HIGH AND LOW LEVEL CONTROL FOR AN UNMANNED GROUND VEHICLE.

BYRON SNAIDER HERNÁNDEZ OSORIO

Universidad Tecnológica de Pereira
Maestría en Ingeniería eléctrica- Línea Automática
Facultad de Ingeniería Eléctrica, Electrónica, Física y Ciencias de la Computación
Pereira, Colombia
2019

HIGH AND LOW LEVEL CONTROL FOR AN UNMANNED GROUND VEHICLE.

BYRON SNAIDER HERNÁNDEZ OSORIO

Submitted for partial fulfillment for the
master degree in electrical engineering

Director:
(Ph.D.) Eduardo Giraldo Suárez

Research line:
Nonlinear control
Research group:
Control Automático

Universidad Tecnológica de Pereira
Maestría en Ingeniería eléctrica- Línea Automática
Facultad de Ingeniería Eléctrica, Electrónica, Física y Ciencias de la Computación
Pereira, Colombia

2019

Dedication

For my dear parents, because they have always inspired and supported me.

Acknowledgements

I would like to thank to "Universidad Tecnológica de Pereira" for welcome me as its student, and for give me the opportunity of continue with my master degree granting me the Jorge Roa Martinez scholarship. I would also thank "Maestría en Ingeniería Eléctrica" for acceptance and financial support during my master process. Finally, thank to "Colciencias" and the program "Jóvenes Investigadores e Innovadores por la paz, convocatoria 775-2017" for finantial support carrying out this research project.

Abstract

This research presents the development of a high and low level control methodology for a mobile robot or unmanned ground vehicle operating into a defined environment, the application of linear and nonlinear automatic control methods, jointly with search and planning algorithms provide the platform of autonomy.

The document begins with a background in locomotion systems for robots, explaining why, a ground vehicle is the most suitable choice for the application in the context of the outer research project. Water robots, unmanned aerial vehicles, and a wide variety of ground locomotion configurations like legged or wheeled structures are presented.

Then, for low level control implementations, dynamic models must be developed for selected systems, chapter 3 shows the approaches in differential driven mobile robots modelling.

The design and implementation of linear and nonlinear low level control method allow to show the difference between those approaches and state the advantages of using nonlinear control structures, a special result is obtained and detailed for a multivariable state feedback linearization or exact linearization controller.

For high level control, a description of graph theory, search algorithms, and path planning approaches are presented, the election and implementation of some algorithms is supported and detailed, the proposed implementation of the probabilistic roadmap algorithm shows an interesting result for the raised system and environment.

Contents

List of Figures

List of Tables

Introduction

Although during history, there appeared references to interesting devices that imitated the animals and human movements in some sense, it was only until past century that the word **robot** appeared and rapidly became a work field. The field of robotics became an important area in a wide variety of disciplines like science, mathematics and engineering.

Robots were initially developed for industrial applications, the first proper applied structure proposed is probably the arm manipulator, for repetitive movement and handling tasks. It is similar to human extremities keeping the tendency in robotics of imitate alive beings behaviours.

The expansion of manipulators allowed to carry out difficult, critical, dangerous, repetitive and inefficient tasks with more reliability, better performance and quality, and very important, with a considerable lower cost. This, aided the extension of manipulators to other areas like commercial and entertainment applications.

The difference between a robot and a fixed automation machine is **flexibility**. Fixed automation is designed only for an specific task and this is the central goal, meanwhile a robot could be programmed over and over again for carry out different tasks with the same efficiency. From the increasing need of labor work and expansion there appeared an important field on robotics nowadays, the **mobile robotics** that looks for expand the flexibility and advantages of robot usage to wider areas or work spaces. Current technologies allow even work with robots in remote areas promoting application of transportation and exploration.

This research is focused in the development of mobile robots for a landmine cleaning application, so the first part of this document makes a background of locomotion systems for mobile robots, special attention were included in differential driven wheeled and tracked robots. For task development it becomes important to point autonomous platforms.

Autonomy is an important topic in modern robotics and it is attained by jointly applying high level tasks and low level control for specific variables. One of the ultimate goals in robotics, with the introduction of intelligent robots concept is to create **autonomous robots**. Such robots may accept generic descriptions of tasks without many details and execute those tasks without further human intervention. The input descriptions only specify what the user wants that robots do, rather than how they do

it. The above implies the use of planning and searching algorithms for deciding each step or set of detailed simple tasks that the robot must do for reach the main generic task [1].

Robotic applications in which robots are designed for moving accross a given workspace, such as exploration or transportation, it is expected from an autonomous robot the ability to plan its own motions across the given environment, avoiding any object or obstacle in there, in literature this problem is stated as **motion planning** [2].

Motion planning is one of the most important tasks that autonomous robots must carry out. It involves the definition of a navigation path across a given environment avoiding any region defined as an obstacle, the found path must connect the initial or current robot position with a specific goal position.

There exist a large number of methods proposed for solving the path planning problem, these methods are based essentially in three main approaches: roadmaps, cell decomposition and potential fields. the document explains in detail the definition and implementation and some algorithms of these approaches.

But path planning is not the only important problem in development of autonomous robots, methods for tracking the found path are also very important. While path planning problem takes into account geometric and kinematic constraints, both of robot and environment, control methods for following the defined path take into account the dynamic behaviour of robots and this problem is addressed by automatic control [3].

Control tasks required in mobile robotics for path tracking imply the ability to reach a set of consecutive reference points or geometrical primitives, such as straight lines or curves given by the planner that performs the breakdown of high-level tasks [1].

Approaches, like shown in [4], [5] and [6], propose essentially linear control techniques which do not consider the complete nonlinearities of the differential driven robot, and it infers disadvantages, like deviation errors or over impulse in transient response. Tracking errors may cause collisions with obstacles due to deviation from the planned path [7].

Nonlinear methods has been proposed, for example, in [8] a nonlinear controller is designed but only for speed control, it reaches path tracking calculating an adaptive reference model using polynomial regressions. Other works like [9] propose adaptive algorithms, but they always have bad behaviour at the beggining which could cause path deaviations and consequently possible collisions.

One common method of nonlinear control is the exact feedback linearization, but it is mainly popular for SISO systems, the approach for MIMO systems has some advanced mathematical fundamentals, they are introduced in [10], those fundamentals are used for designing a nonlinear controller for the Differential Driven Mobile Robot (DDMR).

The structure of this document is as follows: Chapter 1 presents the background in robotics, general definitions and concepts; Chapter 2 is a review in mobile configurations

and its locomotion systems; Chapter 3 describes some mathematical models specially for the differential configuration; Chapter 4 defines and proposes the automatic control techniques, linear and nonlinear used in the robot model; Chapter 5 focuses on high level control tasks, it describes the graph theory, search algorithms and proposes the implementation of planning algorithms; Chapter 6 shows and discuss the results for high and low level control methods and algorithms implemented; finally, Chapter 7 concludes the document. Final part of the document presents discussion articles product of this research.

Chapter 1

Robotics

Robotics is a relatively new word that first appered in 1921 in the theater play called R.U.R (Rosum's Universal Robots) by Karel Capel; in his czech languaje, the word *robota* means to work or to serve. The adoption of the word **robotics** for referring the set of machines that carry out productive tasks came later, the difference between an automatic machine from a robot is essentially that robots could be programmed and re-programmed to develop flexible processes without losing productivity given by dedicated machines [11].

Nowadays, the word robotics is a widely known and used not only by scientist and engineers, but also by a wider community. Robotics defines all machines and devices intended for flexible working and production. A robot is a device capable of carry out processes normally made by a human being or sometimes by other alive beings, they are designed currently by engineers and scientists to make imitations of alive beings movements and behavior.

Robots have become very popular because of their application on difficult, dangerous or maybe impossible tasks for humans, space-limited, repetitive and even other kind of processes related not only to movements but also to decision making. Around 1990's the human labor increased its price while robots became not only cheaper but also more effective, faster, more accurate and specially more flexible [12].

In that sense, robotics is a multidisciplinary field of science and specifically in engineering that covers computer science, automatic control, mechanical and electrical engineering, among others. In general, a robot is composed by mechanical parts like frames, electrical parts like motor actuators, electronic devices like sensors and transducers, a communication system like an RF radio and an information processing system like a micro-controller or micro-processor, or even a computer. Mechanical engineering covers the study of static, kinematic and dynamic condition of the mechanical configurations, Mathematics supplies tools for describing motions and properties of robots. Control theory provides techniques to realize and track desired motions or force applications [11]. Electrical-engineering techniques play a role in the

design of sensors and interfaces for industrial robots, and computer science contributes a basis for programming these devices to perform a desired task, currently, the rising trend in the data science provides tools for information processing improving robots **perception** and **decision making**.

Figure 1.1 shows some examples of general purpose robots. Left side shows an autonomous and configurable robot, focused on the field of research in indoor applications, RB-1, provided by Robotnik, the middle one is an industrial collaborative system, RB-KAIROS, also provided by robotnik [13], and the right side one is a military vehicle used in urban intelligence, surveillance and reconnaissance (ISR) missions, XM1216, manufactured by iRobot and operated by US Army [14].

Figure 1.1: Some samples of general purpose robots.
Sources: [13],[14].

There are several types of robots depending on the task they must carry out, as shown in Figure 1.1 there could be mechanical configurations similar to human extremities, simplistic and advanced configuration for transportation and even synergistic combination of configurations.

Although the wide variety of robot configurations. it could be evidenced that there are two big branches to know when studying robotics: manipulators and mobile robots.

1.1 Manipulators

The concept of a robotic manipulator come together with the image of machines that make productive works and essentially imitate the movements and behaviour of alive beings or part of them as their extremities, which are used to handle materials, pieces, tools, or special devices relevant for some process, these handle tasks are done through varied movements programmed for carry out those processes.

A manipulator belongs to a set of devices commonly anchored to any base (main link), e.g: the floor, and coupled with a defined number of mobile links that provide the system of motion freedom, therefore, its work-space is bounded by the positions reached by the final end link, when all links are completely extended.

Until past decade, industrial applications were successfully addressed by manipulators, then, the Robot Institute of America defined an industrial robot as a manipulator programmable and multi-functional designed for carrying out flexibly different tasks as ones previously mentioned.

In the beginning of current century, the manipulators have taken a big strength in repetitive and accurate tasks like welding and assembly. Then, several discussions arose about what is o not a robotic manipulator and its difference with a simple machine, the most often answer is about flexibility, programmability and perhaps sophistication. Figure 1.2 shows a typical look for an industrial manipulator [12]. Machines which are for the most part limited to one class of task are considered simply fixed automation.

Figure 1.2: Robotic manipulator Adept 6.
Source: [12].

1.2 Mobile Robots

Mobile robots are essentially mobile structures that extend the application of flexible tasks to much bigger environments, those robots are not anchored to a place, but by the contrary, they have a locomotion system for auto-transportation, and movement through a wide defined field or workspace.

Although industrial robotics is generally referred to manipulators, and so, most books focus only on that robots, during 1880's and 1990's, mobile robotics began to cover special importance on literature since the rising growth of applications in autonomous transportation, navigation and exploration. Mobile robots can move in a terrain, on water or even, some can fly freely.

An important concept in the mobile robots field is autonomy, it refers to the ability of carry out complex tasks without further human intervention. An autonomous robot may accept generic descriptions of tasks without many details and execute those tasks following a sequence of operations decided by the robot itself [1].

It is important to consider the difference between an autonomous robot and an automated car for industrial applications like transportation of materials, pieces or products, they generally are cable guided, or include optical sensors for tracking of a line drawn in the floor, those applications refer to strongly structured environments while autonomous robots are featured by perception systems that allow to plan and control the system according with the information read from the environment, highest levels of autonomy assume no acknowledgement of the environment, the system must build a representation from its perception, this is one of the most difficult tasks, and a current open field of research in science and engineering.

Figure 1.3 show some examples of autonomous mobile robots.

Figure 1.3: Mobile Robots.
Sources: [13], [15], [16].

Next chapter will be focused on different models and configurations for mobile robots widely developed by researchers in last decades.

Chapter 2

Common Models of Mobile Robots

As previously said, a mobile robot could be defined as an autonomous system both for indoor applications and outdoor navigation, it is about that the robot is intelligent enough for fast reaction and make decisions according to the observation of the environment, this is reached through a perception system consisting of different sensors that provide specific information about the environment, a central bran, generally a computer, that processes the information and generate high and low level control actions, and finally, the actuators or final control elements, these always include a locomotion system, and sometimes other elements like e.g: a robotic arm.

Some examples of mobile robots are humanoids, unmanned space exploration rovers, entertainment pets, drones, military exploration ground and water vehicles, among others, they could be classified essentially by the kind of locomotion system, according to their bioligual counterpart, they can for example walk, run, jump, and so on.

2.1 Locomotion System

According to the biological principle the systems use to perform their motion, they can be divided into robots that can walk, run or jump based on the following structures:

- Legged Robots (LR)
- Walking Robots (WR)
- Flying Robots (FR)
- Swimming Robots (SR)
- Wheeled Mobile Robots (WMR)
- Tracked Mobile Robots (TMR)

- Articulated Robots (AR)
- Others

Trends in Mobile Robotics are led by artificial intelligence, autonomous driving, network communication, cooperative work, nanorobotics, friendly human–robot interfaces, safe human–robot interaction, and emotion expression and perception. Furthermore, these news trends are applied to different fields such as medicine, health care, sports, ergonomics, industry, distribution of goods, and service robotics. These tendencies will keep going their evolution in the coming years [17].

Locomotion system determines a specific problem in mobile robotics, it is solved by understanding the mechanism kinematics, dynamics, and determining an appropriate control scheme.

The robot's locomotion system is an important aspect of the mobile robot design, understanding and control, and it depends not only on the medium in which the robot moves (on the Earth's surface, under water, in the air, etc.) but also on technical criteria such as maneuverability, controllability, terrain conditions, efficiency, stability, and so on.

2.2 Aerial Robots

An aerial robot, sometimes called as drone, is a device designed and inspired to operate as an airplane normally do, but with no crew in it, a more common name is Unmanned Aerial Vehicle (UAV) defined as an aerial system that operates autonomously. The most advanced UAV systems can now take off and land completely with only high level instructions. At their beginning, the UAV were mostly used in military applications but they have quickly expanded their operation to other applications such as agricultural, scientific, commercial, surveillance, product delivering, distribution, logistics, aerial imaging, altimetry, and even recreational [17]. Figure 2.1 shows the aspect of some common configurations used in UAVs.

Figure 2.1: Unmanned Aerial Vehicles.
Sources: [18], [19], [20].

2.3 Water Robots

Sea and undersea applications are also important topic in the field of robotics. One of man's oldest goals has been to explore the oceans and underwater areas that are inaccessible to him. The ocean is rich in several mineral, marine biological and energetic resources and they can play an important role on a sustainable development of human society. Oceanographers can predict earthquakes, tsunamis and other natural disasters to reduce the damage to human beings by observing the ocean; Biologists make related research by observing the plankton, microorganisms and other living species in the ocean [21].

The observation, exploration, surveillance, cleaning and even renewable energy generation from ocean, are important topics of research nowadays, many devices have been built for this, including robotic systems. Again, the locomotion system of those devices are inspired in biological systems, Figure 2.2 show some examples of sea animals and humanoid based systems.

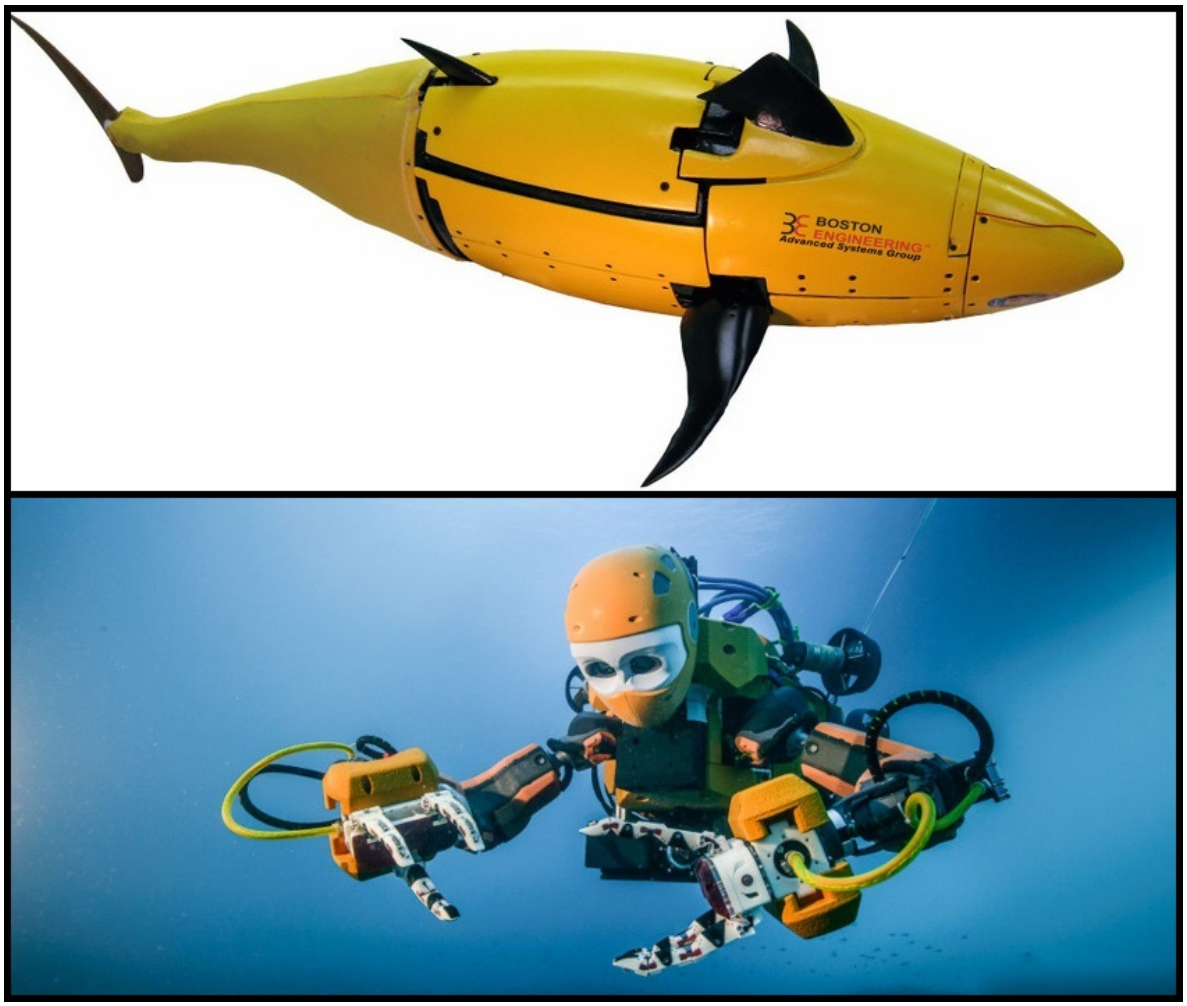


Figure 2.2: Underwater systems.
Source: [22].

As an important branch of mobile robots, the underwater vehicle manipulator system is one the hottest research topics nowadays. OceanOne is an example of a submarine robot. It is a humanoid robot that explores the seabed. It takes advantage of the best of remotely operated vehicles and the advantages of humanoid robots, such as having a robotic hand with which to rescue objects as if it were a human being [23].

2.4 Legged Robots

Legs are one common form of locomotion, giving rise to walking robots. Legs allow to isolate the moving body from the terrain using some specific points of support. The use of legs allow to define a method for passing over obstacles, which implies an advantage over other ground locomotion systems, the biggest handicap of legged systems are probably their price given advanced methods and technology they must use for planning each leg movement or to guarantee the platform stability.

Other important advantages of legged systems are: transversality and efficiency, the fact that they can also move on soft and uneven terrain, better mobility and a smaller impact on the ground, in short, adaptability and maneuverability on rough terrain [17].

In walking robots, stability is the main issue, as the balance of the body and gait are of extreme importance, the mechanism complexity is necessary higher, and sometimes, there is also more the energy consumption. In principle, control (high and low level) is a complex task in these robots because it must guarantee the system stability, both static and dynamic. Static stability refers to the ability of maintain a configuration while dynamic stability is about keeping upright under reaction and inertial forces produced by movements or disturbances.

There are many types of walking robots depending on the number of legs. The most common and important configurations are: biped (two legs) also known as humanoids since their anthropomorphic form, quadruped (four-legged) emulating lots of mammals, six-legged (hexapod - do not confuse with hexapod manipulators) inspired essentially in arthropod animals, and so on. then, a far review of walking configurations are drown.

2.4.1 One Leg Robots

A one leg robot is essentially a jumper robot since they cannot reach static stability, they just cannot stand still on a position, but they can cross terrains and jump some obstacles, one important example is Uniroo, developed by MIT [24]. Figure 2.3 shows this example of one leg robot.

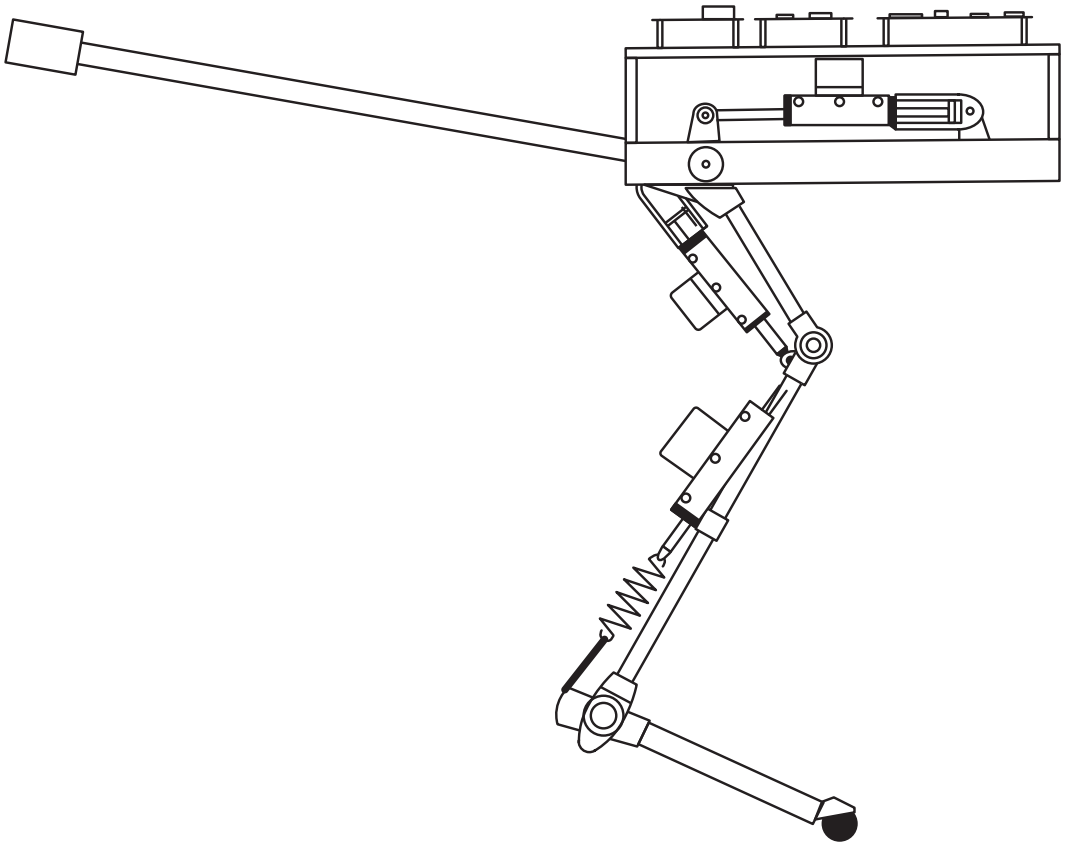


Figure 2.3: Hopper (Jumper) Robot, MIT Uniroo.
Source: [25].

2.4.2 Two Legs Robots

As previously mentioned, it is one of the most important configurations of legged robots, this kind of robots are inspired in human locomotion, so they are called humanoids, one of the most important problems from this point of view is loss of balance.

The motion of bipedal robots is dependent upon dynamic stability. These robots can walk, run, go up and down stairs and even do sophisticated jumps. The system balance and stability has to be dynamic all time. High and low level control are advanced tasks that must take into account observation of the environment for local planning, and also carry out global path planning efforts. Figure 2.4 shows the robot Cassie, a bipedal system developed by Agility Robotics, a spin-off of Oregon State University.

Figure 2.4: Bipedal Robot Cassie by Agility Robotics (Oregon State University). Source: [26].

2.4.3 Three Legs Robots

These kind of walking robots are not very common because of the odd number of legs. STriDER is an example of a robot with three legs presented in [27], it shows a straightforward kinematic structure. It is easier to control than previous legged robots since it is inherently stable like a camera tripod, the article presents evidence of energy efficiency given its lightweight configuration. In order to walk, the system uses two legs as supportive legs while the third one works as a swing leg.

2.4.4 Four Legs Robots

Robots with four legs are called similar to their biological counterpart: quadrupeds, since they are inspired on them. Quadrupeds are used when increased safety or payload capability is needed. They have the advantage that static stability is inherently solved, but they require dynamic walking control, as the gravity center is changing step by step. The control and leg coordination of robots with four and more legs is, therefore, more complicated.

One of the most famous and advanced quadruped robot is BigDog, presented by Boston Dynamics in 2008 [28] and continuously improved. Figure 2.5 shows a cutaway illustration by James Provost.

Other relevant references of quadruped robots are Cheetah [29] by MIT and Spot [30] also by Boston Dynamics.

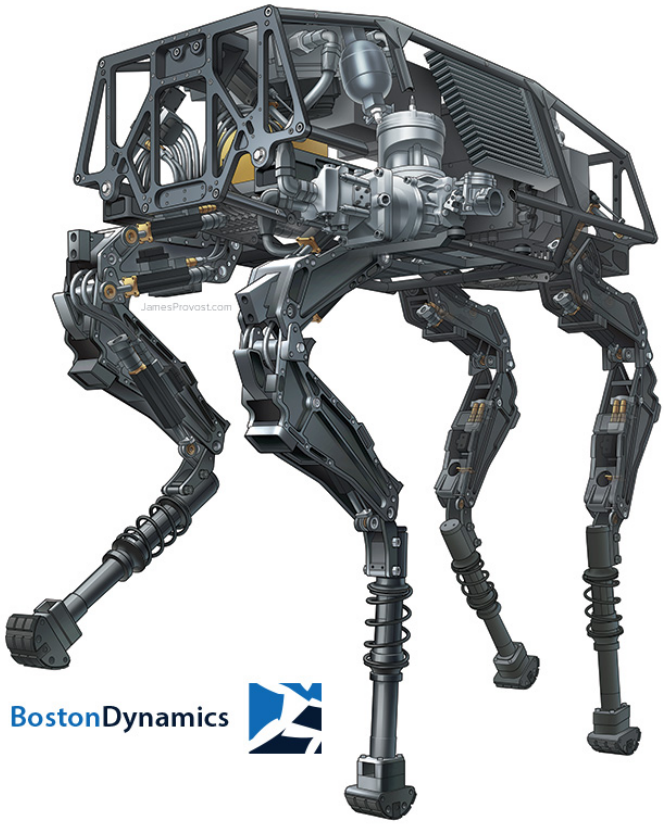


Figure 2.5: BigDog by Boston Dynamics.
Source: Illustration©James Provost.com

Configurations for five, seven, and in general odd number of legs, are least common in literature and applications. Six and eight leg robots are, in practical terms, almost the same, They are inspired by spiders, underwater walkers, and other arachnids, next section presents the hexapod configuration.

2.4.5 Six Legs Robots

Robots with six legs, also known as hexapods have some control advantages from fewer legs configurations, a robot with six or more legs can be controlled with static walking techniques rather than dynamic walking, thus reducing the control complexity. A straight gain in walking could be reached by a three leg step, which guarantees stability.

The challenge of using hexapod configurations is essentially the mechanism design, and reaching good movement speeds.

Figure 2.6 shows the six-legged walking robot LAURON V with improved kinematics and robust mechanical structure. Each leg has four independent joints that enable the platform to cope with steep inclines, large obstacles and makes it possible to manipulate objects with its front legs. Autonomy, robustness and a large payload capacity together with its impressive terrain adaptability make LAURON V highly suitable for wide range of terrain applications [31].

Figure 2.6: LAURON V: Six legged walking robot.
Source: [31].

2.5 Wheeled Robots (WR)

Wheeled robots are one of the simplest and most important solutions for robot locomotion. Wheels are very efficient to get mobility on hard enough and low hindered terrains or environments, and they allow to reach relatively high velocities.

The use of wheels is simpler than using legs or other locomotion systems. wheeled robots are easier to design, build, and program when the robot is intended to moving on a flat, near even terrain. They also tend to be much cheaper than their legged counterparts. Wheel control is less complex since static stability, they do not present any great difficulty in terms of balance issues, because all wheels are usually in contact with the surface.

Most significant limitations when using wheels are related to impulsion slipping, adaptability and sometimes vibrations. Wheeled robots are not very good at navigating over obstacles, such as rocky terrain, sharp surfaces, or areas with low friction as soft terrains.

Wheeled robots could be classified according to the number of wheels or to specific configurations that are worth mentioning

2.5.1 Two Wheels Robots

Robots with only one wheel are not common, they are called unicycle robots and they are inherently unstable systems. Both longitudinal and lateral stability controls are needed simultaneously to maintain the unicycle's posture.

More interesting is the two wheels robot, that with the reduced number of wheel could reach interesting results. These have two alike parallel, conventional wheels linked to each side controlled by two independent actuators. or sometimes follow a bicycle structure, where the wheels are aligned and it has a steering control. It is also considered that the wheels are perpendicular to the ground and the contact between them is pure rolling, that is, no slipping effects are considered. In any case, the stability problem is a real challenge. Two interesting references are the cleaner robot Roomba by iRobot [32], shown in Figure 2.7, and the classic GhostRider robot by DARPA [33].

Figure 2.7: Roomba robot for home cleaning.
Source: [32].

2.5.2 Three Wheels Robots

Three wheeled robots could be presented in two different formats, the first one corresponds to a configuration in which two parallel wheels are driven by independent actuators, and a third wheel is free, Figure 2.8 shows an example of this configuration, the second configuration is detailed in the next section.

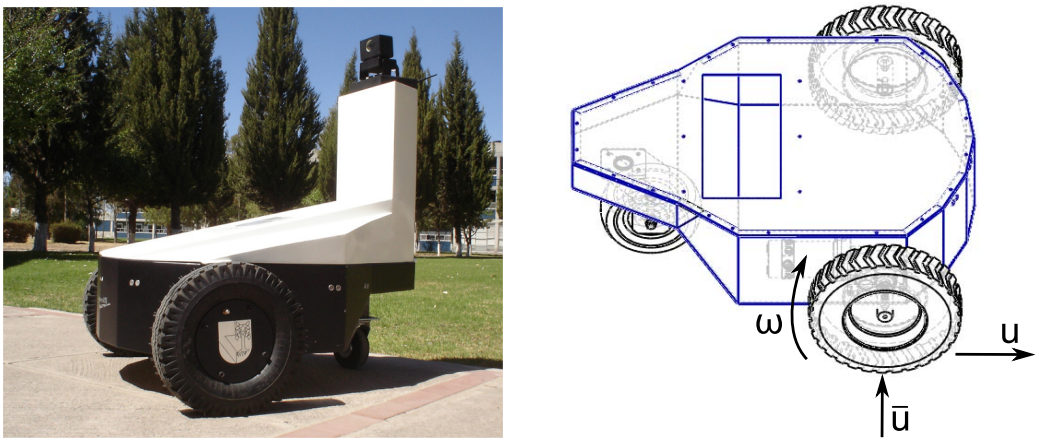


Figure 2.8: Three Wheel Differential Robot.
Source: [34].

2.5.3 Tricycle Robots

Tricycle robots are a classic configuration of robots with three wheels, unlike the previous one, this configuration drives the parallel pair of wheels with a single actuator, and uses a second actuator for steering the robot through the third wheel. Figure 2.9 shows a representation of the tricycle robot.

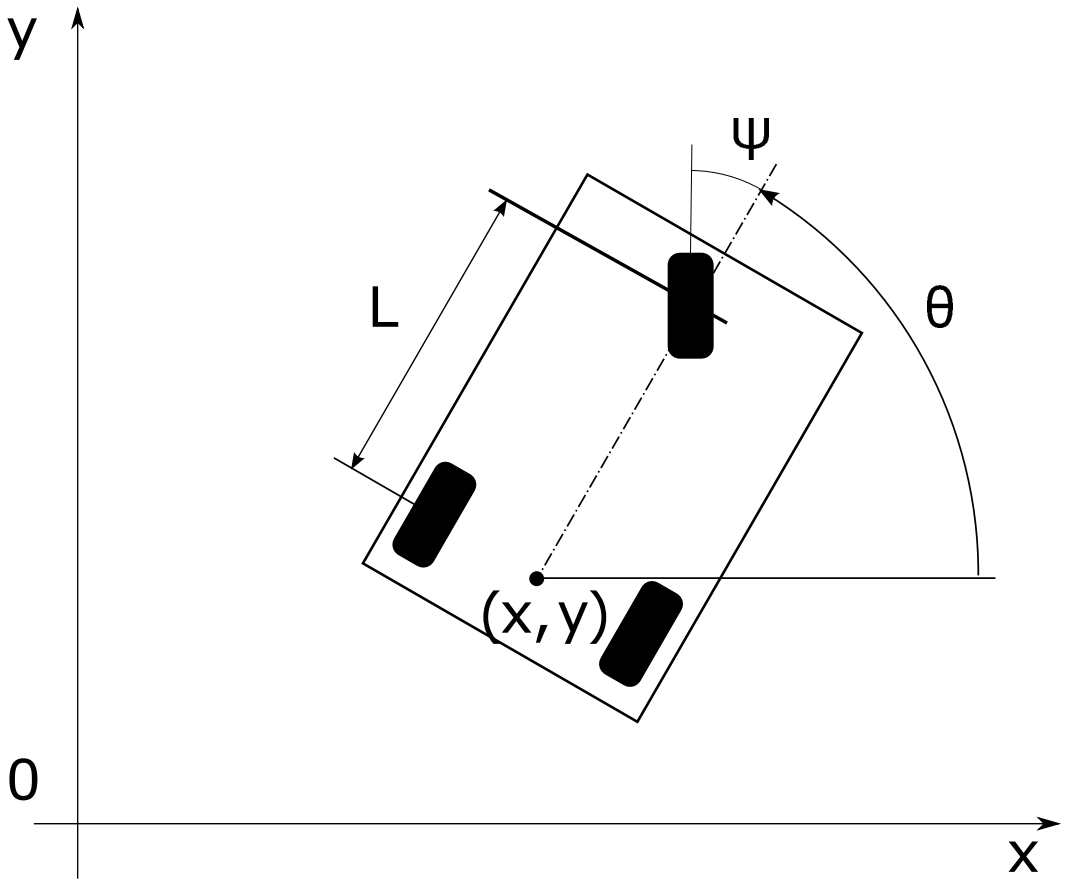


Figure 2.9: Schematic representation of tricycle robot.
Source: [35].

There exist variations of the tricycle configuration, in which the rear axle is passive, and both, steering and traction are driven by front wheel.

In general, manoeuvrability is better than in other configurations, but it could have stability problems in difficult terrains, the mass center can move when moving on a slope which may cause traction loss [11].

2.5.4 Four Wheels Robots

Four wheeled robots are more stable than two and three wheeled ones, that is because the center of gravity is located inside the rectangle formed by the four wheels, furthermore, it implies four self-complementary supporting points. The wheels can be deferentially steered, two-by-two powered wheels, or can have car-like steering. in the Figure 2.10, the Wheeled Mobile Robot Enyo is shown. Its mechanical structure is based on a four wheel differential-drive configuration driven by a belt system. Two active front wheels transfer rotating motion to the two passive rear wheels through belts.

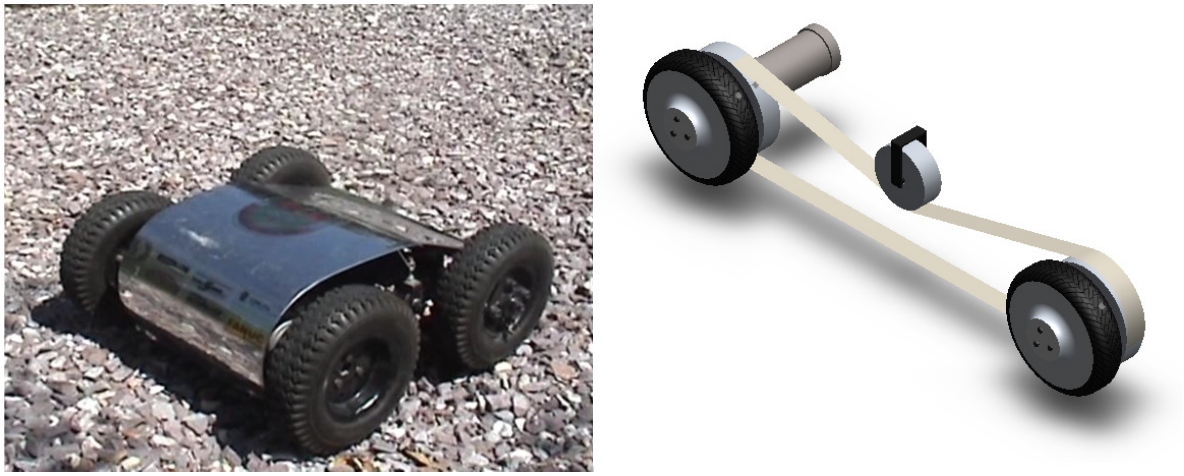


Figure 2.10: Four Wheel Differential Driven Robot Enyo.
Source: [34].

2.5.5 Ackerman Robots

Car-like mobile robots are very important. Google and Uber self-driving cars, AIVs, and other important autonomous self driving prototypes have this configuration. They are acquiring a great importance in transportation, logistics, food industry, and food processing. The classical structure name is Ackerman, Figure 2.11 shows a schematic representation of this structure.

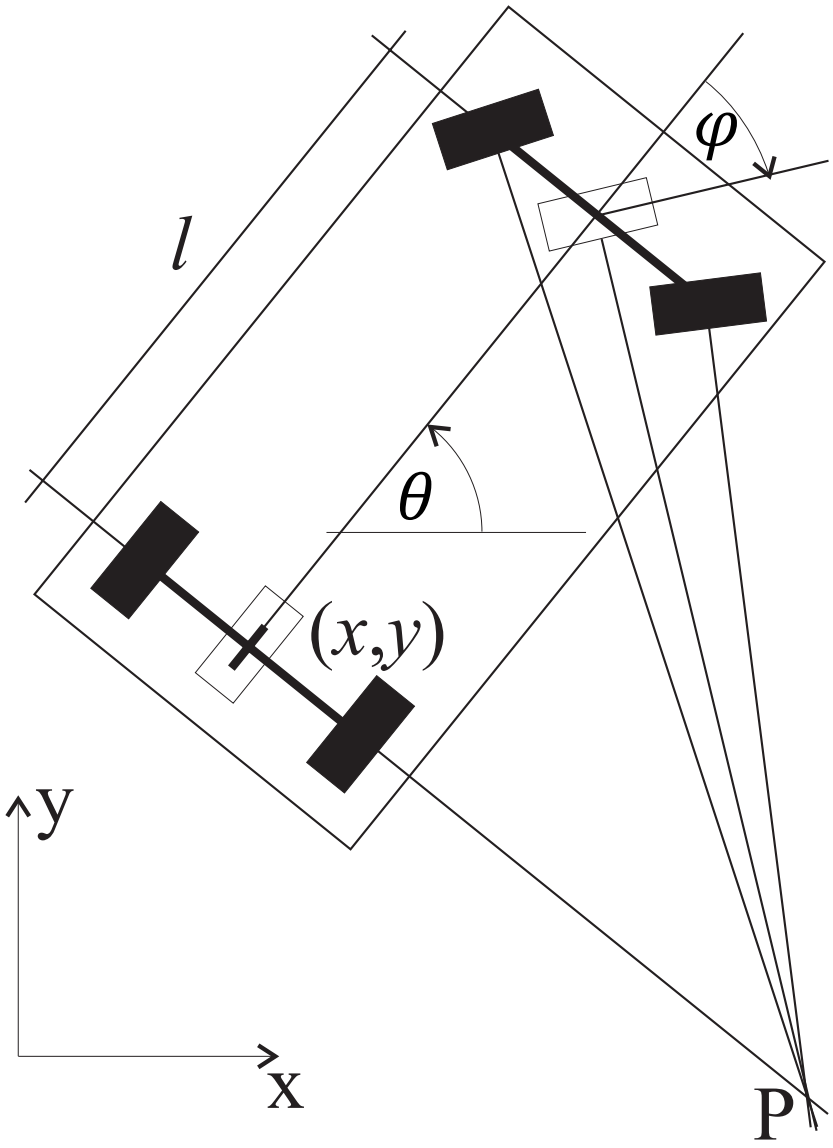


Figure 2.11: Ackerman robot system.
Source: [36].

2.5.6 Five Wheel Robots

The efforts on designing five wheeled robotic systems are done in applications for moving through outdoor rough terrains, they improve the contact and stability. One example is presented by Xu et al at [37], they first proposed an innovative asymmetrical prototype of a five wheeled robot with reconfiguration features which can overcross obstacles and climb on slopes. In reference [38] the authors study the slip on that five wheeled mobile robot when moving on an uneven terrain.

2.5.7 Six Wheel Robots

Six wheeled robot configurations are used when maximum traction is needed, they are normally provided of a suspension system which keeps all six wheels in contact with the surface and helps them go over slopes and sandy terrain. Some good examples are rovers used by NASA for mars exploration, Figure 2.12 shows the Sojourner used in NASA's Mars Pathfinder mission in 1997.

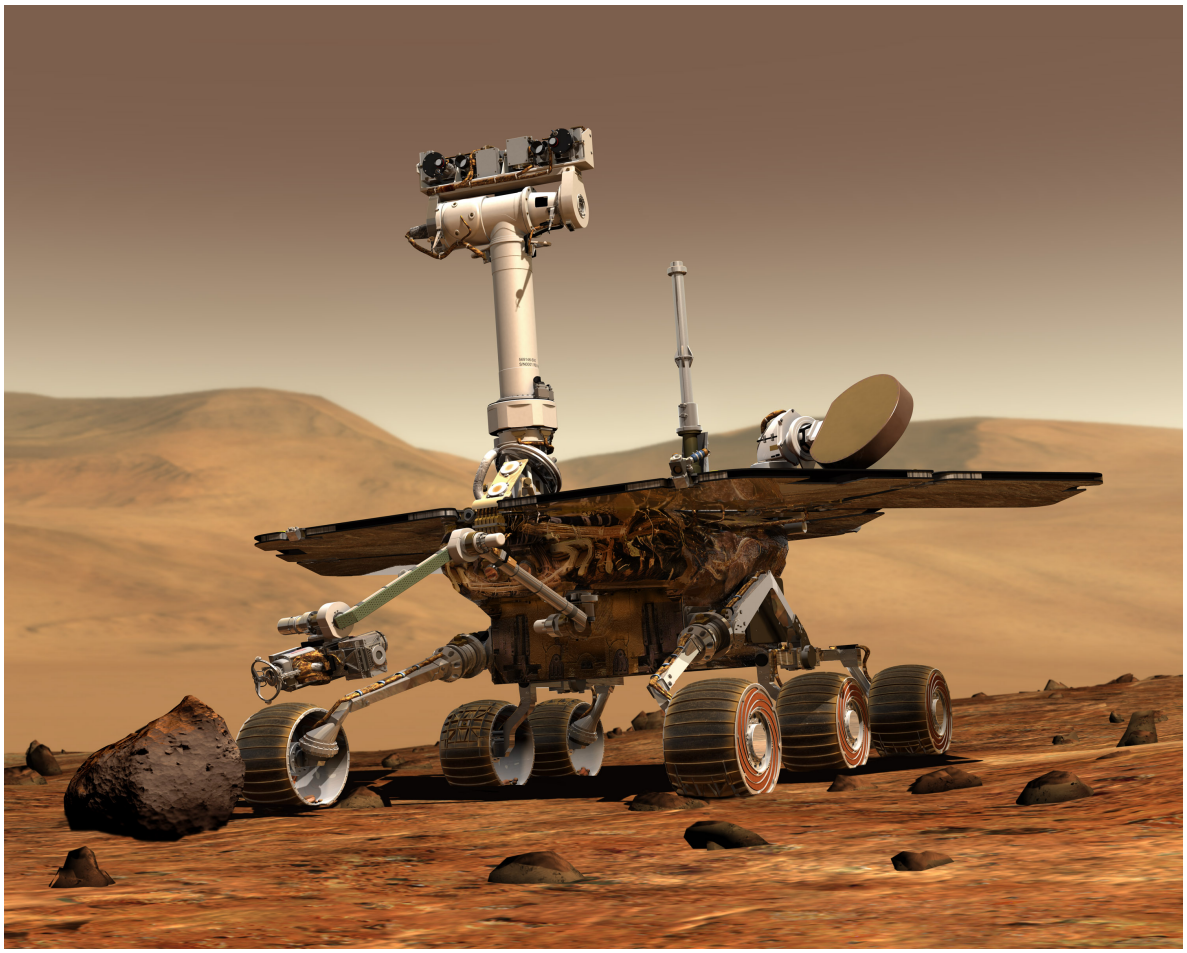


Figure 2.12: NASA Mars Rover.
Source: [39].

2.5.8 More Than Six Wheel Robots

Those are particular configurations used for special applications, some authors put here the articulated wheeled robots, used form climbing large obstacles and overcome lots of terrain challenges [17].

2.5.9 Omnidirectional Wheeled Robots

An interesting configuration explored by researchers are the omnidirectional robots, there exist variations on that configuration, but almost all of them use the Swedish wheels or active caster wheels. Common models use four of these wheels, but one variation, like the prototype NG [34] shown in Figure 2.13 use only three of them, as seen. The main advantage of these Wheeled Mobile Robots is that they exhibit holonomicity, that is, the ability to move in any direction without an orientation change.

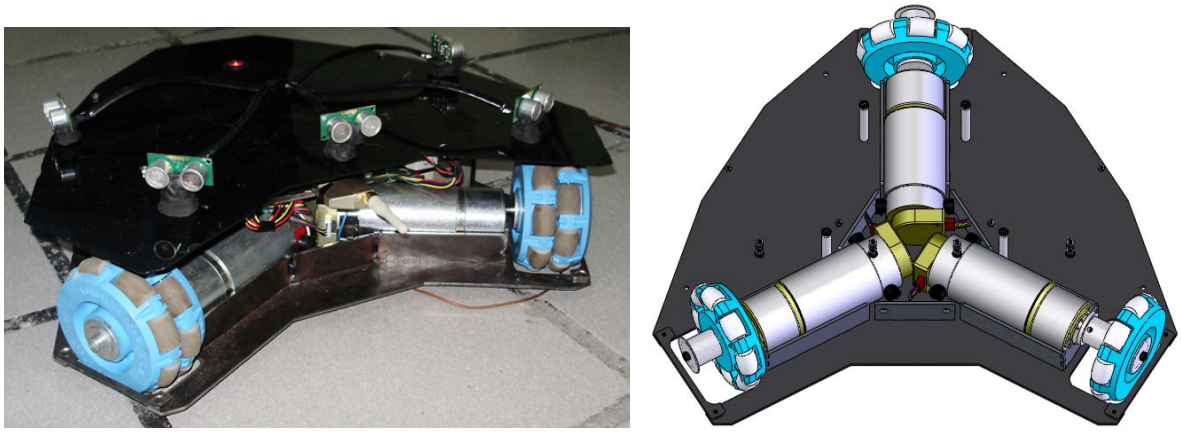


Figure 2.13: Three Swedish wheeled Omnidirectional Robot.
Source: [34].

Most common configuration of omnidirectional robots is shown in Figure 2.14, it is the Mobile Robot Summit-XL Steel by Robotnik.

Figure 2.14: Omnidirectional Mobile Robot Summit-XL Steel by Robotnik
Source: [13]

2.5.10 Differential Wheel Robot

Differential wheeled robots are special cases of some configurations, the key feature is that each side is controlled independently with separated actuators, some examples were shown in Figure 2.7 (Two wheeled), Figure 2.8 (Three Wheeled) and Figure 2.10 (four wheeled), additionally skid-steer systems used in some four and six wheeled robots are also differential and will be detailed in the next section.

2.6 Tracked Robots (TR)

Tracked Robots are a solution provided for the traction problem is soft terrain presented by wheeled robots, they are commonly an extension of differential driven robots configuration. These robots use treads or caterpillar tracks instead of wheels. Tracked robots have much larger ground contact patches, and this fact plays an important role when improving their maneuverability on loose surface in comparison to conventional wheeled robots. Nevertheless, because this large ground contact patch, changing the direction of the robot normally needs a skidding turn, and therefore a large portion of the track must slide against the surface.

Some examples of tracked robots were shown in Figure 1.3 center and right side.

2.6.1 Skid Steer Robots

Every skid steer robot is a differential driven robot, this locomotion system is commonly used on tracked vehicles such as tanks and bulldozers, but is also used on some four and six wheeled vehicles, as previously said, each side can be controlled independently, so the robot turns according to the difference between each side velocity. The actuators can usually go in reverse, this allow the robot turning over its own axle, without displacement [40].

Given the context of development for this research, Ground Tracked Skid-Steer Mobile Robots are attractive options for landmine search and detection. Next chapter will be focused on mathematical representations specially of these configurations.

Chapter 3

Mathematical Models

In this chapter, the development of mathematical description for some models of mobile robots is lead, previous chapter showed a review of robot structures and marked off the use of differential configurations for referred application. Additional math models are presented for reviewing analysis.

3.1 The Differential Driven Mobile Robot (DDMR) Model

Mathematical descriptions given in this section are based specially in approaches detailed in [41] and [42].

To get started, the raised references focus on wheeled configurations, so the initial assumption is that tracked vehicles are essentially wheeled configuration with caterpillar or tread transmissions, then the relationship is kinematic and linear.

It is important to notice in Figure 3.1 that there are three key points of analysis, The first one is a reference point, always fixed, representing the universe or work space origin. The second and third point belong to the robot, are respectively the geometric center, and the mass center, they are separated because on real model they rarely match, also taking into account that over the structure are located all control, perception and communication devices, modifying again the gravity or mass center. According to Figure 3.1 there are then defined three refrence frames:

- $\Sigma_0(X_0, Y_0)$ The inertial frame, a fixed or universal coordinated system representing the environment.
- $\Sigma_1(X_1, Y_1)$ The robot geometric center, conveniently located collinearly with the axles of the motors.

- $\Sigma_2(X_2, Y_2)$ The mass center in a different location from the geometric center for more realistic and flexible modeling.

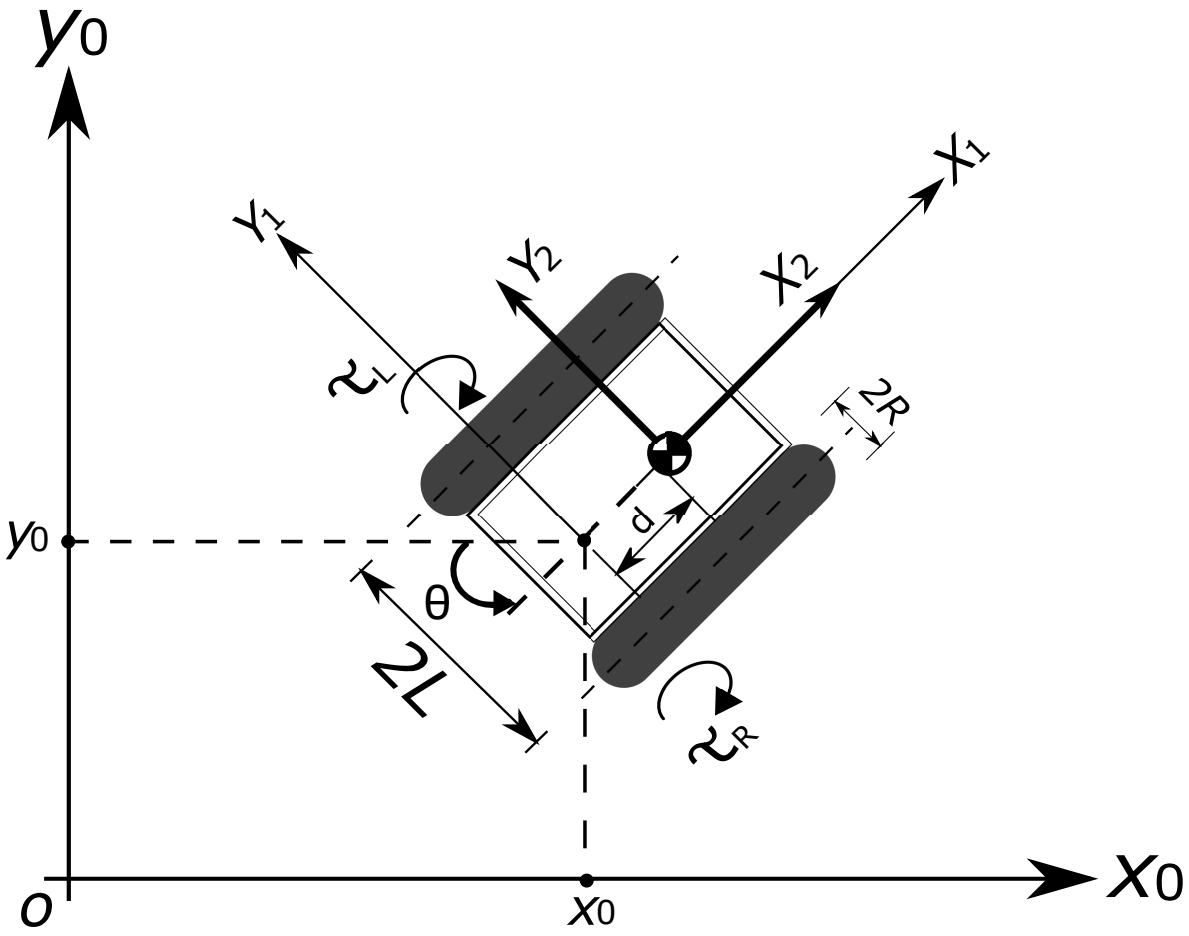


Figure 3.1: Reference frames for DDRM modelling

In order to design a control system considering all robot features the goal is to obtain a dynamic model taking into account the kinematic and non-holonomic constraints, in that sense, there are some parameters drawn in Figure 3.1 important to define:

- τ_R and τ_L: these are, respectively the right and left side torques applied to associated internal wheels.
- θ: this is the orientation of the robot qith respect to the inertial frame Σ_0
- L: this is the distance between the geometric center and the position of each internal tracked wheel.
- R this is the radius of the internal wheel, including the track thickness, this is the distance between the wheel center and the floor.

- d this is the distance beteen the geometric center and the mass center in the x direction of both Σ_1 and Σ_2
- x_0: this is the x coordinate of the robot geometric center position Σ_1 with respect to the inertial frame Σ_0.
- y_0: this is the y coordinate of the robot geometric center position Σ_1 with respect to the inertial frame Σ_0.

From the previous definition, it could be defined a dynamic configuration considering forces, accelerations and velocities, Figure 3.2 shows this approach that meets Newton Euler formulations.

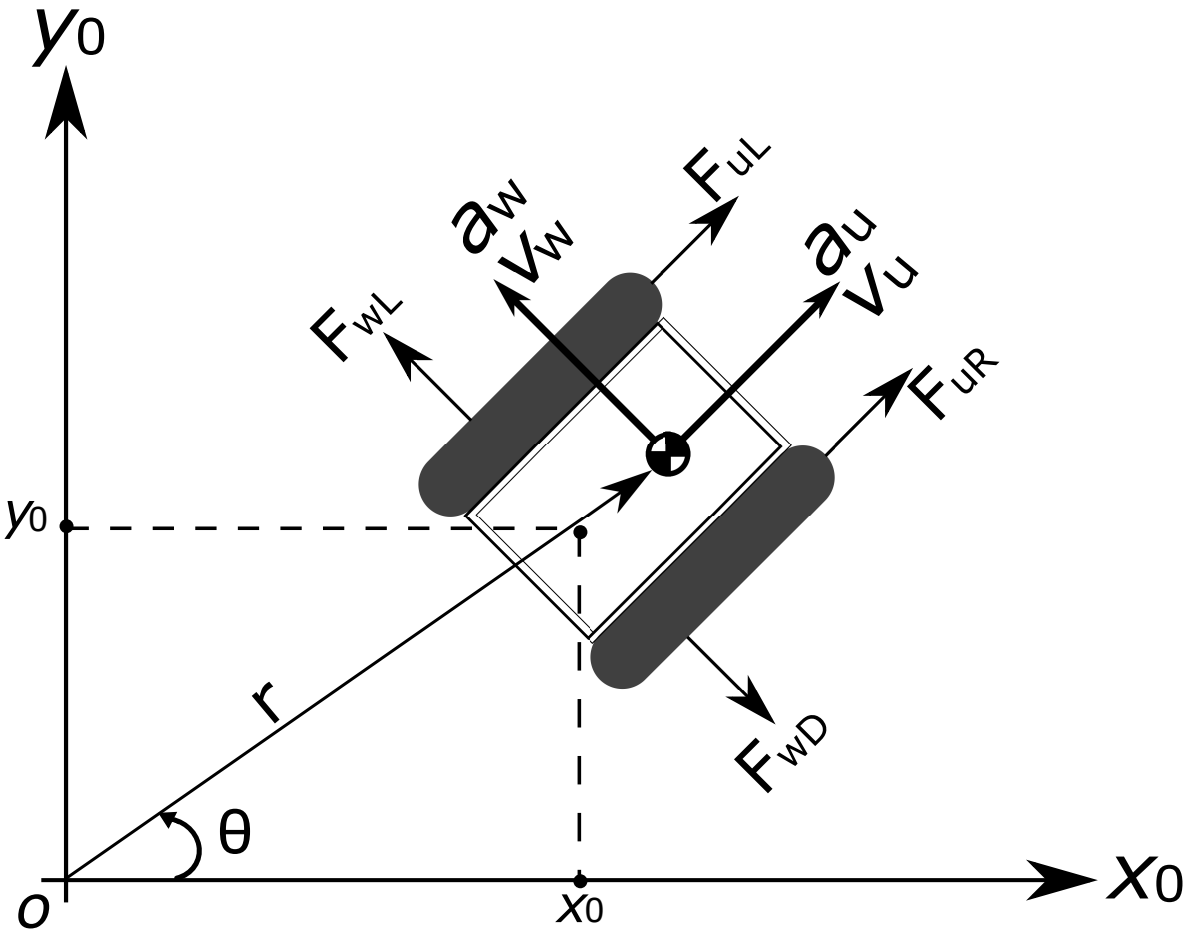


Figure 3.2: Dynamical definition of robot configuration

At this point it is important to note the position of the mass center, recall $\Sigma_1(x_1, y_1)$, and the vector r describing this position, also note that these are time dependent functions

$$\vec{r}(t) = r\left[\cos\theta(t)_x + \sin\theta(t)_y\right]$$

It can be conveniently converted to a complex plane where y represents the imaginary component, and rewrite the previous equation in polar form:

$$\vec{r} = r\left[\cos(\theta) + i\sin(\theta)\right]$$

and compactly

$$\vec{r} = re^{i\theta} \tag{3.1}$$

From Figure 3.2 is important to define the variables:

- F_{uL}: Radial force component provided by the left side actuator.
- F_{uR}: Radial force component provided by the right side actuator.
- F_{wL}: Tangential force component provided by the left side actuator.
- F_{wR}: Tangential force component provided by the right side actuator.
- a_u: Radial component of acceleration.
- a_w: Tangential component of acceleration.
- V_u: Radial component of velocity.
- V_w: Tangential component of velocity.

The relationship between torques $\mathcal{T}$ and forces F applied to each side of the robot is $\mathcal{T} = F \cdot R$, then:

$$F = \frac{\mathcal{T}}{R} \tag{3.2}$$

Differentiating Equation (3.1) in time, the velocities and accelerations are obtained:

$$\dot{\vec{r}} = \dot{r}e^{i\theta} + ir\dot{\theta}e^{i\theta} \tag{3.3}$$

$$\ddot{\vec{r}} = \ddot{r}e^{i\theta} + 2i\dot{r}\dot{\theta}e^{i\theta} + ir\ddot{\theta}e^{i\theta} - r\dot{\theta}^2e^{i\theta} \tag{3.4}$$

separating velocity and acceleration into radal and tangencial terms, we have:

$$\dot{\vec{r}} = [\dot{r}]e^{i\theta} + [r\dot{\theta}]e^{i\left(\theta+\frac{\pi}{2}\right)}$$

$$\ddot{\vec{r}} = [\ddot{r} - r\dot{\theta}^2]e^{i\theta} + [2\dot{r}\dot{\theta} + r\ddot{\theta}]e^{i\left(\theta+\frac{\pi}{2}\right)}$$

and then, variables in Figure 3.2 follow:

$$V_u = \dot{r} \tag{3.5}$$
$$V_w = r\dot{\theta} \tag{3.6}$$
$$a_u = \ddot{r} - r\dot{\theta}^2 \tag{3.7}$$
$$a_w = 2\dot{r}\dot{\theta} + r\ddot{\theta} \tag{3.8}$$

The movement equations are:

$$Ma_u = F_{uR} + F_{uL} \tag{3.9}$$
$$Ma_w = F_{wL} - F_{wR} \tag{3.10}$$
$$I\ddot{\theta} = L(F_{uR} - F_{uL}) + d(F_{wR} - F_{wL}) \tag{3.11}$$

here, M is the total mass of the robot and I the inertia respect to the center of mass, and from Eqs. (3.5)-(3.8) we have:

$$a_u = \dot{V}_u - V_w\dot{\theta} \tag{3.12}$$
$$a_w = \dot{V}_w + V_u\dot{\theta} \tag{3.13}$$

and then

$$M(\dot{V}_u - V_w\dot{\theta}) = F_{uR} + F_{uL} \tag{3.14}$$
$$M(\dot{V}_w + V_u\dot{\theta}) = F_{wR} - F_{wL} \tag{3.15}$$
$$I\ddot{\theta} = L(F_{uR} - F_{uL}) - d(F_{wR} - F_{wL}) \tag{3.16}$$

$$\dot{V}_u = \frac{F_{uR} + F_{uL}}{M} + V_w\dot{\theta} \tag{3.17}$$
$$\dot{V}_w = \frac{F_{wR} - F_{wL}}{M} - V_u\dot{\theta} \tag{3.18}$$
$$\ddot{\theta} = \frac{L}{I}(F_{uR} - F_{uL}) + \frac{d}{I}(F_{wR} - F_{wL}) \tag{3.19}$$

The velocity of mass center Σ_2, $\dot{x}_2$ and $\dot{y}_2$, respect to the inertial frame Σ_0, are given by

$$\dot{x}_2 = V_u\cos\theta - V_w\sin\theta \tag{3.20}$$
$$\dot{y}_2 = V_u\sin\theta + V_w\cos\theta \tag{3.21}$$

and respect to the frame Σ_1:

$$\begin{aligned} x_2 &= x_0 + d\cos\theta \\ \dot{x}_2 &= \dot{x}_0 - d\dot{\theta}\sin\theta \end{aligned} \tag{3.22}$$
$$\begin{aligned} y_2 &= y_0 + d\sin\theta \\ \dot{y}_2 &= \dot{y}_0 + d\dot{\theta}\sin\theta \end{aligned} \tag{3.23}$$

Matching Eqs. (3.20) with (3.22) and (3.21) with (3.23), and taking the sum of their squares, we have:

$$\dot{x}_1^2 + (d\dot{\theta}\sin\theta)^2 + y_1^2 + (d\dot{\theta}\cos\theta)^2 = V_u^2 + V_w^2 \tag{3.24}$$

considering that robot has no lateral slipping ($\dot{y}_1 = 0$) and that $\dot{x}_1 = V_u$ given their coaxiality, then:

$$\begin{aligned} V_u^2 + (d\dot{\theta})^2(\sin^2\theta + \cos^2\theta) &= V_u^2 + V_w^2 \\ d\dot{\theta} &= V_w \end{aligned} \tag{3.25}$$

Finally the set of dynamical equations describing the DDMR are given by Eqs. (3.26)-(3.29) similar to [41]

$$\dot{V}_u = d\dot{\theta}^2 + \frac{1}{MR}(\mathcal{T}_R + \mathcal{T}_L) \tag{3.26}$$

$$\ddot{\theta} = -\frac{Md}{I + Md^2}V_u\dot{\theta} + \frac{L}{(I + Md^2)R}(\mathcal{T}_R - \mathcal{T}_L) \tag{3.27}$$

$$\dot{x} = V_u\cos(\theta) - d\dot{\theta}\sin\theta \tag{3.28}$$

$$\dot{y} = V_u\sin(\theta) + d\dot{\theta}\cos\theta \tag{3.29}$$

3.2 Three Wheeled Differential Robot (IVWAN) Model

The three wheeled robot described and shown in Figure 2.8 has the model formulation starting with the free body diagram shown in Figure 3.3.

The subscripts used have the following meaning: f stands for front wheels and c stands for caster wheel, while the subscripts r and l stand for right and left, respectively. Finally x and y stand for the corresponding direction in the inertial frame.

u, $\bar{u}$ and ω are the radial, tangential and angular velocities, respectively, B represents the center of the axis connecting both traction wheels; G represents the vehicle's center of mass and for simplicity, it is considered as the point to control in position (x, y) and orientation (φ), b is the distance between center of mass and the traction axle, c is the distance between front and caster axles, and d is the distance between the traction wheels.

Resultant forces and momentum in the structure using Newton laws can be expressed by:

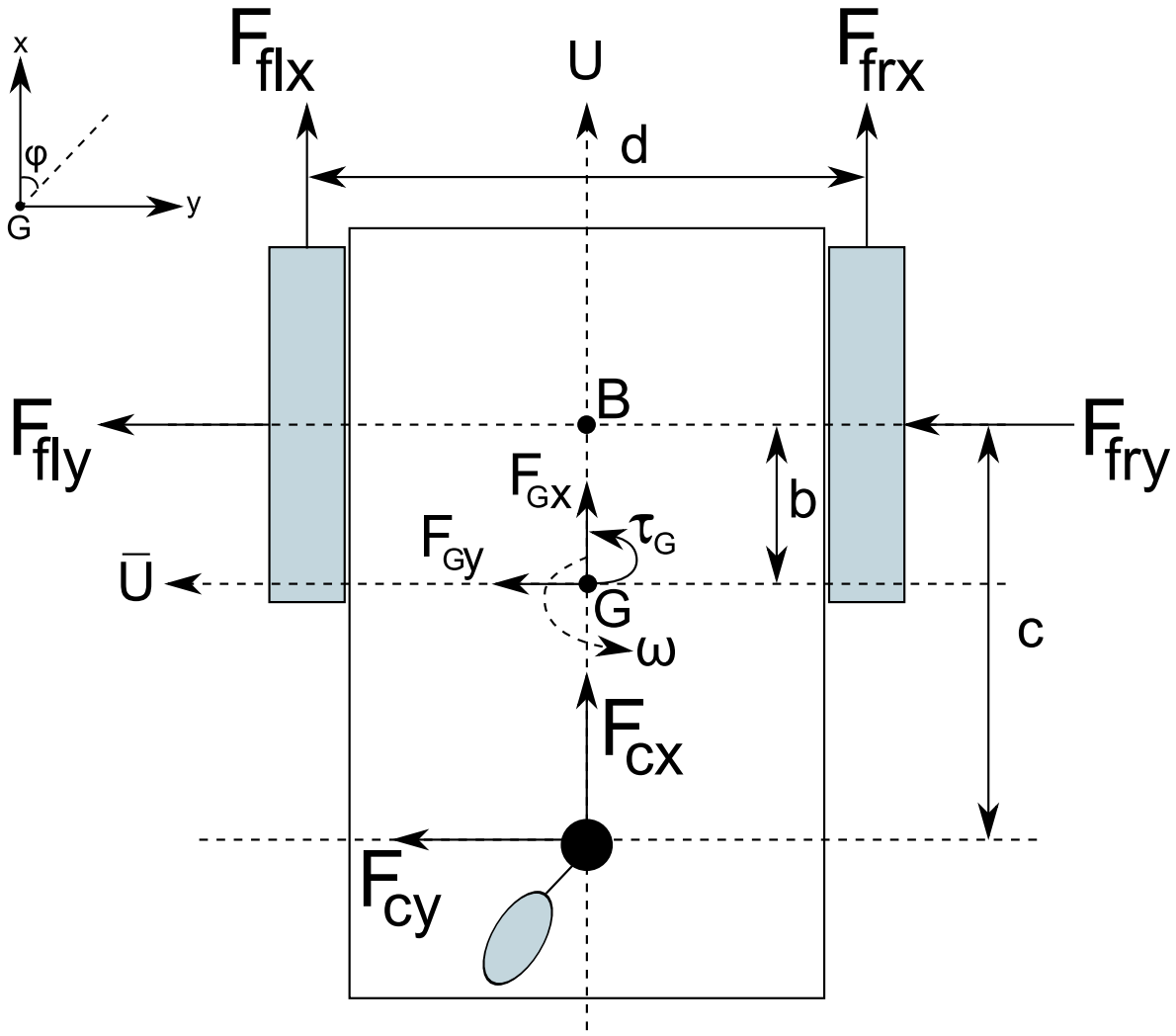


Figure 3.3: Free body diagram for the Intelligent Vehicle With Autonomous Navigation (IVWAN).

Source: [34].

$$m(\dot{u} - \bar{u}\omega) = F_{frx} + F_{flx} + F_{cx} + F_{Gx} \tag{3.30}$$

$$m(\dot{\bar{u}} - u\omega) = F_{fry} + F_{fly} + F_{cy} + F_{Gy} \tag{3.31}$$

$$I\dot{\omega} = \frac{d}{2}(F_{frx} - F_{flx}) - b(F_{fry} + F_{fly}) + (c - b)F_{cy} + \tau_G \tag{3.32}$$

where m is the vehicle's total mass and I is the moment of inertia around G.

velocity $\bar{u}$ can be reasonable neglected assuming that the wheels do not slip in the robot motion. The linear velocity u is the mean of both sides velocity, conditioned by the traction wheel radius r, then:

$$u = \frac{1}{2}\left[r(\omega_r + \omega_l) + (u_r + u_l)\right] \tag{3.33}$$

the angular velocity ω is conditioned by the difference between velocities, then:

$$\omega = \frac{1}{d}\left[r(\omega_r - \omega_l) + (u_r - u_l)\right] \tag{3.34}$$

Similar to the previous model, in Equations (3.20) and (3.21), the rectangular components for the mass center G are given by:

$$\dot{x} = u\cos(\varphi) - b\omega\sin(\varphi) \tag{3.35}$$
$$\dot{y} = u\sin(\varphi) + b\omega\cos(\varphi) \tag{3.36}$$

and consequently $\varphi = \omega$.

As a differential robot, each wheel is driven by a DC motor which can be modelled neglecting inductive voltage as follows:

$$\tau_m = \frac{k_a}{R_a}(E_m - k_b\omega_m)$$

E_m is the motor voltage k_a and k_b are the motor's torque and electromotive force constants, respectively; R_a is the motor's electric resistance.

Taking into account two independent motors, Equations describing the wheel-motor system can be simply written as:

$$I_e\dot{\omega}_r + D_e\omega_r = \tau_r - F_{frx}\hat{r} \tag{3.37}$$
$$I_e\dot{\omega}_l + D_e\omega_l = \tau_l - F_{flx}\hat{r} \tag{3.38}$$

τ_r and τ_l are right and left side motors, respectively, I_e is the moment of inertia of the wheel-motor system, D_e is its coefficient of viscous friction, and $\hat{r}$ is the nominal radius of the traction wheel tires.

Combining and rearranging all previous Equations, the complete robotic system can be represented in the matrix form:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\\ \dot{u}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}u\cos(\varphi) - b\omega\sin(\varphi)\\ u\sin(\varphi) + b\omega\cos(\varphi)\\ \omega\\ \frac{a_3}{a_1}\hat{r}r\omega^2 - \frac{2a_4}{a_1}u\\ -\frac{2a_3}{a_2}\hat{r}ru\omega - \frac{a_4}{a_2}d^2\omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{2r}{a_1} & 0\\ 0 & \frac{2rd}{a_2}\end{bmatrix}\begin{bmatrix}E_u\\ E_\omega\end{bmatrix} \tag{3.39}$$

with

$$E_u = \frac{E_r + E_l}{2}, \quad E_w = \frac{E_r - E_l}{2}, \quad a_1 = \frac{R_a}{k_a}(m\hat{r}r + 2I_e),$$
$$a_2 = \frac{R_a}{k_a}(I_e d^2 + 2\hat{r}r(I + mb^2)), \quad a_3 = \frac{R_a}{k_a}mb, \quad a_4 = \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + D_e\right)$$

Chapter 4

Low Level Control

In the context of this document, *low level control* refers to all techniques, methods, approaches, and strategies for dynamic variables stabilization and reference tracking. In robotics and other engineering areas the specific tasks are attained by **automatic control**, it takes care of regulation, and dynamic tracking of variables like robot position, orientation and speed, through manipulation of forces, torques, and motors angular positions and velocities.

The reaching of variable references is important in navigation tasks, but this chapter covers only the task of reaching a certain position and/or orientation. This **pose** could be part of a sequence of actions given for follow a path or interactive behaviour in the environment, but those are high level tasks that will be covered in the next chapter

4.1 Control Basics

Although robot models are complex expressions, the study of linear system representations and linear control techniques is fundamental in any robot research.

4.1.1 Linear systems

It is said that a system or model is linear if its behaviour can be defined by the equation:

$$\frac{d^n y}{dt^n} + a_1(t)\frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n(t)y = b_1(t)\frac{d^{n-1} u}{dt^{n-1}} + \cdots + b_n(t)u \tag{4.1}$$

for the input $u(t)$ and the output $y(t)$ and the order is n, this is a Single Input Single Output (SISO) system.

A Multiple Input Multiple Output (MIMO) system is linear if there is a set of equations $f_i(y_i, u_j), i = 1, \cdots, p \quad j = 1, \cdots, q$ in the form of Equation (4.1) for q inputs and p outputs that define the system behaviour.

if $a_k(t)$'s and $b_k(t)$'s are constant, then using Laplace transform, the SISO system can be expressed as a transfer function, defined as the relationship between the output $y(t)$ and the input $u(t)$, then:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_1 s^{n-1} + \cdots + b_{n-1}s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n} \tag{4.2}$$

Similarly, for a MIMO system, a transfer matrix can be defined as follows:

$$\begin{bmatrix} Y_1(s) \\ \vdots \\ Y_p(s) \end{bmatrix} = \begin{bmatrix} H_{11}(s) & \cdots & H_{1q}(s) \\ \vdots & \ddots & \vdots \\ H_{p1}(s) & \cdots & H_{pq}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ \vdots \\ U_q(s) \end{bmatrix} \tag{4.3}$$

where $H_{ij}(s)$ is the transfer function for the input j to the output i and has the form of Equation (4.2). The order of a MIMO system is $n = n_1 + n_2 + \cdots + n_p$ where n_i is the order of the linear differential equation f_i.

Modern control techniques are based on vector analysis and then a matrix representation of systems is relevant in any control systems research. A system could be represented as a set of first order differential equations:

$$\dot{x} = Ax + Bu \tag{4.4}$$
$$y = Cx + Du \tag{4.5}$$

it is called the state space of a system of order n with:

$$\begin{aligned} x &\in \mathbb{R}^{n\times 1}, \\ A &\in \mathbb{R}^{n\times n}, \\ B &\in \mathbb{R}^{n\times q}, \\ u &\in \mathbb{R}^{q\times 1}, \\ y &\in \mathbb{R}^{p\times 1}, \\ C &\in \mathbb{R}^{p\times n}, \\ D &\in \mathbb{R}^{p\times q} \end{aligned}$$

Equation (4.4) is the state equation and Equation (4.5) is the output equation. Notice that this is a MIMO system, the SISO case happens when $p = 1$ and $q = 1$.

For the SISO case, a state space representation could be obtained from the transfer function (Equation (4.2)), two forms of interest are [43]:

1. **Canonical Controller form**

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u$$

2. **Canonical Observer form**

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & 1 & \cdots & 0 \\ -a_2 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 \\ -a_n & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u$$

4.1.2 Nonlinear Systems

Based on space state representation, a nonlinear system can be defined by a set of nonlinear differential equations described by:

$$\dot{x} = \mathbf{f}(x, u) \tag{4.6}$$

$$y = \mathbf{h}(x, u) \tag{4.7}$$

where $\mathbf{f}$ and $\mathbf{h}$ are vector fields.

Most real systems are nonlinear, but a substantial set of them have also an interesting property: they are *affine to the control input*, so they can be expressed in the form:

$$\dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u \tag{4.8}$$

$$y = \mathbf{h}(x) \tag{4.9}$$

where $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ are also vector fields.

4.1.3 Basic Linear control methods

Control methods for linear systems are based in both transfer function and state space representations, these are described below.

Transfer function controllers

For transfer function representation, the controller is defined as a dynamic system whose input is the error e, defined as the difference between the system output y and a setpoint or reference r, that is:

$$e = r - y \tag{4.10}$$

The controller output is the control signal u, then the controller is represented also by a transfer function in the form:

$$C(s) = \frac{U(s)}{E(s)} = \frac{l_1 s^{n-1} + \cdots + l_{n-1}s + l_n}{s^n + p_1 s^{n-1} + \cdots + p_{n-1}s + p_n} \tag{4.11}$$

given that configuration, the control system is a feedback scheme which needs the measurement of output y as shown in Figure 4.1.

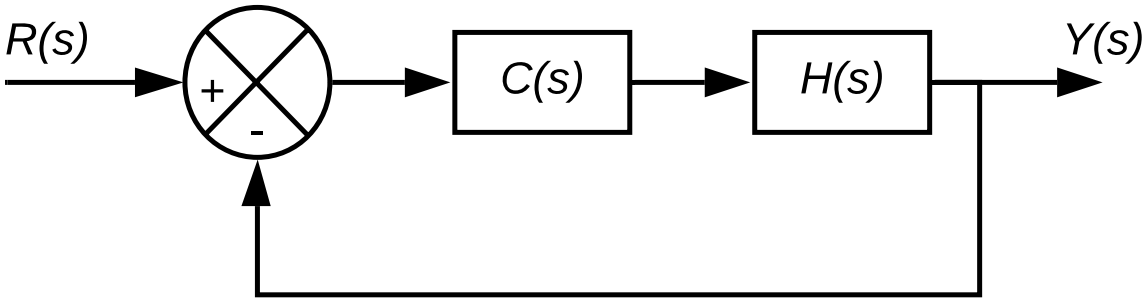


Figure 4.1: Block diagram of feedback transfer function control

Using transfer function representation the system convolutions are simply transfer function products, and from the Figure 4.1 scheme the relationship output-reference is given by:

$$H_{CL}(s) = \frac{Y(s)}{R(s)} = \frac{C(s)H(s)}{1 + C(s)H(s)} \tag{4.12}$$

called the closed loop transfer function. the controller is designed in such a way that H_{CL} is stable.

Although the controller stabilizes the system, sometimes it does not imply that final value will be r.

Known as steady state value, it can be computed using the final value theorem:

$$y_{ss} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} sH_{CL}(s)R(s) \tag{4.13}$$

if $R(s)$ is the unit step function, then $0 \leq y_{ss} \leq 1$, and a direct loop gain k_G can be included as in Figure 4.2, and defined such that:

$$y_{ss} = \lim_{t\to\infty} y(t) = r \quad \rightarrow \quad k_G = \frac{1}{y_{ss}} \tag{4.14}$$

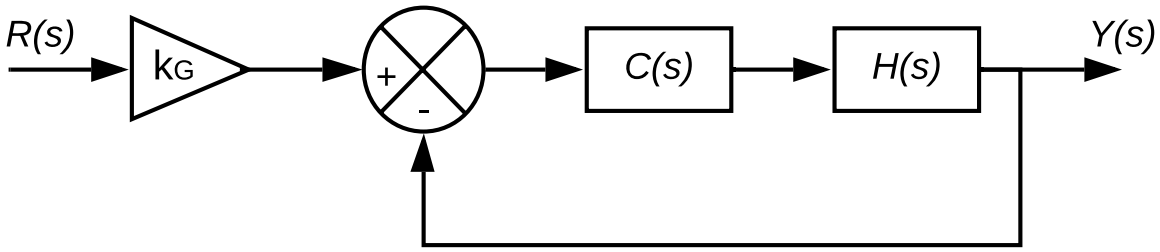


Figure 4.2: Feedback controller with direct loop gain

Another approach, more robust than direct loop gain, is the integral action, if a signal control is defined for error integral:

$$u_i = k_i \int e(t)dt$$

then steady state error will be zero.

As $U_i(s) = k_i E(s)/s$ then, adding the integral action to the controller gives:

$$C(s) = \frac{1}{s}C'(s) \tag{4.15}$$

with $C'(s)$ in the form of Equation (4.11), for a control scheme similar to Figure 4.1 there could be defined:

$$H'(s) = \frac{1}{s}H(s) \tag{4.16}$$

Then, design a controller $C'(s)$ for an $H'(s)$ such that the corresponding H_{CL} stable:

$$H_{CL}(s) = \frac{C'(s)H'(s)}{1 + C'(s)H'(s)} \tag{4.17}$$

some special cases for $C(s)$ widely used on industrial, commercial an general purpose systems are PID controllers.

PID Controller

A Proportional Integral Derivative (PID) controller is a commonly used control structure given its simplicity and effectiveness, the PID controller is based on the actions:

- **Proportional (P)**: for minimizing the error: $u_p = k_p e(t)$
- **Integral (I)**: for removing steady state error: $u_i = k_i \int e(t)dt$
- **Derivative (D)**: for quick response to changes in the system or in reference:

$$u_d = k_d \frac{de(t)}{dt}$$

then the complete control signal is:

$$u = u_p + u_i + u_d = k_p e(t) + k_i \int e(t)dt + k_d \frac{de(t)}{dt} \tag{4.18}$$

and its transfer function is:

$$PID(s) = \frac{k_d s^2 + k_p s + k_i}{s} \tag{4.19}$$

State Space Controllers

The first approach of state space control methods is state regulation, a **state regulator** is structure for state stabilization, it reaches:

$$\lim_{t \to \infty} \dot{x} = \vec{0} \tag{4.20}$$

then the control signal is defined over the state vector as $u = -Kx$, the block diagram is shown in figure

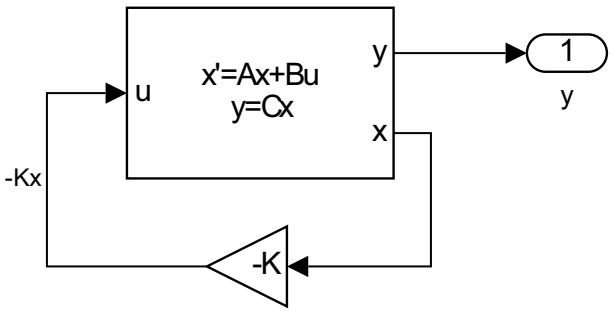


Figure 4.3: State regulator scheme

That scheme generates a closed loop system with poles as roots of the polynomial:

$$\det(sI - A + BK) = 0 \tag{4.21}$$

the controller is designed for K such that this characteristic polynomial has stable poles.

Similar than in transfer function approaches, state regulator stabilizes the system, but it does not imply that the output final value y_{ss} will be as desired, the approaches for reaching desired values are known as **reference tracking** tasks.

The first approach is the **state feedback controller with direct loop gain** shown in Figure 4.4, where the setpoint r is multiplied by a gain k_G as in transfer function case, for unit step response, it can be defined as:

$$y_{ss} = \lim_{s \to 0} C(sI - A + BK)^{-1}B \quad \rightarrow \quad k_G = \frac{1}{y_{ss}} \tag{4.22}$$

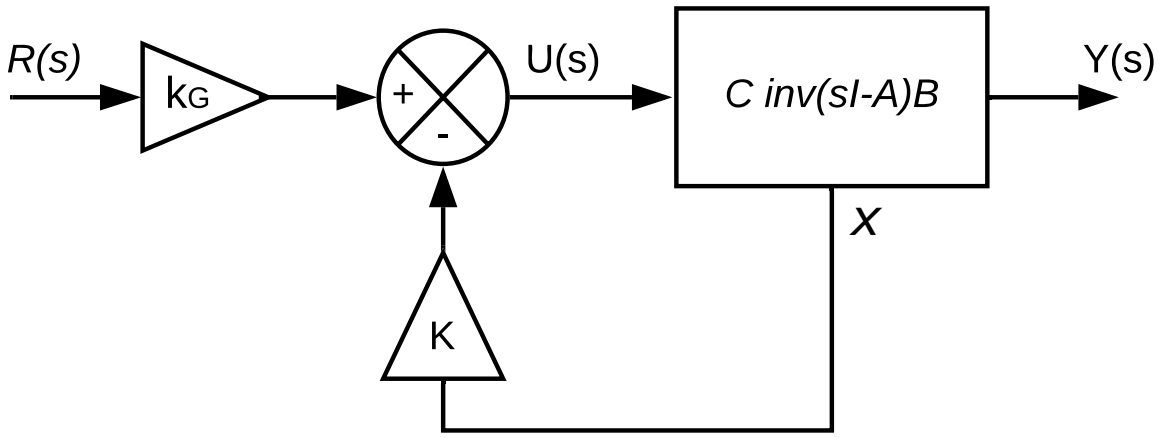


Figure 4.4: State feedback controller with direct loop gain

The **state feedback controller with integral action** is the more robust approach for state space control with reference tracking, starting from the error definition in Equation (4.10), the immediate integral is:

$$e_i = \int [r(t) - y(t)]dt \tag{4.23}$$

and then

$$\dot{e}_i = r - y \tag{4.24}$$

$$\dot{e}_i = r - Cx \tag{4.25}$$

there we can define an augmented system:

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{e}_i\end{bmatrix}}_{\dot{x}_a} = \underbrace{\begin{bmatrix}A & 0\\ -C & 1\end{bmatrix}}_{A_a}\underbrace{\begin{bmatrix}x\\ e_i\end{bmatrix}}_{x_a} + \underbrace{\begin{bmatrix}B\\ 0\end{bmatrix}}_{B_a} u + \underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{B_a} r \tag{4.26}$$

$$y = \underbrace{\begin{bmatrix}C & 0\end{bmatrix}}_{C_a}\underbrace{\begin{bmatrix}x\\ e_i\end{bmatrix}}_{x_a} \tag{4.27}$$

then the control signal is defined as

$$u = -Kx + k_i e_i \tag{4.28}$$

$$u = -\underbrace{\begin{bmatrix}K & -k_i\end{bmatrix}}_{K_a}\underbrace{\begin{bmatrix}x\\ e_i\end{bmatrix}}_{x_a} \tag{4.29}$$

the controller is designed for K_a such that the characteristic polynomial:

$$\det(sI - A_a + B_a K_a) = 0 \tag{4.30}$$

has stable poles. Figure 4.5 shows the block diagram for this controller

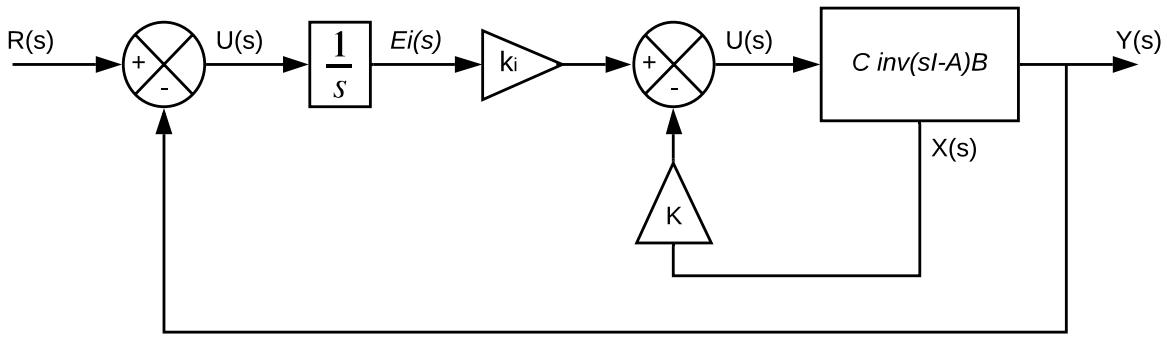


Figure 4.5: State feedback controller with integral action

4.1.4 Basic Nonlinear Control Methods

A nonlinear system could be approximated to a linear representation around an operation point (x_0, u_0) using first Taylor series elements, this allows to use linear control methods in nonlinear systems, this method is called **Approximated Linearization**.

Taylor series expand a function f, the representation is as follows:

$$f(t) = \sum_{n=0}^{\infty} \frac{f^{(n)}(t_0)}{n!}(t - t_0)^n \tag{4.31}$$

$$f(t) = f(t_0) + \frac{f'(t_0)}{1!}(t - t_0) + \frac{f''(t_0)}{2!}(t - t_0)^2 + \frac{f'''(t_0)}{1!}(t - t_0)^3 + \cdot \tag{4.32}$$

then, a linear approximation could be given by

$$f(t) \approx f(t_0) + \frac{f'(t_0)}{1!}(t - t_0) \tag{4.33}$$

$$f(t) - f(t_0) = f'(t_0)(t - t_0) \tag{4.34}$$

this is a good approximation for $\Delta f(t) = f(t) - f(t_0)$ if $\Delta t = t - t_0$ is small. linearization of $f(t)$ around t_0

For linear representation of systems, the differential equation is linear with respect to the state variables $x_i \quad i = 1, 2, \cdots, n$. then for the state Equation (4.6), and output Equation (4.7) its linear approximation is:

$$\Delta \dot{x} = \sum_{i=1}^{n} \left.\frac{\partial f}{\partial x_i}\right|_{x_0,u_0} \Delta x_i + \sum_{j=1}^{q} \left.\frac{\partial f}{\partial u_j}\right|_{x_0,u_0} \Delta u_j \tag{4.35}$$

$$\Delta y = \sum_{i=1}^{n} \left.\frac{\partial h}{\partial x_i}\right|_{x_0,u_0} \Delta x_i + \sum_{j=1}^{q} \left.\frac{\partial h}{\partial u_j}\right|_{x_0,u_0} \Delta u_j \tag{4.36}$$

Then the linear state space is given by:

$$A = \nabla \mathbf{f}(x_0, u_0), \quad \nabla = \left[\frac{\partial}{\partial \mathbf{x}}\right] \tag{4.37}$$

$$B = \nabla \mathbf{f}(x_0, u_0), \quad \nabla = \left[\frac{\partial}{\partial \mathbf{u}}\right] \tag{4.38}$$

$$C = \nabla \mathbf{h}(x_0, u_0), \quad \nabla = \left[\frac{\partial}{\partial \mathbf{x}}\right] \tag{4.39}$$

$$D = \nabla \mathbf{h}(x_0, u_0), \quad \nabla = \left[\frac{\partial}{\partial \mathbf{u}}\right] \tag{4.40}$$

in the Equations (4.4) and (4.5). A is the jacobian of the system.

Another important nonlinear method was proposed by Isidori et. all in the eighties [44], it is the **exact feedback linearization** first presented for SISO systems. It consists of finding a control law that transforms nonlinear dynamics on a complete or partial way into linear dynamics, it is only possible if the system is control input affine (represented as in Equations (4.6) and (4.7)).

if all nonlinearities are concentrated in a

$$\dot{x}_r = f(x) + g(x)u = \nu \tag{4.41}$$

then the system becomes linear and the control law has the following form [45]:

$$u = \alpha(x) + \beta(x)\nu \tag{4.42}$$

where $\alpha(x) = -f(x)/g(x)$ and $\beta(x) = 1/g(x)$

Depending on existence of zero dynamics in the system, there may be two cases:

Input-State Linearisation

The system {(4.6),(4.7)} is n order. If there exist a transformation of state variables, such that only has an n order differential equation then the it is input-state linearizable, and the system:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
&\vdots \\
\dot{x}_n &= f(x_1, x_2, \cdots, x_n) + g(x_1, x_2, \cdots, x_n)u
\end{aligned}$$

becomes linear if

$$\nu = f(x_1, x_2, \cdots, x_n) + g(x_1, x_2, \cdots, x_n)u \tag{4.43}$$

and it has a matrix representation:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \nu \tag{4.44}$$

it could be controlled using a simple linear control law like in Figures 4.3, 4.4 and 4.5, for $\nu = -Kx \cdots$, an then get u from (4.43) as: [45]

$$u = \frac{\nu - f(x_1, x_2, \cdots, x_n)}{g(x_1, x_2, \cdots, x_n)} \tag{4.45}$$

Input-Output Linearization

If a complete representation cannot be reached since, it is said that system has zero dynamics. The method could be carried out choosing a new variable set as the output

$$z = y, \quad \dot{z} = \dot{y}, \quad \dots$$

Differentiate it until u appears for the first time. The $m < n$ derivatives carried out define the relative order of the system, and similar to previous case:

$$\begin{aligned}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
&\vdots \\
\dot{z}_m &= f(z_1, z_2, \cdots, z_m) + f(z_1, z_2, \cdots, z_m)u
\end{aligned}$$

and solving the same control law $\nu = -Kz \cdots$, u can be gotten from [45]:

$$\nu = f(z_1, z_2, \cdots, z_m) + g(z_1, z_2, \cdots, z_m)u$$

$$u = \frac{\nu - f(z_1, z_2, \cdots, z_m)}{g(z_1, z_2, \cdots, z_m)}$$

4.2 Multivariable State Feedback Linearization

For multiple-input multiple-output (MIMO) system, the Isidori's method could be extended using geometric control concepts [10].

Let a MIMO system given by the following equations [46]

$$\dot{x} = \mathbf{f}(x) + \sum_{i=1}^{m} g_i(x)u \tag{4.46}$$

$$y_1 = h_1(x) \tag{4.47}$$

$$\vdots$$

$$y_m = h_m(x) \tag{4.48}$$

or in a compact way

$$\dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u \tag{4.49}$$

where

$$\mathbf{g}(x) = [g_1(x), g_2(x), ..., g_m(x)] \tag{4.50}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{4.51}$$

The system has a relative grade $\{r_1 \cdots r_m\}$ for each output if:

$$\mathcal{L}_{g_j}\mathcal{L}_{\mathbf{f}}^{k} h_i(x) = 0 \text{ for all } 1 \leq j \leq m,\ k < r_i - 1,\ 1 \leq i \leq m$$

where $\mathcal{L}_{\mathbf{f}}h$ is the Lie derivative of the scalar function h in the direction of the vectorial field $\mathbf{f}$, it is defined as $\nabla h \cdot \mathbf{f}$. The k-th Lie derivative is then recursively defined as $\mathcal{L}_{\mathbf{f}}^{0}h = \mathbf{f}$ and $\mathcal{L}_{\mathbf{f}}^{k}h = \mathcal{L}_{\mathbf{f}}(\mathcal{L}_{\mathbf{f}}^{k-1}h)$

The expression $\mathcal{L}_{\mathbf{g}}\mathcal{L}_{\mathbf{f}}^{k}h = \nabla(\mathcal{L}_{\mathbf{f}}h) \cdot \mathbf{g}$ is called Lie derivative of the scalar function h respect to vectorial fields $\mathbf{f}$ and $\mathbf{g}$

Then the decoupling matrix can be defined [47]:

$$A(x) = \begin{bmatrix} \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_1-1}h_1(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_1-1}h_1(x) \\ \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_2-1}h_2(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_2-1}h_2(x) \\ \vdots & \vdots & \vdots \\ \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_m-1}h_m(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_m-1}h_m(x) \end{bmatrix} \tag{4.52}$$

r_i is the relative grade of the i-th output $y_i(t)$, that is, the number of derivatives required for the first emergence of at least one u term [48].

This matrix is used to compute the control signals vector for the system defined from (4.46)-(4.48) as:

$$y_i^{(r_i)} = \mathcal{L}_{\mathbf{f}}^{r_i} h_i + \sum_{j=1}^{m} \mathcal{L}_{g_j} \mathcal{L}_{\mathbf{f}}^{r_i - 1} h_i u_j \tag{4.53}$$

$$\begin{bmatrix} y_1^{(r_1)} \\ y_2^{(r_2)} \\ \vdots \\ y_m^{(r_m)} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_{\mathbf{f}}^{r_1} h_1(x) \\ \mathcal{L}_{\mathbf{f}}^{r_2} h_2(x) \\ \vdots \\ \mathcal{L}_{\mathbf{f}}^{r_m} h_m(x) \end{bmatrix} + A(x) \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{4.54}$$

Leading m equations in the form:

$$y_i^{(r_i)} = z_i^{(r_i)} = \nu_i \tag{4.55}$$

to compute the signal control vector as follows:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} = A^{-1}(x) \begin{bmatrix} \nu_1 - \mathcal{L}_{\mathbf{f}}^{r_1} h_1(x) \\ \nu_2 - \mathcal{L}_{\mathbf{f}}^{r_2} h_2(x) \\ \vdots \\ \nu_m - \mathcal{L}_{\mathbf{f}}^{r_m} h_m(x) \end{bmatrix} \tag{4.56}$$

provided that $A(x)$ is not singular.

Chapter 5

High Level Control

Previous chapter approached techniques in low level control, those that carry out specific tasks for reaching and tracking variables that define the system dynamic behaviour, considered internal or low level variables, it depends on the robot configuration or its electromechanical structure. In the other hand, *High Level Control* are a set of techniques for provide autonomy to the robot, they define high level tasks, such as paths or routes for going from one position to another, or reaction behaviour to any situation in which the agent could be involved.

High level control is normally approached as a **planning** problem; in mobile robotics an important concept is **exploration**, this is a problem or technique of reading or measure the information of the environment and reconstruct it as a data structure.

Autonomous Robots may accept generic descriptions of tasks without many details and execute those tasks without further human intervention. The input descriptions only specify what the robots must do, rather than how to do it. The above implies the use of planning algorithms for determining each step or set of detailed simple tasks robot must do for reach the main generic task.

Robotic applications in which robots are designed for moving across a given work space, such as outdoor exploration or transportation, it is expected from an autonomous robot the ability to plan its own motions across the given environment, avoiding any object or obstacle in there, in literature this problem is stated as "motion planning" [2].

Motion planning is one of the most important tasks that autonomous robots must carry out. It involves the definition of a navigation path across the environment $\mathcal{W}$ [49] avoiding any region $\mathcal{B}_i$ defined as an obstacle, the defined path τ goes from an initial position $\mathbf{q_{init}}$ to a goal position $\mathbf{q_{goal}}$.

There exist a large number of methods proposed for solving the path planning problem, these methods are based essentially in three main approaches: roadmaps, cell decomposition and potential fields [1].

Coming up next, some popular methods are described:

5.1 Roadmap

A **roadmap** is a network of one-dimensional curves (lines) connecting the free-obstacle space, this network is used as a set of standardized or simple paths, and resulting path is chosen as the one connecting $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ with minimum length. The roadmap building begins with the allocation of points in the free-obstacle space, and then find a connected graph using a simple technique (simple or local planner). Figure 5.1 shows a roadmap in the defined environment where black regions are general obstacles. For more details on roadmap building see [50][51][52] and [53].

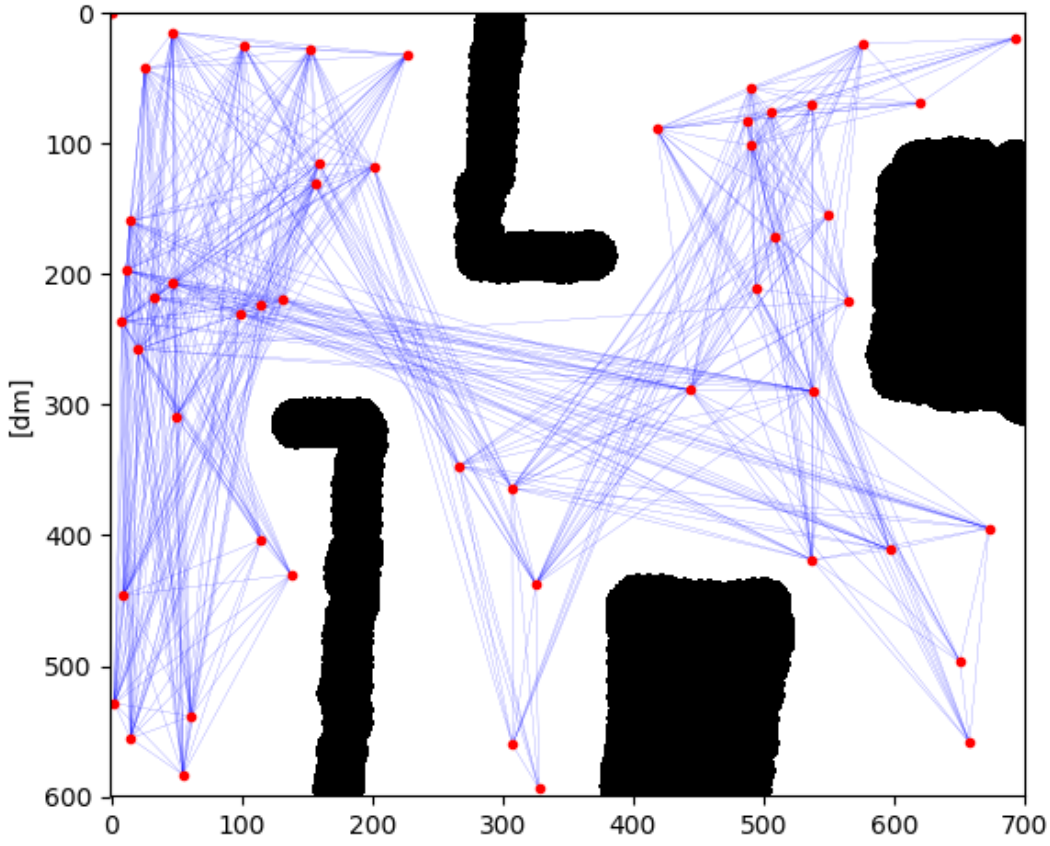
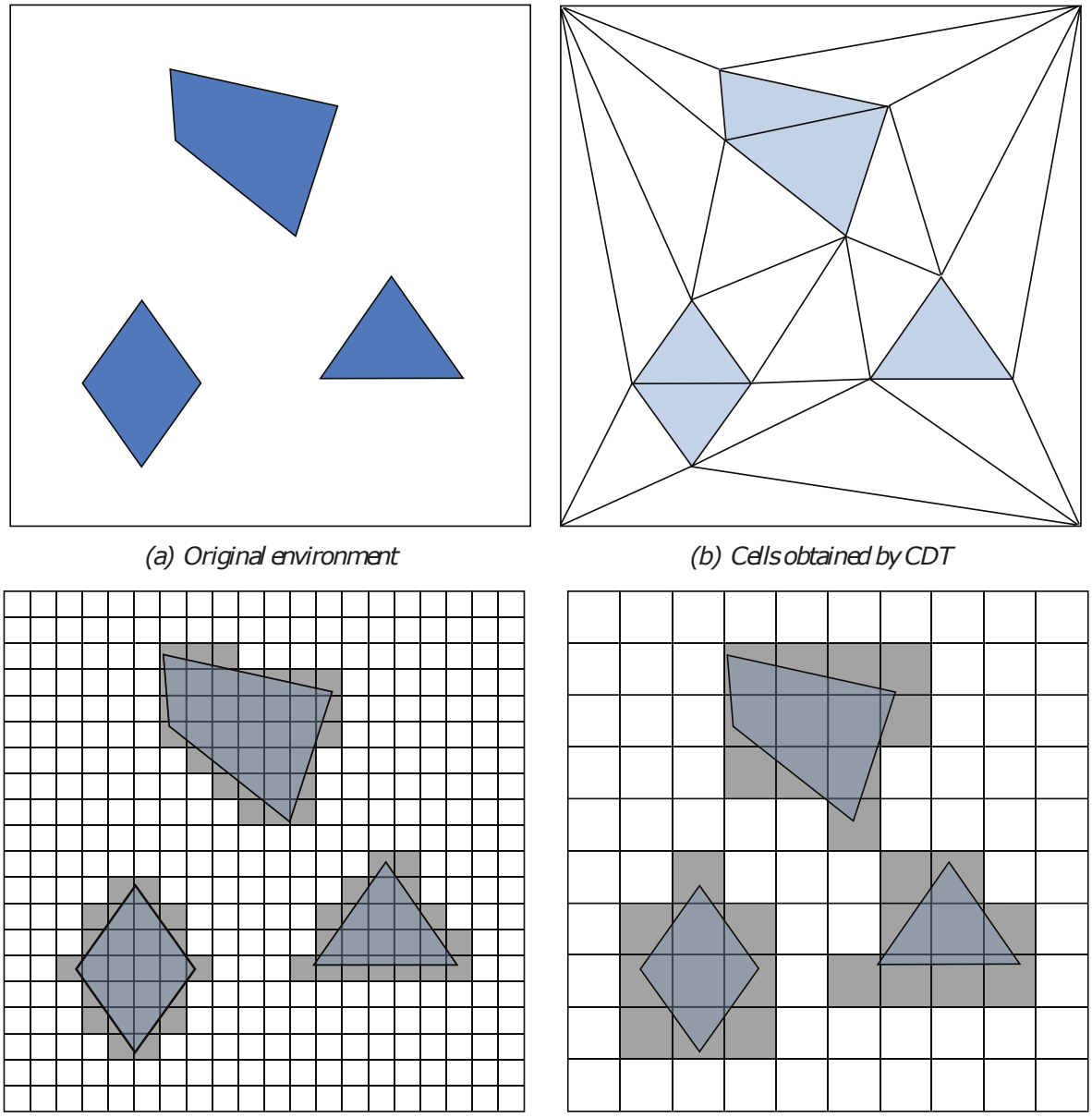


Figure 5.1: Roadmap as a connected graph in the defined environment.

5.2 Cell Decomposition

Cell decomposition methods consist of decomposing the free space into simple regions called *cells* and find a non-directed graph representing the adjacency relation between the cells. This graph is called the *connectivity graph*, its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The method searches a sequence of adjacent cells,

which two of them contain $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$. Each sequence is called a channel and a continuous free path can be computed from them [1]. Figure 5.2 shows some common methods for decomposition such as uniform decomposition Constrained Delaunay Triangulation (CDT) [54].



(a) Original environment

(b) Cells obtained by CDT

(c) Cell decomposition by uniform grids with two resolutions

Figure 5.2: Cell decomposition examples.
Source [54]

5.3 Potential Field

Similar to cell decomposition, in potential field approach the work space is discretized, but in this case into a fine regular grid. Each cell represents particle that repel or attract the mobile robot seen as a mobile particle. The obstacle and the start cells repel the robot and the goal cells attracts it. This approach requires powerful heuristics to guide the search through the grid for a free path. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as potential fields [55].

5.4 Most Studied Algorithms

Based on approaches mentioned above, a lot of specific algorithms have been proposed, some of most studied and popular ones are stated below:

5.4.1 A-star Algorithm

The A-star algorithm or simply A* is an heuristic searching process, its objective is to find the shortest path in a graph from a node $\mathbf{q_{init}}$ to a node $\mathbf{q_{goal}}$. It employs a function $f(\mathbf{q})$ that guides the selection of the next node that will be expanded. $f(\mathbf{q})$ is an estimate of $f^*(\mathbf{q})$ that is the cost of the shortest path that passes through a node $\mathbf{q}$ and achieves $\mathbf{q_{goal}}$. These two functions are computed as shown in equations (5.1) and (5.2) [56].

$$f(\mathbf{q}) = g(\mathbf{q}) + h(\mathbf{q}) \tag{5.1}$$

$$f^*(\mathbf{q}) = g^*(\mathbf{q}) + h^*(\mathbf{q}) \tag{5.2}$$

where $g(\mathbf{q})$ is an estimate of $g^*(\mathbf{q})$, the cost of the shortest path from $\mathbf{q_{init}}$ to $\mathbf{q}$, and $h(\mathbf{q})$ is an estimate of $h^*(\mathbf{q})$, the cost of the shortest path from $\mathbf{q}$ to $\mathbf{q_{goal}}$. If $h(\mathbf{q}) \leq h^*(\mathbf{q})$ for all $\mathbf{q}$ in the graph the heuristic is admissible and the algorithm can be proved optimal.

The implementation of the A* algorithm generally uses two lists named OPEN and CLOSED. The OPEN list stores the nodes that are in the frontier of the search. The CLOSED list stores the nodes that have already been expanded. The expansion of any point $\mathbf{q}$, is carried out using a matrix called the *Connectivity Matrix*, where a number 2 implies current $\mathbf{q}$ position, the number 1 shows where a movement is allowed, and number 0 indicates movement not allowed, examples are C_1: only horizontal-vertical movement allowed, C_2: complete neighbour movement alloved and C_3: Larger movements allowed [55].

$$
\begin{aligned}
C_1 &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \\
C_2 &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \\
C_3 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}
\end{aligned}
\tag{5.3}
$$

In each iteration, the algorithm removes the node from the OPEN list that has the smallest f-value, expands this node inserts its successors that have not been expanded yet in the OPEN list and marks the node as expanded, inserting it in the CLOSED list. It executes these steps until it removes $\mathbf{q_{goal}}$ from the OPEN list or until there are no more nodes available in the OPEN list. In the first case, A* has computed the shortest path between $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ taking into account the connectivity matrix constraints. If the second stop condition is true, there are not any solution available [57].

5.4.2 Probabilistic Roadmap

Path computation using this method is organized in two phases: the preprocessing phase and the query phase. During the preprocessing phase a roadmap is constructed by repeatedly generating random free node positions in the free space of the environment and connecting these configurations using some simple, but very fast planner also known as local planner. In the query phase the algorithm finds a path from the start to the goal position adding those two nodes in the roadmap. Next, a graph search is performed to find a sequence of edges connecting these nodes in the roadmap. Concatenation of the successive path segments transforms the sequence found into a feasible path for the robot [50].

In preprocessing phase is constructed the roadmap randomly as an undirected graph $R = (N, E)$. The nodes N are a set of configurations or points chosen over the free space. The edges in E correspond to simple paths connecting the nodes in N. During the query phase, the roadmap is used to solve individual path planning problems in given environment. For an initial position $\mathbf{q_{init}}$ and a goal position $\mathbf{q_{goal}}$, the method first tries to connect $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ to some two nodes $\mathbf{q'_{init}}$ and $\mathbf{q'_{goal}}$ in N. If successful, it then searches in R for a sequence of edges in E connecting $\mathbf{q'_{init}}$ and $\mathbf{q'_{goal}}$. Finally, it transforms this sequence into a feasible path for the robot by recomputing the corresponding local paths and concatenating them [51].

5.4.3 Genetic Algorithm

The genetic algorithms simulate the natural process selection of any specie population. For robot path planning each chromosome represents a path. Usually, the chromosome consists of initial location (source), target location destination), and intermediate locations crossed by the mobile robot. Each location represents a gene of the chromosome.

When designing a genetic algorithm there are some important steps to follow. First of all the algorithm needs an initial population of chromosomes to manipulate using some operators and produce a new superior generation of chromosomes. The operations could be selection, mutation and crossover. The first generation could be generated randomly or using any (suboptimal) fast motion planner [58].

Each chromosome of any generation is a multidimensional array whose first element are the initial location coordinates, and whose last element are the goal location coordinates, the other elements are the ones changing with genetic algorithm. and their behaviour is evaluated by a cost function J that takes into account criteria like path length, turns, number of obstacles, smoothness and other [51].

The mutation operator chooses different genes from the selected chromosome, and modifies them arbitrarily or following any deterministic criteria, the selection operator normally takes the best chromosomes (best J-valued) and keep them for next generation, this method is called *elitism*, which ensures to conserve the best result always. Finally crossover operator generates a new chromosome for next generation from other two or more chromosomes, using e.g. arithmetic average or gene mapping.

These operations are repeated until the obtaining of a (near)-optimal solution or satisfying some stopping conditions such as a maximum number of iterations or reaching a defined threshold in J-values [58].

Chapter 6

Results and Discussion

For human perception, high level tasks could seem more easy to understand and even to carry out, while low level tasks, although they are implicit in daily activities, if we become aware of them, they are not as simple as they seem. This is the reason why this chapter starts betraying results of high level control implementations.

6.1 High Level algorithms

The implementation of high level tasks involve the use of optimization, graph theory and search algorithms, the heuristic for expansion of the graph representing the work space as a set of cells in regular grids and allocated in the OPEN and CLOSED lists is shown in Figure 6.1 , this expansion is carried out for $\mathbf{q_{init}} = (10, 10)$ and $\mathbf{q_{goal}} = (490, 490)$ in the 500×500 defined environment, the gray region are points in the CLOSED list, while the frontier are the points in the OPEN list. The connectivity matrix in the following:

$$\text{Con} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{6.1}$$

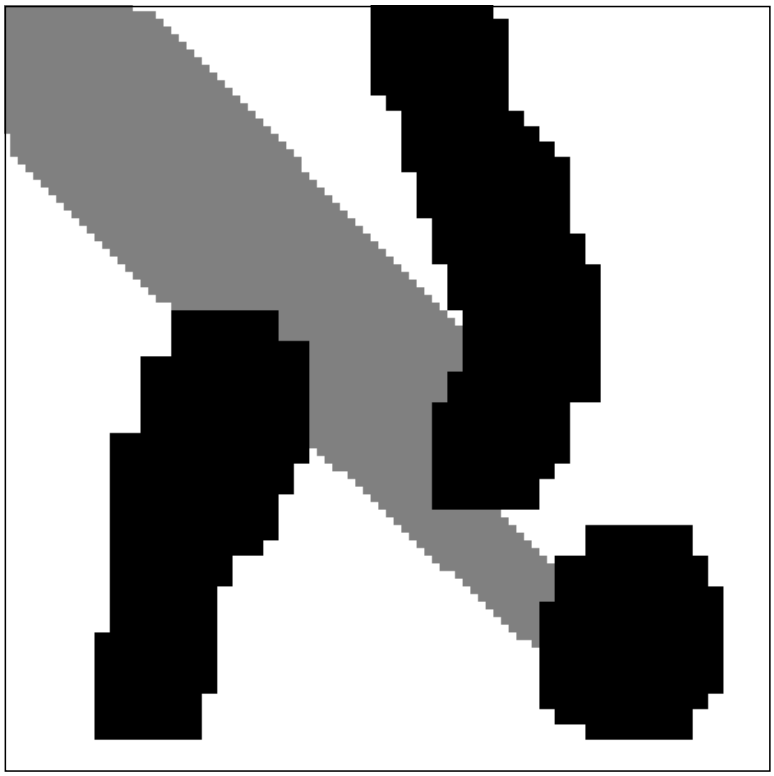


Figure 6.1: Expansion search using connectivity matrix (6.1)

The heuristic for expansion of the graph representing the work space as a set of points drawn from a random generator begins as in Figure 6.2. Then, connect the graph $G = (N, E)$ using connection for simply or straight connected points, i.e. E are one-dimensional curves as shown in Figure 6.3

For genetic algorithms it is used a uniform random $\mathbf{q}$ generator for chromosomes, fulfilling the condition of keeping $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$, the fitness function is de path length using euclidean distance, the stopping criteria is a maximum number of generations defined as 100, or the threshold in J defined as 1.5 times the minimum distance between $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ (straight line).

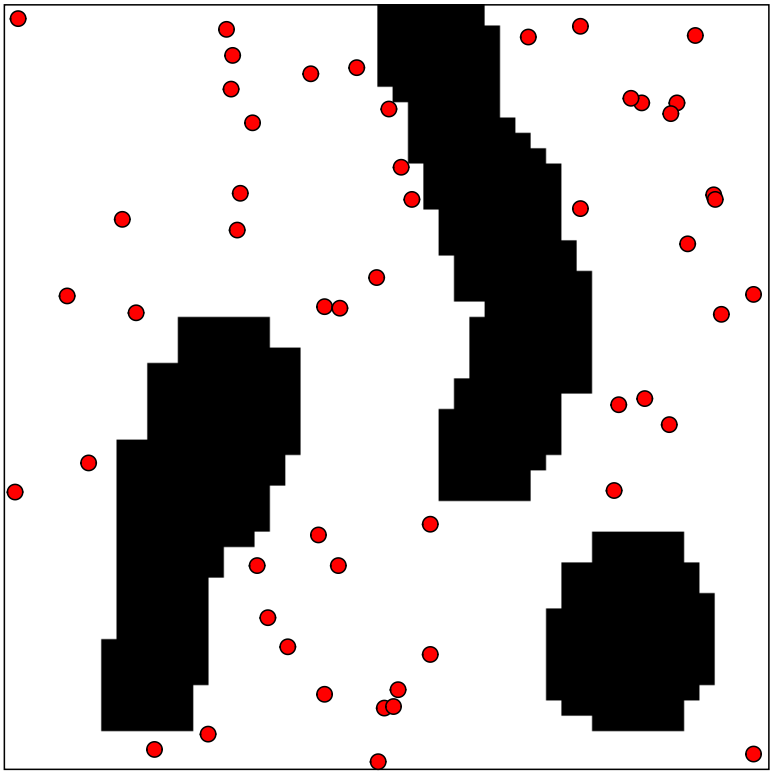


Figure 6.2: Sampling **q** randomly

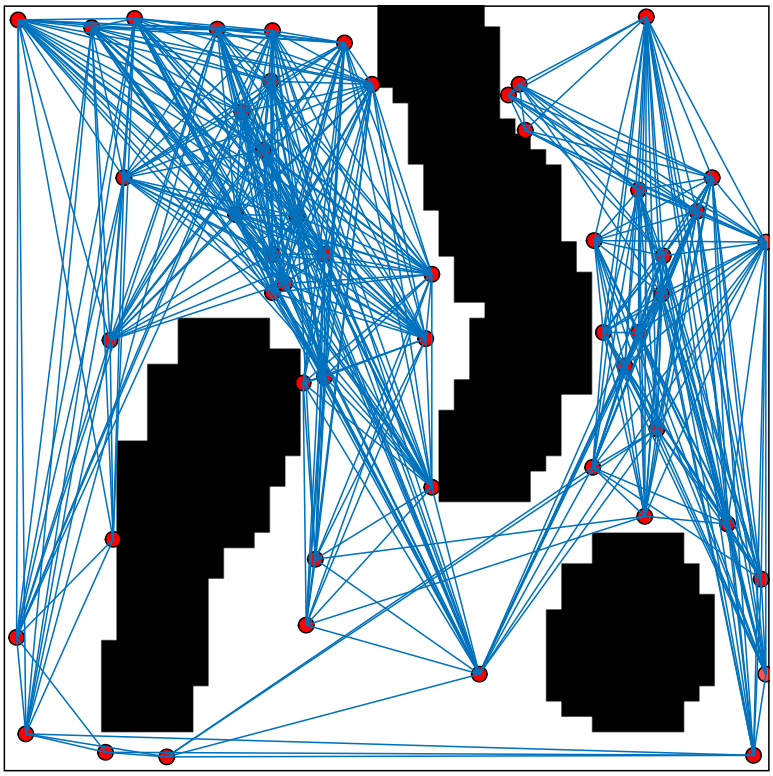


Figure 6.3: Local planning or simple connecting

A* Algorithm

The A algorithm is a search algorithm modified in direction criteria, then, is a *Best search first* algorithm. It looks to expand in the OPEN list for the closest cells to the $\mathbf{q_{goal}}$ location.

Figure 6.4 shows the result of A* algorithm in the shown environment, the result of the A* algorithms is always the same for a given connectivity matrix.

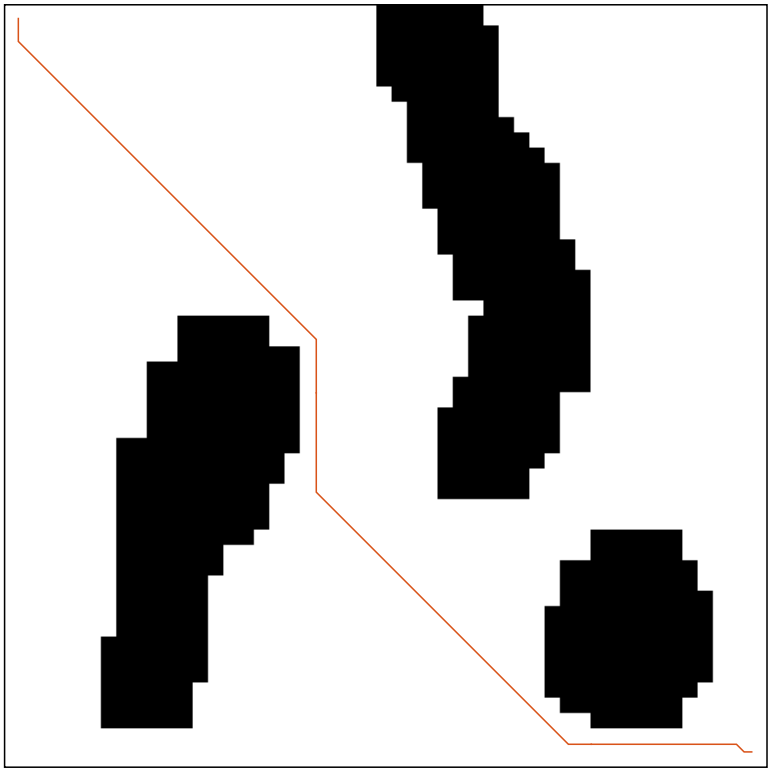


Figure 6.4: Result of the A* Algorithm

Probabilistic Roadmap

Figure 5.1 presents the connected graph, for $\mathbf{q_{init}} = (10, 20)$ and $\mathbf{q_{goal}} = (400, 100)$, these two new points are included to the graph and the connection edges E are found for them, then a Breadth First Search (BSP) is performed to find a sequence of points connecting $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$, Figure 6.5 show the resulting process. Likewise, 6.6 shows a second query phase for $\mathbf{q_{init}} = (400, 100)$ and $\mathbf{q_{goal}} = (100, 500)$ as a following requirement in any arbitrary robot task.

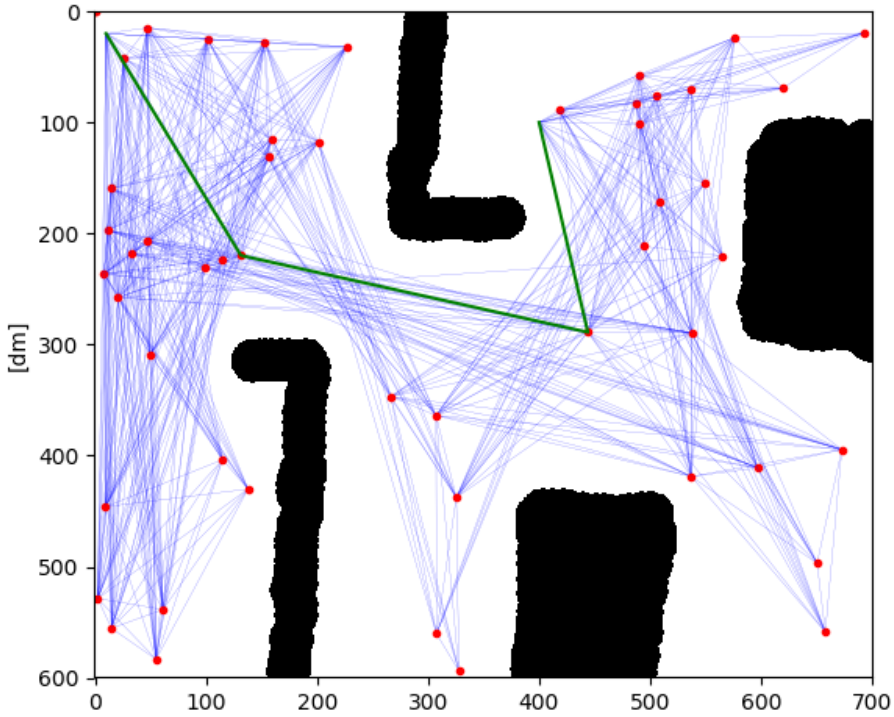


Figure 6.5: Path found between $(10, 20)$ and $(400, 100)$ on a 700×600 environment

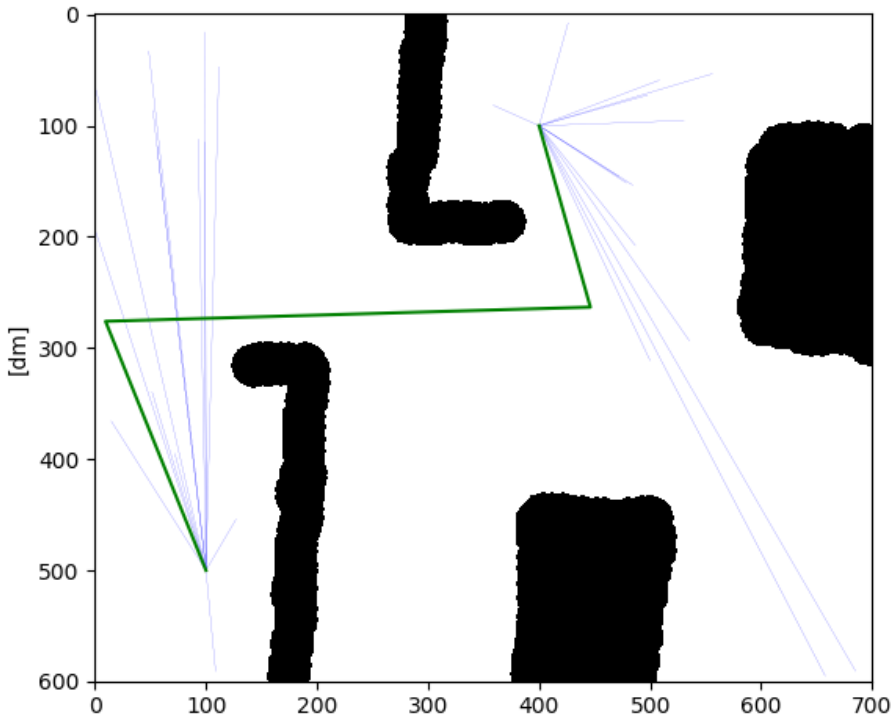


Figure 6.6: Path found between $(10, 20)$ and $(400, 100)$ on a 700×600 environment

In Figure 6.7 the result for the given 500×500 environment is presented

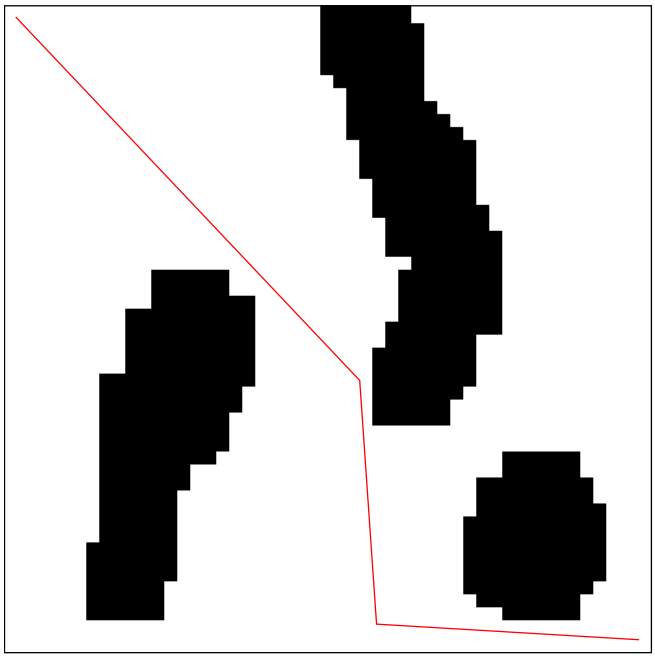


Figure 6.7: Result of the Probabilistic Roadmap Algorithm

Genetic algorithm

Figure 6.8 shows a spline result fitted for a sequence given by the genetic algorithm.

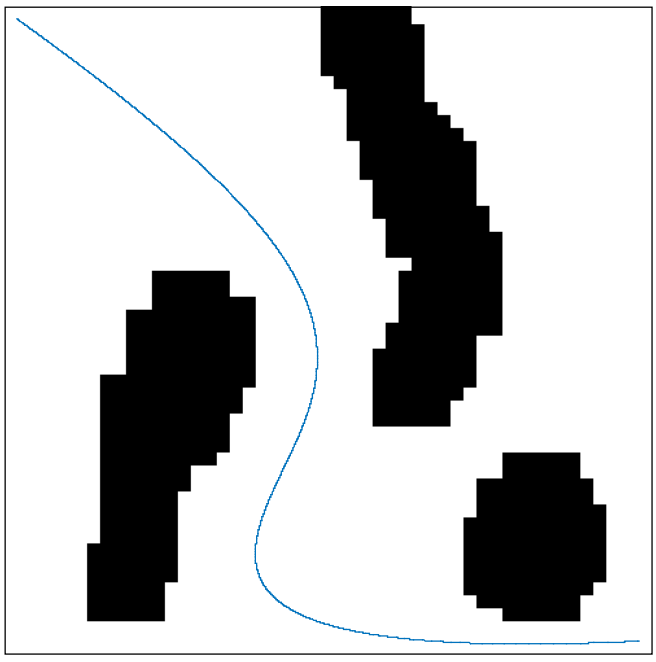


Figure 6.8: Result of the Genetic Algorithm

As experiment, The three methods were executed 12 times in the 500×500 environment, and for each iteration it was registered the elapsed time and the distance of found path, Table 6.1 resumes the results.

Table 6.1: Table of results

Try	A*		PRM		GA	
	time[s]	len	time[s]	len	time[s]	len
1	165.4	746.2	27.3	803.6	155	1074
2	168.3	746.2	25.0	763.9	94.6	1038
3	157.4	746.2	12.8	782.2	90.0	1282
4	168.6	746.2	27.9	716.8	76.2	1036
5	163.2	746.2	7.34	771.4	104.5	1359
6	156.9	746.2	64.6	850.2	115.0	1170
7	155.7	746.2	31.4	832.0	42.7	937
8	163.4	746.2	20.4	681.9	92.9	1224
9	165.8	746.2	29.0	802.2	128.0	964
10	167.7	746.2	17.6	782.6	104.8	1081
11	156.5	746.2	49.9	725.4	91.7	1376
12	160.3	746.2	9.5	798.4	104.5	1193

6.2 Low level control

Low level tasks are attained by automatic control, this section is focused specially in detailing the design of state feedback linearization introduced in Section 4.2 and get a control signal like in Equation (4.56)

The first step define the relative grade of the robot system described by Equations (3.26)-(3.29).

It is known that output variables are x and y: the robot position, in some cases θ is also an output variable, but not for this case, so the output variables are defined as:

$$z_1 = x$$
$$z_2 = y$$

and then:

$$z_1^{(1)} = \dot{x} = v_u \cos(\theta) - d\dot{\theta} \sin\theta$$
$$z_2^{(1)} = \dot{y} = v_u \sin(\theta) + d\dot{\theta} \cos\theta$$
$$z_1^{(2)} = \dot{v}_u \cos\theta - v_u \sin\theta\dot{\theta} - d\ddot{\theta} \sin\theta - d\dot{\theta}^2 \cos\theta \tag{6.2}$$
$$z_2^{(2)} = \dot{v}_u \sin\theta + v_u \cos\theta\dot{\theta} + d\ddot{\theta} \cos\theta - d\dot{\theta}^2 \sin\theta \tag{6.3}$$

replacing Eqs. (3.26) and (3.27) on Eqs. (6.2) and (6.3) respectively, it is easy to see that the terms of the signal control vector

$$u = \begin{bmatrix} \mathcal{T}_R \\ \mathcal{T}_L \end{bmatrix} \tag{6.4}$$

appear in $z_1^{(2)}$ and $z_2^{(2)}$ obtaining $r_1 = 2$ and $r_2 = 2$, then the right equation system is:

$$z_1^{(2)} = \nu_1 \tag{6.5}$$
$$z_2^{(2)} = \nu_2 \tag{6.6}$$

and the left equation system for computing u is:

$$\begin{bmatrix} \mathcal{T}_R \\ \mathcal{T}_L \end{bmatrix} = A^{-1}(x) \begin{bmatrix} \nu_1 + c_0 v_u \dot{\theta} \sin\theta \\ \nu_2 - c_0 v_u \dot{\theta} \cos\theta \end{bmatrix} \tag{6.7}$$

where

$$A(x) = \begin{bmatrix} c_1 \cos\theta - c_2 \sin\theta & c_1 \cos\theta + c_2 \sin\theta \\ c_1 \sin\theta + c_2 \cos\theta & c_1 \sin\theta - c_2 \cos\theta \end{bmatrix} \tag{6.8}$$

with

$$c_0 = \frac{I}{I + Md^2} \tag{6.9}$$
$$c_1 = \frac{1}{MR} \tag{6.10}$$
$$c_2 = \frac{Ld}{R(I + Md^2)} \tag{6.11}$$

The obtained linear system is:

$$\dot{z} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \tag{6.12}$$

where the state vector is given by

$$z = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \tag{6.13}$$

and then the output linear equation is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} z \tag{6.14}$$

The classical state feedback control techniques can be applied to that system, defining a control law as follows:

$$\dot{z} = Az + B\nu \tag{6.15}$$
$$y = Cz \tag{6.16}$$
$$\nu = -Kz \tag{6.17}$$

as a state regulator, but for tracking references with no steady-state errors it can be used a direct loop gain or an integral action approach.

Defining an augmented system:

$$\begin{bmatrix} \dot{z} \\ \dot{e}_i \end{bmatrix} = \begin{bmatrix} A & [0] \\ -C & [0] \end{bmatrix} \begin{bmatrix} z \\ e_i \end{bmatrix} + \begin{bmatrix} B \\ [0] \end{bmatrix} \nu + \begin{bmatrix} [0] \\ R \end{bmatrix} \tag{6.18}$$
$$\dot{z}_a = A_a z_a + B_a \nu + R_a \tag{6.19}$$

where e_i is the error accumulation (integral), then $\dot{e}_i$ is simply the error and R is the setpoint or reference, the signal control vector is now defined as:

$$\nu = -K_a z_a \tag{6.20}$$

where $K_a = \begin{bmatrix} K & -Ki \end{bmatrix}$ with K_i the integral gain.

A simulation is carried out for the DDMR moving across the previously defined 500×500 environment, with a pre-defined path depicted by a set of ordered points (the result of high level control), which is taken as successive setpoints for the controllers; the behavior of the proposed controller is compared with the modified PD controller proposed in [5].

Figure 6.9 shows the performance of the proposed controller working in the DDMR, while Figure 6.10 shows the performance of linear PI controller.

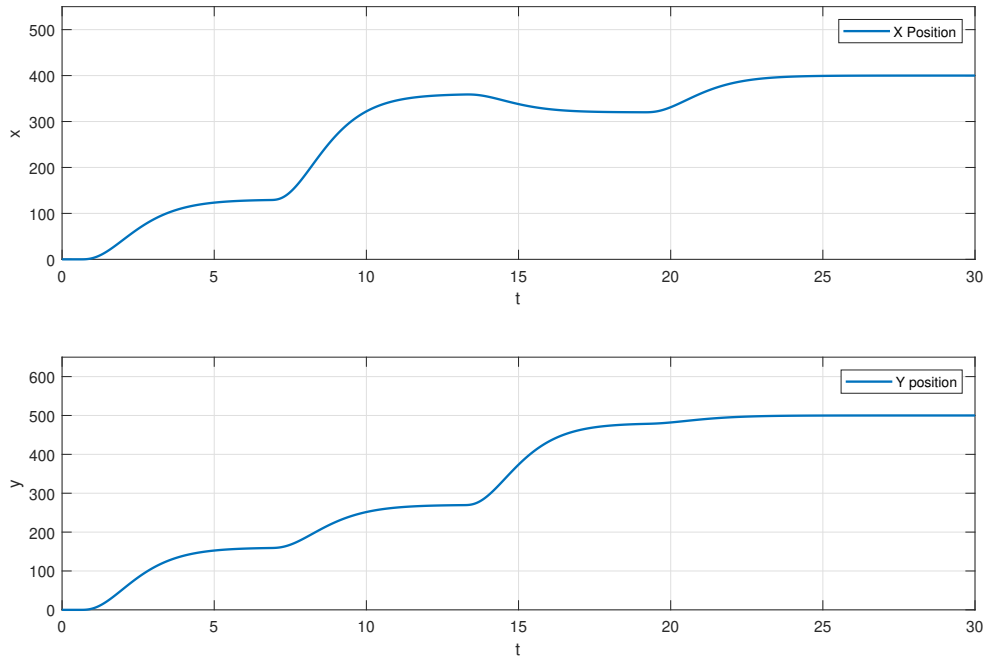


Figure 6.9: Nonlinear controller performance for the given path

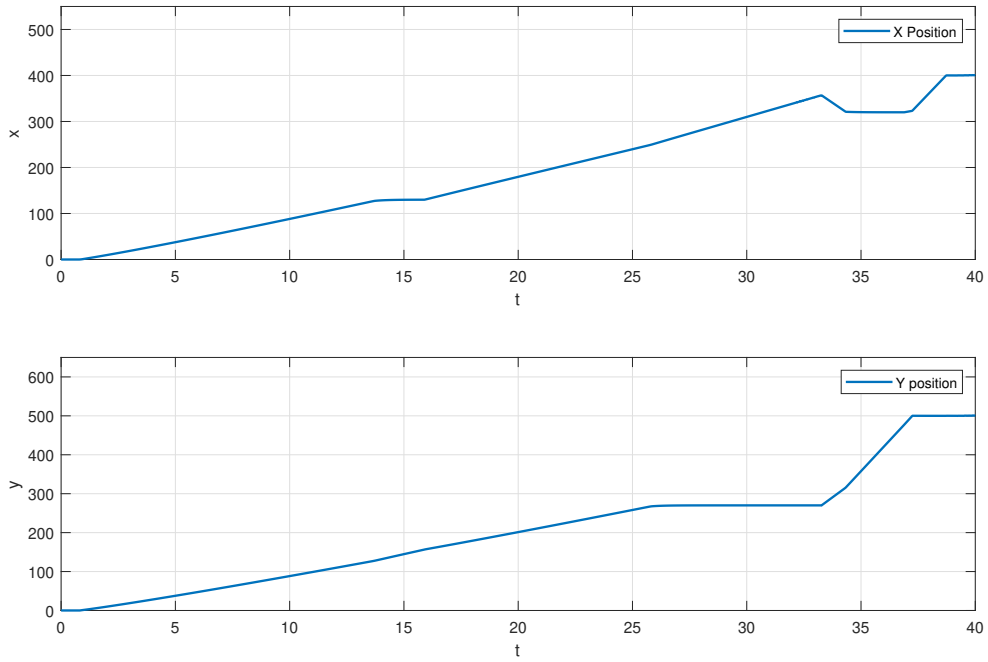


Figure 6.10: Linear PI controller performance for the given path

Figure 6.11 shows the behavior of robot in the environmet; as shown, the linear approaches may have wide deviations from desired paths, while the proposed non-linear controller shows a better performance.

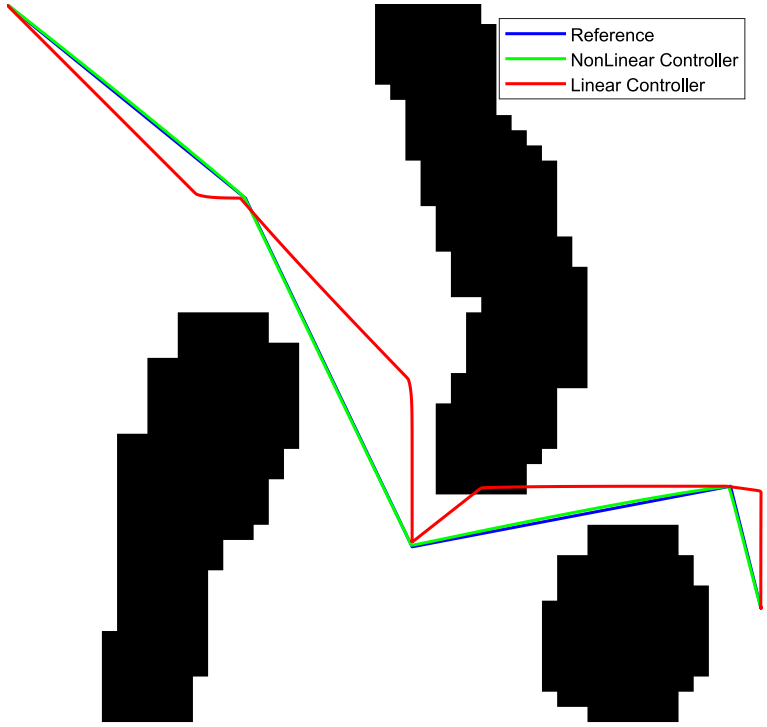


Figure 6.11: Nonlinear and linear controllers behavior for a given path: $\{(0,0),(130,160),(360,270),(320,480),(400,500)\}$

Tables 6.2, 6.3, 6.4 and 6.5 show a numerical analysis of deviations between variables and their references taken from the given point sequence path, as a complement for figures 6.9 and 6.10.

Table 6.2: Nonlinear controller x coordinate analysis

Variable	Min. Dev.	Mean Dev.	Max. Dev.
x	0	36.59	230.04

Table 6.3: Nonlinear controller y coordinate analysis

Variable	Min. Dev.	Mean Dev.	Max. Dev.
y	0	31.14	210.27

Table 6.4: PI Controller x coordinate analysis

Variable	Min. Dev.	Mean Dev.	Max. Dev.
x	0	66.53	230.84

Table 6.5: PI Controller y coordinate analysis

Variable	Min. Dev.	Mean Dev.	Max. Dev.
y	0	71.97	210.63

Now, Table 6.6 shows the distance between the robot position and the reference into the work space shown in Figure 6.11

Table 6.6: Distance analysis between position and reference path

controller	Min. distance	Mean distance	Max. distance
PI	0	53.91	130.86
nonlinear	0	1,53	2,18

Chapter 7

Final remarks

This chapter closes the document stating some important conclusions and final comments on mobile robotics control.

7.1 Conclusions

Robotics is an important field of research in science and engineering, the use of robots for industrial, commercial, military, and even recreational applications has generated the appearance of too many methods for design, manufacturing, measurement, control and even management of robotic platforms. these methods have allowed to become more and more autonomous the current robots.

The study of autonomy in robotics comes coupled with control tasks for reaching generally an alive being behaviour, these tasks are divided in literature as high level tasks, and low level tasks, therefore, two big areas are boarded here, low level control and high level control.

Low level control takes care of very specific tasks, such as reach or track a reference for an internal or external variable, this problem is attained by automatic control, in the other hand, high level control refers to a set of methods and algorithms for motion planning, the combination of high and low level control provide the robot of autonomy, since high level takes an objective and breaks down it into simpler tasks easily executed by low level control structures.

This document presented the general approaches for path planning in mobile robots, and the implementation under computational execution considerations for three well chosen methods: the A-star algorithm, the Probabilistic roadmap algorithm and a genetic algorithm, Figures 6.1 - 6.8 show the implementation results. As shown in these figures and according to the execution resume in Table 6.1, we can say that the application for autonomous robots in this kind of environment, the probabilistic roadmap method is more suitable given a good combination between cost and processing time, results show

path lengths similar to A star algorithm (optimal for a given connectivity matrix), but with processing times considerably lower than other methods.

Low level control consider specially the dynamic behaviour of the robotic system. Not only real systems are essentially nonlinear, but also robotic systems are complex and strongly nonlinear, then, linear approaches for low level control trend to misbehave as shown in figures 6.10 and 6.11, the use of nonlinear methods are definitely the solution for this problem, a good example is the exact feedback linearization presented here, whose performance is shown in figure 6.9 and also in Figure 6.11.

7.2 Future Works

This work was specially focused in low level control and path planning for known environments, the next step for providing each time the robots of more and more autonomy is exploration in partially known or unknown environments. Methods like frontier based exploration and other approaches, both for planning on a known environment or exploration of unknown environments are interesting issues for research.

The implementation of the presented algorithms for space environments (3D) is also an interesting research issues in the sense of this work, we can now consider water and air robots, or simply the unevenness in the terrain a ground robot is moving on.

For low level control, a wide variety of nonlinear methods are now available in literature; optimal, robust, Lyapunov, modified linear methods and others, are some common approaches for testing and advance towards a better control methodology.

7.3 Academic discussion

In this thesis the following articles were published:

- Byron Hernandez, Michael Felipe Cifuentes and Eduardo Giraldo, "A Nonlinear Controller for a Differential Driven Robot", 4th IEEE Colombian Conference on Automatic Control CCAC 2019, Medellin, Colombia. ISBN:978-1-5386-6962-4.
- Byron Hernandez, Eduardo Giraldo; "A Review of Path Planning and Control for Autonomous Robots". 2nd IEEE Colombian Conference on Robotics and Automation. November 1-3, 2018. ISBN:978-1-5386-5541-2.

They are appended in the final part of this document.

Bibliography

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] R. Datouo, F. B. Motto, B. E. Zobo, A. Melingui, I. Bensekrane, and R. Merzouki, "Optimal motion planning for minimizing energy consumption of wheeled mobile robots," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2179–2184, Dec 2017.

[3] R. N. Jazar, *Theory of applied robotics: kinematics, dynamics, and control*. Springer Science & Business Media, 2010.

[4] A. T. Mathew *et al.*, "Design, simulation and implementation of cascaded path tracking controller for a differential drive mobile robot," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1085–1090, IEEE, 2015.

[5] D. E. H. Sánchez, J. R. E. Cuenca, C. C. Sánchez, and J. F. R. Cortés, "Diseño, construcción y modelo dinámico de un robot móvil de tracción diferencial aplicado al seguimiento de trayectorias," in *XXIII Congreso internacional anual de la Sociedad Mexicana de Ingeniería Mecánica (SOMIM)*, 2017.

[6] J. Cornejo, J. Magallanes, E. Denegri, and R. Canahuire, "Trajectory tracking control of a differential wheeled mobile robot: a polar coordinates control and lqr comparison," in *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pp. 1–4, IEEE, 2018.

[7] I. Anvari, *Non-holonomic Differential Drive Mobile Robot Control & Design: Critical Dynamics and Coupling Constraints*. PhD thesis, Arizona State University, 2013.

[8] O. L. Ramírez-Martínez, E. A. Martínez-Garcia, R. E. Mohan, and J. K. Sheba, "Mobile robot adaptive trajectory control: Non-linear path model inverse transformation for model reference," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pp. 877–881, Dec 2014.

[9] R. Kumar, A. Patel, and S. Purwar, "An adaptive control technique for trajectory tracking of a mobile robot using synchronization method," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pp. 344–348, July 2017.

[10] N. Chinthaned and P. Sanposh, "Robust geometric control of a two-tank system," in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp. 1–4, June 2016.

[11] A. O. Baturone, *Robotics: Manipulators and mobile robots, in spanish: "Robótica: manipuladores y robots móviles"*. Marcombo, 2005.

[12] J. J. Craig, *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.

[13] R. Guzman, R. Navarro, M. Beneto, and D. Carbonell, "Robotnik—professional service robotics applications with ros," in *Robot Operating System (ROS)*, pp. 253–288, Springer, 2016.

[14] C. Sandu, M. E. Worley, and J. Morgan, "Experimental study on the contact patch pressure and sinkage of a lightweight vehicle on sand," *Journal of Terramechanics*, vol. 47, no. 5, pp. 343–359, 2010.

[15] W. Commons, "File:lidar equipped mobile robot.jpg — wikimedia commons, the free media repository," 2012. [Online; accessed 16-November-2019].

[16] N. Army, Maritime Security, "U.s. marines will use next-gen robot system to explore dangerous areas," 2019.

[17] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.

[18] M. Travis, "All about drones or unmanned aerial vehicles," 2015.

[19] L. Bañón, S. Ivorra, L. Aragonés Pomares, F. d. B. Varona Moya, M. Cano, R. Tomás, *et al.*, "Innovación en la producción de materiales curriculares para titulaciones de ingeniería civil empleando drones telecomandados," *Universidad de Alicante. Vicerrectorado de Estudios, Formación y Calidad*, 2015.

[20] P. UAV, "Design for business. build for endurance.," 2018.

[21] M. Lingshuai, L. Yang, G. Haitao, X. Hongli, and G. Lingbo, "A new type of small underwater robot for small scale ocean observation," in *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 152–156, June 2016.

[22] N. Degnarain and D. McCauley, "12 robots that could make (or break) the oceans," 2016.

[23] Y. Wang, S. Jiang, F. Yan, L. Gu, and B. Chen, "A new redundancy resolution for underwater vehicle–manipulator system considering payload," *International Journal of Advanced Robotic Systems*, vol. 14, no. 5, p. 1729881417733934, 2017.

[24] G. J. Zeglin, *Uniroo–a one legged dynamic hopping robot.* PhD thesis, Massachusetts Institute of Technology, 1991.

[25] Z. Zhang, J. Zhao, H. Chen, and D. Chen, "A survey of bioinspired jumping robot: takeoff, air posture adjustment, and landing buffer," *Applied bionics and biomechanics*, vol. 2017, 2017.

[26] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*, pp. 4559–4566, IEEE, 2019.

[27] J. Heaston, D. Hong, I. Morazzani, P. Ren, and G. Goldman, "Strider: Self-excited tripedal dynamic experimental robot," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 2776–2777, April 2007.

[28] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008.

[29] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3307–3312, IEEE, 2013.

[30] E. Ackerman, "Spot is boston dynamics nimble new quadruped robot," *IEEE Spectrum*, 2015.

[31] A. Roennau, G. Heppner, M. Nowicki, and R. Dillmann, "Lauron v: A versatile six-legged walking robot with advanced maneuverability," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 82–87, July 2014.

[32] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 258–265, ACM, 2006.

[33] A. Levandowski, A. Schultz, C. Smart, A. Krasnov, D. Song, H. Lee, H. Chau, B. Majusiak, and F. Wang, "Autonomous motorcycle platform and navigation–blue team darpa grand challenge 2005," *Blue Team Autonomous Motorcycle Platform-DARPA. DARPA. Web*, vol. 1, 2010.

[34] R. Velázquez and A. Lay-Ekuakille, "A review of models and structures for wheeled mobile robots: Four case studies," in *2011 15th International Conference on Advanced Robotics (ICAR)*, pp. 524–529, IEEE, 2011.

[35] T. Taniguchi and M. Sugeno, "Trajectory tracking controller design for a tricycle robot using piecewise multi-linear models," in *Proceedings of the international multiconference of engineers and computer scientists IMECS*, 2017.

[36] B. Kiss and E. Szádeczky-Kardoss, "On-line time-scaling control of a kinematic car with one input," in *2007 Mediterranean Conference on Control & Automation*, pp. 1–6, IEEE, 2007.

[37] H. Xu, J. Zhao, D. Tan, and Z. Zhang, "Asymmetrical prototype of a five-wheeled robot and maneuver analysis," in *Intelligent Robotics and Applications* (H. Liu, H. Ding, Z. Xiong, and X. Zhu, eds.), (Berlin, Heidelberg), pp. 488–498, Springer Berlin Heidelberg, 2010.

[38] O. A. Ani, H. Xu, and G. Zhao, "Analysis and modeling of slip for a five-wheeled mobile robot (wmr) in an uneven terrain," in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 154–159, IEEE, 2011.

[39] M. Bajracharya, M. W. Maimone, and D. Helmick, "Autonomy for mars rovers: Past, present, and future," *Computer*, vol. 41, no. 12, pp. 44–50, 2008.

[40] J. L. Martínez, J. Morales, A. Mandow, S. Pedraza, and A. García-Cerezo, "Inertia-based icr kinematic model for tracked skid-steer robots," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 166–171, IEEE, 2017.

[41] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework," *Advances in Robotics & Automation*, vol. 2, no. 2, pp. 1–7, 2013.

[42] D. E. H. Sánchez, J. R. E. Cuenca, C. C. Sánchez, and J. F. R. Cortés, "Diseño, construcción y modelo dinámico de un robot móvil de tracción diferencial aplicado al seguimiento de trayectorias," in *XXIII Congreso internacional anual de la Sociedad Mexicana de Ingeniería Mecánica (SOMIM)*, 2017.

[43] D. Giraldo and E. Giraldo, *Teoría de control análogo*. Universidad Tecnológica de Pereira, Pereira, Colombia, 2009.

[44] A. J. Krener, A. Isidori, and W. Respondek, "Partial and robust linearization by feedback," in *The 22nd IEEE Conference on Decision and Control*, pp. 126–130, Dec 1983.

[45] A. Isidori, *Nonlinear control systems*. Springer Science & Business Media, 2013.

[46] M. R. Kankashvar, H. Kharrati, and A. Khorami, "Design of multivariable controller based on feedback linearization for five-bar linkage manipulator," in *2015 23rd Iranian Conference on Electrical Engineering*, pp. 916–921, May 2015.

[47] L. Meihui, D. Shangfeng, C. Lijun, and H. Yaofeng, "Greenhouse multi-variables control by using feedback linearization decoupling method," in *2017 Chinese Automation Congress (CAC)*, pp. 604–608, Oct 2017.

[48] Y. Zhang, G. Tao, and M. Chen, "Relative degrees and adaptive feedback linearization control of t–s fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 23, pp. 2215–2230, Dec 2015.

[49] M. Korkmaz and A. Durdu, "Comparison of optimal path planning algorithms," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pp. 255–258, Feb 2018.

[50] M. A. Baumann, D. C. Dupuis, S. Léonard, E. A. Croft, and J. J. Little, "Occlusion-free path planning with a probabilistic roadmap.," in *IROS*, pp. 2151–2156, Citeseer, 2008.

[51] R. M. C. Santiago, A. L. D. Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," in *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–5, Dec 2017.

[52] N. Kumar, Z. Vámossy, and Z. M. Szabó-Resch, "Robot path pursuit using probabilistic roadmap," in *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 000139–000144, Nov 2016.

[53] S. R. Cunha, A. C. de Matos, and F. L. Pereira, "An automatic path planing system for autonomous robotic vehicles," in *Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON'93., International Conference on*, pp. 1442–1447, IEEE, 1993.

[54] C. Manta-Caro and J. M. Fernández-Luna, "A discrete-event simulator for the web of things from an information retrieval perspective," in *2014 IEEE Latin-America Conference on Communications (LATINCOM)*, pp. 1–6, IEEE, 2014.

[55] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Springer Science & Business Media, 2012.

[56] J. Yao, C. Lin, X. Xie, A. J. Wang, and C. C. Hung, "Path planning for virtual human motion using improved a* star algorithm," in *2010 Seventh International Conference on Information Technology: New Generations*, pp. 1154–1158, April 2010.

[57] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su, W. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, and Q. Wang, "Path planning of automated guided vehicles based on improved a-star algorithm," in *2015 IEEE International Conference on Information and Automation*, pp. 2071–2076, Aug 2015.

[58] S. Alnasser and H. Bennaceur, "An efficient genetic algorithm for the global robot path planning problem," in *2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pp. 97–102, July 2016.

Academic discussion

The first article "A Review of path planning and control for autonomous robots" presents a review of common planning and controls methods for mobile robots, it was presented as speech and poster in II-CCRA 2019, IEEE Colombian Conference on Robotics and automation.

The second article "A Nonlinear Controller for a Differential Driven Robot" present the use of exact MIMO feedback linearization as central topic, it also shows the design and behaviour of linear PI controller. It was presented as speech in IV-CCAC 2019, IEEE Colombian Conference on Automatic Control.

A Review of Path Planning and Control for Autonomous Robots

Byron Hernández
Electrical Engineering
Universidad Tecnológica de Pereira
Pereira, Colombia
Email: bshernandez@utp.edu.co

Eduardo Giraldo
Department of Electrical Engineering
Universidad Tecnológica de Pereira
Pereira, Colombia
Email: egiraldos@utp.edu.co

***Abstract*—Autonomy is an important topic in modern robotics and it is attained by jointly applying motion planning and control algorithms. This paper presents a review of some well known motion planning techniques, which are: A-star A* , Probabilistic Roadmap and Genetic Algorithms, they are applied to a mobile robot operating into a given environment which contains random-shaped obstacles located arbitrarily. The results of any selected planning algorithm are used as setpoints for a feedback nonlinear controller that operates the motion of the robot in the environment. Simulation results show that probabilistic roadmap has the best performance in time computing, that it has path lengths close to A* algorithm results (optimal under given constraints), and also an effective behaviour on the proposed input-output feedback linearisation control methodology for path tracking.**

I. Introduction

During last decades, robotics has become one of the most important fields of research on engineering, the use of robots allow carrying out critical, difficult and dangerous processes with more precision, reliability and security for humans. Besides, with current technological development, the robots have become not only cheaper but also more effective, faster, flexible and more intelligent [1].

One of the ultimate goals in Robotics, with the introduction of intelligent robots concept is to create *autonomous* robots. Such robots may accept generic descriptions of tasks without many details and execute those tasks without further human intervention. The input descriptions only specify what the user wants that robots do, rather than how they do it. The above implies the use of planning algorithms for determining each step or set of detailed simple tasks robot must do for reach the main generic task [2].

Robotic applications in which robots are designed for moving accross a given workspace, such as exploration or transportation, it is expected from an autonomous robot the ability to plan its own motions across the given environment, avoiding any object or obstacle in there, in literature this problem is stated as "motion planning" [2][3].

Motion planning is one of the most important tasks that autonomous robots must carry out. It involves the definition of a navigation path across the environment $\mathcal{W}$ [4] avoiding any region $\mathcal{B}_i$ defined as an obstacle, the defined path τ goes from an initial position $\mathbf{q_{init}}$ to a goal position $\mathbf{q_{goal}}$ [2].

978-1-5386-5541-2/18/$31.00

There exist a large number of methods proposed for solving the path planning problem, these methods are based essentially in three main approaches: roadmaps, cell decomposition and potential fields [2]. The most important and widely used methods are the A-star algorithm [5][6][7][8], probabilistic roadmaps [9][10][11], and meta-heuristic optimization methods like Particle Swarm or Genetic Algorithms [4][10][12].

But path planning is not the only important problem in development of autonomous robots, methods for tracking the path are also very important [1]. While path planning problem takes into account geometric and kinematic constraints, the methods for following the path take into account the dynamic behaviour of robots and this problem is addressed by automatic control [13].

In this paper is applied an input-output feedback linearisation control method [14], using the results of the path planning algorithm as a simple reference model, sampling the found path from $\mathbf{q_{init}}$ to $\mathbf{q_{goal}}$.

The remaining of this paper is organized as follows: Section 2 introduces the path planning algorithms implemented for comparison, Section 3 explains in detail the control methodology implemented in simulated robot. Section 4 gives the simulation conditions in case study, Section 5 shows the main results of proposed methodology and finally section 6 concludes the article.

II. Path Planning Algorithms

From a large list of methods that have been used for solving the path planning problem, not all of them solve it in its full generality, for instance, some methods require the workspace to be two-dimensional and the objects (robots and obstacles) to be polygonal. Despite external differences, the methods are based on few different general approaches: roadmap, cell decomposition and potential field [2] [15].

A. Roadmap

A roadmap is a network of one-dimensional curves connecting the free-obstacle space, this network is used as a set of standardized paths, and resulting path is chosen as the one connecting $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ with minimum length. In [9][10][11] and [16] is shown in detail how to build a roadmap, and some decision criteria is stated.

B. Cell Decomposition

Cell decomposition methods consist of decomposing the free space into simple regions called *cells* and find a non-directed graph representing the adjacency relation between the cells. This graph is called the *connectivity graph*, its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The method searches a sequence of adjacent cells, which two of them contain $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$. Each sequence is called a channel and a continuous free path can be computed from them [2].

C. Potential Field

Similar to cell decomposition, in potential field approach the workspace is discretized, but in this case into a fine regular grid. This approach requires powerful heuristics to guide the search through the grid for a free path. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as potential fields [15].

D. Most Studied Algorithms

Based on approaches mentioned above, a lot of specific algorithms have been proposed, some of most studied and popular ones are stated below:

*1) **A-star Algorithm**:* The A-star algorithm or simply A* is an heuristic searching process, its objective is to find the shortest path in a graph from a node $\mathbf{q_{init}}$ to a node $\mathbf{q_{goal}}$. It employs a function $f(\mathbf{q})$ that guides the selection of the next node that will be expanded. $f(\mathbf{q})$ is an estimate of $f^*(\mathbf{q})$ that is the cost of the shortest path that passes through a node $\mathbf{q}$ and achieves $\mathbf{q_{goal}}$. These two functions are computed as shown in equations (1) and (2) [5]

$$f(\mathbf{q}) = g(\mathbf{q}) + h(\mathbf{q}) \tag{1}$$

$$f^*(\mathbf{q}) = g^*(\mathbf{q}) + h^*(\mathbf{q}) \tag{2}$$

where $g(\mathbf{q})$ is an estimate of $g^*(\mathbf{q})$, the cost of the shortest path from $\mathbf{q_{init}}$ to $\mathbf{q}$, and $h(\mathbf{q})$ is an estimate of $h^*(\mathbf{q})$, the cost of the shortest path from $\mathbf{q}$ to $\mathbf{q_{goal}}$. If $h(\mathbf{q}) \leq h^*(\mathbf{q})$ for all $\mathbf{q}$ in the graph the heuristic is admissible and the algorithm can be proved optimal.

The implementation of the A* algorithm generally uses two lists named OPEN and CLOSED. The OPEN list stores the nodes that are in the frontier of the search. The CLOSED list stores the nodes that have already been expanded. The expansion of any point $\mathbf{q}$, is carried out using a matrix called the *Connectivity Matrix*, where a number 2 implies current $\mathbf{q}$ position, the number 1 shows where a movement is allowed, and number 0 indicates movement not allowed, examples are C_1: only horizontal-vertical movement allowed, C_2: complete neighbour movement alloved and C_3: Larger movements allowed [15].

$$C_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, C_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

$$C_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

In each iteration, the algorithm removes the node from the OPEN list that has the smallest f-value, expands this node inserts its successors that have not been expanded yet in the OPEN list and marks the node as expanded, inserting it in the CLOSED list. It executes these steps until it removes $\mathbf{q_{goal}}$ from the OPEN list or until there are no more nodes available in the OPEN list. In the first case, A* has computed the shortest path between $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ taking into account the connectivity matrix constraints. If the second stop condition is true, there are not any solution available [6].

*2) **Probabilistic Roadmap**:* Path computation using this method is organized in two phases: the preprocessing phase and the query phase. During the preprocessing phase a probabilistic roadmap is constructed by repeatedly generating random free configurations of the robot and connecting these configurations using some simple, but very fast motion planner also known as local planner. In the query phase the algotithm finds a path from the start and goal configurations to two nodes of the roadmap. Next, a graph search is performed to find a sequence of edges connecting these nodes in the roadmap. Concatenation of the successive path segments transforms the sequence found into a feasible path for the robot [9].

In preprocessing phase is constructed the roadmap randomly as an undirected graph $R = (N, E)$. The nodes N are a set of configurations or points chosen over the free space. The edges in E correspond to simple paths connecting the nodes in N. During the query phase, the roadmap is used to solve individual path planning problems in the input scene. Given an initial position $\mathbf{q_{init}}$ and a goal position $\mathbf{q_{goal}}$, the method first tries to connect $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ to some two nodes $\mathbf{q'_{init}}$ and $\mathbf{q'_{goal}}$ in N. If successful, it then searches in R for a sequence of edges in E connecting $\mathbf{q'_{init}}$ and $\mathbf{q'_{goal}}$. Finally, it transforms this sequence into a feasible path for the robot by recomputing the corresponding local paths and concatenating them [10].

*3) **Genetic Algorithm**:* The genetic algorithms simulate the natural process selection of any specie population. For robot path planning each chromosome represents a path. Usually, the chromosome consists of initial location (source), target location destination), and intermediate locations crossed by the mobile robot. Each location represents a gene of the chromosome.

When designing a genetic algorithm there are some important steps to follow. First of all the algorithm needs an initial population of chromosomes to manipulate using some operators and produce a new superior generation of chromosomes. The operations could be selection, mutation and crossover. The first generation could be generated randomly or using any (suboptimal) fast motion planner [17].

Each chromosome of any generation is a multidimensional array whose first element are the initial location coordinates, and whose last element are the goal location coordinates, the other elements are the ones changing with genetic algorithm. and their behaviour is evaluated by a cost function J that takes into account criteria like path length, turns, number of obstacles, smoothness and other [10].

The mutation operator chooses different genes from the selected chromosome, and modifies them arbitrarily or following any deterministic criteria, the selection operator normally takes the best chromosomes (best J-valued) and keep them for next generation, this method is called *elitism*, which ensures to conserve the best result always. Finally crossover operator generates a new chromosome for next generation from other two or more chromosomes, using e.g. arithmetic average or gene mapping.

These operations are repeated until the obtaining of a (near)-optimal solution or satisfying some stopping conditions such as a maximum number of iterations or reaching a defined threshold in J-values [17].

III. Control Methodology

A. Linear Space State Controller

Let any dynamic system with a linear model as shown in equations (3) and (4).

$$\dot{x} = Ax + Bu \tag{3}$$
$$y = Cx \tag{4}$$

where x is the state vector with n elements for a n-order system, u is the control signal with p elements (the number of inputs) and y the output vector with q elements, thus $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$ and $C \in \mathbb{R}^{p\times n}$.

The feedback controller is designed for solving the regulation problem, that it, to keep stable closed loop system. A block diagram of this controller is shown in Figure 1, as shown, the control signal is [18]

$$u = -Kx \tag{5}$$

where $K \in \mathbb{R}^{p\times n}$, and can be computed using eigen-structure assignment methods [19].

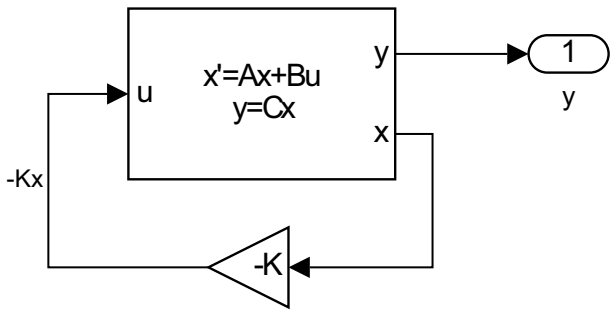


Figure 1. Block diagram of state regulator

B. Feedback Exact Linearization Control

The feedback exact linearisation method transforms on a complete or partial form, non-linear dynamics on linear dynamics, through control input affine non-linear systems in the form of equations (6) and (7).

$$\dot{x} = f(x) + g(x)u \tag{6}$$
$$y = h(x) \tag{7}$$

depending on existence of zero dynamics in the system, there may be two cases:

1) Input-State Linearisation: The system {(6),(7)} is n order. If there exist a transformation of state variables, such that only has an n order differential equation then the it is input-state linearizable, and the system:

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= x_3\\ &\vdots\\ \dot{x}_n &= f(x_1, x_2, \cdots, x_n) + g(x_1, x_2, \cdots, x_n)u\end{aligned}$$

becomes linear if

$$\nu = f(x_1, x_2, \cdots, x_n) + g(x_1, x_2, \cdots, x_n)u \tag{8}$$

and it has a matrix representation:

$$\dot{x} = \begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 0 & 0 & 0 & \cdots & 0\end{bmatrix} x + \begin{bmatrix}0\\0\\\vdots\\0\\1\end{bmatrix} \nu \tag{9}$$

it could be controlled using a simple control law like (5) as $\nu = -Kx$, an then get u from (8) as: [14]

$$u = \frac{\nu - f(x_1, x_2, \cdots, x_n)}{g(x_1, x_2, \cdots, x_n)} \tag{10}$$

2) Input-Output Linearization: If a complete representation cannot be reached since zero dynamics on system, the method could be carried out choosing a new variable set as the output $z = y$, $\dot{z} = \dot{y}$, ... differentiating it until u appears for the first time. The $m < n$ derivatives carried out define the relative order of the system, and similar to previous case:

$$\begin{aligned}\dot{z}_1 &= z_2\\ \dot{z}_2 &= z_3\\ &\vdots\\ \dot{z}_m &= f(z_1, z_2, \cdots, z_m) + f(z_1, z_2, \cdots, z_m)u\end{aligned}$$

and solving the same control law $\nu = -Kz$, u can be gotten from [14]:

$$\nu = f(z_1, z_2, \cdots, z_m) + g(z_1, z_2, \cdots, z_m)u$$

$$u = \frac{\nu - f(z_1, z_2, \cdots, z_m)}{g(z_1, z_2, \cdots, z_m)}$$

IV. Case Study

It is used the model in [20] for probing the path planning and tracking methodology, Figure 2 illustrates the mechanic configuration of the system. Equations (11)-(14) show the dynamical model.

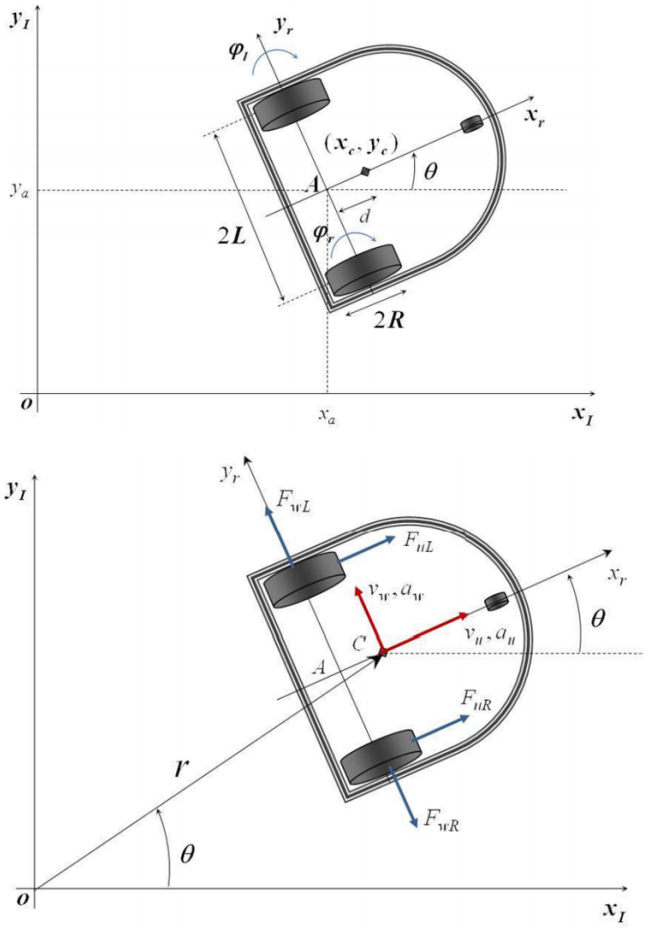


Figure 2. The simplified vehicle dynamic model [20]

$$\dot{v}_u = d\dot{\theta}^2 + \frac{1}{MR}(\tau_R + \tau_L) \tag{11}$$

$$(Md^2 + J)\ddot{\theta} = -Md v_u \dot{\theta} + \frac{L}{R}(\tau_R - \tau_L) \tag{12}$$

$$\dot{x} = v_u \cos(\theta) - v_w \sin\theta \tag{13}$$

$$\dot{y} = v_u \sin(\theta) + v_w \cos\theta \tag{14}$$

where $v_w = d\dot{\theta}$, M is the total robot mass and J its moment of inertia [20]

The conectivity matrix used in A* algorithms is:

$$Con = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{15}$$

Defining an expansion search as shown in Figure 3.

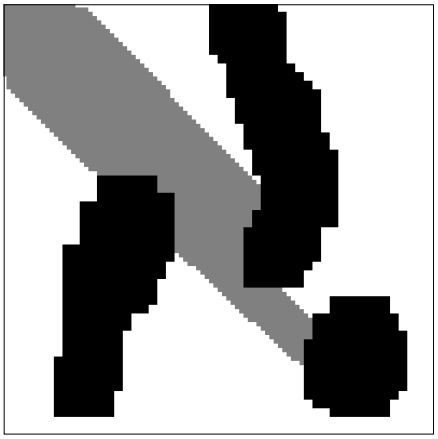


Figure 3. Expansion search using connectivity matrix (15)

For probabilistic Roadmap method the preprocessing phase was developed using a uniform random $\mathbf{q}$ generator, see Figure 4 as an example, and the connection between points (the local planner) is defined as straight lines if feasible. One example is shown in Figure 5

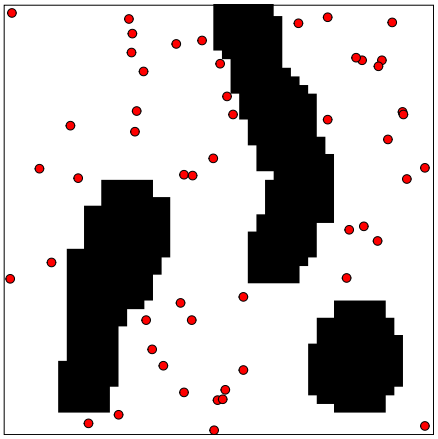


Figure 4. Sampling $\mathbf{q}$ for Probabilistic Roadmap search

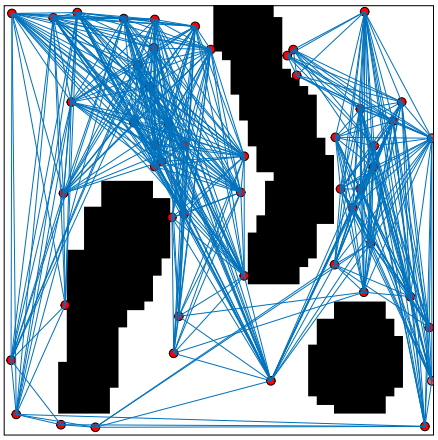


Figure 5. Local planning in Probabilistic Roadmap model

For genetic algorithms it is used a uniform random $\mathbf{q}$ generator for chromosomes, fulfilling the condition of keeping $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$, the fitness function is de path length using euclidean distance, the stopping criteria is a maximum number of generations defined as 100, or the threshold in J defined as 1.5 times the minimum distance between $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$ (straight line).

V. Results

Figure 6 shows the result of A* algorithm in the shown environment, the result of the A* algorithms is always the same for a given connectivity matrix unlike the probabilistic roadmap and genetic algorithm methods. Figures 7 shows one result given by Probabilistic roadmap method and Figure 8 shows one result given by Genetic Algorithm method.

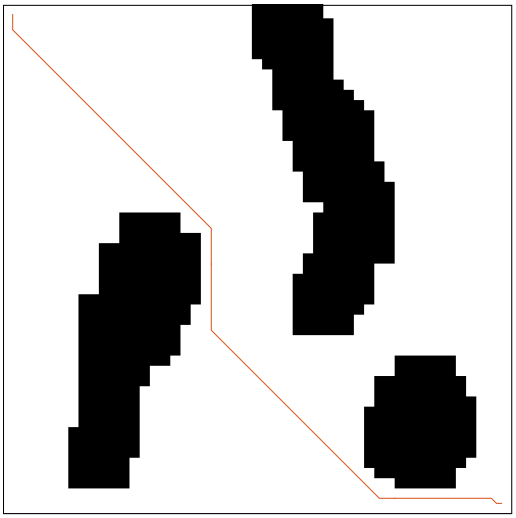


Figure 6. Result of the A* Algorithm

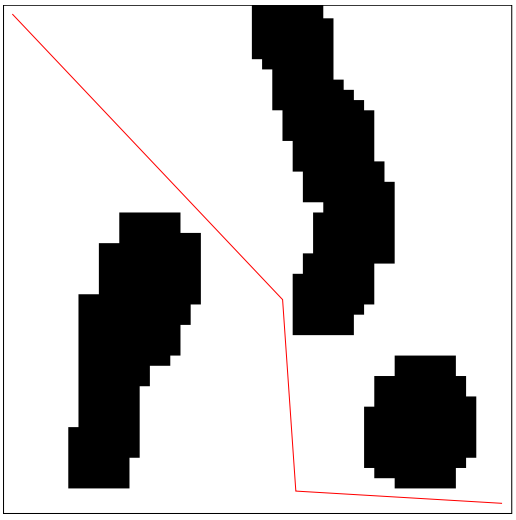


Figure 7. Result of the Probabilistic Roadmap Algorithm

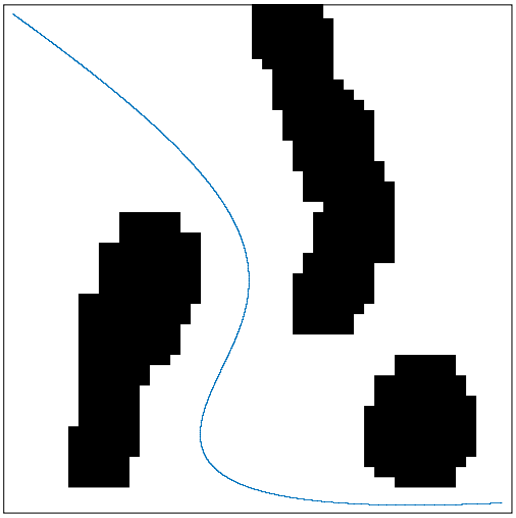


Figure 8. Result of the Genetic Algorithm

Table I
TABLE OF RESULTS

Try	A*		PRM		GA	
	time[s]	len	time[s]	len	time[s]	len
1	165.4	746.2	27.3	803.6	155	1074
2	168.3	746.2	25.0	763.9	94.6	1038
3	157.4	746.2	12.8	782.2	90.0	1282
4	168.6	746.2	27.9	716.8	76.2	1036
5	163.2	746.2	7.34	771.4	104.5	1359
6	156.9	746.2	64.6	850.2	115.0	1170
7	155.7	746.2	31.4	832.0	42.7	937
8	163.4	746.2	20.4	681.9	92.9	1224
9	165.8	746.2	29.0	802.2	128.0	964
10	167.7	746.2	17.6	782.6	104.8	1081
11	156.5	746.2	49.9	725.4	91.7	1376
12	160.3	746.2	9.5	798.4	104.5	1193

The three methods were executed 12 times, and for each iteration it was registered the elapsed time and the distance of found path, Table I resumes the results.

Taking into account the application for autonomous robots in environments with similar characteristics to exposed ones, the probabilistic roadmap method is more suitable. Using the path plan found with this method, the reference model is applied to evaluate the designed controller, Figure 9 shows the behaviour of the controlled robot in the environment and Figure 10 shows the controlled variables response of the system

VI. Conclusions

This work presents a methodology for motion planning and path tracking for an autonomous robot on a known static environment, there were analyzed three of most widely used algorithms in literature, which are, the well known A* algorithm, probabilistic Roadmap and genetic algorithms, the simulation results shows that probabilistic Roadmap show a better performance in processing time. The best result in path length corresponds to A* algorithm, but processing time is considerably higher than both probabilistic Roadmap and

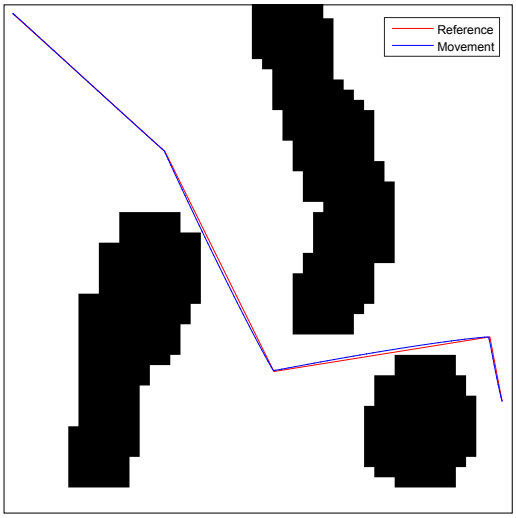


Figure 9. Behaviour of controlled robot in the given environment for a path with $\mathbf{q_{init}} = (10, 10)$ and $\mathbf{q_{goal}} = (390, 490)$

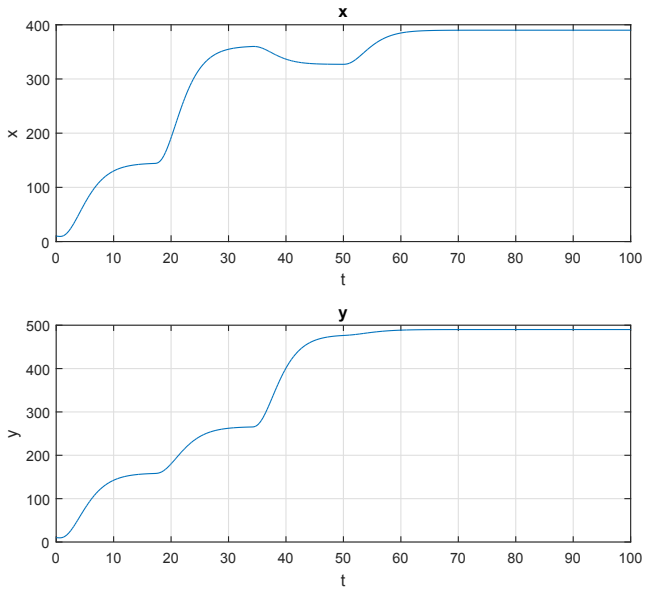


Figure 10. Controlled variables response of the system for the given sequence $\mathbf{q} = \{(10, 10), (145, 159), (361, 266), (327, 478), (390, 490)\}$

genetic Algorithms, simulation also shows that the exact input-output linearization shows a good performance tracking given paths.

AKNOWLEDGEMENTS

This research is supported by "Programa Jóvenes Investigadores e Innovadores por la paz, convocatoria 775-2017" funded by Colciencias and Universidad Tecnológica de Pereira, research project "Diseño de un vehículo terrestre no tripulado para la inactivación de minas anti-persona el el territorio colombiano". The authors also thank to Maestría en Ingeniería Eléctrica at Universidad Tecnolgica de Pereira for the financial support, and the research group on Automatic Control COL0031472.

REFERENCES

[1] J. J. Craig, *Robotica*. Prentice Hall, 2006.
[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[3] R. Datouo, F. B. Motto, B. E. Zobo, A. Melingui, I. Bensekrane, and R. Merzouki, "Optimal motion planning for minimizing energy consumption of wheeled mobile robots," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2179–2184, Dec 2017.
[4] M. Korkmaz and A. Durdu, "Comparison of optimal path planning algorithms," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pp. 255–258, Feb 2018.
[5] J. Yao, C. Lin, X. Xie, A. J. Wang, and C. C. Hung, "Path planning for virtual human motion using improved a* star algorithm," in *2010 Seventh International Conference on Information Technology: New Generations*, pp. 1154–1158, April 2010.
[6] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su, W. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, and Q. Wang, "Path planning of automated guided vehicles based on improved a-star algorithm," in *2015 IEEE International Conference on Information and Automation*, pp. 2071–2076, Aug 2015.
[7] T. Nayl, M. Q. Mohammed, and S. Q. Muhamed, "Obstacles avoidance for an articulated robot using modified smooth path planning," in *2017 International Conference on Computer and Applications (ICCA)*, pp. 185–189, Sept 2017.
[8] X. Huang, Q. Jia, and G. Chen, "Collision-free path planning method with learning ability for space manipulator," in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1790–1795, June 2017.
[9] M. A. Baumann, D. C. Dupuis, S. Léonard, E. A. Croft, and J. J. Little, "Occlusion-free path planning with a probabilistic roadmap.," in *IROS*, pp. 2151–2156, Citeseer, 2008.
[10] R. M. C. Santiago, A. L. D. Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," in *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–5, Dec 2017.
[11] N. Kumar, Z. Vmossy, and Z. M. Szab-Resch, "Robot path pursuit using probabilistic roadmap," in *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 000139–000144, Nov 2016.
[12] Z. Elmi and M. . Efe, "Multi-objective grasshopper optimization algorithm for robot path planning in static environments," in *2018 IEEE International Conference on Industrial Technology (ICIT)*, pp. 244–249, Feb 2018.
[13] R. N. Jazar, *Theory of applied robotics: kinematics, dynamics, and control*. Springer Science & Business Media, 2010.
[14] A. Isidori, *Nonlinear control systems*. Springer Science & Business Media, 2013.
[15] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Springer Science & Business Media, 2012.
[16] S. R. Cunha, A. C. de Matos, and F. L. Pereira, "An automatic path planing system for autonomous robotic vehicles," in *Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON'93., International Conference on*, pp. 1442–1447, IEEE, 1993.
[17] S. Alnasser and H. Bennaceur, "An efficient genetic algorithm for the global robot path planning problem," in *2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pp. 97–102, July 2016.
[18] D. Giraldo and E. Giraldo, "Teoría de control análogo," *Universidad Tecnológica de Pereira, Pereira, Colombia*, 2009.
[19] S. Srinathkumar and Srinathankumar, *Eigenstructure control algorithms: applications to aircraft/rotorcraft handling qualities design*. Institution of Engineering and Technology, 2011.
[20] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework," *Advances in Robotics & Automation*, vol. 2, no. 2, pp. 1–7, 2013.

2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)
October 15-18, 2019. Diez Hotel, Medellin, Colombia.

A Nonlinear Controller for a Differential Driven Robot

Byron Hernández
Department of Electrical Engineering
Universidad Tecnológica de Pereira
Pereira, Colombia
Email: bshernandez@utp.edu.co

Michael Felipe Cifuentes Molano
Department of Electrical Engineering
Universidad Tecnológica de Pereira
Pereira, Colombia
Email: mcifuentes@utp.edu.co

Eduardo Giraldo
Department of Electrical Engineering
Universidad Tecnológica de Pereira
Pereira, Colombia
Email: egiraldos@utp.edu.co

***Abstract*—In this paper, it is presented a nonlinear controller using a multivariable approach for the exact feedback linearization method, applied to the model of a differential driven robot operating into a two-dimensional space. The system shows an excellent performance tracking setpoints from a given path, and this provides a better control behavior compared to linear controllers, like the multivariable PI controller.**

I. Introduction

During the last decades, autonomous robots have become one of the most important fields of research on engineering, the use of robots allow carrying out critical, difficult and dangerous processes with more precision, reliability, and security for humans. Besides, with current technological development and research, robots are easier to access; they are cheaper, more productive, faster, flexible and more intelligent [1].

One of the essential topics in the development of autonomous robots is to control the behavior of the electromechanical arrangement based on its dynamical representation or model [2]. Differential Driven Robots are a particular case lately studied since they have many mechanical advantages, e.g., they can turn around its axle without any displacement, but at the same time, this implies a more complex dynamical model with nonholonomic constraints and strong nonlinearities which are often hard to attain by conventional controllers [3].

Control tasks required in mobile robots are generally related to path tracking; they imply the ability to reach a set of consecutive reference points or a set of geometrical primitives, such as straight lines or curves given by a planner that performs the breakdown of high-level tasks [4].

Approaches, like shown in [5], [6] and [7], propose essentially linear control techniques which do not consider the complete nonlinearities of the differential driven robot, and it infers disadvantages, like deviation errors or over impulse in transient response. Tracking errors may cause collisions with obstacles due to deviation from the planned path [2].

The problem has been previously addressed but not in all its generality, for example, in [8] a nonlinear controller is designed but only for speed control, it reaches path tracking calculating an adaptive reference model using polynomial regressions. Other works like [9] propose adaptive algorithms, but they always have bad behaviour at the beggining which could cause path deaviations and consequently possible collisions.

One common method of nonlinear control is the exact feedback linearization, but it is mainly popular for SISO systems, the approach for MIMO systems has some advanced mathematical fundamentals, they are introduced in [10], in this papers those fundamentals are used for designing a nonlinear controller for the Differential Driven Mobile Robot (DDMR).

The remaining of this paper is organized as follows: Section 2 presents the kinematic constraints and dynamical representation of a differential driven robot; Section 3 introduces the mathematical resources used in multivariate exact feedback linearization method; Section 4 shows the design of the nonlinear controller to the studied vehicle; Section 5 draws some simulation results, and finally Section 6 concludes the article.

II. Robot Model

The Differential Driven Mobile Robot (DDMR) is one of the most studied structures in robotics. It generally consists of a chassis with two fixed electric motors opposed to each other [3]. They usually have one extra rear wheel as a third prop, or they are connected to a continuous track configuration [5] as shown in Figure 1.

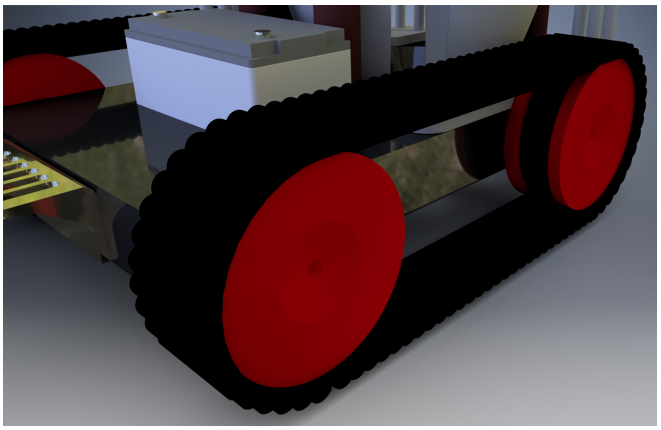


Figure 1. Continuous track configuration in a differential driven robot

Based on references [2], [3], [5] and [6], both kinematic and dynamic analysis is carried out defining three key points as coordinated frames:

978-1-5386-6962-4/19/$31.00 ©2019 IEEE

- $\Sigma_0(X_0, Y_0)$ The inertial frame, a fixed or universal coordinated system representing the environment.
- $\Sigma_1(X_1, Y_1)$ The robot geometric center, conveniently located collinearly with the axles of the motors.
- $\Sigma_2(X_2, Y_2)$ The mass center in a different location from the geometric center for more realistic and flexible modeling.

Figure 2 shows the coordinated frames and some general dimensional considerations for the robot.

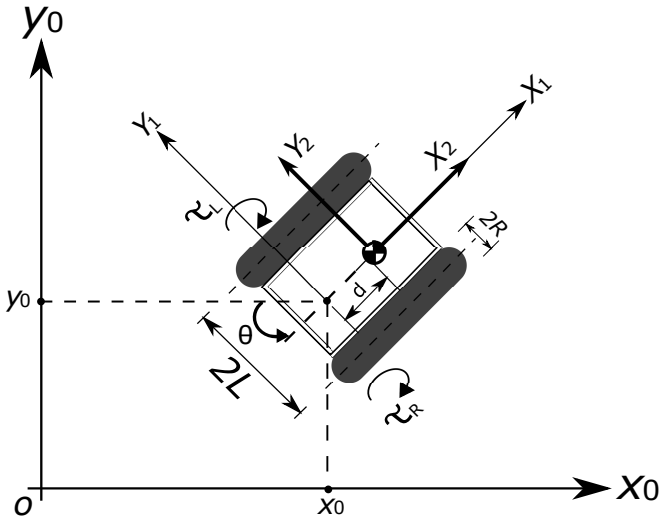


Figure 2. Robot coordinated frames

The relationship between torques $\mathcal{T}$ and forces F applied to the robot is $\mathcal{T} = F \cdot R$, then:

$$F = \frac{\mathcal{T}}{R}$$

Using a polar coordinate approach and drawing a free body diagram in Figure 3 the analysis follows

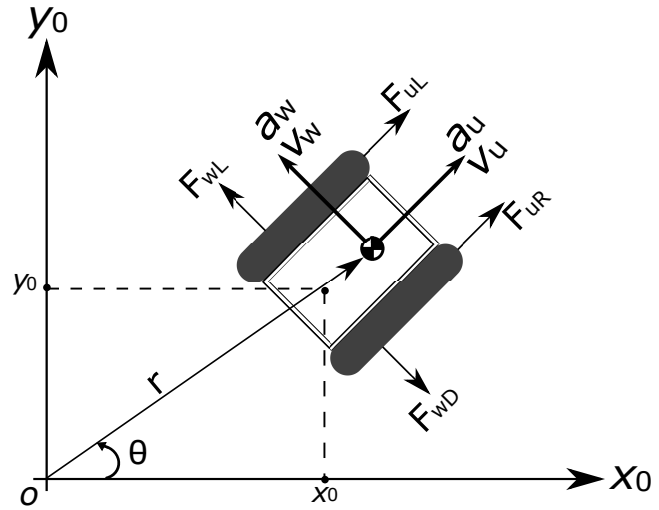


Figure 3. Robot free body diagram

$$\hat{r} = re^{j\theta} \tag{1}$$
$$\dot{\hat{r}} = \dot{r}e^{j\theta} + jr\dot{\theta}e^{j\theta} \tag{2}$$
$$\ddot{\hat{r}} = \ddot{r}e^{j\theta} + 2j\dot{r}\dot{\theta}e^{j\theta} + jr\ddot{\theta}e^{j\theta} - r\dot{\theta}^2e^{j\theta} \tag{3}$$

separating velocity and acceleration into radal and tangencial terms, we have:

$$\dot{\hat{r}} = [\dot{r}]e^{j\theta} + [r\dot{\theta}]e^{j\left(\theta+\frac{\theta}{2}\right)}$$
$$\ddot{\hat{r}} = [\ddot{r} - r\dot{\theta}^2]e^{j\theta} + [2\dot{r}\dot{\theta} + r\ddot{\theta}]e^{j\left(\theta+\frac{\theta}{2}\right)}$$

and then, variables in Figure 3 follow:

$$v_u = \dot{r} \tag{4}$$
$$v_w = r\dot{\theta} \tag{5}$$
$$a_u = \ddot{r} - r\dot{\theta}^2 \tag{6}$$
$$a_w = 2\dot{r}\dot{\theta} + r\ddot{\theta} \tag{7}$$

The movement equations are:

$$Ma_u = F_{uR} + F_{uL} \tag{8}$$
$$Ma_w = F_{wL} - F_{wR} \tag{9}$$
$$I\ddot{\theta} = L(F_{uR} - F_{uL}) + d(F_{wR} - F_{wL}) \tag{10}$$

and from Eqs. (4)-(7) we have:

$$a_u = \dot{v}_u - v_w\dot{\theta} \tag{11}$$
$$a_w = \dot{v}_w + v_u\dot{\theta} \tag{12}$$

and then

$$M(\dot{v}_u - v_w\dot{\theta}) = F_{uR} + F_{uL} \tag{13}$$
$$M(\dot{v}_w + v_u\dot{\theta}) = F_{wR} - F_{wL} \tag{14}$$
$$I\ddot{\theta} = L(F_{uR} - F_{uL}) - d(F_{wR} - F_{wL}) \tag{15}$$

$$\dot{v}_u = \frac{F_{uR} + F_{uL}}{M} + v_w\dot{\theta} \tag{16}$$
$$\dot{v}_w = \frac{F_{wR} - F_{wL}}{M} - v_u\dot{\theta} \tag{17}$$
$$\ddot{\theta} = \frac{L}{I}(F_{uR} - F_{uL}) + \frac{d}{I}(F_{wR} - F_{wL}) \tag{18}$$

The velocity of mass center Σ_2, $\dot{x}_2$ and $\dot{y}_2$, respect to the inertial frame Σ_0, are given by

$$\dot{x}_2 = v_u\cos\theta - v_w\sin\theta \tag{19}$$
$$\dot{y}_2 = v_u\sin\theta + v_w\cos\theta \tag{20}$$

and respect to the frame Σ_1:

$$x_2 = x_0 + d\cos\theta$$
$$\dot{x}_2 = \dot{x}_0 - d\dot{\theta}\sin\theta \tag{21}$$
$$y_2 = y_0 + d\sin\theta$$
$$\dot{y}_2 = \dot{y}_0 + d\dot{\theta}\sin\theta \tag{22}$$

Matching Eqs. (19) with (21) and (20) with (22), and taking the sum of their squares, we have:

$$\dot{x}_1^2 + (d\dot{\theta}\sin\theta)^2 + y_1^2 + (d\dot{\theta}\cos\theta)^2 = v_u^2 + v_w^2 \tag{23}$$

considering that robot has no lateral slipping ($\dot{y}_1 = 0$) and that $\dot{x}_1 = v_u$ given their coaxiality, then:

$$v_u^2 + (d\dot{\theta})^2(\sin^2\theta + \cos^2\theta) = v_u^2 + v_w^2$$
$$d\dot{\theta} = v_w \tag{24}$$

Finally the set of dynamical equations describing the DDMR are given by Eqs. (25)-(28) similar to [3]

$$\dot{v}_u = d\dot{\theta}^2 + \frac{1}{MR}(\mathcal{T}_R + \mathcal{T}_L) \tag{25}$$
$$\ddot{\theta} = -\frac{Md}{I + Md^2}v_u\dot{\theta} + \frac{L}{(I + Md^2)R}(\mathcal{T}_R - \mathcal{T}_L) \tag{26}$$
$$\dot{x} = v_u\cos(\theta) - d\dot{\theta}\sin\theta \tag{27}$$
$$\dot{y} = v_u\sin(\theta) + d\dot{\theta}\cos\theta \tag{28}$$

III. Multivariable exact feedback linearization

The exact feedback linearization method was first proposed by Isidori et al. in the eighties [11] only for single-input single-output (SISO) sytems. It consists of finding a control law that transforms nonlinear dynamics on a complete or partial way into linear dynamics, it is only possible if the system is control input affine (represented as in Equations (29) and (30)).

$$\dot{x} = \mathbf{f}(x) + g(x)u \tag{29}$$
$$y = h(x) \tag{30}$$

if all nonlinearities are concentrated in a

$$\dot{x}_r = f(x) + g(x)u = \nu \tag{31}$$

then the system becomes linear and the control law has the following form [12]:

$$u = \alpha(x) + \beta(x)\nu \tag{32}$$

where $\alpha(x) = -f(x)/g(x)$ and $\beta(x) = 1/g(x)$

Depending on existence of zero dynamics in the system, there may be two cases (see [11]):

- Input-State Linearization, for trivial zero dynamics ($r = n$)
- Input-Output Linearization, for non-trivial zero dyamics ($r < n$)

A. Multvariable extention for feedback linearization method

For multiple-input multiple-output (MIMO) system, the Isidori's method could be extended using geometric control concepts [10].

Let a MIMO system given by the following equations [13]

$$\dot{x} = \mathbf{f}(x) + \sum_{i=1}^{m} g_i(x)u \tag{33}$$
$$y_1 = h_1(x) \tag{34}$$
$$\vdots$$
$$y_m = h_m(x) \tag{35}$$

or in a compact way

$$\dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u \tag{36}$$

where

$$\mathbf{g}(x) = [g_1(x), g_2(x), ..., g_m(x)] \tag{37}$$
$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{38}$$

The system has a relative grade $\{r_1 \cdots r_m\}$ for each output if:

$$\mathcal{L}_{g_j}\mathcal{L}_{\mathbf{f}}^k h_i(x) = 0 \text{ for all } 1 \le j \le m,\ k < r_i - 1,\ 1 \le i \le m$$

where $\mathcal{L}_{\mathbf{f}}h$ is the Lie derivative of the scalar function h in the direction of the vectorial field $\mathbf{f}$, it is defined as $\nabla h \cdot \mathbf{f}$. The k-th Lie derivative is then recursively defined as $\mathcal{L}_{\mathbf{f}}^0 h = \mathbf{f}$ and $\mathcal{L}_{\mathbf{f}}^k h = \mathcal{L}_{\mathbf{f}}(\mathcal{L}_{\mathbf{f}}^{k-1}h)$

The expression $\mathcal{L}_{\mathbf{g}}\mathcal{L}_{\mathbf{f}}^k h = \nabla(\mathcal{L}_{\mathbf{f}}h)\cdot\mathbf{g}$ is called Lie derivative of the scalar function h respect to vectorial fields $\mathbf{f}$ and $\mathbf{g}$

Then the decoupling matrix can be defined [14]:

$$A(x) = \begin{bmatrix} \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_1-1}h_1(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_1-1}h_1(x) \\ \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_2-1}h_2(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_2-1}h_2(x) \\ \vdots & \vdots & \vdots \\ \mathcal{L}_{g_1}\mathcal{L}_{\mathbf{f}}^{r_m-1}h_m(x) & \cdots & \mathcal{L}_{g_m}\mathcal{L}_{\mathbf{f}}^{r_m-1}h_m(x) \end{bmatrix} \tag{39}$$

r_i is the relative grade of the i-th output $y_i(t)$, that is, the number of derivatives required for the first emergence of at least one u term [15].

This matrix is used to compute the control signals vector for the system defined from (33)-(35) as:

$$y_i^{(r_i)} = \mathcal{L}_{\mathbf{f}}^{r_i}h_i + \sum_{j=1}^{m}\mathcal{L}_{g_j}\mathcal{L}_{\mathbf{f}}^{r_i-1}h_i u_j \tag{40}$$

$$\begin{bmatrix} y_1^{(r_1)} \\ y_2^{(r_2)} \\ \vdots \\ y_m^{(r_m)} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_{\mathbf{f}}^{r_1}h_1(x) \\ \mathcal{L}_{\mathbf{f}}^{r_2}h_2(x) \\ \vdots \\ \mathcal{L}_{\mathbf{f}}^{r_m}h_m(x) \end{bmatrix} + A(x)\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{41}$$

Leading m equations in the form:

$$y_i^{(r_i)} = z_i^{(r_i)} = \nu_i \tag{42}$$

to compute the signal control vector as follows:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} = A^{-1}(x)\begin{bmatrix} \nu_1 - \mathcal{L}_{\mathbf{f}}^{r_1}h_1(x) \\ \nu_2 - \mathcal{L}_{\mathbf{f}}^{r_2}h_2(x) \\ \vdots \\ \nu_m - \mathcal{L}_{\mathbf{f}}^{r_m}h_m(x) \end{bmatrix} \tag{43}$$

provided that $A(x)$ is not singular.

IV. Designing the controller for the Differential Driven Robot

The first step to find a linear control law is define the realtive grade of the robot system decribed by Eqs. (25)-(28).

It is known that output variables are x and y, the robot position, in some cases θ is algo an output variables, but not in ths study. Then, the output variables are defined as:

$$z_1 = x$$
$$z_2 = y$$

and then:

$$z_1^{(1)} = \dot{x} = v_u \cos(\theta) - d\dot{\theta}\sin\theta$$
$$z_2^{(1)} = \dot{y} = v_u \sin(\theta) + d\dot{\theta}\cos\theta$$
$$z_1^{(2)} = \dot{v}_u \cos\theta - v_u \sin\theta\dot{\theta} - d\ddot{\theta}\sin\theta - d\dot{\theta}^2\cos\theta \quad (44)$$
$$z_2^{(2)} = \dot{v}_u \sin\theta + v_u \cos\theta\dot{\theta} + d\ddot{\theta}\cos\theta - d\dot{\theta}^2\sin\theta \quad (45)$$

replacing Eqs. (25) and (26) on Eqs. (44) and (45) respectively, it is easy to see that the terms of the signal control vector

$$u = \begin{bmatrix} \mathcal{T}_R \\ \mathcal{T}_L \end{bmatrix} \quad (46)$$

appear in $z_1^{(2)}$ and $z_2^{(2)}$ obtaining $r_1 = 2$ and $r_2 = 2$, then the right equation system is:

$$z_1^{(2)} = \nu_1 \quad (47)$$
$$z_2^{(2)} = \nu_2 \quad (48)$$

and the left equation system for computing u is:

$$\begin{bmatrix} \mathcal{T}_R \\ \mathcal{T}_L \end{bmatrix} = A^{-1}(x) \begin{bmatrix} \nu_1 + c_0 v_u \dot{\theta}\sin\theta \\ \nu_2 - c_0 v_u \dot{\theta}\cos\theta \end{bmatrix} \quad (49)$$

where

$$A(x) = \begin{bmatrix} c_1\cos\theta - c_2\sin\theta & c_1\cos\theta + c_2\sin\theta \\ c_1\sin\theta + c_2\cos\theta & c_1\sin\theta - c_2\cos\theta \end{bmatrix} \quad (50)$$

with

$$c_0 = \frac{I}{I + Md^2} \quad (51)$$
$$c_1 = \frac{1}{MR} \quad (52)$$
$$c_2 = \frac{Ld}{R(I + Md^2)} \quad (53)$$

The obtained linear system is:

$$\dot{z} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \quad (54)$$

where the state vector is given by

$$z = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \quad (55)$$

and then the output linear equation is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} z \quad (56)$$

The classical state feedback control techniques can be applied to that system, defining a control law as follows:

$$\dot{z} = Az + B\nu \quad (57)$$
$$y = Cz \quad (58)$$
$$\nu = -Kz \quad (59)$$

as a state regulator, but for tracking references with no steady-state errors it can be used a direct loop gain or an integral action approach.

Defining an augmented system:

$$\begin{bmatrix} \dot{z} \\ \dot{e}_i \end{bmatrix} = \begin{bmatrix} A & [0] \\ -C & [0] \end{bmatrix} \begin{bmatrix} z \\ e_i \end{bmatrix} + \begin{bmatrix} B \\ [0] \end{bmatrix} \nu + \begin{bmatrix} [0] \\ R \end{bmatrix} \quad (60)$$
$$\dot{z}_a = A_a z_a + B_a \nu + R_a \quad (61)$$

where e_i is the error accumulation (integral), then $\dot{e}_i$ is simply the error and R is the setpoint or reference, the signal control vector is now defined as:

$$\nu = -K_a z_a \quad (62)$$

where $K_a = \begin{bmatrix} K & -Ki \end{bmatrix}$ with K_i the integral gain .

V. Results

Simulation is carried out for the DDMR moving across a given 2D environment, with a pre-defined path depicted by a set of ordered points, which is taken as succesive setpoints for the controllers; the behavior of the proposed controller is compared with the modified PD controller proposed in [6].

Figure 4 shows the performance of the proposed controller working in the DDMR, while Figure 5 shows the performance of linear PI controller.

Figure 6 shows the behavior of robot in the 2D environmet; as shown, the linear approaches may have wide deviations from desired paths, while the proposed non-linear controller shows a better performance.

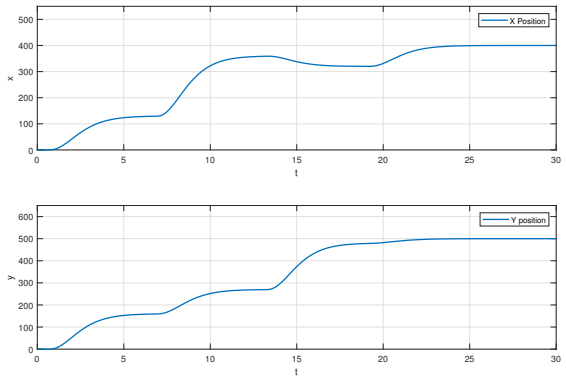


Figure 4. Nonlinear controller performance for the given path

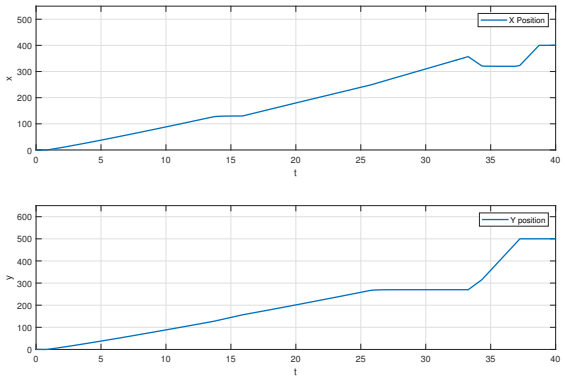


Figure 5. Linear PI controller performance for the given path

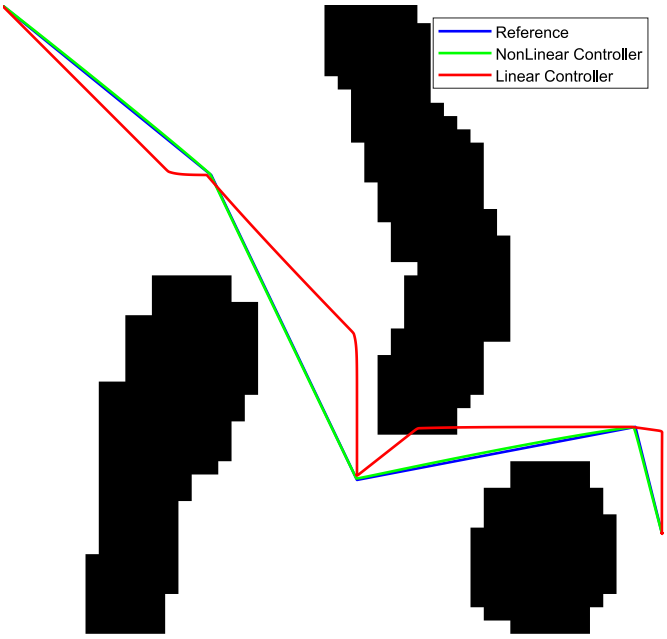


Figure 6. Nonlinear and linear controllers behavior for a given path: $\{(0,0),(130,160),(360,270),(320,480),(400,500)\}$

Tables I, II, III and IV show a numerical analysis of deviations between variables and their references taken from the given point sequence path, as a complement for figures 4 and 5.

Table I
NONLINEAR CONTROLLER x COORDINATE ANALYSIS

Variable	Min. Dev.	Mean Dev.	Max. Dev.
x	0	36.59	230.04

Table II
NONLINEAR CONTROLLER y COORDINATE ANALYSIS

Variable	Min. Dev.	Mean Dev.	Max. Dev.
y	0	31.14	210.27

Table III
PI CONTROLLER x COORDINATE ANALYSIS

Variable	Min. Dev.	Mean Dev.	Max. Dev.
x	0	66.53	230.84

Table IV
PI CONTROLLER y COORDINATE ANALYSIS

Variable	Min. Dev.	Mean Dev.	Max. Dev.
y	0	71.97	210.63

Now, Table V shows the distance between the robot position and the reference into the 2D workspace shown in Figure 6

Table V
DISTANCE ANALYSIS BETWEEN POSITION AND REFERENCE PATH

controller	Min distance	Mean distance	Max idstance
PI	0	53.91	130.86
nonlinear	0	1,53	2,18

CONCLUSIONS

This paper presents a multivariable approach for the feedback exact linearization method applied to control the behavior of a differential driven mobile robot on a given environment, it is compared to the performance of a linear multivariable PI controller. The simulation results show that even when the PI controller, and, in general linear controllers reach references, the trajectory tracked is not exactly the desired one because they attain linear dynamics, and consequently, they show wide deviations between points as shown. The above implies the risk of obstacle collision on a real situation.

In the other hand, the proposed nonlinear controller, shows a much closer trajectory to desired one, not only point to point but between points as a consistent path, the results also show that it needs shorter time to reach the setpoints.

Real systems are, on their majority nonlinear and multivariable, it is then important to develop researh in nonlinear control techniques, and multivariable extensions of that theories, this work is one example of rewarding results it could submit.

ACKNOWLEDGMENTS

This research is supported by "Programa Jóvenes Investigadores e Innovadores por la paz, convocatoria 775-2017" funded by Colciencias and Universidad Tecnológica de Pereira, research project "Diseño de un vehículo terrestre no tripulado para la inactivación de minas anti-persona en el territorio colombiano". The authors also thank to "Maestría en Ingeniería Eléctrica" at Universidad Tecnológica de Pereira for the financial support, and the research group on Automatic Control COL0031472.

REFERENCES

[1] J. J. Craig, *Robotica*. Prentice Hall, 2006.

[2] I. Anvari, "Non-holonomic differential drive mobile robot control & design: Critical dynamics and coupling constraints," Ph.D. dissertation, Arizona State University, 2013.

[3] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework," *Advances in Robotics & Automation*, vol. 2, no. 2, pp. 1–7, 2013.

[4] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[5] A. T. Mathew *et al.*, "Design, simulation and implementation of cascaded path tracking controller for a differential drive mobile robot," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2015, pp. 1085–1090.

[6] D. E. H. Sánchez, J. R. E. Cuenca, C. C. Sánchez, and J. F. R. Cortés, "Diseño, construcción y modelo dinámico de un robot móvil de tracción diferencial aplicado al seguimiento de trayectorias," in *XXIII Congreso internacional anual de la Sociedad Mexicana de Ingeniería Mecánica (SOMIM)*, 2017.

[7] J. Cornejo, J. Magallanes, E. Denegri, and R. Canahuire, "Trajectory tracking control of a differential wheeled mobile robot: a polar coordinates control and lqr comparison," in *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*. IEEE, 2018, pp. 1–4.

[8] O. L. Ramírez-Martínez, E. A. Martínez-Garcia, R. E. Mohan, and J. K. Sheba, "Mobile robot adaptive trajectory control: Non-linear path model inverse transformation for model reference," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, Dec 2014, pp. 877–881.

[9] R. Kumar, A. Patel, and S. Purwar, "An adaptive control technique for trajectory tracking of a mobile robot using synchronization method," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, July 2017, pp. 344–348.

[10] N. Chinthaned and P. Sanposh, "Robust geometric control of a two-tank system," in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, June 2016, pp. 1–4.

[11] A. J. Krener, A. Isidori, and W. Respondek, "Partial and robust linearization by feedback," in *The 22nd IEEE Conference on Decision and Control*, Dec 1983, pp. 126–130.

[12] A. Isidori, *Nonlinear control systems*. Springer Science & Business Media, 2013.

[13] M. R. Kankashvar, H. Kharrati, and A. Khorami, "Design of multivariable controller based on feedback linearization for five-bar linkage manipulator," in *2015 23rd Iranian Conference on Electrical Engineering*, May 2015, pp. 916–921.

[14] L. Meihui, D. Shangfeng, C. Lijun, and H. Yaofeng, "Greenhouse multivariables control by using feedback linearization decoupling method," in *2017 Chinese Automation Congress (CAC)*, Oct 2017, pp. 604–608.

[15] Y. Zhang, G. Tao, and M. Chen, "Relative degrees and adaptive feedback linearization control of t–s fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2215–2230, Dec 2015.

[16] J. C. Montesdeoca, M. C. P. Santos, M. Monllor, and D. Herrera, "Trajectory tracking controller for differential-drive mobile robots," in *2017 XVII Workshop on Information Processing and Control (RPIC)*, Sep. 2017, pp. 1–4.

[17] D. Diaz and R. Kelly, "On modeling and position tracking control of the generalized differential driven wheeled mobile robot," in *2016 IEEE International Conference on Automatica (ICA-ACCA)*, Oct 2016, pp. 1–6.

[18] W. Meiling, W. Zhen, Y. Yi, and F. Mengyin, "Model predictive control for ugv trajectory tracking based on dynamic model," in *2016 IEEE International Conference on Information and Automation (ICIA)*, Aug 2016, pp. 1676–1681.