Università degli Studi di Padova
Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Tesi di laurea magistrale

Distributed control based on evolutionary game theory: multi-agent experiment

Candidato:
Sofia Filippi
Matricola 1084115

Relatore:
Prof. Mauro Bisiacco

Anno Accademico 2015–2016

Contents

List of Figures

Notation

Graph Theory

$\mathbf{A}$: adjacency matrix of a graph
a_{ij}: elements of adjacency matrix $\mathbf{A}$
$\mathbf{B}$: incidence matrix
b_{ij}^k: elements of incidence matrix $\mathbf{B}$
$\mathbf{D}$: degree matrix of a graph
$\mathbf{Diag}(\mathbf{v})$: square diagonal matrix with the elements of vector $\mathbf{v}$ on the main diagonal
$\deg(i)$: degree of the i-th node
$\mathcal{E}$: set of edges of a graph
$\mathcal{G}$: undirected connected graph
$\mathbf{L}(\mathcal{G})$: Laplacian of the graph $\mathcal{G}$
$\lambda_i(\mathbf{L}(\mathcal{G}))$: i-th eigenvalue of $\mathbf{L}(\mathcal{G})$
$\mathcal{N}_i$: set of neighbours of agent i
$\boldsymbol{\Upsilon}$: matrix with columns the eigenvectors $\boldsymbol{v}_i$ of $\mathbf{L}(\mathcal{G})$
$\boldsymbol{v}_i$: i-th eigenvector of $\mathbf{L}(\mathcal{G})$ corresponding to eigenvalue $\lambda_i(\mathbf{L}(\mathcal{G}))$
$\mathcal{V}$: set of nodes of a graph

Population Dynamics

$[\cdot]_+ := \max\{0, \cdot\}$
$\arg\min f$: argument of the function f that minimizes it over its domain and constraint set

$\mathbf{d}_{ij}$: desired displacement among agent i and agent j
$d_{ij}^{\mathbf{x}}$: desired displacement among agent i and agent j along axis x
$d_{ij}^{\mathbf{y}}$: desired displacement among agent i and agent j along axis y
$\mathbf{D}_{\mathbf{x}}$: matrix of distances in formation with elements $d_{ij}^{\mathbf{x}}$
$\mathbf{D}_{\mathbf{y}}$: matrix of distances in formation with elements $d_{ij}^{\mathbf{y}}$
Δ: radius of the disk
f: potential function
F_i: fitness function of i-th agent
$\mathbf{F}$: column vector of fitness functions F_i
$KT(f)$: Kuhn-Tucker first order conditions on f
L: side of the square environment in simulations
ℓ: side of the triangle or length of the segment in formations
$\mathcal{L}$: Lagrangian
$\boldsymbol{\lambda}$: "payoff slack" of strategy $i \in \mathcal{V}$
m: population mass
μ: equilibrium payoff in a population
n: number of strategies
$NE(\mathbf{F})$: set of Nash equilibria of $\mathbf{F}$
$\nabla f(\mathbf{x})$: gradient of $f(\mathbf{x})$
R: rate at which agents receive a revision opportunity
$\mathbb{R}_+$: set of non-negative real numbers
$\mathbb{R}_{++}$: set of positive real numbers
$\boldsymbol{\rho}$: revision protocol
ρ_{ij}: conditional switch rate from strategy i to strategy j
$\mathcal{V}$: set of strategies available for all agents
$\mathbf{V}^{\mathbf{F}}(\mathbf{x})$: mean dynamics of $\mathbf{x}$
$\mathbf{x}$: population state vector
x_i: portion of agents selecting strategy i, element of $\mathbf{x}$ (unidimensional case)
$\mathbf{x}_i$: state of agent i, element of $\mathbf{x}$ (multidimensional case)
x_i^j: j-th component of the state of agent i, element of $\mathbf{x}_i$ (multidimensional case)
X: simplex set of population states

Unicycle

α: orientation proportional controller
δ: distance maintained from the pursuit point
e: error when tracking a path
g: control law
h: holonomic constraint
ω: angular velocity
$\mathbf{q} = [x\ y\ \theta]^\mathsf{T}$: pose vector
θ: orientation
$\mathbf{u} = [v\ \omega]^\mathsf{T}$: velocity vector
v: linear velocity

Mobile Robot

B_v, B_ω: translational and rotational friction coefficients
d: length of the semi-axis
e: error when tracking a path
J: inertia of the vehicle
K_v, K_m, K_ω, K_d: constant which map voltages, linear and angular velocities in forces and torques
M: mass of the vehicle
ω: angular velocity
$\mathbf{q} = [x\ y\ \theta]^\mathsf{T}$: pose vector
θ: orientation
V_d, V_m: differential and average voltages applied to the wheels
V_L, V_R: voltages applied to left and right wheels
v_L, v_R: linear velocities provided by the left and right wheels

Implementation

β: constant that converts distances in wheels-rotation angle
ℓ: side of the triangle or length of the segment in formations
τ: sampling time
$V_i^{\mathbf{F}}$: mean dynamics of x_i

Chapter 1

Introduction

In this Master's Thesis project a platform that simulates the behaviour of a multi-agent population has been created. The platform uses LabVIEW as main tool and Lego Mindstorm robots as agents. Mindstorms can be controlled through the toolkit NTX of LabVIEW and by implementing algorithms in G language from a PC. The Mindstorms are going to be associated to agents in order to test, analyse and develop multi-agents control strategies. The communication between the computer and the agents is established through USB and Bluetooth connections.

The other important component of the platform is the camera, which is directly connected to the PC. The role of the camera is to acquire information from the environment where the Mindstorms are moving in order to identify their positions. The localization is made by using the pattern matching virtual instrument that is responsible to recognize and distinguish a pattern attached to each agent. Besides, the virtual instrument evaluates its coordinates position in the space.

The theoretical aspect of this work is constituted by the selection and design of algorithms that can control these agents in a distributed manner, using game theory concepts, and graph-based techniques.

The application of game theory in engineering as control technique permits to model the interaction among different agents, which make individual local decisions pursuing a global and common objective, the Nash equilibrium [38]. Moreover, game theory allows to solve optimization problems because the achievement of the Nash equilibrium corresponds to the extreme of a potential function [36].

1.1 Motivation

In this Master's Thesis project, the motivation that has determined the choice of working with evolutionary game theory is that it can be applied to design non-centralized controllers. These tools are used to coordinate the motion of agents to perform particular formations in the space or to solve rendezvous problem.

Optimization techniques based on evolutionary game theory allow to model the behaviour of a population of agents that interact to each other. In this non-cooperative game approach, each agent pursues its individual benefit. Moreover, under a special class of games known as *potential games*, the solution of the game corresponds to a maximum point of the respective potential function. Consequently, evolutionary game theory can be used for optimization purposes [54].

The topics studied in this thesis could be adapted and applied to solve more complex problems that concern large-scale systems. Instead of considering a single population, a complex system can be seen as a connection of multiple populations. The populations that compose the system are formed by agents that act according to individual rules. A system of this kind is characterized by a non-centralized structure and the control scheme is non-centralized too. The latter is constituted of local controllers that take decisions independently. The local decisions taken inside the populations can be made to achieve a global objective, expressed with a cost function.

1.2 Objectives

1.2.1 General objective

The final goal of the project is to control the trajectories a multi-agent population of robots, Lego Mindstorms EV3, using distributed evolutionary game theory algorithms.

1.2.2 Specific objectives

Considering the specific case of the project, the general objective is developed by achieving the following points

- design and implement a working platform that may illustrate the control of multi-agent systems, where the role of agents is taken by LEGO Mindstorm EV3 robots;
- design distributed and non-centralized versions of projection and replicator dynamics equations of evolutionary game theory;
- test with simulations projection and replicator dynamics approaches of evolutionary game theory to solve the rendezvous problem in a multi-agent situation;
- test with simulations projection and replicator dynamics approaches to perform formation in the space in a multi-agent situation;
- design dynamical models of mobile vehicles to simulate the tracking of the generated trajectories;
- implement in the developed platform non-centralized control algorithms to make the Mindstorms perform assigned tasks like reaching a common position in the space and generate particular formations.

1.3 Thesis structure

The body of the thesis is organized as follows. Chapter 2 presents a general overview of state of the art regarding distributed control techniques applied in multi-agent systems. In Chapter 3, the control problem is formulated, the components of the platform are introduced, and finally, several algorithms are proposed and tested with simulations. Firstly, the trajectories are generated to reach the goal and then these trajectories are established as references for local controllers that consider the dynamical behaviours of the agents. Chapter 4 describes the real implementation with the Mindstorms and presents the results obtained applying the algorithms introduced in the previous chapter. Chapters 6 and 7 introduce the environmental impact, and the budget evaluation related to the realization of this project, respectively. Chapter 8 presents the conclusions. In this section are summarized the work developed, the results obtained, the contributions, and are proposes some directions for the future research.

Chapter 2

State of the art

The origin of control theory is related to the control of a single system through different control methodologies, such as proportional-integral-derivative (PID) control, adaptive control, intelligent control, and robust control, like in [44] and among many others.

In the past two decades, the attention has shifted to the control of multiple interconnected systems because many benefits can be obtained replacing a unique complex system with several simple systems [32].

Three approaches are commonly used for the control of multiple interconnected systems: a centralised, a decentralised, and a distributed approach.

The centralised approach is based on the assumption that a powerful central station is responsible to control a group of systems and it is a direct extension of the traditional single-system based control methodology. Instead, the distributed approach does not require the existence of a central station, but a trade-off is that the design of a non-centralised control is far more complex than the centralised counterpart and in case of malfunction the failing point is difficultly detected. The centralised approach requires the collection of all the information measured from the system in a central point, then, the costs associated to communication structures can be discussed since distributed approaches require smaller communication networks. The distributed approach is the most promising in presence of physical constraints, such as limited communication/sensing range, low bandwidth, and large number of systems involved [70].

One of the first practical experiments of distributed control applied to mobile robotic has been done in the Institute for Process Control and Robotics (IPR) of the University of Karlsruhe [51]. It has been developed the Karlsruhe Autonomous Mobile Robot (KAMRO) which is a system composed of several subcomponents like two manipulators, hand-eye-cameras, one overhead-camera and a mobile platform. It uses the distributed control architecture KAMARA (Karlsruher Multi Agent Robot Architecture) that can overcome coordination problems such as independent task execution in a decentralised and distributed manner. With this platform, they solved problems that with centralised control architectures cannot be overcome. For example, if a system component obstructs the scene to be examined by the overhead camera, KAMARA requests the blocking component to leave the scene before the camera mission is started. Moreover, if the state controller recognizes that a mission cannot be performed by the system itself, it delivers the mission to the cell planning system, which involves other systems in the solution process. It is also possible that a system component breaks, in this case KAMARA recognizes the damaged agent because these components fails during task execution. This project has three innovative skills: it is hybrid because can intervenes a centralised in special cases and the independent task execution by agents is used in the normal case; it solves the problem of deadlocks in communication, cooperation and coordination; it optimizes the system performance by reallocation of tasks to the agents [51].

Recently, the control of a group of autonomous agents has been investigated intensively from different perspectives [19]. The main control objective is to have the agents work together in a cooperative manner. Cooperation refers to the close relationship among all agents in a team, where the fundamental characteristic is the sharing of information. Distributed coordination of multiple autonomous agents has become an active research topic because many advantages can be obtained with its application, such as robustness, adaptivity, flexibility, and scalability [26].

The study of distributed control of multiple autonomous agents was motivated by the work in distributed computing [3], management science [35], and physics [72]. But more specifically, in controls society, the pioneer work was given in [4] and [71], where an asynchronous agreement problem has been studied for distributed decision making problems.

Some examples of agreement problem in multi-agent scenario solved distributively are the following. In [11] the authors consider the problem of information agreement among multiple agents in the presence of limited and unreliable information exchange with dynamically changing interaction topologies. In [29], they propose a method for decentralised information exchange between vehicles which realizes a dynamical system that supplies each vehicle with a common reference to be used for cooperative motion. In [39] authors propose a distributed approach based on a Newton-type method for solving minimum cost network optimization problems. In [55] the author studies a network where agents interact via time-dependent communication links and each agent updates his current state based upon the current information received from neighbouring agents. In [56] authors discuss agreement problems for networks of dynamic agents with directed/undirected and fixed/switching topologies and introduce two protocols for networks with and without time-delays.

Moreover, remaining in the area cooperative strategies in multi-agent systems, as explained in [18], the intrinsic distributed feature of the problem allows to use a graph that models agent-to-agent interactions depending on the relative locations of agents in space, as in wireless or line-of-sight communication. Given a coordination task to be performed by the network and a proximity graph representing communication constraints, the problem can be solved with an approach based on gradient flows, on the analysis of emergent behaviours (by notion of neighbouring agents and an interaction law), on optimizing local objective functions to achieve the desired global task and on the composition of simple behaviours to design more complex strategies.

Going into details, distributed control algorithms are able to solve several specific problems of motion-coordination tasks such as deployment [16][17], rendezvous [2], cohesiveness [42], and consensus [5] by using an aggregate objective function. Non-smooth analysis is used to identify the extreme points of the objective functions. Moreover, in [18] methods from circulant and Toeplitz tridiagonal matrices and the invariance principle for set-valued discrete-time dynamical systems are used.

Regarding the game theoretical algorithms that can be implemented to control a dynamical population of agents, the field that is considered is the evolutionary game theory [27] and in particular the projection and replicator dynamics equation. Useful researches in this area can be found in the following sources. In [59], the invariance property of the simplex set under the replicator dynamics has been used to satisfy a

coupled constraint in a resource allocation problem. Some works take advantage of this property for control purposes, e.g., [7]. However, this population dynamics approach is unable to take into account more constraints besides the positiveness of variables and the one imposed by the simplex. To overcome this issue, authors in [8] have proposed a population dynamics approach that may consider multiple constraints by adding dynamics over population masses. Moreover, a distributed model-free control based on population dynamics is conceived in [9]. The subject method guarantees that the feasible region is attractive, so that the distributed control system is robust against disturbances.

In this Master's thesis control techniques based on distributed evolutionary game theory are designed and applied to solve the formulated problem and to achieve the pre-established control objectives. It follows a brief overview and description of the sources from which the algorithms have been deduced.

Algorithms of game theory, and in particular of population dynamics, are tested to find the optimal solution to reach common position in a situation that involves multiple agents. In [36] stable games are studied. Stable games are a class of population games characterized by self-defeating externalities[1]. In [45], a class of matrix games is considered. Under these games, successful strategies are rewarded by high reproductive rates and, over time, the strategy mix evolves to a stable state. In [64],[66],[65], and [67], authors present the notion of an evolutionarily stable strategy and define the concepts of population games and revision protocol. Starting with a population game and a revision protocol, a dynamic process can be derived, which describes how the aggregate behaviour of agents changes over time: the evolutionary game dynamics. Deterministic differential equation models of game dynamics are considered and several applications of evolutionary game theory are introduced. Moreover, the notions of ESS, local stability theorem, and potential games for large population are defined. Furthermore, an approach to model recurring strategic interactions in large populations of agents built upon population games is described.

The projection and replicator equations are classic game dynamics used in evolutionary game theory and they require full information to evolve to the solution. This property cannot always be observed in multi-agents context because situations in which each

[1]Self-defeating externalities: when agents revise their strategies, the improvements in the payoffs of strategies to which agents are switching are always exceeded by the improvements in the payoffs of strategies which agents are abandoning

agent can get information only from a neighbourhood defined by a network topology are taken into account. In these cases, non-centralised controllers are used in order to consider only local information rather than full-information.

In [60], the authors propose a novel method inspired by the replicator dynamics, the local replicator dynamics, which uses only local information in neighbourhoods defined by a connected graph. A game theoretical approach for addressing distributed optimization problems that permits relaxations in the structure of the communication graph is proposed in [49]. In particular the graph not need to be time-invariant and connected, a common condition in representation of multi-agent systems. An overview of developments of game theoretic methods for distributed control in engineered systems is provided in [52]. The design of utility functions and learning processes is also presented. In [8], a methodology for solving constrained optimization problems distributively is proposed. To this end, dynamics of the population masses to the population dynamics are added. [48] focuses on obtaining a local control laws for the individual agents to ensure that the emergent global behaviour is desirable with respect to a given objective, using game theoretical notions. Finally, in [6], authors propose a general method that allows us to deduce several distributed population dynamics, such as replicator, Smith, logit, and projection dynamics.

Chapter 3

Case setup

3.1 Problem formulation

The control objective is to make a population of agents perform optimal trajectories and reach assigned tasks. Algorithms of distributed projection dynamics and replicator dynamics are applied to solve these problems.

A fundamental component of the Master's thesis is the proper selection and the design of algorithms that can be used to control this multi-agent experiment in empirical tests.
The experiment is not centralised, even though a PC controls all Mindstorms movements. Instead, the control scheme is distributed because evaluation of the control action to be applied to each agent is based on local information, determined by neighbourhood in graph theoretical point of view.

First of all, distributed algorithms of projection and replicator dynamics are applied in order to generate the ideal trajectories that lead the agents to reach a common destination point. Secondly, these algorithms are modified so that the agents are able to position themselves in particular formations in the space, e.g., to generate a triangular shape or alignment.

Afterwards, the trajectories obtained with the previous controllers are set as references that agents with dynamical behaviour have to track. The dynamical model that taken into account are the unicycle and the three-wheels mobile robots, since they are a suitable representation of the real robots used in practical experiments.

3.2 Tools

3.2.1 Agents

The role of agents is taken by LEGO Mindstorm EV3 robots. The chosen configuration presents two rear wheels driven by two independent motors and a small back wheel that has the purpose to keep in balance the robot (unicycle model). Each Mindstorm is able to move around an environment going forward and backward and rotating.

3.2.2 Camera and pattern recognition

The camera is connected through USB port with the computer. Its position is fixed so that the reference frame of the pictures taken by the camera is always attached to the world reference frame. The role of the camera is to film the environment in which the robots are moving. The sequential images are continuously analysed by the pattern recognition module that identifies the positions of the robots inside the pictures. The recognition is made by searching a pattern that contains the image of the robot. At each time, a correspondence is found and the program returns the coordinates x-y of the pixel in the center of the rectangular area containing the robot. In this way, the overall proposed scheme can identify the position of the robots inside the environment.

3.2.3 Bluetooth

The PC and all the robots are equipped with Bluetooth and it is used to remotely control the actions of the Mindstorms. It is assumed that the robots cannot exchange informations directly to each other because the commands exchanged through Bluetooth go unidirectionally from the PC to them. The Mindstorms are not aware of their positions and this is the reason why it is necessary to use the camera in order to have a feedback of their locations. Moreover, the Mindstorm cannot communicate directly to each other since they are not connected.

Figure 3.1 represents the platform described so far. The experiment set-up in structured in this manner, with a central unit embodied by the computer, but the control algorithms respect a distributed topology, i.e., distributed algorithms are tested with this platform.

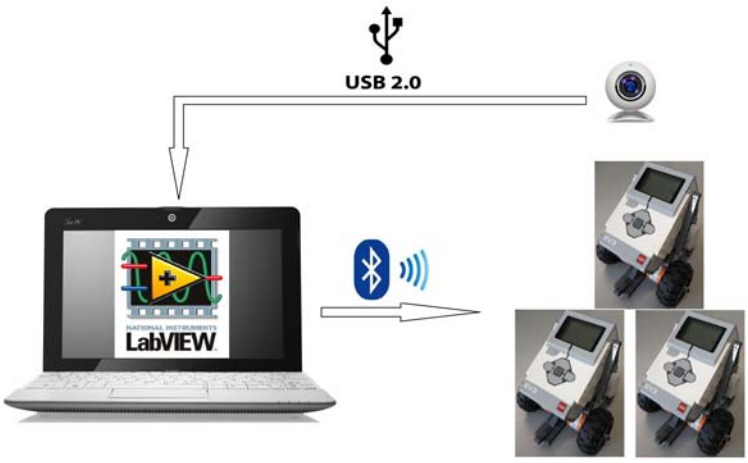


Figure 3.1: Platform components.

3.3 Control problem scheme

3.3.1 Preliminaries

3.3.1.1 Distributed and decentralised control

Before explaining how the non-centralised control is structured and how it is implemented, it is necessary to define which are the features that distinguish centralised, decentralised, and distributed systems.

A centralised system directly controls the operation of the individual units and information flow from a single central unit. All agents are directly dependent on the decisions of the central controller and they are forbidden to coordinate and work-together among themselves.

In a decentralised system the decision power is not entrusted to a central unit. According to the graph[1] in Figure 3.2, a decentralised system can be represented with a hierarchical structure composed by a central node that is connected with middle nodes that in turn control the most external nodes. In such decentralised system each node controls others directly below it and becomes controlled by the one directly above it. In doing so, the central node can control the entire system. The external

[1]A graph is a diagram representing a system of connections or interrelations among entities. These entities are called nodes and the connections are the links or edges.

nodes are forbidden to coordinate and work-together among themselves.

Unlike centralised and decentralised systems, a distributed system has not a central unit control. Referring once more to the graphical representation of the system in Figure 3.2, all the nodes have equal decision-making power. Each node can interact with its neighbouring nodes using commonly agreed protocols, building reliable networks that can be many times more resilient than centralised or decentralised systems. In case one of the nodes fails, with a distributed structure the risk of isolate a subset of nodes is lower. The reliability of the system grows with the increase in the number of nodes. Each node and its neighbours coordinate and work-together among themselves.

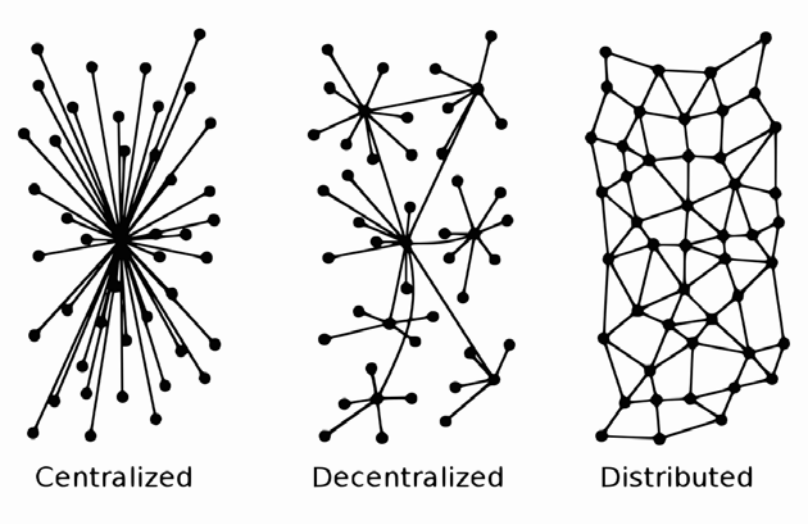


Figure 3.2: Types of network topology.

When the considered system involves a large number of states with multiple constraints, it is disadvantageous to solve the optimization problem with a centralised control because of the amount of variables that have to be taken in consideration. A solution is to adopt a distributed control that can solve effectively the problems of dimensionality and information structure constraints.

Distributed optimization algorithms can be based on convex optimization [10], Newton method [12][40][41], gradient and subgradient methods [4][13][43][58], proximal procedures [13][14], consensus algorithms [20][47] and game theory [48][49][62].

In particular, in this Master's thesis, the interest is to apply evolutionary game theoretical algorithms to solve constrained optimization problems in a distributed way

by the use of population dynamics. System is seen as a set of interconnected subsystems that allows to obtain a global optimal solution by exploiting local information and the use of local information also improves computational efficiency.

3.3.1.2 Graph theory

Network science is applied to study multi-agent coordination and control, and it is important in order to understand the role that interactions among single entities play in the collective functionality of systems.

Distributed multi-agent networks are constituted of agents that require to operate in concert with each other in order to achieve a global objective, while having access to limited computational resources and local informations.

Systems not completely connected have problems of communication among agents, then it is important to find a way to control the global system with limited connections and local decision-making.

An useful tool to describe networks where agents are not all connected to each other is graph theory, where agents are represented with nodes in a graph and an edge symbolizes the existence of an interaction among nodes it connects.

Consider a network constituted of n dynamical nodes, or vertices, labelled as $1, 2, \ldots, n$, interconnected via information-exchange links, or edges. The network that describes the relationships among nodes can be represented with an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes and $\mathcal{E} \subseteq \{(i, j) : i, j \in \mathcal{V}\}$ is the set of links of the form (i, j) with $i \neq j$. If a link connect two nodes they are considered adjacent. From this definition, the adjacency matrix $\mathbf{A} = [a_{ij}]$ can be defined as

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, the set of neighbours of node i is defined as $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$.

Another fundamental concept of graph theory is the degree of a node that represent the number of edges connected to a node, and it is denoted by $\deg(i)$, $i \in \mathcal{V}$. The degree matrix $\mathbf{D}$ is the diagonal matrix with elements equal to $\deg(i)$ along the main diagonal and zero outside, i.e., $\mathbf{D} = \mathbf{Diag}\left([\deg(i) \quad \ldots \quad \deg(n)]^{\mathsf{T}}\right)$.

Besides, the graph $\mathcal{G}$ should be connected, i.e., for every pair of vertices in $\mathcal{V}$, there is a path that has them as its end vertices. Since $\mathcal{G}$ is undirected, then the information flow through all the links is bi-directional and $(i, j) = (j, i)$.

Given a graph $\mathcal{G}$ with n vertices, its Laplacian matrix denoted by $\mathbf{L}(\mathcal{G}) = [l_{ij}]$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

According to [74], the spectrum of the Laplacian for a connected undirected graph assumes the form

$$0 = \lambda_1(\mathbf{L}(\mathcal{G})) < \lambda_2(\mathbf{L}(\mathcal{G})) \leq \ldots \leq \lambda_n(\mathbf{L}(\mathcal{G})), \tag{3.1}$$

where $\lambda_i(\mathbf{L}(\mathcal{G}))$, $i = 1, \ldots, n$, are the eigenvalues of $\mathbf{L}(\mathcal{G})$ and in particular $\lambda_1(\mathbf{L}(\mathcal{G}))$ is the zero eigenvalue with corresponding eigenvector equal to $\mathbb{1}$.

Let $\boldsymbol{v}_i$ be the eigenvector that corresponds to the eigenvalue $\lambda_i(\mathbf{L}(\mathcal{G}))$. These eigenvector are normalized and mutually orthogonal and define the matrix $\boldsymbol{\Upsilon} = [\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \ldots \quad \boldsymbol{v}_n]$.

The incidence matrix $\mathbf{B}$ of dimension $|\mathcal{E}| \times |\mathcal{V}|$ has elements b_{ij}^k where (i, j) is the edge connecting nodes $i, j \in \mathcal{V}$ and $k \in \mathcal{V}$. The elements of the $\mathbf{B}$ are defined as

$$b_{ij}^k = \begin{cases} 1 & \text{if } k = i \text{ or } j, \\ 0 & \text{otherwise,} \end{cases}$$

and the Laplacian matrix $\mathbf{L}(\mathcal{G})$ can be expressed by using the incidence matrix as $\mathbf{L}(\mathcal{G}) = \mathbf{B}^\mathsf{T}\mathbf{B}$. Consider the i-th eigenvalue $\lambda_i(\mathbf{L}(\mathcal{G}))$ and the corresponding eigenvector $\boldsymbol{v}_i$, then

$$\begin{aligned} \lambda_i(\mathbf{L}(\mathcal{G})) &= \boldsymbol{v}_i^\mathsf{T}\mathbf{L}(\mathcal{G})\boldsymbol{v}_i \\ &= \boldsymbol{v}_i^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{B}\boldsymbol{v}_i \\ &= (\mathbf{B}\boldsymbol{v}_i)^\mathsf{T}(\mathbf{B}\boldsymbol{v}_i). \end{aligned}$$

Due to the fact that $\lambda_i(\mathbf{L}(\mathcal{G}))$ can be written as the inner product of the vector $\mathbf{B}\boldsymbol{v}_i$ with itself, this shows that $\lambda_i(\mathbf{L}(\mathcal{G})) \geq 0$ and so the eigenvalues of $\mathbf{L}(\mathcal{G})$ are all non-negative.

The graphical representation of a network is used throughout the document to show the features of population dynamics and describe the relationships between agents. In particular, the notion of adjacency matrix is useful to represent the set of neighbours of the agents in distributed control algorithms.

3.3.1.3 Population dynamics

The following sections about evolutionary game theory a nd population dynamics are based on the approaches proposed in [67] and [64] for notation, math definitions, theorems and propositions.

Consider a population composed of a large and finite number of rational decision makers (agents[2]). All the decision makers can select among a set of strategies denoted by $\mathcal{V} = \{1, \ldots, n\}$, which represent the actions that the agents can make. The scalar $x_i \in \mathbb{R}_+$ represents the portion of decision makers selecting strategy $i \in \mathcal{V}$. The vector $\mathbf{x} = [x_1 \quad \ldots \quad x_n]^\mathsf{T} \in \mathbb{R}_+^n$ is the population state or strategic distribution, whose entries are non-negative real numbers.

The set of possible states is

$$X = \left\{ \mathbf{x} \in \mathbb{R}_+^n : \sum_{i \in \mathcal{V}} x_i = m \right\},$$

where $m > 0$ is the population mass.

Agents playing the i-th strategy obtain a reward given by a fitness function denoted by $F_i(\mathbf{x})$, where $F_i : X \mapsto \mathbb{R}$ is a continuous map that specifies the payoff associated with the strategy $i \in \mathcal{V}$. The population game is completely characterized by the fitness function vector $\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}) \quad \ldots \quad F_n(\mathbf{x})]^\mathsf{T}$.

The fundamental solution concept of noncooperative game theory is Nash equilibrium, the requirement that each agent choose a strategy that is optimal given the choices of the others.

Definition 3.3.1. *Population state* $\mathbf{x}^*$ *is a Nash equilibrium of* $\mathbf{F}$*, denoted by* $NE(\mathbf{F})$*, if no agent can improve his payoff by unilaterally switching strategies. More explicitly,* $\mathbf{x}^*$ *is a Nash equilibrium if*

$$x_i^* > 0 \Longrightarrow F_i(\mathbf{x}) > F_j(\mathbf{x}) \quad \forall\, j \in \mathcal{V}.$$

Nash equilibria always exists as affirmed in Theorem 3.3.1 by [67].

Theorem 3.3.1. *Every population game admits at least one Nash equilibrium.*

[2]Agents are rational in the sense they make decisions pursuing an improvement of their benefits.

The existence of equilibria could not be sufficient because these results can be only locals. It is necessary to look for global convergence properties, which establish that equilibrium is achieved starting from any initial state. Potential games and stable games are classes of dynamics that converge to equilibrium from all initial conditions [67].

Definition 3.3.2. *Let* $\mathbf{F} : \mathbb{R}^n_+ \mapsto \mathbb{R}^n$ *be a population game, if there exists a continuously differentiable potential function* $f : \mathbb{R}^n_+ \mapsto \mathbb{R}$ *that satisfies* $\nabla f(\mathbf{x}) = \mathbf{F}(\mathbf{x})$*, for all* $\mathbf{x} \in \mathbb{R}^n_+$*, then* $\mathbf{F}$ *is a full potential game.*

In potential games, all information about payoffs can be captured in the single scalar-valued function f.

Consider the population state $\mathbf{x} \in X$ such that $F_j(\mathbf{x}) > F_i(\mathbf{x})$. This assumption means that for the agents would be better choosing strategy $j \in \mathcal{V}$ rather than strategy $i \in \mathcal{V}$. If a small subset of agents switch from strategy i to strategy j, the impact that these switches have on the value of potential is equal to

$$\frac{\partial f}{\partial x_j} - \frac{\partial f}{\partial x_i} = F_j(\mathbf{x}) - F_i(\mathbf{x}) > 0.$$

This means that switching to a advantageous strategy increase the potential.

Given this, can be shown that Nash equilibria of full potential games correspond to the local maxima of the potential. Consider the nonlinear constrained optimization problem

$$\max f(\mathbf{x}) \quad \text{s.t.} \quad \sum_{i \in \mathcal{V}} x_i = m,\ x_i \geq 0, \quad \forall\, i \in \mathcal{V}$$

with associated Lagrangian with the following form:

$$\mathcal{L}(\mathbf{x}, \mu, \boldsymbol{\lambda}) = f(\mathbf{x}) + \mu \left(m - \sum_{i \in \mathcal{V}} x_i \right) + \sum_{i \in \mathcal{V}} \lambda_i x_i, \quad \boldsymbol{\lambda} \in \mathbb{R}^n,\ \mu \in \mathbb{R}. \tag{3.2}$$

The necessary Kuhn-Tucker first-order conditions [46] related to (3.2) can be derived and are

$$\frac{\partial f}{\partial x_i} = \mu - \lambda_i \quad \forall\, i \in \mathcal{V}. \tag{3.3}$$

$$\lambda_i x_i = 0 \quad \forall\, i \in \mathcal{V}, \tag{3.4}$$

$$\lambda_i \geq 0 \quad \forall\, i \in \mathcal{V}. \tag{3.5}$$

Definition 3.3.3. *The Kuhn-Tucker first order conditions on* $f(\mathbf{x})$ *under* μ *and* $\boldsymbol{\lambda}$ *are*

$$KT(f) = \{\mathbf{x} \in X : (\mathbf{x}, \mu, \boldsymbol{\lambda}) \text{ satisfies } (3.3) - (3.5) \text{ for some } \boldsymbol{\lambda} \in \mathbb{R}^n \text{ and } \mu \in \mathbb{R}\}.$$

The multiplier μ represents the equilibrium payoff in population, and $\boldsymbol{\lambda}$ the "payoff slack" of strategy $i \in \mathcal{V}$.

The validity of the Kuhn-Tucker conditions is necessary for local maximization of the full potential function. Considering that a continuous function on a compact set achieves its maximum, Theorem 3.3.2 proves that exists the Nash equilibrium of the full potential games.

The Kuhn-Tucker conditions are necessary but not sufficient for maximizing potential in X, therefore, can exists Nash equilibria that do not maximize f. On the other hand, if the full potential function f is concave, the Kuhn-Tucker conditions are not only necessary but also sufficient condition for maximizing f.

The following theorem shows that the Kuhn-Tucker first-order conditions for maximizing f on X characterize the Nash equilibria of $\mathbf{F}$.

Theorem 3.3.2. *If* $\mathbf{F}$ *is a full potential game with full potential function* f*, then* $NE(\mathbf{F}) = KT(f)$*.*

This means that satisfying the Kuhn-Tucker conditions for f on X is equivalent to maximizing the linearised version of f on X.

Definition 3.3.4. *The population game* $\mathbf{F} : X \mapsto \mathbb{R}^n$ *is a stable game if:* $(\mathbf{y} - \mathbf{x})^{\mathsf{T}}(\mathbf{F}(\mathbf{y}) - \mathbf{F}(\mathbf{x})) \leq 0$ *for all* $\mathbf{x}, \mathbf{y} \in X$*.*

In this case, if $\mathbf{F} \equiv \nabla f(\mathbf{x})$ is also a potential, Definition 3.3.4. is the requirement that the potential function f be concave [67].

3.3.2 Revision protocol and mean dynamics

Population games involve large numbers of agents, and the equilibrium assumption is quite strong. Therefore, is necessary to relax this requirement and suppose that agents gradually adjust their choices inside the available strategy set in order to generate trajectories that converge to Nash equilibrium.

If a population game $\mathbf{F}$ describes a strategic environment; a revision protocol denoted by $\boldsymbol{\rho}$ describes the procedures that agents follow in order to adapt their behaviour in the environment. $\mathbf{F}$ combined with $\boldsymbol{\rho}$ defines an evolutionary process, which is represented by the population dynamics [67].

Definition 3.3.5. *The revision protocol is a function* $\boldsymbol{\rho} : \mathbb{R}^n \times X \mapsto \mathbb{R}_+^{n\times n}$ *that describes the timing and the result of the decisions of agents in the strategic interaction. The revision protocol takes as inputs the payoff vector* $\mathbf{F}(\mathbf{x})$ *and a population state* $\mathbf{x} \in X$*, and returns as output a non–negative matrix, whose element of the* i*-th row and* j*-th column* $\rho_{ij}(\mathbf{F}(\mathbf{x}); \mathbf{x})$ *represents the conditional switch rate from strategy* $i \in \mathcal{V}$ *to strategy* $j \in \mathcal{V}$*.*

Depending on the revision protocol used by the agents, different kinds of population dynamics can be defined, e.g., replicator dynamics [45], and projection dynamics [57].

According to [67], consider a population composed by N agents receiving a revision opportunity given by an exponential distribution with rate R. Then, the revision opportunity received by each agent during a small interval of time dt is given by Rdt. When the current state is $\mathbf{x}$, the number of revision opportunities received by agents playing strategy i in this interval is equal to Nx_iRdt, because the value of x_i is considered constant during time interval $[0, dt]$ if dt is small.

Agents selecting strategy i switches to strategy j with probability ρ_{ij}/R, and the expected number of such switches during the next interval of length dt is $Nx_i\rho_{ij}dt$. The expected change that involve strategy i during the next interval time of length dt is

$$N\left(\sum_{j\in\mathcal{V}} x_j\rho_{ij} - x_i\sum_{j\in\mathcal{V}}\rho_{ij}\right)dt, \quad \forall i \in \mathcal{V}, \tag{3.6}$$

The first term of (3.6) contain switches to strategy i from other strategies, and the second term captures switches from strategy i to other strategies. Dividing (3.6) by N, the expected change in the proportion of agents choosing strategy i is obtained. The differential equation for the social state x_i is given deriving (3.6), i.e.,

$$\dot{x}_i = \sum_{j\in\mathcal{V}} x_j\rho_{ij} - x_i\sum_{j\in\mathcal{V}}\rho_{ij}, \quad \forall i \in \mathcal{V}, \tag{3.7}$$

and the differential equation in (3.7) represents the mean dynamics corresponding to revision protocol $\boldsymbol{\rho}$. Denote the mean dynamics of a population as $\dot{\mathbf{x}} = \mathbf{V}^{\mathbf{F}}(\mathbf{x})$.

Before defining the relationship between revision protocol and Nash equilibria, let introduce four desiderata for revision protocols

(C) Continuity: $\boldsymbol{\rho}$ is Lipschitz continuous[3].
(SC) Scarcity of data: ρ_{ij} only depends on F_i, F_j, and x_j.
(NS) Nash stationarity: $\mathbf{V}^{\mathbf{F}}(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} \in NE(\mathbf{F})$.
(PC) Positive correlation: $\mathbf{V}^{\mathbf{F}}(\mathbf{x}) \neq 0 \Rightarrow \mathbf{V}^{\mathbf{F}}(\mathbf{x})^{\top}\mathbf{F}(\mathbf{x}) > 0$.

As a consequence of (C), small changes in aggregate behaviour do not cause large changes in responses of the agents. (SD) means that the switch rate from strategy $i \in \mathcal{V}$ to strategy $j \in \mathcal{V}$ only depends on the payoffs of these two strategies and on the state of strategy j. Accordingly to (NS), the equilibrium points of the mean dynamics corresponds the Nash equilibria of the game, as affirmed in Proposition 3.3.3. Finally, (PC) requires that in all the other points the mean dynamics be positively correlated with the payoffs.

Proposition 3.3.3. *If* $\mathbf{V}^{\mathbf{F}}(\mathbf{x})$ *satisfies (PC), then* $\mathrm{x} \in NE(\mathrm{F})$ *implies that* $\mathbf{V}^{\mathbf{F}}(\mathbf{x}) = \mathbf{0}$.

3.3.2.1 Distributed mean dynamics

The populations described so far consider well–mixed[4] populations described by a complete graph, whereas non–well–mixed populations are the ones represented with non–complete graphs [6]. The concept introduced so far regarding population dynamics can be adapted to non–well–mixed populations in order to consider systems with distributed information-exchange structures.

As seen before, interactions among agents selecting different strategies can be represented by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, that may be interpreted as a population structure, where the set of nodes $\mathcal{V}$ is associated with the available strategies and the set of links $\mathcal{E}$ represents the possible interaction among agents selecting certain strategies. More precisely, the elements of the adjacency matrix $\mathbf{A} = [a_{ij}]$ are $a_{ij} = 1$ if strategies i and j can encounter each other, while $a_{ij} = 0$ if not.

In non–well–mixed populations should be taken in consideration the fact that only neighbouring strategies can interact. For this reason the switch rate ρ_{ij} from strategy

[4]In well-mixed populations any pair of agents playing any pair of strategies can interact with each other.

i to strategy j is pre-multiplied by a_{ij} to indicates that can happen only if i and j are neighbours in the graph. Then, the probability with which agents selecting strategy $i \in \mathcal{V}$ switch to strategy $j \in \mathcal{V}$ becomes $a_{ij}\rho_{ij}/R$. Finally, the expected number of agents switching from strategy $i \in \mathcal{V}$ to strategy $j \in \mathcal{V}$ during time dt is $Nx_i a_{ij}\rho_{ij}dt$.

The expected change in the use of strategy i during the next dt time units become

$$N\left(\sum_{j\in\mathcal{V}} x_j a_{ij}\rho_{ij} - x_i \sum_{j\in\mathcal{V}} a_{ij}\rho_{ij}\right) dt, \quad \forall i \in \mathcal{V}, \tag{3.8}$$

and the differential equation for the social state that corresponds to the distributed mean dynamics is

$$\dot{x}_i = \sum_{j\in\mathcal{N}_i} x_j\rho_{ij} - x_i \sum_{j\in\mathcal{N}_i} \rho_{ij}, \quad \forall i \in \mathcal{V}. \tag{3.9}$$

3.3.2.2 Projection dynamics

The projection dynamics are deduced from the mean dynamics (3.7) and using the modified pairwise comparison revision protocol

$$\rho_{ij} = \frac{[F_j - F_i]_+}{x_i}, \tag{3.10}$$

with $[\cdot]_+ := \max\{0, \cdot\}$.

Substituting (3.10) in (3.7), the projection dynamics equation is obtained

$$\begin{aligned}
\dot{x}_i &= \sum_{j\in\mathcal{V}} x_j \frac{[F_i(\mathbf{x}) - F_j(\mathbf{x})]_+}{x_j} - x_i \sum_{j\in\mathcal{V}} \frac{[F_j(\mathbf{x}) - F_i(\mathbf{x})]_+}{x_i} \\
&= \sum_{j\in\mathcal{V}} (F_i(\mathbf{x}) - F_j(\mathbf{x})) \\
&= nF_i(\mathbf{x}) - \sum_{j\in\mathcal{V}} F_j(\mathbf{x}), \quad \forall\, i \in \mathcal{V}.
\end{aligned} \tag{3.11}$$

Distributed projection dynamics (DPD) are deduced from the distributed mean dynamics (3.9) and revision protocol (3.10)

$$\dot{x}_i = |\mathcal{N}_i| F_i(\mathbf{x}) - \sum_{j\in\mathcal{N}_i} F_j(\mathbf{x}), \quad \forall\, i \in \mathcal{V}, \tag{3.12}$$

where $|\mathcal{N}_i|$ denotes the cardinality of the set $\mathcal{N}_i$, which is the number of neighbours of i-th node. The equilibrium points in (3.15) are given by $F_i = F_j$, for all $i, j \in \mathcal{V}$.

The following algorithms refer to a particular choice of fitness functions:

$$F_i(\mathbf{x}) = -x_i, \quad \forall\, i \in \mathcal{V}. \tag{3.13}$$

In a graph, the strategies are represented by the nodes. Denoting the scalar state of strategy i as $x_i \in \mathbb{R}_+$ for all $i \in \mathcal{V}$, the rate of change of each state is assumed to be governed by the sum of its relative states with respect to a subset of neighbours.

The differential equation that represents these dynamics is

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)), \quad \forall\, i \in \mathcal{V}. \tag{3.14}$$

Th expression in (3.14) has been obtained by substituting (3.13) in (3.15).

The equilibrium point of (3.14) is obtained when $\dot{x}_i = 0$ for all $i \in \mathcal{V}$ and this happen when $x_i = x_j$ for for all $i, j \in \mathcal{V}$. The fact that the equilibrium is reached when all the state component converge to a common value solves the problem agreement. In other words, if the states are used to represent the coordinates of mobile agents, with (3.14) the convergence to a common position in the space is obtained. This is called rendezvous problem.

3.3.2.2.1 Rendezvous with projection dynamics

Considering the practical application of these algorithms, mobile robots can be seen as agents that compose a multi-agent system. The network that describes the system can be represented with a graph where each agent occupies a node and the links represent the connections among agents.

If the communication between mobile robots has a limited range due to technological limitations, it is useful to introduce the concept of time-variant graph $\mathcal{G}(t) = \{\mathcal{V}, \mathcal{E}(t), \mathbf{A}(t)\}$. In particular if the presence of a link between two nodes is determined by the distance between the corresponding agents, the graph is a Δ-disk proximity graph [28], where

$$(i, j) \in \mathcal{E}(t) \iff \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| \leq \Delta, \quad \forall\, i, j \in \mathcal{V}, t \geq 0.$$

Here, $\Delta \in \mathbb{R}_{++}$ is a positive scalar value that represents the threshold distance among two agents to guarantee the establishment of a connection link among them.

Such graphs are dynamic in nature, as edges may appear or disappear when agents move in or out of the communication range of each other. The p-dimensional position of agent $i \in \mathcal{V}$ is given by

$$\mathbf{x}_i(t) = [x_i^1(t) \quad \dots \quad x_i^p(t)]^\mathsf{T}, \quad \forall\, i \in \mathcal{V},$$

thus, the overall state becomes equal to $\mathbf{x}(t) = [\mathbf{x}_1(t)^\mathsf{T} \quad \dots \quad \mathbf{x}_n(t)^\mathsf{T}]^\mathsf{T}$.

The control goal is to achieve the position agreement, i.e., to make the states agents reach a common value in a non-centralised manner[5], taking the geometric range constraints into account in an explicit way.

Applying the considerations with respect to the vector positions, the distributed projection dynamics (3.14) assumes the form

$$\dot{x}_i^k(t) = \sum_{j \in \mathcal{N}_i} (x_j^k(t) - x_i^k(t)), \quad \forall\, i \in \mathcal{V}, \text{ and } k = 1, \dots, p. \tag{3.15}$$

The dynamical equation in (3.15) can be applied to solve problems of agents aggregation because its equilibrium point corresponds to the agreement position that the agents should reach. The following simulations give a practical example of this affirmation.

3.3.2.2.1.1 Rendezvous simulations with projection dynamics [6]

The algorithm just presented has been implemented with MatLab. In the simulations the number of agents involved is arbitrarily selected to be to $n = 6$ and the environment is represented by a square area with side $L = 50\ m$. In the first case $\Delta = 20\ m$ and in the second $\Delta = 30\ m$ in order to generate two different scenarios and compare the results. Both these choices of Δ guarantee the connectivity of the graph $\mathcal{G}$, since the initial conditions are selected in a manner such that there are no isolated nodes, as presented in Figure 3.3.

[5] Non-centralised since it is desired that each agent only has partial information about other agents.

[6] The following simulations and all the others presented in the document refers to the bi-dimensional case. In the previous theoretical part the state or position of agent i was expressed with the p-dimensional variable $\mathbf{x}_i$. To maintain a consistent notation it should have been chose $\mathbf{x}_i = [x_i^1 \ x_i^2]^\mathsf{T}$, $i = 1, \dots, n$, but to reduce the complexity of the indices, the position of each agent is expressed with $[x_i \ y_i]^\mathsf{T}$ where $x_i \equiv x_i^1$ and $y_i \equiv x_i^2$. Moreover, $\mathbf{x} = [x_1 \quad \dots \quad x_n]^\mathsf{T}$ and $\mathbf{y} = [y_1 \quad \dots \quad y_n]^\mathsf{T}$.

Figure 3.3 presents the initial positions of the agents in the space and the red arcs are the interconnections determined by distance Δ. In Figure 3.3(a), $\Delta = 20\ m$ and the adjacency matrix related to the graph is sparse. Instead, in Figure 3.3(b), $\Delta = 30\ m$ and the graph has a bigger number of links.

When there are more links in the graph, then the velocity of convergence of the agents to the equilibrium in higher. This can be seen in Figures 3.4 and 3.5 where the agents need more time to reach the common point when they are less connected to each other.

Different evolutions in time of the coordinates generate different trajectories represented in Figure 3.6, starting from the initial positions to the final destination, which is the equilibrium point.

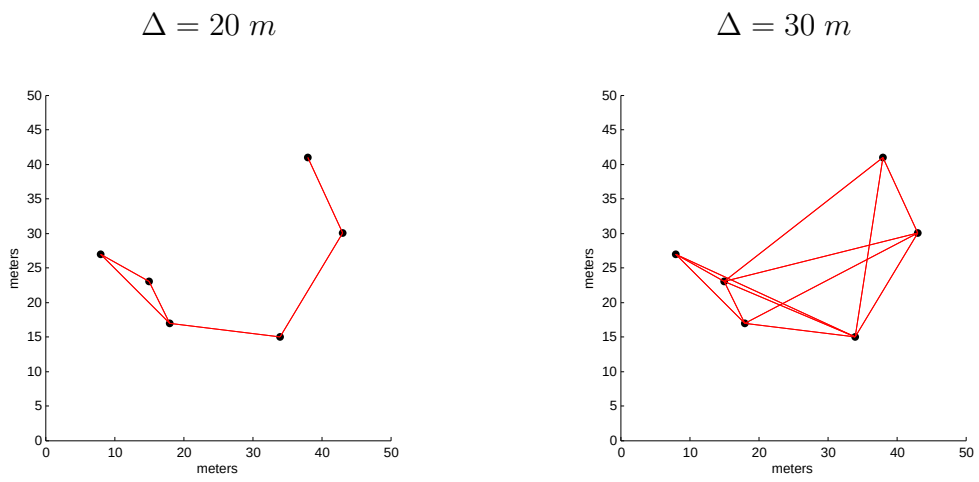


Figure 3.3: DPD: graph of interconnections among agents.

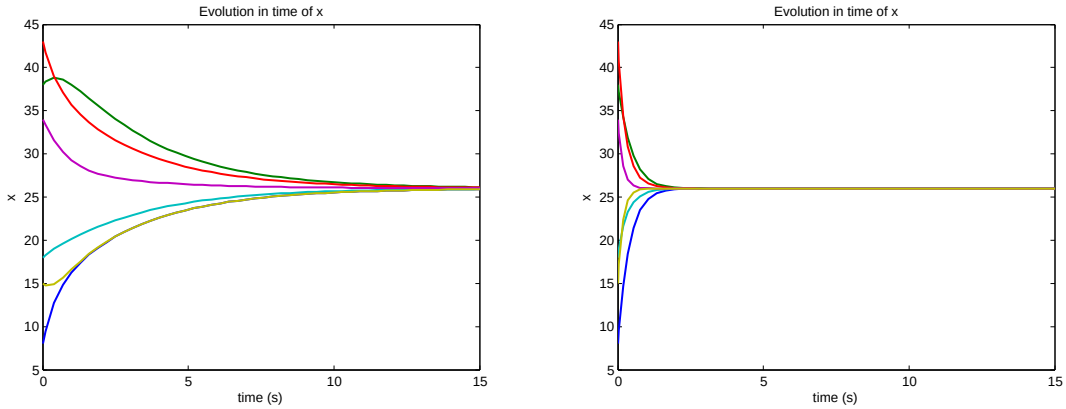


Figure 3.4: DPD: evolution of x coordinate.

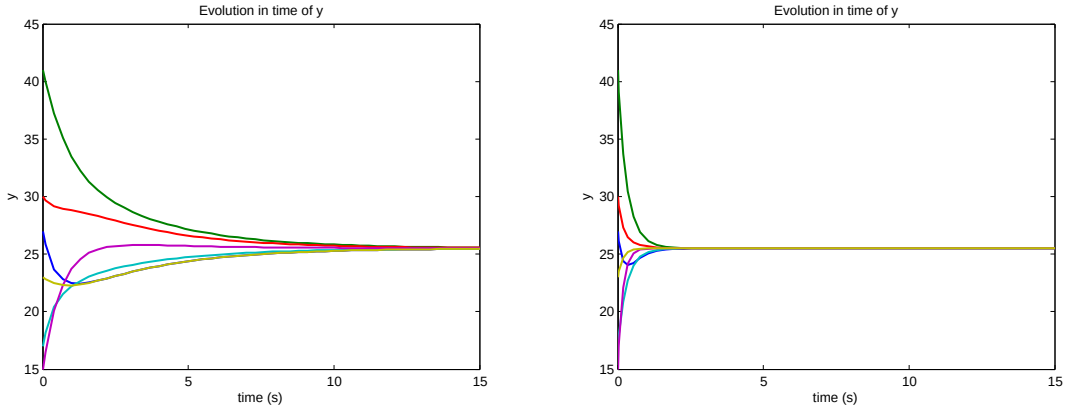


Figure 3.5: DPD: evolution of y coordinate.

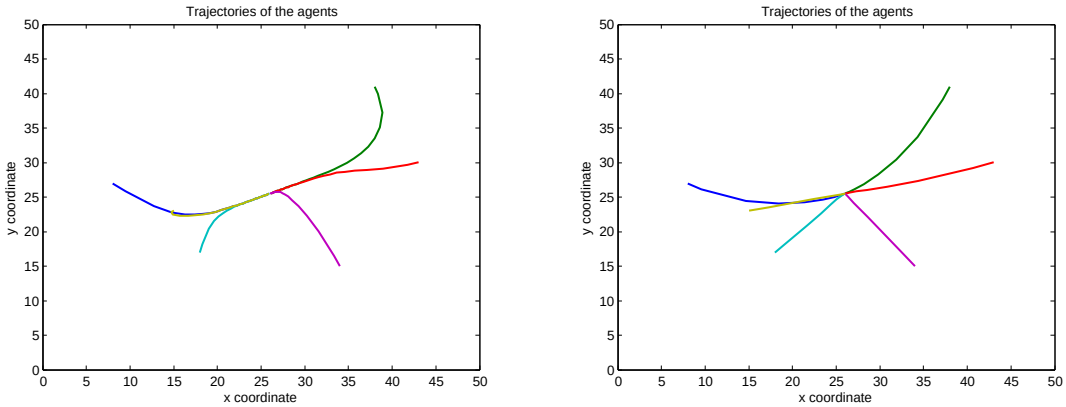


Figure 3.6: DPD: trajectories of the agents.

3.3.2.2.2 Formation with projection dynamics

Now consider a procedure to solve the distributed formation control problem. The inter-agent distance constraints can be described by a connected edge-labelled graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{E}_d, \mathbf{A}_d\}$, where the subscript d means "desired", $\mathcal{E}_d$ encodes the desired robot interconnections and the edge labels d define the desired relative inter-agent displacements, with $\|\mathbf{d}_{ij}\| < \Delta$, for all $i, j \in \mathcal{V}$, $\mathbf{d}_{ij} \in \mathbb{R}^p$, such that $(i, j) \in \mathcal{E}_d$. The graph $\mathcal{G}_d$ is the one that is established at the end of the motion of the agents, when they all reach the desired formation determined by the inter-agent displacements $\mathbf{d}_{ij} = [d_{ij}^1 \quad \dots d_{ij}^p]^\mathsf{T}$.

The goal of the distributed formation control is to find the dynamical equations that respect the following conditions

1. The dynamic interaction graph $\mathcal{G}(t)$ converges to a graph that is a subgraph of the desired graph $\mathcal{G}_d$ in finite time: $\mathcal{E}_d \subseteq \mathcal{E}(t)$, $\forall\, 0 \leq t < \infty$.
2. The pairwise distances $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|$ converge asymptotically to $\|\mathbf{d}_{ij}\|$ for all $i, j \in \mathcal{V}$ such that $(i, j) \in \mathcal{E}_d$.
3. Each equation utilises only local information.

The modified version of (3.15) to perform formation is

$$\dot{x}_i^k(t) = \sum_{j \in \mathcal{N}_i} (x_j^k(t) + d_{ij}^k - x_i^k(t)), \quad \forall\, i \in \mathcal{V}, \text{ and } k = 1, \dots, p. \tag{3.16}$$

3.3.2.2.2.1 Formation simulations with projection dynamics

It follows a MatLab simulation that performs the distributed control problem with the particular task of obtaining a triangular formation. It is selected $\Delta = 30\ m$ and the initial condition for the positions of agents are chosen inside a square area of side $L = 50\ m$. Agents involved are $n = 6$ and the value of the side of the triangle has been fixed equal to $\ell = 15\ m$. Also the final positions that constitute the triangular formation are inside the square area.

In order to reach the formation represented in Figure 3.7, before applying the algorithm, it is necessary to assign the desired mutual distances $\mathbf{d}_{ij} = [d_{ij}^{\mathbf{x}} \quad d_{ij}^{\mathbf{y}}]^\mathsf{T}$ that the agents should have at the end of the motion. These values are listed in Tables 3.1 and 3.2 and they refer to inter-agents distances along x and y directions.

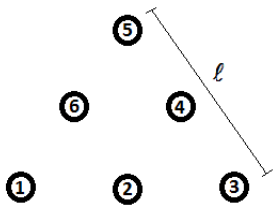


Figure 3.7: Triangular formation to be obtained with distributed consensus control.

Table 3.1: Distances along x direction.

$d_{ij}^{\mathbf{x}} = x_i - x_j$						
agents	**1**	**2**	**3**	**4**	**5**	**6**
1	0	$-\frac{\ell}{2}$	$-\ell$	$-\frac{3\ell}{4}$	$-\frac{\ell}{2}$	$-\frac{\ell}{4}$
2	$\frac{\ell}{2}$	0	$-\frac{\ell}{2}$	$-\frac{\ell}{4}$	0	$\frac{\ell}{4}$
3	ℓ	$\frac{\ell}{2}$	0	$\frac{\ell}{4}$	$\frac{\ell}{2}$	$\frac{3\ell}{4}$
4	$\frac{3\ell}{4}$	$\frac{\ell}{4}$	$-\frac{\ell}{4}$	0	$\frac{\ell}{4}$	$\frac{\ell}{2}$
5	$\frac{\ell}{2}$	0	$-\frac{\ell}{2}$	$-\frac{\ell}{4}$	0	$\frac{\ell}{4}$
6	$\frac{\ell}{4}$	$-\frac{\ell}{4}$	$-\frac{3\ell}{4}$	$-\frac{\ell}{2}$	$-\frac{\ell}{4}$	0

Table 3.2: Distances along y direction.

$d_{ij}^{\mathbf{y}} = y_i - y_j$						
agents	**1**	**2**	**3**	**4**	**5**	**6**
1	0	0	0	$-\frac{\sqrt{3}\ell}{4}$	$-\frac{\sqrt{3}\ell}{2}$	$-\frac{\sqrt{3}\ell}{4}$
2	0	0	0	$-\frac{\sqrt{3}\ell}{4}$	$-\frac{\sqrt{3}\ell}{2}$	$-\frac{\sqrt{3}\ell}{4}$
3	0	0	0	$-\frac{\sqrt{3}\ell}{4}$	$-\frac{\sqrt{3}\ell}{2}$	$-\frac{\sqrt{3}\ell}{4}$
4	$\frac{\sqrt{3}\ell}{4}$	$\frac{\sqrt{3}\ell}{4}$	$\frac{\sqrt{3}\ell}{4}$	0	$-\frac{\sqrt{3}\ell}{4}$	0
5	$\frac{\sqrt{3}\ell}{2}$	$\frac{\sqrt{3}\ell}{2}$	$\frac{\sqrt{3}\ell}{2}$	$\frac{\sqrt{3}\ell}{4}$	0	$\frac{\sqrt{3}\ell}{4}$
6	$\frac{\sqrt{3}\ell}{4}$	$\frac{\sqrt{3}\ell}{4}$	$\frac{\sqrt{3}\ell}{4}$	0	$-\frac{\sqrt{3}\ell}{4}$	0

Figure 3.8 shows how the coordinates converge to the final desired values. Finally, in Figure 3.9 the trajectories of the agents are represented and the triangular formation that they perform is highlighted.

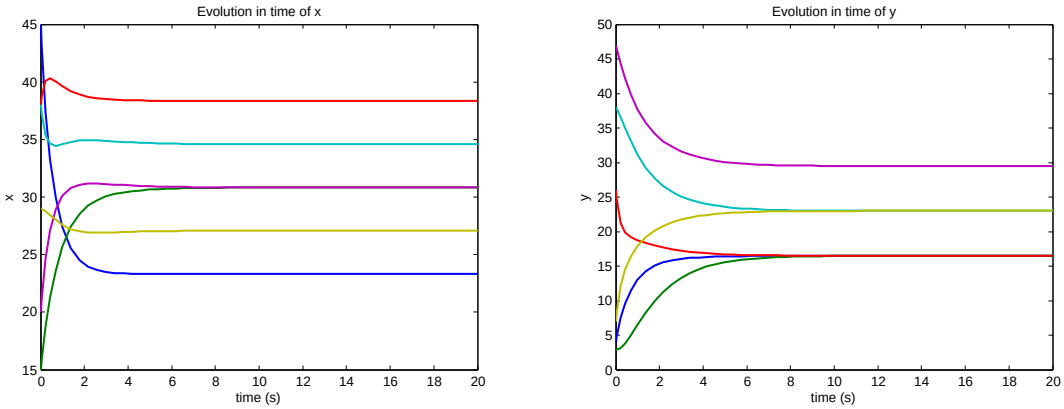


Figure 3.8: Evolution in time of the coordinates with distributed formation control.

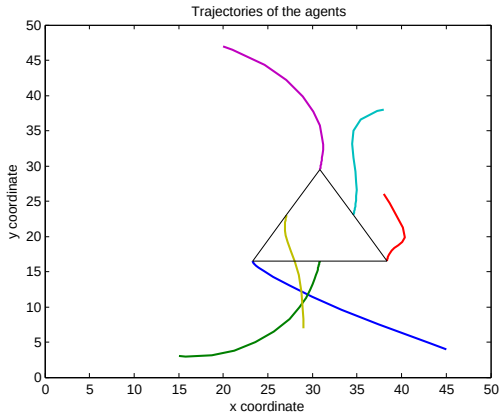


Figure 3.9: Trajectories of the agents with distributed formation control.

3.3.2.3 Replicator dynamics

The replicator dynamic are deduced from the mean dynamics (3.7) and using the pairwise proportional imitation protocol

$$\rho_{ij} = x_i[F_j - F_i]_+, \tag{3.17}$$

Substituting (3.17) in (3.7) the replicator dynamics equation is obtained

$$\begin{aligned}\dot{x}_i &= \sum_{j\in\mathcal{V}} x_j x_i [F_i(\mathbf{x}) - F_j(\mathbf{x})]_+ - x_i \sum_{j\in\mathcal{V}} x_j [F_j(\mathbf{x}) - F_i(\mathbf{x})]_+ \\ &= \sum_{j\in\mathcal{V}} x_j x_i (F_i(\mathbf{x}) - F_j(\mathbf{x})) \\ &= x_i \left(m F_i(\mathbf{x}) - \sum_{j=1}^{n} F_j(\mathbf{x}) x_j \right), \quad \forall\, i \in \mathcal{V}\end{aligned} \tag{3.18}$$

Similarly as with the projection dynamics, if the considered population is non-well-mixed, it is necessary to adopt a distributed version of (3.18).

The distributed replicator dynamics (DRD) are deduced from the distributed mean dynamics (3.9) and revision protocol (3.17)

$$\dot{x}_i = x_i \left(F_i(\mathbf{x}) \sum_{j\in\mathcal{N}_i} x_j - \sum_{j\in\mathcal{N}_i} x_j F_j(\mathbf{x}) \right), \quad \forall\, i \in \mathcal{V}. \tag{3.19}$$

As with projection dynamics, the equilibrium point of (3.19) is obtained when $x_i = x_j$ for for all $i, j \in \mathcal{V}$. The rendezvous problem can be solved also with replicator dynamics and the common position to which agents converge corresponds to the Nash equilibrium.

3.3.2.3.1 Rendezvous with replicator dynamics

It follows an example where potential games $F_i(\mathbf{x})$ and $F_i(\mathbf{y})$ describe distinctly the evolutionary behaviour of coordinates of the agents $\mathbf{x} = [x_1 \quad \dots \quad x_n]^\mathsf{T}$ and $\mathbf{y} = [y_1 \quad \dots \quad y_n]^\mathsf{T}$ in the bi-dimensional space.

The potential games are obtained from the expression of the potential function $f(\mathbf{x}, \mathbf{y})$

$$f(\mathbf{x}, \mathbf{y}) = -\sum_{i=1}^{n}\sum_{j=1}^{n}((x_j - x_i)^2 + (y_j - y_i)^2), \quad \forall\, i, j \in \mathcal{V}. \tag{3.20}$$

The expression of the potential function 3.20 is directly dependent to the Euclidean distances among agents, where the distance in the bi-dimensional case is given by $\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$.

By computing the partial derivatives of $f(\mathbf{x}, \mathbf{y})$ the fitness functions are obtained

$$\begin{array}{rcl} F_i(\mathbf{x}) & = & \dfrac{\partial f(\mathbf{x},\mathbf{y})}{\partial x_i} = 2\sum_{j=1}^{n}(x_j - x_i), \\ F_i(\mathbf{y}) & = & \dfrac{\partial f(\mathbf{x},\mathbf{y})}{\partial y_i} = 2\sum_{j=1}^{n}(y_j - y_i), \end{array} \quad \forall\, i \in \mathcal{V}. \tag{3.21}$$

Substituting the expressions of the fitness functions, the replicator equations are obtained as follows:

$$\dot{x}_i = x_i \left(F_i(\mathbf{x}) \sum_{j\in\mathcal{N}_i} x_j - \sum_{j\in\mathcal{N}_i} F_j(\mathbf{x}) x_j \right), \quad \forall\, i \in \mathcal{V},$$

$$\dot{y}_i = y_i \left(F_i(\mathbf{y}) \sum_{j\in\mathcal{N}_i} y_j - \sum_{j\in\mathcal{N}_i} F_j(\mathbf{y}) y_j \right), \quad \forall\, i \in \mathcal{V}.$$

3.3.2.3.1.1 Rendezvous simulations with replicator dynamics

Simulations have been made with MatLab and the number of agents has been fixed $n = 4$. As in the previous experiments, the environment is represented by a square area with side $L = 50\ m$.

The experiments are of two types: the first uses an algorithm based on time-invariant graph, more precisely the incidence matrix that represents the neighbouring relationships among agents is fixed and it depends only on the initial positions of the agents. If two agents are not neighbours before starting moving because their distance is higher than the threshold value $\Delta = 30\ m$, they will not become neighbours even if during the motion their distance decrease under the value of Δ.

On the other hand, the second experiment works with an algorithm that uses a time-variant incidence matrix that is redefined at each time instant. For this reason, agents can establish new neighbouring connections during the motion or existing interactions can blow up.

In Figure 3.10 the graph that describes the neighbouring relationships among agents when they are in the initial positions is represented. The graph is not complete and, as explained before, this is determined by the initial condition of the positions of the agents and by the value of Δ.

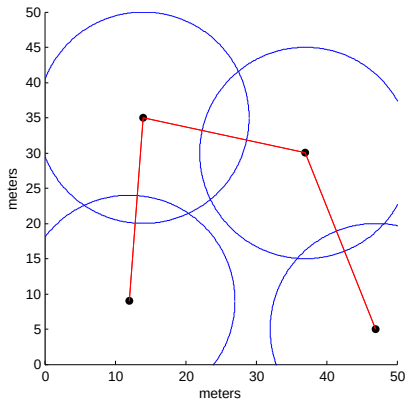


Figure 3.10: DRD rendezvous: incomplete graph representing initial agents relationships.

In Figures 3.11 and 3.12 the evolutions in time of the coordinates of the agents are compared. The convergence speed is considerably higher when the algorithm that uses a time-varying incidence matrix is applied if compared with the one based on time-invariant graph. Finally, Figure 3.13 represents the different trajectories that the agents follow in the two cases.

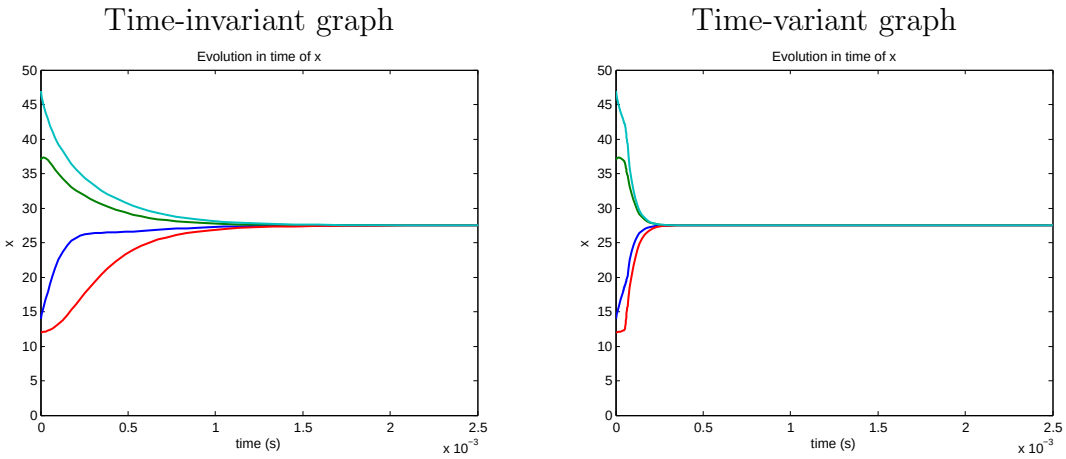


Figure 3.11: DRD rendezvous: evolution of x coordinate with incomplete initial graph.

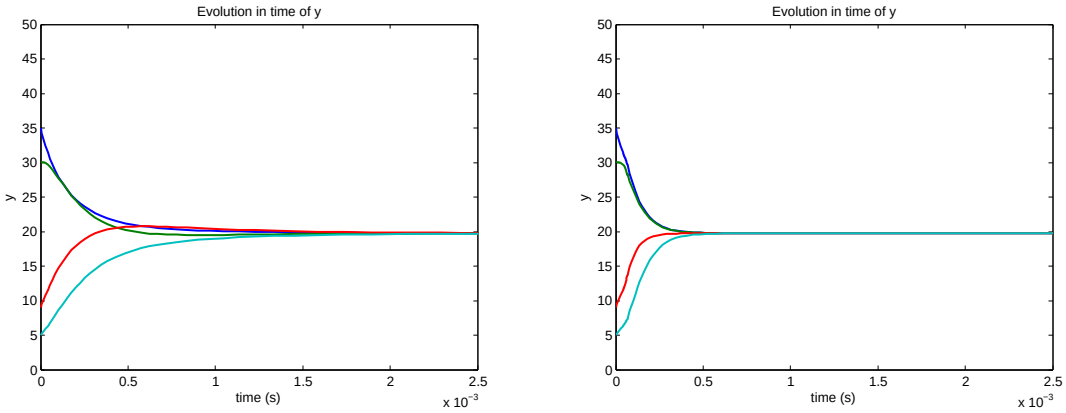


Figure 3.12: DRD rendezvous: evolution of y coordinate with incomplete initial graph.

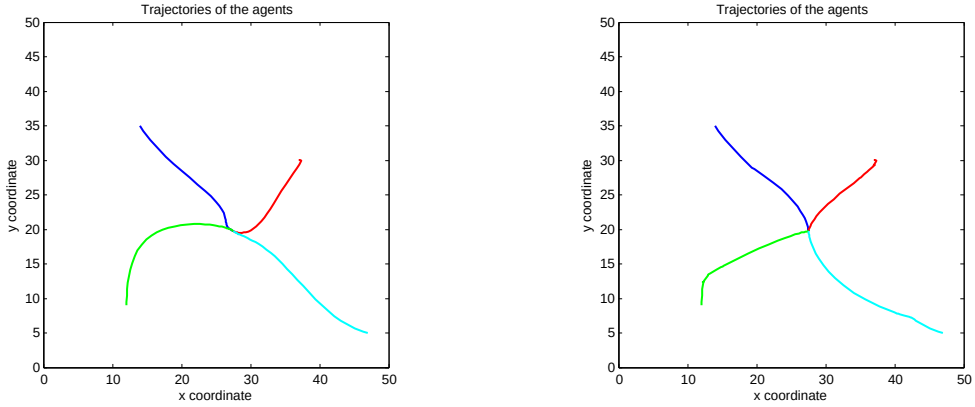


Figure 3.13: DRD rendezvous: trajectories of the agents with incomplete initial graph.

3.3.2.3.2 Formation with replicator dynamics

The problem of formation with multi-agent system can be also solved with a game theoretical approach applying the distributed replicator dynamics equation. To convert the rendezvous solution into formation task it is only necessary to modify the expression of the potential function f.

Starting from (3.20), the new expression of the potential function takes the following form:

$$f(\mathbf{x},\mathbf{y}) = -\sum_{i=1}^{n}\sum_{j=1}^{n}((x_j + d_{ij}^{\mathbf{x}} - x_i)^2 + (y_j + d_{ij}^{\mathbf{x}} - y_i)^2),$$

where $d_{ij}^{\mathbf{x}}$ and $d_{ij}^{\mathbf{y}}$ represent respectively the distances, with sign, along the axes that agent i should maintain from agent j when they reach the equilibrium, in order to perform the desired formation.

With the same steps carried out in the previous paragraph, the potential games can be obtained

$$F_i(\mathbf{x}) = \frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial x_i} = 2 \sum_{j=1}^{n} (x_j + d_{ij}^{\mathbf{x}} - x_i), \quad \forall\, i \in \mathcal{V},$$

$$F_i(\mathbf{y}) = \frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial y_i} = 2 \sum_{j=1}^{n} (y_j + d_{ij}^{\mathbf{y}} - y_i), \quad \forall\, i \in \mathcal{V},$$

which have to be substituted inside the replicator dynamics equation in (3.19).

3.3.2.3.2.1 Formation simulations with replicator dynamics

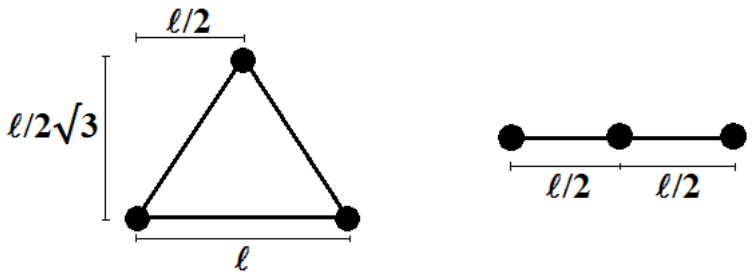


Figure 3.14: Triangle and segment formations.

In order to generate the triangular formation or linear formation represented in Figure 3.14, in the case of three agents, the values of the parameter $d_{ij}^{\mathbf{x}}$ and $d_{ij}^{\mathbf{y}}$ should be the ones indicated by the elements of the following matrices:

$$\text{Triangle}: \quad \mathbf{D_x} = [d_{ij}^{\mathbf{x}}] = \begin{bmatrix} 0 & -\frac{\ell}{2} & \frac{\ell}{2} \\ \frac{\ell}{2} & 0 & \ell \\ -\frac{\ell}{2} & -\ell & 0 \end{bmatrix}, \quad \mathbf{D_y} = [d_{ij}^{\mathbf{y}}] = \begin{bmatrix} 0 & -\frac{\sqrt{3}\ell}{2} & -\frac{\sqrt{3}\ell}{2} \\ \frac{\sqrt{3}\ell}{2} & 0 & 0 \\ \frac{\sqrt{3}\ell}{2} & 0 & 0 \end{bmatrix}.$$

$$\text{Segment}: \quad \mathbf{D_x} = [d_{ij}^{\mathbf{x}}] = \begin{bmatrix} 0 & \frac{\ell}{2} & \ell \\ -\frac{\ell}{2} & 0 & \frac{\ell}{2} \\ -\ell & -\frac{\ell}{2} & 0 \end{bmatrix}, \quad \mathbf{D_y} = [d_{ij}^{\mathbf{y}}] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

MatLab simulations, in which the number of agents has been fixed $n = 3$, are taken in the usual square environment with side $L = 50\ m$. The experiments use the algorithm based on time-variant graph with neighbouring relationship redefined at each instant depending on current distances among agents. The following results are obtained with threshold sensing distance equal to $\Delta = 30\ m$ and parameter $\ell = 20\ m$.

In Figure 3.15 the initial graph that describes the neighbouring relationships among agents at the beginning of the experiment is represented. Initially the graph is not complete but it is connected so the distance-based time-varying algorithm can be applied.

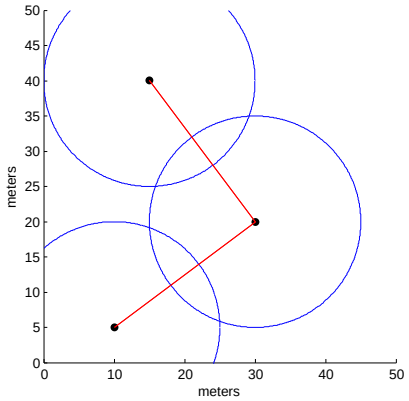


Figure 3.15: DRD formation: incomplete graph representing initial agents relationships.

Figure 3.16 represents the two different formation situation with the agents starting from the same initial positions. In Figure 3.17 and 3.18 are compared the evolutions in time of the coordinates of the agents.

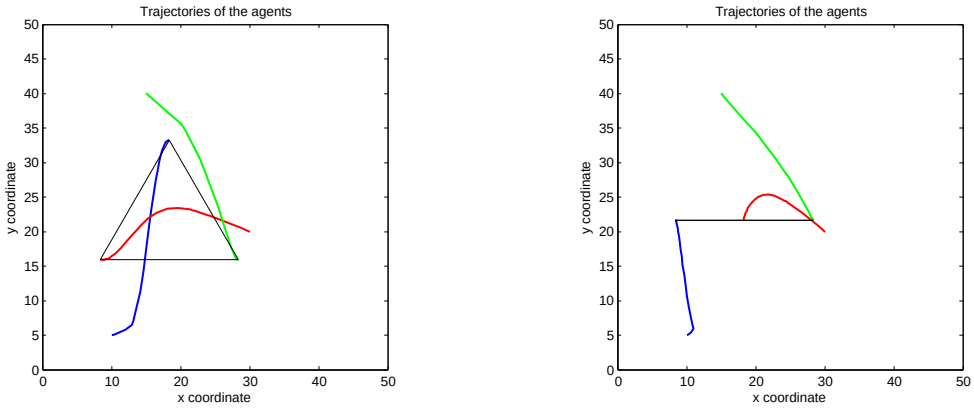


Figure 3.16: DRD formation: trajectories of the agents.

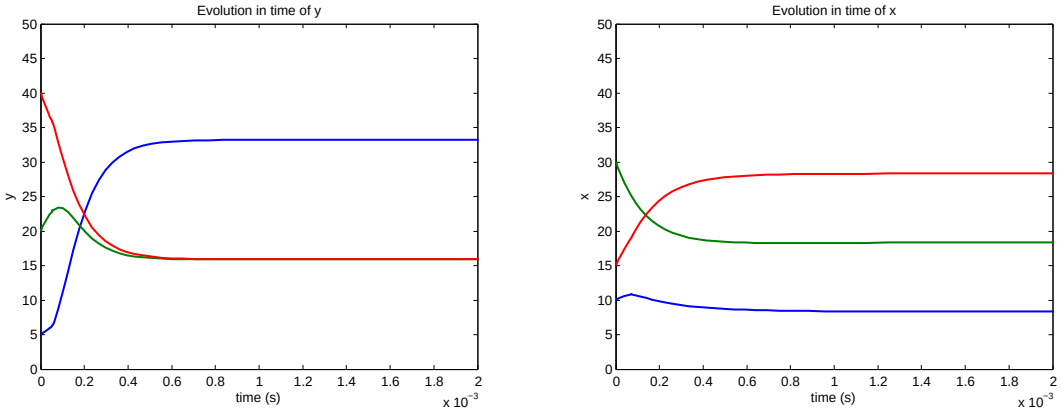


Figure 3.17: DRD formation: evolution of x coordinate.

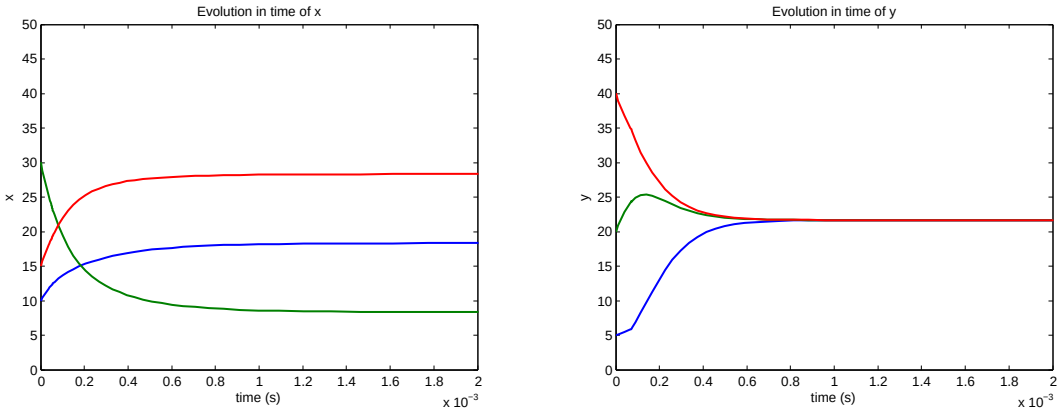


Figure 3.18: DRD formation: evolution of y coordinate.

3.3.3 Control of dynamical models

In this section dynamical models of two vehicles are taken in consideration in order to generate simulations that are closer to the real experiments that will be performed with the Mindstorms.

Ideal trajectories generated in the previous sections are established as references for local controllers that consider the dynamical behaviours of the agents.

The models mentioned above are the one of the unicycle and of the three-wheels mobile robot.

3.3.3.1 Unicycle model

The unicycle is the simplest model of a wheeled vehicle and its system is equivalent to the one of the shopping cart. The shopping cart has two independent motor drives at the rear wheels. The centre of mass is at centre of the axle.

Considering the unicycle moving in an x-y plane, denote with v its forward velocity and with θ the heading angle. The unicycle or shopping cart can move only in the direction it is heading. That is, there is a no-slip condition

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \perp \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix} \implies \dot{x}\sin\theta - \dot{y}\cos\theta = 0.$$

The last condition is called *nonholonomic constraint*.

In general, a constraint with the form $h(\mathbf{q}) = 0$, where $\mathbf{q}$ is the position vector is called holonomic. A constraint of the form $h(\mathbf{q})\dot{\mathbf{q}} = 0$, where $\dot{\mathbf{q}}$ is the velocity vector, cannot be integrated into a position constraint so it is called nonholonomic. Considering the unicycle

$$\dot{x}\sin\theta - \dot{y}\cos\theta = \begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0,$$

the kinematic model of the unicycle is the following:

$$\begin{cases} \dot{x} = v\cos\theta, \\ \dot{y} = v\sin\theta, \\ \dot{\theta} = \omega, \end{cases} \tag{3.22}$$

and defining the configuration vector as $\mathbf{q} := \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ and input vector $\mathbf{u} := \begin{bmatrix} v \\ \omega \end{bmatrix}$, the model takes the compact form $\dot{\mathbf{q}} = g(\mathbf{q}, \mathbf{u})$ where

$$g(\mathbf{q}, \mathbf{u}) = g(x, y, \theta; v, \omega) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix},$$

or equally

$$\dot{\mathbf{q}} = v g_1(\mathbf{q}) + \omega g_2(\mathbf{q}) = \begin{bmatrix} g_1(\mathbf{q}) & g_2(\mathbf{q}) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

where

$$g_1(\mathbf{q}) = f(x, y, \theta; 1, 0) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ 0 \end{bmatrix}, \qquad g_2(\mathbf{q}) = f(x, y, \theta; 0, 1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The complete unicycle model, linear in the input, is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ 0 \end{bmatrix} + \omega \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

3.3.3.1.0.2 Trajectory tracking with unicycle

It is necessary to create an outer loop that can control the motion of the robot in order to make it tracking a path. The input of the control is the desired trajectory $(x_d(t), y_d(t))$, while a negative feedback is obtained extracting the coordinates of the robot in terms of position in the space (x, y) and its orientation (θ) from the kinematic model given by (3.22).

To reach a destination point, it is sufficient to apply a proportional controller that controls the velocity of the robot to be proportional to its distance from the goal. Instead, when the vehicle has to follow a trajectory, the target can be considered as a moving point, therefore a simple proportional controller cannot be enough to satisfy the request. In order to lead the error of the velocity to zero, it is necessary use a PI controller [21].

The robot is assumed to maintain a distance δ behind the pursuit point that is moving along the reference trajectory and the error between the current position and this pursuit point is

$$e = \sqrt{(x_d - x)^2 + (y_d - y)^2} - \delta,$$

and it has to be regulated to zero with the PI controller given by

$$v^* = K_{vP}\, e + K_{vI} \int e\, dt.$$

The integral term is required to provide a finite velocity demand v^* when the error is zero.

The second controller steers the robot toward the target that is at the relative angle

$$\theta^* = \arctan\left(\frac{y_d - y}{x_d - x}\right),$$

and a simple proportional controller

$$\alpha = K_{\omega P}(\theta^* - \theta),$$

turns the steering wheel so as to drive the robot toward the target.

Here, θ is constrained to lie in the angular interval $[-\pi, \pi)$.

Figure 3.19 represents the complete modelling of the mobile robot and in Figures 3.20 and 3.21 are represented respectively the content of Position Control block and Kinematics block.

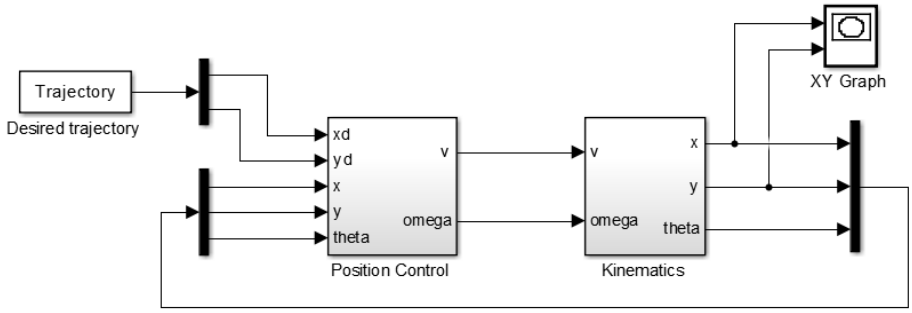


Figure 3.19: Unicycle: trajectory tracking.

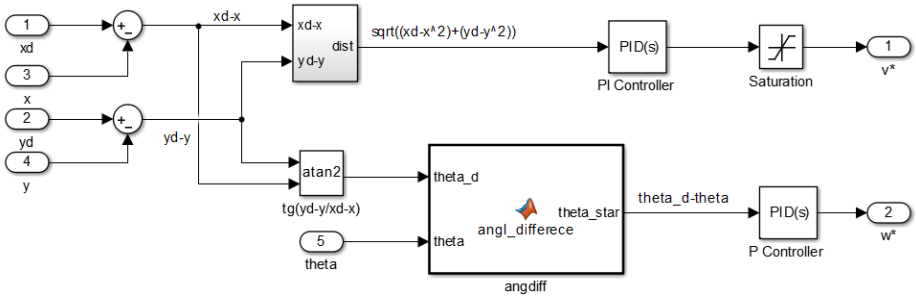


Figure 3.20: Unycycle: Position Control.

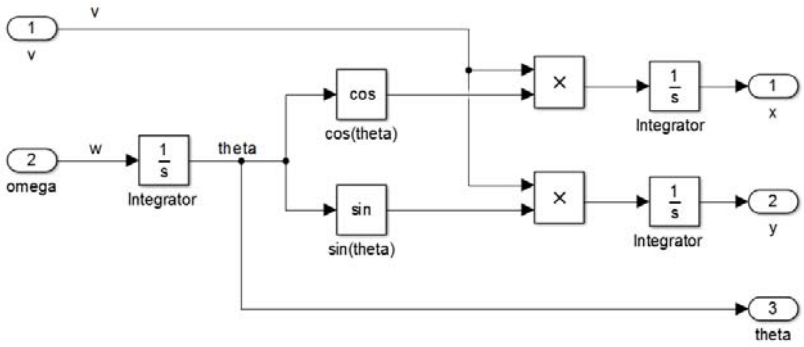


Figure 3.21: Unicycle: Kinematics.

In the following figures are put in comparison the behaviours of trajectories obtained with ideal point agents and the ones obtained with the dynamical model of the unicycle. The top figures represent the reference trajectories that the unicycles have to track. Instead, bottom images are actually the trajectories of the unicycles. In Figure 3.22 the references are obtained with projection dynamics equation, while the references in Figure 3.23 belongs to replicator dynamics.

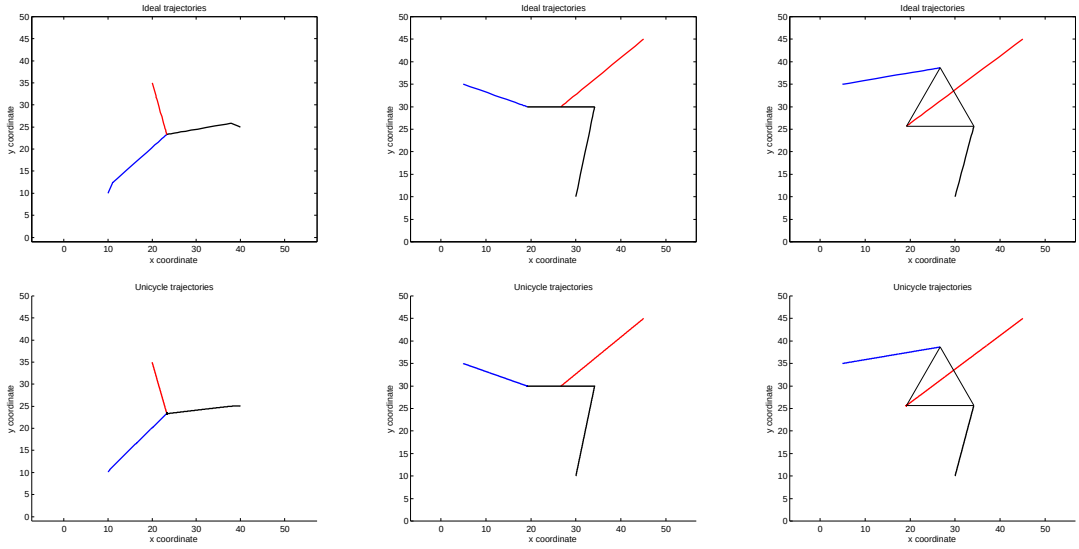


Figure 3.22: Unicycle: projection dynamics

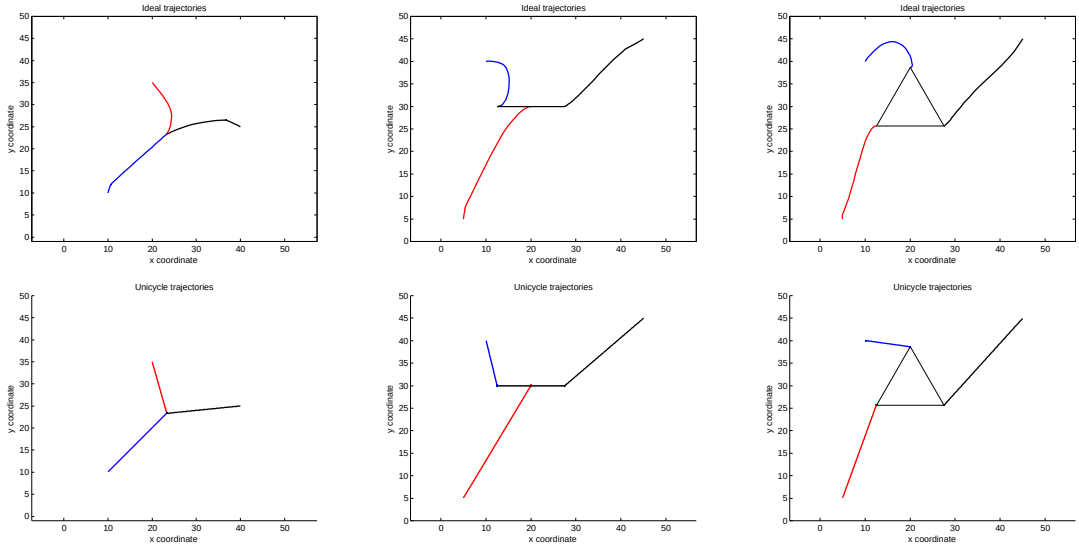


Figure 3.23: Unicycle: replicator dynamics

3.3.3.2 Mobile robot model

The kinematic equation of the mobile robot [21] are given by

$$\begin{cases} \dot{x} = \frac{1}{2}(v_L + v_R)\cos(\theta), \\ \dot{y} = \frac{1}{2}(v_L + v_R)\sin(\theta), \\ \dot{\theta} = \frac{1}{2d}(v_R - v_L), \end{cases} \tag{3.23}$$

where x, y, θ is the pose of the vehicle, e.g. position and orientation in the plane, v_R and v_L are the linear speed provided by the right and left wheel, while d is the length of the semi-axis of the vehicle.

The dynamic of the vehicle is

$$\begin{cases} M\dfrac{dv}{dt} = -K_v v + K_m V_m - B_v v, \\ J\dfrac{d\omega}{dt} = -K_\omega \omega + K_d V_d - B_\omega \omega, \end{cases} \tag{3.24}$$

where v, ω are the linear and angular velocities of the vehicle, M and J are the mass and the inertia of the vehicle, respectively. V_m and V_d are the average and differential voltages applied to the wheels. The constants K_v, K_m, K_ω, K_d map voltages, linear and angular velocities in forces and torques. here, B_v and B_ω are the translational and rotational friction coefficients, respectively. Finally, V_R and V_L are the voltages applied to the wheels and $V_m = \dfrac{V_R + V_L}{2}$ and $V_d = V_R - V_L$.

The relation among the linear velocities of the wheel and the linear and angular velocities of the vehicle is

$$\begin{cases} v_R = v + \omega d, \\ v_L = v - \omega d. \end{cases} \tag{3.25}$$

3.3.3.2.0.3 Control of the velocities of the vehicle

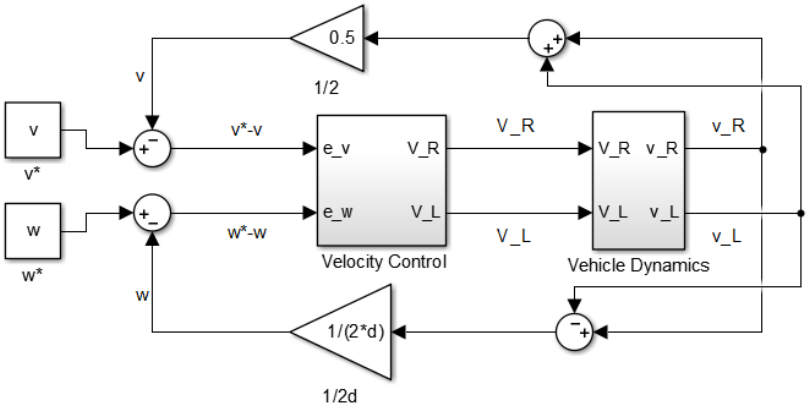


Figure 3.24: Mobile robot: velocity loop control.

The control has to be built knowing the linear velocities of the wheels. They can be computed using the equations

$$v_R = v + \omega d, \tag{3.26}$$

$$v_L = v - \omega d. \tag{3.27}$$

From the dynamic model of the vehicle can be derived the corresponding transfer functions in order to obtain the linear and angular velocities of the vehicle v and ω

$$\begin{cases} M\dfrac{dv}{dt} = -K_v v + K_m V_m - B_v v, \\ J\dfrac{d\omega}{dt} = -K_\omega \omega + K_d V_d - B_\omega \omega, \end{cases} \xrightarrow{\mathcal{L}} \begin{cases} MsV(s) = -K_v V(s) + K_m V_m - B_v V(s), \\ Js\Omega(s) = -K_\omega \Omega(s) + K_d V_d - B_\omega \Omega(s), \end{cases}$$

$$V(s) = \frac{K_m}{Ms + K_v + B_v} V_m, \tag{3.28}$$

$$\Omega(s) = \frac{K_d}{Js + K_\omega + B_\omega} V_d. \tag{3.29}$$

The construction of the velocity loop is based in the measurements of the velocities of the wheels so the linear and angular velocities of the vehicle have to be evaluated in this way:

$$v = \frac{v_R + v_L}{2}, \tag{3.30}$$

$$\omega = \frac{v_R - v_L}{2d}. \tag{3.31}$$

In the scheme in Figure 3.25 is represented the relationship between mean and differential voltages and voltages applied to the wheels

$$V_R = V_m + \frac{V_d}{2},$$

$$V_L = V_m - \frac{V_d}{2}.$$

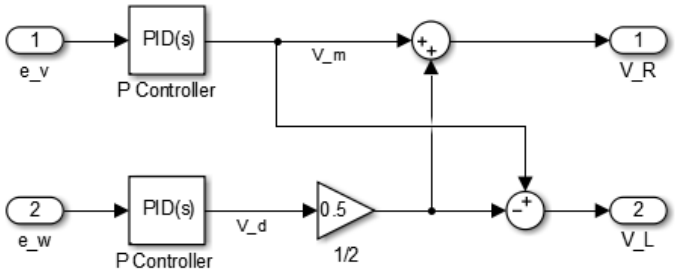


Figure 3.25: Mobile robot: Velocity Control.

The transfer functions of (3.28) and (3.29) are implemented in Figure 3.26 followed by (3.30) and (3.31).

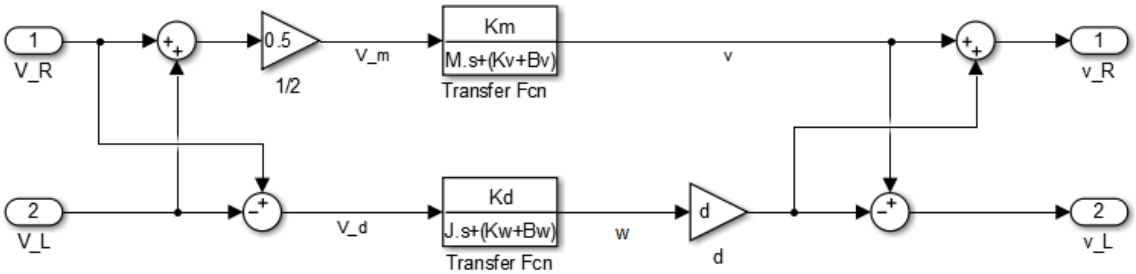


Figure 3.26: Mobile robot: Vehicle Dynamics.

Here, K_{Pm} and K_{Pd} are the proportional controllers inside the velocity control loop in Figure 3.26.

3.3.3.2.0.4 Trajectory tracking with mobile robot

Figure 3.27 represents the complete modelling of the mobile robot, the Position Control block used in this case is exactly the same one that is applied for the unicycle in Figure 3.20. Instead, the Kinematics of the vehicle is different and is reported in (3.23) and represented in Figure 3.28.

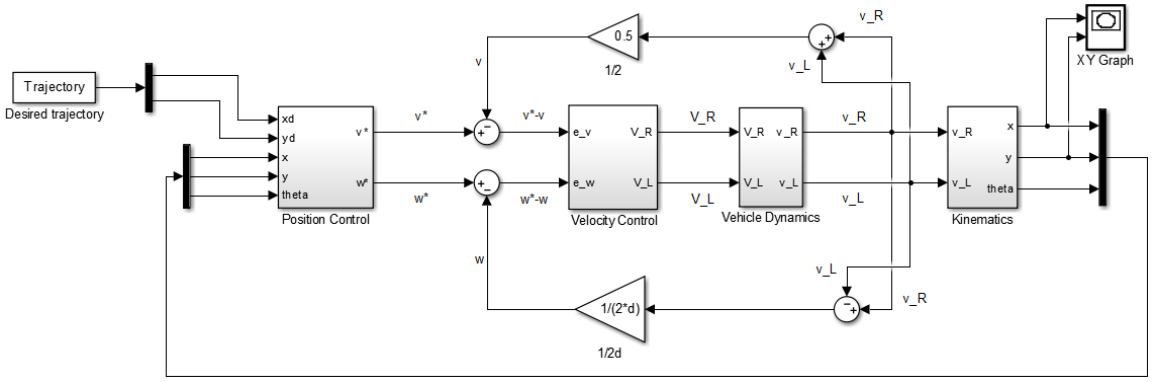


Figure 3.27: Mobile robot: trajectory tracking.

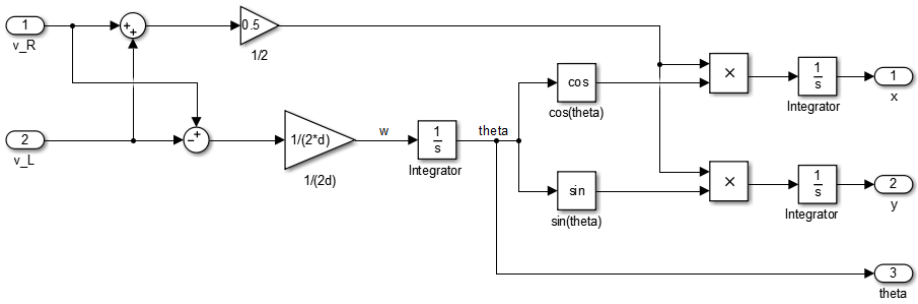


Figure 3.28: Mobile robot: Kinematics.

As with the example of the unicycle, also with the mobile robot model it is possible to compare the behaviours obtained with ideal trajectories the ones generated by the real dynamical model. Simulations in Figures 3.29 and 3.30 refer to rendezvous and formation.

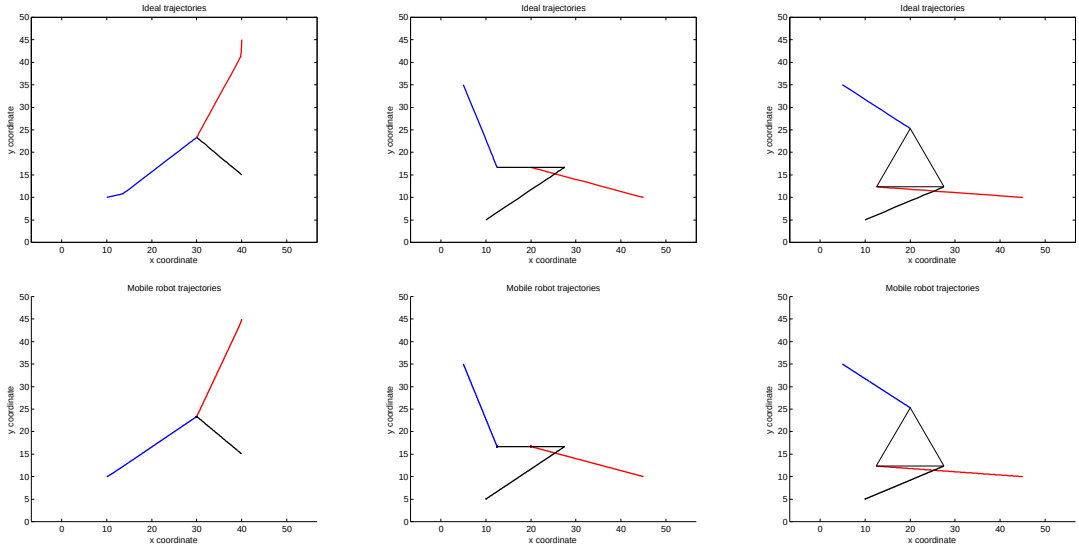


Figure 3.29: Mobile robot: projection dynamics

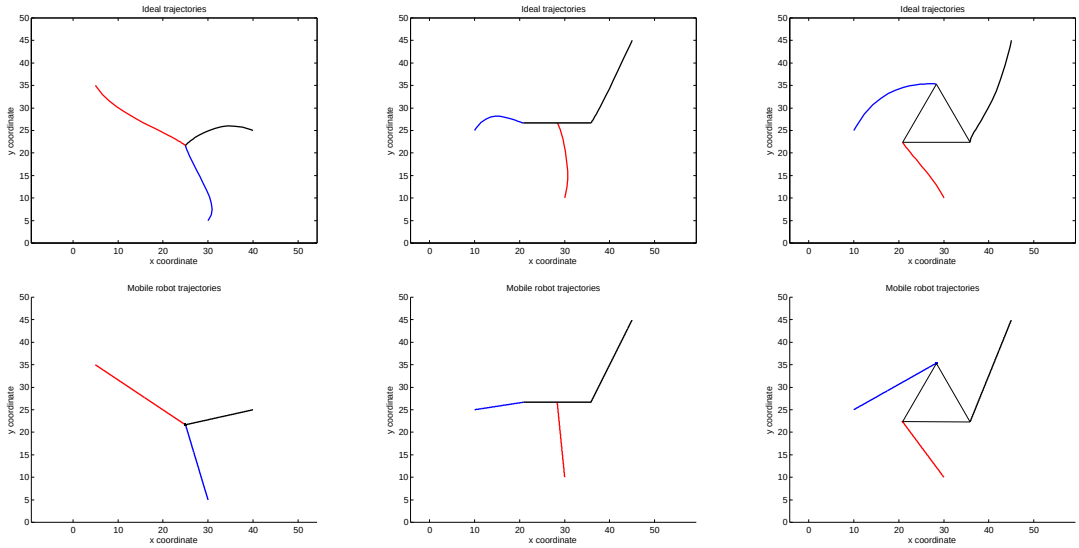


Figure 3.30: Mobile robot: replicator dynamics

Chapter 4

Implementation

4.1 Experimental setup

4.1.1 Platform operation

Before evaluating the efficiency of the application of game theory in distributed control, it is necessary to describe in practice which is the framework of the experiments performed.

The environment is constituted by an uniform plain rectangular area without obstacles. A fixed camera positioned above the area constantly films the events that occur. The camera is directly connected to the PC through an USB cable so the top view of the area can be see in real time from the computer monitor.

When the three LEGO Mindstorm robots are introduced into the environment, the next step is to establish a connection between the Mindstorms and the PC. This connection can be wireless and implemented through the Bluetooth protocol, or can be performed with USB connection with cables. Each agent has a specific name use ad identification, therefore the computer is able to send commands distinctly to the different robots.

A pattern matching tool, implemented with LabVIEW 2012, is responsible of the recognition of each robot inside the environment. The identification is made by comparison with a pattern that has been defined before starting the experiments.

Each Mindstorm is associated with a different shape pattern and this allows the software to distinguish between them.

In practice, the LabVIEW developed routine receives a continuous stream of images from the camera and returns in real time the position of the center of the pattern attached to the detected robot. The position is expressed in x-y coordinates of the central pixel, in the reference frame fixed with the image which has the origin in the upper left corner.

After setting the task to achieve, for example a point in space to reach, the software evaluates the distance from the current position and the desired one and sends the command to one of the Mindstorms, selecting it by choosing its specific Bluetooth ID.

As mentioned above, the position of the agents is given in pixels but the commands that the computer sends to the motors driving the wheels of the robots are expressed in rotation angle. For this reason a conversion is made passing through a first function that maps the pixel into centimetres and a second function that maps centimetres into degrees of rotation.

In the setup phase, one of the side the environment plane has been measured both in centimetres, with a meter in the real world, and in pixels simply considering the size of the image returned by the camera. With the following proportion it is possible to obtain the number of pixels in the image that correspond to the length of 1 centimetre in the real environment

$$\#pixel/cm = \frac{1\ cm}{length_{[pixel]}}. \tag{4.1}$$

The constant value obtained with the previous ratio is used to convert the distances from pixels scale to centimetres scale.

The last step consists in evaluating the rotation angle that the wheels must rotate in order to cover the given distance. That value can is obtained by considering the radius R_w of the wheels:

$$\beta = \frac{distance_{[cm]} \cdot 360^o}{2\pi R_w}. \tag{4.2}$$

4.1.1.1 Pattern recognition

Going into details, Figure 4.1 shows the structure of the LabVIEW component designed to perform pattern matching. The Vision Acquisition module is responsible of the video streaming and it is inserted into a while loop in order to acquire an image at each loop, in this way a rapid sequence of frames is performed.

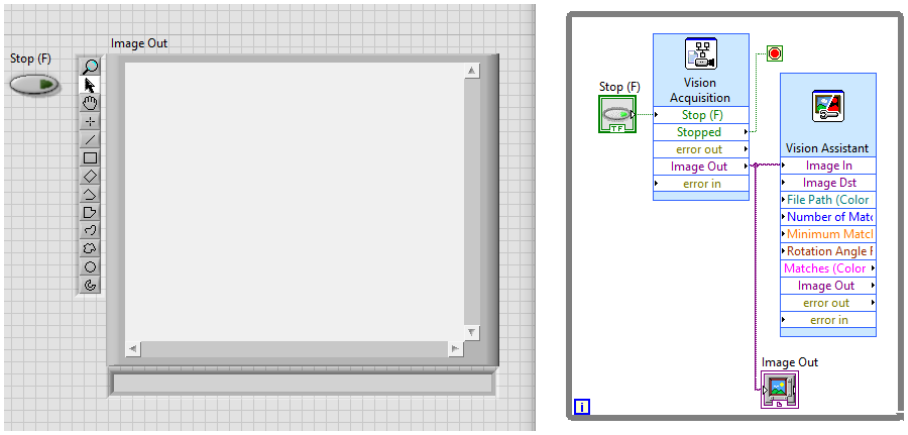


Figure 4.1: Vision Acquisition and Vision Assistant tools.

On the other hand, the Vision Assistant is the module that actually implements the patter matching by taking as input the image returned by the Vision Acquisition block. By setting a pattern image, this module can identify and localize the pattern inside the input image. Finally, the coordinates in pixels of the center of the pattern are returned as output.

In Figure 4.2 are reported the three different patterns used to identify the Mindstorms. The shapes have been chosen to be distinguishable to each.

EV3 03 EV3 09 EV3 10

Figure 4.2: Different patterns used to distinguish between Mindstorms.

The Vision Assistants support the possibility of recognizing a particular pattern even if it is rotated of a certain range of degree. The interval of rotation is imposted as a parameter, in this case the admissible rotation belongs to $[-\pi; \pi]$.

The impossibility of evaluating the current orientation of the Mindstorms with the Vision Assistants requires to make an assumption: the starting orientation of the

Robots must be known. For this reason, every time the Mindstorms move, the angle of rotation are evaluated and new values of the orientations saved inside specific variables. When these variables are used in the next iteration, they represent the new current orientations.

4.1.1.2 USB and Bluetooth connections

When the robots are localized into the acquired image, it is necessary to establish a connection between them and the PC.

The first time the Mindstorms connect with the computer through Bluetooth it is necessary to perform the devices association. Moreover, different identification names are assigned to the Mindstorms in order to distinguish among them when commands are sent to the motors.

In Figure 4.3 is represented the LabVIEW implementation of this operation. The first step is to assign a name to the searched Mindstorm, EV3 in this case, and the type of connection, Bluetooth, USB or WiFi. When the Bluetooth of the PC find a device, it asks to set up the connection indicating the passkey (1234). Once the connection is established, the Bluetooth address of the Mindstorm is obtained and also it has been assigned to it the identification name EV3. When the moment of controlling the action of the robots arrives, it is sufficient to select the Minstorm by using its identification name and type of connection as inputs.

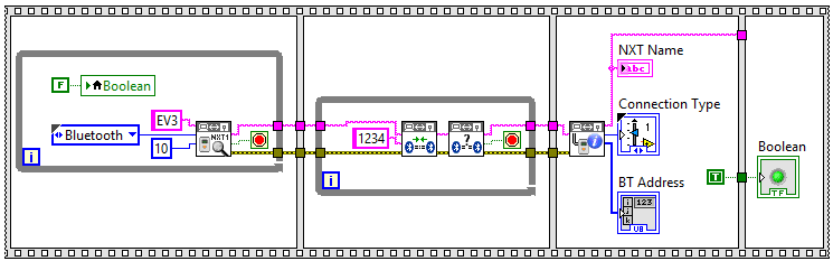


Figure 4.3: Establish Bluetooth connection.

Once the identification names are assigned to the Mindstorms, it is no more necessary to execute these operations, it is enough to indicate the name and the connection type. USB connection can be established selecting the USB option in the control that specifies the connection type.

4.1.1.3 Robots motion

4.1.1.3.1 Setup operations

The LabVIEW scheme in Figure 4.4 performs a forward rotation for 10s of the wheels attached to motor A and C and than stops the motion.

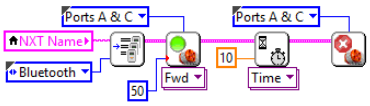


Figure 4.4: Put in motion a robot.

In Figure 4.5 are represented the computations done to evaluate the distance that one robot has to cover in order to reach an assigned destination, starting from is current position. This value is obtained in pixels because the coordinates are expressed in pixels too, therefore, it is necessary to apply the conversion from pixels to centimetres dividing by 13.05. This number has been found applying (4.1).

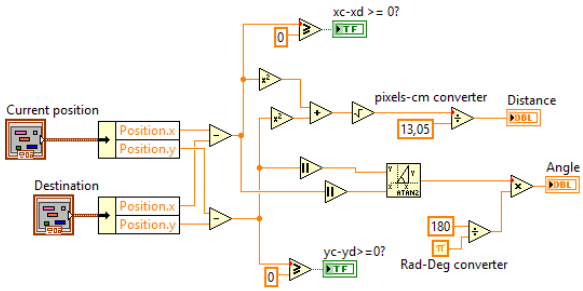


Figure 4.5: Evaluation of distance and direction.

Moreover, the angle of rotation that the wheels have to turn in order to align the Mindstorm to the destination point is computed. The function used to evaluate this angle is the $atan2$ of the absolute value of the distances along horizontal and vertical directions. In this way the result is always between 0 and $\frac{\pi}{2}$.

To distinguish the four possible cases that can occur regarding the mutual position between current location and destination, two boolean variables are created. One for the horizontal distance and the other one for the vertical distance. In Figure 4.6 are explained in detail the four possible scenarios and the corresponding actions associated to each choice.

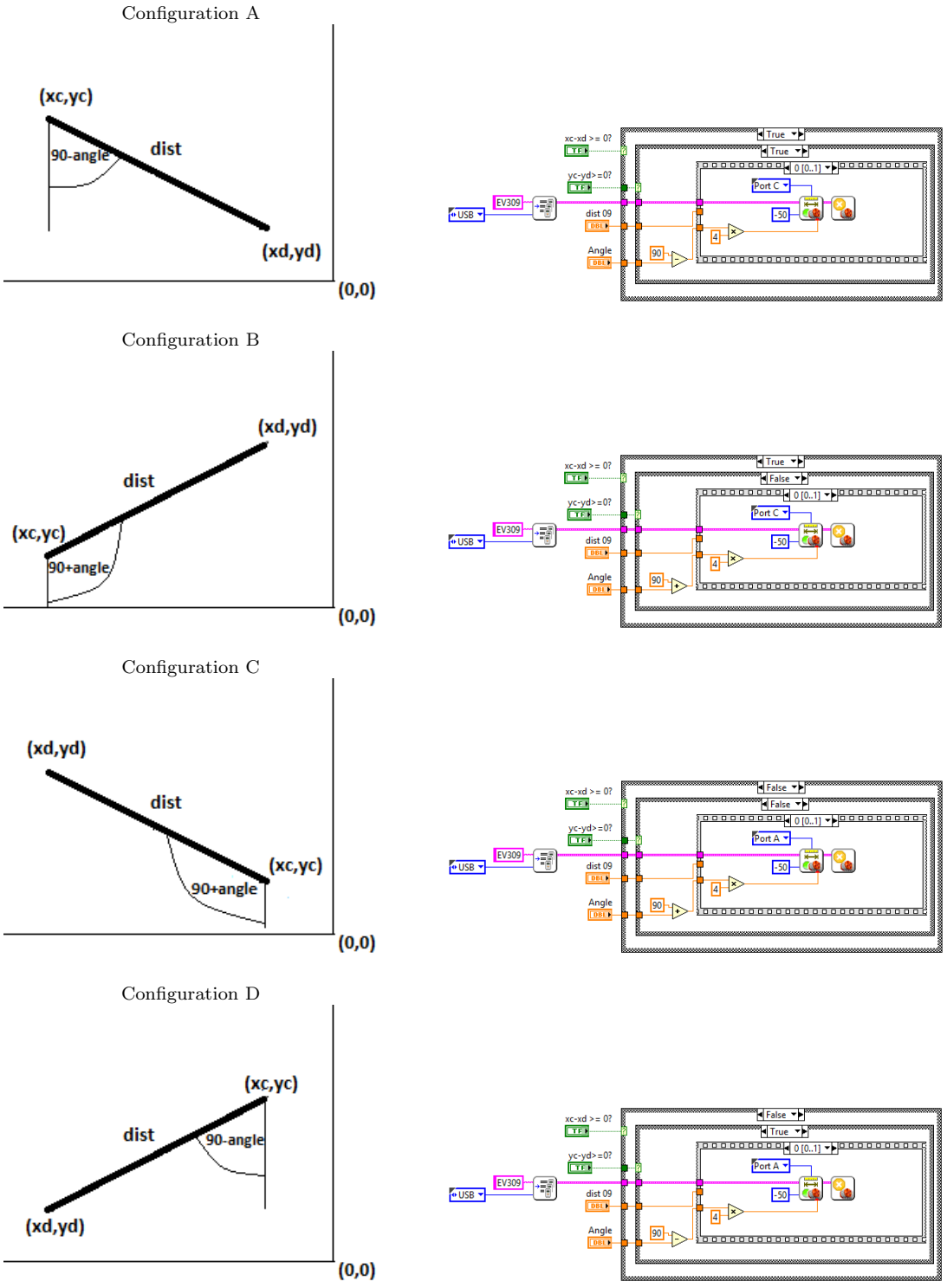
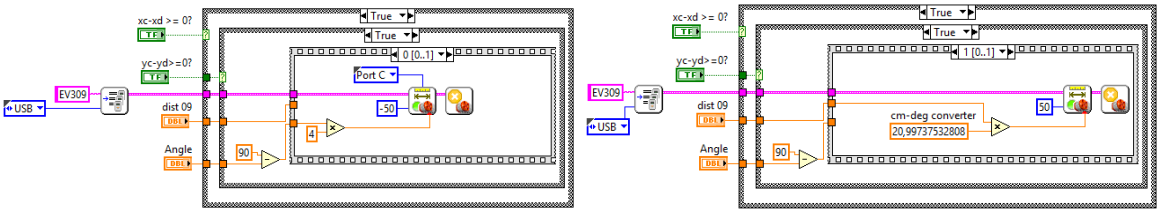


Figure 4.6: Possible configuration of current position and destination.

Port C of the brick is connected to the motor that drives the left wheel thus the backward rotation of this motor produces an anticlockwise rotation of the Mindstorm around itself. Analogously the backward rotation of the other wheel driven by motor A produces the clockwise rotation of the Mindstorm. When the rotation is performed and the robot is finally oriented in the direction of the destination point, it can start moving in straight direction. The distance that it has to cover has been already computed in Figure 4.5 but it is necessary to convert this value in degree of rotation of the wheels. The conversion number $20,99737532808$ is obtained applying (4.2). If the name of the port is not specified, like in Figure 4.7(b), all the motors attached to the brick are started.



(a) Turning around (b) Straight advance

Figure 4.7: Phase of motion.

4.1.1.3.2 Reaching a destination: Solution A

The first experiment consists in moving the Mindstorm from its current position to a final destination in the environment. The program, during the stream of the camera, continuously locates and returns the current position of each agent. After choosing one of the Mindstorm, by assigning a desired position target, the application evaluates automatically the distances along x and y directions that the robot has to cover to reach the destination.

At the beginning of the experiment the Mindstorms are oriented in the same way facing the y direction. Every time one of them is activated, it turns of 90° in the correct verso, determined by the position of the target, than goes straight along x direction, turns again of 90° and finally follows the y direction until it reaches the destination. At the end of the motion, when the target is reached, the robot repositions itself in the initial orientation.

Solution A reports the pseudo-code of the algorithm that discriminates between the possible motion solutions according to the current position of the robot with respect

to the destination.

SOLUTION A

$(x_c; y_c) \leftarrow$ *current position*
$(x_d; y_d) \leftarrow$ *desired position*
if $x_c - x_d > 0$
  then if $y_c - y_d > 0$
    then *rotate of* 90° *counterclockwise*
      go straight for $|x_c - x_d|$
      rotate of 90° *clockwise*
      go straight for $|y_c - y_d|$
    else *rotate of* 90° *counterclockwise*
      go straight for $|x_c - x_d|$
      rotate of 90° *counterclockwise*
      go straight for $|y_c - y_d|$
      rotate of 180°
    end
  else if $y_c - y_d > 0$
    then *rotate of* 90° *clockwise*
      go straight for $|x_c - x_d|$
      rotate of 90° *counterclockwise*
      go straight for $|y_c - y_d|$
    else *rotate of* 90° *clockwise*
      go straight for $|x_c - x_d|$
      rotate of 90° *clockwise*
      go straight for $|y_c - y_d|$
      rotate of 180°
    end
  end

4.1.1.3.3 Reaching a destination: Solution B

A simple evolution of the previous solution is to drive the Mindstorms directly in the correct direction of the target. In order to perform this procedure, it is necessary to evaluate the angle of rotation between the current orientation and the one that

permits to align with the destination point. The pseudo-code of this algorithm is reported in SOLUTION B.

SOLUTION B

$(x_c; y_c) \leftarrow current\ position$
$(x_d; y_d) \leftarrow desired\ position$
$d \leftarrow \sqrt{(x_c - x_d)^2 + (y_c - y_d)^2}$
$\alpha \leftarrow \arctan_{deg}(|y_c - y_d|/|x_c - x_d|)$
if $x_c - x_d > 0$
  then if $y_c - y_d > 0$
    then $rotate\ of\ (90 - \alpha)°\ counterclockwise$
      $go\ straight\ for\ d$
      $rotate\ of\ (90 - \alpha)°\ clockwise$
    else $rotate\ of\ (90 + \alpha)°\ counterclockwise$
      $go\ straight\ for\ d$
      $rotate\ of\ (90 + \alpha)°\ clockwise$
    end
  else if $y_c - y_d > 0$
    then $rotate\ of\ (90 - \alpha)°\ clockwise$
      $go\ straight\ for\ d$
      $rotate\ of\ (90 - \alpha)°\ counterclockwise$
    else $rotate\ of\ (90 + \alpha)°\ clockwise$
      $go\ straight\ for\ d$
      $rotate\ of\ (90 + \alpha)°\ counterclockwise$
    end
  end

4.1.1.3.4 Reaching a destination: Solution C

The choice of resetting initial orientation every time one of the Mindstorm reach the destination is only a convention that can be easily modified by taking in memory the current orientation that the robot has when it stops.

As mentioned before, the Vision Assistant is set in a way that it can recognize the patterns even if they are rotated during the motion. A drawback is that the angle of rotation is not returned as output by the Vision Assistant module. This is a

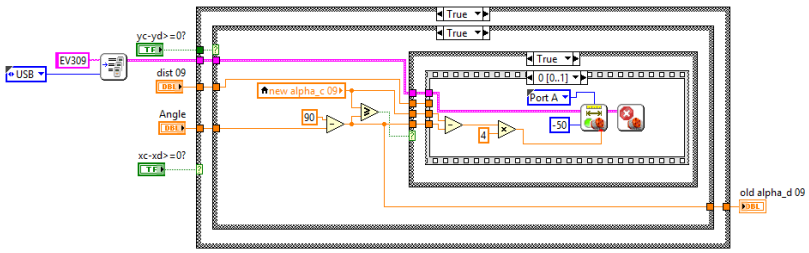
disadvantage of pattern recognition because that application can only return the positions in the space of the agents but not how they are oriented.

Pseudo-code in SOLUTION C shows how the algorithms works when the orientation is not reset at the end of every motion.

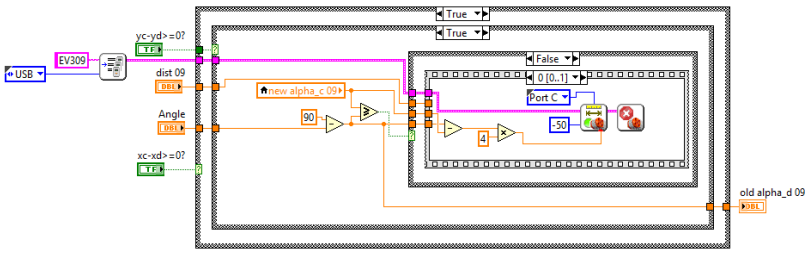
SOLUTION C

$(x_c; y_c) \leftarrow$ *current position*
$\alpha_c \leftarrow$ *current orientation*
$(x_d; y_d) \leftarrow$ *desired position*
$d \leftarrow \sqrt{(x_c - x_d)^2 + (y_c - y_d)^2}$
$\alpha_d \leftarrow \arctan_{deg}(|y_c - y_d|/|x_c - x_d|)$
if $\alpha_c - \alpha_d > 0$
  then *rotate of* $(\alpha_c - \alpha_d)°$ *clockwise*
    go straight for d
    $\alpha_c \leftarrow \alpha_d$
  else *rotate of* $(\alpha_d - \alpha_c)°$ *clockwise*
    go straight for d
    $\alpha_c \leftarrow \alpha_d$
  end

Figure 4.8 represents the two possible cases that can occur when the information about the current angle is taken in memory. This figure refers to the configuration A already explained in Figure 4.6, however, with the other three configurations the content of the innermost case structure is identical.



(a) $\alpha_c > \alpha_d$



(b) $\alpha_c < \alpha_d$

Figure 4.8: Different cases with current orientation info.

4.1.1.4 USB connection and virtual robot

During these simple experiments emerged a critical problem related to physical limitation of Bluetooth technology. The LabVIEW NTX Terminal is able to maintain connected only two Mindstorms per time with Bluetooth. This drawback is incompatible with the objective of the project because the central aim is to implement multi-agent experiments.

The solution adopted is twofold. The first choice is to discard Bluetooth and establish USB connection between the Mindstorms and the PC. This solution does not require to modify the code and the structure of the platform because it is only necessary to change the constant parameter that identifies the connection type. The only disadvantage is given by the encumbrance caused by the physical cables and their limited lengths.

The second option is to connect two of the Mindstorms with Bluetooth and to simulate the presence of the third one. It is assumed that the virtual robot has an ideal behaviour, thus it follows the reference that indicates the position to reach without error.

4.1.2 Discretisation

In order to generate the trajectories that the Mindstorms have to follow, has been necessary to discretise the continuous dynamic equations used in Section 3.3.2.

If x_i is one of the coordinates of the robots, the continuous dynamic equation is given by $\dot{x}_i = V_i^{\mathbf{F}}(\mathbf{x})$, where in this case $V^{\mathbf{F}}(\mathbf{x})$ represents the expression of projection or replicator dynamics defined in the previous chapter. In the discrete case, the related updating law is

$$x_i(t+\tau) = x_i(t) + \tau V_i^{\mathbf{F}}(\mathbf{x}), \tag{4.3}$$

where the sampling time is denoted with τ.

In Figure 4.9 is represented the LabVIEW implementation of equation (4.3) in the case when $V_i^{\mathbf{F}}$ stands for projection dynamics. The number of robots is $n = 3$, then $\mathbf{x} = [x_1 \quad x_2 \quad x_3]^{\mathsf{T}}$. The analogous scheme for the replicator dynamics differential equation is shown in Figure 4.10.

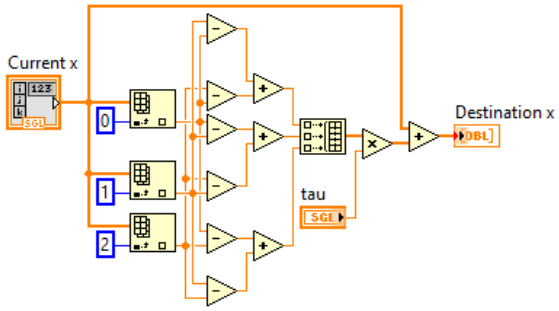


Figure 4.9: Discrete dynamic updating law scheme: projection dynamics.

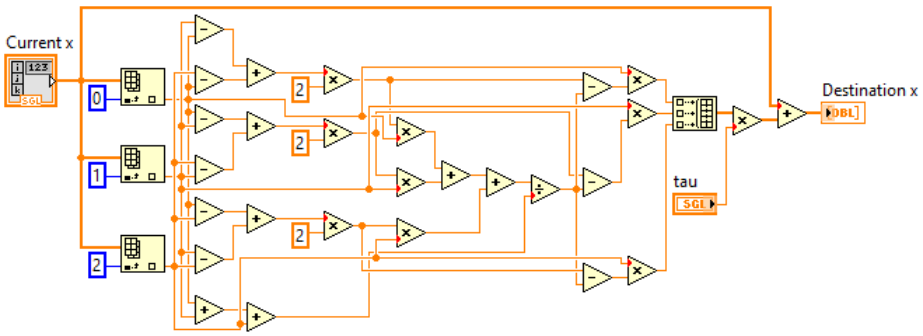


Figure 4.10: Discrete dynamic updating law scheme: replicator dynamics.

4.2 Results

4.2.1 Projection dynamics with USB connection

4.2.1.1 Rendezvous with PD and USB

The first consistent experiments with the Mindstorms using a distributed control has been made applying the projection dynamics algorithm simulated in Section 3.3. The goal is to make the robots converge to a common position in the space.

In this experiment $\tau = 0.2s$ and the dynamics has been split in six phases of length equal to τ. The sequential positions detected with the camera during the motion are the one reported in Table 4.1. Moreover, Figure 4.11 represents the discrete evolution in time of the coordinates of the Mindstorms and their real trajectories during the experiment.

Table 4.1: Coordinates in pixels of the Mindstorms in projection dynamics rendezvous experiment with USB connection.

PD rendezvous			
time	EV310	EV309	EV303
$t = 0$	(247,479)	(1039,739)	(1028,178)
$t = \tau$	(629,455)	(860,654)	(981,450)
$t = 2\tau$	(709,504)	(855,610)	(922,460)
$t = 3\tau$	(772,516)	(825,579)	(898,488)
$t = 4\tau$	(782,530)	(818,593)	(890,508)
$t = 5\tau$	(777,530)	(808,599)	(884,521)

The numerical values of the final positions of the Mindstorms need an explanation. These experiments are performed with real robots that occupy a volume in the space, then it is impossible to make them reach exactly the same position without collide. This is the reason why the coordinates of the Mindstorms cannot coincide perfectly at the end of the motion.

Figure 4.12 represents the sequence of images captured during the experiment by the camera. The positions occupied by the Mindstorms in these images are exactly the

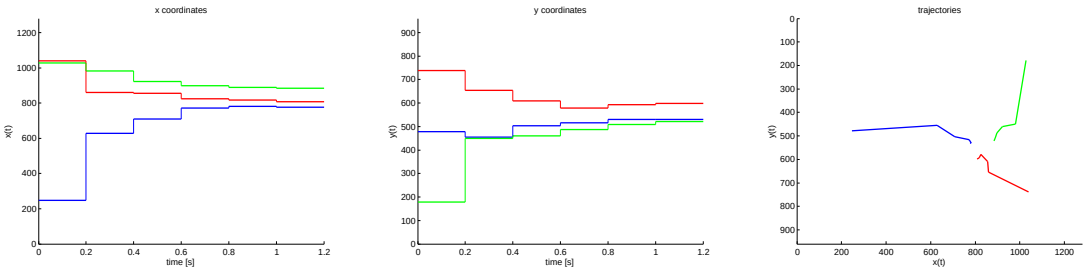


Figure 4.11: PD: rendezvous (USB).

ones reported in Table 4.1.

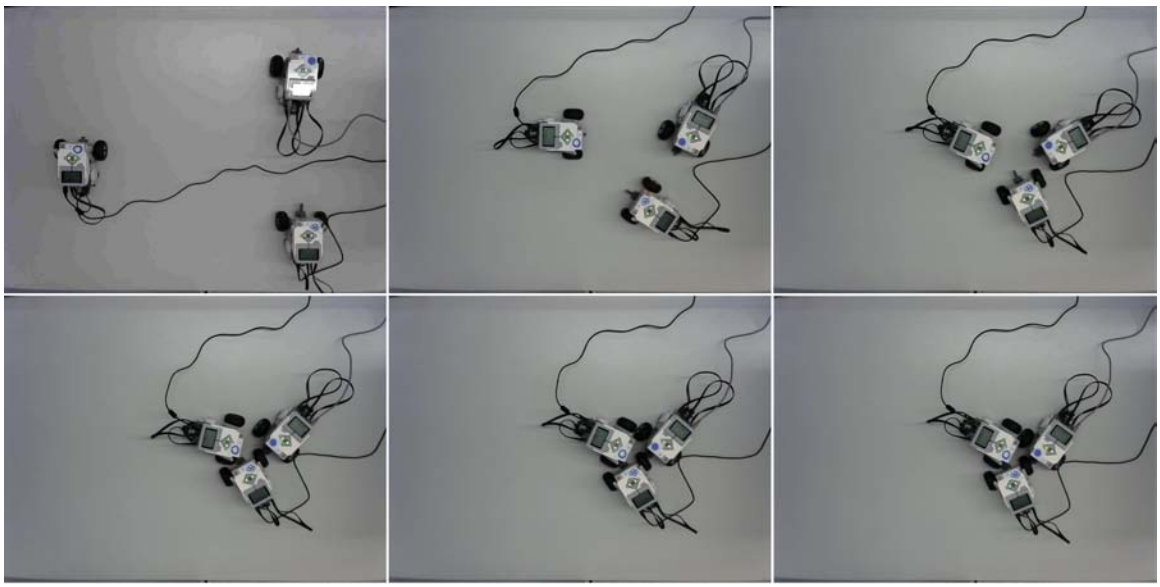


Figure 4.12: Projection dynamics rendezvous sequence with Mindstorms (USB).

4.2.1.2 Formation with PD and USB

Formation experiments with projection dynamics produced the desired results because the numerical values obtained are consistent with theoretical ones. In these case, the non-null dimension of the robots do not influence the experiment because in formation the final distance between Mindstorms has been chosen grater than their size.

The formation performed are the same of the simulations in the previous chapter, triangle and segment with $\ell = 350$ pixels side.

Table 4.2 refers to triangular formation and alignment. The first one adopts $\tau = 0.3s$ and the graphical representation of the correspondent numerical values in the table are represented in Figure 4.13. On the other hand, Figure 4.15 is related to alignment with $\tau = 0.2s$.

Table 4.2: Coordinates in pixels of the Mindstorms in Projection Dynamics formation experiment with USB connection.

	PD triangular formation			PD alignment		
time	EV310	EV309	EV303	EV310	EV309	EV303
$t = 0$	(266,536)	(943,759)	(1112,150)	(274,571)	(958,718)	(1089,121)
$t = \tau$	(595,348)	(810,764)	(1005,271)	(510,469)	(840,620)	(1106,393)
$t = 2\tau$	(565,377)	(856,735)	(1081,239)	(524,507)	(842,579)	(1057,405)
$t = 3\tau$	(608,334)	(819,685)	(1043,274)	(575,507)	(814,546)	(1017,459)
$t = 4\tau$	(628,339)	(797,638)	(1007,268)	(604,511)	(809,530)	(996,489)
$t = 5\tau$	(578,337)	(815,653)	(1008,276)	(616,517)	(804,517)	(986,494)

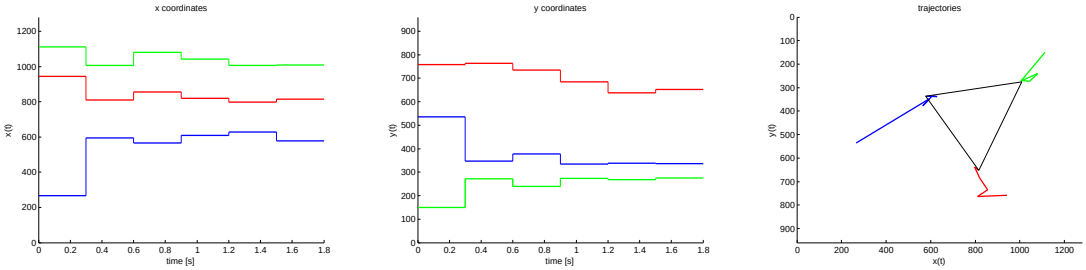


Figure 4.13: PD: triangular formation (USB).

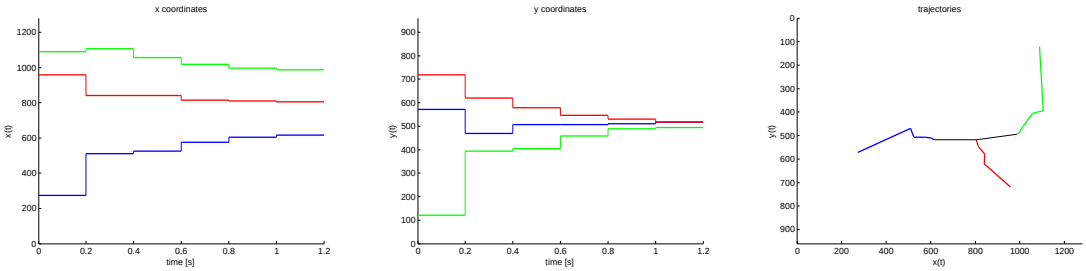


Figure 4.14: PD: alignment (USB).

Figure 4.14 represents the sequence of actions performed by the Mindstorms during experiments in order to reach the triangular formation with projection dynamics.

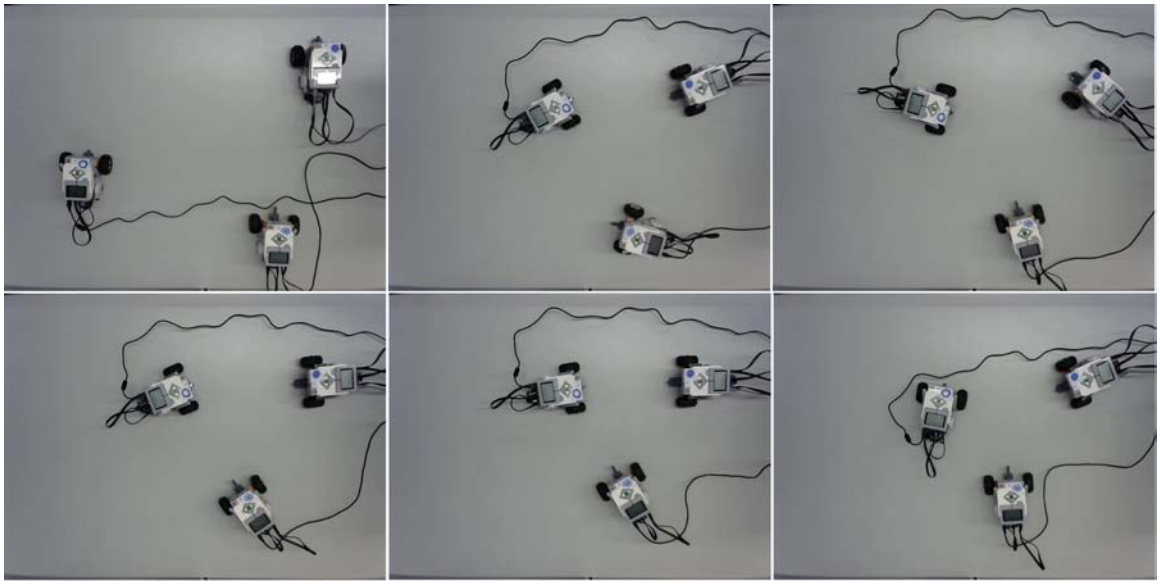


Figure 4.15: Projection dynamics triangular formation sequence with Mindstorms (USB).

In Figure 4.16 are shown the steps of the motion that the robots execute to align themselves along a line with projection dynamics.

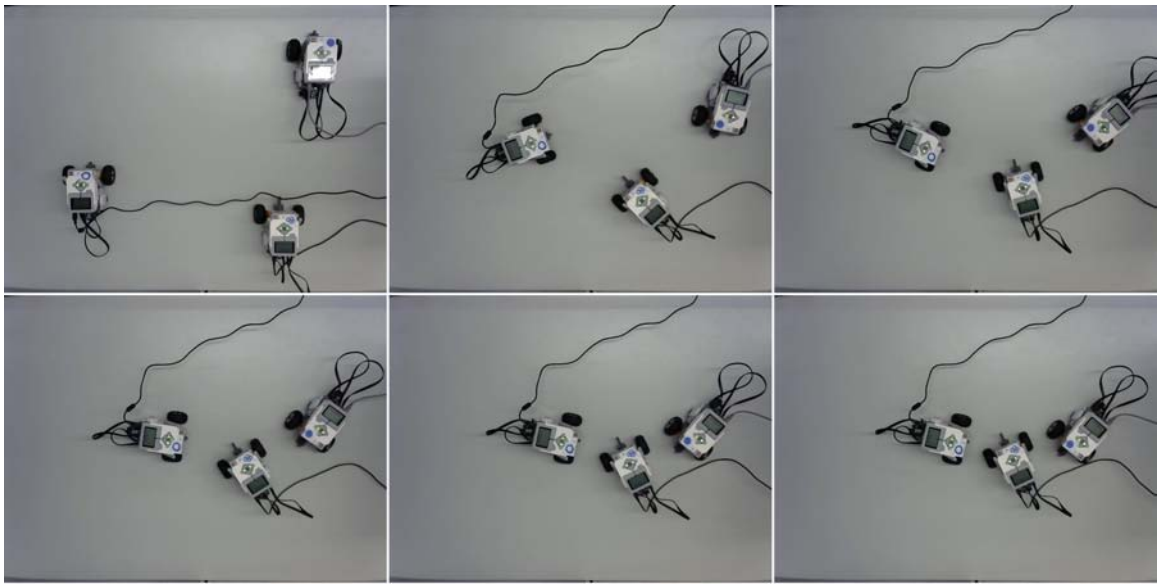


Figure 4.16: Projection dynamics alignment sequence with Mindstorms (USB).

4.2.1.3 External intervention in triangular formation with PD and USB

The code was also modified to solve a different problem. When the Mindstorms reach their final positions to perform triangular or linear formation, an external agent moves manually one of the robots and change its position. In that case, the Vision Assistant detects that the coordinates of this Mindstorm changed. After that, the program re-activates the robot and brings him back to the position that he left in order to restore the previous formation.

In Table 4.3 are reported the coordinates of the Mindstorms obtained in the experiment of triangular formation. The new positions are the one detected by the Vision Assistant when the Mindstorm EV303 has been moved from its final position that it reached at time $t = 5\tau$. The program detects that the coordinates of the Mindstorm EV303 changed and brings him back to the right position. Finally, at the end of the experiments, the positions detected are the ones that occupy the last line of the table. Also in this case, the error between positions at time $t = 5\tau$ and final positions is acceptable for practical purposes.

Table 4.3: Coordinates in pixels of the Mindstorms in triangular formation experiment with external intervention and USB connection.

PD triangular formation			
time	EV310	EV309	EV303
$t = 0$	(276,536)	(974,756)	(1014,132)
$t = \tau$	(612,390)	(796,773)	(989,233)
$t = 2\tau$	(596,378)	(754,638)	(1100,228)
$t = 3\tau$	(633,342)	(779,695)	(1045,290)
$t = 4\tau$	(582,385)	(769,663)	(1048,277)
$t = 5\tau$	(578,350)	(776,666)	(995,295)
new positions	(578,350)	(776,666)	(1102,109)
final positions	(578,350)	(776,666)	(1002,301)

In Figure 4.17 the dashed lines represent the displacement caused by the external agent.

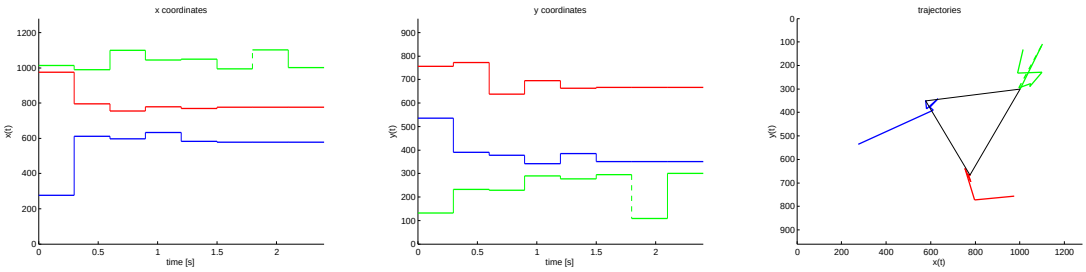


Figure 4.17: Projection dynamics triangular formation sequence with Mindstorms, external intervention (USB).

Figure 4.18 represents the sequence of positions that the Mindstorms occupy during the experiments of triangular formation with projection dynamics. In the first picture the robots are in the initial positions, in the sixth in the final positions, in the seventh they are in the new positions generated after the moving of robot EV303, and finally, the last picture shows the new final positions. The Mindstorm EV303 returns to the correct position, indeed, the sixth and the last situations look alike.

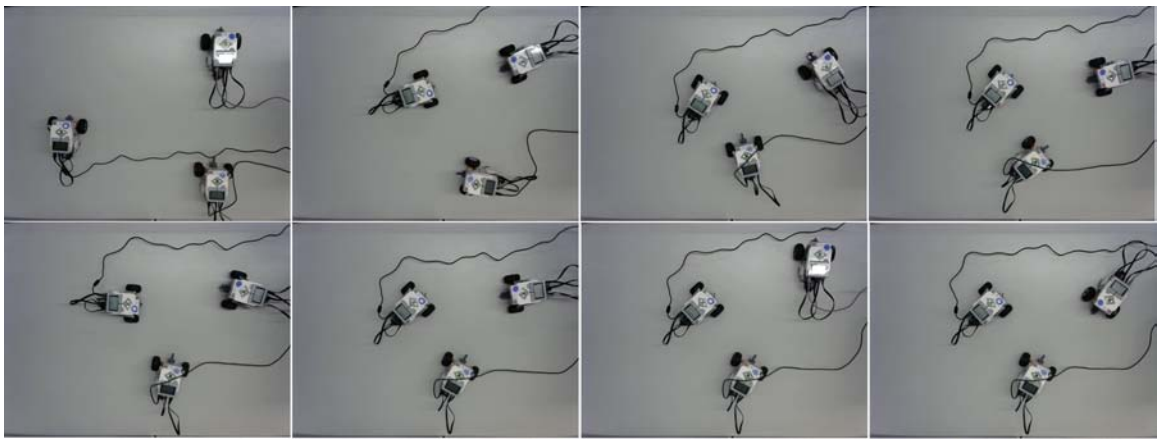


Figure 4.18: Triangular formation sequence with Mindstorms with external position modification (USB).

4.2.2 Replicator dynamics with USB connection

4.2.2.1 Rendezvous with RD and USB

The convergence speed of replicator dynamics is higher than the one of projection dynamics with the choice of fitness function made in Chapter 3 in (3.13) and (3.21). This fact has already been shown with MatLab simulations where the convergence

time with projection is in the order of seconds and with replicator of hundredths of a second. Because of this, the choice of the sampling time in replicator dynamics experiments is shorter and equal to $\tau = 0.0001s$.

Figure 4.19 shows the real behaviour of the Mindstorms during the rendezvous experiment with replicator dynamics. The convergence of y coordinates seems to be not achieved. The reason is that the robots are positioned in a way that if they continued to get closer they collide. It is important to remind that the results of these experiments are always related to real agents with non-null dimensions.

Table 4.4: Coordinates in pixels of the Mindstorms in replicator dynamics rendezvous experiment with USB connection.

RD rendezvous			
time	EV310	EV309	EV303
$t = 0$	(269,356)	(974,745)	(1122,107)
$t = \tau$	(453,418)	(968,736)	(1071,234)
$t = 2\tau$	(572,406)	(953,719)	(1008,252)
$t = 3\tau$	(629,461)	(949,706)	(960,305)
$t = 4\tau$	(721,494)	(940,672)	(949,345)
$t = 5\tau$	(783,519)	(918,639)	(925,390)
$t = 6\tau$	(825,531)	(896,607)	(920,426)
$t = 7\tau$	(833,534)	(896,595)	(913,421)

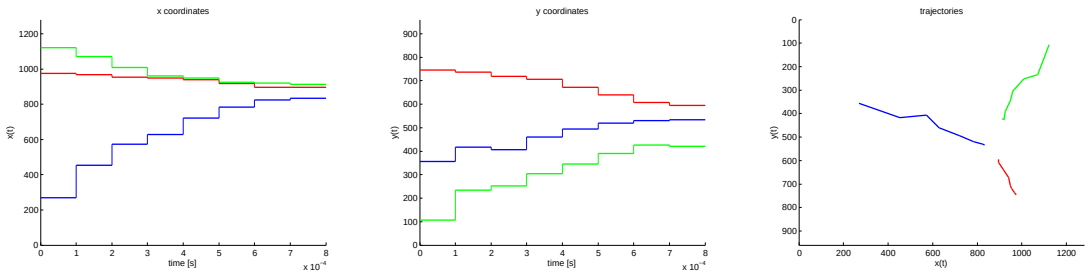


Figure 4.19: RD: rendezvous (USB).

In Figure 4.20 is represented the sequence of positions covered by the Mindstorms during the path that brings them from the initial positions to the destination during replicator dynamics experiment.

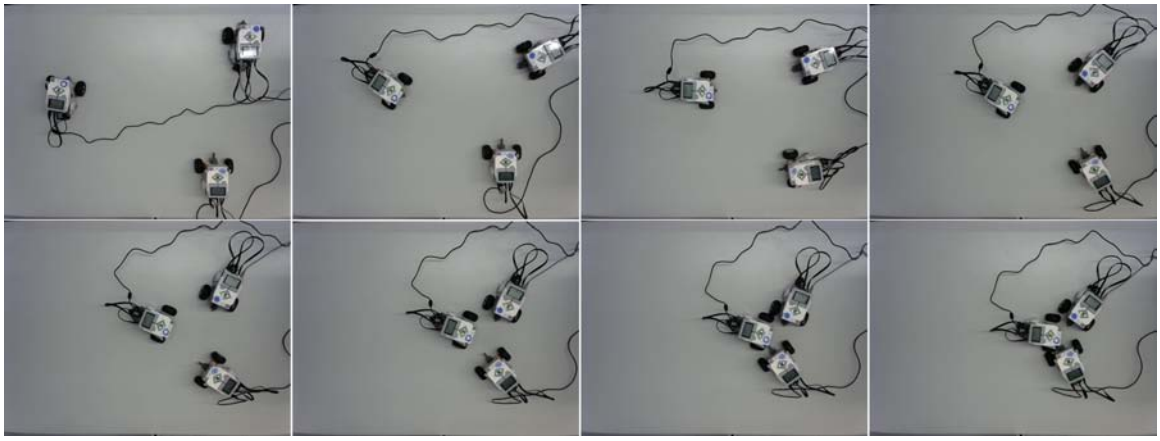


Figure 4.20: Replicator dynamics rendezvous sequence with Mindstorms (USB).

4.2.2.2 Formation with RD and USB

As with projection dynamics, the length of the triangle side and of the segment in formation problem is equal to $\ell = 350$ pixels. In Table 4.5 are reported the phases in which the trajectories have been split. The effectiveness of control can be better seen from Figures 4.21 and 4.22, where the achievement of the desired formation is highlighted in black.

Table 4.5: Coordinates in pixels of the Mindstorms in replicator dynamics triangular formation experiment with USB connection.

	RD triangular formation			RD alignment		
time	EV310	EV309	EV303	EV310	EV309	EV303
$t = 0$	(282,268)	(926,736)	(1108,124)	(312,490)	(950,725)	(1108,112)
$t = \tau$	(425,250)	(903,745)	(1137,217)	(468,484)	(928,697)	(1106,220)
$t = 2\tau$	(491,268)	(916,740)	(1094,252)	(531,500)	(906,648)	(1083,264)
$t = 3\tau$	(546,286)	(891,725)	(1076,280)	(574,506)	(886,607)	(1060,309)
$t = 4\tau$	(588,300)	(878,709)	(1058,305)	(601,505)	(866,571)	(1049,346)
$t = 5\tau$	(609,316)	(864,694)	(1047,313)	(621,498)	(859,551)	(1040,377)
$t = 6\tau$	(629,326)	(858,685)	(1037,336)	(638,497)	(852,534)	(1033,400)
$t = 7\tau$	(636,330)	(854,680)	(1031,331)	(644,493)	(849,523)	(1029,422)
$t = 8\tau$	-	-	-	(652,491)	(844,510)	(1029,436)

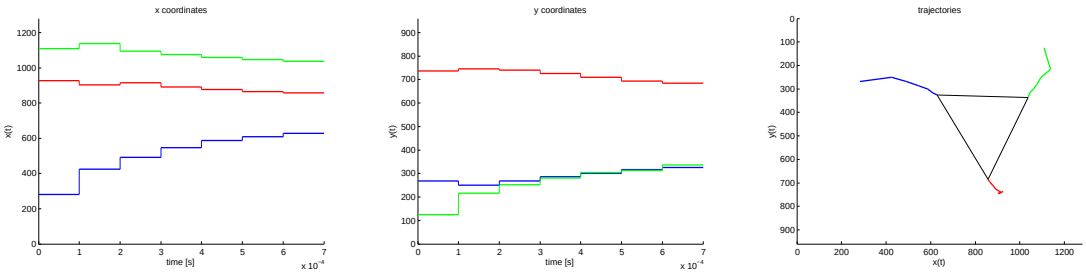


Figure 4.21: RD: triangular formation (USB).

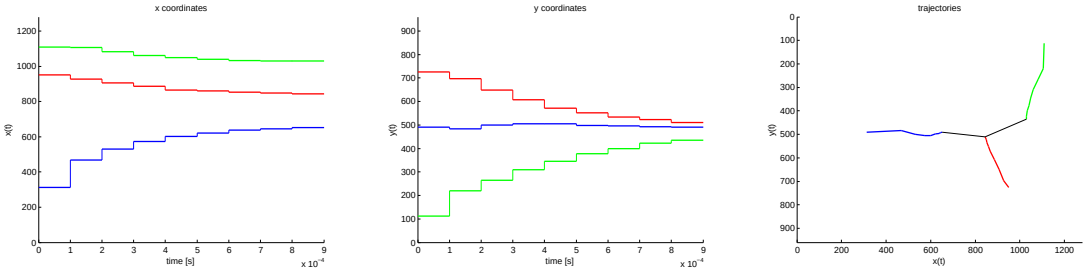


Figure 4.22: RD: alignment (USB).

Figures 4.23 and 4.24 represent the sequence of positions occupied by the Mindstorms during triangular formation and alignment with replicator dynamics, respectively.

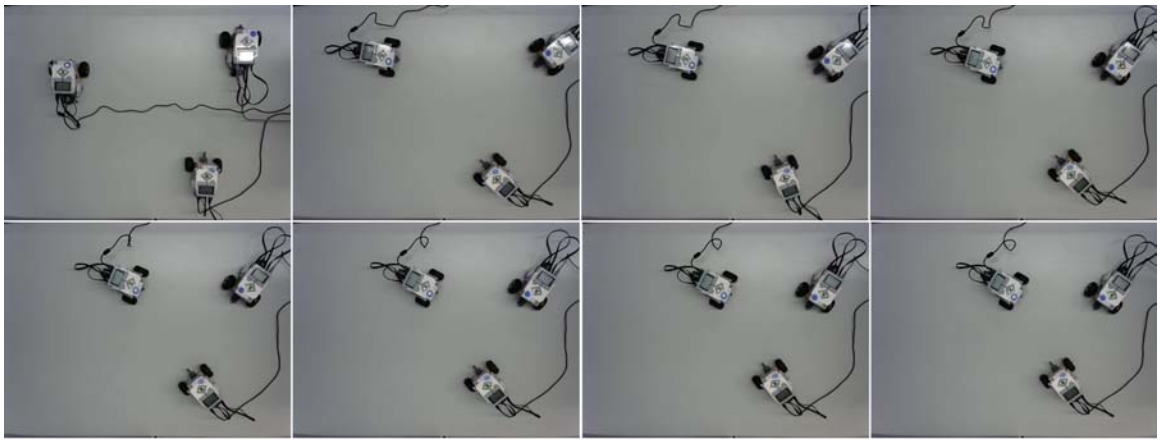


Figure 4.23: Replicator dynamics triangular formation sequence with Mindstorms (USB).

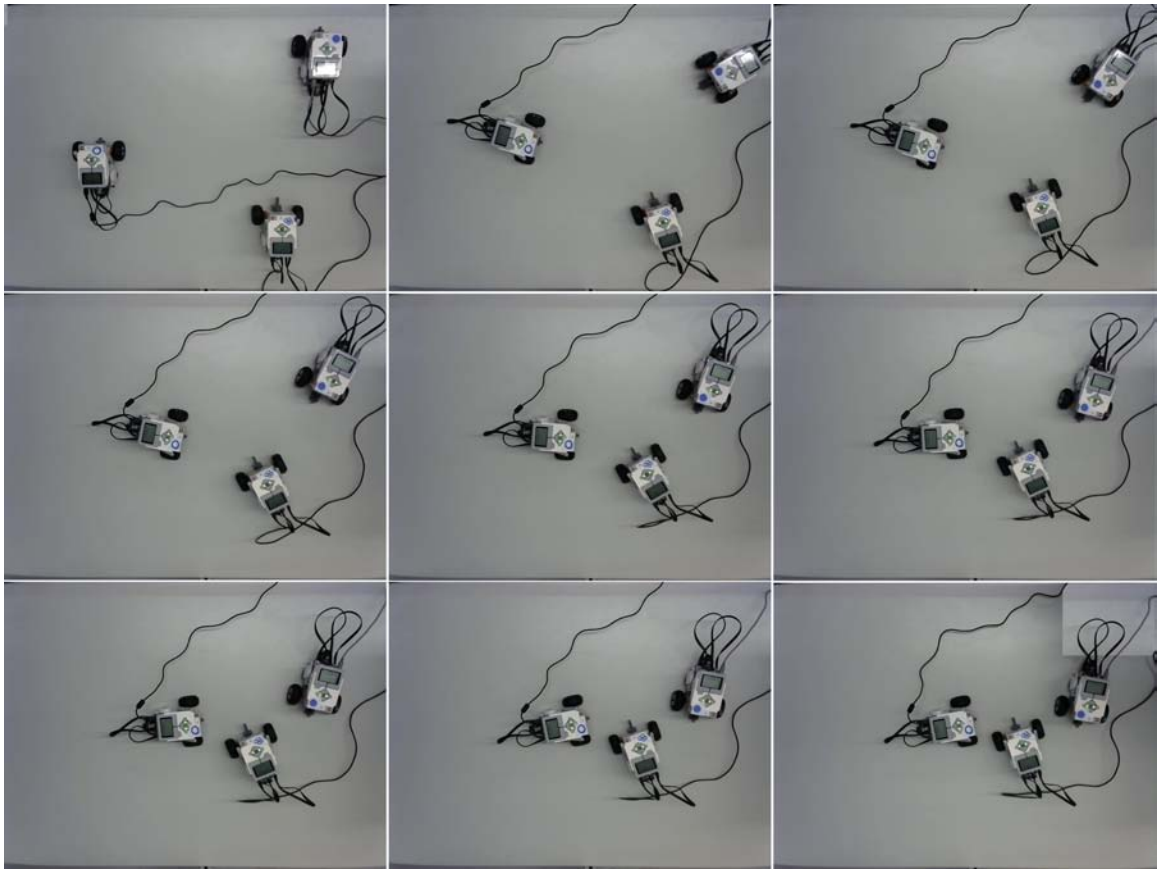


Figure 4.24: Replicator dynamics alignment sequence with Mindstorms (USB).

4.2.3 Projection dynamics with Bluetooth connection

As mentioned before, with the NXT LabVIEW Terminal is impossible to maintain the connection of more than two Mindstorms per time using the Bluetooth. In the experiments with Bluetooth the behaviour of the third Mindstorm that cannot be connected is simulated. The interesting fact associated with this solution is that the virtual robot has the role of connecting the other two that cannot communicate directly. To be more clear, the neighbouring graph that explains the relationships between the agents is the one represented in Figure 4.25. The Mindstorm EV310 is the virtual agent, instead, the EV303 and the EV309 are effectively connected to the PC with Bluetooth.

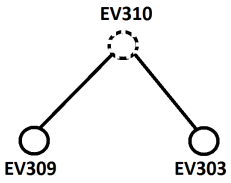


Figure 4.25: Neighbouring graph in experiments with virtual robot.

The initial position of the Mindstorm EV310 has been chosen randomly, instead, the initial positions of the other two robots are detected by the Vision Assistant as in the previous experiments.

4.2.3.1 Rendezvous with PD and Bluetooth

Table 4.6 and Figure 4.26 report the results of an experiment with the virtual agent. The goal is to make the Mindstorms reach a common position in the environment using the projection dynamics equation. In this case the sampling time is equal to $\tau = 0.2s$ and the trajectories have been split in nine phases.

It is possible to compare the results obtained in this experiment with the ones of the correspondent experiment with USB connection. The consistency of the extracted values is higher because with the virtual agent the error produced is lower. Moreover, the distances between EV310 and the other two Mindstorms are negligible due to its null dimensions.

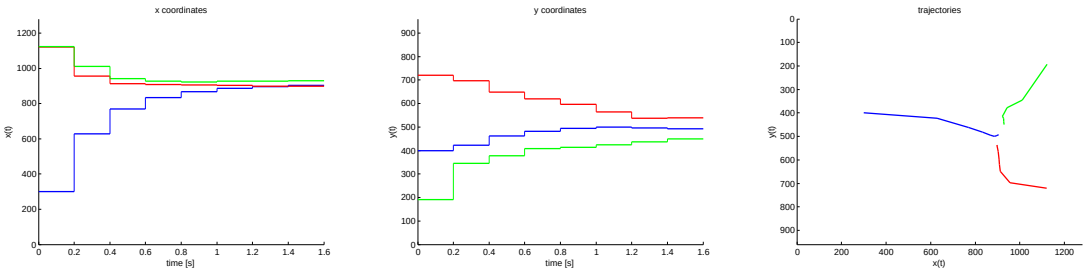


Figure 4.26: PD: rendezvous (Bluetooth).

Table 4.6: Coordinates in pixels of the Mindstorms in projection dynamics rendezvous experiment with Bluetooth connection.

PD rendezvous			
time	EV310	EV309	EV303
$t = 0$	(300,400)	(1121,721)	(1123,192)
$t = \tau$	(629,423)	(956,698)	(1010,346)
$t = 2\tau$	(770,462)	(913,648)	(942,377)
$t = 3\tau$	(833,482)	(909,620)	(926,408)
$t = 4\tau$	(867,495)	(906,596)	(922,414)
$t = 5\tau$	(886,499)	(902,565)	(927,424)
$t = 6\tau$	(897,497)	(898,537)	(928,437)
$t = 7\tau$	(904,493)	(899,539)	(930,449)
$t = 8\tau$	(908,494)	(900,537)	(905,540)

Figure 4.27 shows the sequential positions that the two real Mindstorms occupy during the experiment described in Table 4.6.

Figure 4.27: Projection dynamics rendezvous sequence with Mindstorms (Bluetooth).

4.2.3.2 Formation with PD and Bluetooth

The same experiments of formation performed with 3 real robots, have been done also with two Mindstorms connected with Bluetooth and the third one simulated.

In the triangular formation experiment $\tau = 0.3s$, instead, for the alignment the sampling time is equal to $\tau = 0.2s$. In both cases the motion is divided in six steps. The sides of the triangle and of the segment are always fixed at $\ell = 350$ pixels.

Table 4.7 shows the sequence of coordinates that the robots occupy during the experiments and, as mentioned before, these positions are the ones detected with pattern matching with the camera.

Figures 4.28 and 4.29 show the evolution of the coordinates and the trajectories of the Mindstorms in triangular formation ad alignment, respectively. The numerical values in the graphs are the ones extracted from Table 4.7.

The results can be considered acceptable for practical purposes, as underlined by the final formations that have been obtained.

Table 4.7: Coordinates in pixels of the Mindstorms in projection dynamics triangular formation experiment with Bluetooth connection.

	PD triangular formation			PD alignment		
time	EV310	EV309	EV303	EV310	EV309	EV303
$t = 0$	(300,400)	(952,687)	(1150,185)	(300,400)	(1028,733)	(1136,202)
$t = \tau$	(593,331)	(836,700)	(1028,298)	(543,427)	(894,601)	(1111,429)
$t = 2\tau$	(639,341)	(886,701)	(1058,274)	(622,462)	(889,559)	(1081,437)
$t = 3\tau$	(681,338)	(865,659)	(1055,256)	(662,477)	(878,520)	(1077,459)
$t = 4\tau$	(691,319)	(857,634)	(1044,266)	(683,482)	(882,512)	(1069,472)
$t = 5\tau$	(689,306)	(849,628)	(1064,270)	(695,486)	(881,501)	(1064,473)

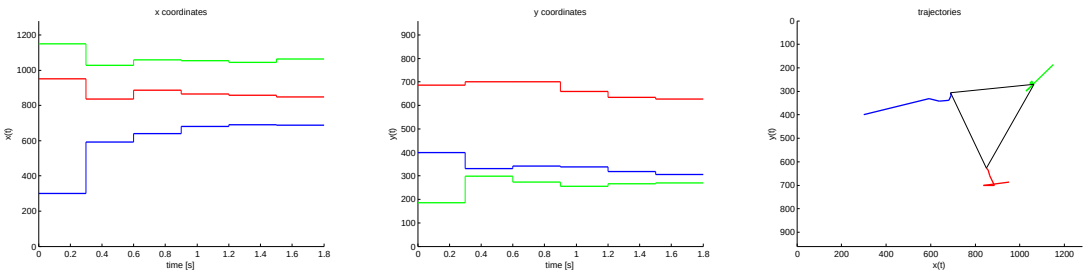


Figure 4.28: PD: triangular formation (Bluetooth).

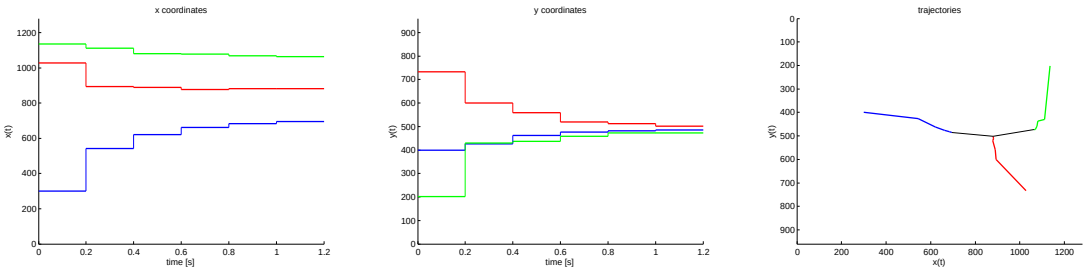


Figure 4.29: PD: alignment (Bluetooth).

In Figure 4.30 is represented the sequence of images captured during the triangular formation experiment performed with projection dynamics equation.

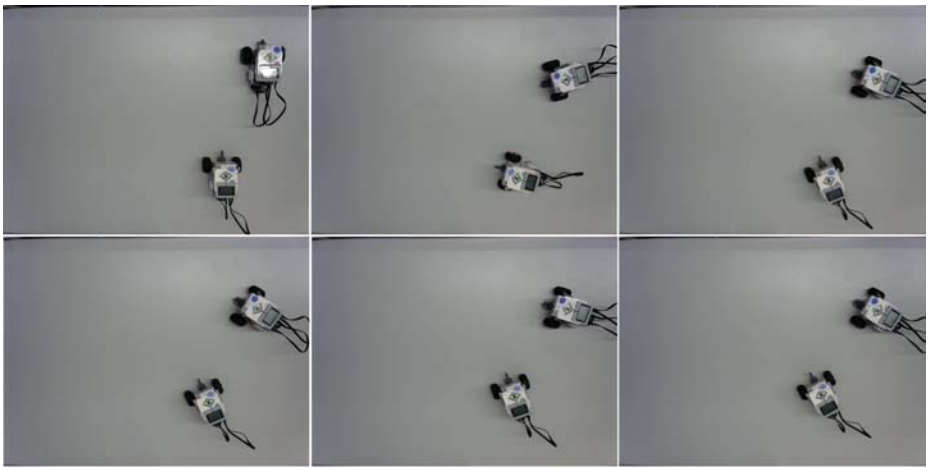


Figure 4.30: Projection dynamics triangular formation sequence with Mindstorms (Bluetooth).

The sequential positions of the Mindstorms in Figure 4.31 correspond to the alignment experiment with projection dynamics described in Table 4.7.

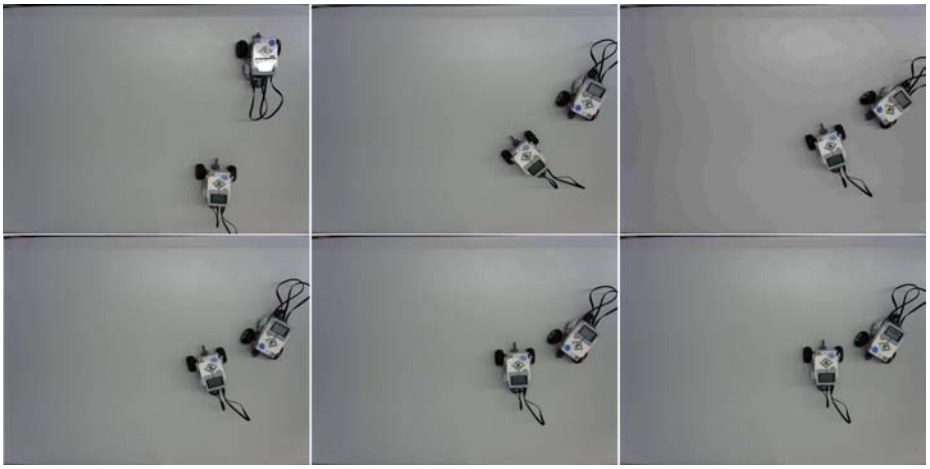


Figure 4.31: Projection dynamics alignment sequence with Mindstorms (Bluetooth).

4.2.4 Replicator dynamics with Bluetooth connection

4.2.4.1 Rendezvous with RD and Bluetooth

All the following experiments apply the discretised equation of replicator dynamics with sampling time $\tau = 0.0001s$.

In rendezvous experiment, the motion has been stopped after eight intervals. The Mindstorms approach gradually the meeting point and this is an indicator of the efficiency of the control but also of the correctness in the choice of the sampling time.

In Table 4.8 and Figure 4.32 are reported the numerical and graphical results of the experiment.

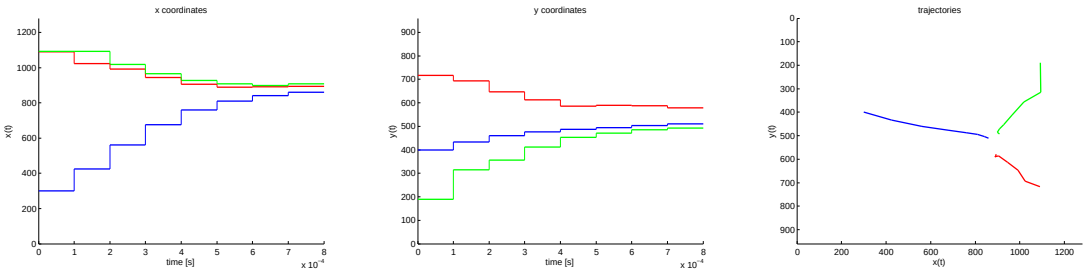


Figure 4.32: RD: rendezvous (Bluetooth).

Table 4.8: Coordinates in pixels of the Mindstorms in replicator dynamics rendezvous experiment with Bluetooth connection.

RD rendezvous			
time	EV310	EV309	EV303
$t = 0$	(300,400)	(1090,716)	(1091,189)
$t = \tau$	(425,434)	(1023,694)	(1093,315)
$t = 2\tau$	(560,460)	(991,646)	(1019,357)
$t = 3\tau$	(677,476)	(944,613)	(966,411)
$t = 4\tau$	(760,487)	(905,585)	(928,453)
$t = 5\tau$	(811,495)	(888,590)	(908,471)
$t = 6\tau$	(840,503)	(892,587)	(899,485)
$t = 7\tau$	(859,511)	(893,579)	(909,492)

Figure 4.33 shows with a sequence of images the real behaviours of the Mindstorms during the experiment reported in Table 4.8.

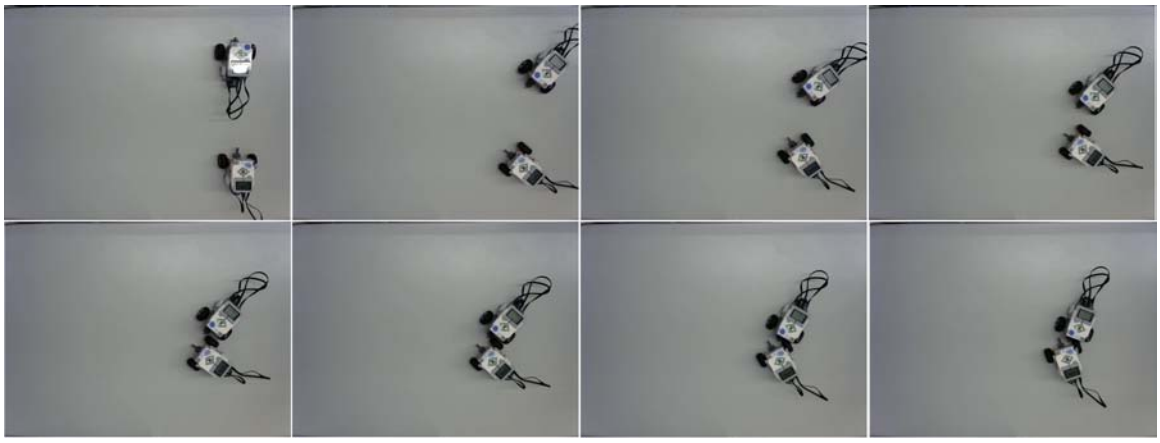


Figure 4.33: Replicator dynamics rendezvous sequence with Mindstorms (Bluetooth).

4.2.4.2 Formation with RD and Bluetooth

In formation experiments the Mindstorms can be left to perfect their positions in order to achieve the goal. For this reason the motion has been stopped after nine or eight steps. From Figure 4.34 can be concluded that in triangular formation would be

enough less steps to reach de final formation, especially regarding the convergence of y coordinates. As a whole, looking at the trajectories and at the numerical values in Table 4.9, the final result is acceptable taking into account the physical nature of the experiment.

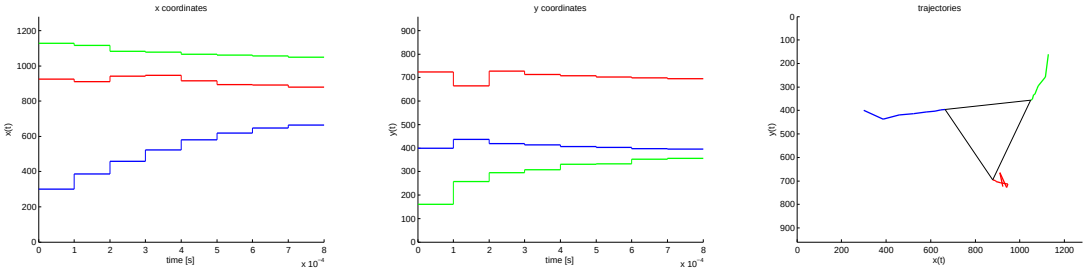


Figure 4.34: RD: triangular formation (Bluetooth).

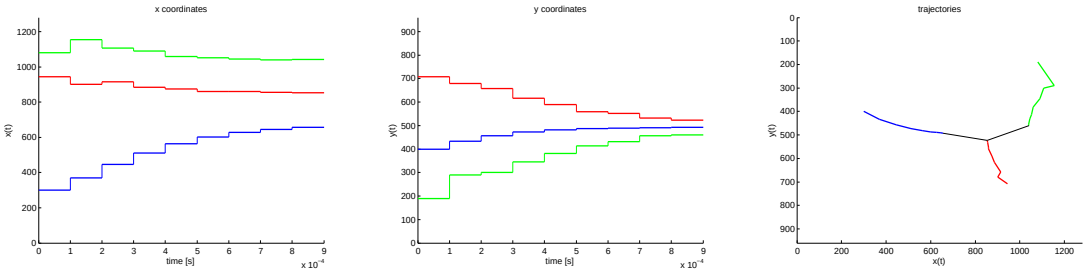


Figure 4.35: RD: alignment (Bluetooth).

Table 4.9: Coordinates in pixels of the Mindstorms in replicator dynamics triangular formation experiment with Bluetooth connection.

	RD triangular formation			RD alignment		
time	EV310	EV309	EV303	EV310	EV309	EV303
$t = 0$	(300,400)	(924,724)	(1128,161)	(300,400)	(945,707)	(1081,190)
$t = \tau$	(387,437)	(911,664)	(1115,257)	(370,433)	(900,680)	(1154,289)
$t = 2\tau$	(458,419)	(941,727)	(1083,296)	(446,456)	(914,657)	(1107,301)
$t = 3\tau$	(523,413)	(946,714)	(1077,307)	(512,473)	(885,617)	(1090,345)
$t = 4\tau$	(579,407)	(914,708)	(1067,331)	(564,482)	(875,590)	(1059,381)
$t = 5\tau$	(619,403)	(894,703)	(1061,332)	(601,487)	(859,559)	(1052,414)
$t = 6\tau$	(646,398)	(891,699)	(1056,353)	(627,489)	(859,552)	(1045,431)
$t = 7\tau$	(665,396)	(879,695)	(1049,356)	(645,491)	(855,532)	(1039,456)
$t = 8\tau$	-	-	-	(656,492)	(853,523)	(1041,460)

In Figure 4.36 is represented the sequence of images captured during the previous triangular formation experiment and, also in this case, the positions of the Mindstorms in the images are the ones reported in Table 4.9.

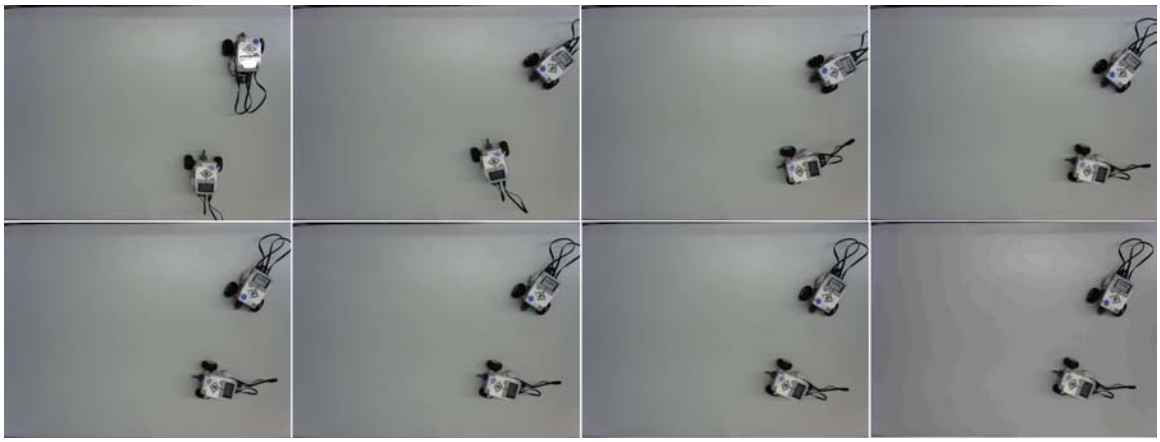


Figure 4.36: Replicator dynamics triangular formation sequence with Mindstorms (Bluetooth).

Finally, Figure 4.37 represents the sequential positions of Mindstorms EV303 and EV309 during the last alignment experiment with replicator dynamics.

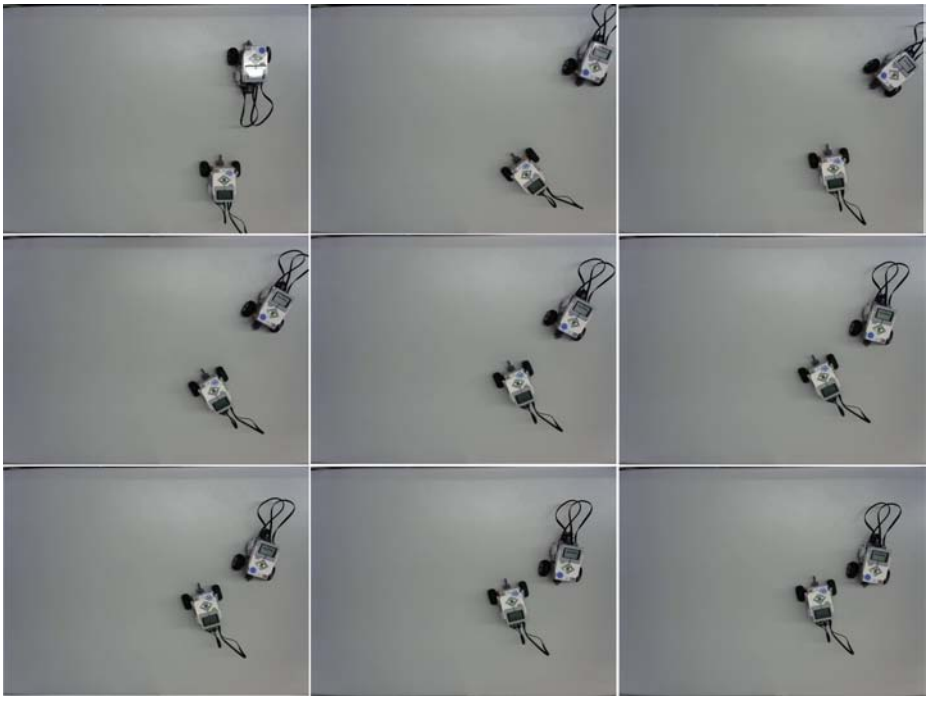


Figure 4.37: Replicator dynamics alignment sequence with Mindstorms (Bluetooth).

Chapter 5

Environmental impact

The term environment is used to indicate groups of people, ecosystems, goods, cultures, socio-economic structures, etc. that constitute the operational framework, not only the environment intended as physical space.

Regarding the effects of integration of mobile robots in the environment, it is necessary to distinguish between interaction and cohabitation with human people and produced pollution in the environment.

In the last twenty years the use of robots in working environment has led to significant changes for what it concerns the productivity. In particular robots led to an increase in both total factory productivity and wages. The use of robots also increased labour productivity and value added from labour In other words, each human worker was more productive and added more value to the economy than before the implementation of industrial robots.

Robots had no effect on the hours worked by high-skilled workers, while low-skilled and middle-skilled workers can be employed in new and hard-less applications such as maintenance and supervising of the robots. If manual and repetitive jobs are occupied by the robots, human workers can improve their positions in the workplace with a consequent increase of salary.

Mobile robots can also be introduced to assist humans and in such a way it is possible to reduce fatigue, increase precision, and improve quality; whereas the human can bring experience, global knowledge, and understanding to the execution of tasks.

During an assistance task, the robot must be capable of performing basic autonomous operations involving both navigation and manipulation. For more elaborate and delicate operations, the assistant, in its supporting role, must be able to interact and cooperate with the human when performing a guided task.

Regarding the problem of safety inside a workspace shared by humans and robots, the prevention of accidents is entrusted to algorithms of collision avoidance that can be implemented with the use of proximity, pattern matching or movement sensors.

On the other hand, mobile robots generally do not affect the environment with air pollution but the areas in witch the contents of this project can be applied, may employ robots that produce noise and light pollution.

The advantages of using a distributed strategy to control the trajectories of the vehicles are several. Compared with centralised control strategy, the number of connections and elements of the network is reduced and this decreases the impact on the environment where the robots are moving. Moreover, game theoretical approaches are based on the achievement of optimal solution, therefore, the agents can reach the task with less time covering the shortest path, also ensuring energy saving.

Chapter 6

Budget evaluation

This chapter considers the total expense of execution of the project considering the cost of the equipment, licences and human resources employed. The wage for a student to realise the Master's Thesis Project (TFM) is of 8€/h, that is the indicative price that the Technical University of Catalonia (UPC) fixes as remuneration for a student for the realisation of a traineeships in a company.

1. SOFTWARE		
Licence MATLAB R2014a		1.200,00€
Licence LabVIEW 2012 SP1		1.450€
Licence Texmaker 4.1		0,00€
TOTAL SOFTWARE		1.510,00€
2. HARDWARE		
HP Pavilion dv6 Notebook Laptop		1.200,00€
n.3 Lego Mindstorm EV3		1.049,97€
Webcal Logitech C310		44,99€
TOTAL HARDWARE		2.294,96€
3. SALARY		
Remuneration for a student of TFM	8€/h	
Total number of hours	600h	
TOTAL SALARY		4.800,00€
TOTAL BUDGET		8.604,96€

Chapter 7

Conclusions

7.1 Conclusion

Evolutionary game theoretical distributed techniques are presented to control the motion of a population of robots in order to perform different tasks, such as rendezvous and formation. To an in-depth initial analysis of the state of the art in this field, has followed a research of the most suitable algorithms that can be applied to the specific object of study.

Initially the work is developed on the implementation of the algorithms with MatLab and Simulink with the goal of generating simulations gradually more complex and realistic.

When the simulations have led to desired results, a LabVIEW platform has been developed to apply the algorithms even in real robots experiments.

The platform is constituted of various components such as a PC, a camera and three robots Lego Mindstorms EV3. The connections between the laptop and the other elements are initially established via USB cables and then with Bluetooth. Problems with multiple Bluetooth connections emerged during the experiments, therefore in this case the behaviour of one of the Mindstorms has been simulated with a virtual robot.

The LabVIEW developed routine includes a tool of pattern recognition that permits to localize in real time the positions of the robots in the workplace and, starting from this information, has been developed a control scheme in LabVIEW that reproduces

the one implemented with Simulink but in discrete version.

The discretisation of the model is necessary to generate in a sequential manner the real command to be sent to the brick of the Mindstorms in order to make them cover the right distance in the correct direction.

The practical experiments includes the convergence to a common position using control techniques based on distributed projection and replicator dynamics and also tests of triangular formation and alignment in the space have been done.

7.2 Contributions

The results obtained in this Master's thesis project could be adapted to solve more complex engineering problems with large-scale complex systems. These kind of systems can be partitioned in populations and controlled with distributed techniques like the ones introduced in this work, in order to reduce the complexity of the controller. Each population is modelled as a group of agents which are interconnected accordingly to a graph-based topology that constitute a network, like in the experiments done so far. Populations can be associated to an individual rule and decisions taken by local controllers. The global objective that the entire large-scale system has to reach, is associated to a cost function that regulate the local decisions by imposing resource constraints.

This research gives a contribution to different applications that are modelled as a multi-agent system. In the universe of mobile robots and vehicles, the results obtained can be applied to generate optimal trajectories in order to avoid traffic congestion or to reach targets in the shortest possible time, in particular utilising the distance-based projection and replicator dynamics equations.

However, there is a huge variety of fields that can take advantages to the distributed solutions adopted in the multi-agent experiments. An example can be the problem of sensor coverage, where mobile sensors are modelled as agents of a population and the main objective is to properly assign the motion to the mobile sensors in order to maximise the detection probability.

Furthermore, another alternative application sector can be the one of mobile weapons connected to form a network. The weapons can be controlled in a distributed way to

choose the optimal targets to strike in order to obtain the maximum global damage.

7.3 Further work

Even if the theoretical study and experimental validation have solved many problems in distributed multi-agent coordination and game theoretical approach to this field, there is still a huge number of researching areas that deserve further investigation. Some examples are the following:

- Quantisation effect in distributed coordination problems. Until now, researches have been focused on the study of distributed coordination problems by assuming that both control inputs and measurements are continuous values. The use of digital signal processing techniques, that are more commonly diffused, requires discrete inputs and measurements. Therefore, it would be useful to shift the focus of researches on the quantisation effect in distributed coordination problems.

- Optimisation with combination of individual and global cost functions. The optimisation problem in distributed multi-agent coordination has been studied in the presence of either an individual or a global cost function but in real systems, each agent must comply with both local and global objectives that have their corresponding cost functions. The further studies should be focused on the combination of these objectives in order to investigate the relationship between the individual cost function and the global cost function and also to balance their reciprocal weight.

- Intelligent coordination. An issue which is still an open problem in this field is the understanding of group behaviour in the presence of intelligence. In other world, how to interpret complex networks and stabilise them when intelligent entities are present.

- Centralisation and decentralisation. Decentralisation brings advantages in sense of scalability and robustness of the system. However, a not negligible drawback of decentralisation is the impossibility of single agents or subsystems to predict the group behaviour because they are in possession only of local information. An interesting direction of studies could be to find the optimal way to balance decentralisation and centralisation to improve the system performance.

Concerning the experiments done so far in this particular project, there are several possible directions of development.

An example could be the extension of the population of robots by adding other agents and the consequent enlargement of the environment may require the inclusion of additional cameras to be able to cover the whole area.

At this point, with a larger number of agents involved, it becomes necessary to establish wireless connections between the PC and the Mindstorms and, if Bluetooth technology is not powerful enough, Wi-Fi could be the adapt solution to the problem.

Another important aspect related to generation of trajectories for robots involved in multiple-agents experiments is the guarantee to avoid collisions among them. In the future it could be possible to modify the distributed controllers used so far in order to solve this problem.

Finally, different evolutionary dynamics strategies could be tested to achieve the same tasks, like the BNN dynamics [15], the best response dynamics [33], the logit dynamics [31], and the Smith dynamics [68]. In this context, would be interesting to compare the performances of all these differential equations that describe the dynamics.

References

[1] S. Alizon and D. Cownden. *Evolutionary games and evolutionarily stable strategies*. 2009.

[2] B. D. O. Anderson, J. Lin, and A. S. Morse. "The multi-agent rendezvous problem: An extended summary". In: ed. by Springer. Vol. 309. 2004, pp. 257–282.

[3] G. R. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Ed. by Addison-Wesley. 2000.

[4] M. Athans, D. P. Bertsekas, and J. N. Tsitsiklis. "Distributed asynchronous deterministic and stochastic gradient optimization algorithms". In: *IEEE Transactions on automatic control* AC-31.9 (1986), pp. 803–812.

[5] E.M. Atkins, R.W. Beard, and W. Ren. "Information consensus in multivehicle cooperative control: collective group behavior through local interaction". In: *IEEE Control Sustems Magazines* 27.2 (2007), 71–82.

[6] J. Barreiro-Gómez, G. Obando, and N. Quijano. "Distributed population dynamics: optimization and control applications". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* PP.99 (2016), pp. 1–11.

[7] J. Barreiro-Gómez, C. Ocampo-Martinez, G. Obando, N. Quijano, and G. Riano-Briceno. "Decentralized control for urban drainage systems via population dynamics: Bogotá case study". In: *Proceedings of the European Control Conference* (2015), pp. 2426–2431.

[8] J. Barreiro-Gómez, C. Ocampo-Martinez, and N. Quijano. "Constrained distributed optimization based on population dynamics". In: *Proceedings of the 53rd IEEE conference on Decision and Control (CDC)* (2014), pp. 4260–4265.

[9] J. Barreiro-Gómez, C. Ocampo-Martinez, and N. Quijano. "Distributed control of drinking water networks using population dynamics: Barcelona case study". In: *Proceedings of the 53rd IEEE Conference on Decision and Control* (2014), pp. 3216–3221.

[10] D. P. Bartsekas. *Convex Optimization Algorithms*. Ed. by Athena Scientific. 2015.

[11] R. W. Beard and W. Ren. "Consensus seeking in multiagent systems under dynamically changing interaction topologies". In: *IEEE Transactions on automatic control* 50.5 (2005), pp. 655–661.

[12] D. P. Bertsekas. *Centralized and distributed Newton methods for network optimization and extensions*. Report. Laboratory of Information and Decision Systems (LIDS), MIT, 2011.

[13] D. P. Bertsekas. *Incremental gradient, subgradient, and proximal methods for convex optimization: a survey*. Report. Laboratory of Information and Decision Systems (LIDS), MIT, 2015.

[14] S. Boyd and N. Parikh. "Proximal algorithms". In: *Foundations and Trends in Optimization* 1.3 (2013), 123–231.

[15] G. W. Brown and J. von Neumann. "Solutions of games by differential equations". In: ed. by Princeton University Press. 1950.

[16] F. Bullo and J. Cortés. "Coordination and geometric optimization via distributed dynamical systems". In: *SIAM Journal on Control and Optimization archive* 44.5 (2005), pp. 1543–1574.

[17] F. Bullo, J. Cortés, T. Karatas, and S. Martínez. "Coverage control for mobile sensing networks". In: *IEEE Transaction on Robotics and Automation* 20.2 (2004), pp. 243–255.

[18] F. Bullo, J. Cortés, and S. Martínez. "Motion coordination with distributed information". In: *IEEE Control System Magazine* 27.4 (2007), pp. 75–88.

[19] W. Cao and W. Ren. *Distributed coordination of multi-agent networks. Emergent Problems,models, and Issues*. Ed. by Springer. 2011.

[20] T. Chang, M. Hong, and X. Wang. "Multi-agent distributed optimization via inexact consensus ADMM". In: *IEEE Transactions on signal processing* 63.2 (2015), pp. 482–497.

[21] P. Corke. *Robotics, vision and control*. Ed. by Springer. 2011.

[22] J. Cortés. "Distributed algorithms for reaching consensus on general functions". In: *Automatica* 44.3 (2008), pp. 726–737.

[23] J. Cortés, S. S. Kia, and M. Martínez. "Distributed event-triggered communication for dynamic average consensus in networked systems". In: *Automatica* 59 (2015), pp. 112–119.

[24] J. Cortés, S. S. Kia, and M. Martínez. "Dynamic average consensus under limited control authority and privacy requirements". In: *International Journal of Robust and Nonlinear Control* 25.13 (2014), 1941–1966.

[25] J. Cortés and C. Nowzari. "Distributed event-triggered coordination for average consensus on weight-balanced digraphs". In: *Automatica* 68 (2016), pp. 237–244.

[26] Z. Duan and Z. Li. *Cooperative Control of Multi-Agent Systems: A Consensus Region Approach*. Ed. by CRC Press. 2015.

[27] D. Easley and J. Kleinberg. *Networks, crowds, and markets: reasoning about a highly connected world*. Ed. by Cambridge University Press. 2010.

[28] M. Egerstedt and M. Mesbahi. *Graph theoretic methods for multiagents networks*. 2010.

[29] J. A. Fax and R. M. Murray. "Information flow and cooperative control of vehicle formations". In: *IEEE Transactions on automatic control* 49.9 (2004), pp. 1465–1476.

[30] B. Francis. *Distributed control of autonomous mobile robots*. 2011.

[31] D. Fudenberg and D. K. Levine. *Theory of Learning in Games*. Ed. by MIT Press. 1998.

[32] F. Garin and L. Schenato. "A survey on distributed estimation and control applications using linear consensus algorithms". In: *Network Control Systems*. Ed. by Springer. 2010, pp. 75–107.

[33] I. Gilboa and A. Matsui. "Social stability and equilibrium". In: *Econometrica* 59 (1991), 859–867.

[34] C. R. Graunke, K. A. Intlekofer, S. Mastellone, M. W. Spong, and D. M. Stipanović. "Formation control and collision avoidance for multi-agent non-holonomic systems: theory and experiments". In: *The International Journal of Robotics Research* 27.1 (2008), pp. 107–126.

[35] Z. Guo, G. J. Koehler, and A. B. Whinston. "A Market-Based Optimization Algorithm for Distributed Systems". In: *IEEE Transactions on signal processing* 53.8 (2007), pp. 1345 –1358.

[36] J. Hofbauer and W. H. Sandholm. "Stable games and their dynamics". In: *Journal of Economic Theory* 144.4 (2009), pp. 1665–1693.

[37] J. Hofbauer, P. Schuster, and K. Sigmund. "A note on evolutionary stable strategies and game dynamics". In: *Journal of Theoretical Biology* 81 (1979), pp. 609–612.

[38] J. Hofbauer and K. Sigmund. *The theory of evolution and dynamical systems.* Ed. by Cambridge University Press. 1988.

[39] A. Jadbabaie, J. Lin, and A. S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on automatic control* 48.6 (2003), pp. 988–1001.

[40] A. Jadbabaie, A. Ozdaglar, and E. Wei. "A distributed Newton method for network utility maximization". In: *IEEE Transactions on automatic control* 58.9 (2013), pp. 2162–2175.

[41] A. Jadbabaie, A. Ozdaglar, and M. Zargham. "A distributed Newton method for network utility maximization". In: *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2009).

[42] A. Jadbabaie, G. J. Pappas, and H. G. Tanner. "Flocking in fixed and switching networks". In: *IEEE Transactions on automatic control* 52.5 (2007), pp. 863–868.

[43] J. Jakovetic, J. M. F. Moura, and J. Xavier. "Fast distributed gradient methods". In: *IEEE Transactions on automatic control* 58.5 (2014), pp. 1131–1146.

[44] S. John and A. Zulu. "A review of control algorithms for autonomous quadrotors". In: *Open Journal of Applied Sciences* 4 (2014), pp. 547–556.

[45] L. B. Jonker and P.D. Taylor. "Evolutionarily stable strategies and game dynamics". In: 1978, 145–156.

[46] H. W. Kuhn and A. W. Tucker. "Nonlinear programming". In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press* (1951), pp. 481–492.

[47] S. Lawlor, M. G. Rabbat, and K. I. Tsianos. "Consensus-based distributed optimization: practical issues and applications in large-scale machine learning". In: *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing* (2012), 1543–1550.

[48] N. Li and J. R Marden. "Designing games for distributed optimization". In: *IEEE Journal of Selected Topics in Signal Processing* 7.22 (2013), 230–242.

[49] N. Li and J. R Marden. "Designing games for distributed optimization with a time varying communication graph". In: *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)* (2012), pp. 7764 –7769.

[50] Z. Lin and H. H. T. Liu. "Consensus based on learning game theory". In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference* (2014), pp. 1856–1861.

[51] T. Längle and T. C. Lueth. "A distributed control architecture for autonomous mobile robots - Implementation of the Karlsruhe Multi-Agent Robot Architecture (KAMARA)". In: *Advanced Robotics, International Journal of the Robotics Society of Japan* 4.12 (1997), pp. 411–432.

[52] J. R. Marden and J. S. Shamma. "Game theory and distributed control". In: *Handbook of Game Theory*. Ed. by Elsevier. 2015, pp. 861–900.

[53] S. Mastellone, M. W. Spong, and D. M. Stipanović. "Remote formation control and collision avoidance for multi-agent nonholonomic systems". In: *Proceedings of the IEEE International Conference on Robotics and Automation* (2007), pp. 1062–1067.

[54] D. Monderer and L. S. Shapley. "Potential games". In: *Games and Economic Behavior* 14 (1996), 124–143.

[55] L. Moreau. "Stability of multiagent systems with time-dependent communication links". In: *IEEE Transactions on automatic control* 50.2 (2005), pp. 169–182.

[56] R. M. Murray and R. Olfati-Saber. "Consensus problems in networks of agents with switching topology and time-delays". In: *IEEE Transactions on automatic control* 49.9 (2004), pp. 1520–1533.

[57] A. Nagurney and D. Zhang. "Projected dynamical systems in the formulation, stability analysis, and computation of fixed demand traffic network equilibria". In: *Transportation Science* 31 (1997), 147–158.

[58] A. Nedic and A Ozdaglar. "Distributed subgradient methods for multi-agent optimization". In: *IEEE Transactions on automatic control* 54.1 (2009), pp. 48–61.

[59] A. Pantoja and N. Quijano. "A population dynamics approach for the dispatch of distributed generators". In: *IEEE Transactions on Industrial Electronics* 58.10 (2011), pp. 4559–4567.

[60] A. Pantoja and N. Quijano. "Distributed optimization using population dynamics with a local replicator equation". In: *Proceedings of the 51st IEEE conference on Decision and Control (CDC)* (2012), pp. 3790–3795.

[61] G. R. Prince and J. M. Smith. "The logic of animal conflict". In: *Nature* 246 (1973), pp. 15–18.

[62] D. Qi, J. Zhang, and G. Zhao. "A game theoretical formulation for distributed optimization problems". In: *Proceedings of the 10th IEEE International Conference on Control and Automation (ICCA)* (2013), pp. 1939 –1944.

[63] C. W. Reynolds. "Flocks, herds, and schools: a distributed behavioral model". In: *Computer Graphics* 21.4 (1987), pp. 25–34.

[64] W. H. Sandholm. "Evolutionary Game Theory". In: *Encyclopedia of complexity and system science*. Ed. by Springer. 2009, pp. 3176–3205.

[65] W. H. Sandholm. "Large population potential games". In: *Journal of Economic Theory* 144.4 (2009), 1710–1725.

[66] W. H. Sandholm. "Local stability under evolutionary game dynamics". In: *Theoretical Economics* 5.1 (2010), pp. 27–50.

[67] W. H. Sandholm. *Population games and evolutionary dynamics*. Ed. by MIT Press. 2010.

[68] J. M. Smith. *Evolution and the theory of games*. Ed. by Cambridge University Press. 1982.

[69] S. Tadelis. *Game Theory. An Introduction*.

[70] *The Law of Rule: Centralized, Decentralized and Distributed Systems*. Prepared for CFFN, NRN-Canada, and NRNA as an input to the constitutional development process in Nepal. 2009. URL: `http://www2.cffn.ca/usha/part-iii-article-by-pramod-dhakal/129-the-law-of-rule-centralized-decentralized-and-distributed-systems`.

[71] J. N. Tsitsiklis. "Problems in decentralized decision making and computation". PhD Thesis. Massachusetts Institute of Technology, 1984.

[72] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, and O. Schochet. "Novel type of phase transitions in a system of self-driven particles". In: *Physical Review Letters* 75.6 (1995), 1226–1229.

[73] E. C. Zeeman. "Population dynamics from game theory". In: *Global theory of dynamical systems*. Ed. by Springer. 1980, pp. 471–497.

[74] X. Zhang. "The Laplacian eigenvalues of graphs: a survey". In: *Linear Algebra Research Advances*. Ed. by Gerald D. Ling. 2007, pp. 203–230.