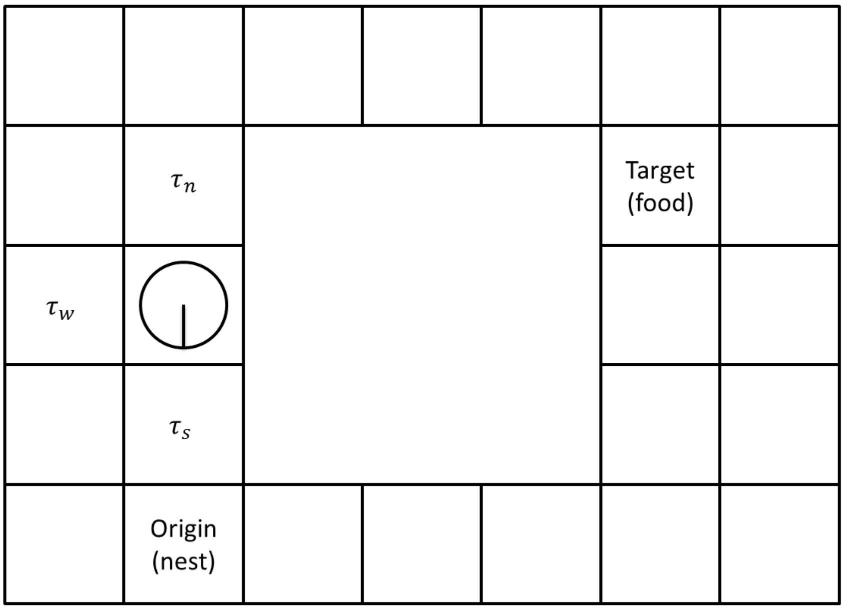

Figure 22: A situation where the agent must select one path based on the pheromone readings.

In this particular case the probability of the agent move to north direction is given by (with $n = 1$, linear case):

$$P_n = \frac{\tau_n}{\tau_n + \tau_s + \tau_w} \tag{4.2}$$

Pheromones have a decaying factor of ρ that follows the pheromone evaporation equation:

$$\tau_{(i,j)}(n) = (1-\rho)\tau_{(i,j)}(n-1) + \Delta\tau_{(i,j)}(n) \tag{4.3}$$

Where:

$\tau_{(i,j)}(n)$ is the amount of pheromone at node (i,j) at step n.

ρ is the evaporation rate.

$\Delta\tau_{(i,j)}(n)$ is the amount of pheromone left by an agent at node (i,j) at step n.

If the $\tau_{(i,j)}(n)$ becomes less than 10 it artificially is set to 10, because nodes cannot have no amount of pheromone. This restriction is due equation (4.2) that have the sum of the pheromone amounts as denominator.

With those rules, there is an emergent convergence to the shortest path to be the one with the strongest pheromone trail that creates a positive feedback loop attracting more agents to it. This emergent effect happens because the longer paths require a bigger travel time to the agent causing it´s natural evaporation rates to decrease the probability of this path to be chosen by the next agent.

Because of the discrete implementation the robots have only a reduced set of movement instructions, they are: Step forward, turn 90 degrees clockwise and turn 90 degrees counter clockwise. All necessary movements can be achieved with combinations of those three (the turn clockwise was unnecessary, but it accelerates the turning to higher angles).

4.1.1.Pheromone virtual sensors

For this algorithm virtual sensors capable of detecting the amount of pheromones in the four neighboring nodes (one in each cardinal direction) were implemented. Those are ideal sensors of a virtual artifact, the pheromone trails, and are limited to read only the amount of pheromones in the immediately connected nodes.

It´s considered that the sensor is capable of instantly reading all four (or less in some cases) neighboring nodes, or quadrants.

The sensor implementation is based on a four data packets, as described in chapter 3, protocol, where each packet contains the agent ID, the sensor ID, one byte indicating the direction and the last byte indicating the amount of pheromone (with special case of 255 to a not available node, in case of walls).

4.2.Collective Transport

The collective transport algorithm is a direct implementation of the reactive and behavioral approach presented by Kube and Zhang, but with modified internal implementations of each modular behavior. Some practical aspects were also treated with special attention to performance and energy consumption.

This algorithm relies mainly on the reactive paradigm, what means that agents are simply reacting to environment changes. In this case the environment is perceived by the virtual sensors. To this algorithm three kinds of sensors were implemented:

Goal Sensor

The goal sensor is able to feel the direction of the goal if it is within a certain threshold. It's a radial sensor that returns the angle and the strength of the signal.

Robot Sensor

This sensor is capable of detecting another robot moving nearby, again is a radial sensor that can tell the distance and the angle of close robots.

Obstacle Sensor

Is a model of an ideal radial LIDAR scanner sensor, which means it can detect any obstacle inside it active surrounding area.

And five behaviors, that are combined by subsumption operator as described in chapter 2 at Figure 7. They are:

Find behavior

This state is active only in idle state when no stimuli are received. The agent moves randomly, making fixed size step forward then turning to a random direction to make the next step forward. Creating a random movement pattern, that is repeated until any of the sensors receive a stimulus.

Follow Behavior

This state is active when the robot sensor detects another agent within a safe distance, as explained in Figure 23. The agent aligns itself with the detected agent and moves in the same direction keeping the distance.

Slow Behavior

The activation of this state is conditioned to the distance to the detected agent by the robot sensor, it´s only triggered when the distance is below the safe radius. In this state the agent stops moving forward, and waits until the slow behavior is deactivated (automatically triggering the follow behavior if no other sensor is stimulated).

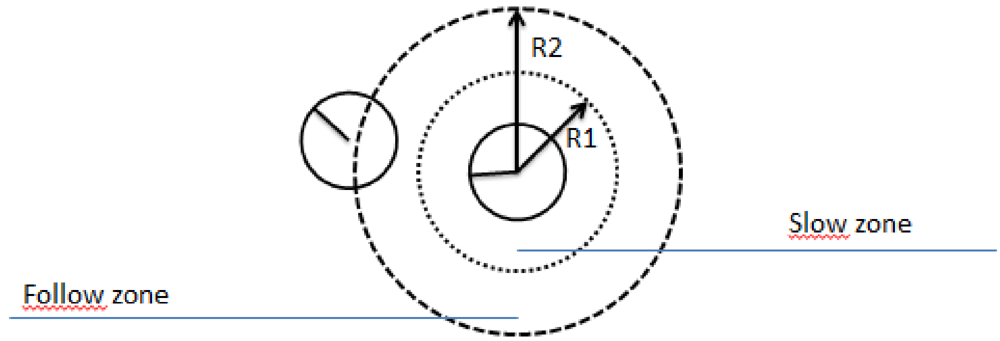

Figure 23: Areas of different behavior triggering for the robot sensor, R1 and R2 can be adjusted according to arena size and number of robots

Goal Behavior

Activated only by the “sight” of the goal, this behavior makes the agent aligns itself towards the detected direction of the goal (given by the goal sensor) and move forwards until the objective is reached or the obstacle sensor triggers the next, and most proprietary, behavior.

Avoid Behavior

A reaction to the obstacle stimulus, this behavior makes the agent contour the obstacle by a rotation in its own axis until a clear direction when the agent moves forward. Repeating this action makes the robot avoids complex obstacles and walls, as illustrated in the example case of Figure 24.

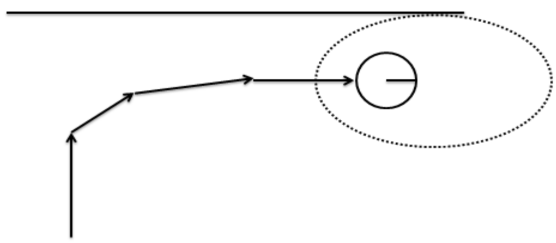

Figure 24: Agent’s path avoiding a wall using the obstacle sensor

Those local sensors and behaviors combined generate an emerging global behavior of the system that causes and aggregation of agents surrounding the goal, or prey in a metaphor of the biological model. This phase is crucial to the development of further collective transport implementations.

5 Results

In this session details about the algorithms implementation and tests are explained and results are discussed. First some general features of the platform are tested and evaluated, following by the results of experiments on both algorithms.

5.1.Platform testing

To ensure that the functions of the whole system are correct a simple test with two robots was constructed. In this test a predator-prey behavior is used to evaluate the computer vision system, virtual sensors simulation, RF communications between the simulator and the agents, agent's programmability and locomotion. With this integration test the basic platform functionality can be assured in terms of basic software and hardware.

In this test one agent becomes a predator, that is, it wanders around until it detects a prey then follows it with the goal of reaching it. The other agent acts as a prey, it wanders slowly by the arena and when a predator is detected it flees in the opposite direction. Both agents' behavioral diagrams are shown at Figure 25.

In the sequence of images on Figure 26 the evolution of the predator-prey system with images from the simulator and captures from the camera can be seen.

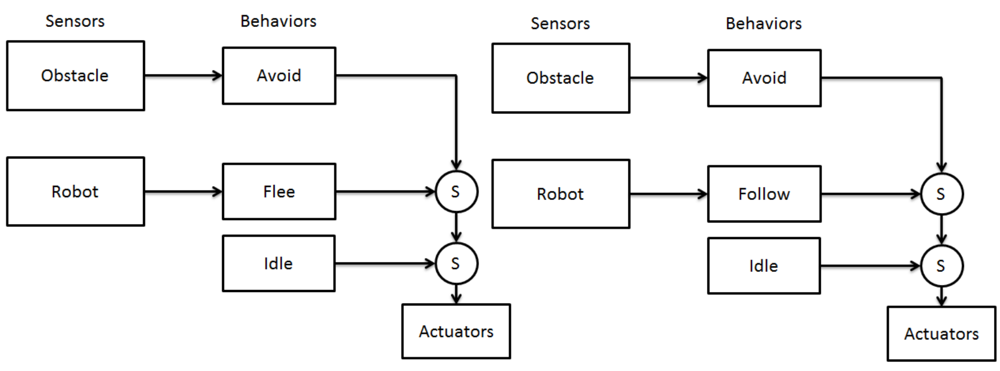

Figure 25: Behavioral diagrams of the predator agent (right) and the prey agent (left).

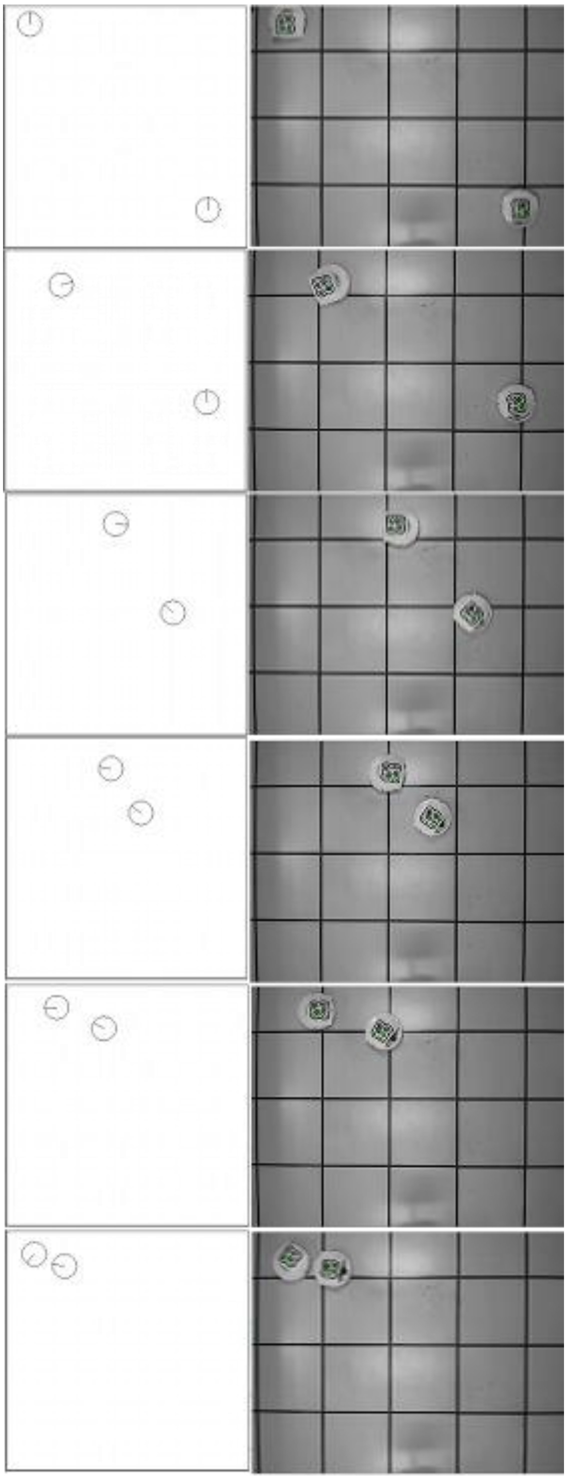

Figure 26: Time lapse of the predator-prey experiment with views from the graphical interface of the simulator and the capture of the camera

With this simple example, all systems could be tested. In slow speeds the performance was as expected, but for higher agents velocities the frame rate of the camera drops from 30 FPS (Frames per Second) to lower than 15 FPS causing lags on the simulator. Lags interfere with the system as a delay in the reading of the sensors.

5.2.Path finding results

The first test is to make a single robot detect and follow a predefined pheromone trail. Then analyze it´s behavior and adjust the parameters of evaporation and maximum pheromone amount of each node (node saturation). The simulated arena initial condition is shown at Figure 27.

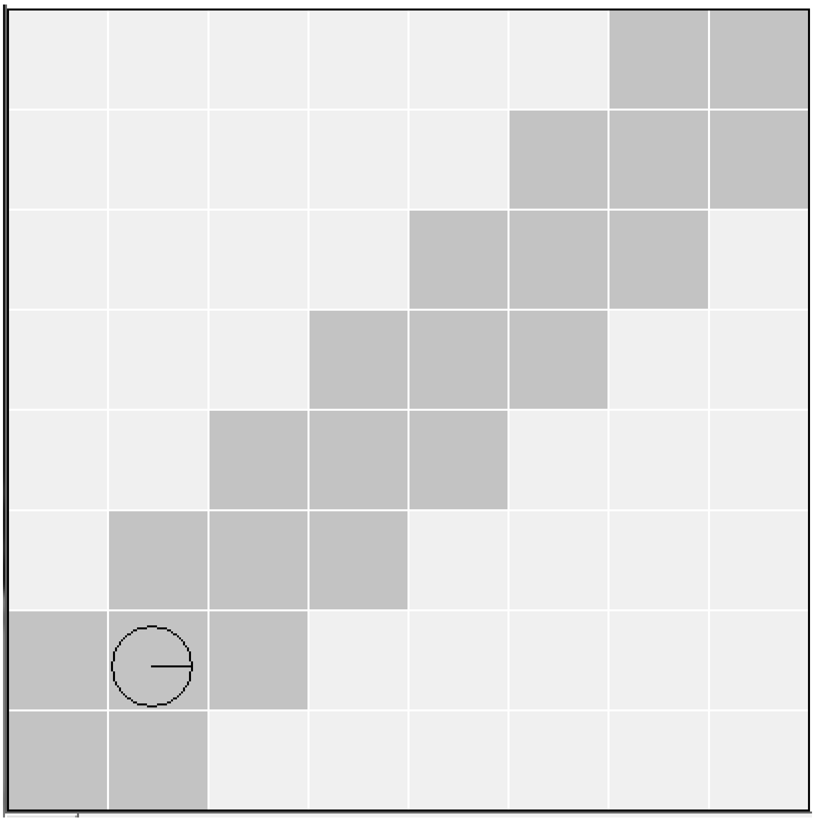

Figure 27: Screen shot of the simulator with artificial pheromone trail (darker regions have a higher amount of pheromones).

With this simple experiment with the pheromone evaporation rate $\rho = 0.09$ what, solving the recurrence equation (4.3) gives:

$$\tau_{(i,j)}(n) = c(1 - 0.09)^{(n-1)} \tag{5.1}$$

Where c is the initial amount of pheromone at node (i, j), and the saturation value (the maximum amount of pheromone per node) is 255. By this equation a saturated node becomes with the minimal amount of pheromone (10) in 35 simulation steps.

The next phase was the evaluation of the robots with a more complex arena, with a gap in the middle shown on Figure 28.

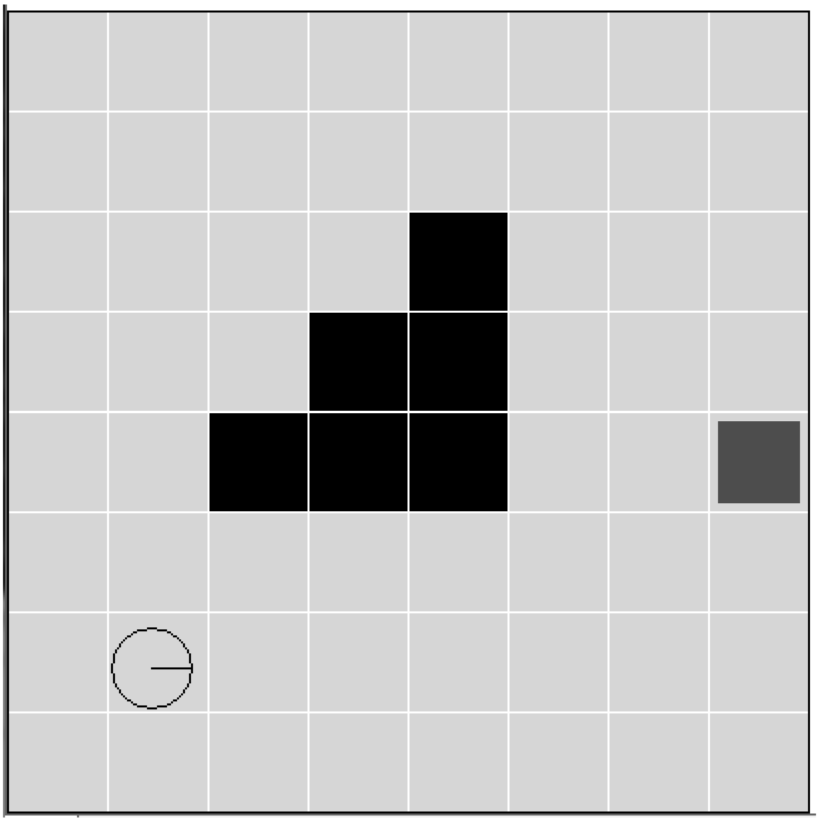

Figure 28: arena with virtual walls in the middle (black areas), agents are forced around it to reach the target (marked with a darker inner square).

Running the algorithm in this scenario with four agents, the following results were found:

a) To find the first path (starting from bottom-left corner in an unbiased arena)

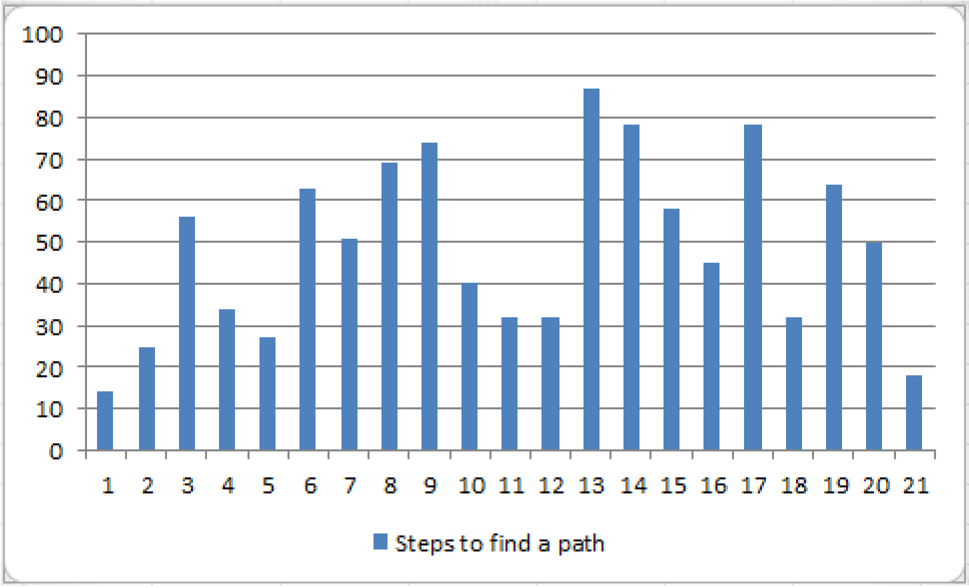

Figure 29: Number of steps to find any route. $\mu \cong 49.00$ e $\sigma \cong 21.37$

b) Steps to converge from the first found path to optimal or quasi-optimal path.

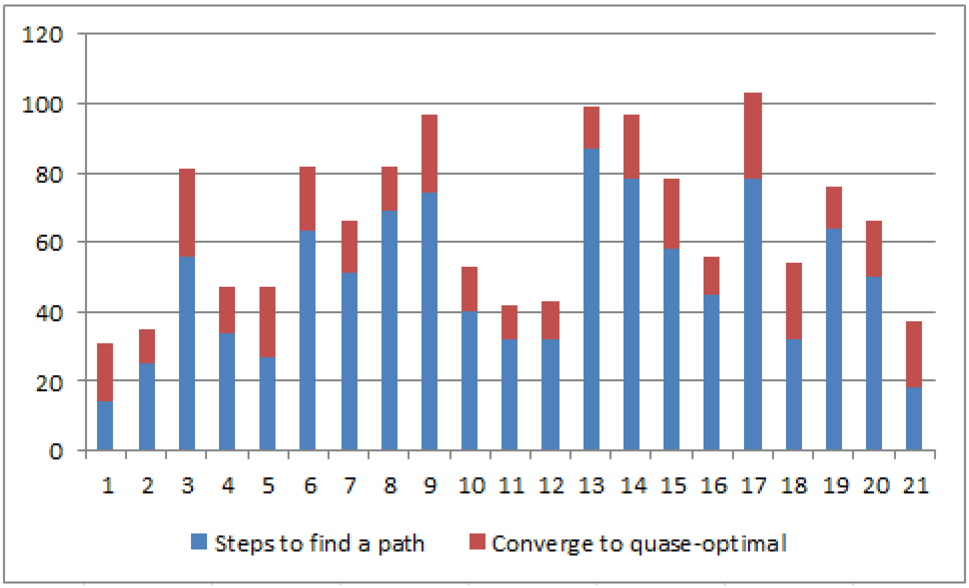

Figure 30: Convergence to optimal path.

Quasi-optimal concept was used because the optimal path is infeasible because of the possibility of collision between agents, so small detours of the optimal path are considered valid solutions.

Those results are preliminary and the number of samples is insufficient to make any robust inference from them, but as a qualitative indicator those results show the functionality of the platform and algorithm.

5.3.Collective Transport results

For the collective transport, a group of four robots was deployed at random starting positions to aggregate around an also random positioned heavy prey. On those tests the prey was a heavy carton box. An example run is shown on Figure 31.

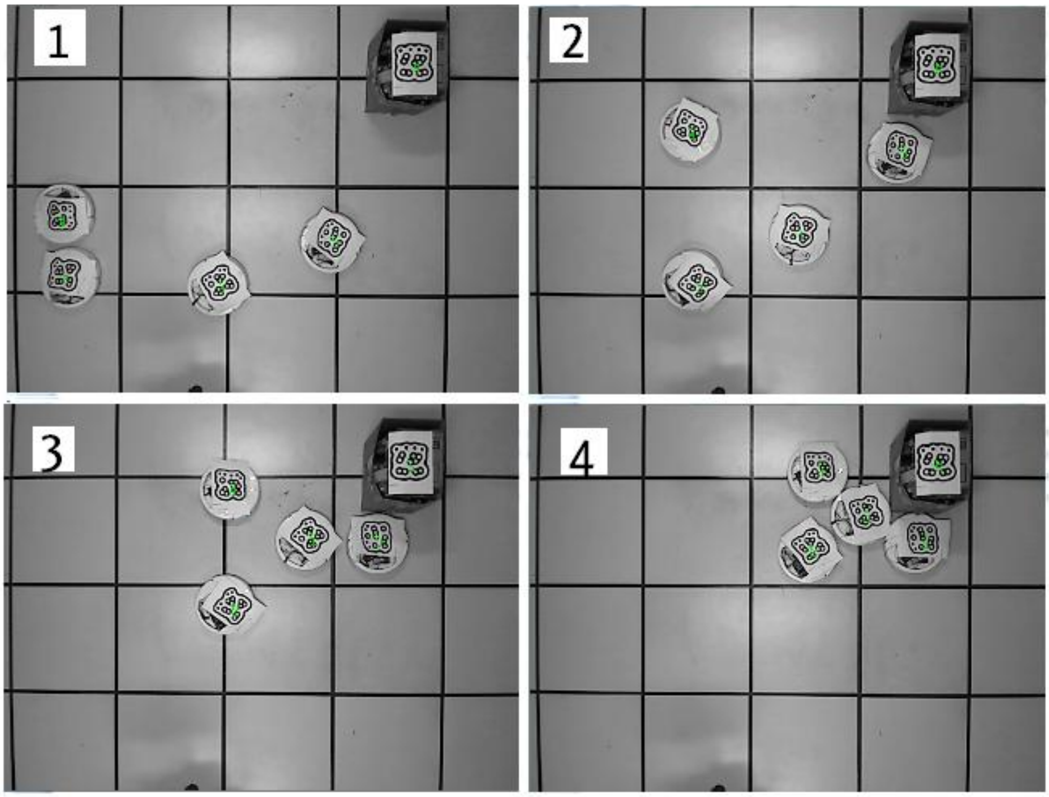

Figure 31: four frames of an example run of the aggregation algorithm.

The implemented algorithm is considered naïve, for its not treating cases where the agents get stuck, or in a behavioral deadlock (an example is when two agents get very close to each other causing both to stop and no other stimulus causes a change of behavioral state, an example case is in frame 1 of Figure 31 where the two agents in the left halted). Therefore some tests were executed to measure the average time of convergence of the agents to the prey.

In 20 complete runs (runs with any kind of deadlock were discarded) the average time to reach convergence was $\mu = 12s$ (seconds).

5.3.1.Centrally Coordinated Collaborative Transport

To demonstrate the capability of the platform to perform coordinated control a central coordinator to the collaborative transport was implemented. In this example three agents move a carton box to three directions. The configuration of agents' position, orientation and movement was pre-configured in a central coordinator implemented at the simulator. Results of this test are shown at Figure 32.

In those configurations each agent receives a command indicating the direction of the movement and it reacts accordingly:

Left

The agent bellow the object turns left and moves one step forward, the agent at the left side of the object moves one step backwards and the agent at the right moves on step forward pushing the box.

Right

The agent bellow the object turns right and moves one step forwards, the agent at the right side of the object moves one step backwards and the agent at the left moves one step forward pushing the box.

Up

The agent bellow the object moves one step forward pushing the box, the agents at both sides turn upwards (left side agent turns 90 degrees in counterclockwise direction and the right side agent turns 90 degrees in clockwise direction) and moves on step forward.

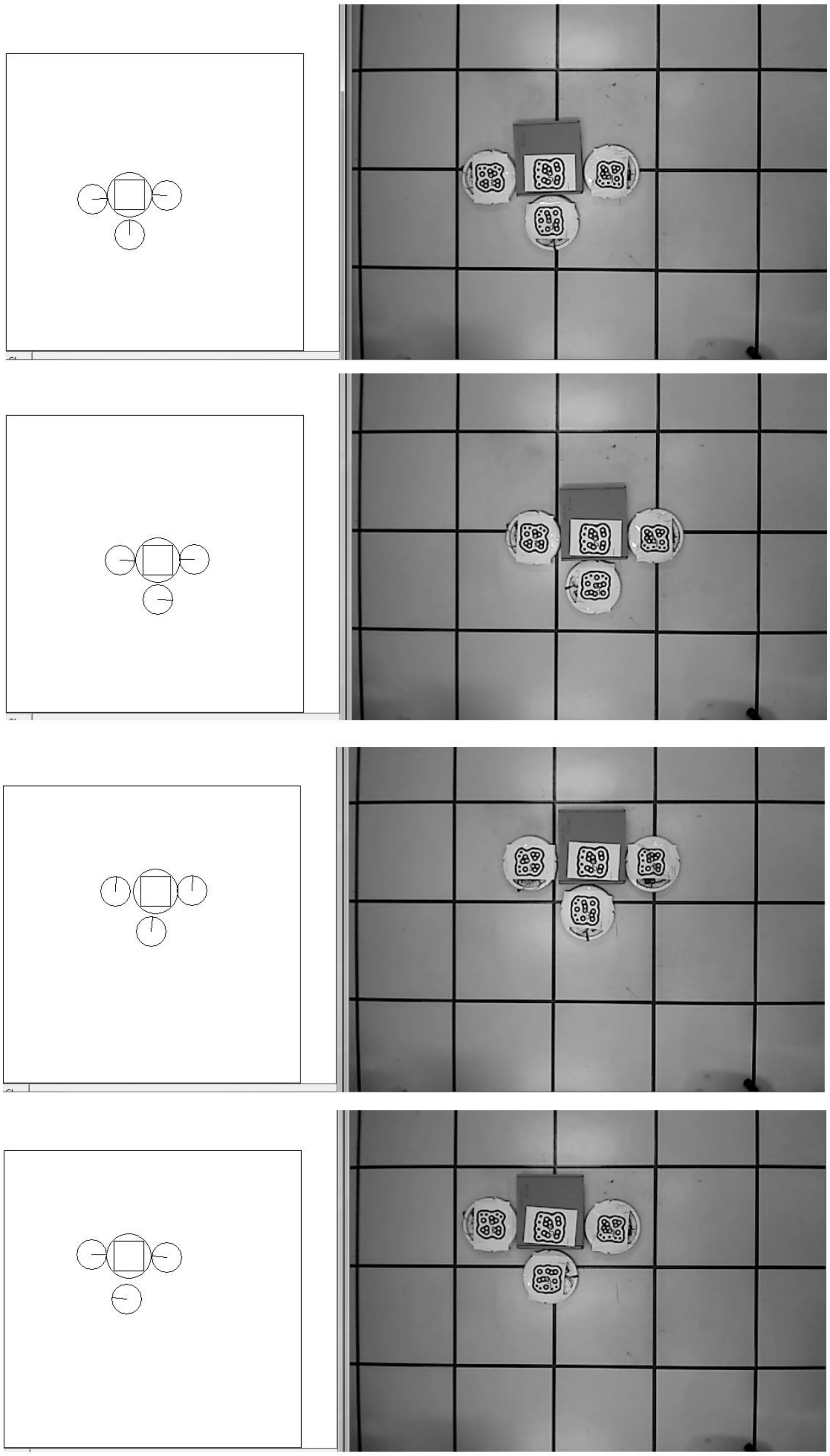

Figure 32: Collective transport of a box by three agents, in the sequence of movements: Right, Upwards and left. Initial position is shown in first frame.

5.4.General observations

Both tests cannot be considered complete for two main factors.

The first one is the reduced arena size, caused by the small distance between the floor and the camera (hanged on the ceiling of the test room). This reduced arena size, compared to the size of one agent, makes the algorithms not very representative to solve the proposed problem thus causing an imprecise test scenario.

The second main issue on the realized tests was the lack of samples to realize a statistical relevant amount of experiment, caused mainly by technical difficulties related to hardware and space issues.

But, even with the related issues, results were able to proof the functionality of the platform and the concept of the algorithms.

6 Conclusions

This work presented a complete platform for experimental analysis of a group of robots that is generic and extensible to accommodate different kinds of multi-robots (and even single robot) experiments. Based on commercially available materials and components it can be built with low budget and is a valuable tool for research. Within the developed platform two algorithms have been successful implemented, one is a biological model based path finding and the other a reactive implementation of the aggregation phase collaborative transport.

Results are still inconclusive due short number of experiments made, limited space and resources. By the other side the capacity of expansion of the platform is a factor that can ease future developments, turning the developed work into a starting point for more deep researches.

Other conclusion is the robustness of swarm robotics and the confirmation that some theoretical models can be physically implemented without meaningful alterations on a generic hardware, validating both, the platform and the algorithms implementation.

6.1.Virtual Sensors approach

The virtual sensors approach was introduced as an experimental mechanism and showed to be a very useful scheme for multi-robots experiments, mainly for:

- **Cost Effective**

  With virtual sensors the sensor price is irrelevant since the user just need it´s theoretical model implemented in the platform. The only hardware needed is the nRF24L01 radio link that is largely available on electronics stores for affordable prices even for students. Future version may rely on different communication devices, including very low cost IR transmitters.

- **Fast response speeds**

Even the response speed hasn't been tested directly the sensor speeds was not a bottleneck on the realized tests. So for a small amount of simple sensors the mechanism has a reasonable perceived performance.

- **Freedom to the user**

Users can change their entire sensorial setup just reconfiguring the platform, in this sense users are free to implement any set of sensors that can be computationally described. That segregation of the conceptual sensor from the hardware itself gives the final user the power to create arbitrary complex sensor schemes.

- **Lower hardware complexity**

Sensors usually are complex equipment to assemble, correctly install, calibrate and interface. With virtual sensors these hardware difficulties are mitigated, allowing the user to conduct experiments without struggling with miscalibration, bad installation or hardware and software interfacing.

Virtual sensors can be considered the main contribution of this work for its qualities referred above. Together with the computer vision system and the simulator virtual sensor form an ideal mechanism for low cost, easy to implement and fast experiments.

6.2.Platform Applications

Different kinds of experiments can be performed with the developed platform; even it was designed for multi-robot experiments single robot experiments can be realized seamlessly, besides the experimental applications in a wide range of areas, such as: SLAM (Self Location and Mapping), Collaborative Behaviors, Coordinated traffic and many others.

The same hardware and software can be used for autonomous robotics competitions, like robot football cups and others common robotic team games competitions.

7 Bibliography

[1] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, MA, USA: MIT Press, 2000.

[2] CAT, "CAT Minestar System," CAT, [Online]. Available: http://www.catminestarsystem.com/. [Accessed December 2011].

[3] Kiva Systems, "Kiva," Kiva Systems, [Online]. Available: http://www.kivasystems.com. [Accessed December 2011].

[4] IFR, "Annual Report of Robot Market," IFR, 2011. [Online]. Available: www.worldrobotics.org. [Accessed december 2011].

[5] S. Garnier, J. Gautrais and G. Theraulaz, "The Biological Principles of Swarm Intelligence," *Swarm Intelligence*, pp. 3-31, 2007.

[6] E. Şahin and W. Spears, "Swarm Robotics," pp. 10-29, 2005.

[7] Nissan Corporation, "Nissan EPORO Robot Car "Goes to School" on Collision-free Driving by Mimicking Fish Behavior," 2010. [Online]. Available: http://www.nissan-global.com/EN/NEWS/2009/_STORY/091001-01-e.html. [Accessed Dec 2011].

[8] D. Floreano, J.-C. Zufferey, S. Hauert, T. Stirling, S. Leven and J. Roberts, "Aerial collective systems," *Handbook of Collective Robotics*, 2011.

[9] eLua Project, "eLua," [Online]. Available: http://www.eluaproject.net/. [Accessed 19 April 2012].

[10] P.-P. Grassé, " La Reconstruction du nid et les coordinations interindividuelles. La théorie de la stigmergie," *Insectes Sociaux*, vol. 6, pp. 41-84, 1959.

[11] C. Reynolds, " Flocks, Herds, and Schools: A Distributed

Behavioral Model, in Computer Graphics," in *SIGGRAPH*, 1987.

[12] J. Deneubourg, S. Aron, S. Goss and J. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of Insect Behavior*, vol. 3, no. 2, pp. 159-168, March 1989.

[13] J. Deneubourg, S. Goss, S. Aron and J. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, no. 76, pp. 579-581, 1989.

[14] M. Dorigo, V. Maniezzo and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Ants," *IEEE Tans. Syst. Man Cybern. B26*, pp. 29-41, 1996.

[15] M. Dorigo and L. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Trans. Evol. Comp. 1*, pp. 73-81, 1997.

[16] M. Dorigo and L. Gambardella, "Ant Colonies for the Traveling Salesman Problem," *BioSystems 43*, pp. 73-81, 1997.

[17] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz, "Ant-Based Load Balacing in Telecommunications Networks," *Adaptative Behavior*, pp. 169-207, 1996.

[18] B. Bullnheimer, R. Hartl and C. Strauss, "An Improved Ant System Algorithm for the Vehicle Routing Problem," *POM*, 1997.

[19] R. Graham, E. Lawler, J. Lenstra and A. Rinnooy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey," *Annual Discrete Mathematics* , pp. 287-326, 1979.

[20] M. Dorigo, E. Bonabeau and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Santa Fe Institute Studies in the Sciences of Complexity, 1999.

[21] C. Kube and H. Zhang, "Collective Robotic Intelligence," in *Second International Conference on Simulation of Adaptive Behavior*, 1992.

[22] C. Kube and H. Zhang, "Collective Robotics: From Social Insects to Robots," *Adaptive Behavior*, pp. 189-219, 1992.

[23] C. Kube and E. Bonabeau, "Cooperative Transport by ants and Robots," *Robotics and Autonomous Systems,* pp. 85-101, 2000.

[24] M. Moffet, "Cooperative Food Transport by an Asiatic Ant," *National Geographic Research,* vol. 4, pp. 386-394, 1988.

[25] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, 1984.

[26] R. Brooks, "Challenges for Complete Creatures Architectures," in *First International Conference on Simualtion of Adaptative Behavior*, 1990.

[27] R. Ierusalimschy, Programming in Lua, Second Edition, 2006.

[28] R. Becina, M. Kaltenbrunner and S. Jordá, "Improved Topological Fiducial Tracking in the reacTIVision System.," Barcelona, 2005.

[29] R. Becina and M. Katelbrunner, "The Design and Evolution of Fiducials for the reacTIVision System," Barcelona, 2005.

[30] J. Bernsen, "Dynamic thresholding of grey-level images," in *8th International Conference on Pattern Recognition (ICPR)*, 1986.

[31] L. Figueiredo, M. Gattass and C. Levy, "Uma Estratégia de Portabilidade para Aplicações Gráficas Interativas," in *VI SIBGRAPI*, 1993.

[32] A. Scuri, "Canvas Draw, A 2D Graphics Library," TecGraf, 2012. [Online]. Available: http://www.tecgraf.puc-rio.br/cd/. [Accessed dec 2011].

8 Appendix A – Geometric calculation of angle measurements

The angle result from the Robot sensor is the angle, from the coordinate system, to the other robot.

Consider two agents $A_1$and A_2 with arena coordinates (X_1, Y_1, θ_1) and (X_2, Y_2, θ_2) respectively. (Figure 33)

The final angle γ is calculated based on the robot orientation θ_1 and the direction between the agent itself and the detected agentφ.

$$\varphi = atan2(Y_2 - Y_1, X_2 - X_1) \qquad (8.1)$$

The angle γ is calculated as

$$\gamma = [(\theta - \varphi) + 2\pi] \; mod \; 2\pi \qquad (8.2)$$

The result is summed with 2π to guarantee a positive final angle, and the modulo operation guarantees the final result in interval $[0{,}2\pi]$

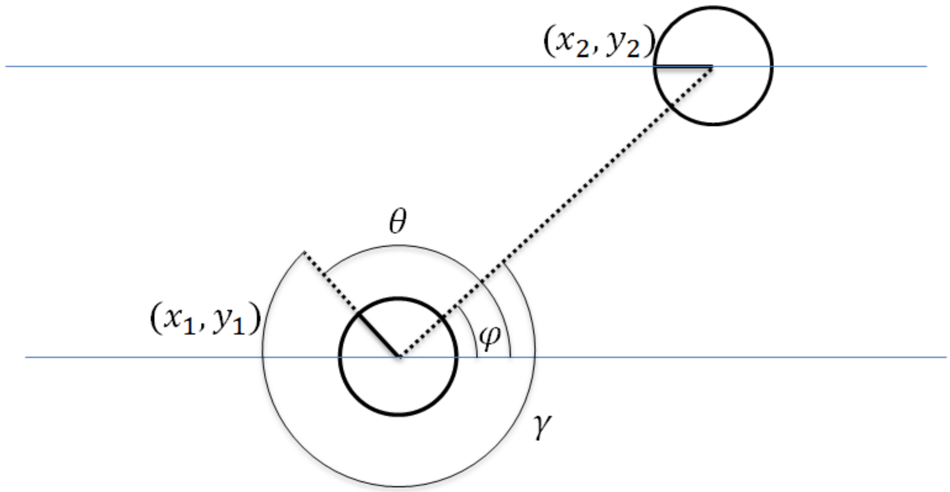

Figure 33: Sample case where $\boldsymbol{\theta}$ is the orientation of the robot and $\boldsymbol{\varphi}$ is the result of atan2 function. The final result is $\boldsymbol{\gamma}$.

9 Appendix B – Components Datasheets

This section contains the DataSheets of the main electronic components of the platform:

- Voltage Regulator LD33V by ST Microelectronics
- STM32 Microprocessor by ST Microelectronics
- FTDI USB to serial converter by Future Technology Devices International
- nRF24L01 radio transceivers by Nordic Semiconductors