الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

جامعة سعد دحلب البليدة
University of SAAD DAHLAB- BLIDA

كلية التكنولوجيا
Faculty of Technology

قسم الإلكترونيك
Department of Electronics

Masters Dissertation

Sector: Automation

Specialty: Automation And Industrial Computer science

Presented by:

TAIBI MOHAMED SALAH

&

ZERRARI NABIH

Mobile Robot Formation Control

Proposed by: Pr. BENSLAMA ZOUBIR & Mr. AMROUCHE MAHFOUD

College year: 2020-2021

Acknowledgements

Praise be to Allah. We seek refuge in Allah from the evils of ourselves and from the evils of our actions. Whomever Allah guides is not misguided nor mislead. We bear witness that there is no god but Allah alone, and He have no partner. Moreover, we bear witness that Muhammad is His Servant and His Messenger.

Our sincere gratitude and deep thanks to **Mr. AMROUCHE MAHFOUD** and

Pr. BENSLAMA ZOUBIR for directing us along our work with educational availability, also for their effective advice and encouragement, as well as for the great sense of intellectual property.

We also thank **Dr.MOHAMED LAMINE FAS** for his invaluable help and all the teachers in the electronics department who contributed to our education. We wish to express our sincere thanks and our deepest gratitude to all those who have contributed from near or far to the realization of this modest work. We sincerely appreciate all the members of the jury for taking an interest in our work, and for enriching it with their suggestions and proposals.

Dedications

Praise be to ALLAH who endowed us with the wonderful gift of reasoning.

Praise be to our Creator who prompted us to acquire knowledge.

I dedicate this work

To my Parents who supported and encouraged me during these years of study.

May they find here the testimony of my deep gratitude.

To my brother, my sister, my grandparents and those who shared with me all the emotional moments while carrying out this work. They have warmly supported and encouraged me throughout my journey.

To my family, loved ones and those who give me love and liveliness.

To all my friends who have always encouraged me, and to whom I wish more success.

To all those I love

Taibi Mohamed Salah

Dedications

بسم الله الرحمن الرحيم و الصلاة و السلام على الرسول الله محمد اما بعد.

نشكر الله عز وجل على توفيقه لنا لقيام بهذا العمل و نسال الله عز وجل ان يوفقنا للقيام بما هو اكثر .

We thank our family, specially our mothers and fathers for their support from the beginning of our studying phase. We thank also our professor Benslama and Mr. Amrouch for their orientation and help in this dissertation, and we hope we reach their expectations. Finally the staff of the university, which make this journey so easy for us.

Ferrari Nabih

ملخص: تحتوي الأنظمة متعددة الروبوتات على العديد من التطبيقات المحتملة في مجالات مختلفة مثل تتبع الهدف ، المراقبة البيئية ، الاستجابة للطوارئ والإنقاذ ، الرعاية المنزلية ، مراقبة الموارد الطبيعية ، والعمليات الصناعية الخارجية مثل تشخيص الأخطاء وإصلاحها.

يركز هذا المشروع على التحكم في مجموعة من الروبوتات المتنقلة غير الشاملة بناءً على حركة قائد-تابع لاتباع المسار المطلوب في بيئة ما ، مع الحفاظ على التكوين الهندسي المحدد مسبقًا.

كلمات المفتاحية: الأنظمة متعددة الروبوتات ؛ الروبوتات المتنقلة غير الشاملة ؛ قائد-تابع .

Résumé : Les systèmes multi-robots ont de nombreuses applications potentielles dans divers domaines tels que le suivi des cibles, la surveillance de l'environnement, les interventions d'urgence et le sauvetage, les soins à domicile, la surveillance des ressources naturelles et les opérations industrielles en plein air telles que le diagnostic et la réparation des pannes.

Ce projet se concentre sur le contrôle d'un groupe de robots mobiles non holonomiques basés sur le mouvement leader-suiveur pour suivre une trajectoire souhaitée dans un environnement, tout en conservant une configuration géométrique prédéfinie.

Mots clés : Systèmes multi-robots; Robots mobiles non holonomiques; Leader-suiveur.

Abstract: Multi-robot systems have many potential applications in various fields such as target tracking, environmental monitoring, emergency response and rescue, homecare, natural resource monitoring, and outdoor industrial operations such as fault diagnosis and repair.

This project focuses controlling a group of non-holonomic mobile robots based on the leader-follower motion using “Lyapunov’s Direct Method” to follow a desired trajectory in an environment, while maintaining a predefined geometrical configuration.

Keywords: Multi-Robot Systems; Non-Holonomic Mobile Robots; Leader-Follower; Lyapunov’s Direct Method.

List of acronyms and abbreviations

MRS: Multi Robot Systems

UAV: Unmanned Air Vehicles

AUV: Autonomous Underwater Vehicles

MAS: Multi Agent Systems

P.E: Potential Energy

SSC: Separation – Bearing Controller

SBC: Separation – Separation Controller

RWS: Rolling Without Slipping

ICR: Instantaneous Center of Rotation

EKF: Extended Kalman Filter

SVSF: Smooth Variable Structure Filter

Table of Contents

List of Figures

List of tables

General Introduction

The cooperative control of mobile Multi-Robot Systems is one of the most important themes of this generation of research, due to the wide array of applications, such as rescue missions, transporting large objects, surveillance, sensor networks, cooperative transport, manufacturing, waste management, space exploration and military transport ... etc.

The prospects for multi-robot systems have multiplied in recent years. Moreover, MRS are more flexible, more adaptable, more scalable, and more affordable than single robots, which can be increased by modularization **[1] [2]**. In fact, MRS are useful not only when the robots can accomplish different functions, but also when they have the same capabilities **[1] [3]**.

One of the main areas of research in order to achieve successful autonomous robots with the positive characteristics mentioned above is Path Planning. It is defined as the optimal collision-free navigation for the autonomous vehicles to be able to maneuver from a source to a desired destination. In this document, we will use the leader-follower approach where the leader robot moves along a predefined trajectory, on the other hand, the follower robots track the path of the leader while maintaining a predefined spacing between him and the leader. Using this formation gives us with two main options regarding the distribution structures of the tasks between the leader and the followers. The first one is to let the leader control the formation and make all the decisions and calculations using the data and informations sent from the followers, the second structure focuses on distributing the tasks between the leader and the followers (the leader is given the task of navigation and path planning, while the tasks of followers involve following the leader, collecting data and managing communication).

Moving a group of robots in a desired formation would be beneficial in many real world applications that would reduce the human interference in dangerous environments such as search and rescue missions during natural disasters, battlefield exploration missions, planetary missions, and minesweeping. It is also used in industrial applications including

factory floor, resource monitoring, transporting large objects, conveying and outdoor industrial operations such as fault diagnosis and repair.

Chapter I: Autonomous Multi-Robot Systems

I.1-Introduction

Our main objective in this dissertation is to build a stable control architecture that will allow us to maintain a desired formation in a busy work environment; for example, Robots (R1 to R6) will maintain the pre-defined formation as it is illustrated in **(Figure1.1)**. We want to make this architecture as flexible as possible

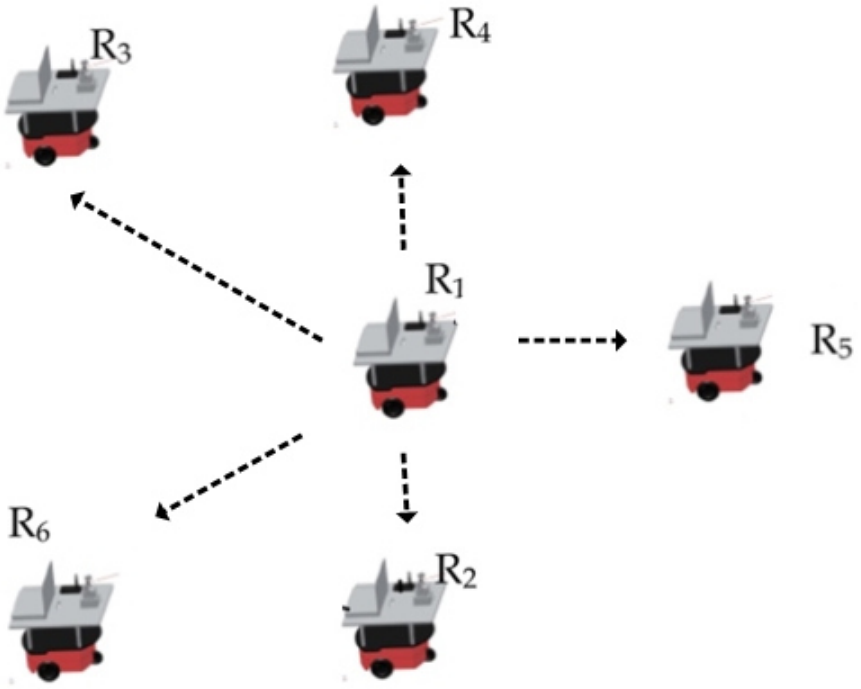
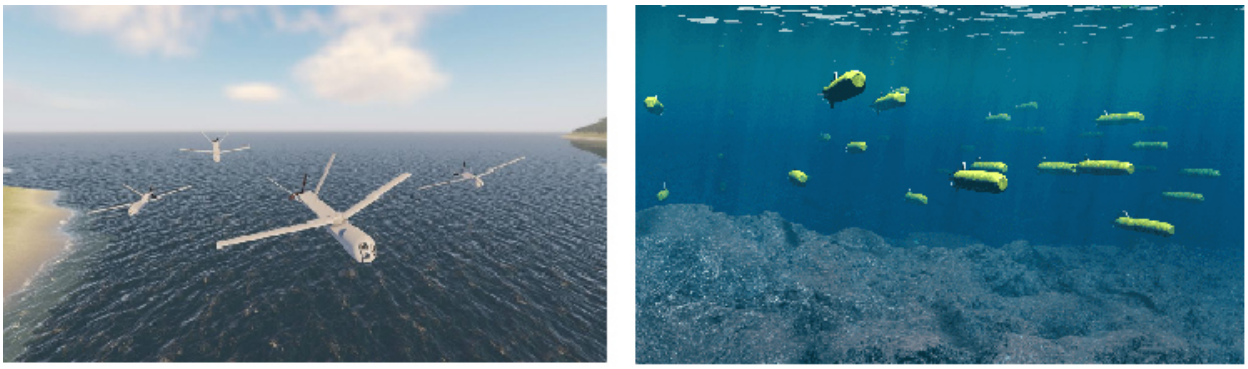


Figure 1. 1: A Multi Robot System

Multi-Robot Systems are the subject of many studies in many application domains in the recent decades due to its utility and flexibility in the work environment. As a matter of fact, these systems have many potential advantages over a single robot, including performing tasks that cannot be performed with a single robot and it allows a high level of flexibility in the execution of tasks, improving the efficiency of tasks in terms of time and quality, in addition to high adaptivity and easier maintenance.

Moreover, a MRS has a better spatial distribution and can achieve better overall system performance (total time required to complete a task **[4]**, the energy consumption of the robots **[5]**), it also introduces robustness that can benefit from data fusion and information sharing among the robots, and fault-tolerance that can benefit from information redundancy. For example, multiple robots can localize themselves more efficiently if they exchange information about their position whenever they sense each other **[6] [7] [8]**. Finally, these systems (MRS) have a lower cost, Using a number of simple robots can be simpler to program and cheaper to build than using a single robot that is complex and expensive to accomplish a task.

Using formation of multiple robots to accomplish an objective offers obvious advantages. These include increasing feasibility, accuracy, robustness, flexibility, cost and energy efficiency that is why it is used in many real world applications such as Unmanned Air Vehicles (UAV) and Autonomous Underwater Vehicles (AUV) (**Figure 1.2**).



(a) (b)

Figure 1. 2: Examples of formations in real world applications; (a) UAV, (b) AUV

We can notice that the idea of cooperation in Multi-Agent Systems is largely inspired by nature and the world around us. In these very powerful systems, individuals follow distant leaders without hitting their neighbors, including, but not limited to, schools of fish **(a)**, swarms of bees **(b)**, flocks of birds **(c)** and teams of ants **(d)** etc… (**Figure 1.3**).

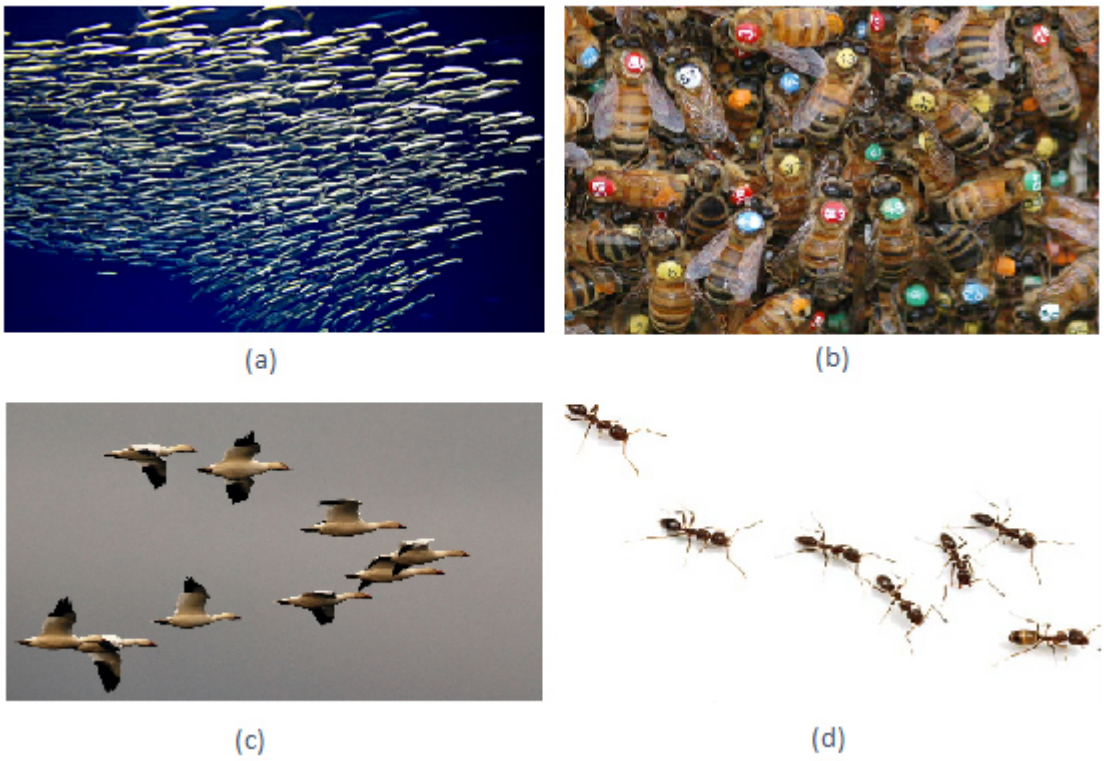


Figure 1. 3: Examples of formations in the nature

I.2-Fundamental analysis perspectives

The study of Multi-Agent Systems (MAS) has taken a large interest in the scientific community in the last two decades **[9] [10] [11]** , the use of MAS is particularly appropriate in situations where different robots need to coordinate and possibly cooperate in order to achieve individual objectives or a shared goal. Coordination typically takes place when it comes to using a resource in turn, or to order the actions performed by each robot. Cooperation goes further in the sense that the robots share at least a subset of their goals and coordinate to achieve them, Based on **[3]**, there are two fundamental analysis perspectives: control structures and control approaches.

In the following, we shall present a general overview of main notions concerning these two perspectives.

I.2.1-Control Structures

The robustness of multiple mobile robot systems is related to the control structure that organize the robots and to obtain the desired formation behaviors. When we talk about mobile robots formation control, the control structures can be identified as centralized control structure or distributed control structure.

I.2.1.a-Centralized control structure

In a centralized control model, an agent (Leader Robot) is in charge of organizing the work of the other agents; the leader is involved in the decision process for the whole team, while the other members can act only according to the directions of the leader. The classification of centralized systems can be further refined depending on the way the leadership of the group is played. Specifically, Strong centralization is used to characterize a system in which the same pre-defined leader agent takes decisions during the entire mission duration, while in a weakly centralized system more than one agent is allowed to take the role of the leader during the mission.

The advantages of a centralized structure typically include faster convergence and enhanced stability. These benefits come with a greater financial cost due to the required

processing and communications resources needed by the single computational unit. Although these guarantee a complete solution, centralized control schemes require higher computation power and are less robust due to the heavy dependence on a single controller. Additionally, architectures involving a single computational unit typically do not work well for large systems due to limited communication range and limited processing power of the single computational unit.

I.2.1.b- Distributed control structure

This structure is the most used structure to control multi-robot systems. It is composed of agents, which are completely autonomous in the decision process with respect to each other, in this class of systems each robot acts based only on knowledge of local teammate's state and of the state of the environment.

This structure is typically more budget friendly and works better for larger systems than a centralized structure. However, this approach can result in slower convergence and reduced stability.

There are a lot of related results about formation control using distributed control structure we take for example **[8] [11] [12] [13]**.

I.2.2-Control Approaches

There are various strategies and approaches, which can be roughly categorized as virtual-structure, graph theory, behavior-based approach, artificial potential and leader-follower approach have been emerged for the formation control of multiple mobile robots.

I.2.2.a-Virtual structure

The virtual structure approach treats the complete formation as one rigid entity [14]. Desired motion is allotted to the virtual structure as a whole; and therefore the trajectories for every robot to follow are outlined. The virtual structure will evolve as an entire body in any direction with some given orientations and maintaining a rigid geometric relationship among cluster members.

The advantage of this approach is that it is easier to explain the coordinated behavior for the cluster of formation, however the biggest disadvantage of the virtual structure's implementation is centralization that results in a single point of failure for the entire system.

I.2.2.b-Graph Theory

Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called links or lines). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. Graphs are one of the principal objects of study in discrete mathematics.

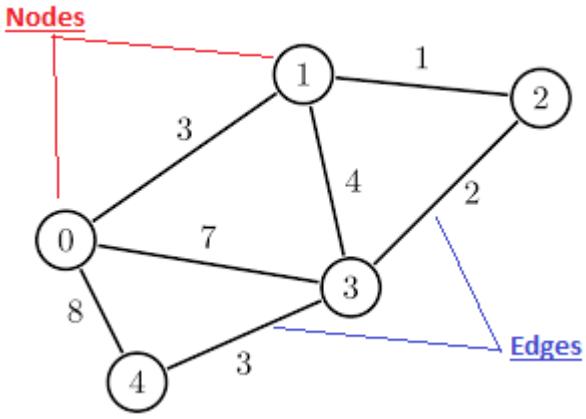


Figure 1. 4:General representation of a graph

In the graph theory approach, every robot is taken into account as a node, and every communication or sensing info link between robots is represented as an edge. The analysis approach uses graph theory, control theory, and dynamic systems theory along to review the formation controller and its stability.

Using this approach makes the representation of any formation easy, which has well-developed and stable results **[15]**. The disadvantage is that it is difficult to fully consider all the limitations of the real robot configuration.

I.2.2.c- Behavior-based

The basic idea of the behavior-based approach is to dictate many desired behaviors for every robot (for example: goal seeking, obstacle avoidance, collision avoidance and formation keeping), and so the final action of every robot is derived by weighting the relative importance of all the behavior. A behavior is implemented, as a sub-controller for achieving a particular goal, which makes this technique, is appropriate for a large group of mobile robots.

The advantages of this sort of approach are its parallel, distributed and real-time characteristics, and fewer data must be communicated among robots. Therefore, it is very helpful to guide a multiple mobile robots, the limitation of the behavior-based approach is that it is troublesome to investigate the system's performance mathematically. Thus, it is hard to ensure a precise formation control.

I.2.2.d- Artificial potential

Using this approach, each body within the environment produces a special kind of Potential energy (P.E); the target point produces an attractive force that pulls the robot toward the target point. On the other hand, the obstacles within the environment produces a repulsive force that pushes the robot far from them. The repulsive and attractive forces are delineated as a repulsive potential function and an attractive potential function. These functions are used together in practical applications to satisfy the convergence, collision-free and obstacle-free: the robot will move on the direction that minimize the P.E.

The advantages of this approach are that it needs fewer calculations, and can be used for real-time control applications. The drawbacks is that it is tough to design potential field functions satisfying native minimums.

I.2.2.e- Leader-Follower

In the leader-following approach, one or a few robots are designated as the leader/leaders, with the rest being followers. The follower robots need to position themselves relative to the leader and maintain a desired relative position with respect to the leader.

To prescribe a formation maneuver we only need to specify the leader's trajectory and the desired relative positions and orientations between leader/leaders and followers. When the motion of the leader is known, the desired positions of the followers relative to the leader can be achieved by local control of each follower. Therefore, in a certain sense, the formation control problem can be essentially viewed as an extension of the traditional trajectory-tracking problem. According to scientific literature **[16] [17]**, two modeling methods can be used to generate the control algorithm: "Separation-Bearing Controller (SBC)" and "Separation- Separation Controller (SSC)".

The SBC is used for two robots. The follower "f" follows the leader"l" while maintaining a desired relative distance and separation-bearing angle with respect to the leader robot. Such type of leader-follower formation control strategy is also denoted by"$l-\varphi$" control strategy.

$\boldsymbol{\varphi}$**:** The separation-bearing angle between the leader and follower robot.

$\boldsymbol{l}$**:** The separation distance between the center of axis between the rear wheels of the leader, and the front castor of the follower robot.

$(\boldsymbol{x}, \boldsymbol{y})$**:** The position coordinates for the front castor of the follower robot.

$\boldsymbol{d}$**:** The distance between the front castor and the center of axis between the rear wheels for each robot.

$\boldsymbol{(x_l, y_l)}$**:** represent the mid-point on the axis between the rear wheels.

The leader robot pose is expressed by $\boldsymbol{p_l} = [\boldsymbol{x_l}, \boldsymbol{y_l}, \boldsymbol{\theta_l}]^{\boldsymbol{T}}$.

The follower robot pose is represented by $\boldsymbol{p_f} = [\boldsymbol{x_f}, \boldsymbol{y_f}, \boldsymbol{\theta_f}]^{\boldsymbol{T}}$.

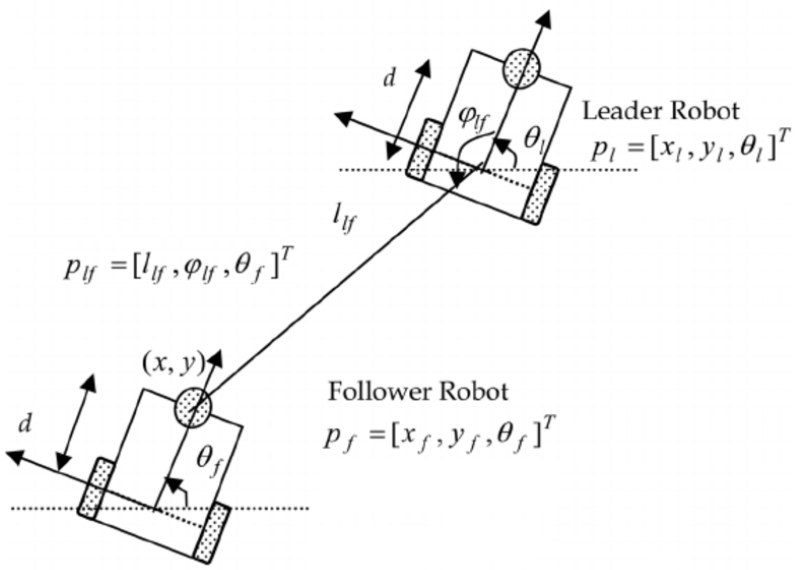


Figure 1. 5: Representation of the SBC [18]

The SSC is used when multiple robots are present in the formation. Such type of control strategy is also denoted by $l-l$. In the $l-l$ formation strategy, the leader robot 2 is actually a follower relative to leader robot 1. The leader robot 2 can be modeled using $l-\varphi$ controller. The follower robot can be expressed relative to the leader robot1 and leader robot2 as $\boldsymbol{p_f} = [\boldsymbol{l_{1f}}, \boldsymbol{l_{2f}}, \boldsymbol{\theta_f}]^{\boldsymbol{T}}$. In the $l-l$ control strategy, the aim is to maintain the desired lengths l_{2f}^d and l_{2f}^d with respect to leader robot 1 and leader robot 2.

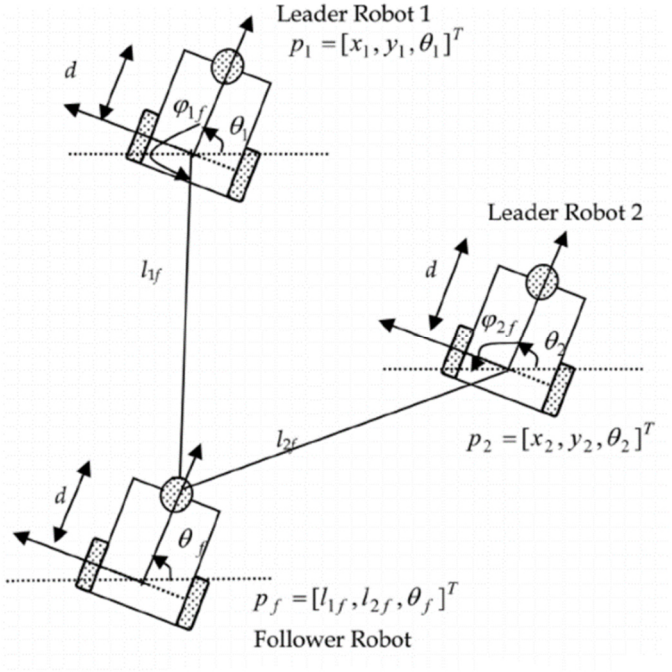


Figure 1. 6: Representation of the SSC [18]

The leader-follower approach has become popular because it can be explored empirically and the stability of the formation can be theoretically guaranteed.

We will see in the following chapters that we try to make the proposed control architecture as distributed as possible using the Leader-Follower approach.

I.3-Multi-Robot Systems

Real-world applications that are ideal for robotic solutions are very complicated and difficult. Several of those applications are set in dynamic environments that require capabilities distributed in functionality, space, or time. These applications, therefore, typically need groups of robots to work together cooperatively to successfully complete the mission.

Nowadays, extensive research is focused on multi-robot systems. These systems offer many advantages, as they have a high potential to solve a problem. Multitude of problems such as industrial warehouses, surveillance and missions of search and rescue. They can even be used in underwater missions such as seabed mapping, the recovery of stranded beacons or even black boxes following the spitting of planes. Multi-robot systems offer high reliability and performance for complex tasks, even if their design is simple.

Figure 1. 7: Examples of Real- world applications of Multi-Robot systems

I.3.1-Mobile Robot control modules

The control of a mobile robot naturally requires the total or partial perception of the environment in which it is located, as well as its location in this environment, in order to decide on the action to be taken (**Figure 9**). We then retain three main modules that are essential to the control of the mobile robot: perception, decision-making and action.

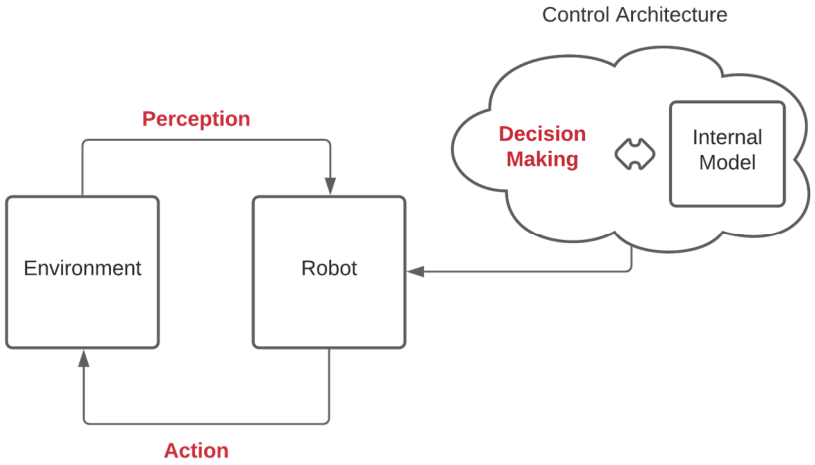


Figure 1. 8: Robot interactions with the environment

I.3.1.a- *Perception*

In the mobile robot, the perception module is responsible for transforming the data, coming from the sensors (camera, ultrasonic sensor, infrared sensor, laser sensor… etc.), into symbolic data. It also contains modules for reaching a point or following a trajectory, or else avoiding an awkward obstacle, the mobile robot generally has sensors making it possible to recover appropriate stimuli in order to feed a model and produce a representation of this environment.

I.3.1.a- *Decision making*

The autonomy of the mobile robot is a faculty that allows it to adapt or to make a decision in order to reach a point of known coordinates in a known environment. The decision is made according to the task to be accomplished in the context of execution as a function of the local

information of the environment delivered by its own sensors and of its communication with its neighbors (determined by elements of perception).

I.3.1.a- *Action*

The mobile robot can be represented by a classic closed loop control diagram (**Figure 1.9**). And the control law block generates the command to be applied to the actuators of the actuator block as a function of the information supplied by the sensors (perception) and set point (decision).

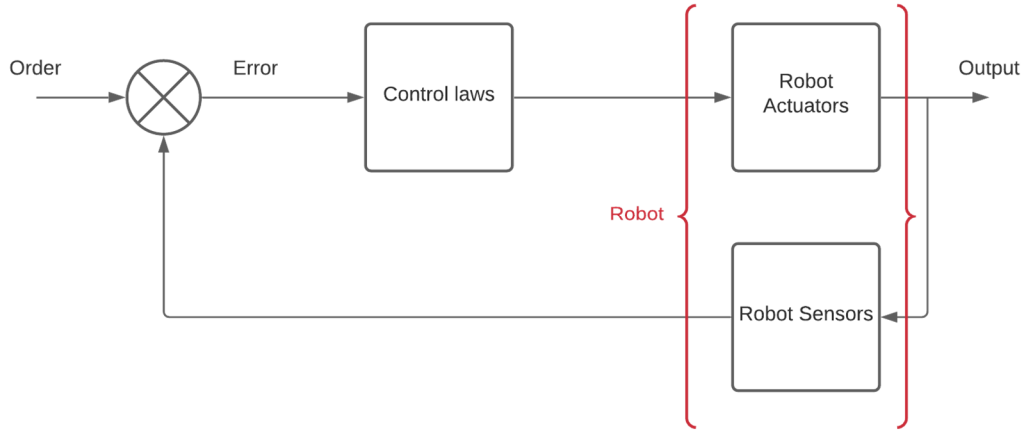


Figure 1. 9: Classic closed loop control diagram for a robot

I.3.2-Modelling

Usually for the control of mobile robots, a speed control model is used rather than a torque control model. The main reasons for this choice are as follows:

- The computation of the order is simpler for a kinematic model than for a dynamic model.
- There are no complicated geometric or inertial parameters to identify for a kinematic model.

For these reasons, we only consider kinematic models thereafter, taking into account the following simplifying assumptions:

- The mobile robot is considered as a rigid vehicle moving on a plane horizontal.
- Conventional wheels are assumed to be dimensionally stable and of radius r.
- Each wheel / ground contact is reduced to one point.
- The wheels roll without slipping on the ground.

Keep in mind! In reality:

- Contact is made on a surface.
- Undeformability hypothesis is false with wheels fitted with tires (made from synthetic rubber, natural rubber, fabric and wire).
- Continuous contact with the ground: essential for the odometry of a robot.

I.3.2.a-Setting up frames

We denote $R_g(O, x_g, y_g, z_g)$ a fixed global frame whose z axis is vertical, and $R_l(P_i, x_l, y_l, z_l)$, a mobile local frame linked to the robot. A remarkable point on the robot's platform is generally chosen for P_i typically the center of the axis of the drive wheels if there is one, as illustrated in (**Figure 1.10**).

By analogy with manipulation, we call the robot's situation, or often posture [12], the vector:

$$\xi_i = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \qquad \textbf{1 - 1}$$

The configuration of a mobile robot is known when the position of all its points in a given frame of reference is known, the configuration of the mobile robot will be defined by a vector:

$$q = \begin{pmatrix} q_1 \\ \vdots \\ \vdots \\ q_n \end{pmatrix} \qquad \textbf{1 - 2}$$

the n coordinates called generalized coordinates. The configuration is thus defined on a space N of dimension n, called the space of the configurations.

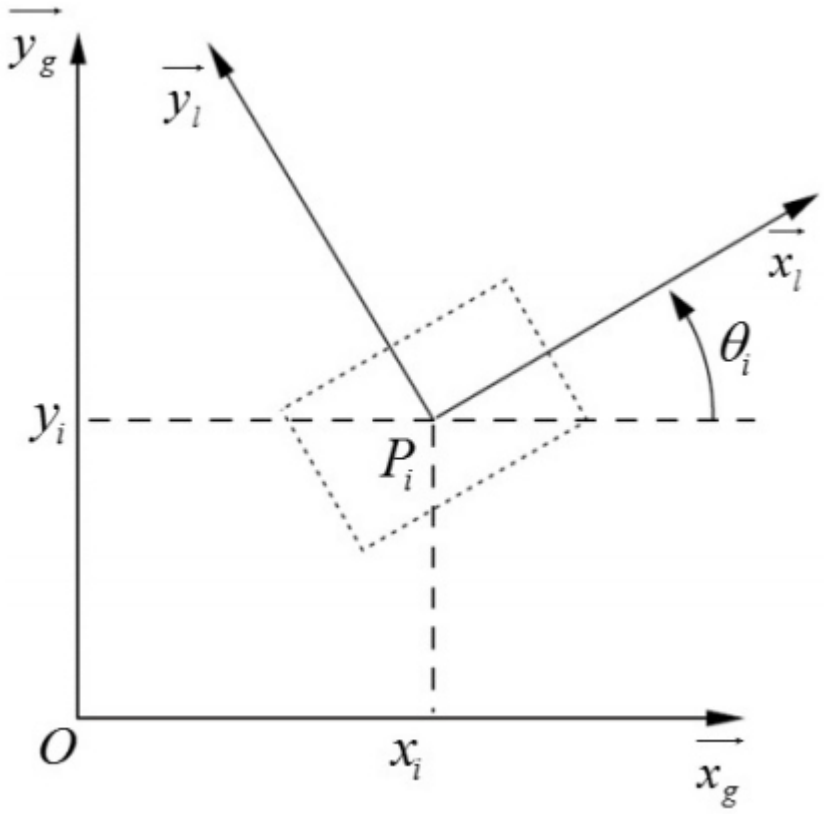


Figure 1. 10: Location of a mobile robot

I.3.2.b-Rolling without slipping (RWS) and Non-holonomic constraints

Locomotion occurs through friction between the vehicle's wheel and the ground, and the efficiency of movement depends in particular on the type of ground. For the hypothesis of non-slip rolling to be validated, it is theoretically necessary that the wheel / ground contact only occurs at one point, that the ground is perfectly flat and that the radius of the wheel is perfectly constant over its entire periphery **[21]**. That is, the relative speed of the wheel compared to the ground at the point of contact is null. Theoretically, to verify this condition, it is necessary to meet the following hypotheses:

- Each wheel is assumed dimensionally stable and the wheel to ground contact area is assumed point.
- The linear speed of the point of contact of a wheel with the ground is zero.

However, practically the contact is made on a surface, which generates slight slips. Likewise, the assumption that solid wheels cannot deform is largely false, especially when the wheels are fitted with tires.

Mathematically, we can translate the condition of RWS on a wheel. Let $P_r\ (x, y, r)$ be the center of the wheel, $Q_r(x, y, 0)$ the point of contact of the wheel with the ground, φ is the proper angle of rotation of the wheel and θ the angle between the plane of the wheel and the plane $(O, \vec{x}, \vec{y})$ as indicated in (**Figure 1.11**)

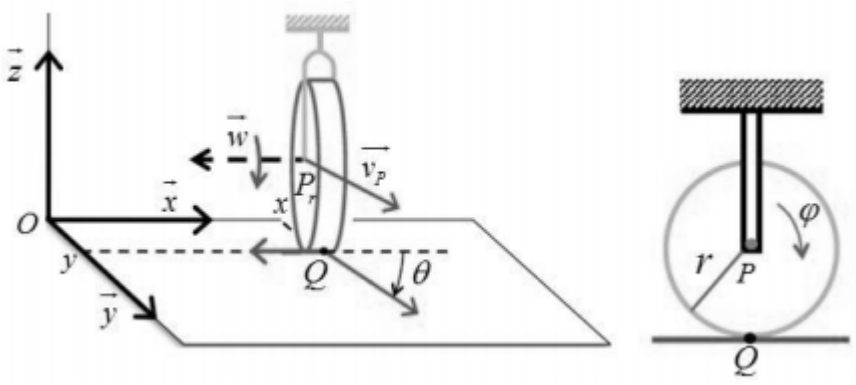


Figure 1. 11: RWS properties

The nullity of the relative speed $\overrightarrow{v_Q}$ wheel / ground at the point of contact makes it possible to obtain a vector relation between the speed $\overrightarrow{v_{P_r}}$ of the center P_r of the wheel and the vector of rotation speed $\vec{w}$ of the wheel:

$$\overrightarrow{v_Q} = \overrightarrow{v_{P_r}} + \vec{w} \times \overrightarrow{P_r Q} = 0 \qquad \textbf{1 - 3}$$

If we use the expression of the pointsP_r. Q, we find:

$$\dot{x}\vec{x} + \dot{y}\vec{y} + (\dot{\theta}\vec{z} + \dot{\varphi}(\vec{x}\sin\theta - \vec{y}\cos\theta))\Lambda(-r\vec{x}) = \vec{0} \qquad \textbf{1 - 4}$$

If we calculate the cross product of equation (**1 - 4**), we have that:

$$(\dot{x} + r\dot{\varphi}\cos\theta)\vec{x} + (\dot{y} + r\dot{\phi}\sin\theta)\vec{y} = \vec{0} \qquad \textbf{1 - 5}$$

This gives us the following scalar constraint system:

$$\begin{aligned} \dot{x} + r\dot{\varphi}\cos\theta &= 0 \\ \dot{y} + r\dot{\phi}\sin\theta &= 0 \end{aligned} \qquad \textbf{1 - 6}$$

Where $\dot{\varphi}$ is the angular speed of the wheel and r is the radius of the wheel.

We can transform these constraints to show the speed components in the plane of the wheel on the one hand and perpendicular to the wheel on the other hand:

- **Slipping constraints:**

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \qquad \textbf{1 - 7}$$

- **Rolling constraints:**

$$\dot{x}\cos\theta + \dot{y}\sin\theta = r\dot{\varphi} \qquad \textbf{1 - 8}$$

The kinematic model of a mobile robot can be obtained from the non-holonomic constraints, can be described as:

$$\begin{cases} \dot{x} = v\,cos\theta \\ \dot{y} = v\,sin\theta \\ \dot{\theta} = \omega \end{cases} \qquad \textbf{1 - 9}$$

We write it in matrix form:

$$\dot{\xi} = \begin{pmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} v \\ w \end{pmatrix} = C(q).u \qquad \textbf{1 - 10}$$

We will be more particularly interested in the non-holonomic constraints characterizing among others the movements of a mobile robot of the unicycle type.

I.3.2.c-Kinematic model of a unicycle robot

A unicycle type robot is powered by two independent wheels. Its center of rotation is located on the axel connecting the two driving wheels. It is a non-holonomic robot, in fact, it is impossible to move it in a direction perpendicular to the wheels of locomotion. Its control can be very simple; it is quite easy to move it from one point to another by a series of simple rotations and straight lines.

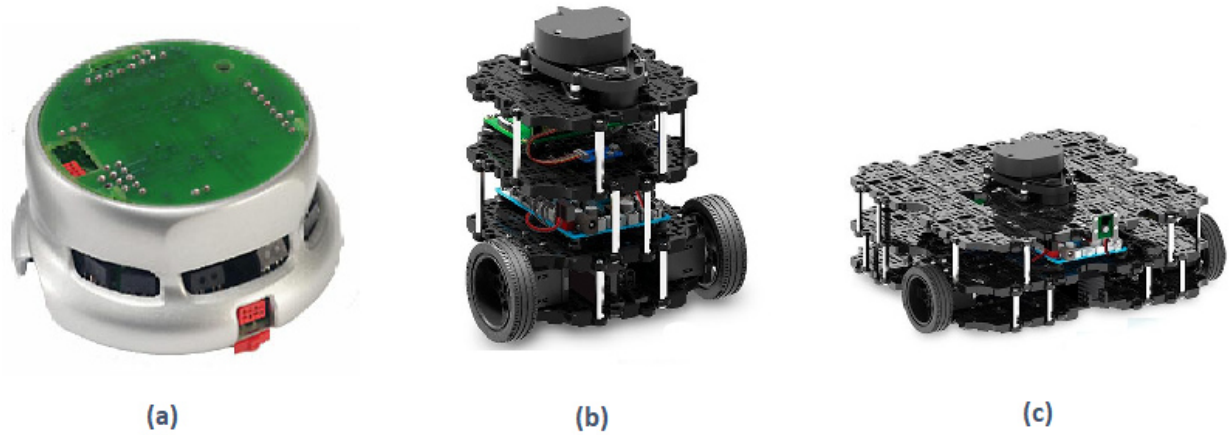


Figure 1. 12: Examples of unicycle type robots [(a) khepera II, (b) TurtleBot3 Burger, (c) TurtleBot3 Waffle]

To model the movement of the mobile robot taking into account these constraints; the pure rolling stress represents the fact that each wheel maintains a point of contact Q with the ground as shown in (**Figure 12**). There is no slippage of the wheel in its longitudinal axis y_l nor of skidding in its orthogonal axis x_l. The speeds of the contact points in the frame of the robot are linked to the wheel speeds:

$$v_R = -r\dot{\varphi}_R$$
$$v_L = r\dot{\varphi}_L \qquad \textbf{1 - 11}$$

The wheels have the same axis of rotation and the instantaneous center of rotation (ICR) of the robot is a point on this axis (**Figure 14**). Let ρ be the radius of curvature of the robot trajectory, b the center distance and ω the speed of rotation of the robot around the ICR. The speeds of the right and left wheels are:

$$v_R = -r\dot{\varphi}_R = (\rho + b)\omega$$
$$v_L = r\dot{\varphi}_L = (\rho - b)\omega \qquad \textbf{1 -12}$$

We define ρ and ω from the speeds of the wheels:

$$\rho = b\frac{\dot{\varphi}_R - \dot{\varphi}_L}{\dot{\varphi}_R + \dot{\varphi}_L}$$
$$\omega = -r\frac{\dot{\varphi}_R + \dot{\varphi}_L}{2b} \qquad \textbf{1 -13}$$

To develop a strategy more than moving, it is interesting to know how the posture of the robot is linked to the control of its wheels.

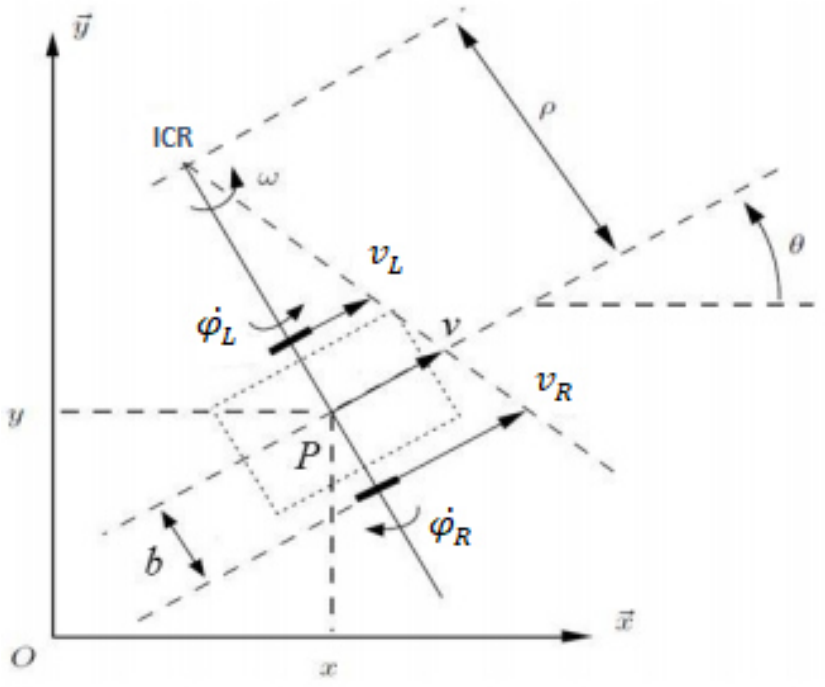


Figure 1. 13: Representation of the ICR

I.3.2.d-The actuation model

Actuation modeling is the study of the movement of mechanical systems without considering the forces that affect the movement. The main goal of actuation modeling is to relate the posture of the robot to the control of its wheels, therefore to represent the speeds of the robot as a function of the speeds of the driving wheels as well as of the geometric parameters of the robot. From these two equations (1-7) and (1-8) we can then deduce the actuation model of our robot. The motion of a differential drive robot is characterized by two non-holonomic constraints. We can then deduce the actuation model of our robot by:

$$v = \dot{x}_l = \frac{v_R + v_L}{2} = r\frac{\dot{\varphi}_L - \dot{\varphi}_R}{2}$$

$$\omega = \dot{\theta} = \frac{v_R - v_L}{2b} = -r\frac{\dot{\varphi}_R - \dot{\varphi}_L}{2b} \qquad \textbf{1–14}$$

Generally, we define the necessary control of the robot by establishing the linear and angular speeds $u = (v \quad \omega)^T$ the control of the wheels is then deduced thanks to the model

described by equation (1-11). The inverse kinematic model makes it possible to change operational speeds v and ω to the speeds of each wheel. We admit the following equations:

$$\dot{\varphi}_R = \omega_R = -\frac{v + b\omega}{r}$$

$$\dot{\varphi}_L = \omega_L = \frac{v + b\omega}{r} \qquad \textbf{1 – 15}$$

I.4-Chapter conclusion

This chapter has firstly allowed us to have a general introduction to mobile robotics and its applications and the main specifications of SRMs. Since the objective of this dissertation is to achieve a hybrid control architecture allowing the maintenance of group formation of mobile robots in a congested environment, we first saw the different control structures existing in the literature; they are classified according to their method of implantation: centralized or distributed their advantages and disadvantages.

This chapter is also an opportunity to present the main control approaches and their metrology which shares common issues in the cooperation of a group of mobile robots in a more specific way and that the leader-follower approach has always become popular. Because they can be explored empirically and the stability of internal formation can be theoretically guaranteed.

Chapter II: Modelization of the Non holonomic mobile robot Leader-Follower formation

II.1-Introduction

The formation control problem can be seen primarily as a natural extension of the traditional trajectory following problem. Only a few researchers have examined the problem of following the trajectory when dealing with the problem of multi-robot formation [47], in this chapter we will extend the traditional problem of tracking control of the trajectory for a single robot to a problem of formation control for several mobile robots.

However, in our leader-follower training system, the trajectories of the followers are generally not predefined; they are decided by their leader in real time. This chapter extends trajectory following control for a single non-holonomic mobile robot to the control for multiple

Non-holonomic mobile robots in which the follower can follow its leader in real time by the proposed kinematic controller.

In this chapter, we will try to propose a mathematical model for a system that is as distributed as possible, using the Leader-Follower approach, some simple frame transformations , trigonometric formulas and the Slipping and Rolling constraints to find the error dynamics to be used later to build a controller for our system.

II.2-Leader-Follower formation control

II.2.1-Mathematical model of a nonholonomic robot

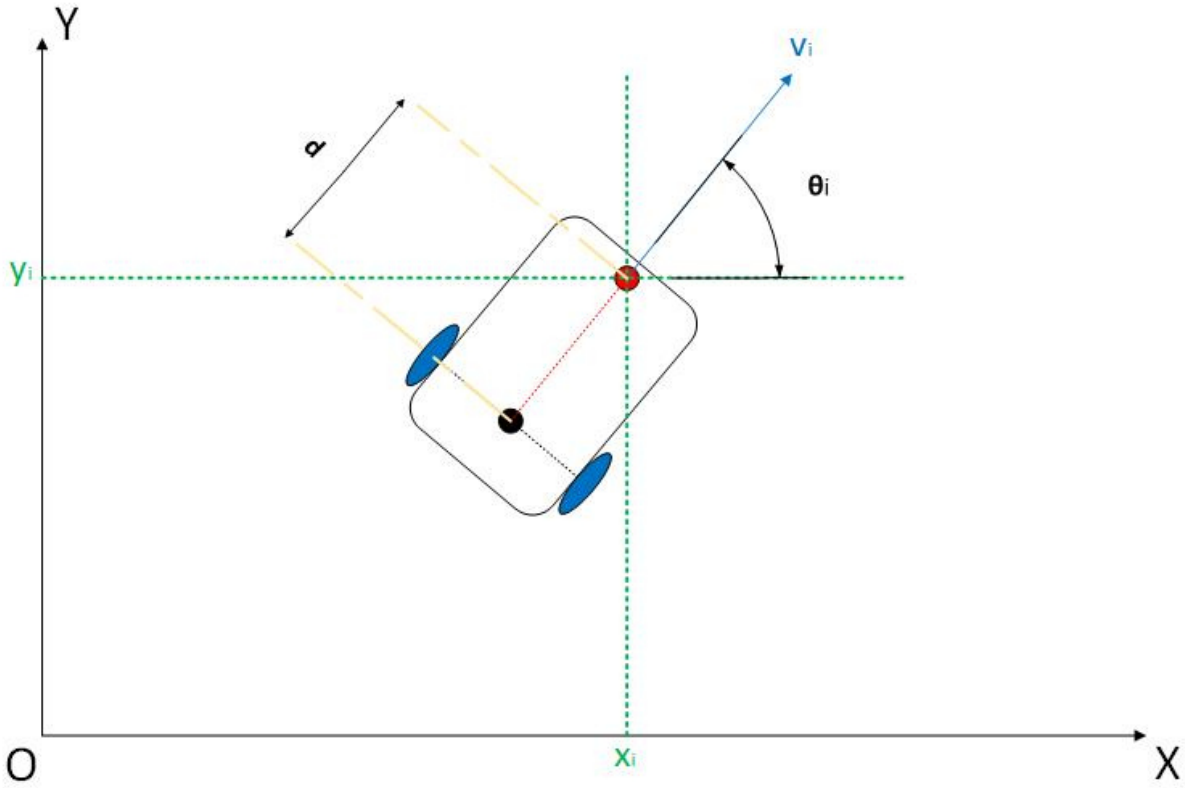


Figure 2. 1 :Representation of a Nonholonomic mobile robot R_i

The two-wheel-drive mobile robot illustrated in (**Figure 15**) is a typical example of non-holonomic mechanical systems. Under the RWS and non-slipping hypothesis, the kinematic constraints of the non-holonomic mobile robot R_i is given as:

$$-\dot{x}_i \sin\theta_i + \dot{y}_i \cos\theta_i = d\,\dot{\theta}_i \qquad \textbf{2 – 1}$$

For the description of the motion of each robot on a 2D plane two coordinate frames have to be used: a global inertial frame (O, X, Y) and a local frame fixed to the i^{th} follower robot. The configuration of the i^{th} robot (**equation 2-1**) represents the position and orientation in the

local coordinate system in the global frame, where the orientation θ_i is taken counterclockwise from the global X-axis (using the right hand rule).

$$p_i = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \qquad \textbf{2 – 2}$$

The motion of the mobile robot is controlled by its linear and angular velocities $v(t)$ and $\omega(t)$ respectively. The relation between $[\mathrm{v(t)} \quad \omega(\mathrm{t})]^T$ and the velocities in Cartesian frame is given by (**equation 2-2**).

$$\dot{p}_i = \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{pmatrix} = \begin{pmatrix} \cos(\theta_i) & 0 \\ \sin(\theta_i) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathrm{v(t)} \\ \omega(\mathrm{t}) \end{pmatrix} \qquad \textbf{2 – 3}$$

II.2.2- Leader-follower formation control and error dynamics

According to the formation control diagram shown in (**Figure 16**), the follower robot R_f follows its leader R_l with the desired separation l_{lf}^d , the desired bearing ψ_{lf}^d and the desired orientation θ_f^d (in this case $\theta_f^d = \theta_l$).

We denote the posture and the control law of the leader robot R_l :

- $p_l = [x_l \quad y_l \quad \theta_l]^T$
- $u_l = [v_l \quad \omega_l]^T$

And the posture of the follower robot R_f as:

- $p_f = [x_f \quad y_f \quad \theta_f]^T$
- $u_f = [v_f \quad \omega_f]^T$

The desired position of the follower in the leader's coordinate system is (x_d, y_d), and the actual position in the global coordinate system $p_f^d = (x_c, y_c)$ can be obtained using the following equation (**equation 2-4**):

$$p_f^d = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x_l \\ y_l \end{bmatrix} + \begin{bmatrix} \cos\theta_l & -\sin\theta_l \\ \sin\theta_l & \cos\theta_l \end{bmatrix} \cdot \begin{bmatrix} x_d \\ y_d \end{bmatrix} \quad \textbf{2-4}$$

By knowing the constant values of (x_d, y_d) we can easily calculate the desired separation l_{lf}^d and bearing ψ_{lf}^d using the following equations:

$$l_{lf}^d = \sqrt{{x_d}^2 + {y_d}^2} \quad \textbf{2-5}$$

$$\psi_{lf}^d = \tan^{-1}\left(\frac{y_d}{x_d}\right) + \pi - \theta_l \quad \textbf{2-6}$$

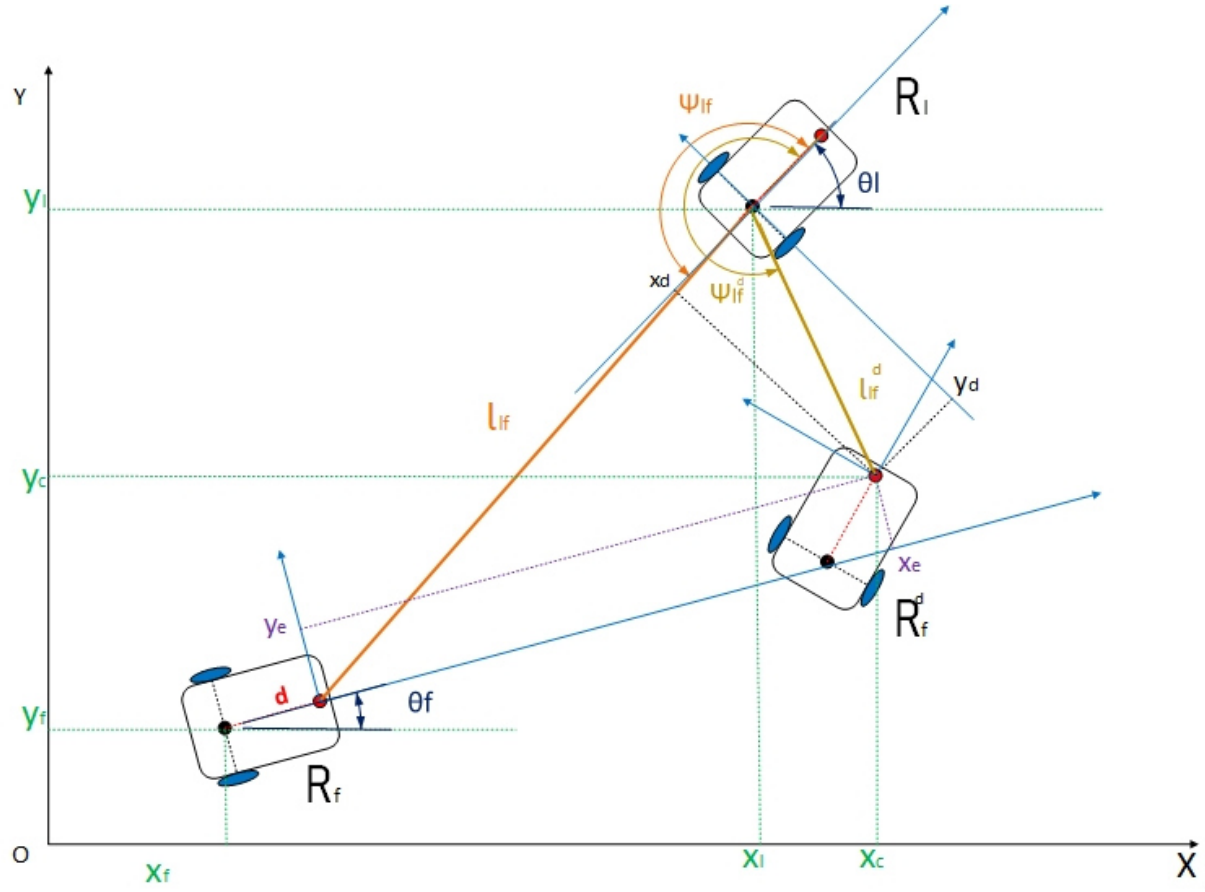


Figure 2. 2: Representation of the Leader-Follower non-holonomic mobile robot formation

Thanks to the geometric relationship between the robots [26], it is easy to obtain the desired position p_f^d of the follower robot:

$$p_f^d = \begin{bmatrix} x_l - d\cos\theta_l + \mathrm{l}_{lf}^d \cos(\psi_{lf}^d + \theta_l) \\ y_l - d\sin\theta_l + \mathrm{l}_{lf}^d \sin(\psi_{lf}^d + \theta_l) \\ \theta_l \end{bmatrix} \quad \textbf{2-7}$$

The real-time posture of the follower robot p_f is defined by:

$$p_f = \begin{bmatrix} x_l - d\cos\theta_l + l_{lf}\cos(\psi_{lf} + \theta_l) \\ y_l - d\sin\theta_l + l_{lf}\sin(\psi_{lf} + \theta_l) \\ \theta_l \end{bmatrix} \quad \textbf{2-8}$$

Taking the time derivative of (**equation 2-4**) and using (**equation 2-3**) and some simple trigonometric formulas which are obtained as follows:

$$\dot{x}_c = \dot{x}_l - x_d\,\dot{\theta}_l \sin\theta_l - y_d\,\dot{\theta}_l\cos\theta_l$$

$$\dot{y}_c = \dot{y}_l + y_d\,\dot{\theta}_l \cos\theta_l - y_d\,\dot{\theta}_l\sin\theta_l$$

$$\dot{x}_c = v_l\cos\theta_l - x_d\omega_l\sin\theta_l - y_d\omega_l\cos\theta_l$$

$$\dot{y}_c = v_l\sin\theta_l + y_d\,\omega_l\cos\theta_l - y_d\omega_l\sin\theta_l$$

$$\dot{x}_c = (v_l - y_d\omega_l)\cos\theta_l - x_d\omega_l\sin\theta_l$$

$$\dot{y}_c = (v_l - y_d\omega_l)\sin\theta_l + x_d\omega_l\cos\theta_l$$

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \end{bmatrix} = \begin{bmatrix} \cos\theta_l & -\sin\theta_l \\ \sin\theta_l & \cos\theta_l \end{bmatrix} \cdot \begin{bmatrix} v_l - y_d\omega_l \\ x_d\omega_l \end{bmatrix} \quad \textbf{2-9}$$

If the distance between the point in the front of the robot (the red dot in **Figure 15**) and the point in the center of wheels axle (the black dot in **Figure 15**) is d in the global coordinate system, the relation can be presented as follows:

$$\begin{cases} x_{offset} = x_f + d\ \cos\theta_f \\ y_{offset} = y_f + d\ \sin\theta_f \end{cases} \quad \textbf{2-10}$$

We calculate the time derivative of (**equation 2-10**):

$$\begin{cases} \dot{x}_{offset} = \dot{x_f} - d\ \dot{\theta}_f \sin\theta_f \\ \dot{y}_{offset} = \dot{y_f} + d\ \dot{\theta}_f \cos\theta_f \end{cases} \quad \textbf{2-11}$$

We use equation (**equation 2-3**):

$$\begin{cases} \dot{x}_{offset} = v_f\ \cos\theta_f - d\ \dot{\theta}_f \sin\theta_f \\ \dot{y}_{offset} = v_f\ \sin\theta_f + d\ \dot{\theta}_f \cos\theta_f \end{cases} \quad \textbf{2-11}$$

Tracking error for leader follower formation in the global coordinate system is obtained as follows:

$$\begin{bmatrix} x_{feG} \\ y_{feG} \end{bmatrix} = \begin{bmatrix} x_c - x_{offset} \\ y_c - y_{offset} \end{bmatrix} \quad \textbf{2-12}$$

Now we transform the tracking error from the global frame to the follower's frame:

$$\begin{bmatrix} x_{fe} \\ y_{fe} \end{bmatrix} = \begin{bmatrix} \cos\theta_l & \sin\theta_l \\ -\sin\theta_l & \cos\theta_l \end{bmatrix} \cdot \begin{bmatrix} x_c - x_{offset} \\ y_c - y_{offset} \end{bmatrix} \quad \textbf{2-13}$$

In the leader-follower approach, the angular and linear velocity of the leader is given, we will only need to control the angular and linear velocity of the follower to maintain the relative separation and relative bearing between them in order to satisfy the desired formation.

We calculate the time derivative of (**equation 2.13**):

$$\begin{cases} x_{fe} = x_c \cos\theta_f - x_{offset} \cos\theta_f + y_c \sin\theta_f - y_{offset} \sin\theta_f \\ y_{fe} = -x_c \sin\theta_f + x_{offset} \sin\theta_f + y_c \cos\theta_f - y_{offset} \cos\theta_f \end{cases}$$

$$\dot{x}_{fe} = \dot{x}_c \cos\theta_f + \dot{y}_c \sin\theta_f - x_c\omega_f \sin\theta_f + y_c\omega_f \cos\theta_f - \dot{x}_{offset} \cos\theta_f - \dot{y}_{offset} \sin\theta_f + x_{offset}\omega_f \sin\theta_f - y_{offset}\omega_f \cos\theta_f$$

$$\dot{y}_{fe} = -\dot{x}_c \sin\theta_f + \dot{y}_c \cos\theta_f - x_c\omega_f \cos\theta_f - y_c\omega_f \sin\theta_f + \dot{x}_{offset} \sin\theta_f - \dot{y}_{offset} \cos\theta_f + x_{offset}\omega_f \cos\theta_f + y_{offset}\omega_f \sin\theta_f$$

We denote $\alpha = \theta_l - \theta_f$, that means $\dot{\alpha} = \omega_l - \omega_f$:

$$\dot{x}_{fe} = (v_l - y_d\omega_l)\cos\alpha - x_d\omega_l \sin\alpha - v_f + \omega_f(-x_c \sin\theta_f + x_{offset} \sin\theta_f + y_c \cos\theta_f - y_{offset} \cos\theta_f)$$

$$\dot{y}_{fe} = (v_l - y_d\omega_l)\sin\alpha - x_d\omega_l \cos\alpha - \omega_f(d + x_c \cos\theta_f - x_{offset} \cos\theta_f + y_c \sin\theta_f - y_{offset} \sin\theta_f)$$

We can obtain the error dynamics of the mobile robot as follows:

$$\begin{bmatrix} \dot{x}_{fe} \\ \dot{y}_{fe} \\ \dot{\theta}_{fe} \end{bmatrix} = \begin{bmatrix} (v_l - y_d\omega_l)\cos\alpha - x_d\omega_l \sin\alpha - v_f + \omega_f\, y_{fe} \\ (v_l - y_d\omega_l)\sin\alpha - x_d\omega_l \cos\alpha - \omega_f(d + x_{fe}) \\ \omega_l - \omega_f \end{bmatrix} \qquad \textbf{2-14}$$

II.3-Chapter conclusion

In this chapter, we have proposed a mathematical model for our MRS making it as distributed using the Leader-Follower approach.

First, we have described the pose of a non-holonomic Mobil robot in the global frame (**equation 2 – 2**) then we used the derivative of this pose to get the displacement of the robot (**equation 2–3**). After that, we managed to find the Tracking error (**equation 2–13**) using frame transformations, trigonometric formulas and the non-holonomic constraints .Finally, we used the derivative of the Tracking error to extract the error dynamics (**equation 2–14**).

In the next chapter, we will try to propose two different controllers for the nonholonomic mobile robot formation.

Chapter III: Controller Conception

III.1-Introduction

The subject of nonlinear control deals with the analysis and the design of nonlinear control systems .this means, control systems containing at least one nonlinear component.

In the analysis, a nonlinear closed-loop system (**Figure 1.9**) is assumed to have been designed, and we wish to determine the characteristics of the system's behavior. In the design, we are given a nonlinear System to be controlled and some specifications of closed-loop system behavior, and our task is to construct a controller so that the closed loop system meets the desired characteristics. In practice, of course, the issues of design and analysis are intertwined, because the design of a nonlinear control system usually involves an iterative process of analysis and design.

In this chapter, we will use the mathematical model calculated in the previous chapter to create two controllers each one of theme uses a different method then the other. For the first one we will use the feedback linearization method which is a common approach used in controlling nonlinear systems. The approach involves coming up with a transformation of the nonlinear system into an equivalent linear system through a change of variables and a suitable control input .and for the second one we will us a Lyapunov's direct method based controller that guarentees the asymptotical stability of the system .

III.2-Feedback Linearization Control

The modeling of real systems is more or less precise, it is done by neglecting certain nonlinearities and disturbances, often even, by reducing the number of state variables of the system. And it is from this approximate representation of physical reality that we want to build as simply as possible a robust and efficient control. When the controlled part of the process is weakly disturbed, conventional control algorithms. One of the most well-known non-linear control methods is Exact Feedback Linearization control in either the input-state direction or the input-output direction, can be sufficient if the demands on the accuracy and performance of the system are not too strict.

Feedback Linearization is an approach to nonlinear control design that has received a great deal of research interest in recent years. The central idea of this approach is to algebraically transform a dynamic of a nonlinear system into a (fully or partially) linear dynamic, so that linear control techniques can be applied. This differs entirely from conventional linearization in that feedback linearization is achieved by state transformations and exact feedback, rather than

linear approximations of dynamics. The idea of simplifying the form of the dynamics of a system by choosing a different state representation is not entirely unknown. In mechanics, for example, it is well known that the form and the complexity of a system model depends considerably on the choice of reference frames or coordinate systems. Feedback linearization techniques can be seen as a means of transforming original system models to an equivalent models of a simpler form. So they can also be used in the development of robust or adaptive nonlinear controllers [20].

We take a mobile robot described by the equation of state (**equation 3-1**)

$$x = f(x,u) \quad \backslash x \in \mathbb{R}^n \ , \ u \in \mathbb{R}^m \qquad \textbf{3 – 1}$$

The idea is to cancel non-linearities and impose a desired linear dynamic, perhaps simply applied to a class of non-linear systems described by the so-called canonical form of controllability, we say that a system is in companion form if its dynamic is represented by:

$$\begin{cases} x^{(k)} = A(X).u + b(x) \\ y = h(X) \end{cases} \qquad \textbf{3 – 2}$$

- ***u*:** The scalar control input.
- ***X*:** The state vector $[x \quad \ddot{x} \quad \dots \quad x^{(k-1)}]^T$.
- ***k*:** The number of times of derivative to make the entry u appear.
- ***A(X) et b(x):*** Nonlinear functions

Keep in mind! : The singularity of Feedback Linearization control

$$\det\big(A(X)\big) \neq 0$$

The form is unique in that although derivatives of x appear in this equation, no derivative of the input u is present. Note that, in the state space representation, (**equation 3-2**) can be written in the following form (**equation 3-3**):

$$\frac{d}{dx}\begin{bmatrix} x_1 \\ \vdots \\ x_{k-1} \\ x_k \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_k \\ A(X).u + b(x) \end{bmatrix} \qquad \textbf{3 – 3}$$

For systems that can be expressed in the canonical form of controllability, we use the control entry:

$$u = A^{-1}(X).[V - b(X)] \qquad \textbf{3 – 4}$$

Then we can cancel the non-linearities and obtain the simple input-output relation using the multiple integrator form **[25]**:

$$x_k = v \qquad \textbf{3 – 5}$$

With the vector v being our new input and of the same dimension as u which itself of the same dimension as y, from this we can consider our looped control system as follows:

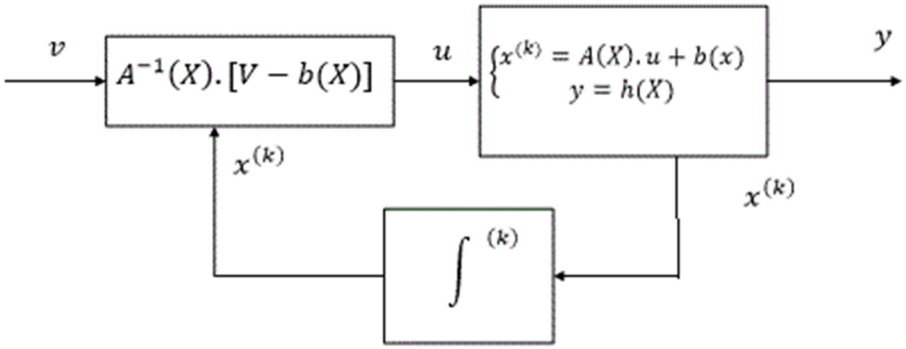


Figure 3. 1: Feedback Linearization control model

We will use a PD type regulator (Proportional, Derivative), the use of such a regulator. It is necessary to have derivative outputs. As we have assumed to have, access to all the state variables x of the system, a formal expression of these derivatives as a function of x is easily obtained using the equations of state.

We first propose to stabilize this system by a proportional and derivative regulator of the type:

$$v = \alpha_0(w - y) + \alpha_1(\dot{w} - \dot{y}) + \cdots + \alpha_2\left(w^{(k-1)} - y^{(k-1)}\right) + w^{(k)} \qquad \textbf{3 – 6}$$

Where w is the desired set point for y . And w can be time dependent like our leader follower system. The fact that this regulator requires the derivatives of y is not a problem in the context of linearization loopback. Indeed, all these derivatives can be written like analytical functions of the state x of the system and of the entry u. Regarding the set point $w\ (t)$, it is chosen by the user and an analytical expression of $w\ (t)$ may be assumed to be known.

Thus, the calculation of the derivatives of w is done in a formal way and no sensitivity of the operator derivation with respect to noise is to be feared.

$$x^{(k)} = v = \alpha_0(w - y) + \alpha_1(\dot{w} - \dot{y}) + \cdots + \alpha_{k-1}\left(w^{(k-1)} - y^{(k-1)}\right) + w^{(k)} \qquad \textbf{3 – 7}$$

We define the error e between the set point w and the output y by $e = w - y$, this equation becomes:

$$\alpha_0 e + \alpha_1(\dot{e}) + \cdots + \alpha_{k-1}\left(e^{(k-1)}\right) + e^{(k)} = 0 \qquad \textbf{3 – 8}$$

This differential equation (**equation 3–8**) is called the error dynamics.

II.2.1-Applying the feedback linearization control to the non-holonomic mobile robot formation

- Feedback linearization control can be used to tackle this model (**equation 2-14**), but first we need write the system as shown in (**equation 3-2**), we get the following equation:

$$\begin{bmatrix}\dot{x}_{fe}\\ \dot{y}_{fe}\end{bmatrix}=\begin{bmatrix}-1 & y_{fe}\\ 0 & -(d+x_{fe})\end{bmatrix}\begin{bmatrix}v_f\\ \omega_f\end{bmatrix}+\begin{bmatrix}(v_l-y_d\omega_l)\cos\alpha-x_d\omega_l\sin\alpha\\ (v_l-y_d\omega_l)\sin\alpha-x_d\omega_l\cos\alpha\\ \omega_l-\omega_f\end{bmatrix} \qquad \textbf{3 – 9}$$

Where:

- $A(X)=\begin{bmatrix}-1 & y_{fe}\\ 0 & -(d+x_{fe})\end{bmatrix}$
- $u=\begin{bmatrix}v_f\\ \omega_f\end{bmatrix}$
- $b(X)=\begin{bmatrix}(v_l-y_d\omega_l)\cos\alpha-x_d\omega_l\sin\alpha\\ (v_l-y_d\omega_l)\sin\alpha-x_d\omega_l\cos\alpha\\ \omega_l-\omega_f\end{bmatrix}$

Consider the following linearizing loopback control inputs:

$$\begin{bmatrix}v_f\\ \omega_f\end{bmatrix}=\begin{bmatrix}-1 & y_{fe}\\ 0 & -(d+x_{fe})\end{bmatrix}^{-1}\left[V-\begin{bmatrix}(v_l-y_d\omega_l)\cos\alpha-x_d\omega_l\sin\alpha\\ (v_l-y_d\omega_l)\sin\alpha-x_d\omega_l\cos\alpha\\ \omega_l-\omega_f\end{bmatrix}\right] \qquad \textbf{3 – 10}$$

Where V is the new entry, our looped system is rewritten in the following form:

$$V=\begin{bmatrix}V_1\\ V_2\end{bmatrix}=\begin{bmatrix}\dot{x}_{fe}\\ \dot{y}_{fe}\end{bmatrix} \qquad \textbf{3 – 11}$$

To prove that the path following control system (**equation 2-14**) under the law of the controller (**equation 3-10**) is asymptotically stable and that the following error converges to zero, we choose the Proportional-Derivative controller:

$$V=\alpha_0\begin{bmatrix}x_{fe}\\ y_{fe}\end{bmatrix}+\dot{Wf} \qquad \textbf{3-12}$$

- Where Wf is the desired pose in the followers frame.

Using pole placement (using identification or Pascal's triangle **[25]**) we find that $\alpha_0=1$, we take $\dot{W}f=0$.

II.2.2- Simulation results on MatLab

We take the distance d from the rear axle to the front of the robot is (0.2m), the linear speed and the angular speed of the leader robot are respectively (1.25, 0.4), and the initial posture of the leader robot is at (0, 0, 0) and the follower robot is at (-3, -5, 0). In order to maintain the leader-follower formation, the follower must maintain the desired separation $l_{lf}^{d} = 1.5\,m$ and the desired orientation with the leader $\Psi_{lf}^{d} = \frac{7\pi}{6}\,rad = 210°$.

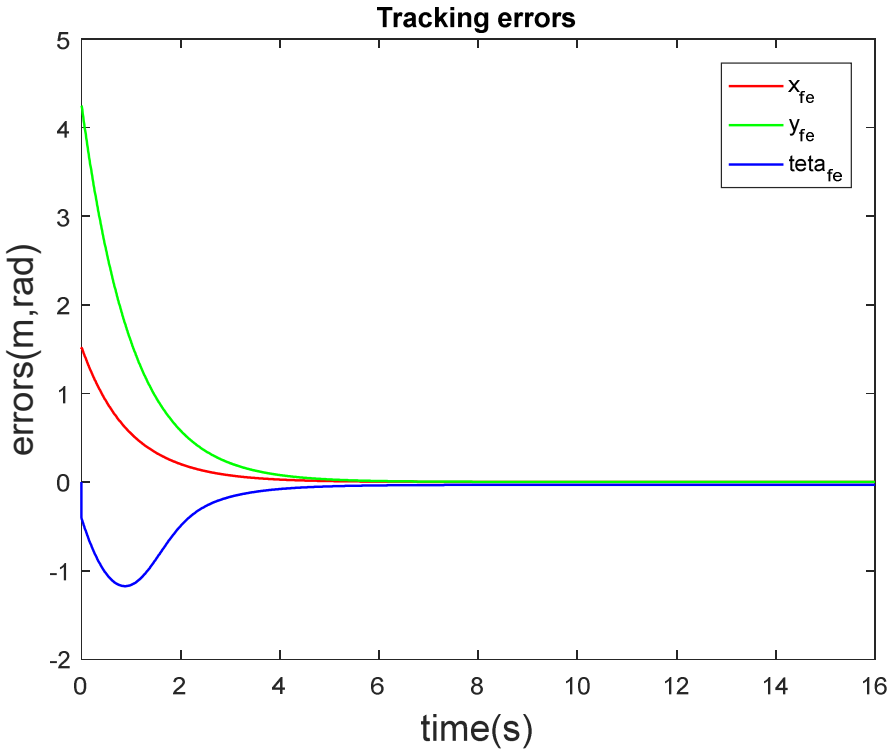


Figure 3. 2: Tracking Errors

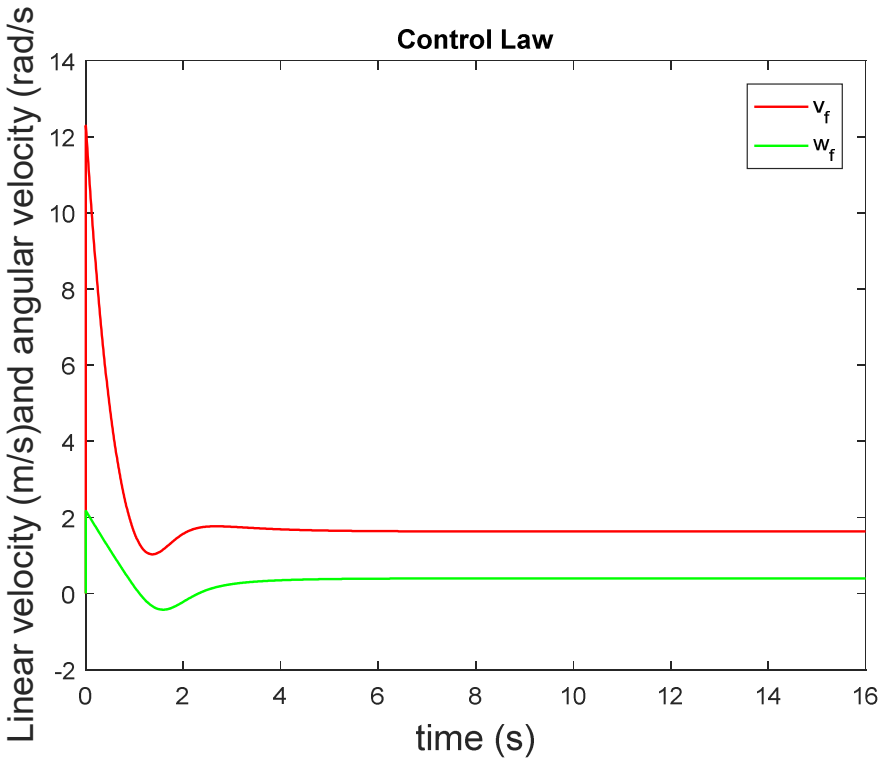


Figure 3. 3: Control Law

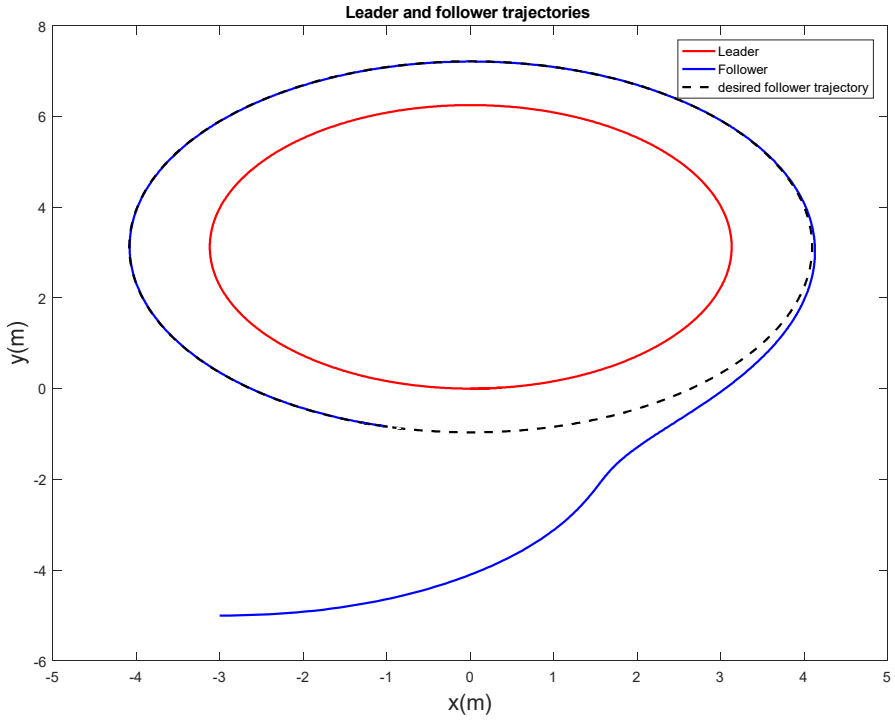


Figure 3. 4: Trajectories

II.2.3- Results discussion:

Figure (3.2) is the follower errors. It shows that the tracking errors of the follower converge towards zero after (t = 6s).

Figure (3.3) is the angular velocity and linear velocity of the follower. From this, the follower starts with a high linear velocity and a high angular velocity. This implies that the initial linear acceleration and angular acceleration are very large, which means that the follower's force and torque are very large.

Figure (3.4) is the trajectory of the robots. It shows that the follower can follow the leader well and maintain the desired separation and bearing with the leader.

III.3- Lyapunov's Direct Method based controller

Given a control system, the primary and most vital question concerning its numerous properties is whether it is stable or not, because an unstable control system is typically useless and potentially dangerous. Qualitatively, a system is delineated as stable if starting the system close to its desired operating point implies that it will stay around the point ever after. The movements of a pendulum starting near its two equilibrium points, (the vertical up "12" and down"6" positions) are frequently used to illustrate the stability of a dynamic system.

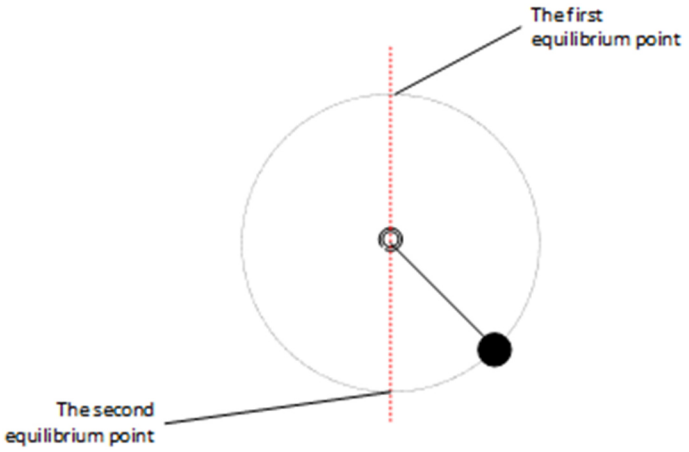


Figure 3. 5: Equilibrium points of a pendulum

For aircraft control systems, a typical stability problem is intuitively related to the following question: will a trajectory perturbation due to a gust cause a significant deviation in the later flight trajectory? Here, the desired operating point of the system is the flight trajectory in the

absence of disturbance **[25]**. Every control system, whether linear or nonlinear, involves a stability problem that should be carefully studied.

The most useful and general approach for studying the stability of nonlinear control systems is the theory introduced in the late 19th century by the Russian mathematician Alexandr Mikhailovich Lyapunov. Lyapunov's work, The General Problem of Motion Stability, includes two methods for stability analysis (the so-called linearization method and direct method) and was first published in 1892. The linearization method draws conclusions about a nonlinear system's local stability around an equilibrium point from the stability properties of its linear approximation. The direct method is not restricted to local motion, and determines the stability properties of a nonlinear system by constructing a scalar "energy-like" function for the system and examining the function's time variation **[20]**.

The objective of this part of the chapter is to present Lyapunov stability theory and illustrate its use in the analysis and the design of controllers for nonlinear systems in general then apply it to control our nonholonomic mobile robot formation.

III.3.1- Equilibrium Points

A nonlinear dynamic system can usually be represented by a set of nonlinear differential equations in the form:

$$\dot{x} = f(x, u) \qquad \textbf{3-13}$$

- $\boldsymbol{f}$: Is a $n \times 1$ nonlinear vector function.
- $\boldsymbol{x}$:Is the $n \times l$ state vector

It is possible for a system trajectory to correspond to only a single point. Such a point is called an equilibrium point. Many stability problems are naturally formulated with respect to equilibrium points.

A state x^* is an equilibrium state (or equilibrium point) of the system if once $x\ (t)$ is equal to x, it remains equal to x for all future time.

Mathematically, this means that the constant vector x^*satisfies

$$f(x^*) = 0 \qquad \textbf{3-14}$$

Equilibrium points can be found by solving the nonlinear algebraic equations (**equation 3-14**).

III.3.2-Lyapunov's direct method

Stability is a binary property of a system, that is, a system cannot be simultaneously stable or not stable. However, a stable system is characterized by a degree or index that shows what quantity close to instability the system is (relative stability). A system is outlined to be bounded-input bounded-output (BIBO) stable if any bounded input leads invariably to a bounded output. A linear time-invariant system is BIBO stable if and only if all the poles of its transfer function or the eigenvalues of the matrix A of its state-space model lie strictly on the left-hand complex semi plane. The matrix A with the above property is named a Hurwitz matrix. The Routh and Hurwitz algebraical criteria specify the conditions that the coefficients of the system's characteristic polynomial should satisfy in order for the system to be stable [19].But the Routh and Hurwitz stability criteria will solely be used for time-invariant linear single-input single-output (SISO) systems.

Lyapunov's stability method may be applied to time-varying systems and to nonlinear systems. Lyapunov has introduced a generalized notion of energy (called Lyapunov function) and studied dynamic systems without external input. Combining Lyapunov's theory with the idea of BIBO stability, we are able to derive stability conditions for input-to-state stability (ISS). Lyapunov has introduced two stability methods. The first method needs the availability of the system's time response (i.e., the solution of the differential equations). The second method, conjointly known as direct Lyapunov method, does not need the knowledge of the system's time response.

Reminder:

Definition 1: Positive definiteness

A scalar function $f(x)$ is said to be positive definite in a particular region which includes the origin of state space if $f(x) > 0$ for all non-zero states x in that region and $x(0) = 0$

Definition 2: Negative definiteness

A scalar function $f(x)$ is said to be negative definite if $-f(x)$ is positive definite.

Definition 3: Positive Semi-definiteness

A scalar function $f(x)$ is said to be positive semi-definite if it is positive at all states in the particular region except at the origin and at a certain other states where it is zero.

Definition 4: Negative Semi-definiteness

A scalar function $f(x)$ is said to be negative semi-definite if – $f(x)$ if positive semi-definite.

Theorem: (Lyapunov's Direct Method) [19] [20]

1. If a scalar function $V\ (x, t)$ satisfies the following conditions:
   - ✓ $V\ (0, t) = 0$.
   - ✓ $V\ (x, t)$ Positive definite.
   - ✓ $\dot{V}\ (x, t)$ Negative definite.

   The system is asymptotically stable.

2. If a scalar function $V\ (x,t)$ satisfies the following conditions:
   - ✓ $V\ (0,t) = 0$.
   - ✓ $V\ (x,t)$ Positive definite.
   - ✓ $\dot{V}\ (x,t)$ Negative semi-definite.

   The system is locally stable.

3. If a scalar function $V\ (x,t)$ satisfies the following conditions:
   - ✓ $V\ (0,t) = 0$.
   - ✓ $V\ (x,t)$ Positive definite for $x \neq 0$.
   - ✓ $\dot{V}\ (x,t)$ Negative definite for $x \neq 0$.
   - ✓ $\lim_{\|x\|\to\infty} V\ (x,t) \to \infty$

The system is globally asymptotically stable.

III.3.3-Conception of the Lyapunov's direct method based controller:

The first step is to choose a viable candidate for the Lyapunov function V that must be positive definite, for this we chose the following function:

$$V = \frac{1}{2}(x_e^2 + y_e^2) \qquad \textbf{3-15}$$

Now we calculate the derivative of the Lyapunov function mentioned in (**equation 3-15**):

$$\dot{V} = x_e(v_l - y_d.\omega_l)\cos(\alpha) + x_d.\omega_l.x_e.\sin(\alpha) - v_f.x_e + \omega_f.y_e.x_e + (v_l - y_d.\omega_l).y_e.\sin(\alpha) + x_d.\omega_l.y_e.\cos(\alpha) - y_e.\omega_f.d - y_e.\omega_f.x_e \qquad \textbf{3-16}$$

$$\dot{V} = x_e(v_l - y_d.\omega_l)\cos(\alpha) + x_d.\omega_l.x_e.\sin(\alpha) - v_f.x_e + (v_l - y_d.\omega_l).y_e.\sin(\alpha) + x_d.\omega_l.y_e.\cos(\alpha) - y_e.\omega_f.d \qquad \textbf{3-17}$$

Finally, we choose v_f and ω_f in a way that makes $\dot{V}$ Negative definite to guarantee the asymptotical stability of the system:

$$\begin{cases} v_f = (v_l - y_d.\omega_l)\cos(\alpha) + x_d.\omega_l.\sin(\alpha) + k_1.x_e \\ w_f = ((v_l - y_d.\omega_l).\sin(\alpha) + x_d.\omega_l.\cos(\alpha) + k_2.y_e)/d \end{cases} \qquad \textbf{3-18}$$

With $k_1 > 0$ and $k_2 > 0$.

III.3.4- Simulation results on MatLab

We take the distance d from the rear axle to the front of the robot is (0.2m), the linear speed and the angular speed of the leader robot are respectively (1.25, 0.4), and the initial posture of the leader robot is at (0, 0, 0) and the follower robot is at (-3, -5, 0). In order to maintain the leader-follower formation, the follower must maintain the desired separation $l_{lf}^{d} = 1.5\ m$ and the desired orientation with the leader $\Psi_{lf}^{d} = \frac{7\pi}{6}\ rad = 210°$.

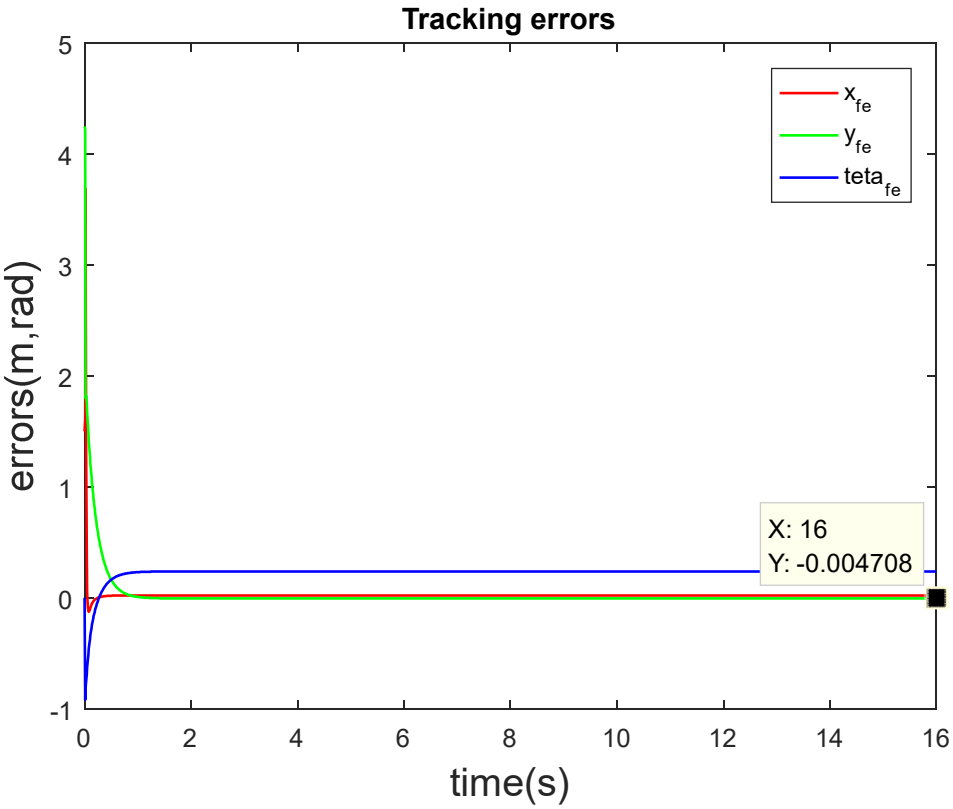


Figure 3. 6: Tracking errors

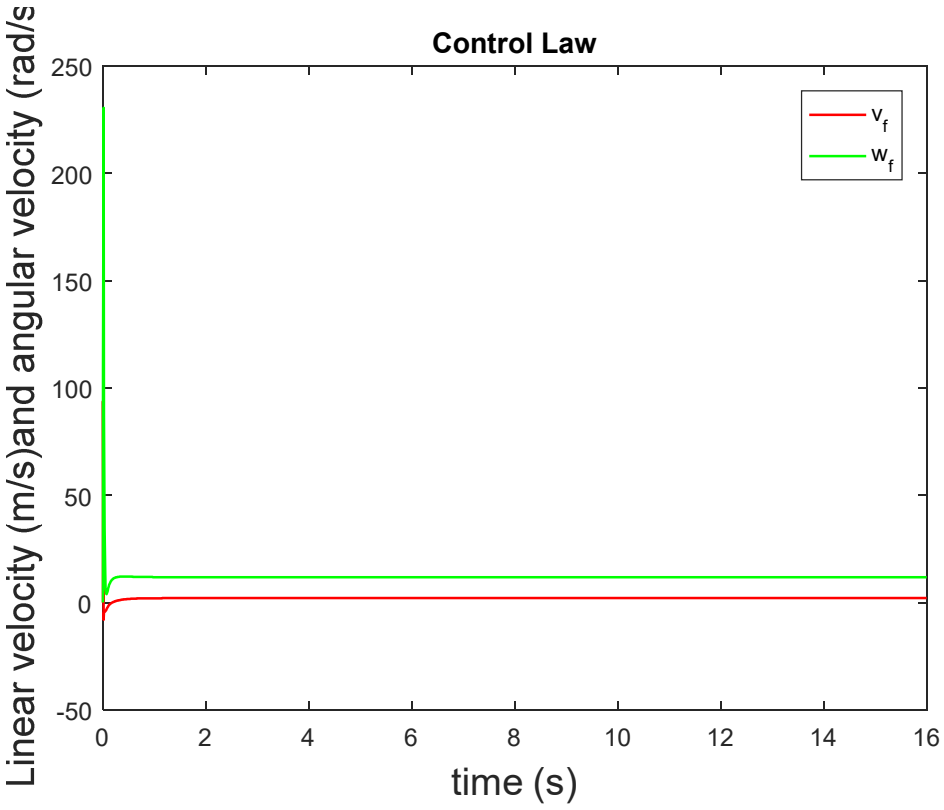


Figure 3. 7: Control Law

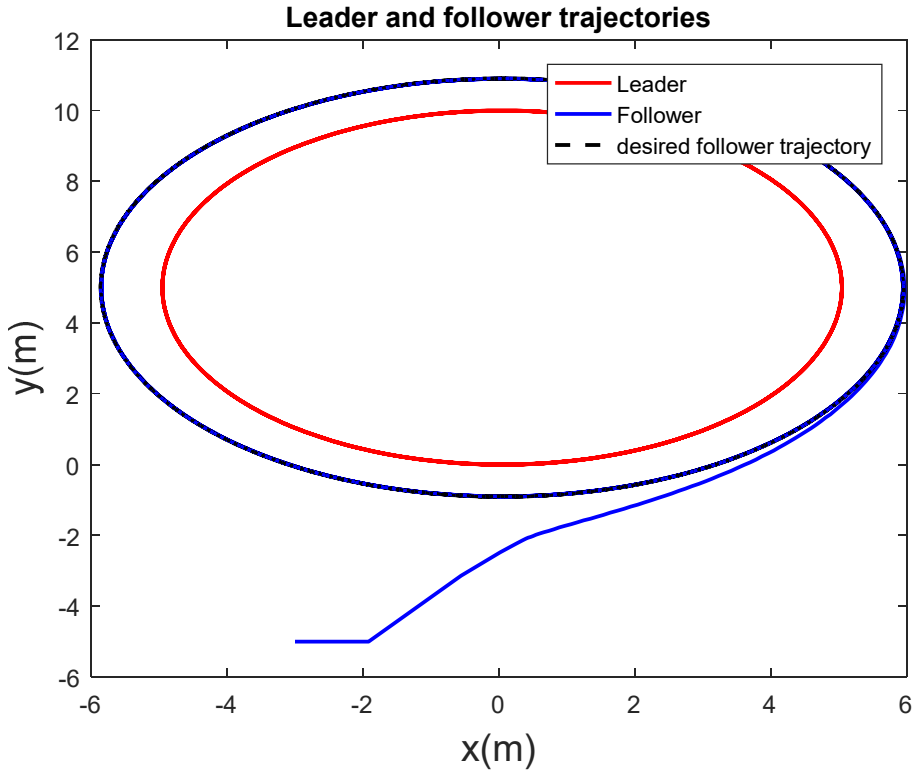


Figure 3. 8: Trajectories

III.2.3- Results discussion:

Figure (3.6) is the follower errors. It shows that the tracking errors of the follower converge towards zero after (t = 0.53s) which is excessively fast comparing to the feedback linearization method but the tracking on theta is a little bit less accurate.

Figure (3.7) is the angular velocity and linear velocity of the follower. From this, the follower starts with a high linear velocity and a high angular velocity. This implies that the initial linear acceleration and angular acceleration are very large, which means that the follower's force and torque are very large then converges to a lower value.

Figure (3.8) is the trajectory of the robots. It shows that the follower can follow the leader well and maintain the desired separation and bearing with the leader.

III.4-Smooth Variable Structure Filter (SVSF) [26] [23] [30]:

Saeid R. Habibi introduced the SVSF (Smooth Variable Structure Filter) in 2007 **[26]**. This filter is based on the sliding mode control and estimation techniques. It is formulated in a predictor-corrector mode, where gain switching is used to ensure that the estimated values converge to the real state values. As shown in **(Figure3.9)**.

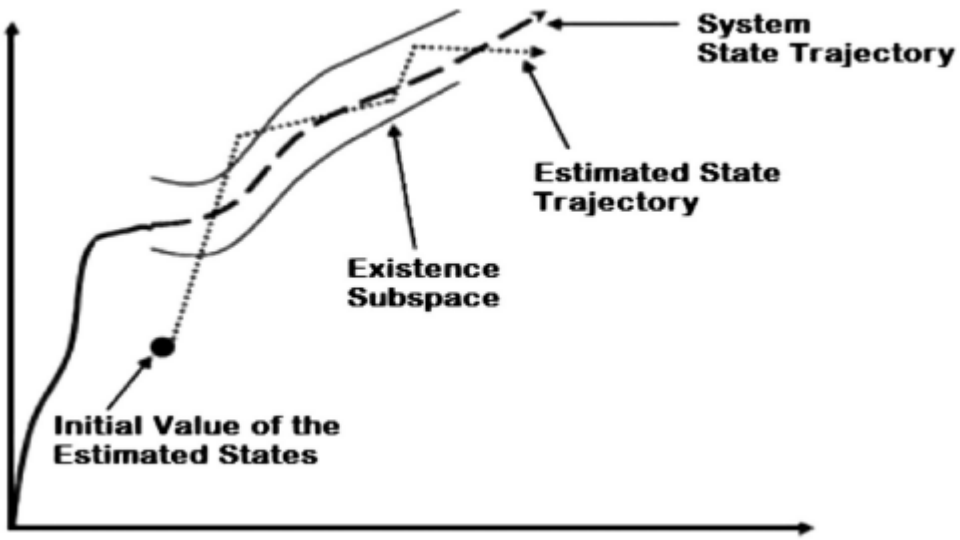


Figure 3. 9: Estimation using the SVSF filter

The SVSF uses an existing subspace and a smoothing boundary layer to maintain the limited estimates in a region of the actual state path. The estimation process is summarized in equations (3.28) - (3.37). SVSF is very different from the Extended Kalman Filter **[31]**, which neither has nor uses covariance matrix. The SVSF uses a switching gain (SVSF filter gain) to converge the estimates within a band around the actual path. This band is called "Existence Subspace β ".

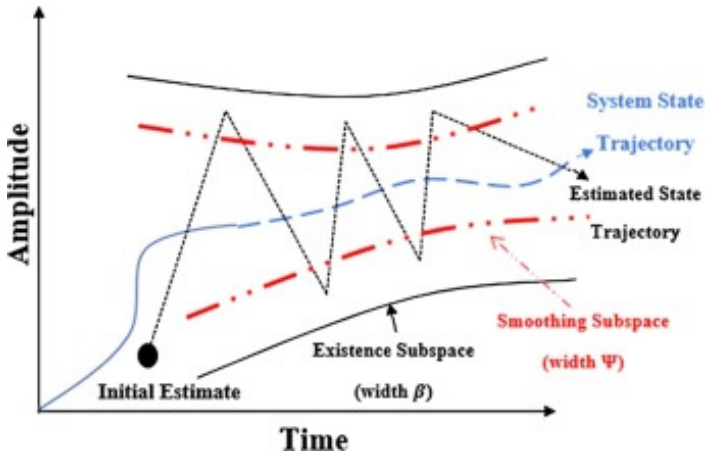


Figure 3. 10: Presence of chattering (β>Ψ) [26]

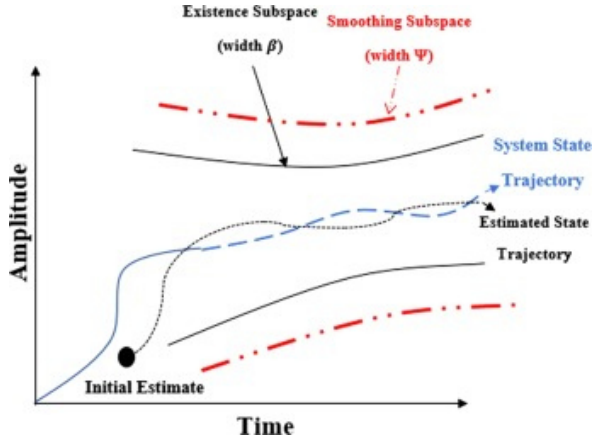


Figure 3. 11: Post estimation smoothed Trajectory (β<Ψ) [26]

Mathematical formulation of the SVSF filter [26]:

In 2003, the Variable Structure Filter (VSF) was introduced as a new predictor-corrector method used for the estimation of states and parameters. A corrective term is then applied to calculate the posterior state estimate, and the estimation process is repeated iteratively. SVSF was later derived from VSF, and uses a simpler and less complex gain calculation. In this form, the SVSF is stable and robust to uncertainties and noise.

- We consider a system, linear or nonlinear, modeled in its following state form:

$$\begin{cases} x_{k+1} = F(x_k, u_k, w_k) \\ z_{k+1} = H.x_k + v_k \end{cases} \quad \textbf{3-19}$$

-The SVSF passes through three steps:

1- Initialization

$$\begin{cases} x_{0\,|\,0} = x_0 \\ E_{0\,|\,0} = E_0 \end{cases} \quad \textbf{3-20}$$

2- Prediction

1-The estimation of the current state (k) is predicted using the estimated model of the system:

$$\hat{x}_{k+1\mid k} = \hat{F}(x_{k\mid k}, u_k) \quad \textbf{3-21}$$

2- Using (**equation 3-21**) we can calculate the current estimation of the measures (k)

$$\hat{z}_{k+1\mid k} = \widehat{H}.\hat{x}_{k+1\mid k} \quad \textbf{3-22}$$

3- Now we calculate the current measurement error

$$E_{k+1\mid k} = z_{k+1} - \hat{z}_{k+1\mid k} \quad \textbf{3-23}$$

3- Updating the Values

4- For the state estimation, the correction gain "K_{SVSF}" is calculated as follows:

$$K_{SVSF} = \widehat{H}^{+}(\gamma.\left|E_{k\mid k}\right| + \left|E_{k+1\mid k}\right|)\circ \text{Sign}(E_{k+1\mid k}) \quad \textbf{3-24}$$

Keep in mind:

- '$\boldsymbol{\gamma}$' Is referred to as the convergence rate (values between 0 and 1).
- ' $+$ ' Indicates the pseudo-inverse of a matrix **[32]**.
- '$\circ$' Represents the element-by-element product between two vectors.

5- The estimation of the state after the instance k+1 is calculated as follows

$$\hat{z}_{k+1\mid k+1} = \widehat{H}.\hat{x}_{k+1\mid k+1} \quad \textbf{3-25}$$

6- Finally we calculate the error after the instance k+1 is calculated as follows

$$E_{k+1\mid k} = z_{k+1} - \hat{z}_{k+1\mid k+1} \quad \textbf{3-26}$$

The discontinuous correction gain K_{SVSF} is formulated to guarantee the convergence and the stability of the estimation process, this means that the estimated state does not diverge and always remains in the vicinity of the real state within an uncertainty band. However, the use of the gain K_{SVSF} with thesignum "$sign$" function (**Figure 3.12**) introduces chattering. In order to remedy this phenomenon, the use of a form of saturation "Sat" (**Figure 3.13**) within the uncertainty band can attenuate this chatter.

This means that (**equation 3.24**) becomes:

$$K_{SVSF} = \widehat{H}^{+}(\gamma.\left|E_{k\mid k}\right| + \left|E_{k+1\mid k}\right|)\circ \text{Sat}(E_{k+1\mid k}, \Psi) \quad \textbf{3-27}$$

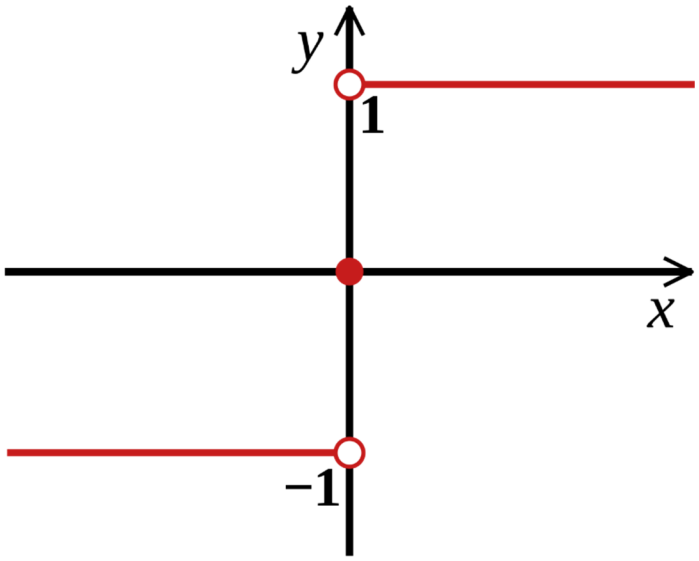


Figure 3. 12: Signum function

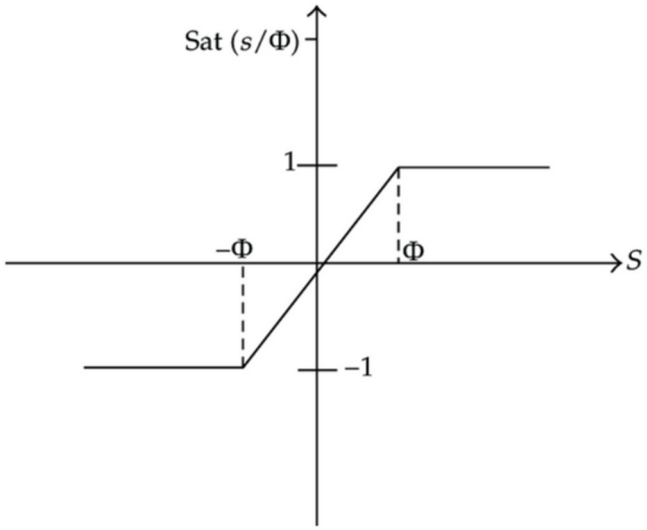


Figure 3. 13: Saturation function

III.5- Chapter conclusion

In this chapter, we have used the mathematical model from the previous chapter to propose two controllers .the first one was made using the feedback linearization method and the second one was made using Lyapunov's direct method that guarantees the stability of the system. Finally, we introduced the concept of leader position estimation using the SVSF FILTER to reduce the chattering phenomena.

In the next chapter, we will try implement our two controllers in the ROS Gazebo simulator.

Chapter IV: Simulation in “Robot Operation System”

IV.1-Introduction

In the first chapter, we approached the different types of formation structures and the control approaches of multi-robot systems. Then, in the second chapter, we have addressed the modeling and control of unicycles robots. And in the third one, we studied the Lyapunov direct method and the SVSF corrector-predictor filter. Based on all of these informations, we chose a specific method for our project: "control of a multi-robot system by Lyapunov’s direct method ".

The leader robot is equipped with a laser to assist him in his autonomous navigation, and a wireless access point to share his odometry with the followers. These latter followers are equipped with a receiver (in order to receive the information sent by the leader to enable them to follow his path).

In this fourth and last chapter, we will define the software environment necessary to achieve the desired scenario (in this case ROS). Then write the implementation of the simulation environment in 3D as well as the simulation results.

IV.2-Robot Operating System

Robot Operating System (ROS) is an open source **robotics middleware**. This means that it is designed to manage the complexity and heterogeneity of the hardware and applications, promote the integration of new technologies, simplify software design, hide the complexity of low-level communication and the sensor heterogeneity of the sensors, improve software quality, reuse robotic software infrastructure across multiple research efforts, and to reduce production costs. Although ROS is not an operating system as the name suggests, but it is a collection of software frameworks for robot software development. it provides services designed for a heterogeneous computer cluster such as low-level device control,

implementation of commonly used functionality, message passing between processes, package management and **hardware abstraction**, which is the main feature.

Hardware abstraction are sets of routines in software that provide programs with access to hardware resources through programming interfaces. The programming interface allows the access to all devices through identical interfaces; this allows developers to write device-independent, high performance applications by providing standard operating system (OS) calls to hardware **[29]**.

Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor data, control, state, planning, actuator, and other messages.

Software in the ROS Ecosystem can be separated into three groups:

- language-and platform-independent tools used for building and distributing ROS-based software;
- ROS client library implementations such as roscpp, rospy, and roslisp.
- Packages containing application-related code, which uses one or more ROS client libraries.

Both the language-independent tools and the main client libraries (C++, Python, and Lisp) are released under the terms of the BSD license **[27]**, and as such are open source software and free for both commercial and research use. The majority of other packages are licensed under a variety of open source licenses. These other packages implement commonly used functionality and applications such as hardware drivers, robot models, datatypes, planning, perception, simultaneous localization and mapping, simulation tools, and other algorithms.

IV.2.1-ROS Building blocks and tools

(Figure 3.1) shows the main components of ROS and their interconnections. The red outlined blocks correspond to the different parts between the simulation and the real-life environment.

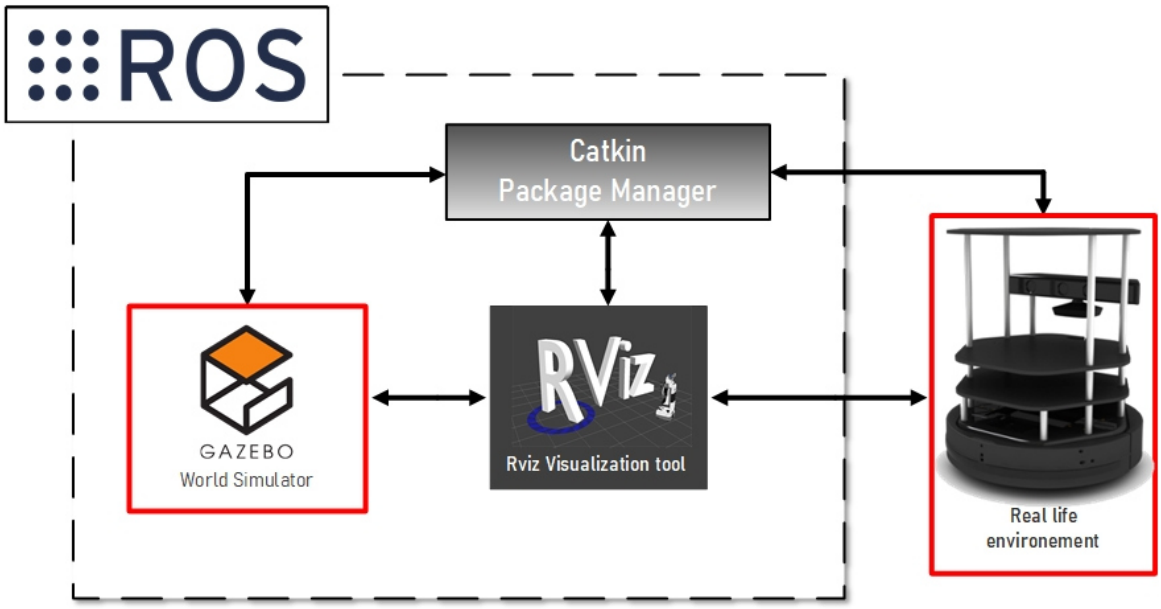


Figure 4. 1: ROS building blocks

IV.2.1.a-catkin

Catkin is the official build system of ROS and the successor to the original ROS build system, “rosbuild”. Catkin was designed to allow a better distribution of packages, better cross-compiling support, and better portability. catkin's workflow is very similar to CMake's but adds support for automatic 'find package' infrastructure and building multiple, dependent projects at the same time.

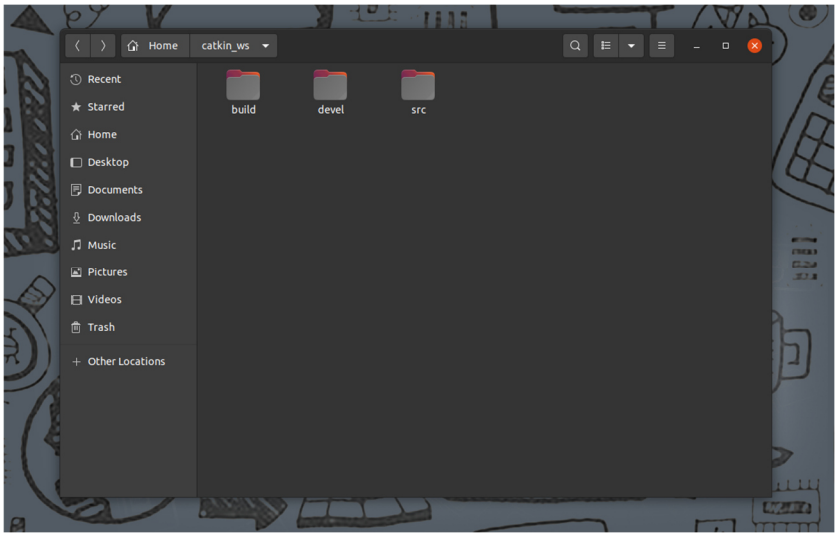


Figure 4. 2: catkin workspace

IV.2.1.b-Rviz

Rviz, abbreviation for ROS visualization, is a powerful 3D visualization tool for ROS. It allows the user to view the simulated robot model, log sensor information from the robot's sensors, and replay the logged sensor information. By visualizing what the robot is seeing, thinking, and doing, the user can debug a robot application from sensor inputs to planned (or unplanned) actions.

IV.2.1.c-Gazebo

Gazebo is a 3D simulator that allows the user to create a 3D scenario with robots, obstacles and many other objects. Gazebo also uses a physical engine for illumination, gravity, inertia, etc. You can evaluate and test your robot in difficult or dangerous scenarios without any harm to your robot. Most of the time it is faster to run a simulator instead of starting the whole scenario on your real robot.

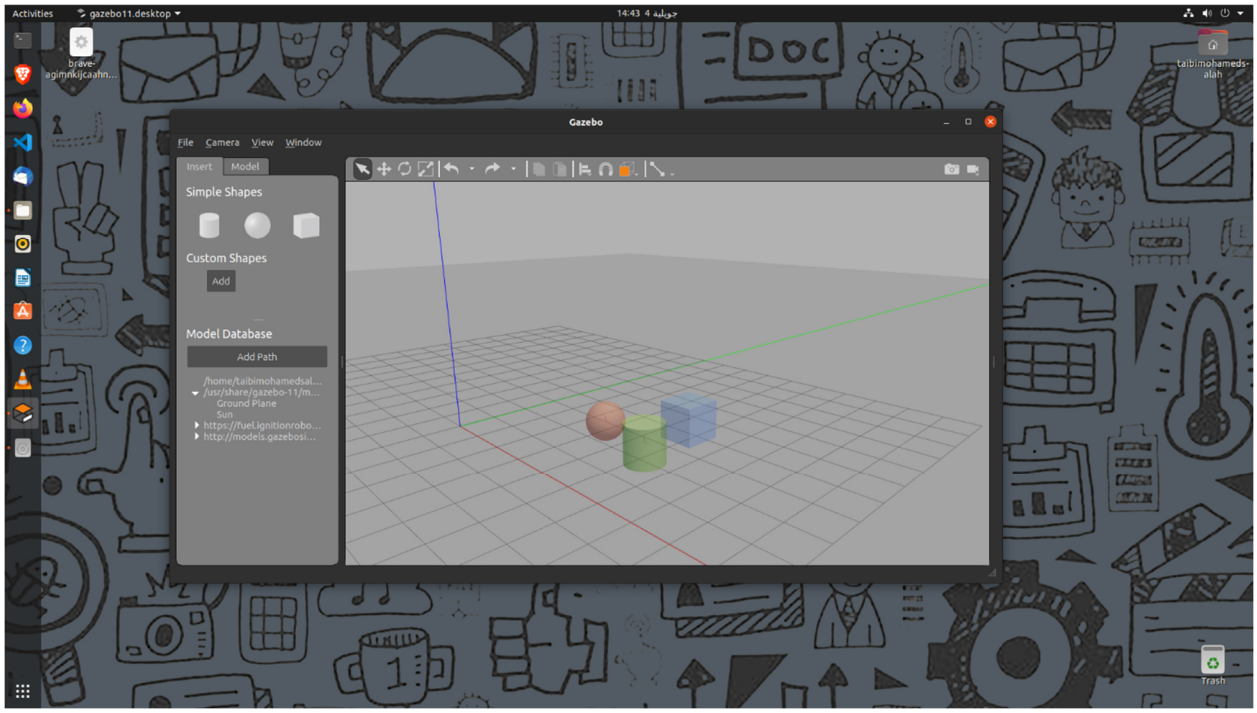


Figure 4. 3: Gazebo Simulator

IV.2.1.c-rqt

rqt is a Qt-based framework for GUI development for ROS.it contains two main function that are widely used:

- Rqt_graph

  It is a graphical interface allowing the analysis of the application graph and data transfer.

- Rqt_multiplot

  rqt_multiplot provides a GUI plugin for visualizing numeric values in multiple 2D plots.

IV.2.2-ROS basics

IV.2.2.a-Package

Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module. The goal of these packages it to provide this useful functionality in an easy-to-consume manner so that software can be easily reused. In general, ROS packages follow a "Goldilocks" principle: enough functionality to be useful, but not too much that the package is heavyweight and difficult to use from other software.

Or simply, a package is the folder that contains our project. Packages are located in our catkin workspace in the **src** file (**figure 4.2**).

IV.2.2.b-Node

A node is a process that performs computation. Nodes are combined together into a graph and communicate with one another using streaming **topics**, RPC **services**, and the **Parameter Server**. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. The use of nodes in ROS provides several benefits to the overall system. There is additional fault tolerance as crashes are isolated to individual nodes. Code complexity is reduced in comparison to monolithic systems.

IV.2.2.c-topic

Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic. There can be multiple **publishers** and **subscribers** to a topic.

ROS currently supports TCP/IP-based and UDP-based message transport. The TCP/IP-based transport is known as TCPROS and streams message data over persistent TCP/IP connections. TCPROS is the default transport used in ROS and is the only transport that client libraries are required to support. The UDP-based transport, which is known as UDPROS and is currently only supported in roscpp.

IV.2.2.d- Service

The publish / subscribe model is a very flexible communication paradigm, but its many-to-many one-way transport is not appropriate for RPC request / reply interactions, which are often required in a distributed system. Request / reply is done via a Service, which is defined by a pair of messages: one for the request and one for the reply. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply. Client libraries usually present this interaction to the programmer as if it were a remote procedure call. Services are defined using srv files, which are compiled into source code by a ROS client library.

A client can make a persistent connection to a service, which enables higher performance at the cost of less robustness to service provider changes.

IV.2.2.e- Message

A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, Boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays.

In this project, we will use the following message types:

- geometry_msgs/Pose2D

  This message gives the current two-dimensional pose of the robot as a vector of three components$[x \quad y \quad theta]$

- **x:** the position on the x-axis
- **y:** the position on the y-axis
- **theta:** the orientation of the robot

- geometry_msgs/Twist

This expresses velocity in free space broken into its linear $[x \quad y \quad z]$ and angular$[x \quad y \quad z]$ parts:

- **linear:**
  - ***x:*** linear velocity on the x-axis
  - ***y:*** linear velocity on the y-axis
  - ***z:*** linear velocity on the z-axis
- **angular:**
  - ***x:*** angular velocity around the x-axis
  - ***y:*** angular velocity around the y-axis
  - ***z:*** angular velocity around the z-axis

- std_msgs/Float32MultiArray

The MultiArray declares a generic multi-dimensional array of a particular data type.

IV.2.2.f- Parameter server

A parameter server is a shared, multi-variate dictionary that is accessible via network APIs (Application Programming Interface). Nodes use this server to store and retrieve parameters at runtime. As it is not designed for high-performance, it is best used for static, non-binary data such as configuration parameters. It is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify if necessary.

IV.2.2.f- Launch file

Launch files are very common in ROS to both users and developers. They provide a convenient way to start up multiple nodes and a master, as well as other initialization requirements such as setting parameters.

IV.3-TurtleBot3

TurtleBot is a ROS standard platform robot. Turtle is derived from the Turtle robot, which was driven by the educational computer programming language Logo in 1967. In addition, the TurtleSim node, which first appears in the basic tutorial of ROS, is a program that mimics the command system of the Logo turtle program. It is also used to create the Turtle icon as a symbol of ROS. The nine dots used in the ROS logo derived from the back shell of the turtle. TurtleBot, which originated from the Turtle of Logo, is designed to easily teach people who are new to ROS through TurtleBot as well as to teach computer-programming languages using Logo. Since then TurtleBot has become the standard platform of ROS, which is the most popular platform among developers and students.

There are three versions of the TurtleBot model. Tully (Platform Manager at Open Robotics) and Melonee (CEO of Fetch Robotics) from Willow Garage developed TurtleBot1 on top of the iRobot's Roomba-based research robot, Create, for ROS deployment. It was developed in 2010 and has been on sale since 2011. In 2012, TurtleBot2 was developed by Yujin Robot based on the research robot, iClebo Kobuki. In 2017, TurtleBot3 was developed with features to supplement the lacking functions of its predecessors, and the demands of users. The TurtleBot3 adopts ROBOTIS smart actuator DYNAMIXEL for driving.

TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. The goal of TurtleBot3 is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality, while at the same time offering expandability. The TurtleBot3 can be customized into various ways depending on how you reconstruct the mechanical parts and use optional parts such as the computer and sensor. In addition, TurtleBot3 is evolved with cost-effective and small-sized SBC that is suitable for robust embedded system and a 360-degree distance sensor.

The technical characteristics of the TurtleBot3 robot are as follows:

- Maximum translational velocity: 0.22 m/s
- Maximum rotational velocity: 2.84 rad/s (162.72 deg/s)
- Maximum payload: 15kg
- Size (L x W x H): 138mm x 178mm x 192mm
- Weight (+ SBC + Battery + Sensors): 1.8kg
- Expected operating time: 2h 30m
- Expected charging time: 2h 30m
- IMU(Inertial Measurement Unit): Gyroscope (3Axis), Accelerometer (3Axis), Magnetometer (3Axis)
- Battery: Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
- MCU(microcontroller unit): 32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)

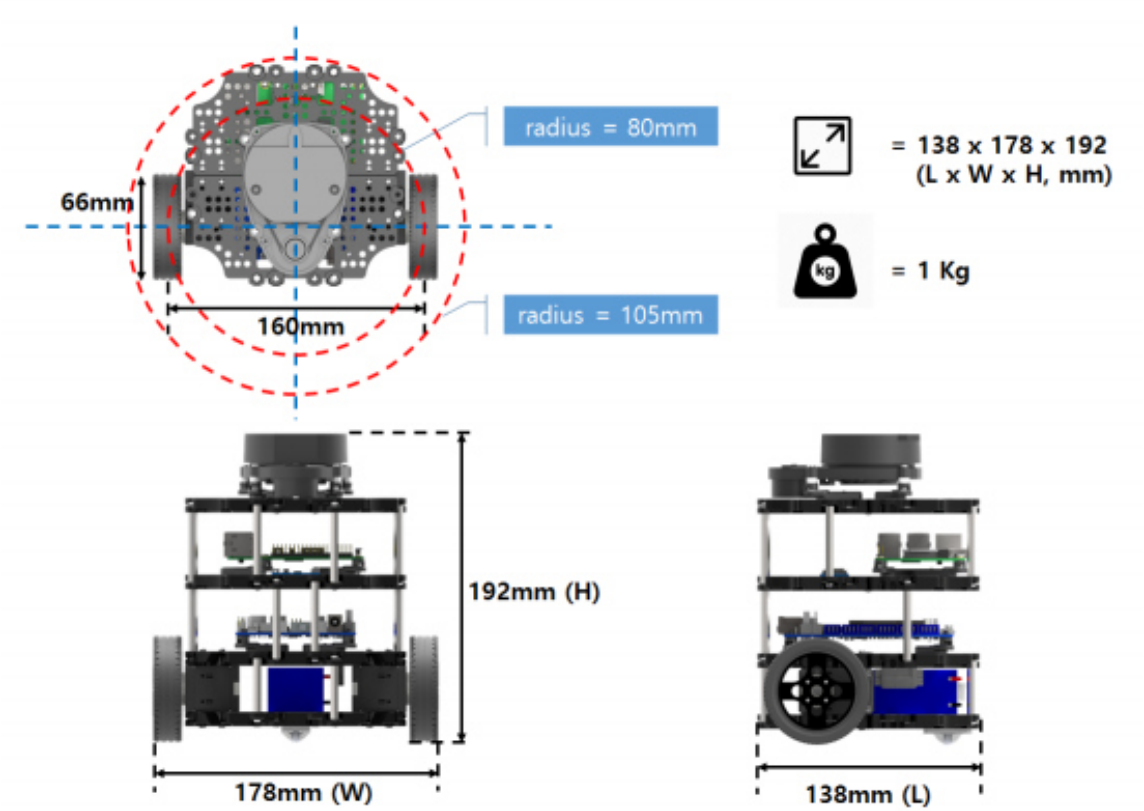


Figure 4. 4: Side and top view of the TurtleBot3 Burger [29]

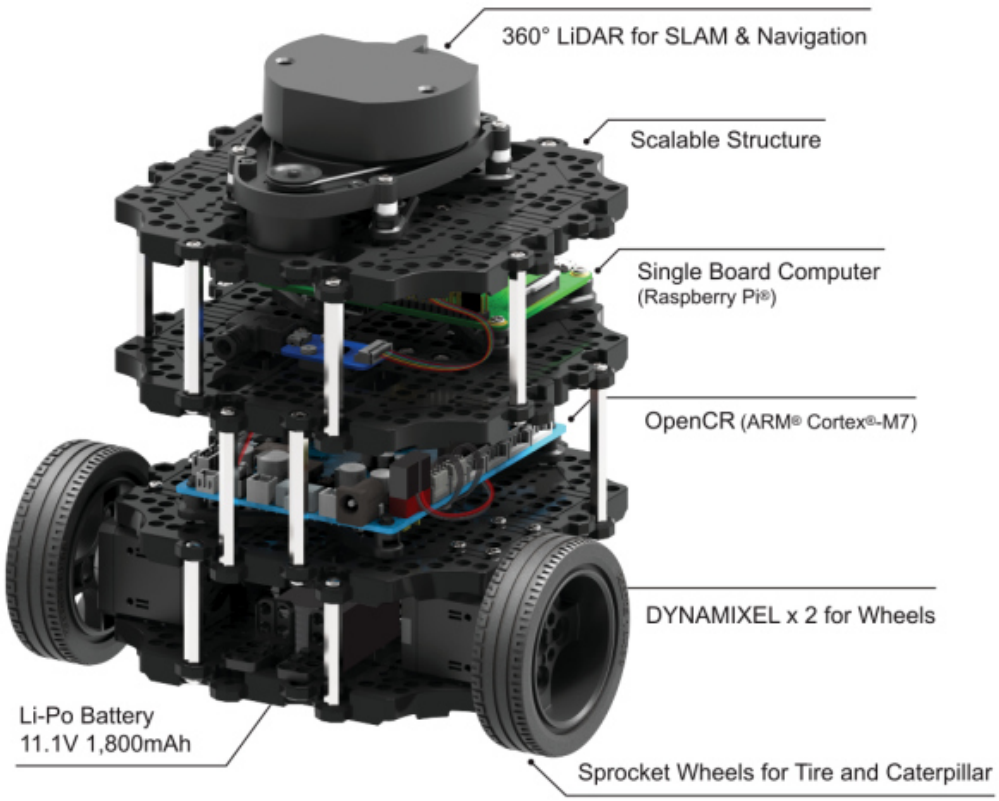


Figure 4. 5: TurtleBot3 components [29]

IV.4-Simulation

The simulation was carried out with a pre-build PC with the following specifications:

- Centrale Processing Unit: Intel(R) core i7-4790 CPU
- Graphic Processing Unit: intel hd 4000 2gb
- RAM: (2x8gb) 16gb Crucial DDR3 1200 MHz
- Storage: Kingstone 500 GB HDD.

We used four (04) TurtleBot3 burger robots. The overall diagram of the environment and control of the Leader-Follower multi-robot system built with ROS (Gazebo) is shown in **(Figure4.6)**.We considered three (02) scenarios to test the performance of the formation control algorithm and the correction-estimator filter (SVSF).

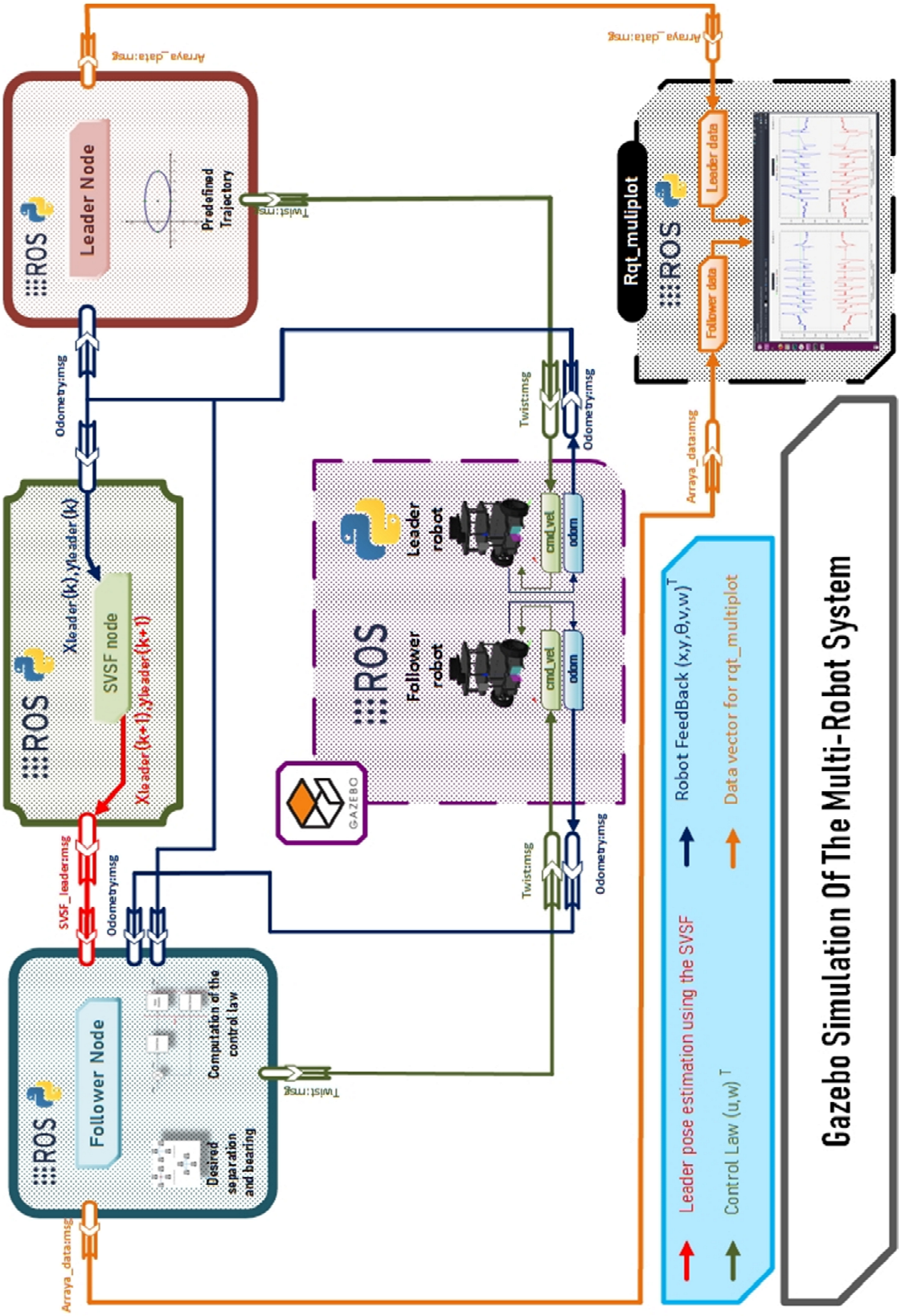


Figure 4. 6: Gazebo Simulation of the Multi-Robot System

IV.4.1-Package components

IV.4.1.a-The nodes

We used the following nodes:

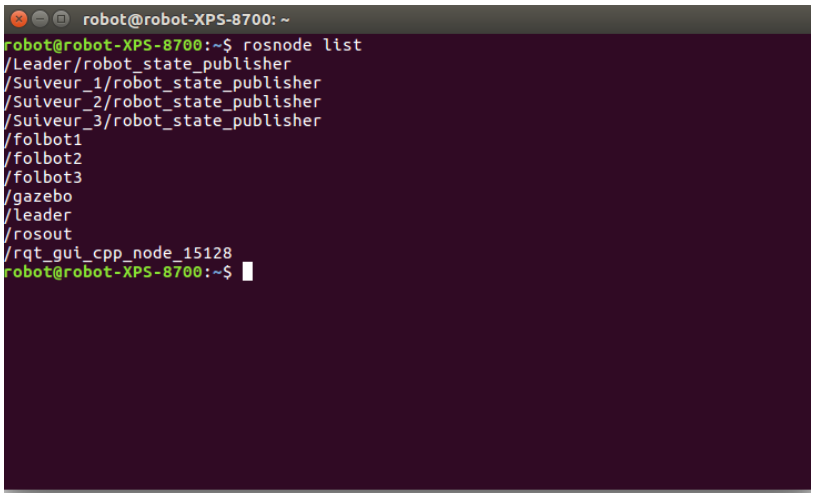


Figure 4. 7: The node List

IV.4.1.b-The topics

We used the following topics:

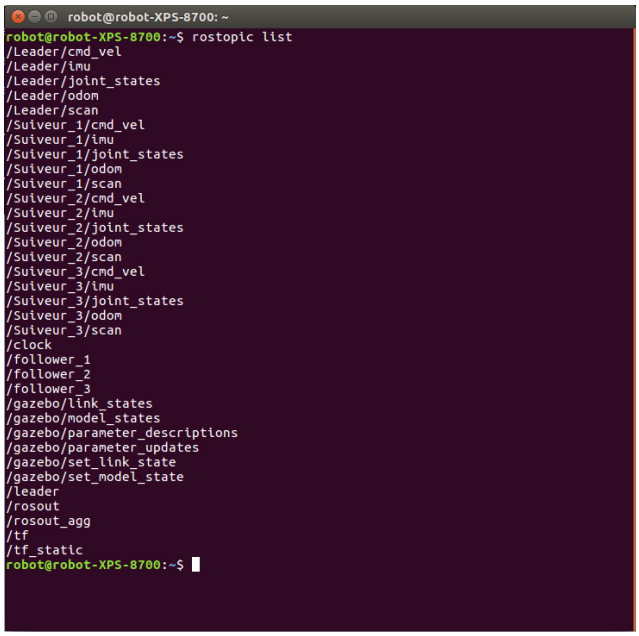


Figure 4. 8: The topic List

IV.4.1.c-Launch files

We have used two types of launch file. The first one is to launch the gazebo environment (**env_O.launch** and **env_S.launch**) and the second one is to launch the formation (**form_O.launch** and **form_S.launch**).

IV.4.1.c-rqt_multiplot

We will use four plots to describe the behavior of our MRS:

1. Trajectories of the robots
2. Tracking errors
3. Linear velocity
4. Angular velocity

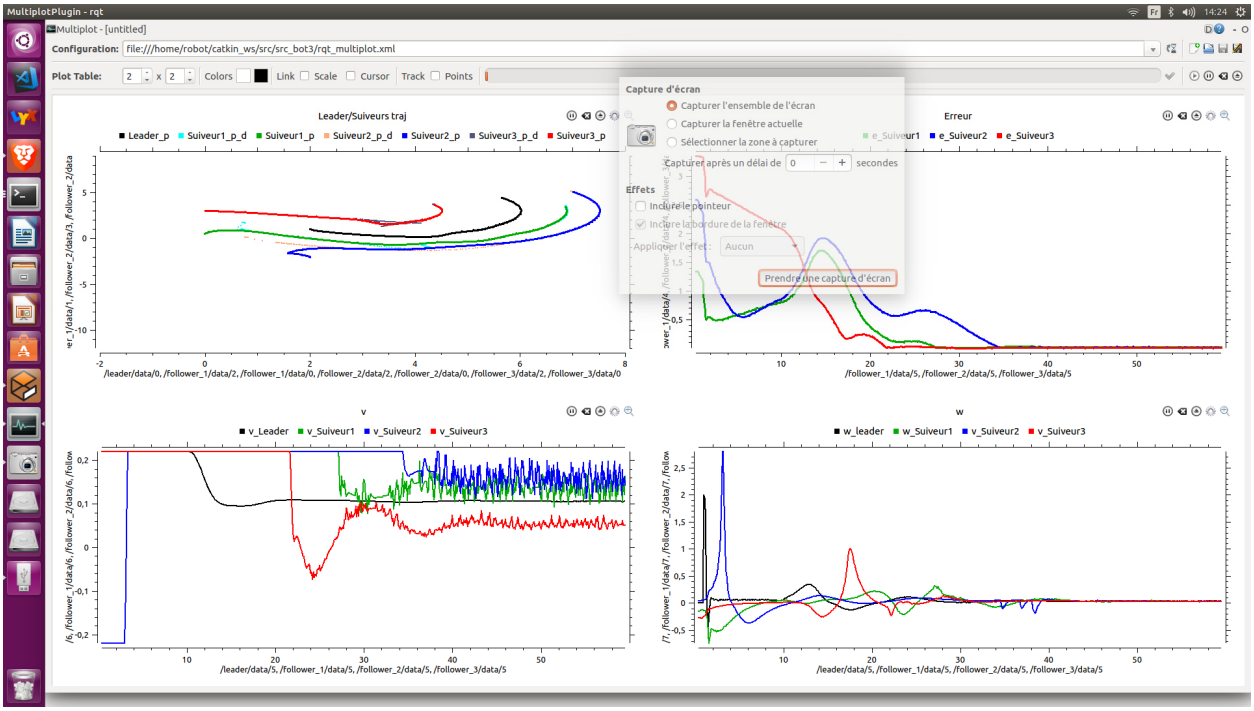


Figure 4. 9: The rqt_multiplot window

IV.4.1.c-rqt_graph

This dynamic graph represents the state of our MRS.

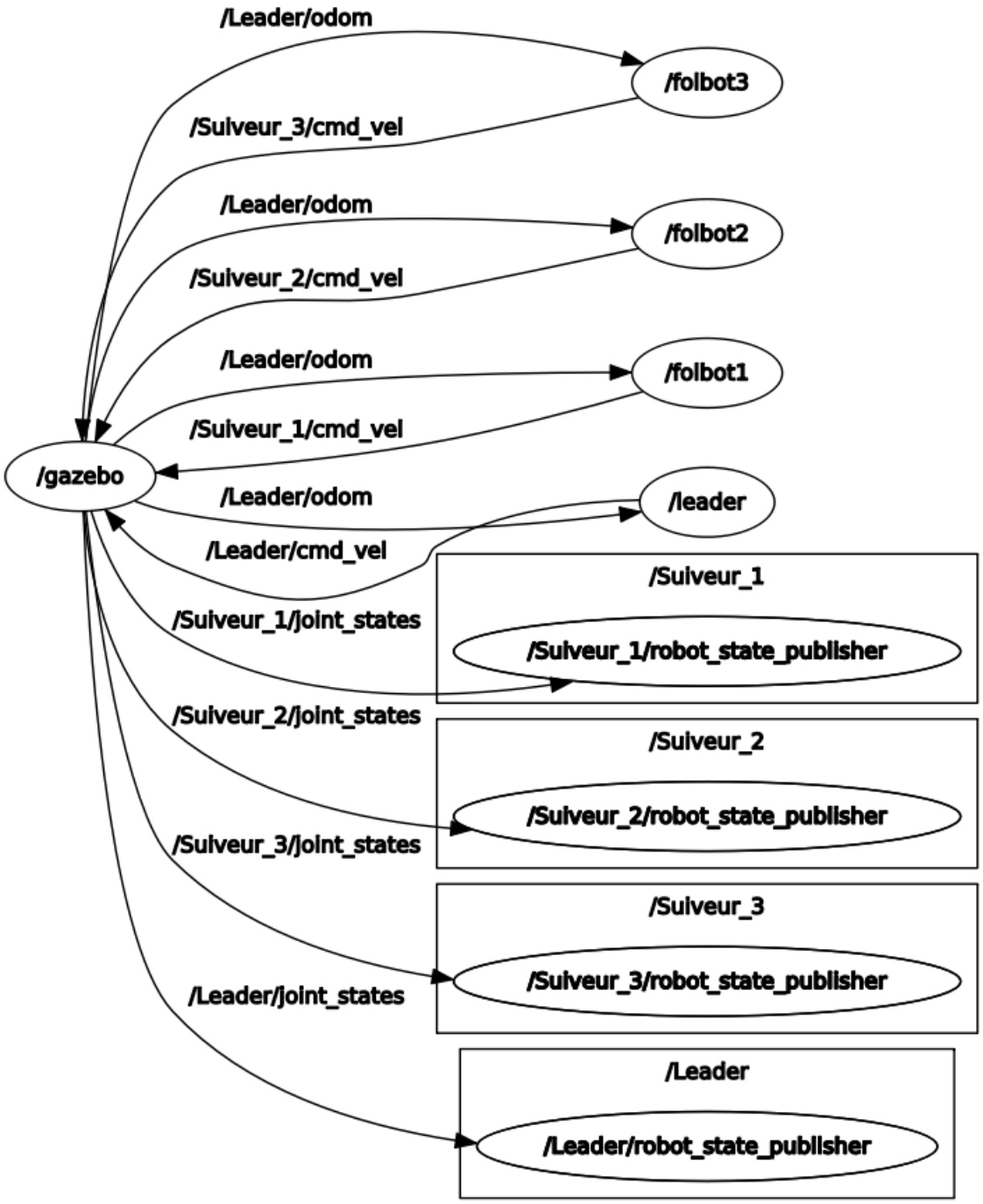


Figure 4. 10: rqt_graph of our ROS package

IV.4.2-Lyapunov's Direct Method controller simulation

IV.4.2.a: rounded shape trajectory:

The four robots we used are TurtleBot3 Burger robots. They are placed in a free space environment. The leader will try to follow a predefined rounded trajectory while the other three try to follow him with the desired separation and bearing.

The initial positions of the robot are as follows:

	Initial pose	Desired separation	Desired bearing
Leader	X=2 m; y=1m ;θ=-1 rad		
Fol_1	X=0 m; y=0.5m ;θ=1.2 rad	1.5	$\frac{8\pi}{7}$
Fol_2	X=2 m; y=-2m ;θ=-1 rad	1.5	$\frac{3\pi}{2}$
Fol_3	X=0 m; y=3m ;θ=0 rad	1.5	$\frac{\pi}{2}$

Table 4. 1: Initial parameters for the first scenario

We chose the following controller parameters:

- $k_1 = 5$
- $k_2 = 1.5$

Keep in mind! The simulation's accuracy totally depends on the computational power and the graphic processing power of the PC. It is recommended to restart the PC after two simulations to free up some space in the RAM that is why we didn't try the effects of the SVSF Filter on removing the chattering phenomena.

The results of the simulation are shown in the four figures below:

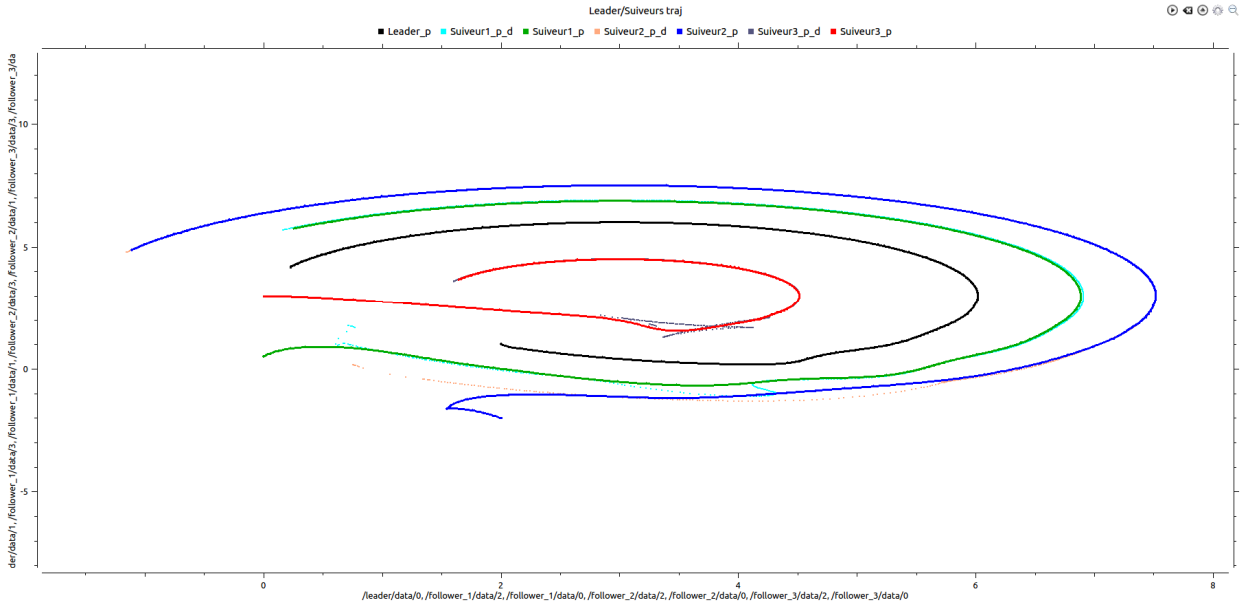


Figure 4. 11: The trajectories

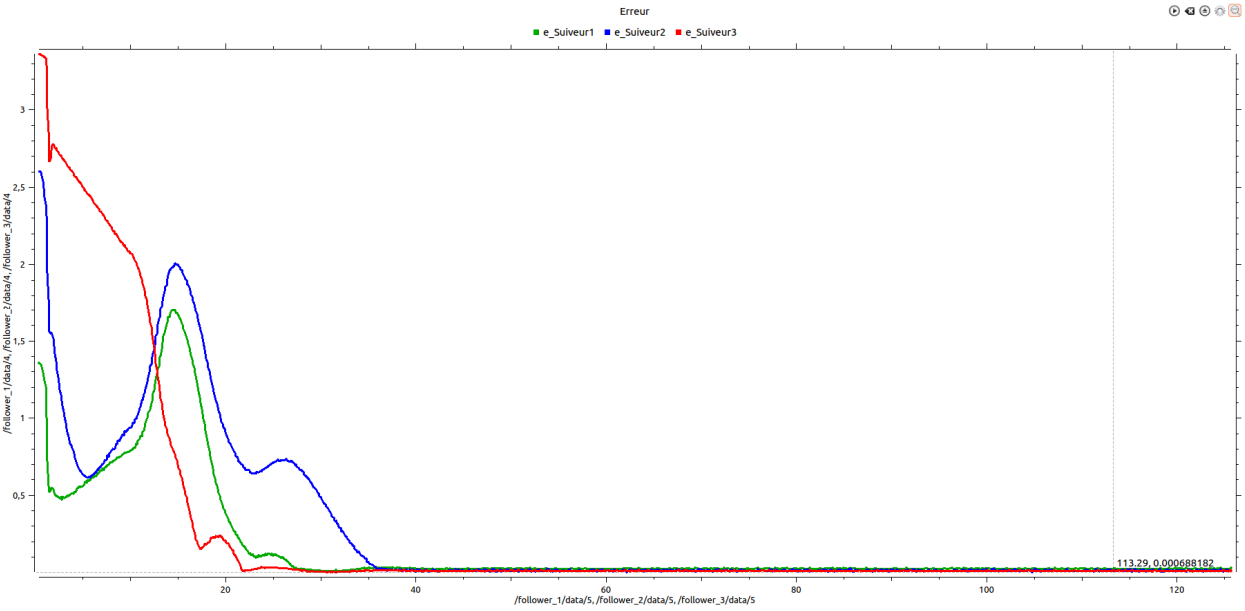


Figure 4. 12: The tracking errors

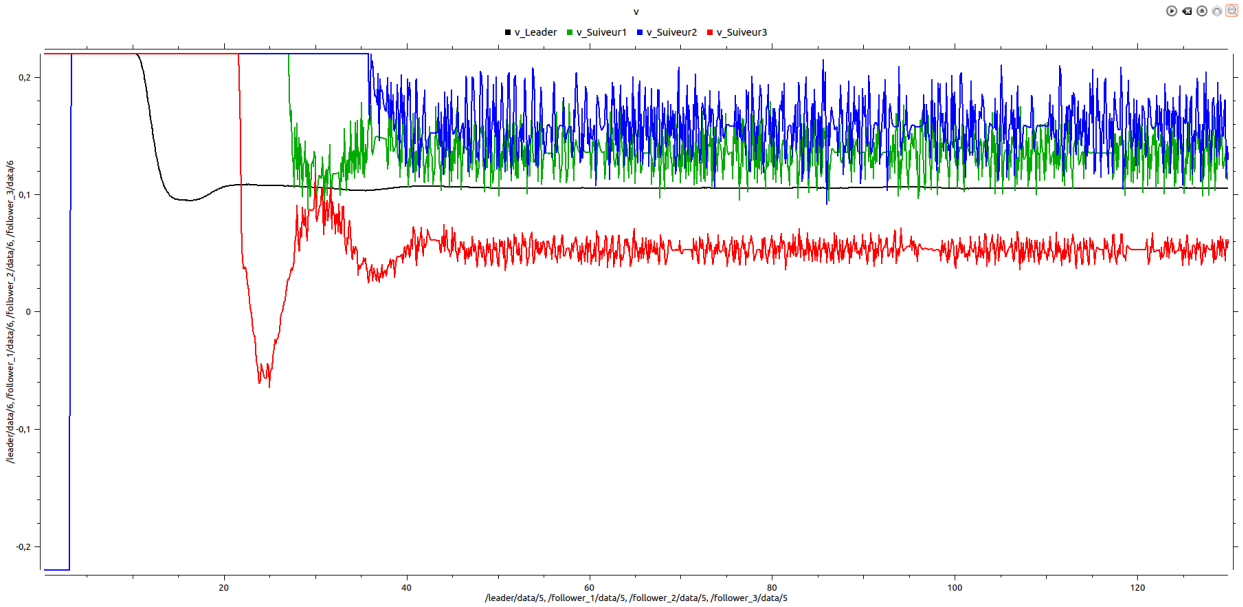


Figure 4. 13: The linear velocity

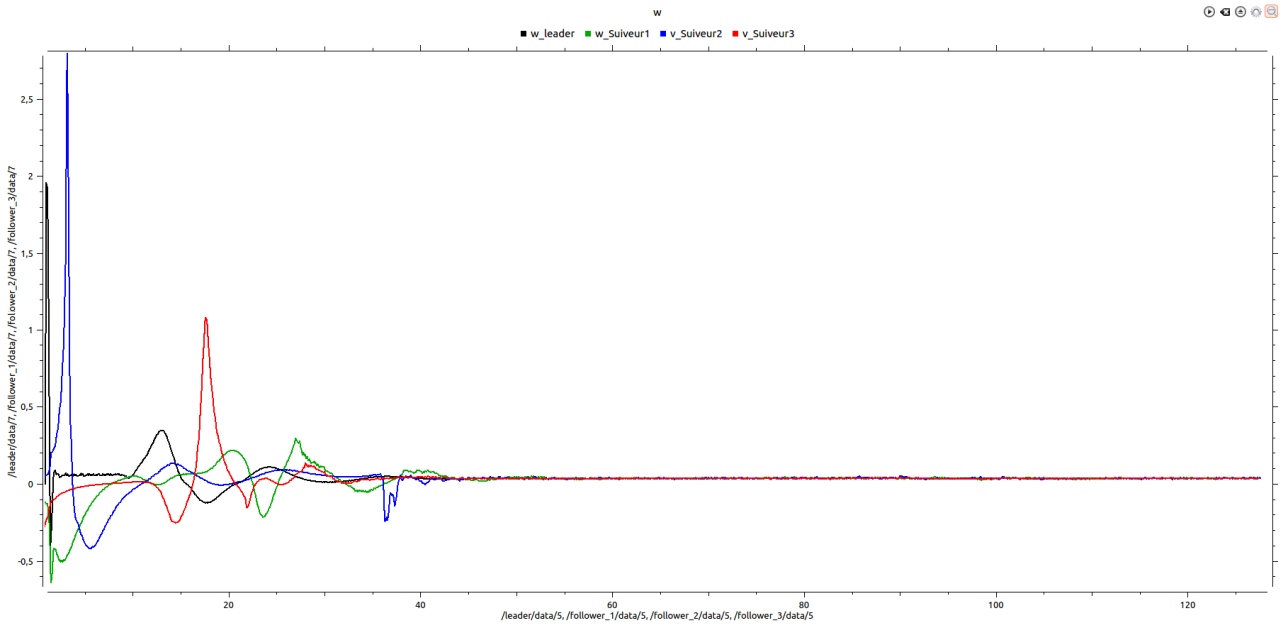


Figure 4. 14: The angular velocity

1. The first figure (**Figure 4.11**) shows a good coordination between the trajectories of the followers (red, green, blue) and that of the leader, all that while maintaining the desired separation and bearing.
2. The second figure (**Figure 4.12**) represents the tracking errors between the leader and the followers, we notice that all the tracking errors converge to zero after a given time.
3. The third and fourth figure (**Figure 4.13**) (**Figure 4.14**) represents the linear and angular velocities of the robots, the first part of these two graphs shows that both velocities start at a high value so the followers can catch up with the leader respecting the desired separation and bearing as fast as possible. Meanwhile, the second part shows slight modification of both velocities to achieve the desired separation and bearing.

IV.4.2.b: "S shaped" trajectory:

The goal of this scenario is to compare between the behavior of the robots when we change the leader's trajectory.

The initial positions of the robot are as follows:

	Initial pose	Desired separation	Desired bearing
Leader	X=2 m; y=4m ;θ=-0.7 rad		
Fol_1	X=1 m; y=7m ;θ=-0.7 rad	2	$\frac{8\pi}{7}$
Fol_2	X=-1 m; y=5.5m ;θ=-1 rad	1.8	$\frac{4\pi}{3}$
Fol_3	X=7m; y=6m ;θ=0 rad	1.8	$\frac{2\pi}{3}$

Table 4. 2: Initial parameters for the second scenario

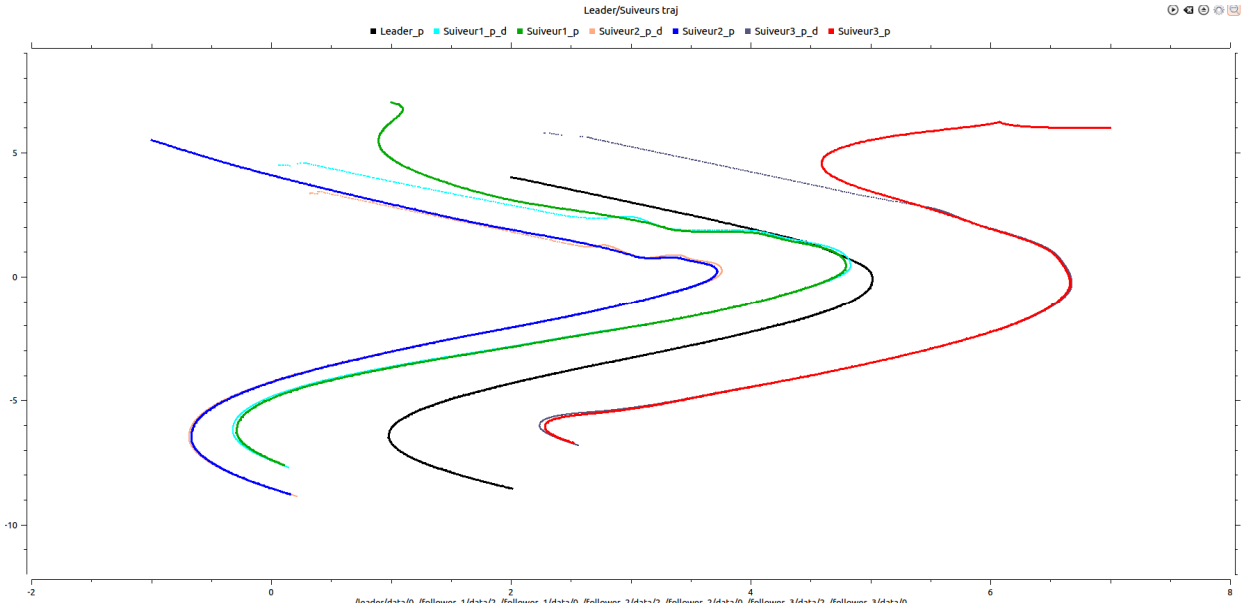


Figure 4. 15: The trajectories

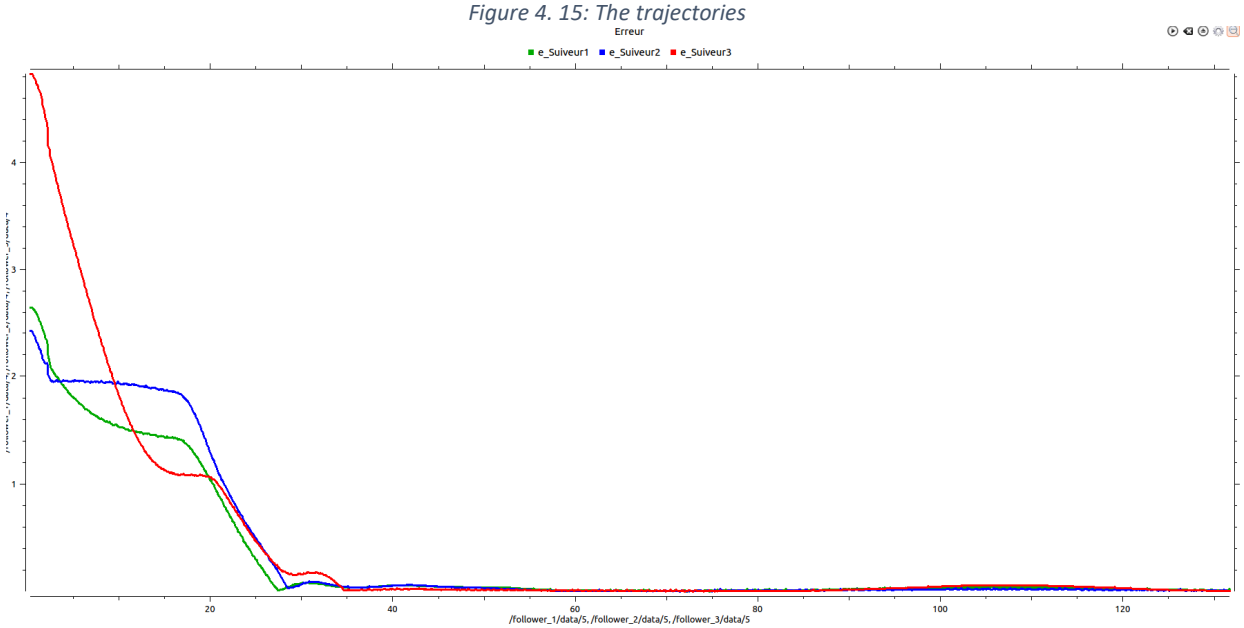


Figure 4. 16: The tracking errors

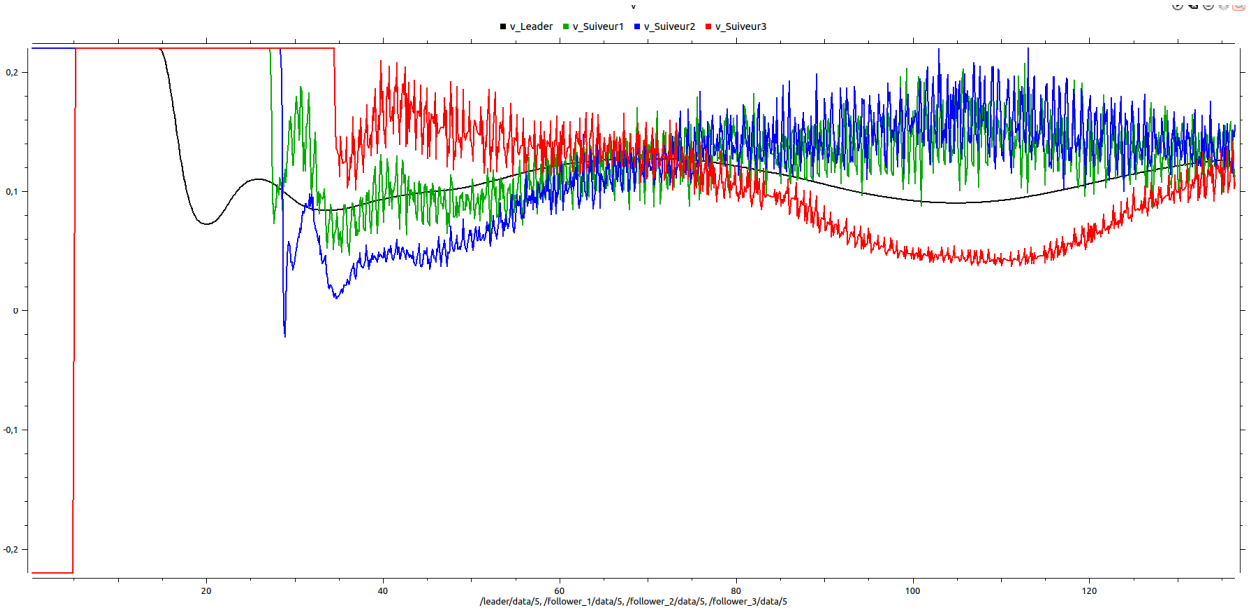


Figure 4. 17: The linear velocities

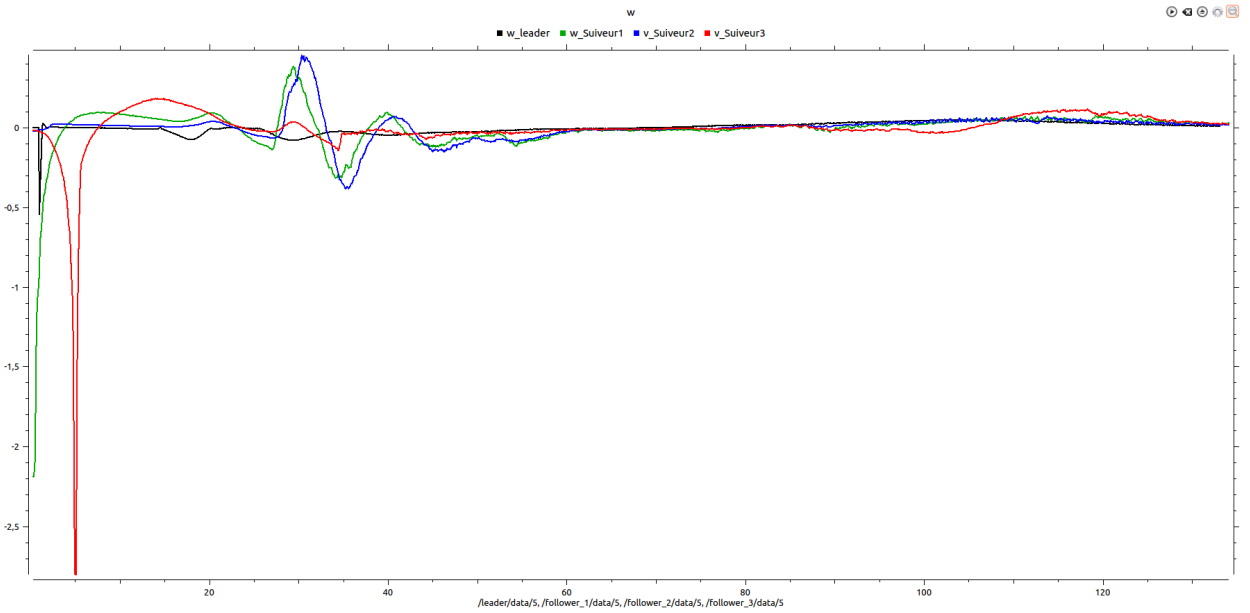


Figure 4. 18: The angular velocities

When we changed the desired trajectory of the leader, the followers keep up with the leader while maintaining the desired separation and bearing in a fast time. This shows the robustness of out controller.

The chattering problem can be negated using the SVSF but unfortunately we can't show the effects of the filter because of the weak performance of the processor in our PC .

IV.4.3-Feedback Linearization controller simulation

IV.4.3.a: rounded shape trajectory:

We will use the same parameters as the first scenario (IV.4.2.a) while using the feedback linearization method.

We get the following results:

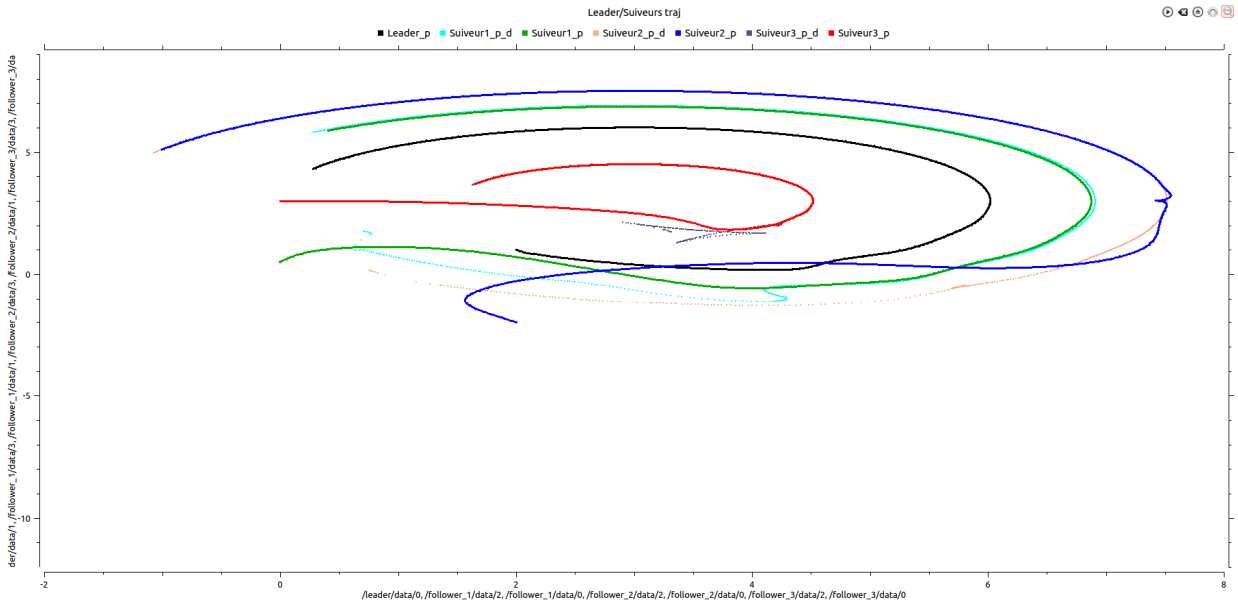


Figure 4. 19: The trajectories

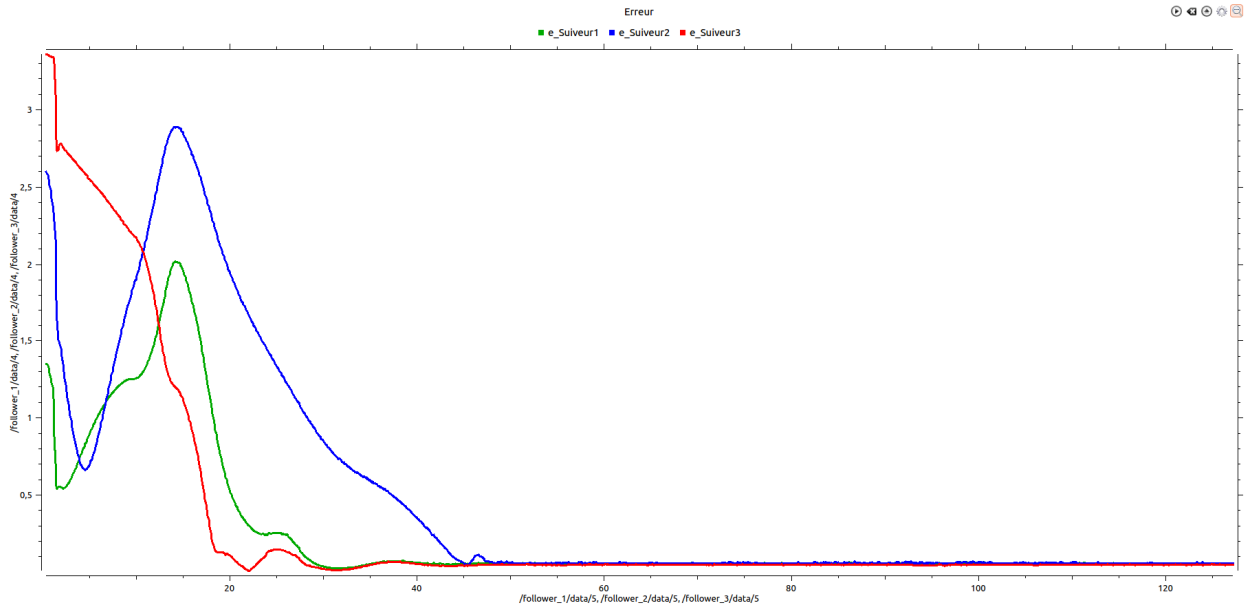


Figure 4. 20: The tracking errors

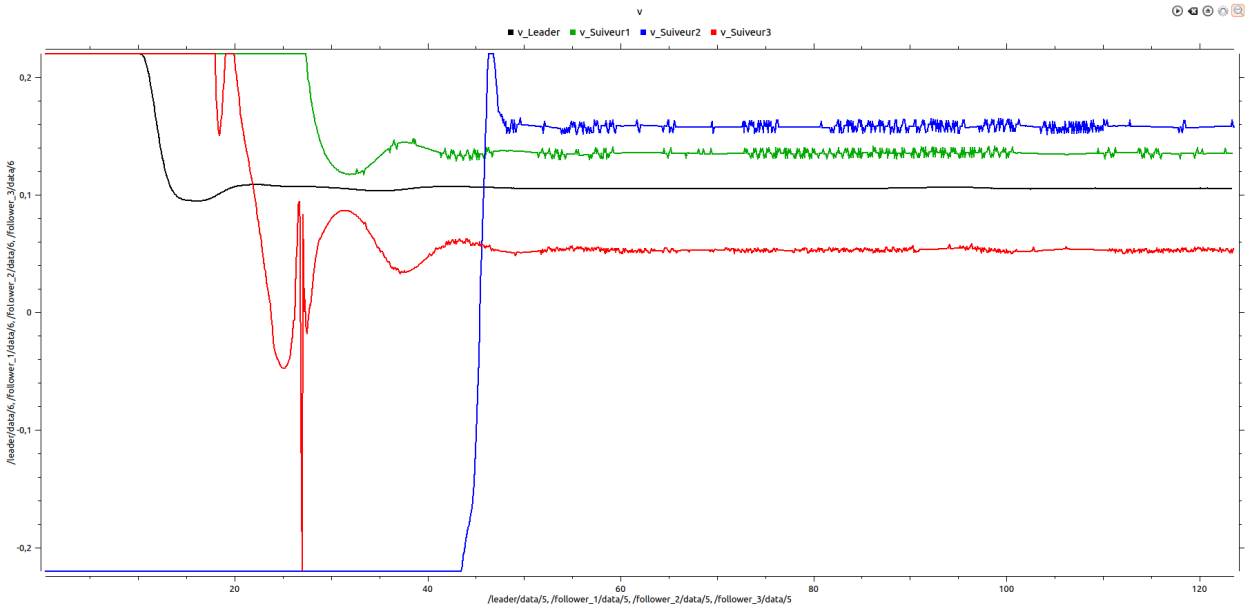


Figure 4. 21: The linear velocities

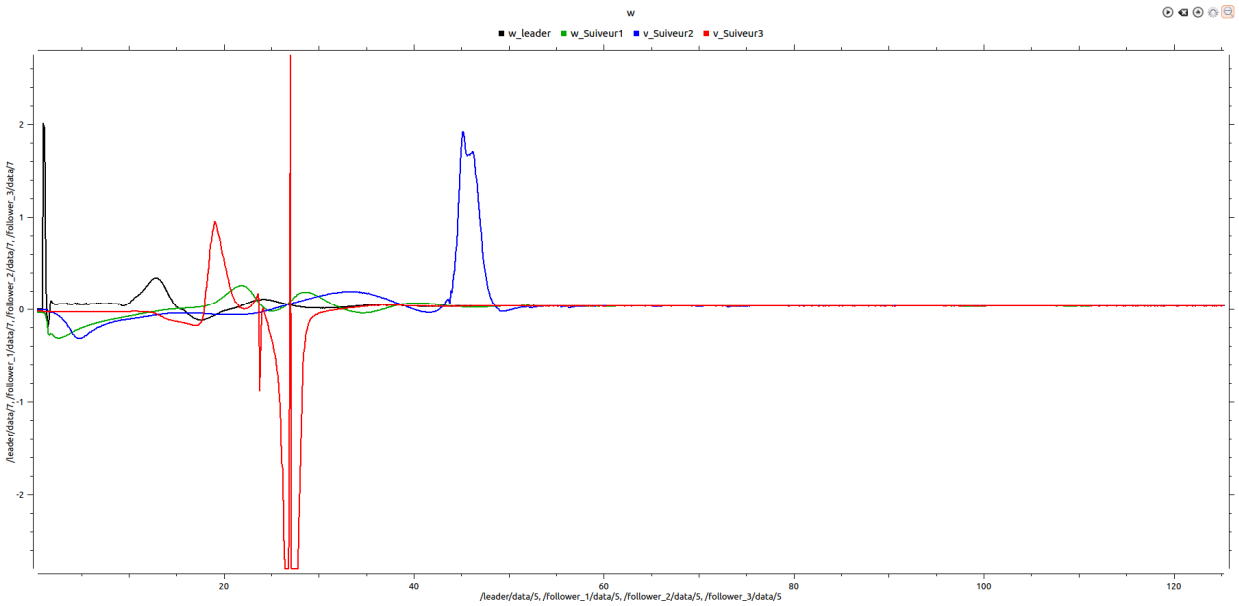


Figure 4. 22: the angular velocities

1. The first figure (**Figure 4.19**) shows a decent coordination between the trajectories of the followers (red, green, blue) and that of the leader, all that while maintaining the desired separation and bearing.
2. The second figure (**Figure 4.20**) represents the tracking errors between the leader and the followers, we notice that all the tracking errors converge to zero after a given time, but the convergence time is a lot slower than the previous control.
3. The third and fourth figure (**Figure 4.21**) (**Figure 4.22**) represents the linear and angular velocities of the robots, the first part of these two graphs shows that both velocities start at a high value so the followers can catch up with the leader respecting the desired separation and bearing as fast as possible. Meanwhile, the second part shows slight modification of both velocities to achieve the desired separation and bearing.

IV.4.2.b: "S shaped" trajectory:

The goal of this scenario is to compare between the behaviors of the robots when we change the leader's trajectory.

The initial positions of the robot are as follows:

	Initial pose	Desired separation	Desired bearing
Leader	X=5 m; y=5m ;θ=-1 rad		
Fol_1	X=-40 m; y=15m ;θ=1.2rad	1.8	$240\ deg$
Fol_2	X=1m; y=-2m ;θ=-1 rad	2	$205\ deg$
Fol_3	X=10m; y=-25m ;θ=0 rad	1.8	$140\ deg$

Table 4.3: Initial parameters for the fourth scenario

We got the following results:

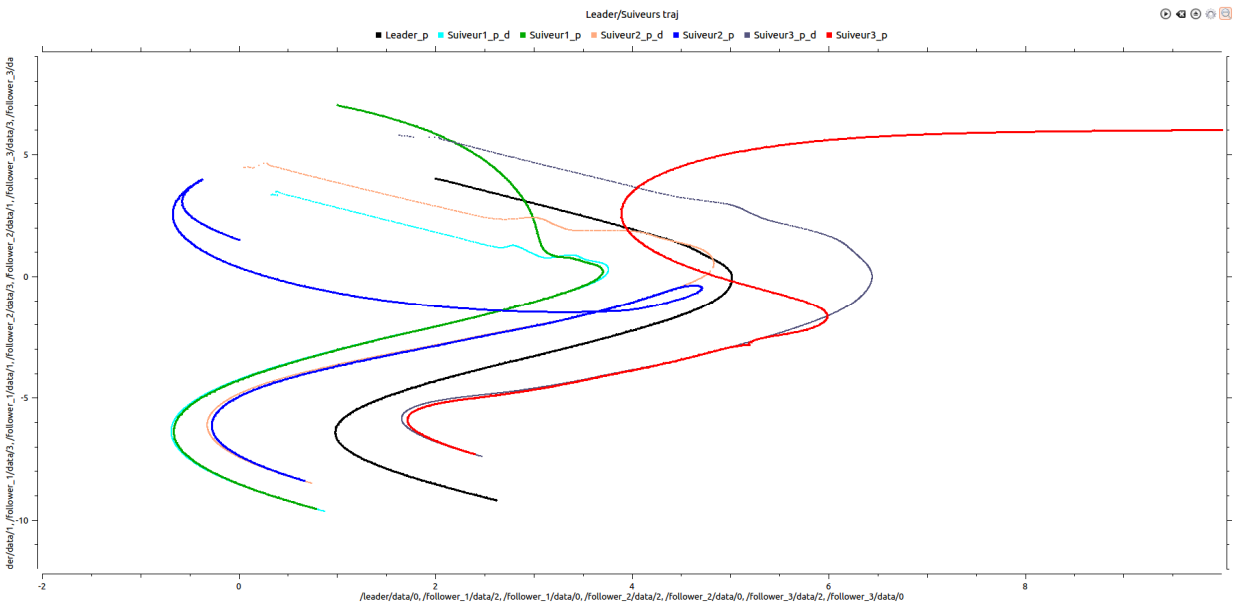


Figure 4. 23: The trajectories

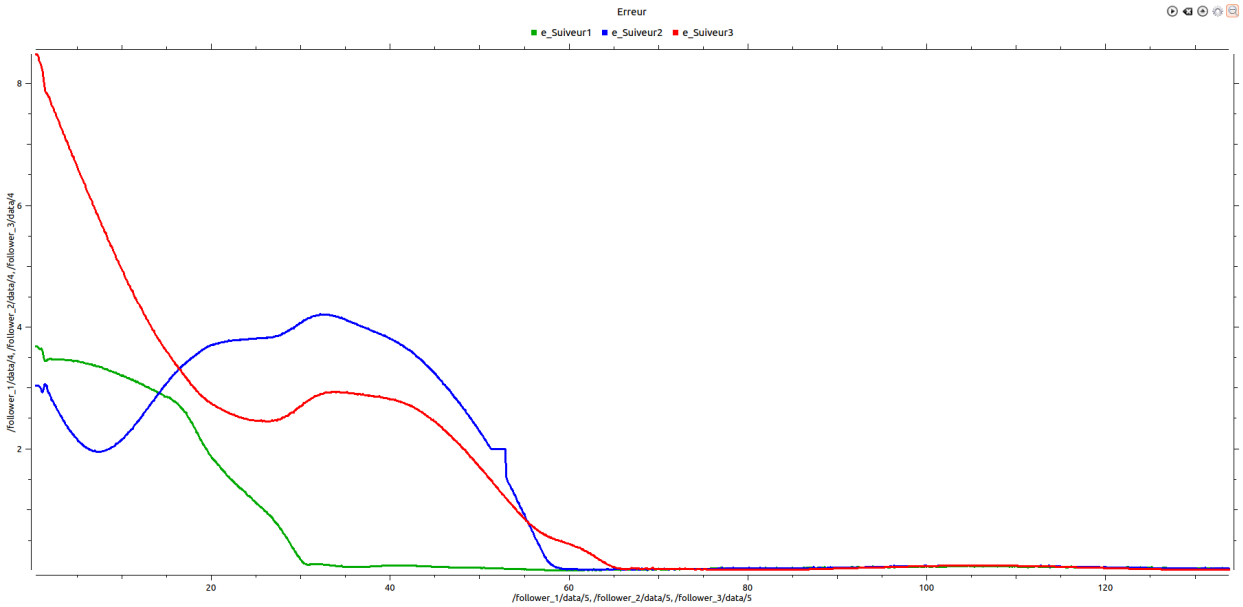


Figure 4. 24: The tracking errors

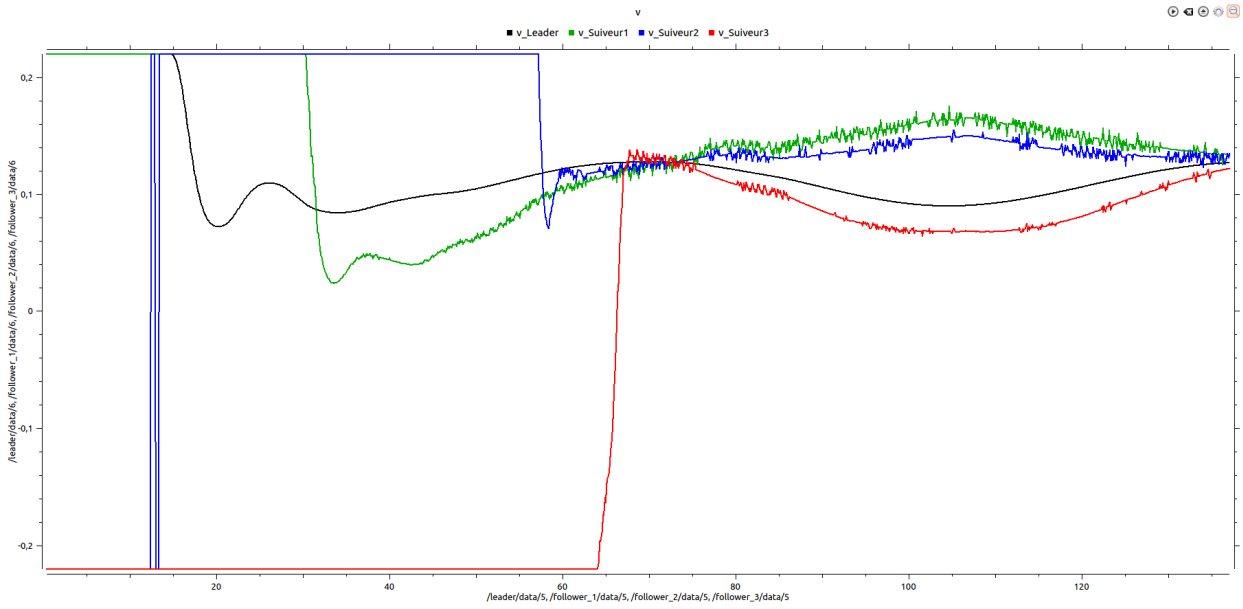


Figure 4. 25: The linear velocities

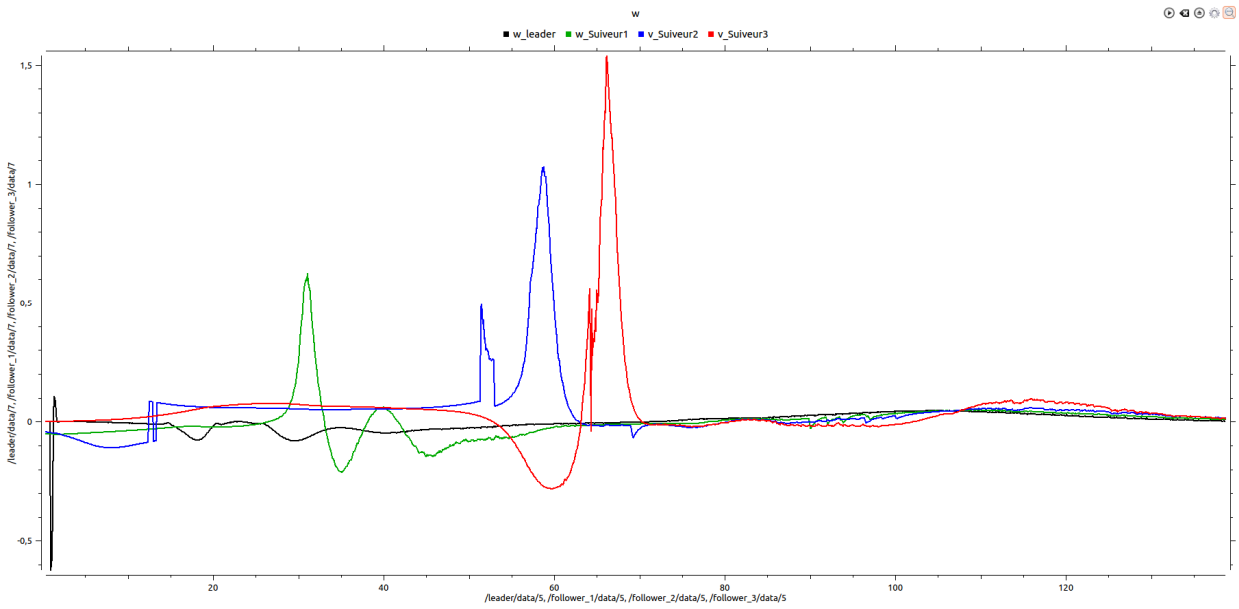


Figure 4. 26: The angular velocities

When we changed the desired trajectory of the leader, the followers keep up with the leader while maintaining the desired separation and bearing after a decent amount of time(slower than the first controller).

IV.5-Chapter conclusion

In this Fourth and last chapter we have introduced some basic informations about ROS and its building blocks and tools.

Then we simulated our Leader-Follower formation controllers on two different trajectories (“S” shaped & rounded) using the TurtleBot3 Burger because it better meets the needs of our project, and the availability of ready to use library in the ROS official web site.

The direct Lyapunov’s method controller gave us better results than the Feedback linearization controller (faster response times and better tracking).

General Conclusion

This thesis presents a multi-robot control framework using the leader-follower approach. The main objective of this dissertation is to study the feasibility of developing a leader tracking robot system using a sliding mode control system. In order to achieve this objective, the following objectives have been achieved in this thesis:

- Giving general introduction to the world of the multirobot systems.
- Introducing two methods of controlling the Leader-Follower formation using the feedback linearization method and the Lyapunov direct method.
- Simulation on MatLab.
- Simulation on ROS.

In this dissertation, we have well defined the multi-robot systems and we have presented the control structures in order to choose the hybrid structure, which was the most efficient. Then we exposed the control approaches that are available and we chose the leader follower approach. Which is the main subject of the dissertation. We have defined the feedback linearization and Lyapunov's direct method and their principle, we have developed its algorithm after the results of simulations on MatLab have shown the robustness of this control.

In the world of robotics, gazebo is the best-known software and the most efficient in 3D simulations intended to implement realistic scenarios. We built the environment there that made this project a reality. The Robot used is TurtleBot3, we carried out several scenarios in order to demonstrate the robustness of the control by Feedback linearization and Lyapunov's direct method using python. The simulation results on ROS and MatLab demonstrated the efficiency and stability of the robots and the tracking algorithm using the leader-follower approach, which also proved their correct functioning.

The prospects of this modest work are numerous until the writing. We can cite a few that are achievable (in the short and medium term) below:

- Leverage neural networks to remedy the Chattering problem, while keeping the advantages of the previous controllers.
- Creating an obstacle avoidance algorithm using artificial potential and creating a switching algorithm between formation tracking and obstacle avoidance using the Fuzzy Logic control
- Creating a SLAM robot equipped with a manipulator arm.

Appendix

Trigonometric formulas:

1. Periodicity Identities (in Radians)

These formulas are used to shift the angles by π/2, π, 2π, etc. They are also called co-function identities.

- $sin\,(\pi/2 - A) = cos\,A$ & $cos\,(\pi/2 - A) = sin\,A$
- $sin\,(\pi/2 + A) = cos\,A$ & $cos\,(\pi/2 + A) = -\,sin\,A$
- $sin\,(3\pi/2 - A) = -\,cos\,A$ & $cos\,(3\pi/2 - A) = -\,sin\,A$
- $sin\,(3\pi/2 + A) = -\,cos\,A$ & $cos\,(3\pi/2 + A) = sin\,A$
- $sin\,(\pi - A) = sin\,A$ & $cos\,(\pi - A) = -\,cos\,A$
- $sin\,(\pi + A) = -\,sin\,A$ & $cos\,(\pi + A) = -\,cos\,A$
- $sin\,(2\pi - A) = -\,sin\,A$ & $cos\,(2\pi - A) = cos\,A$
- $sin\,(2\pi + A) = sin\,A$ & $cos\,(2\pi + A) = cos\,A$

2. Sum & Difference Identities

- $sin(x + y) = sin(x)cos(y) + cos(x)sin(y)$
- $cos(x + y) = cos(x)cos(y) - sin(x)sin(y)$
- $tan(x + y) = (tan\,x + tan\,y)/\,(1 - tan\,x \bullet tan\,y)$
- $sin(x - y) = sin(x)cos(y) - cos(x)sin(y)$
- $cos(x - y) = cos(x)cos(y) + sin(x)sin(y)$
- $\tan(x - y) = \frac{tan\,x - tan\,y}{}(1 + tan\,x \bullet tan\,y)$

Bibliography and References

[1] Farinelli, Alessando & Iocchi, Luca & Nardi, Daniele. (2004). Multirobot Systems: A Classification Focused on Coordination. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society.

34. 2015-28. 10.1109/TSMCB.2004.832155.

[2] R. C. Arkin, Behavior Based Robotics. MIT press, 1998.

[3] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," Autonomous Robots, vol. 3, no. 4, 1996.

[4] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multirobot exploration. In Proceedings of ICRA'00, San Francisco, CA, USA, April 2000.

[5] Jens Wawerla and Richard T. Vaughan. A fast and frugal method for team-task allocation in a multirobot transportation system. In Proceedings of ICRA'10, Anchorage, AK, USA, May 2010.

[6] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. Autonomous Robots, 2000.

[7] Stergios I. Roumeliotis and George A. Bekey. Distributed multirobot localization. IEEE Transactions on Robotics and Automation, 2002.

[8] Raj Madhavan, Kingsley Fregene, and Lynne E. Parker. Distributed heterogeneous outdoor multi-robot localization. In Proceedings of ICRA'02, Washington, DC, USA, May 2002.

[9] Dorri, Ali & Kanhere, Salil & Jurdak, Raja. (2018). Multi-Agent Systems: A survey. IEEE Access. 6. 1-1. 10.1109/ACCESS.2018.2831228.

[10] Yan, Zhenzhen & Xiao, Liang & Liu, Jianzhou & Liu, Xing & Hu, Yumin & Wei, Qiuju & Liu, Xusong. (2014). Research on Applications of Multi-Agent System Based on Execution Engine in Clinical Decision-Making. 261-273. 10.1007/978-3-319-06269-3_28.

[11] Franzé, Giuseppe & Tedesco, Francesco & Famularo, Domenico. (2020). Resilience against Replay Attacks: A Distributed Model Predictive Control Scheme for Networked Multi-Agent Systems. IEEE/CAA Journal of Automatica Sinica. PP. 1-13. 10.1109/JAS.2020.1003542.

[12] Desai, J., Ostrowski, J. & Kumar, V. (1998). Controlling formations of multiple mobile robots. In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, vol. 4, 2864 –2869 vol.4.

[13] Mohamed Maghenem, Antonio Loria, Elena Panteley. Lyapunov-based formation-tracking control of nonholonomic systems under persistency of excitation. 10th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2016), Aug 2016, Monterey, CA, United States. Pp.404-409. ffhal-01357287f.

[14] YANG, Tian-Tian & LIU, Zhi-Yuan & Chen, Hong & PEI, Run. (2008). Formation Control and Obstacle Avoidance for Multiple Mobile Robots. Acta Automatica Sinica. 34. 588-593. 10.3724/SP.J.1004.2008.00588.

[15] Barca, Jan & Sekercioglu, Ahmet & Ford, A. (2013). Controlling Formations of Robots with Graph Theory. Advances in Intelligent Systems and Computing. 194. 563-574. 10.1007/978-3-642-33932-5_52.

[16] Li, Xiaohai & Xiao, Jizhong. (2005). Robot Formation Control in Leader-Follower Motion Using Direct Lyapunov Method. 10.

[17] Widyotriatmo, A., E. Joelianto, Agung Prasdianto, Hafidz Bahtiar and Y. Y. Nazaruddin. "Implementation of Leader-Follower Formation Control of a Team of Nonholonomic Mobile Robots." Int. J. Comput. Commun. Control 12 (2017): 871-885.

[18] Ahmed, Salman & Karsiti, Mohd & Loh, Robert. (2009). Control Analysis and Feedback Techniques for Multi Agent Robots. 10.5772/6597.

[19] Tzafestas, Spyros. (2013). Introduction to Mobile Robot Control.

[20] Slotine, J. & Li, W. (1991). Applied nonlinear control. Prentice Hall.

[21] MIT OpenCourseWare. "35.4 Rolling without Slipping: slipping and Skidding", Jun 2, 2017, https://www.youtube.com/watch?v=1UD560RQ684 , visited (Friday, May 7, 2021 at 3:44 am).

[22] Hermosillo, Jorge & Sekhavat, Sepanta. (2003). Feedback Control of a Bi-steerable Car Using Flatness; Application to Trajectory Tracking. 4. 3567 - 3572 vol.4. 10.1109/ACC.2003.1244101.

[23] BAHRIZ Taha & AMROUCHE Mahfoud. Contrôle d'un système multi-robots par mode glissant, embedded systems, University of SAAD DAHLEB BLIDA 1, 2020.

[24] Witkowska, Anna & Tomera, Miroslaw & Smierzchalski, Roman. (2007). A Backstepping Approach to Ship Course Control. Applied Mathematics and Computer Science. 17. 73-85. 10.2478/v10006-007-0007-2.

[25] Luc Jaulin. (2019). Mobile Robotics. Wiley.ISBN 1119663555, 9781119663553.

[26] Habibi, S. (2007). The Smooth Variable Structure Filter. Proceedings of the IEEE. 95. 1026 - 1059. 10.1109/JPROC.2007.893255.

[27] Sinclair, Andrew. (2010). Licence Profile: BSD. International Free and Open Source Software Law Review. 2. 10.5033/ifosslr.v2i1.28.

[28] Anis Koubaa. (2019)Robot Operating System volume 1&2. Springer International publishing .Softcover ISBN 978-3-319-85523-3

[29]Robotise-manual."TURTLEBO3", https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/, visited (Thursday, July 8, 2021 at 11:27am).

[30] Demim, Fethi & Boucheloukh, Abdelghani & NEMRA, Abdelkrim & Kobzili, Elhaouari & Hamerlain, Mustapha & Bazoula, Abdelouahab. (2019). An Adaptive SVSF-SLAM Algorithm in Dynamic Environment for Cooperative Unmanned Vehicles. IFAC-PapersOnLine. 52. 394-399. 10.1016/j.ifacol.2019.11.707.

[31] Kalman, R. E. (March 1, 1960). "A New Approach to Linear Filtering and Prediction Problems." ASME. J. Basic Eng. March 1960; 82(1): 35–45.

[32] MIT OpenCourseWare. "33. Left and Right Inverses; Pseudoinverse", May 7, 2009, https://www.youtube.com/watch?v=Go2aLo7ZOlU , visited (Friday, July 9, 2021 at 4:12 am).

[33] Mahmoodabadi, M. & Bagheri, Ahmad & Nariman-Zadeh, N. & Jamali, Ali & Maafi, R.. (2012). Pareto Design of Decoupled Sliding-Mode Controllers for Nonlinear Systems Based on a Multiobjective Genetic Algorithm. Journal of Applied Mathematics. 2012, Special Issue. 10.1155/2012/639014.