ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

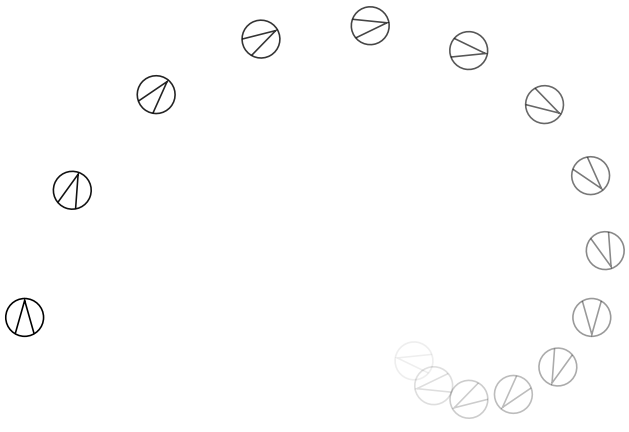
UNIVERSITÉ LIBRE DE BRUXELLES

Human - Robots Swarms Interaction

An Escorting Robot Swarm that Diverts a Human away from Dangers one cannot perceive.

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil en Informatique à finalité Intelligence Computationnelle

Anthony Debruyn



Directeur
Professeur Mauro Birattari

Co-Promoteur
Professeur Marco Dorigo

Superviseurs
Gaëtan Podevijn, Andreagiovanni Reina

Service
IRIDIA

Année académique
2014 - 2015

First Matter

Acknowledgements

I thank...

Mauro Birattari for his precious ideas and support during the whole year, for the construction of the augmented shoes, and also for proposing this master thesis topic

Gaëtan Podevijn for his precious ideas, support and patience during the whole year, for his attentive corrections of my texts, and also for proposing this master thesis topic

Andreagiovanni Reina for his precious ideas, support and patience during the whole year, for his attentive corrections of my texts, and also for proposing this master thesis topic

Anthony Antoun for his help in the construction of the electronic part of the shoes

Brian Delhaisse for the table tennis matches that I won, not for all the ones that I lost, but mostly for all the knowledge he shared with me

Lorenzo Garattoni for the help provided in the arena when I needed it the most

Family & Friends for their endless support from the beginning of my studies until now

Résumé

L'objectif de ce mémoire est d'introduire un nouveau type de protection pour l'être humain dans les environnements dangereux. Cette protection est basée sur la robotique en essaim. Un essaim de robots protège un être humain en augmentant les capacités de ce dernier. Les robots informent l'être humain à propos de zones dangereuses que seuls les robots sont à même de percevoir. En contraste avec la littérature courante qui explore davantage les solutions avec un canal de communication unidirectionnel depuis l'être humain vers les robots, notre solution prévoit un canal de communication bidirectionnel. L'être humain contrôle indirectement les robots en se déplaçant, tandis que l'essaim de robots encercle l'être humain et le met en garde contre les zones dangereuses invisibles. Les robots restent à la frontière de la zone dangereuse pour former une barrière entre l'être humain et le danger. Une paire de chaussure équipée de LEDs a été construite. Ces chaussures permettent aux robots de détecter l'être humain. Notre solution a été testée lors de nombreuses simulations et sur de vrais robots. Les résultats sont prometteurs, l'essaim encercle l'être humain et le prévient de dangers à proximité.

Mots-clés : robotique en essaim, escorte d'être humain, zones dangereuses, chaussures équippées

Summary

The objective of this master thesis is to introduce a novel type of protection for humans in dangerous environments. This protection is based on swarm robotics. A swarm of robots protects a human by augmenting his/her abilities. The robots provide the human with feedback about dangerous areas that only the robots are able to perceive. In contrast with the current literature that mostly explores solutions with a one-way feedback from the human operator to the robots, our solution provides a bidirectional feedback. The human indirectly controls the robots by changing his/her position. The swarm of robots, encircling the human, notifies the human about near invisible dangerous areas. The robots stay at the boundary of the dangerous area to form a barrier between the human and the danger. A pair of augmented shoes was built and equipped with LEDs. They allow the robots to locate the human. Our solution was tested heavily in simulation and on the real robots. The experiments conducted yielded promising results. The swarm of robots encircles the human and notifies him/her about dangerous areas.

Keywords: swarm robotics, human escorting, dangerous areas, augmented shoes

Contents

List of Figures

List of Tables

Main Matter

Chapter 1

Introduction

New progresses in robotics have opened a new branch of studies. Taking inspiration from social animals like ants, bees or fishes, researchers are now able to create groups of robots performing tasks that could not be undertaken individually. These groups are called swarms. We envision swarm robotics to be useful for applications like search and rescue, environment exploration, or oil spill cleaning (Dorigo et al., 2014). In swarm robotics system, each robot executes the same controller code. The interaction between the robots and between the robots and the environment enable the emergence of a collective behaviour. There is no hierarchy in the swarm. That is, all the robots behave autonomously. Research in swarm robotics is important as it underpins potential future disruptive innovations. Nanorobotics is going to be one of these innovations. The use of nanorobotics in medicine will grow over the next years. It could be one of the future solutions to cure cancer or other diseases by targeting the faulty constituents of the body with a swarm of very small robots. Swarm robotics is also important because it could constitute an adequate solution to other real problems. Swarm robotics systems are robust, i.e., losing a robot of the swarm is not a critical issue. Some tasks are dangerous for humans (e.g., demining, search and rescue) and the use of robots is preferred to avoid any injuries. However there is also a high risk of losing robots. Hence this task requires a fault-tolerant approach that swarm robotics can provide. Thanks to their scalability and flexibility, swarm robotics solutions are also suitable for applications where it is difficult to estimate the resources needed to carry out the task (e.g., search and rescue, cleaning). Swarm robotics solutions quickly adapt to new operation conditions and are thus appropriate for tasks in environments that change over time (e.g., patrolling, disaster recovery, and search and rescue). Since the robots behave autonomously, swarm robotics solutions are also suitable for large or unstructured environments where no infrastructure, like a communication system

or a global localisation system, can be used to control the robots. Examples are underwater or extra-terrestrial planetary exploration, surveillance, demining, and search and rescue (Dorigo et al., 2014).

Even though swarm robotics can carry out tasks autonomously, they do not have a global understanding of the environment and of the task that they must carry out. A human operator can, therefore, interact with those swarm systems to issue them commands (i.e., what type of tasks to carry out and where to carry out the task). Issuing commands relies on a one-way communication between the human and the swarm of robots. For swarm robotics to be adopted outside of research laboratories and tackle real world issues, one should always be able to take control of the swarm at any time. This is a legitimate safety requirement when considering the use of a large amount of potentially harmful robots around humans. Hence one can understand the vital aspect of the interaction between human and swarms of robots. In this thesis, however, we did not implement a method granting the human to completely control the swarm. We leave it as a future work. For the purpose of this thesis, it was not necessary.

To date, little attention has been paid to robotics swarm feedback, i.e., messages sent from the swarm to the human operator. Most of the research works focus on issuing commands to swarms. However, there might be situations in which the human does not know everything (e.g., where a gas leak comes from). For instance, we can leverage swarm robotics systems to help humans move in dangerous environments. To the best of our knowledge, no study has considered a human being escorted by a swarm of robots in a dangerous environment.

In this thesis we make an attempt to address this lack of consideration. We study how a swarm of robots can help a human move in dangerous environments. We use inspiration from flocking and pattern formation to allow a swarm of robots to prevent a human from entering dangerous areas. In this thesis, these dangerous areas are invisible to the human. These dangerous areas could, for instance, contain mines or be radioactive, or present another type of danger (e.g., poisonous gas, unstable floor). In order for the human to avoid these dangerous areas, we designed a swarm system that escorts the human in an environment. There is a bidirectional feedback between the swarm and the human. The robots warn the human about the danger, and the human indirectly controls the position of the robots by changing his/her position. It contrasts with most of the studies that only contain a unidirectional feedback (the human controlling the swarm). For the robots to stay around the human, we had to find a way to make the human detectable for the robots. We built an entirely new portable

device for that purpose.

Our solution went through a series of tests. The majority of the tests was used to incrementally improve the solution. These tests were made in a simulator and on real robots. At the end of the implementation, more tests were performed to assess the quality of the solution. Overall results are promising: the robots are following the human and warn him/her about near dangers.

This thesis is outlined as follows. In chapter 2 we present studies related to this master thesis. We introduce swarm robotics and provide a state of the art in human - robots interaction, and human - robotic swarm. In chapter 3 we expose the problem we address in more details. We also provide a complete description of our solution. Chapter 4 contains detailed explanations on the tests we conducted. Finally we conclude our work in chapter 5.

Chapter 2

State of the Art

In this section, we provide some general insight in the world of human - robot interaction and swarm robotics. We then review the literature of human-swarm interaction.

2.1 Human - Robot Interaction

Human - robot interaction has become a subject of great importance. A lot of research has been done in this field recently in order to mimic the human communication behaviours in different circumstances. One of the most interesting examples is Minerva (Thrun et al., 1999). Minerva is a tour-guide robot that educates and entertains people in public places. It has a motorised face to imitate human emotions. It was tested in a museum where it successfully performed what it was created for while learning. Dautenhahn et al. (2006) studied how a robot should best approach and place itself relative to a seated human subject. They performed experiments where a human asked for an object that would then be fetched by a robot. The robot would come to the human using different approach directions. The results reveal that most humans disliked the frontal approach and preferred to be approached from either the left or the right side. Itoh et al. (2006) studied the negative physical and psychical effect of robots on humans. Their study aims to complete other studies that only subjectively measured these negative effects by questionnaires. They built a device to measure physiological parameters such as respiration, heart rate, perspiration and pulse wave, and arm motion to obtain an objective measure of the effects of robots on humans. Using this device, the robot was able to react to the stress of the human and move accordingly in order to decrease the stress. Bethel et al. (2009) investigated on non-facial and non-verbal methods of affective expression for

enabling social interaction in appearance-constrained robots and found 5 main methods: body movements, postures, orientation, color, and sound. Example of appearance-constrained robots can be found in search and rescue, law enforcement, and military applications. They conducted a study involving 128 participants in a confined-space simulated disaster site with the robots interacting in the dark. *'Statistically significant results indicated that participants felt the robots that exhibited affective expressions were more calming, friendly, and attentive, which improved the social human-robot interactions'* (Bethel et al., 2009).

The objective of this project is to protect a human from dangerous areas using robots. For this purpose, the robots have to recognise and locate the human. Several solutions have emerged trying to solve that problem. However only few of them considered the sensor to detect the human as being close to the ground. These solutions rely either on 2D laser range data on an horizontal plane, or RGB-D data. RGB-D (Red Green Blue-Depth/Distance) associates the usual video stream to a 'depth' channel (Wikipedia, 2014). Gritti et al. (2014) propose a new robust solution for people detection and tracking using Kinect (Microsoft, 2015) or Asus Xtion (Asus, 2015). The floor must be flat. It uses a statistical classifier trained with a big dataset containing real-world data. After classification either as human leg or obstacle, the data is fed to a tracking algorithm to track people. Such lasers are very expensive, and at the time of realising this thesis it was not possible to use Kinect sensors on the robots. As robotics swarms imply the use of many robots, buying one of these sensors for each robot is too expensive. Thus we have to implement our own solution based on the sensors available on the robots that we have.

2.2 Swarm Robotics

For Şahin (2005), swarm robotics is defined as *'the study of how large numbers of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment'* (Şahin, 2005). These groups of agents do not depend on any external structure or centralised control (Dorigo et al., 2014). The development of swarms of robots rely on the principles of swarm intelligence. Swarm robotics can be separated from other robotic studies by the following characteristics (Brambilla et al., 2013):

- Robots are *autonomous*;
- Robots evolve *in the environment* and can interact with it;

- Robots' interactions are *local* (sensors and communications);
- No *centralised control* or *global knowledge*;
- Robots *cooperate* to achieve a certain goal.

As in swarm robotics, one is always looking for *robust*, *scalable* and *flexible* systems, the main source of inspiration is the group of social animals (e.g.: ants, birds, fishes). When some of these simple animals gather in groups, they are able to perform tasks that could not be achieved individually (collective behaviour emerges from local interactions). Below are listed the definitions of these three terms (Brambilla et al., 2013):

Robustness: Resistance against *loss of group entities*. One can increase it by adding redundancy or remove the need for a leader.

Scalability: Low variation in the performance of a system with respect to the *size of the system*. It can be increased by encouraging local interactions, such as sensing and communications.

Flexibility: Low variation in the performance of a system with respect to the *type of environment or the task*.

With these definitions in mind, we can explain swarm engineering as:

> *'Swarm engineering is an emerging discipline that aims at defining systematic and well founded procedures for modeling, designing, realizing, verifying, validating, operating, and maintaining a swarm robotics system.'*
> *- Brambilla et al. (2013)*

In this thesis the robots have to protect the human. We opted for a solution where the robots escort the human. For that purpose they need to stay close to him/her. The group composed by the human and the robots have to move in a coordinated manner. The scientific term for this coordinated behaviour is *flocking*. Many examples can be found in nature, e.g. birds, fishes. In nature this behaviour offers many advantages. The most common approach for implementing a flocking behaviour is virtual physics design. One can also obtain these types of motion through artificial evolution (Brambilla et al., 2013). Reynolds (1987) was the first to propose a model for flocking. It was meant for computer

graphics applications. With only 3 simple rules that every agent in the swarm has to respect, he was able to obtain realistic simulated swarm behaviours. The 3 rules are as follows:

Separation: Each agent tries not to be too close from its neighbours to avoid collisions.

Alignment: Each agent changes its heading to the average heading of its neighbours.

Cohesion: Each agent tries to centre itself at the average position of its neighbours.

Baldassarre et al. (2003) used evolutionary techniques to develop collective flocking behaviours. They demonstrate that these techniques are a powerful method for implementing collective behaviours. One of the most effective strategy that arose contained forms of 'situated specialisations'. Robots with identical evolved controllers have behaviours that depend on the circumstances. They think this is caused by the objective to reduce the interference between the attraction to the target and the need to maintain an aggregation state. Thanks to these specialisations, they observed that one of the robots was leading the way to the target while the others just tried to stay close to it. This is close to the idea behind the solution we want to implement.
Turgut et al. (2008) proposed one of the first implementations of Reynolds' (Reynolds, 1987) behaviour on real robots. Each robot knows the heading of its neighbours and their distance via a virtual heading sensor and infrared sensor respectively.
Çelikkanat and Şahin (2010) investigated on the very important problem of the control and guidance of a swarm of robots: how to control the swarm and to what extend it can be controlled. They guided a swarm of robots by controlling a minority of the agents inside the swarm: 'informed robots'. 'Naive robots' only consider the flocking objective while the 'informed robots' also took into account the direction orders given by the operator.
Ferrante et al. (2012) focused on motion control. They implemented a new way to translate the output of the flocking rules into wheel speed. They made the speed of the wheels proportional to the norm of the resulting direction vector. This technique can be implemented on the simplest robots, even those which cannot detect neighbours heading. They added a few 'informed robots' in the swarm and showed that the swarm could travel longer distances using this norm dependant technique instead of the constant speed technique.

2.2.1 Human - Robots Swarm Interaction

Human - Robotic swarm interaction is the study of how humans can interact with a swarm to control it and receive feedback from it (Brambilla et al., 2013). A proper feedback is needed by the operator in order to make the right decisions. Since swarms must ideally be autonomous and make decisions in a distributed way, it is difficult to insert a communication with a human operator in the system to gain control.

Currently, little attention has been devoted to the study of the interaction between humans and robotic swarms, how one can send instructions and receive feedback. One of the challenges in human-swarm interaction is the difference in perspective between the swarm and the human operator. The human only observes the global collective behaviour, not the local interactions or individual behaviours driving the robots. The simplicity of the hardware found on the robots, or the efficient synthesis of all the information sent by the robots (i.e., feedback) are also challenging. Most of the existing works in the literature present a major disadvantage: they require an extra layer between the group of robots and the human. This requirement might not always be satisfied when we remember that swarms like this are mostly destined to evolve in an unknown environment. The monitoring equipment necessary to operate the swarm may not be safely deployed. Furthermore, a synthesis of all the local information pieces must be done in order to provide an understandable state of the system to the human. This synthesis involves modelling, additional overheads and perhaps heavy computations, and the gathering of all information at a central point (eliminating by the way the distributed and decentralised properties of the swarm system) (Podevijn et al., 2012).
Daily et al. (2003) used a head-mounted display and augmented reality to add information right on top of the robot in the environment itself, suppressing the need for an additional display. Baizid et al. (2009) proposed a platform to interact with multiple robots simultaneously through a graphical user interface, or a head-mounted display, in virtual reality. They also studied how virtual reality abstraction affected the human perception and cognitive capabilities, i.e, they created a virtual environment by filtering useless information. McLurkin et al. (2006) developed an centralised graphical user interface taking inspiration from real-time strategy video games, where one must control armies. They also imagined a feedback approach based on LEDs and sounds. The robots transmit their internal state by applying to their LEDs and sound system a defined pattern, recognisable by the operator, now able to quickly understand the state of the swarm without looking at a supplementary interface.
Podevijn et al. (2012) argue that self-organised mechanism, as those ruling the be-

haviour of the swarm, should be used to provide feedback to the operator. They suggest that the best entity which could communicate the status of the system and the whole swarm is the swarm itself. They performed experiments using colour feedback to distinguish different internal states and split the swarms into groups to tackle different tasks.
Giusti et al. (2012) present a novel approach for distributed real-time recognition tasks using a swarm of mobile robots. Robots from multiple points of view exchange their local information to generate a global classification. They validated their solution on real robots for hand gesture recognition.
Ayanian et al. (2014) implemented an iOS application with multitouch. The multitouch gestures are translated into low-level commands for the robots composing the swarm. On the screen, the swarm of robots is represented as a bounding box that the user can manipulate to control the robots.

As swarm robotic systems are mostly destined to operate on risky floors, unknown environment, it would seem logical to consider their application in exploration and/or protection missions. However, at the time of writing this thesis, we could not find any study on the subject. Exploration experiments never included a human, or other living organism. The objective of this thesis is to address this lack of study by designing and implementing a protective behaviour executed by a robotic swarm.
The human operator is here part of the swarm system. The swarm has to protect him by preventing him from going into dangerous areas, in the same way a group of bodyguards protects someone. The swarm has to follow the operator anywhere to ensure permanent protection.
We believe this work to be important since it might open doors to new types of applications of swarm robotics: human protection, escort or swarm turn-by-turn navigation.

Chapter 3

An Escorting Swarm

As discussed in the introduction, the problem we want to address is the protection of a human evolving in a dangerous environment. The human is unable to see the danger. Chapter 2 introduced some of the works that are related to the problem. In this chapter, we will present the problem in details and the proposed solution at a high level of description. Then, we will describe the implementation details.

3.1 The Problem

Since the early days, human beings have explored new territories to expand their control and get a better understanding of the world surrounding them. Among those new territories, some were relatively safe but some were dangerous. To reduce risks, we have invented equipment, suits, and other kinds of protections (e.g. guards, sensors, alarms). In this work, we suggest a solution to the problem presented in this section.

Figure 3.1 shows a graphical representation of a possible scenario to our problem. This scenario will help exemplify the challenges we address. In this scenario, the human must go from point a to point b. Between these two points, the human might be confronted to hazardous areas. These hazardous areas are represented by the red circles in Figure 3.1. Example of such hazardous areas are radioactive zones, mine fields. The human cannot perceive them. The protection created should prevent the user from going inside those areas.

Exploration is not the only real application for the proposed solution that comes into mind. Rescue in disaster areas would also benefit from it (evacuation

of people to safe zones). A solution to this problem should be able to constantly protect the person using it, and constantly provide feedback. It should be robust and fit to the environment in which it would be used.

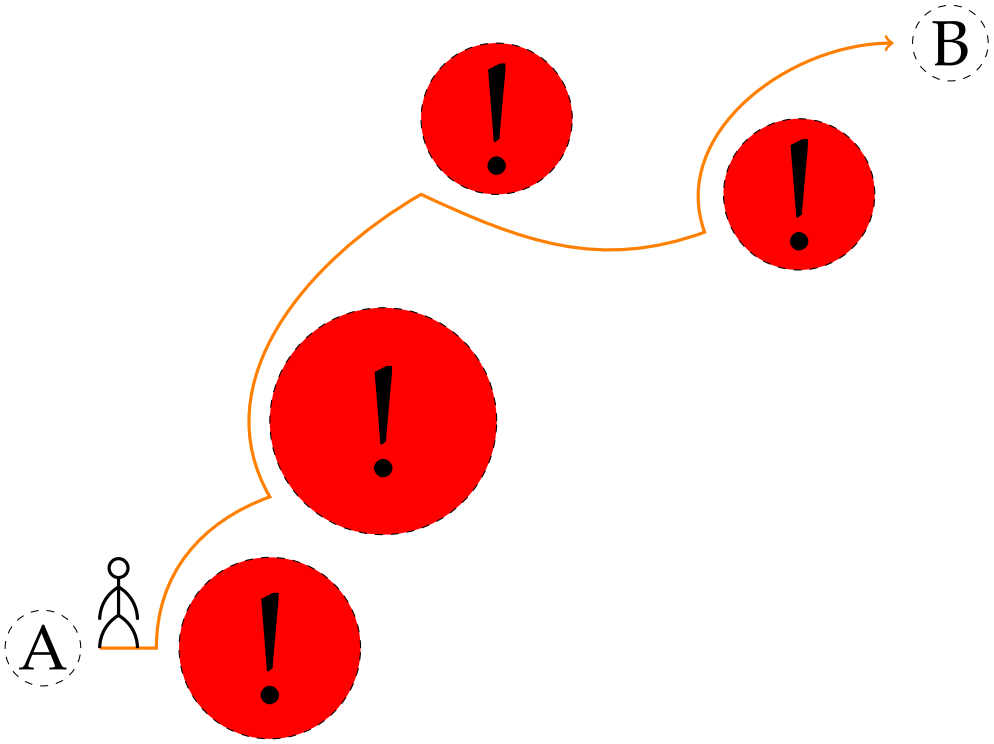


Figure 3.1 – Unknown dangerous environment: This image illustrates an environment, observed from above, in which a human must move from point *A* to point *B* while avoiding invisible dangerous areas. *A* is the start location, *B* is the goal ant the red circles represent dangerous zones. We provide in this thesis a solution to guarantee safeness in such circumstances. Possible applications for this type of solutions are: mine fields crossing and cleaning, radioactive areas avoidance,...

3.2 Solution

The solution we propose involves the use of a swarm of robots. Swarm robotics systems are well suited for this types of application because it is compatible with unknown environments thanks to its flexible, robust and scalable characteristics (Brambilla et al., 2013). In case of failure of one or a few robots, the system would continue to provide sufficient performance thanks to its scalability and robustness.

In our solution, a swarm of robots forms a round shield around a user (i.e., they form a circle around the user). The round shield formed by the swarm enables a 360° protection of a user. All the robots try to stay at the same distance from each other and the human. However, when a danger is detected, the robots might not respect that rule to form a boundary on the edge of the danger zone. To achieve this, the solution relies on the pattern formation theory widely used in swarm robotics. The corresponding techniques will be explained in the next chapter with more details. If the number of robots is not high enough to form a complete circle, an arc is formed at the front to always shield the most critical zone.

As shown on Figure 3.2, the robots in contact with a dangerous zone will report the danger through visual communication with the human. Here the robots light on their orange LEDs and stay on the boundary of the zone to prevent the human from getting into it. Since the human cannot see the danger, and only the robots can, we can see that the swarm is increasing the perception capabilities of the human.

One issue that had to be resolved was the detection of the human by the robots. As Podevijn et al. (2012) suggested, the interface between the human and the robots swarm should be restricted to the strict minimum because in the field the infrastructure needed to operate the swarm might not be easy to build and manipulate. The swarm should handle the communication on its own. As a big infrastructure such as a tracking system, or any interface of the same kind would have been difficult to use in real life applications, we designed and implemented a compact, wearable device that allows a human to be recognised by the robots: a pair of shoes.

Figure 3.3 illustrates the use of the shoes (no user is wearing them to not occlude the field of view). In Figure 3.3 (left), the robots have just recognised the shoes thanks to the LED system inside and begin to move in order to form a circle around the shoe. On Figure 3.3 (right), we show an example of one configuration obtained after 3 minutes, viewed from above.

Our objective in this thesis, is to present an innovative protection using swarm robotics. The results obtained from experiments with a swarm of real robots are presented in the chapter 4.

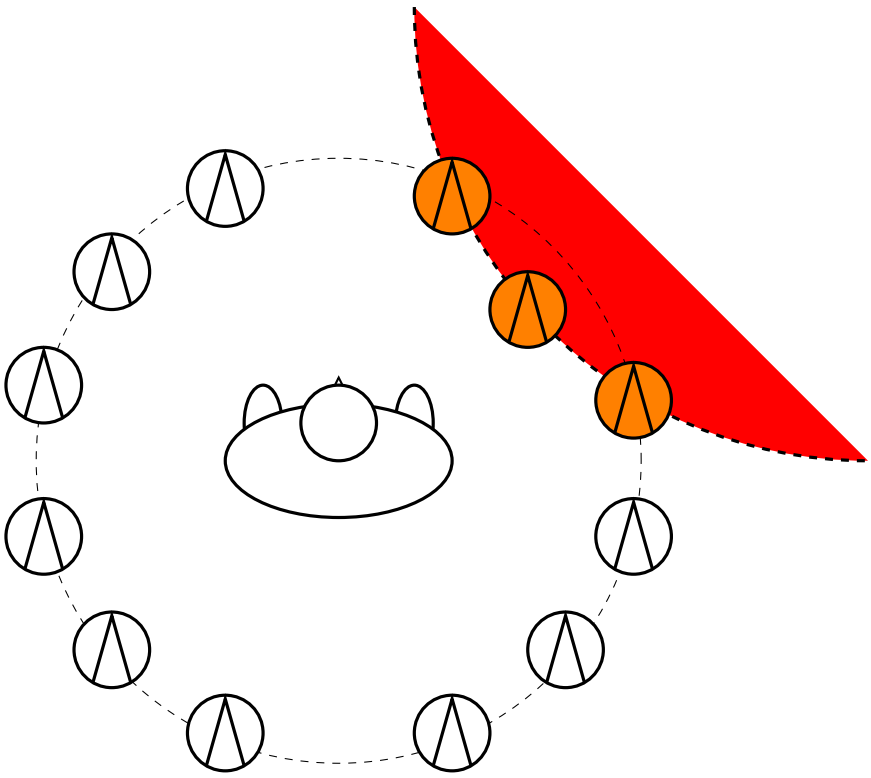


Figure 3.2 – Swarm prevention: This figure is a symbolic representation of a human helped by a swarm. The circles with a triangle inside are representations of a robot. The swarm tells the human that a dangerous zone is located at the front right by visual communication (here the robots change their colour to red). The swarm stays at the boundary to form a 'shield'. The direction taken by each individual in the swarm is given by the triangle inside (here heading north).

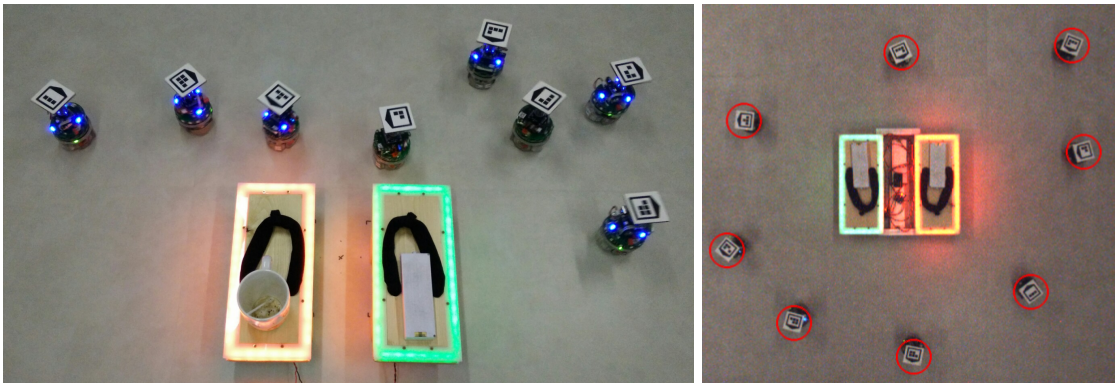


Figure 3.3 – The shoes: This picture shows a prototype of the shoes viewed from above, and the robots interacting with the shoes. The interaction is enabled through the recognition of the colours, one for each shoe, indicating left (red) or right (green) side. This pair of shoes enables the robots to locate the user, allowing them to evolve at the target distance from him/her. On the left image, the robots are still in the process of placing themselves in a correct circle. The right image depicts the situation after a 3 minutes experiment where the robots were initially placed in lines around the shoes. Objects are put on the shoes to close the lights switch (normally activated by the weight of the user).

3.3 Implementation

In this section, we present the solution to the problem described in section 3.1 and all the choices that resulted in it. The explanation will follow a top-down approach. We first review the hardware and the code architecture. Then, we will detail our implementation.

We decided on swarm robotics because, as stated in section 2.2, robustness, scalability and flexibility are characteristics that make swarm robotics systems well suited for unknown environments (Brambilla et al., 2013). In case one of the agents is broken, we do not want to see the whole system collapse and leave the human unattended. The solution guarantees that the solution will work in different conditions, environments, which is an advantage for exploration and rescue (flexibility). In case of loss of robots, scalability would maintain the protection performance to an acceptable level.

3.3.1 The Hardware

In the following section, we present the robotic platform used in our experiments, and the device allowing the robots to detect the human.

3.3.1.1 E-puck

The robotic platform chosen was the e-puck (Mondada et al., 2009) because the laboratory possess approximately 28 of them, along with the appropriate modules (omnidirectional camera, top LEDs). Furthermore, the academic personnel had developed a good knowledge of the platform. The e-puck robot platform was made for educational purposes. Its shape is cylindrical with a diameter of 7.5 cm. It is moved by two diametrically opposed wheels. Figure 3.4 (left) shows an e-puck from the laboratory. Several extensions (modules) were plugged onto it to increase its capabilities. In this thesis, in the final solution, we used the proximity sensors, the omnidirectional camera sensor and the virtual ground sensor. The proximity sensor is made up by 8 infrared sensors. Each infrared sensor returns a value proportional to the proximity of a nearby obstacle in the $[0, 1]$ interval. The infrared sensors are placed along the perimeter of the robot. The omnidirectional camera is a vertical camera placed on the top of the e-puck's base, aiming at a convex mirror to provide a 360° view of the environment. It translates this view into a list of colour blobs. A colour blob is a cluster of pixels being almost of the same colour. During the calibration, among other parameters, one can tune the degree of similarity, the minimum size of the cluster, and the recognised colours. The last sensor is different: it

is not real, and not physically present on the board (Reina et al., 2015). It is simulated through the ARGoS simulator used to develop the controller of the robot (Pinciroli et al., 2012; Garattoni et al., 2015). It sends data created inside the simulator to the robot, from the simulated environment. In our case, this data contains the colour of the ground, symbolising the presence of danger. We actually simulated red discs on the floor through the simulator to artificially set up dangerous areas that were not visible by the testing user. That way, the conditions of real life application were the closest possible to ours. An example of red zone is in Figure 3.4 (right). The positions of the robots inside the simulator are determined by the use of a tracking system (Stranieri et al., 2013). The virtual positions correspond to the real positions. Using a virtual sensor enables early experimentation by removing the need to implement a real sensor. The real robots actually have a ground sensor. However, as during the tests the human cannot see the dangerous zones, we could not use any visible colour. A new type of ground sensors, able to detect other types of information was necessary. Hence we chose to create a ground virtual sensor that detects colours that are only present in the simulator, not physically present in the laboratory. That way, the dangers remain invisible to the human doing the experiments, but visible by other operators watching the simulator screen.

Before considering the omnidirectional camera as the best option, another sensor was examined: the range and bearing sensor (Gutiérrez et al., 2009). The range and bearing extension is a infrared communication board that also provides the range (distance) and the bearing (angle) of the emission source. On paper it seemed like a better option because it was less restrictive than the omnidirectional camera. With the camera, a precise calibration of the different colours is required to avoid errors. The range and bearing uses a CRC code to detect errors in the data received, thus ensuring the correct recognition of the different entity types in the environment. The number of different entities (colours) is very limited with the omnidirectional camera as a result of the algorithm behind it. The range and bearing actuator can send up to 4 bytes of data, multiplying the amount of different entities (one string of bits for each entity). It is also less sensitive to light conditions. Unfortunately, after multiple attempts to use the range and bearing, we realised that it was unusable. It behaves in a completely unpredictable way with the range value. We tried to calibrate it, but to no avail. Figure 3.5 illustrates the wide distribution of message intensities causing an imprecise measure of the range (distance of the source). This figure is only for one run of the experiment. Consecutive runs yielded very different results for the same experiment configuration.

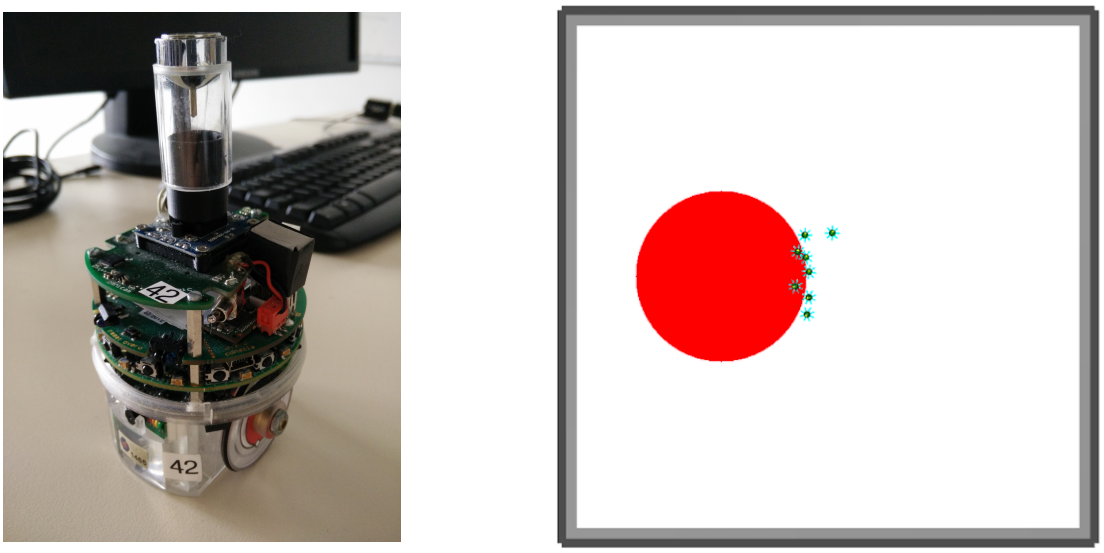


Figure 3.4 – The E-puck and its virtual sensor: On the left, one can see a picture of the robotic platform we used: the E-puck. Our swarm is composed of several robot like this one. On the right is a screen capture of the ARGoS simulator (Pinciroli et al., 2012) where the danger virtual sensor is used. Virtual means that it is not real, not physically present on the board. It is simulated through the ARGoS simulator. It sends data created inside the simulator to the robot, from the simulated environment. It augments the robot sensing capabilities. In our case, this data contains the colour of the ground, symbolising the presence of danger. The red circle represents a danger zone in which no human can go. No human can see it though. The small dots are representations of the real robots inside the simulator.

Limitations Although the omnidirectional camera worked better than the range and bearing, it was still restrictive. The difficulty of the calibration, and the error rate, increase with the amount of recognised colours. We limited ourselves to 3 colours: red, green and blue. Another limitation was the speed of the robot, limited to 10 cm/s. Normal speed for a human walking is about 5 km/h or 140 cm/s (Wikipedia, 2015b). In these circumstances we had to reduce the walking speed of the user during the demonstration and the tests. The battery was also an issue. The autonomy of one robot is between 30 minutes and one hour when moving a lot. Changing batteries and restarting the robots multiple times was necessary during long testing sessions. Since the robot only has two wheels, it maintains its equilibrium by lowering the chassis. The front or the back of the robot always touches the ground, making it impossible to use on uneven grounds. The floor of the arena, the room where all the experiments were done, is composed of adjacent squares. Sometimes the robots would get stuck between two squares. Moreover, the activity LEDs on the different circuit

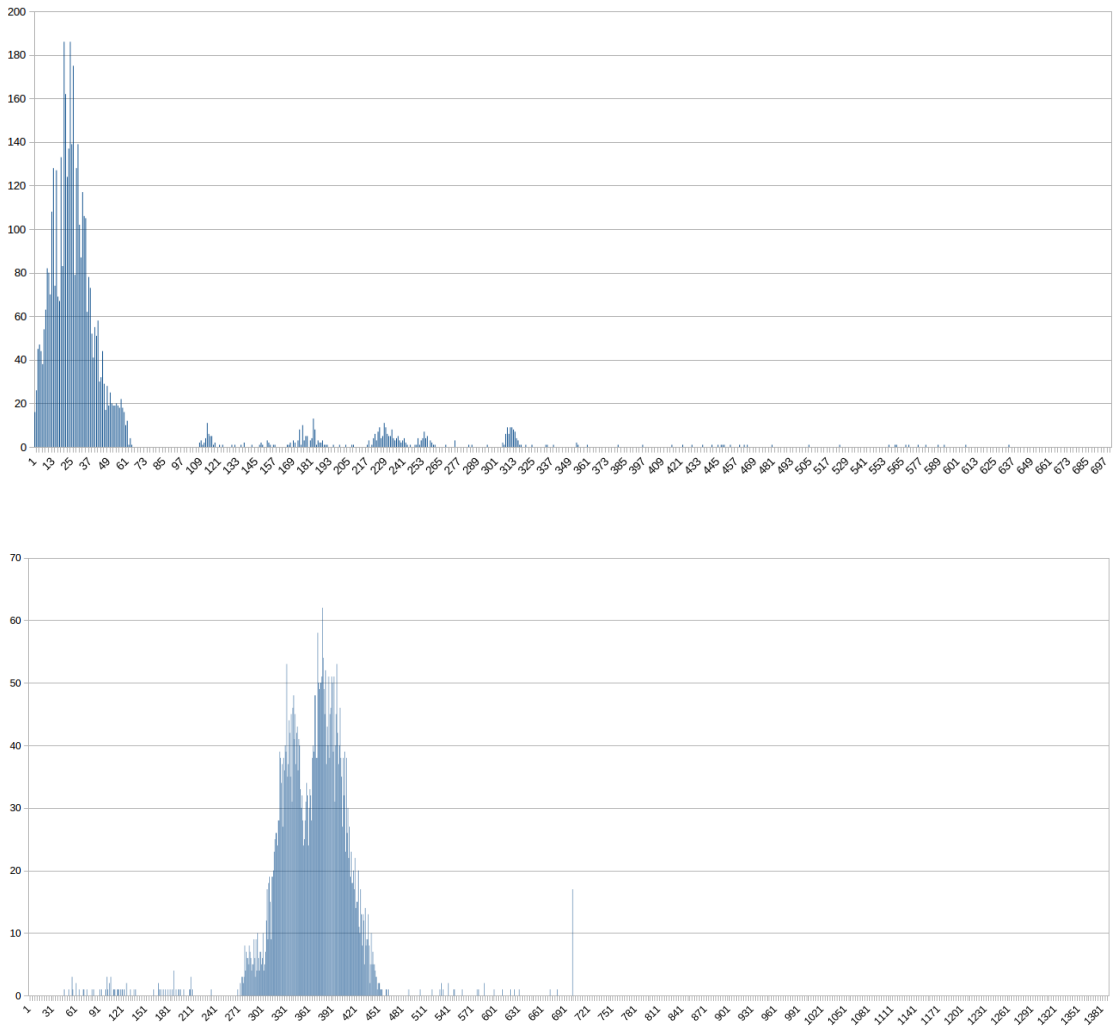


Figure 3.5 – Intensities of the RAB messages: This figure shows the absolute frequencies of the range and bearing messages intensities for a duration of 3 minutes. During these 3 minutes, one robot is emitting messages and rotating on itself. Another robot is placed at 25 centimetres of the first one and receives the messages. The message intensity can vary from 0 to 1023 on the x axis. The y axis is the absolute frequency. Each graph corresponds to a different emitting robot. On the upper graph, one can see that frequency pikes can appear far away from the 'normal' average value. On the lower graph, the interval of intensity is very big. Other runs of the experiment with the same robots yielded very different results. The current implementation of the RAB sensor is thus a bad choice for the measure of the distance between robots.

boards composing the robots interfered with the omnidirectional camera, taking them as other robots. We had to cover them with pieces of tape, and change the calibration to not take into account the blobs that were too small.

3.3.1.2 E-geta

As explained in section 3.2, one of the main issue was to enhance the robots so they could detect the user and position themselves with respect to him without any large external equipment. Large external equipments are not recommended since, in the targeted unknown environment, they might not be usable. For example, one might use a tracking system to get the position of the robots in real time and communicate it to the robots for them to adjust their speed. In a controlled environment, this may work very well, but in the field it would be difficult to deploy such a tracking system.

We thus opted for a compact, wearable device that would act as a 'landmark' for the robots: a pair of shoes. In order for the robots to understand on which side of the human they are (to go in front of him/her), the two shoes have to emit a different message. The two shoes have to emit a different colour to give the robot information on the direction of the human. In section 3.3.2 we come back on the algorithm used to deduce the direction of the human from the observed colours.

Figure 3.6 – E-Geta: Left image by Haragayato [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons. Found on Wikipedia (2015a). On the right is a picture of our prototype shoes, connected to their battery through voltage regulators. See Figure 3.7 for the circuit.

We took inspiration from the Japanese 'geta' shoes for the design. Figure 3.6 shows a picture of a Japanese wooden shoe called geta. The right side of the

figure is a picture of our prototype. The two have the same overall design. We chose this design in order to slow down the human's speed. Indeed, the speed of the robots is limited to maximum 10 cm/s per wheel. Another advantage is the simplicity of the structure, and the low number of assembly parts.

The base of the shoe is made with wood. The surrounding piece covering the LEDs was cut in sheets of semi-opaque plexiglass to diffuse the light. The LEDs are standard strip LEDs (one red strip, and one green). The electronic circuit (Figure 3.7) is made up by a 9 volt battery connected to two variable voltage regulators. The two regulators are step up in parallel in order to increase the potential on the output. Each regulator is connected to a shoe LED strip. A separation is necessary since the green LED strip requires more energy than the red one. To get an equivalent luminosity for both shoes, a different voltage had to be applied.

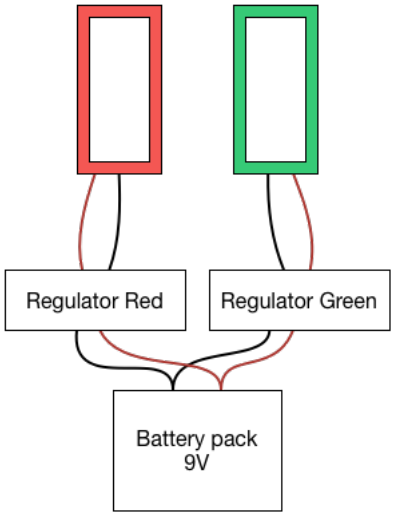


Figure 3.7 – The circuit: The figure shows a sketch of the final electronic circuit for the shoes. The battery delivers 9 volts to two regulators in parallel, one for each shoe since each one of them need a different energy supply. Indeed, the green LED strip is more power hungry than the red one. The mass (black cable) is common for all the circuit.

3.3.2 The Robot Behaviour

The first step of the design of the solution is to imagine how the system will look like and how we will implement it (how do the robots move around the human, what shape will they try to form). This part is important because it will define the overall look and performance of the system.

We decided on a circle shape because it is the easiest to realise in practice in the pattern formation theory. It offers the best ratio

$$\frac{Surface}{Perimeter} = \frac{\pi r^2}{2\pi r} = \frac{r}{2},$$

where r is the radius of the circle. That means that fewer robots are needed for the same protected area, and more space for the human with a certain amount of robots than any other possible shape. The Figure 3.8 represents the kind of circle that we would like to obtain for 6 robots and 1 human in the centre.

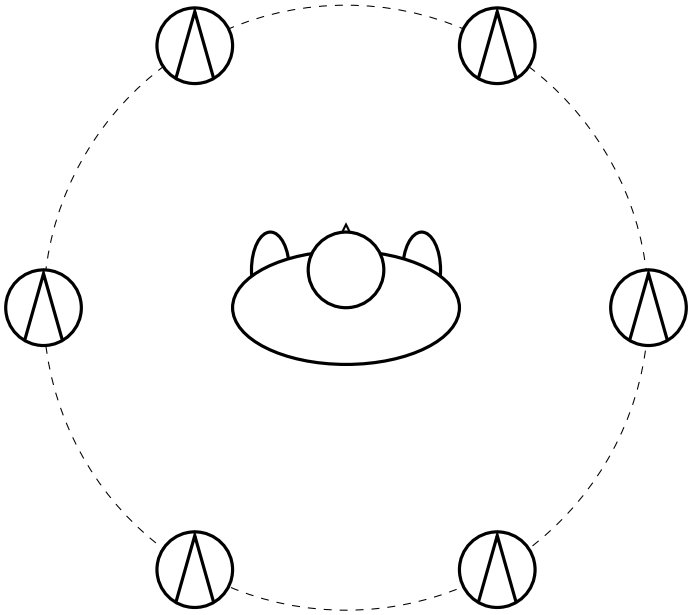


Figure 3.8 – Ideal behaviour in absence of danger: This figure symbolises the ideal behaviour required in absence of danger. The swarm forms a circle to cover the widest protected surface for a given amount of robots. All the robots are equally distanced from each other and the human. The human is protected in all directions. In presence of danger, the robots in contact with it should report it to the human and prevent him from going towards it, as seen on Figure 3.2. In that case, the conditions concerning the target distance from the human and between robot may not be respected.

Implementing a robots swarm behaviour means writing a controller code for its individual components: the robots. The laboratory provides a template for this purpose. The logic of the individual behaviour is added inside callback methods called either by the simulator when performing simulations inside ARGoS (Pinciroli et al., 2012), or by the robot main method when testing on

real robots. The final code was written in C++. The code can be compiled for the ARGoS simulator and cross-compiled for the real robots (E-puck) without any modification. After the compilation, a simple transfer of the binary codes over WiFi allows the operator to store the controller on the robots to launch an experiment.

The implementation of the controller is built on 2 layers. The upper layer is a deterministic finite state machine, containing for each state a specific behaviour in the lower layer. Figure 3.9 illustrates the whole structure of the upper layer in a simplified fashion. A complete state machine gathering all the states can be found in Figure 3.10. Although the complete state machine is containing all the aspects of the behaviour, it does not allow to grasp the idea quickly. Dividing the final behaviour into several connected sub-behaviours modularises the solution. Adding a new state, a new sub-behaviour is easy.

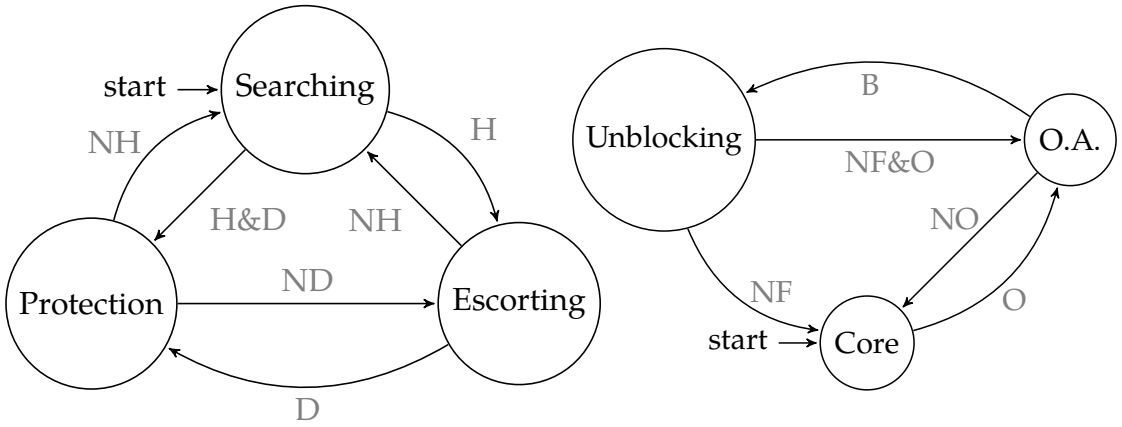
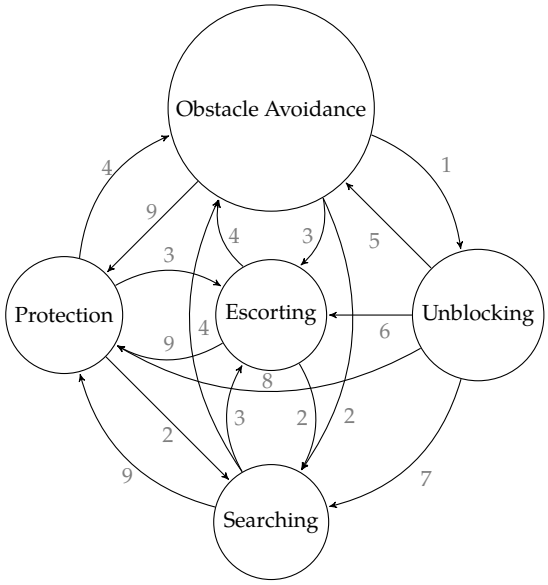


Figure 3.9 – State machine of the final behaviour: This figure is the visual representation of the state machine of the final behaviour. Figure 3.9 (left) is what could be called the 'core' of the behaviour, while Figure 3.9 (right) would be considered as additions to enhance the behaviour. The core part of the state machine is present on Figure 3.9 (right) since it is one of its constituents. It is composed of 3 states: *Searching* when the robots do not detect any human nor obstacle, *Escorting* when a human is detected, and *Protection* when a human is detected and there is a danger nearby. If an obstacle is detected by a robot, the controller changes its state to the *Obstacle Avoidance (O.A.)* state. If the robot gets blocked, another state takes over to unblock it: *Unblocking*. When no more obstacle is in front of the robot, it can go back to its obstacle avoidance if any obstacle remains close. If none, it switches back to its core actions. The labels next to each transition must be read as follows: H(uman), D(anger), O(bstacle), F(ront obstacle), B(locked. The robot detects that it is blocked. See page 28 for details.). N stands for the negation, so NH means 'no human found'.

Once the actions the robots have to execute have been defined, we have to



1. Amount of direction change (left/right) while having an obstacle around reaches a threshold.
2. No human & no obstacle.
3. No obstacle & human & no danger.
4. Obstacle.
5. No front obstacle & obstacle elsewhere.
6. No front obstacle & human & no danger.
7. No front obstacle & no human & no obstacle.
8. No front obstacle & human & danger.
9. No obstacle & human & danger.

Figure 3.10 – Complete state machine of the final behaviour: This figure is the visual representation of the complete state machine of the final behaviour. On Figure 3.10 (left), the states and their connections are drawn. On Figure 3.10 (right), the information on the conditions needed to take the transitions are listed.

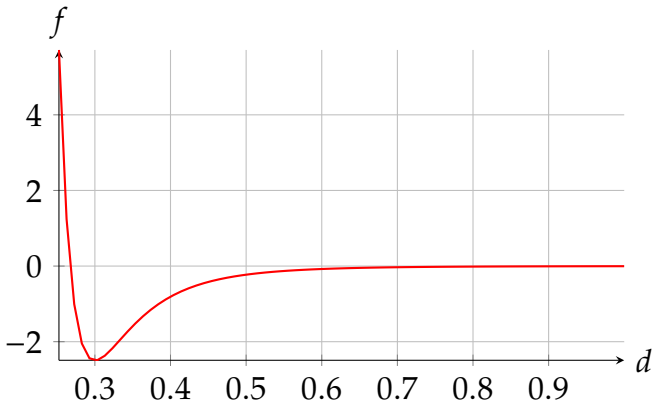
implement them, code them in the controller that will be run. So the next step was to find a way to translate those actions into code. Our behaviour is a kind of coordinated motion and pattern formation. Thus the common way of implementing it was to make use of the virtual physics design. Using this framework, each robot is a particle subject to virtual forces exerted by the environment (the other robots, the obstacles, and other elements). Khatib (1986) was among the first to use this method. His goal was to implement an obstacle avoidance swarm behaviour where the obstacles create repulsive forces and the goals attractive forces. The overall resulting potential presented local minima at goals and maxima at obstacles. Reif and Wang (1999) introduces 'social potential fields' consisting in virtual forces applied on robots by other robots, obstacles, objectives, or other elements. The robot resultant motion is defined by the sum of all the forces applied to it. The individual robots carry out the calculations themselves, so the final control is completely distributed. The laws they used are similar to those found in molecular dynamics (inverse-power laws). For example, a law could favour attraction over long distances and repulsion over

short distances. One of these is the Lennard-Jones potential, depicted in Figure 3.11. Inverse-power laws, while being extremely simple, can form interesting and elaborate patterns with molecules and plasma gases (Reif and Wang, 1999). Spears et al. (2004) proposed a framework they call 'physicomimetics' to grant distributed control over a large swarm of robots with 'artificial physics'.

The laws of physics force a system to go to a state of minimum energy, i.e., to reach a minimum of the potential function of the system. Since the force exerted on the system is proportional to the derivative of the corresponding potential –

$$\vec{f} = -\vec{\nabla}P,$$

with $\vec{\nabla}$ being the nabla operator for the gradient computation, P the potential and $\vec{f}$ the force – the minimum of the potential function means the disappearance of the forces. For every robot to be at the desired location, the forces need to disappear. In this thesis, we only consider forces and not the virtual potentials associated to them. The implemented behaviour is expressed in terms of virtual forces.



$$f(d) = \epsilon\left[\left(\frac{\sigma}{d}\right)^{12} - 2\left(\frac{\sigma}{d}\right)^{6}\right]$$

$$\epsilon = 2.5$$

$$\sigma = 0.3$$

Figure 3.11 – The Lennard-Jones potential: This figure shows a graph of one of the most used virtual potentials in virtual physics, the Lennard-Jones potential, where ϵ is the gain, σ is the target distance and d is the current real distance. In this example, the equilibrium is reached at the global minimum at 0.3.

Using virtual physics offers some advantages over the other methods (Brambilla et al., 2013):

1. Only one mathematical formula fluently and elegantly converts all the inputs into outputs for the actuators. It removes the need for multiple

rules and behaviours.

$$f : \mathbb{R}^m \rightarrow \mathbb{R}^n : x_1, x_2, ..., x_m \rightarrow y_1, y_2, ..., y_n = f(x_1, x_2, ..., x_m),$$

where m is the number of inputs, n is the number of outputs and f is the translating function.

2. Multiple behaviours can be combined by simply summing the corresponding resulting vectors.

$$y_1, y_2, ..., y_n = g(x_1, x_2, ..., x_m) = \sum_{i=0}^{s} f_i(x_1, x_2, ..., x_m),$$

where s is the number of behaviour components.

One disadvantage is that it might be difficult to find an expression that implements perfectly the behaviour we want.

As written above, the robots need inputs to compute the values to send to the actuators, i.e., the wheels. The two types of inputs are the *colour blobs* and the *proximity sensor values*, respectively provided by the omnidirectional camera and the proximity sensors. Both are explained in section 3.3.1.1. Three blob colours are used for our solution: red, green and blue, the three basic components of every colour in computer graphics. It was decided to a low number of colours to ease the calibration process and lower the amount of errors when detecting the blobs (wrong colour). To each blob is associated a distance - angle couple. The angle is taken from the front of the robot in radians. With these two values, the robot is able to situate all the blobs and use them as attractive or repulsive points. The proximity values are a list of 8 angle - value couples, where the angle is the position of the sensor on the perimeter of the robot. The whole perimeter of the robot is covered to detect any nearby obstacle. The value is comprised between 0 and 1, inversely proportional to the distance to the obstacle.

The following paragraphs explain in details the various forces that were implemented to obtain the desired behaviour from the given inputs. They are grouped by states of the state machine in which they are used (see fig. 3.9). As explained on page 22, each state in the upper layer of the general behaviour (the state machine) contains a sub-behaviour (a particular action) executed by means of virtual physics.

Searching When no human nor obstacle is around, the robot enters in the *Searching* mode with its LED off. It tries to stay at a constant distance from a detected colour blob, whatever colour it is. Since robots in *Escorting* or *Protecting* mode light their LEDs in blue, the searching robots always stay around the robots helping the human, whom they will detect at some point. This measure prevents the robots from going away too far from the human. This sub-behaviour is implemented through a sum of simplified Lennard-Jones virtual forces, one for each colour blob detected. The term 'simplified' is used because the force is not the real derivative of the Lennard-Jones potential, but a simplified version with lower exponents on the fractions:

$$\vec{f}(d) = \frac{-4\epsilon}{d}\left[\left(\frac{\sigma}{d}\right)^4 - \left(\frac{\sigma}{d}\right)^2\right]\vec{1_s} \tag{3.1}$$

The original version is:

$$\vec{f}(d) = \frac{-12\epsilon}{d}\left[\left(\frac{\sigma}{d}\right)^{12} - \left(\frac{\sigma}{d}\right)^6\right]\vec{1_s} \tag{3.2}$$

The simplified version is exposed in Figure 3.12. If no blob is detected, the robot goes forward with a speed of 5 cm/s. The unit vector $\vec{1_s}$ is heading towards the source of the force (the applier).

Escorting If a human is found nearby and no danger is around, the controller enters into the *Escorting* state. The robot then tries to stay at a fixed distance from the human and other robots. If all the robots around the human follow the same pattern formation rules, a circle appears encircling him/her. The complete virtual force for this state is the sum of 3 components: the *human force*, the *robot repulsion force* and the *gravity force*. All three components are fed with the detected colour blobs as inputs to evaluate the distances and angles.

- The *human force* uses a stronger version of the Lennard-Jones force to keep the robot at a certain distance from the human (see Figure 3.13). Its expression is given by equation 3.3. Only the closest human colour blob is fed to the force computation.

$$\vec{f}(d) = \begin{cases} \frac{-4\epsilon}{d}\left[\left(\frac{\sigma}{d}\right)^4 - \left(\frac{\sigma}{d}\right)^2\right]\vec{1_s} & \text{for} \quad d < \sigma \\ 15(d-\sigma)\,\vec{1_s} & \text{for} \quad d \geq \sigma \end{cases} \tag{3.3}$$

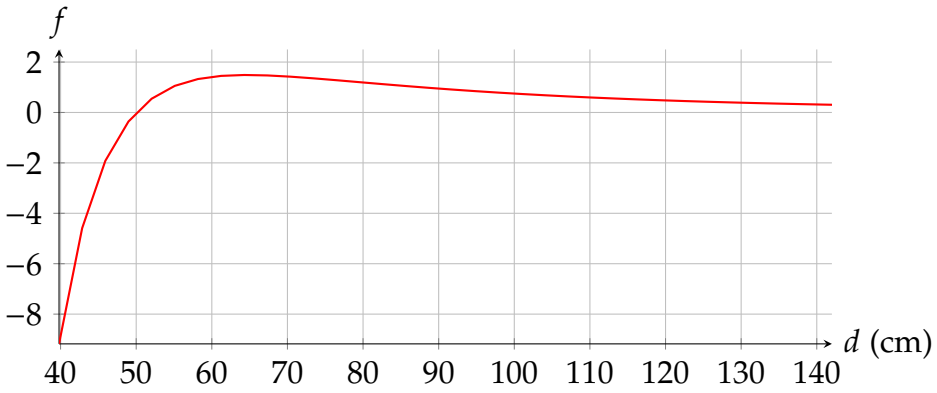


Figure 3.12 – The simplified Lennard-Jones virtual force: This figure exposes a simplified version of the Lennard-Jones force derivated from the corresponding potential. Its expression is given by equation 3.1. The parameters values are: $\epsilon = 100$ and $\sigma = 50$, and are conform to those used with robots. One can observe the root at 50 corresponding to the state of minimum energy in the system. Above 50, the force is positive, so the robot is attracted by the source applying the force. Below 50, it is negative, so the robot is repulsed from the source of the force.

- The *robot repulsion force* maintains a fixed distance between the robots escorting or protecting a human (those with LEDs lit in blue). The simplified Lennard-Jones force is used, given by Equation 3.1 and Figure 3.12.
- The *gravity force* pushes the robots in front of the human like if the floor was 'sloped down' to the front of the human, hence the name. Figure 3.14 illustrates the idea of the gravity force and Figure 3.15 its norm. This force is expressed in polar coordinates:

$$\vec{f}(\alpha) = \begin{cases} \epsilon \, \vec{1_\theta} & \text{for} \quad \alpha < 0 \\ -\epsilon \, \vec{1_\theta} & \text{for} \quad \alpha \geq 0 \end{cases} \tag{3.4}$$

  $\vec{1_\theta}$ is the standard axis in polar coordinates $(\vec{1_r}, \vec{1_\theta})$ where the origin is the centre of the human. ϵ is the norm of the force. This force can be deactivated in the configuration file. Sometimes it might not be needed, like in the experiments without human in chapter 4, or when the robots do not need to be pushed in front of the human.

Protection The *Protection* is reached when the robot detects a human and a danger. It makes its blue LEDs blink and stays at the border of the encountered danger area while maintaining the escorting formation. Hence this

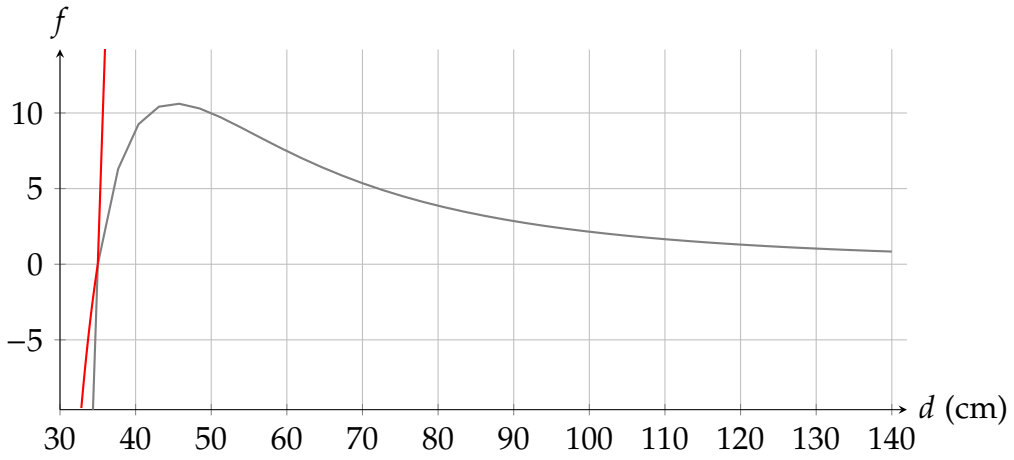


Figure 3.13 – The stronger Lennard-Jones virtual force: This figure exposes a stronger version of the Lennard-Jones force derivated from the corresponding potential. Its expression is given by the equation system 3.3. The functions are plotted on the whole domain but only the red part is used. It allows to compare the two values for the same distance. The force is stronger in the attraction part, hence the name. The repulsion part is unchanged because of the more interesting asymptotic behaviour at $d = 0$, increasing the norm of the force quicker than the linear attraction replacement (in gray). The parameters values are: $\epsilon = 500$ and $\sigma = 35$, and are conform to those used with robots. One can observe the root at 35 corresponding to the state of minimum energy in the system. Above 35, the force is positive, so the robot is attracted by the source applying the force. Below 35, it is negative, so the robot is repulsed from the source of the force.

sub-behaviour is based on the one from the *Escorting* state. The only difference is in the *human virtual force*. Since everything stays the same as in *Escorting*, except the fact that the robot must not cross the danger border, we just modify the target distance from the human. In presence of a danger, σ (the target distance) will be decreased incrementally until no danger is detected any more. As a result, the robot gets closer to the human like seen on Figure 3.2 at page 13. We speak about *dynamic target distance*. Figure 3.16 illustrates the concept.

Obstacle Avoidance The *Obstacle Avoidance* is activated when the robot encounters an object. Since the robot cannot go back, only the front sensors are used. Figure 3.17 shows the four sensors that are taken into account and explains how the state is activated.

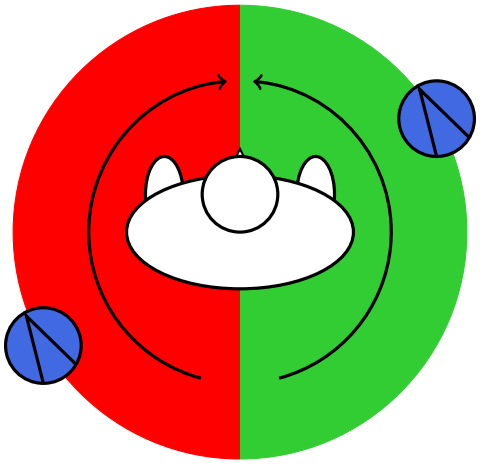


Figure 3.14 – The gravity virtual force concept: This figure illustrates the idea of the gravity virtual force. This force that can be disabled in the configuration of the experiment if there is no need to push the robot in front of the human. The closest human colour blob to the human is taken as reference (the closest shoe). Then depending on colour of the blob (shoe), two different things can happen: the blob is red and the robot turns clockwise around it, or the blob is green and the it turns anti-clockwise. The goal is to go in front of the human like if the floor was 'sloped down' to the front of the human, hence the name. Two robots are both on the opposite side of the human. The left one sees the red shoe as the closest one and turns clockwise heading for the front. The right robot does the opposite with the same intention.

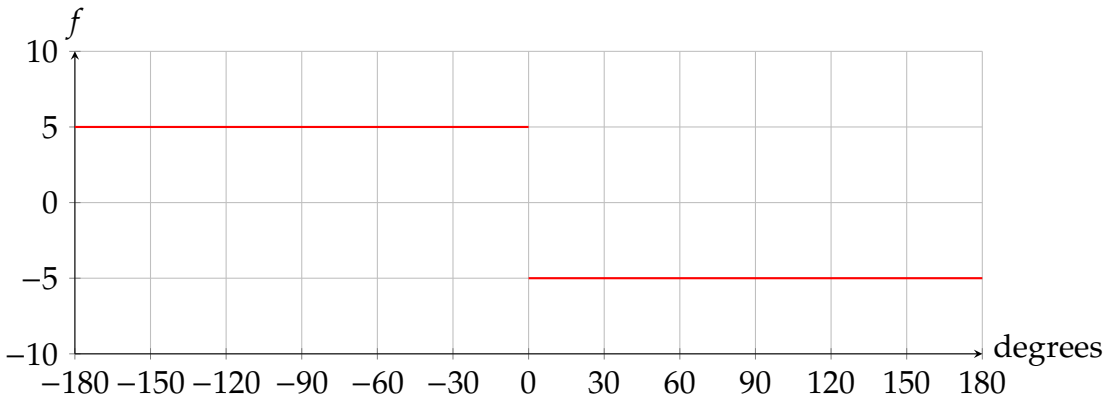


Figure 3.15 – The gravity virtual force: This depicts the adopted value for the gravity virtual force with respect to the angle from the front of the human. Any angle outside this domain can fall back in the $[-180; 180]$ interval by normalising it. An angle of 0° means that the robot is in front of the human. The angle increases by turning anti-clockwise.

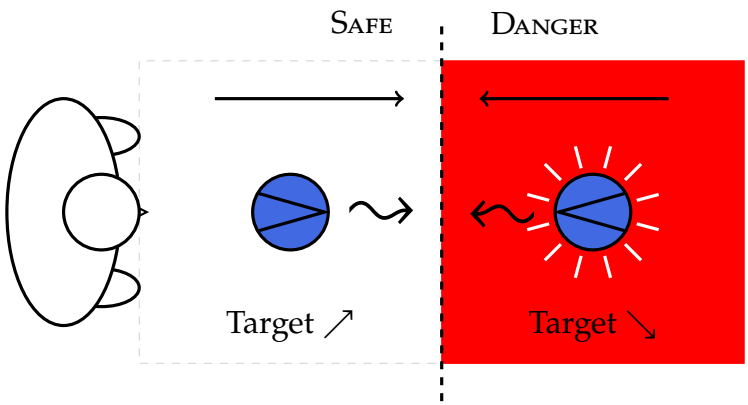


Figure 3.16 – The dynamic target distance: The concept of dynamic target distance is explained in this drawing. On the figure, the same robot is represented at two different time steps. On the right, the robot is inside the red dangerous area after the human took a step towards it. The robot enters *protection mode* and starts blinking. Its human target distance begins to decrease incrementally at each time step by a user-defined amount. As a consequence, it comes closer to the human. When it crosses the border of the danger zone, it goes back to *Escorting* mode (the danger is not detected any more) and raises its target distance again. At some point, it will re-enter in the area and the whole process will start over. To avoid human confusion regarding the presence of danger, there is a delay before the robot stops blinking. That way, when the robot oscillates around the border, it keeps blinking to report the close danger.

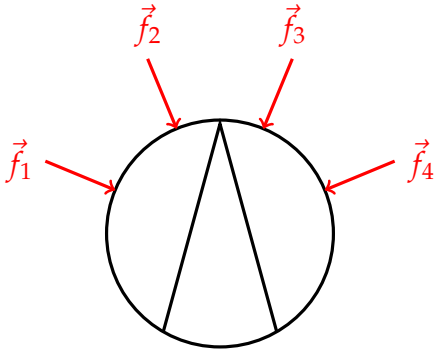


Figure 3.17 – The obstacle avoidance concept: This figure shows the four sensors that are taken into account to detect a nearby object. Each sensor is seen as a force pushing the robot whose intensity is inversely proportional to the distance to the sensed object. If the norm of the strongest force is over a threshold, the robot goes into *Obstacle Avoidance* and uses the resultant as direction vector.

Unblocking If the robot stays in *Obstacle Avoidance* and changes its direction for some time, it enters into *Unblocking* mode. Figure 3.18 shows a situation likely to provoke it. The robot enters a narrow path between obstacles. Let us say that it is closer to the left object and goes towards it. Since the forces generated by the proximity sensors on the left are stronger, the resultant is heading to the right, bringing the robot to the other obstacle. The same event occurs on the right side, and the robot returns to the left obstacle. If the robot keeps doing for a certain amount of time, the *Unblocking* mode is activated. At that point, two different things can occur: an obstacle lies ahead of the robot within a certain distance, or nothing is in front of the robot.

- If there is nothing, the robot moves forward for one time step, and returns to the appropriate state: *Obstacle Avoidance* if there is an obstacle around, or a state of the *core* if none.
- If there is an obstacle in front, the robot turns on itself until there is none. It is the case in Figure 3.18.

All the parameters presented until now can be tuned by the human to ensure the best performance. Here is a list of the important parameters available inside the configuration file:

Human Force Gain: The factor ϵ multiplying the force in expression 3.3. Increasing it strengthens the force the human exerts on the robots.

Human Force Distance: The target distance to the human for all the robot escorting him/her.

Human Force Distance Variation Delta: The value added or subtracted to the target distance to run the dynamic target distance mechanism.

Agent Force Gain: The factor ϵ multiplying the force in expression 3.1. Increasing it strengthens the force robots exert on each other.

Agent Force Distance: The target distance all the robots must keep between each other.

Gravity Force: A boolean activating the *gravity force* that pushes robots in front of the human.

Gravity Force Gain: The norm of the force that pushes the robots in front of the human.

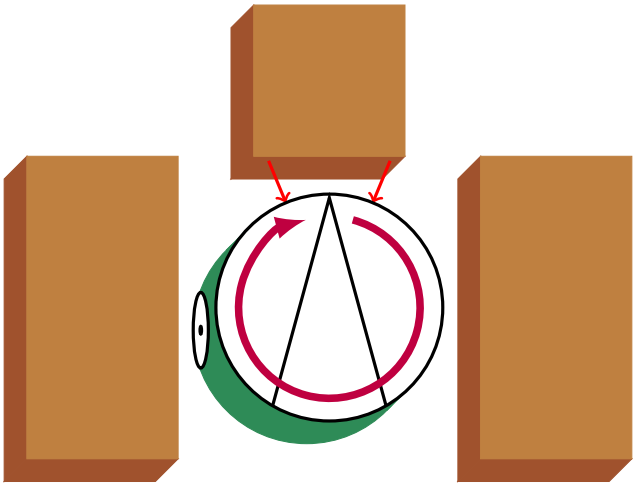


Figure 3.18 – The unblocking concept: On this figure, the robot is trapped between two obstacles, one on the left and the other on the side. Let us say that it is closer to the left object and goes towards it. Since the forces generated by the proximity sensors on the left are stronger, the resultant is heading to the right, bringing the robot to the other obstacle. The same event occurs on the right side, and the robot returns to the left obstacle. If the robot keeps doing for a certain amount of time, the *Unblocking* mode is activated. At that point, two different things can occur: an obstacle lies ahead of the robot within a certain distance, or nothing is in front of the robot. If there is nothing, the robot moves forward for one time step, and returns to the appropriate state: *Obstacle Avoidance* if there is an obstacle around, or a state of the *core* if none. If there is an obstacle in front, the robot turns on itself until there is none. Here, it will turn until heading south of the image, and then get out.

Direction Vector Window Size: The direction vector sent to the wheels is computed by averaging a number of direction vectors: the one returned from the forces exerted on the robot at this time step, and a number of previous vectors sent to the wheels. The parameter gives the amount of vectors used for the average. Increasing it makes the motion smoother.

Once the direction vector has been computed by the robot, it still has to translate it into wheel speeds. As mentioned in section 3.3.1.1, it has two diametrically opposed wheels. Figure 3.19 exposes in pseudo code the algorithm behind the conversion.

```
/* Get the direction values:                                    */
angle := angle of the direction vector with respect to the front of the robot;
speed := norm of the direction vector;
/* Check if speed is not too high for the robot:                */
if speed > 10 then
    speed = 10;
end
/* Multiply angle by 3 to accelerate the robot rotations:       */
angle = 3 · angle;
/* Check if angle is still correct:                             */
if angle > π then
    angle = π;
end
if angle < −π then
    angle = −π;
end
/* Assign wheels speed:                                         */
if 0 < angle < π then
    leftWheelSpeed = speed · cos(angle);
    rightWheelSpeed = speed;
else
    leftWheelSpeed = speed;
    rightWheelSpeed = speed · cos(angle);
end
```

Figure 3.19 – The direction vector to wheel speeds translation: This pseudo code explains how the computed direction vector is translated into the robot wheel speeds. The speed limit for the e-pucks is about 10 cm/s. Thus, the first action after getting the direction vector is to limit its norm. The angle of the vector is then multiplied by a factor to accelerate the rotation of the robots. The idea behind the rotation algorithm is to reduce the speed of the wheel corresponding to the direction of the rotation. E.g.: if the robot needs to turn left, the left wheel speed decreases by a factor proportional to the angle. The function that best fits is the cosinus of the angle, decreasing from 1 at 0° to -1 when the angle is 180°, making the robot progressively increase the curvature of the path from straight line to rotation on itself.

Chapter 4

Experiments

In this chapter, we introduce the different experiments we conducted with the real robots. These experiments were conducted with the E-puck robotic platform (Mondada et al., 2009). We can classify the experiments into two sets: one assessing the system quantitatively, and one assessing the system qualitatively (i.e., the demonstration). We first describe the quantitative assessments, and their results in the section 4.1. Then we switch to the other kind of experiments in the section 4.2.

4.1 Characterisation of the System

In this section, we assess the performance of our solution without any human controlling the swarm. The solution is evaluated quantitatively and qualitatively with 5 criteria. In this section, we introduce these criteria.

4.1.1 The Robot Speed

The robots have a speed limit. Their wheel speed cannot be higher than 10 cm/s (0.36 km/h). This limit has been fixed as one of the best practices by the laboratory staff to ensure that the robots do not get damaged in an experiment. The average walk speed of a human is 5 km/h (Wikipedia, 2015b). The experiments that will follow can thus be seen as proof of concept. Faster robots would be required for real application. Furthermore, the robots should be able to move around in multiple kinds of environments. The E-puck (Mondada et al., 2009) is only suited for flat surfaces.

4.1.2 The Shoe Detection Range

In this section, we determine the range of detection of the shoes by the robots. We determine the maximum distance at which the robots see the shoes. Figure 4.1 illustrates the experiment we conducted to determine this distance. The maximal distance goes from 59 cm in front and behind the shoes, to 65 cm on the right and 70 cm on the left. This distance is acceptable given the size of the robots. It gives enough room to the user. There is no risk of stepping on a robot. The probable cause of the lower distance for the green shoe is the lower intensity of the green LEDs inside the shoe. We use a lower luminosity to increase the recognition rate of the green shoe by the robots. A brighter green shoe leads to a more difficult calibration and worse blob detection rate on the robots.

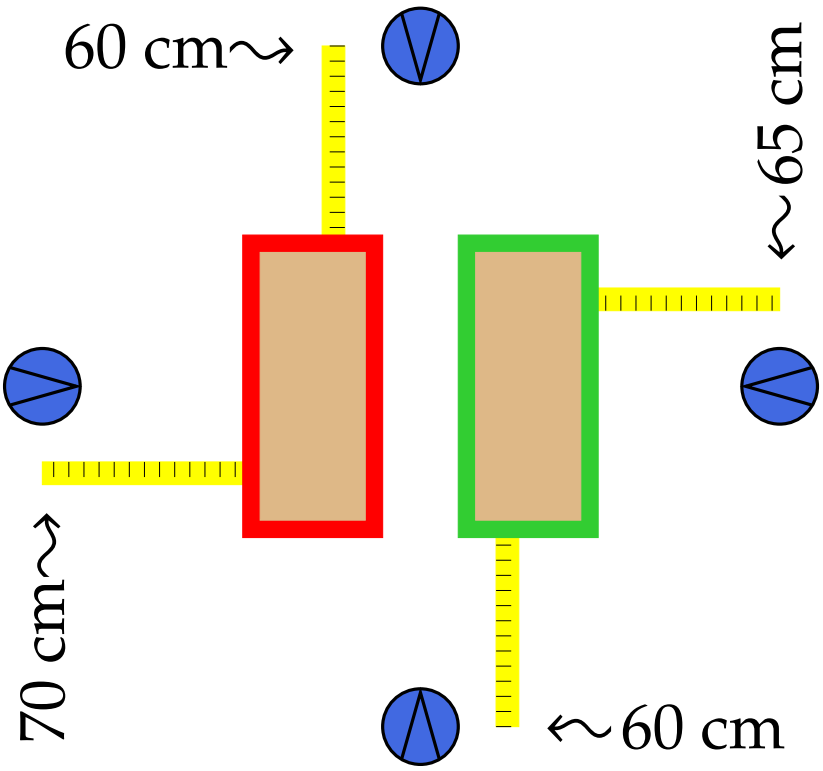


Figure 4.1 – The shoe detection range: This figure depicts the experiment we conducted to evaluate the maximum distance from the shoes at which the robots detect the shoes. A robot was placed on the left, right, front and behind the shoes. We manually incrementally moved the robot closer to the shoes until the robot detected them. We measured the corresponding distance to the shoes. The red shoe is the left shoe, and the green one is the right one.

4.1.3 Circle Properties

In this section, we evaluate the properties of the circle formed by the swarm of robots. We first expose the three metrics we used to perform the evaluations. Then we detail the conditions of the experiments. Finally, we analyse the observations made with the experiments.

4.1.3.1 Metrics

In order to evaluate the performance of our solution, we needed to create metrics to put numbers on the behaviour observed (how we compute the performance of our solution). We defined three metrics which we think correspond the most to what we want to capture from the observations: *Correct Distance*, *Robot Density* and *Time*.

1. *Correct Distance*. This metric measures to what extent the robotic swarm respects the target distance to the human.
2. *Robot Density*. It measures how regularly spaced the robots are around the human. The more regularly spaced they are, the best it is. It checks if the human is protected from all directions.
3. *Time*. It measures how long it takes for the swarm to reach a configuration such that the *Correct Distance* and *Robot Density* metrics go under a given threshold.

Correct Distance This metric measures to what extent the robotic swarm respects the target distance to the human. To measure the error related to the distance human-robot, we consider the distance from every robot to the rectangle formed by the two shoes (see Figure 4.2). Once we have these distances, we use the next formula to compute an error for every time step:

$$\text{error}_t = \frac{1}{N} \sum_{i=1}^{N} \frac{|d_{ti} - \overline{d}|}{\overline{d}}, \tag{4.1}$$

where t is the number of the time step, N is the number of robots, d_{ti} is the distance human-robot for robot i at time step t, and $\overline{d}$ is the target distance. Since the detected colour blob on the shoe might not always be the nearest point to the robot in the rectangle, this error measure is not perfect. Indeed, the robot adjusts its distance based on the closest human colour blob, not on the closest point of

the shoe (perimeter of the rectangle). Although this error measure is not perfect, it is still reasonable enough for our purpose. To measure the error between the real distance and the target distance, the robots use the colour blobs on the shoes. One blob may cover a big part of one shoe. This is caused by the diffusion of the light in the plexiglass material we used for the shoes. So its center might not be the closest point to the robot on the shoe itself. On the other side, the measure of the error we make with the arena tracking system takes as reference the closest point to the robot on the shoe itself. There is thus a difference between the way the robot computes its error and adjusts its position with respect to the human, and the way we compute the error afterwards. It would be difficult to use the same methodology to compute the error, as the arena tracking system cannot see the colour blobs. We would have to compute the error on each robot separately, and then gather all the data to get the final value. In that case, however, the measure would not come from an outside device. It would come from the robots themselves. This would be less interesting because if there is an error in the distance measure in the robot, the error measure would be good but in reality the robot would be misplaced.

Robot Density It measures how regularly spaced the robots are around the human. The more regularly space they are, the more the human is protected from all directions. The error on the angular density of robots is computed as follows. The positions of all the robots are captured, and the angle between each consecutive pair of robots is calculated (see Figure 4.3). This list of angles is fed to the next formula to return the error for the current timestep:

$$\text{error}_t = \frac{1}{N} \sum_{i=1}^{N} \frac{|\alpha_{ti} - \overline{\alpha}|}{\overline{\alpha}}, \tag{4.2}$$

where t is the number of the time step, N is the number of robots, α_{ti} is the angle separating the pair of consecutive robots i at time step t, and $\overline{\alpha}$ is the target angle.

Time It measures how long it takes for the swarm to reach a configuration such that the *Correct Distance* and *Robot Density* metrics go under a given threshold. To get this value, we average the evolutions of the error over time for every type of experiment. We compare this evolution to a defined threshold representing a certain quality of the solution. We obtain an average time for every type of experiment corresponding to the time step where the threshold is crossed.

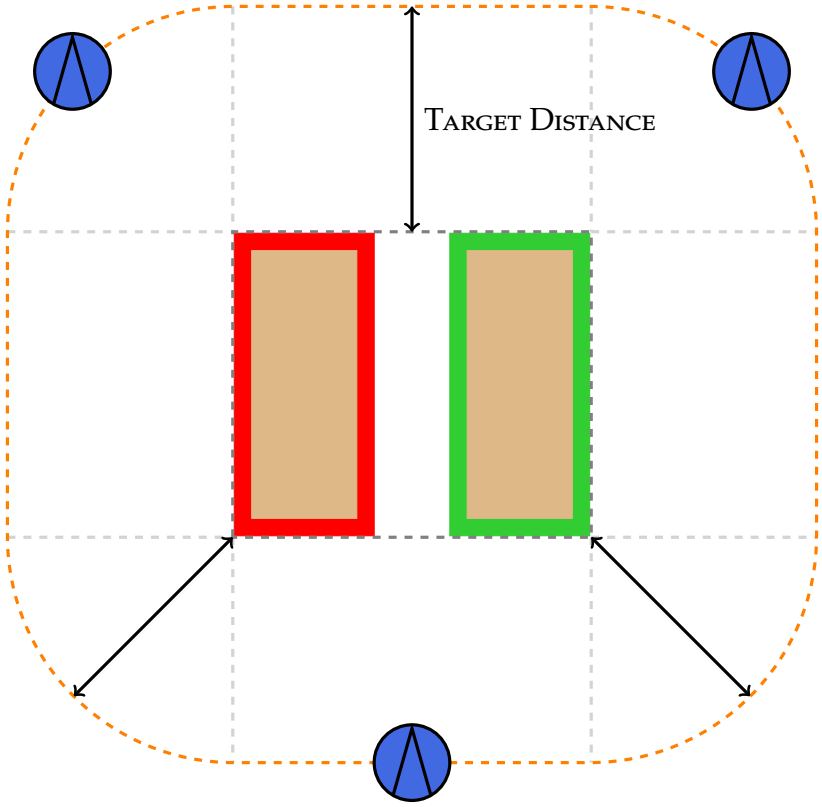


Figure 4.2 – The distance metric: This figure represents the algorithm used to compute the distance metric. The red and green shoes in the middle form a rectangle. The position of all robots is obtained thanks to the arena tracking system (see Appendix **??**). The distance of all robots to the rectangle is computed. We compare every distance to the target human distance (we take absolute value of the difference). Then we divide by the human target distance to normalise. We normalise since an error of 10 cm for a target distance of 30 cm is not the same as an error of 10 cm for a target distance of 100 cm. The average of all those divisions is the error of the current time step. The example depicted on the figure would return an error of 0. See expression 4.1. One can see that the geometric shape formed by the points that are at the target distance from the human is not a perfect circle. This is because the robots position themselves with respect to the shoes, not the center of the human. We try to provide a measure that corresponds to the way robots compute their own distance.

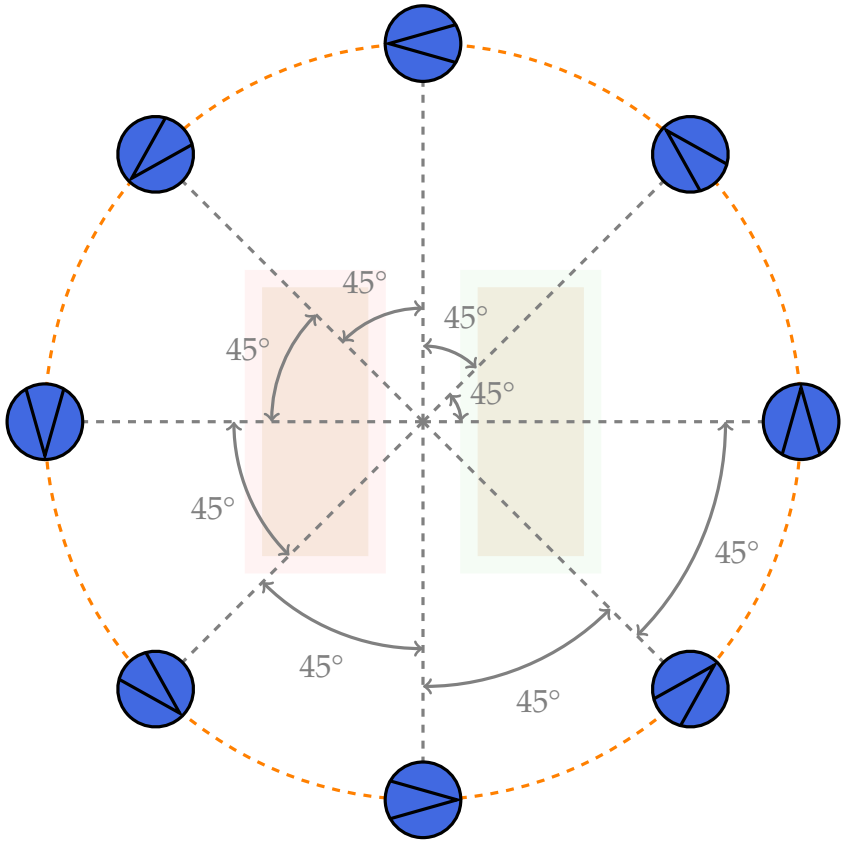


Figure 4.3 – The density metric: This figure represents the algorithm used to compute the density metric. The position of all robots with the origin centred on the shoes is obtained thanks to the arena tracking system (see Appendix **??**). Then the angle between every consecutive pair of robots is computed. We compare every angle to the target angle (360°/#robots, here 45°) by subtracting the second to the first. Then we divide by the target angle to normalise. Normalisation is necessary since an error of 10° for a target of 30° is not the same as an error of 10° for a target of 90°. The average of all those divisions is the error of the current time step. The example depicted on the figure would return an error of 0. See expression 4.2.

4.1.3.2 Setup

These experiments took place in the IRIDIA laboratory, in the arena room. The arena room is the place were the academic staff can manipulate the robots and run experiments. We used 8 e-pucks for every experiment. They were all equipped with tags to allow the arena tracking system (Stranieri et al., 2013) to locate them, and transmit the location to the simulator. At the beginning of each experiment, the robots were placed according to the type of experiment undertaken. There are two types of experiment. The first one consists in a random starting position for all the robots. Figure 4.4 illustrates the first set-up type. The other type consists in a specific starting position: all the robots are placed on the right of the shoes. The arena is a rectangle, bounded with wood boards or walls (on the right of the picture on the two figures there is a wall). Each experiment lasted 3 minutes. At each time step of the 3 minute long experiment, the positions of robots are saved into a file on the tracking system server. When the experiment is finished, the file is transferred to another computer located in the arena room. All the log files are then filtered by a Matlab script we implemented to remove the errors in the positions. Some robots are indeed misplaced or not found for several time steps due to some tracking system errors. Once the files have been cleaned, other Matlab scripts compute the values for the metrics listed in section 4.1.3.1.

4.1.3.3 Analysis

In this section we analyse the information gathered from our experiments. The goal of these experiments was to show that the error measures we defined in section 4.1.3.1 are decreasing over time during each experiment and stabilising after some acceptable time. We grouped the results by type of error measure. First we analyse the distance results, then the density results. We end with the time metric. For both types of starting configuration, the results are based on 10 runs of the experiment.

On figures 4.6 and 4.7, we show the evolution of the average distance error over time in the context of the random starting point experiment and in the context of the specific starting point configuration. On both figures, we observe a clear decrease of the error over time. On figure 4.6, the stabilisation is more obvious than for figure 4.7 where the median continues to decrease slightly at the end of the 3 minutes. Furthermore we can observe that the values of the error are a lot more spread for the specific setup (Figure 4.7) than for the random setup (4.6). We can explain this observation by the fact that the robots are close to each other at the beginning of the experiment. That, and the nature of the

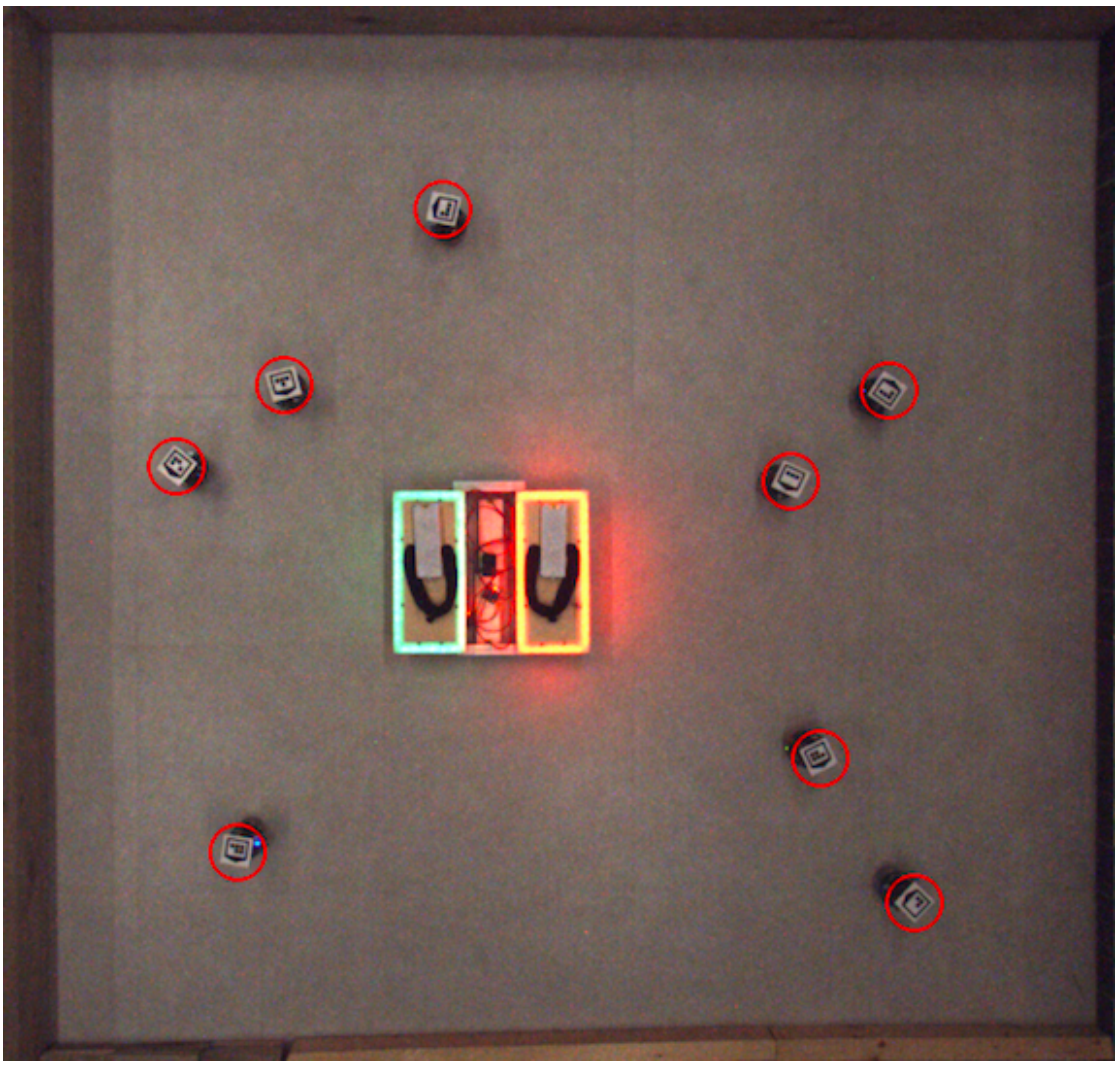


Figure 4.4 – The random setup: On this figure, we show an example of a random starting position for all the robots. The initial positions of all the robots are computed by a random plot of 8 points in Matlab, 8 couples (x,y) such that $x \in [-1, 1]$, $y \in [-1, 1]$. The robots are then manually placed as the points are shown on the Matlab generated plot. The shoes are centred under the camera. The camera is the origin of the environment: $(0, 0)$. The origin is not centred on the picture as we cropped the image to keep only the relevant part. The picture was taken from one of the 16 camera of the arena tracking system (Stranieri et al., 2013).

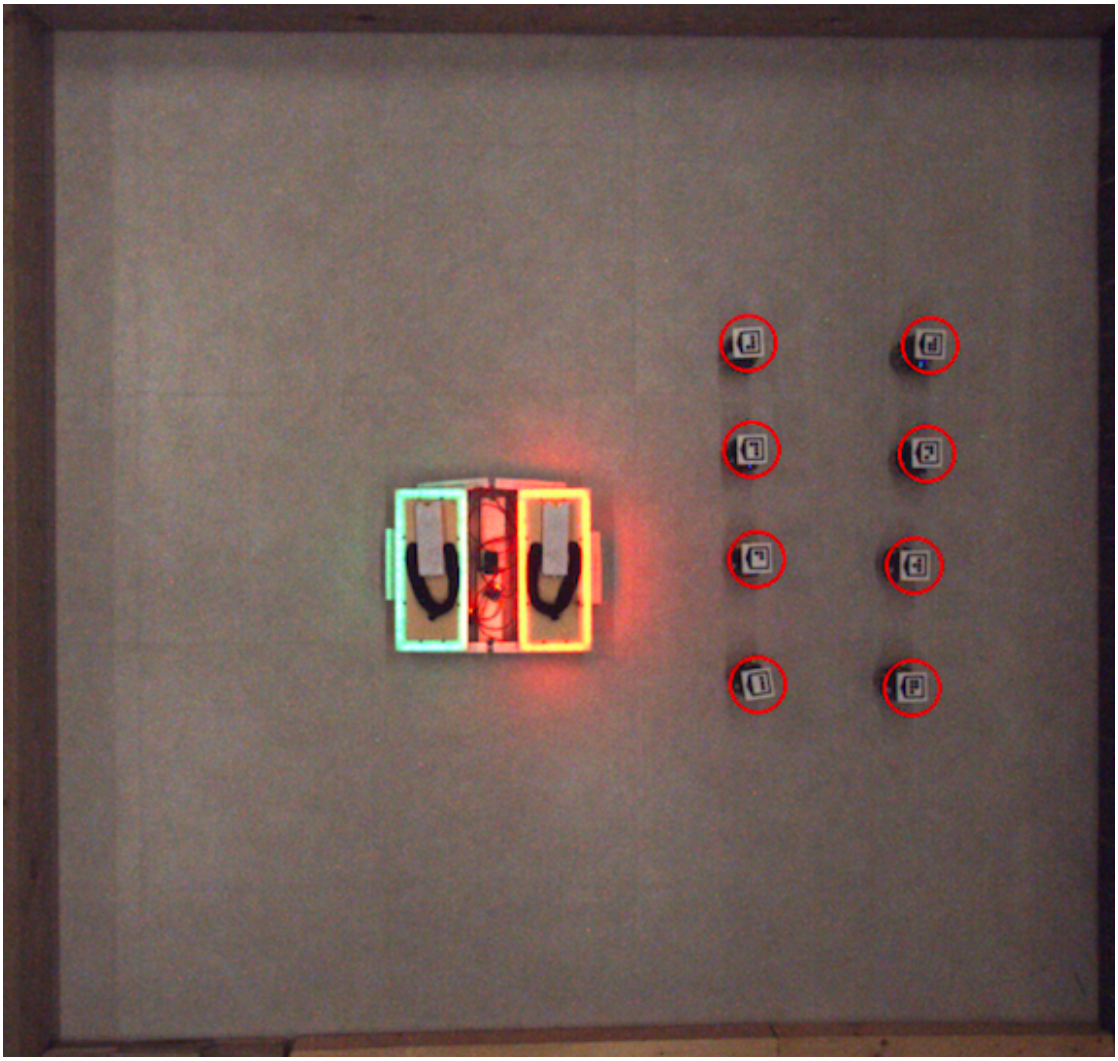


Figure 4.5 – The specific setup: On this figure, we show an example of a specific setup. All the robots are placed on the right of the shoes. This starting configuration is interesting since it could represent the usual configuration where all the robots are stored and waiting for the activity to begin. The shoes are centred under the camera. The camera is the origin of the environment: $(0, 0)$. The origin is not centred on the picture as we cropped the image to keep only the relevant part. The picture was taken from one of the 16 camera of the arena tracking system (Stranieri et al., 2013).

virtual forces exerted on the robots make the robots disperse in almost random directions at the beginning of the experiments. Typically, on figure 4.5, the robots on the right column would disperse to the right because of the column of robots on their left. Although the robots on the right are also attracted by the shoes, the repulsive forces of the robots on the left are stronger. Since for the random start configuration the robots are more likely to be dispersed in the arena, they are less subject to the forces of the other robots. Once the experiment starts, they just move towards the shoes. The behaviours observed during these experiments were not so different, leading to a small distribution for the error value, as seen on figures 4.6 and 4.8. From the distance metric point of view, the random starting position is the best.

As expected, the density error value is bigger for the specific configuration than for the random starting configuration as the robots are clustered on the right of the shoes (Figure 4.8 and 4.9). The angles separating the robots are further from the target angle (see section 4.1.3.1) than for the random setup. Once again, the values are decreasing over time for the two types of configurations. The values are stabilising after 2 minutes for the random configuration. The specific configuration does not stabilise completely at the end of the 3 minutes. The variance of the values distribution is bigger for the specific configuration (0.0241 for the random configuration, and 0.0560 for the specific one). However the difference is less important than for the distance metric. The error values are higher for the specific configuration. This could be due to the difficulty of the reorganisation undertaken by the robots: half of the robots have to go on the other side of the shoes. In the other configuration, the robots are already more dispersed around the shoes. As for the distance metric, the random configuration is the best alternative.

As the values are not completely stabilising for the specific starting configuration, it was interesting to see what was the evolution of the error for a longer experiment. We thus launched one experiment of 9 minutes with the specific starting configuration. Figure 4.10 (a) and 4.10 (b) show the evolution of the error over time. The two figures show a global decrease of the error value. However, the distance error on Figure 4.10 (a) has a more monotonous evolution than the density one. The latter undergoes an augmentation around 3 minutes. During the first 180 seconds, the two evolutions correspond to what was observed in figures 4.7 and 4.9. The two series of values tend to stabilise at 0.8. Figure 4.10 (c) shows the final state of the swarm at the end of the experiment. The robots are forming an acceptable circular shield around the human. One can conclude that 0.8 is a nice value for the error measure. The value 0.8 will thus be used as

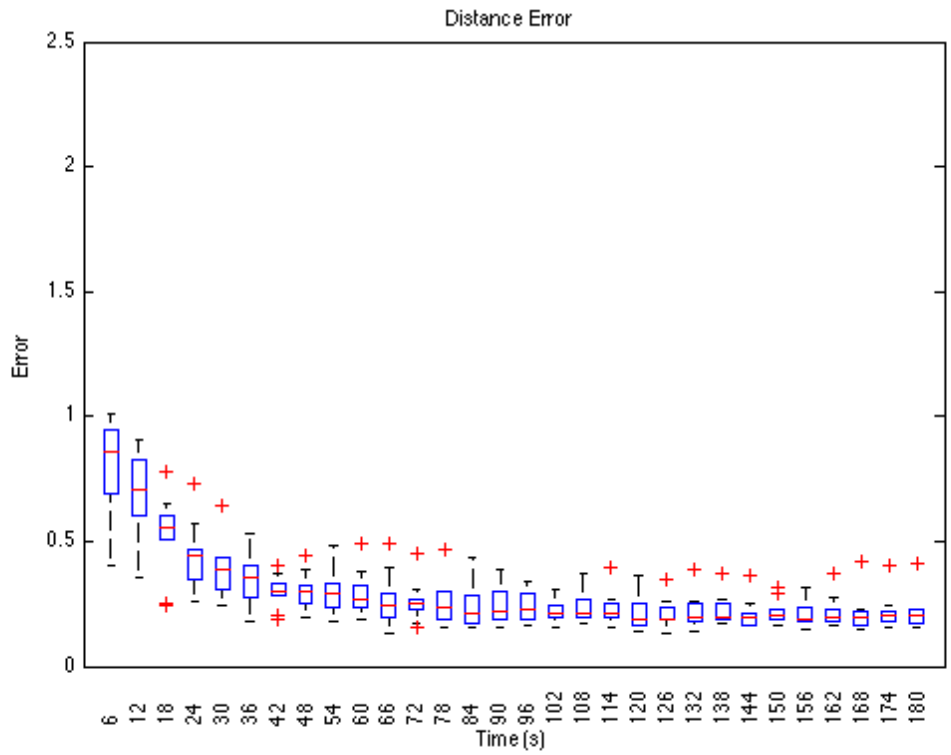


Figure 4.6 – The distance error for a random starting configuration: On this figure, we show the evolution of the average distance error through boxplots. It is an average on 10 experiments with real robots. The initial positions of all the robots are randomly selected. We observe that the behaviour of the swarm tends to lower the error measure. The variance of the values distribution is very low (0.0087 on average). A boxplot is a way of representing the distribution of a group of values. Each box corresponds to one group of values. On each box, one can see the median (the red line) of the distribution. The blue box is the interval between the first quartile and the third quartile: the interquartile range (IQR). The dashed black line is the interval of all the non outlier points. The upper whisker (the upper end of the black line) is the highest datum still within 1.5 IQR of the upper quartile (Wikipedia, 2015c). The lower whisker is the lowest datum still within 1.5 IQR of the lower quartile. Any point above the upper whisker or below the lower whisker is called outlier.

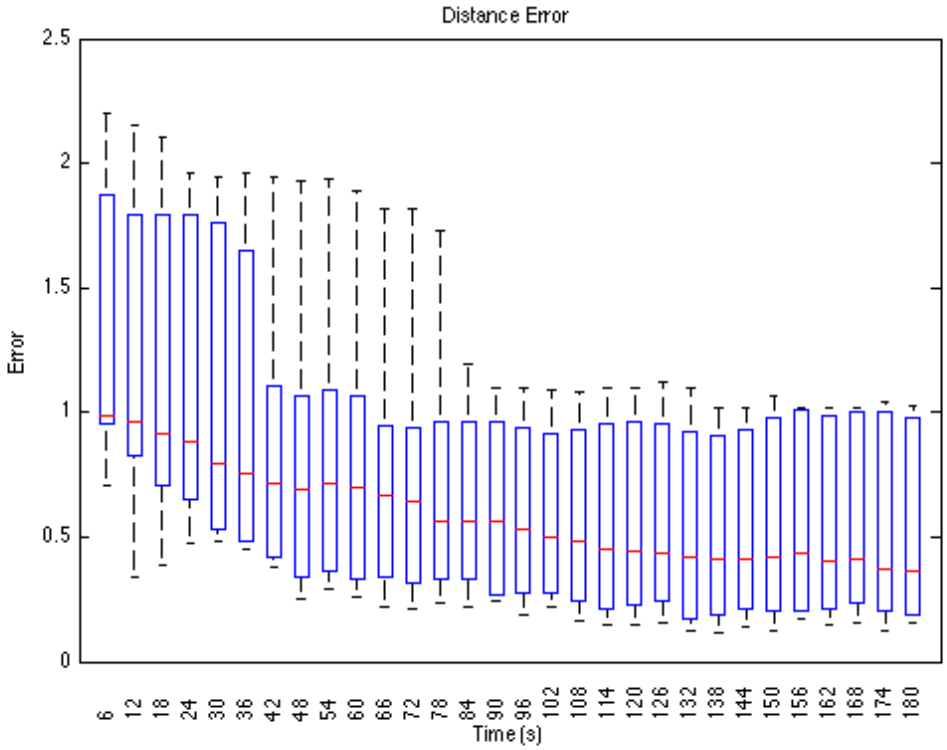


Figure 4.7 – The distance error for a specific starting configuration: On this figure, we show the evolution of the average distance error through boxplots. It is an average on 10 experiments with real robots. The robots are placed on a grid on the right of the shoes like shown on Figure 4.5. We observe that the behaviour of the swarm tends to lower the error measure. The variance of the values distribution is very high (0.2034 on average).

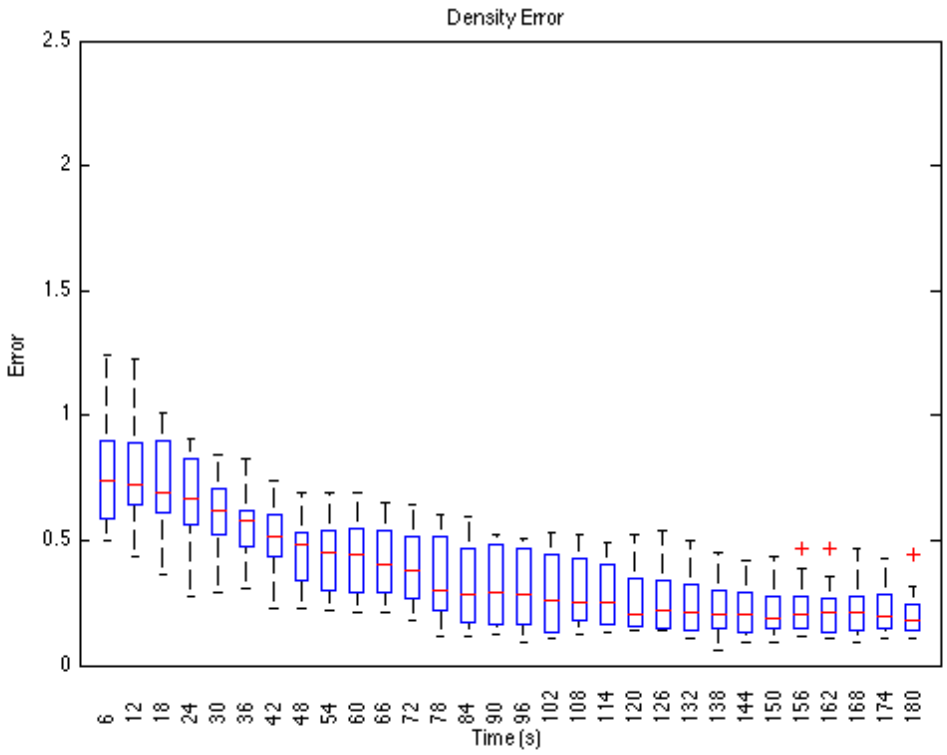


Figure 4.8 – The density error for a random starting configuration: On this figure, we show the evolution of the average density error through boxplots. It is an average on 10 experiments with real robots. The initial positions of all the robots are randomly selected. We observe that the behaviour of the swarm tends to lower the error measure. The variance of the values distribution is low (0.0241 on average).

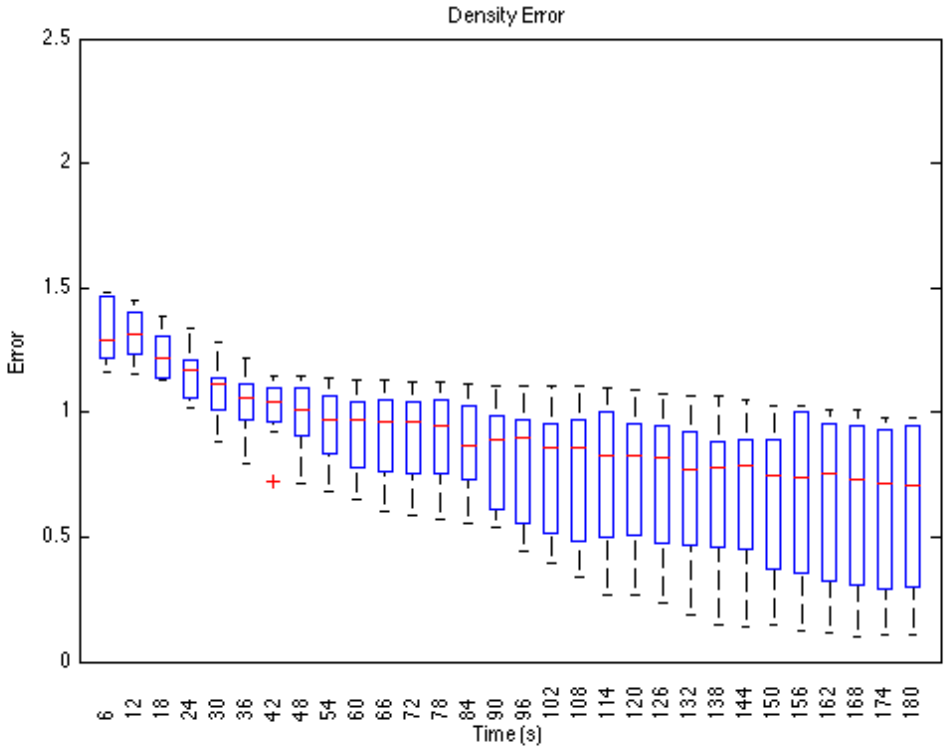


Figure 4.9 – The density error for a specific starting configuration: On this figure, we show the evolution of the average density error through boxplots. It is an average on 10 experiments with real robots. The robots are placed on a grid on the right of the shoes like shown on Figure 4.5. We observe that the behaviour of the swarm tends to lower the error measure. The variance of the values distribution is high (0.0560 on average).

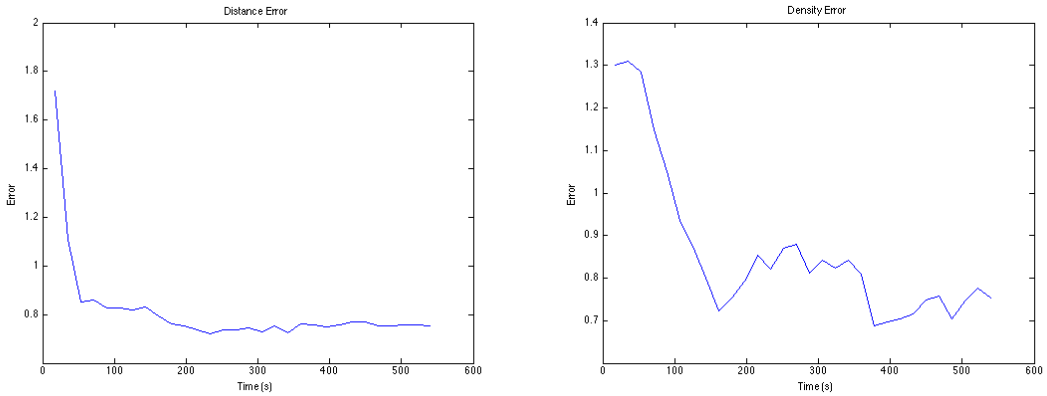
threshold of quality for the time metric.

If we use a threshold value of 0.8 for the quality measure, we observe a great difference between the random and specific configuration in the time it takes for the error measure to cross the threshold. Since for the distance and density metrics the best alternative was the random start configuration, it is logical that the alternative taking the less amount of time to cross the threshold of 0.8 is the random alternative. We take the median as reference. If the configuration is random, the swarm takes a few seconds on average to cross the threshold. On the other side, for the specific configuration, the swarm takes approximately 1min30s to do the same for the density metric, and 30 seconds for the distance metric.

As a conclusion, we can say that the random starting configuration is the best in the experiments we performed. It offered a consistent behaviour, as seen on Figure 4.6 and Figure 4.8. The variance is a lot smaller than for the other solution (see Table 4.1). At the end of the experiments, the value for the two types of errors were almost the same. Even if the specific starting configuration is a worse solution than the random starting configuration, the robots still managed to reach a state of acceptable quality in our standards after some time. In future applications, one would thus have to choose between the compactness of the specific configuration, leading to bigger delays, and the speed of the random solution, harder to set up.

Variances	Distance	Density
Random	0.0087	0.0241
Specific	0.2034	0.0560

Table 4.1 – The variances: On this figure we gather all the variances from the experiments we conducted. Each of the 4 values is an average of the variances of one of the 4 experiments.



(a) (b)

(c)

Figure 4.10 – The long experiment for the specific starting configuration: On this figure, we show the evolution of the distance error (a) and the density error (b) for one run of the experiment. The experiment lasted 9 minutes. One can see a global decrease in the two graphs showing that the behaviour tends to create a circle that presents good characteristics for our application. On (b) one can observe an augmentation around 200 seconds. The video from the experiment showed that the robots moved slightly to the right of the shoes, putting more space between the robots on the left. The robots returned to their correct position afterwards, as the graph suggests. (c) depicts the final state of the 9-minutes experiment using the specific configuration as starting configuration. It corresponds to a value of 0.8 for the two error measures. We defined this value as an acceptable quality.

4.2 Demonstration

In the previous section we performed experiments that did not involve any human. In this section, one human is using the shoes to walk in a virtually dangerous zone and stay protected by the robots. We called this experiment 'the demonstration'. The goal was to check if the robots were able to follow the human and warn him/her about the dangerous areas. It acts as a proof of concept.

4.2.1 Setup

We ran this experiment in the arena room, in the IRIDIA laboratory. We simulated one dangerous area on the floor using the arena tracking system (Stranieri et al., 2013; Reina et al., 2015) and ARGoS (Pinciroli et al., 2012; Garattoni et al., 2015). Eight E-pucks were used (Mondada et al., 2009). The experiment was recorded with a digital camera. Only one human took part of the experiment. The human had to walk slowly towards a circular dangerous zone whose diameter was 1 meter. At some point, the robots should notify him/her about the danger. The human would then walk around the dangerous area as depicted on Figure 4.11. The dangerous area was not visible to the human. If the robots would not tell to the human that a danger is nearby, the experiment would fail. The human started the experiment as shown on Figure 4.11, on the left of the dangerous area.

4.2.2 Analysis

In this section, we describe the unfolding of the experiment on the basis of pictures taken from the video we recorded. On Figure 4.12, we present the sequential events that occurred during the experiment. Figure 4.12 (a) shows the starting configuration of the experiment. The human is equipped with the shoes we built. The robots are surrounding him. He is ready to move towards danger. After a few steps, the human has reached the point where the robots see the dangerous area. They notify the danger to him by blinking their LEDs. One can see on Figure 4.12 (b) the two frontal robots blinking (their LEDs are currently off for a short period of time). The human decides to avoid the danger and turns left. While moving around the dangerous area, the robots in contact with it stay blinking. The 3 robots on the right side of the human on Figure 4.12 (c) are touching the dangerous area. Thanks to this feedback sent by the robots, the human knows its relative position to the danger. On this picture, one can also see the mechanism that switches off the LEDs of the shoe working. The left shoe is not touching the ground any more, releasing the switch and opening the

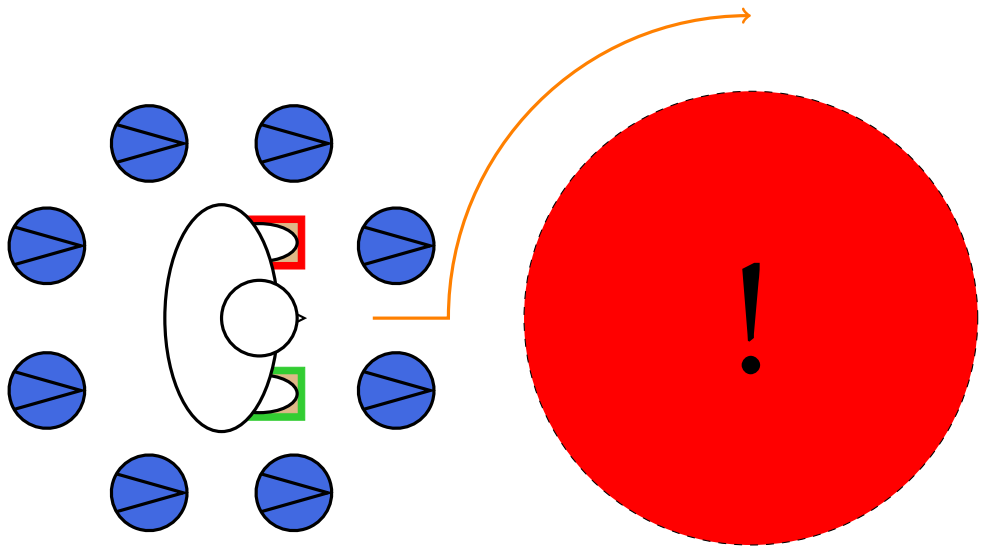


Figure 4.11 – The demonstration setup: On this figure, we show the staring configuration of the demonstration experiment, and the expected path the human has to follow. The human first walk slowly towards the dangerous area (the red disc). He is surrounded by 8 protecting E-pucks that follow him. The human is wearing the two shoes that were built to allow the robots to detect him.

circuit powering the LEDs. On Figure 3.2 we showed how the swarm of robots had to stay on the boundary of the dangerous area, even if the distance between the robots and the human is under the target distance. One can observe on Figure 4.12 (d) that the robots on the right of the human follow the curvature of the dangerous area. On Figure 4.12 (e) the human has the same heading he had at the beginning of the experiment. The danger area is on his right, as signalled by the robots on his right. Soon after, the robots on the right of the human will retrieve their correct distance to the human, and stop blinking. The danger area has been avoided.

The experiment was a success. The robots notified the human about the dangerous area. They stayed on the boundary of the area. However, for some of the robots, the distance to the human was oscillating. One can see on Figure 4.12 (b) that one of the robot on the back of the human is closer to the human than it should be. The speed of the human in this experiment was not constant. At the beginning, the human walked very slowly to the dangerous area. Then, he progressively increased the speed. The final speed was still low, though. For a real life application, the robots would have to be further from the human and

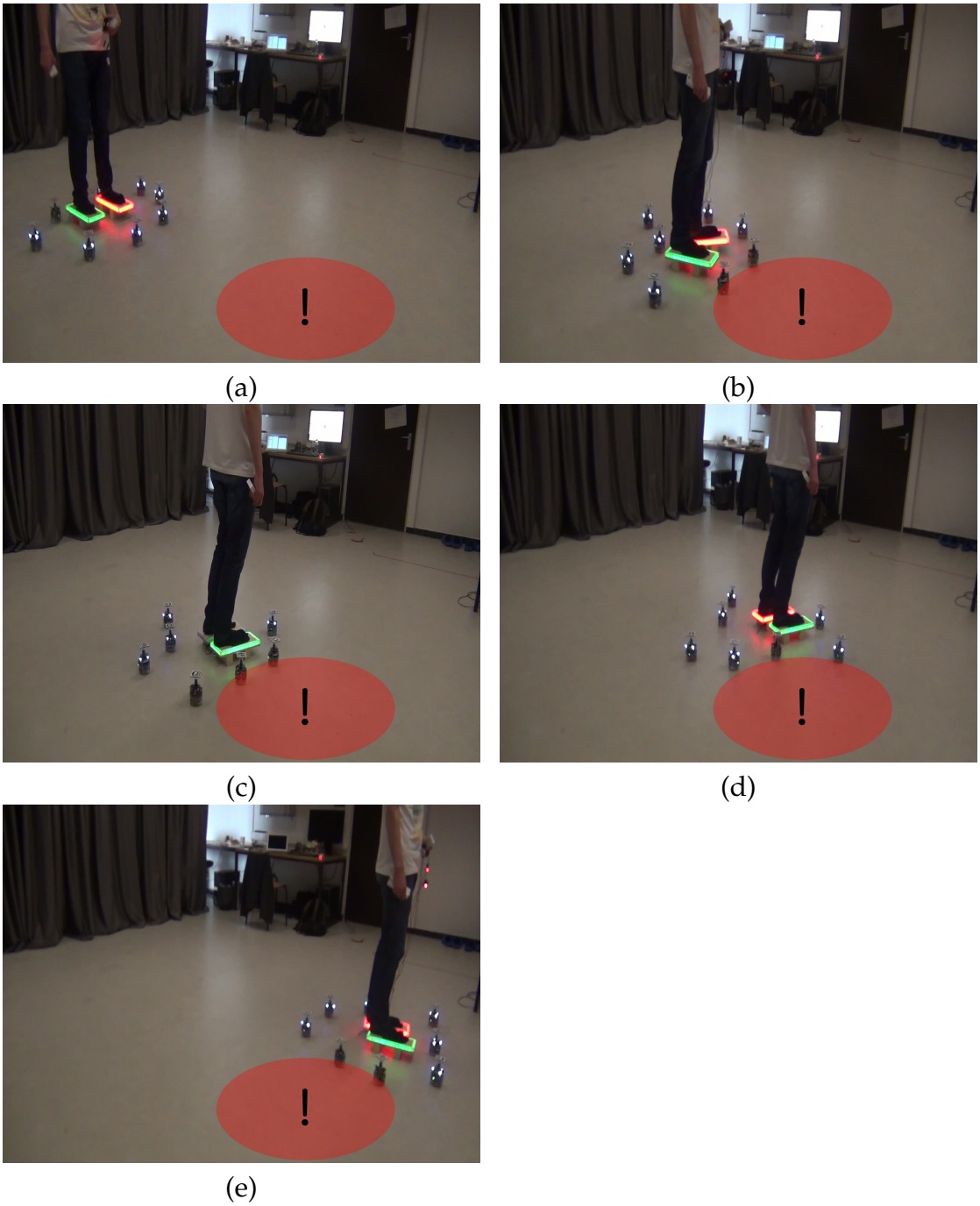


Figure 4.12 – I'm fabulous: Captured from the video of the experiment. The virtual dangerous area is added as an overlay as a red disc.

form a wider circle. This would allow a higher anticipation, and higher walking speeds as the robots would have larger margins to react to the human changes of direction and speed. This experiment must be seen as a proof of concept. Further development must be undertaken to obtain a system ready for real life applications.

Chapter 5

Conclusion

In this thesis, we addressed the problem of providing a protection to a human walking in a dangerous environment. This dangerous environment contains dangerous areas that must be avoided. These areas could be mines or radioactive areas. The human is unable to perceive them. We addressed this problem with an approach based on swarm engineering. We decided on the E-pucks (Mondada et al., 2009) as robotic platform to form a swarm encircling a human. The swarm is augmenting the capabilities of the human. The human is able to perceive dangers that he would not be able to perceive without the help of our robotic system. The controller is based on the principles of virtual physics and pattern formation. One of the challenges we had to address was to make the robots detect the human. In order to allow robots to detect the human, we built a portable device. We built shoes with coloured LEDs that the robots can detect with their camera.

We conducted multiple experiments in simulations and with real robots to show that our solution addresses our problem. We characterised the system composed by the swarm of robots and the shoes by performing more tests: we tested the range of detection of the shoes to see how far away the robots could detect them. The maximum distance we obtained is sufficient for our purpose, but not for real applications. We also analysed the time needed by the robots to form a circle around the human from different starting positions. We obtained the best results for the starting configuration where all the robots are randomly placed in the arena. On the other hand we obtained longer delays for the configuration where all the robots are clustered near the shoes. However, in all the cases, the robots surrounded the human. We ran an experiment with a human walking with the augmented shoes towards a dangerous area. The robots in contact with the dangerous area correctly warned the human.

Even though our solution satisfied our main problem (i.e., how to protect a human from going into dangerous areas), we believe that the current implementation of our solution has some limitations. First of all, the speed of the robots is too low for a real application. We made a comparison of the two speeds in section 4.1.1. The average speed of a human is 5 km/h (1.39 m/s). The maximum speed of the robots is 0.1 m/s. Faster robots would be needed for real applications. The scope of the omnidirectional camera is acceptable in our case, but in a real application it would be better if the robots detect the human from farther. The omnidirectional camera that the robots use to detect the human colour blobs is too sensitive to light conditions. Another sensor that is more robust with respect to the conditions of the experiments would be better. It could be interesting, as future work, to implement our solution on an other robotic platform, e.g., the foot-bot, mounted with a more precise camera and a combination of tracks and wheels (referred to as 'treels') (Dorigo et al., 2013). The robotic platform we used is only suitable for flat surfaces. However the outside real environments are everything but flat. Furthermore, for the purpose of a real life experiment, the robots would need batteries with higher capacity.

Future Works This solution can be enhanced by other future works. Once enhancements on the hardware side have been realised, one could update the controller to perform other related activities. One interesting application one could look into is guidance. The human would have less freedom of movement. Instead, the robots would encircle the human and help him to move in a previously computed and optimised direction. Guidance could also apply to animals. Below are presented 3 examples of guidance:

- **Vehicle Guidance:** This application is very similar to our project: helping someone or a vehicle that cannot see the danger augment his/her/its abilities to detect it, or preventing this vehicle from making any damage. One could imagine a vehicle whose driver cannot see the danger because of unusual circumstances (e.g., smoke, fog), or a vehicle behaving in a dangerous way. For example, one could imagine a boat entering an unsafe region of the sea (shallow water, streams), a channel or a harbour. In the case of the harbour, the robots would also act as buoys or shock absorbers. In the future, a greater part of vehicle will operate without a driver. Guidance could also be extended to driverless vehicles. These vehicles could be encircled by a swarm of robots to progress safely towards the destination. If we consider a boat without a driver, the robots acting as shock absorbers around the boat in the harbour are interesting. Indeed if an error occurs in the boat driving program, or if someone hacks into the boat, the robots

surrounding the vehicle would prevent big damages. One could argue that it would be easier to equip the boat or the other vehicles with the needed sensors or equipment. However, as these risky situations might occur very sporadically, it might be a bad investment to buy the ad hoc equipment. The U.S. Navy performed tests where a swarm of boats had to protect a main ship against enemies (Hsu, 2014).

◎ **Human Motion Synchronisation:** The second application we thought about was the synchronisation of multiple humans. For instance, one can imagine a restricted area where security rounds have to be made. With the help of the robots, the members of the security personnel could maximise the covered area by synchronising their progression and walking speeds. In case of intrusion or attack, the robots next to each member of the security personnel could help to defend the personnel and provide additional equipment (shields, aimbots, additional ammunition).

◎ **Crowd Control and Fishing:** A swarm of robots could also act as a crowd container to limit important crowd movements or to channel the flow of humans by forming barriers along the path. The same idea could be used for fishing. One could build robots that imitate fish behaviours to avoid frightening them and encircle schools of fishes. The robots would then release nets around the swarm of fishes that would connect to each other to form a sphere.

Bibliography

Bibliography

Asus. Asus xtion pro live, 2015. URL `http://www.asus.com/be-fr/Multimedia/Xtion_PRO_LIVE/`. Last consulted on 6th August 2015.

Nora Ayanian, Andrew Spielberg, Matthew Arbesfeld, Jason Strauss, and Daniela Rus. Controlling a team of robots with a single input. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1755–1762. IEEE, 2014.

Khelifa Baizid, Zhao Li, Nicolas Mollet, and Ryad Chellali. Human multi-robots interaction with high virtual reality abstraction level. In *Intelligent Robotics and Applications*, pages 23–32. Springer, 2009.

Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. Evolving mobile robots able to display collective behaviors. *Artificial life*, 9(3):255–267, 2003.

Cindy L Bethel, Christine Bringes, and Robin R Murphy. Non-facial and non-verbal affective expression in appearance-constrained robots for use in victim management: robots to the rescue! In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 191–192. ACM, 2009.

Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

Hande Çelikkanat and Erol Şahin. Steering self-organized robot flocks through externally guided individuals. *Neural Computing and Applications*, 19(6):849–865, 2010.

Mike Daily, Youngkwan Cho, Kevin Martin, and Dave Payton. World embedded interfaces for human-robot interaction. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 6–pp. IEEE, 2003.

Kerstin Dautenhahn, M Walters, Sarah Woods, Kheng Lee Koay, Chrystopher L Nehaniv, A Sisbot, Rachid Alami, and Thierry Siméon. How may i serve you?: a robot companion approaching a seated person in a helping context. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 172–179. ACM, 2006.

M. Dorigo, M. Birattari, and M. Brambilla. Swarm robotics. 9(1):1463, 2014. revision 138643.

Marco Dorigo, Dario Floreano, Luca M Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *Robotics & Automation Magazine, IEEE*, 20(4): 60–71, 2013.

Eliseo Ferrante, Ali Emre Turgut, Cristián Huepe, Alessandro Stranieri, Carlo Pinciroli, and Marco Dorigo. Self-organized flocking with a mobile robot swarm: a novel motion control method. *Adaptive Behavior*, page 1059712312462248, 2012.

L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, and M. Birattari. Software infrastructure for e-puck (and tam). Technical Report TR/IRIDIA/2015-004, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, July 2015.

Alessandro Giusti, Jawad Nagi, Luca Gambardella, Gianni Di Caro, et al. Cooperative sensing and recognition by a swarm of mobile robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 551–558. IEEE, 2012.

Armando Pesenti Gritti, Oscar Tarabini, Jerome Guzzi, Gianni Di Caro, Vincenzo Caglioti, Luca M Gambardella, Alessandro Giusti, et al. Kinect-based people detection and tracking from small-footprint ground robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4096–4103. IEEE, 2014.

Álvaro Gutiérrez, Alexandre Campo, Marco Dorigo, Jesus Donate, Félix Monasterio-Huelin, and Luis Magdalena. Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3111–3116. IEEE, 2009.

Jeremy Hsu. U.s. navy tests robot boat swarm to overwhelm enemies, 2014. URL `http://spectrum.ieee.org/automaton/robotics/military-robots/us-navy-robot-boat-swarm`. Last consulted on 17th August 2015.

Kazuko Itoh, Hiroyasu Miwa, Yuko Nukariya, Massimiliano Zecca, Hideaki Takanobu, Stefano Roccella, Maria Carrozza, Paolo Dario, and Atsuo Takanishi. Development of a bioinstrumentation system in the interaction between a human and a robot. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2620–2625. IEEE, 2006.

Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

James McLurkin, Jennifer Smith, James Frankel, David Sotkowitz, David Blau, and Brian Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75, 2006.

Microsoft. Microsoft kinect, 2015. URL `https://www.microsoft.com/en-us/kinectforwindows/`. Last consulted on 6th August 2015.

Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012. doi: 10.1007/s11721-012-0072-5.

Gaëtan Podevijn, Rehan O'Grady, and Marco Dorigo. Self-organised feedback in human swarm interaction. In *Proceedings of the workshop on robot feedback in human-robot interaction: how to make a robot readable for a human interaction partner (Ro-Man 2012)*, 2012.

John H Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999.

Andreagiovanni Reina, Mattia Salvaro, Gianpiero Francesca, Lorenzo Garattoni, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks. In *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2015))*, pages 1–6. IEEE Computer Society Press, Los Alamitos, CA, 2015. Paper ID sB_p3.

Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM Siggraph Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer, 2005.

William M Spears, Diana F Spears, Jerry C Hamann, and Rodney Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3): 137–162, 2004.

A. Stranieri, A.E. Turgut, M. Salvaro, L. Garattoni, G. Francesca, A. Reina, M. Dorigo, and M. Birattari. Iridia's arena tracking system. Technical Report TR/IRIDIA/2013-013, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, January 2013.

Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hähnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, et al. Minerva: A second-generation museum tour-guide robot. In *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, volume 3. IEEE, 1999.

Ali E Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120, 2008.

Wikipedia. Zcam — wikipedia, the free encyclopedia, 2014. URL `https://en.wikipedia.org/w/index.php?title=ZCam&oldid=639709035`. [Online; accessed 6-August-2015].

Wikipedia. Geta (footwear) — wikipedia, the free encyclopedia, 2015a. URL `http://en.wikipedia.org/w/index.php?title=Geta_(footwear)&oldid=651925222`. [Online; accessed 20-May-2015].

Wikipedia. Walking — wikipedia, the free encyclopedia, 2015b. URL `https://en.wikipedia.org/w/index.php?title=Walking&oldid=671922191`. [Online; accessed 1-August-2015].

Wikipedia. Box plot — wikipedia, the free encyclopedia, 2015c. URL `https://en.wikipedia.org/w/index.php?title=Box_plot&oldid=669188116`. [Online; accessed 5-August-2015].