NATIONAL DISTANCE UNIVERSITY

Artificial Intelligence Department

Master Thesis

CIRCLE ESTIMATION VIA THE KALMAN FILTER AND SLAM TECHNIQUES

GUILLERMO PEIRO AZNAR

Directed by: JOSÉ MANUEL CUADRA TRONCOSO

Year: 2018-2019

CIRCLE ESTIMATION VIA THE KALMAN FILTER AND SLAM TECHNIQUES

Master Thesis

Escrito por: Guillermo Peiro Aznar

Dirigido por: José Manuel Cuadra Troncoso

Tribunal calificador

Presidente: D/Dª.

Secretario: D/Dª.

Vocal: D/Dª.

Fecha de lectura y defensa:

Calificación:

Acknowledgment

El primer agraïment és per a ma mare. Sense la seua ajuda no haguera pogut estudiar durant tots aquests anys. També gràcies a Sasha per entendre que el pare no pot jugar, quan cal que estudie. A més a més, agrair als meus germans i al meu pare per sempre donar-li un poc d'humor aquestes situacions tan estressants. Finally, I want to thank my wife for helping me along the way.

Thank you, gràcies, spasibo.

Abstract

The exploration for better navigational algorithms that allow robots to move in the environment is one of the main fields of research in Robotics. In the literature, many innovations have been proposed in order to improve the performance of these navigational methods. The complexity associated makes the research both exciting and challenging. One crucial step in the design of a proper navigational model is the recognition of the objects that compose the surroundings of the robot. In order to obtain a precise representation, the robot relies on sensors. The data obtained from the laser rangefinder comes in the form of segments that rarely represent the whole object. Furthermore, this data has a particular noise that does not allow to apply the basic geometrical techniques for shape estimation.

Circular objects, such as columns or trees, are persistent in many of these situations. Following the research done by Cuadra Troncoso (2011) and Muñoz Bañón (2016), we use the Kalman Filter in order to acquire a better approximation of the parameters that represent the circle. The Kalman Filter is a widely used technique for estimating parameters in dynamic systems. Despite the many examples found in the literature, research on the topic of circle estimation is not abundant.

In this paper, we propose a model for estimating circles with the Extended Kalman Filter. Through experimentation, we have adapted this model to improve the results. Moreover, we take a step forward and apply the technique designed for circle estimation to a more practical problem. Using Simultaneous Localization and Mapping techniques, we benefit from the Kalman Filter to locate a moving robot and create a map composed of different circular and linear objects.

Resumen

El estudio para encontrar algoritmos de navegación, que permitan a los robots moverse en el entorno, es uno de los campos de investigación más importantes de la Robótica. Muchas innovaciones han sido propuestas con el fin de mejorar el rendimiento de estos métodos. La complejidad asociada hace que esta investigación sea, al mismo tiempo, difícil e interesante.

Un aspecto muy importante en el diseño de un modelo de navegación es el reconocimiento de los objetos de su alrededor. El robot obtiene la información necesaria para la representación de dichos objetos, a través de los sensores. Los datos obtenidos de estos sensores, como por ejemplo un láser, suelen venir en forma de segmentos que no representan todo el objeto al que pertenecen. Además, estos datos lleven un error asociado que impiden aplicar métodos geométricos para la estimación.

Lo objetos circulares, como por ejemplo columnas o árboles, son muy comunes en estas situaciones. Continuando con el trabajo realizado por Cuadra Troncoso (2011) y Muñoz Bañón (2016), haremos uso del Filtro de Kalman para obtener una aproximación de los parámetros que forman el modelo de un círculo. A pesar de los numerosos ejemplos de aplicación del Filtro de Kalman en la literatura, no se encuentran muchas referencias relacionadas con los círculos. En este trabajo, propondremos un modelo de aproximación de círculos con el Filtro de Kalman Extendido. Hemos adaptado este modelo para mejorar los resultados en la experimentación. Asimismo, aplicaremos este modelo a un problema más práctico. A través de técnicas de localización y mapeo, usaremos el Filtro de Kalman para localizar un robot en movimiento y crear un mapa compuesto por objetos lineales y circulares.

Contents

Nomenclature

AI Artificial Intelligence, page 1

AR Autonomous Robotics, page 1

EKF Extended Kalman Filter, page 23

GPS Global Positioning System, page 14

MSE Mean Squared Error, page 20

SLAM Simultaneous Localization and Mapping, page 13

List of Figures

List of tables

Chapter 1

Introduction

1.1. Contextualization

In the last years, we have been able to see how the use of new technologies has become a more crucial part of our life. Everywhere we look, we see that we are connected and dependant on a different type of computers that can ease our life in many ways. From both software and hardware perspectives, aspects, which theoretic part used to be hidden, have been brought to light. Algorithms that were defined in the early days of the Artificial Intelligence (AI) are now beginning to be studied with more interest in many computer science faculties. There is a big necessity of improving these algorithms, as well as being able to integrate them into different devices in order to allow us to get away from repetitive tasks, that can be easily performed by a computer, for instance, house chores, such as cleaning, cooking, or doing the laundry.

Among all the significant areas of the AI, that have been of matter in these last years, Autonomous Robotics (AR) have produced a considerable impact on both industrial production and everyday house life. As the International Federation of Robotics (IFR) points out, the consume of industrial robot reached a record in 2017, and it keeps increasing by an average of 14 per cent per year [IFR (2017)]. As it is presented in figure 1.1, the production in different industries is on Robotics will continue to increase. This increment might have an unfavourable impact on the tasks carried out by workers, who might be replaced by autonomous machines, but this is not considered as the focus of this paper.

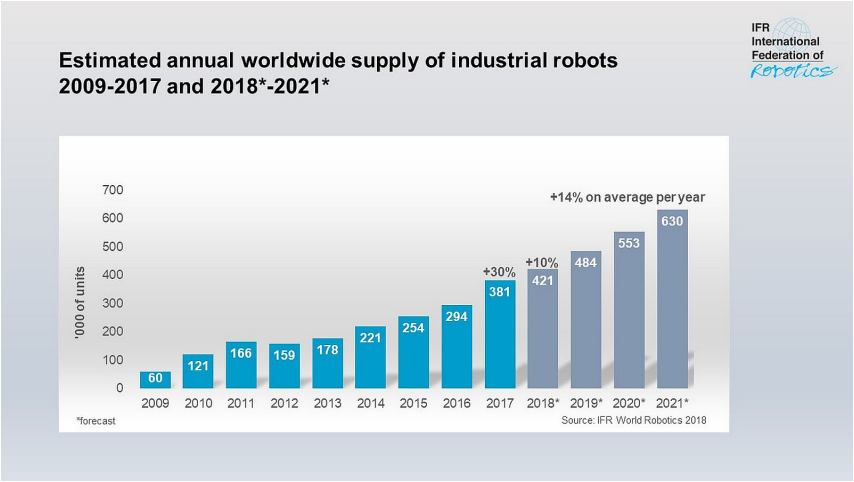

Figure 1.1: Estimated annual supply of industrial robots. (IFR World Robotics, 2018).

The significant growth of the usage of robots has been noticed not only in large scale industries but also in the everyday routine life, for instance, robot vacuum cleaners or robotic lawnmowers. Accordingly, it is notable that the trust put by people on those machines has grown significantly. In the figure 1.2 by [Tractica (2016)], it is shown how the Robotics have been entering a vast market. People trust these machines in a wide range from toys to household. The figure also shows the expected boost in the consumes of such technologies in the coming years.

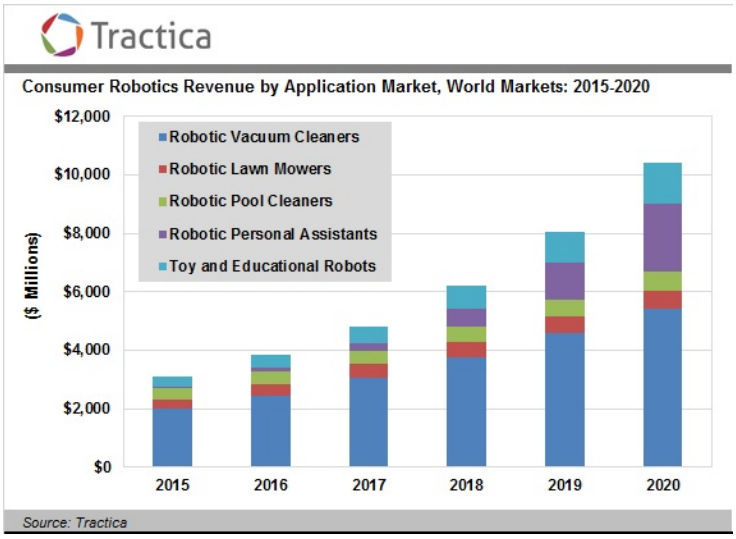

Figure 1.2: Costumers revenue by Application Market. (Tractica, 2016).

As it is mentioned in the article, despite many new emergent companies are producing such robots for home usage, a general distrust on these machines, which are supposed to ease

the life of the people, still prevails. According to the author of the article, it is due to two main reasons: the prices set upon the robots and the general apprehension people may have to the real effectiveness of these house gadgets. For this reason, the study of this area has to be considered with a significant distinction, in particular, find better resources and techniques to gain the people confidence, along with the opportunity to create something that one day might be understood as one of the most important applied sciences there is [Boesl (2016)]. Considering the changes in the market related to Robotics, in this paper, we centre on Computer Science and Mathematics perspective. Hence, in the next section, an introduction is presented, together with the goals and motivation of this project.

1.2. Motivation and Goals

In this project we aim to gather the information extracted from the research performed by [Muñoz Bañón (2016)] and [Cuadra Troncoso (2011)] on segment estimation. In his thesis, Cuadra Troncoso (2011) describes an approximation for estimating linear segments using the Kalman Filter. The success of this research brings improvements in the navigation algorithms that require to possess geometrical information about the surrounding objects [Cuadra Troncoso et al. (2015)]. With this goal in mind, Muñoz Bañón (2016) proposes an extension of the design of the Kalman Filter, but this time for circular segments. In real life experiments, the circular segments are prevalent; in many situations, the presence of objects such as trees or columns is usual. By estimating the circumference that represents the object, we can improve the navigation and mapping techniques.

In this project, we want to provide a continuation of what started in both cited projects. The project aims to research for a better approach to estimating the circular segments. Furthermore, we take a step forward and analyze the behaviour of this algorithm in a SLAM based approach. With this respect, the general goals of the project can be enumerated as:

1. To provide a better initialization method for the Kalman Filter. In work by [Muñoz Bañón (2016)], the conclusion is extracted that the initial estimation influences the results of the Kalman Filter. Therefore, a better initialization method is required.

2. Study and try to provide a more stable Extended Kalman Filter. Many aspects affect the circle fitting process. Although on many occasions, the results obtained match the expectations; in many others, they are not precise. A study of why this happens and how to solved it must be fulfilled.

3. Whether or not better results will be accomplished. This project aims to test the circle estimation in the SLAM problem. Thus, from the one side that a localization method must be provided. Also, from the other side, a mapping approach must be designed. In this case, we will concentrate on mapping scenes with circular objects, see chapter 5.

The goals described above can be seen as easy from a naive perspective. Nevertheless, the estimation of geometrical figures, specifically circular ones, prove itself to be complex. Not only a good mathematical knowledge is required, but also a large time window of experimentation and analysis of the design is needed. Figure 1.3 shows a scheme of the process that we will follow in this project. The overall goal of the project is to do small steps into reality by hardening the estimation process one bit at a time.

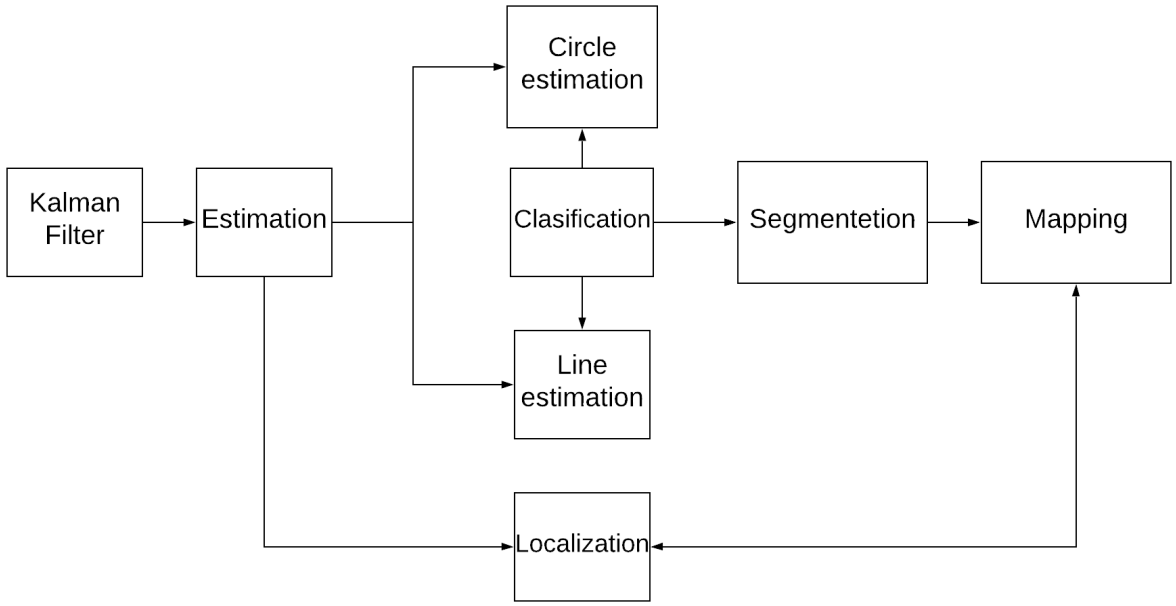

Figure 1.3: Project goals and scheme. The scheme describes the different steps taken during the project.

1.3. Report structure

This report is organized in different chapters structured to provide a progressive and incremental view on the knowledge of the project matters:

1. Introduction and contextualization

2. Theoretical Foundations
3. State of the art
4. Circle fitting
5. SLAM: localization and mapping
6. Conclusions

The introductory chapter provides a brief contextualization, along with a little explanation of the goals of the project. In the second chapter, a deeper explanation of theoretical concepts related to Robotics is given, in pursuance of introducing a bit more the reader into the topic of the experiment. After that, we deliver some mathematical definitions and introduce the Kalman Filter. Later on, in the third chapter, we add up the research of the circular segments estimation by [Muñoz Bañón (2016)]. The fourth chapter will propose an innovation of the Extended Kalman Filter and a new initialization method. The fifth chapter is dedicated to the SLAM process divided into two phases localization and mapping, centred in circular segments. To conclude, in chapter 6, we will analyze what we have accomplished and what is yet to do.

Chapter 2

Theoretical Foundations

In the previous chapter, we have talked about the influence of robotics on people's lives in the present days. In this chapter, we present a more in-depth look into the concepts that are the bottom line of this project. At first, a brief general understanding of the robotic paradigms is given, followed by an introduction of *SLAM* and some mapping techniques. Finally, at the end of the chapter, mathematical aspects, that are crucial for a proper understanding of the implementation and definition of the domain of the current experiment, are presented.

2.1. Artificial Intelligence and Autonomous Robotics

The first thing that comes to people's mind when talking about AI and, more specifically, about robots, is perhaps the pop culture, created in the movies from the '70s until our days. The robots have been displayed as anthropomorphic machines, usually assigned to a repetitive task, for instance, a robot which serves in a restaurant and behaves and moves like a real waiter. This conception could explain the expectations pointed in this direction on the first days of Artificial Intelligence. However, having gained more experience and solved more problems of the development of 'thinking' computer systems, engineers and scientist construed a more realistic description of the robots.

The epistemology of the word 'Robot' brings us to Prague in the early 20th century, when the writer Karel Capek created a play in which he introduced machines that are working for the owner. More specifically, the word is related to the Slavic word *Rabot* that means "to work" [Etimology (2012)]. Furthermore, in this definition, we eventually encounter ourselves

facing the first valid affirmation about what the Robot is supposed to be.

> "Robot – a machine that can perform a complicated series of tasks automatically, usually programmable by a computer." [Oxford Dictionary (2019)]

The interpretation given by the dictionary turns out to be more exact, and there is no reference to any human biological aspect. Following the history of IA and Robotics, one can comprehend how the reality has grown bigger in the eyes of the scientists, for instance, in some situations the robots shall adopt a certain morphology in order to perform the task they have been created for more effectively. Referring again to a robot vacuum cleaner, which certainly does not look like a janitor, rather resembling a normal human-operated vacuum, for the reason that it has to be able to reach places in the house that require a small size.

In this perspective, the definition of a robot can take another step forward. The way of controlling a robot, or how a robot can carry out its assigned task and reach its goals, make us differentiate between two types of robots. On the one hand, we have robots that work with human interaction, which can be done, for example, via teleoperation, and on the other hand, we have autonomous robots which are supposed to be able to accomplish certain goals by themselves. According to [Murphy (2018)], a more formal description of an Autonomous Robot would be:

> "An intelligent robot, or autonomous, is a mechanical creature which can function individually. *Intelligent* meaning that it is not doing the things mindlessly, thus in a repetitive way."

The definition above is how we understand a robot in this project. The implication that carries giving the word *intelligent* to a machine brings with it a great deal of concern of how our mind works, and that is one of the main questions that Artificial Intelligence must answer, step by step.

Another fundamental concept, explained in more details in section 2.2, are the Paradigms of Robotics. Whenever we are about to enter into the development phase of a robotics-related project, we must inevitably consider the three paradigms described and improved from the first days of the AI till now. As it happens with any other engineering task, the concept of design plays a significant role and paves us the way to a more consistent project, that is adjusted to general conventions. Hence, specifying generic concepts and then specializing

in the topic of the project seems the best way to procure a good understanding and perhaps lead to easier maintainability.

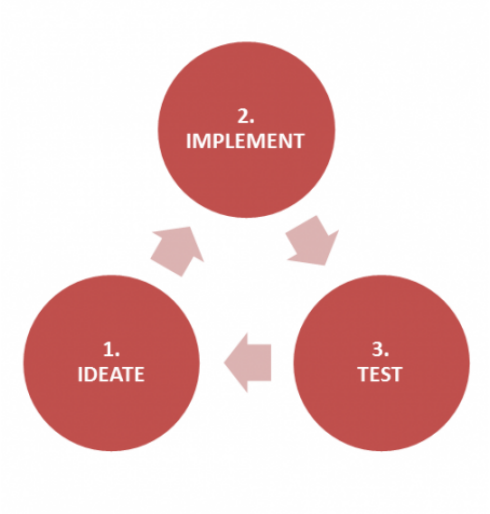

Figure 2.1: Desired process of robotic problems development.

Besides what the concept of the implementation and design might give from other experiences, the field of robotics is also based on creativity, in the way that for solving specific tasks the engineer has to be able to imagine different solutions and then test them [VexRobotics (2019)]. The figure 2.1 illustrates what can be understood as a think-and-try way of developing a process. This creativity depends considerably on the mathematical knowledge the scientist has and how he or she can turn it into a real computational solution.

2.2. Robotic paradigm

There exist many fields in Artificial Intelligence. In this paper, among others, a problem related to the robots and their interaction with the world around is discussed. It is essential then to make a comparison on the different perspective that has been developed through the history of the AI, and it is displayed in the **Robotic Paradigm**.

In the early days of research on how intelligence relates to computers, many assumptions were made. Moreover, there were particular expectations on how computers could one day complete the tasks that humans perform every day without difficulty. Theses expectations became more realistic with years of study when the researchers discovered how complex could

it be, for example, to make a robot wander in a room without hitting any obstacle. Deeds that from a naive perspective could seem easy to achieve can be difficult to apply with the mathematical laws and concepts, which work in the computational world. This dualism has been since then one of the most decisive points of research in the field of Robotics [Murphy (2018)].

The scientists and engineers who decide to take over a robot developing project, more specifically, an autonomous robot, must consider aspects that are related to both mathematics and biology. In particular, a deeper understanding of how an animal accomplishes a simple task from a third-person point of view can improve the development of the indicated task with a robot. A representative example of this is given in [Cuadra Troncoso (2011)] with the navigation based on area centres. The robot calculates the centre of the areas among the detected obstacles and navigates towards it. This method has a more natural approximation to how a human being walks through a room, focusing more on the empty spaces rather than trying to avoid the objects. It results in smoother and more natural navigation. Many other examples can be illustrative when studying a specific robotic problem. However, there have been generalizations that allow engineers and scientists to find common ground.

The Robotic Paradigms are related to how the robot interacts with the world around it. Thus, three main primitives are defined: Sense, Plan and Act. The order in which these primitives execute defines the control architecture of the project, and in consequence, the paradigm within it belongs [Faigl (2015)]. We define three paradigms, namely, hierarchical paradigm, reactive paradigm and hybrid paradigm. A more detailed explanation follows.

2.2.1. Hierarchical paradigm

The first paradigm is the **Hierarchical paradigm**. In this deliberative architecture, the robot senses first, then plans and then acts.

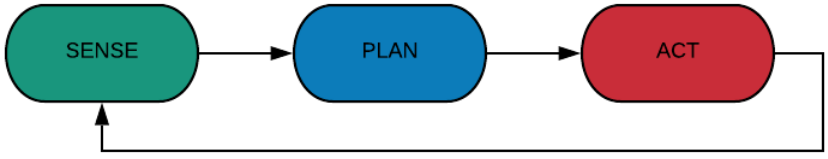

Figure 2.2: Order of the main primitives in the hierarchical paradigm.

The figure 2.2 shows how interaction with the world occurs from a hierarchical perspective. First of all, a sensing phase determinate the surroundings of the robot. After this, a plan is made considering what has been sensed. Lastly, an actuation takes place depending on the plan made in the previous phase. This paradigm follows the Closed World assumption. The model of the world provided to the robot includes everything that it needs to know.

The sensing phase is characterized by different calculations and the construction of a world model that must be accurate in order to elaborate a solid plan for the next phase, which leads to two main disadvantages of the paradigm. Firstly, the sensing and planning phases are usually lengthy and require high performance of the hardware of the robot. Secondly, the closed world assumption can lead to unwanted situations, for example, when in the *"real world"* the fundamental situation changes, and the initially considered plan has no sense for a further step in the experiment. For instance, a new moving obstacle, such as a person or a pet, could enter a room, initiating a whole new sensing process.

Regardless of its disadvantages, various important projects have been implemented using this architecture. For example, a robot *Shakey*, based on this deliberative perspective, was designed and built at the Stanford Research Institute between the years 1966 and 1972 [rob (2015)]. Shakey can be considered the world's first intelligent robot. The project took a big step forward in its time because it applied theoretical algorithms, such as A* algorithm, and techniques from the early days of the AI, such as **STRIPS** [str (2019)]. It was named Shackey because of the shaking movement it used to do during the planning process, induced by an alteration of the sensors as a result of finding a new obstacle.

Nowadays, hardware performance has increased considerably, which could solve the main problems of the hierarchical paradigm, but the problem of framing the world persists, concerning the time and the computational resources it requires. Thus, to create an accurate world model, representation is still a challenge in almost every task associated with robotics.

2.2.2. Reactive paradigm

The need for a faster connection between the robot sensors and actuators caused the definition of a new perspective, the **Reactive paradigm**. It takes inspiration from the biological world. For example, an impulse received through the eyes activates an immediate muscular response, meaning that there is no planing in between the sense and the act primitives.

The figure 2.3 shows that now we abandon the closed world assumption. In the

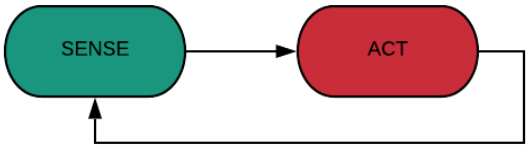

Figure 2.3: Order of the main primitives in the reactive paradigm.

reactive method, the unexpected inputs from the surrounding world are accepted and controlled by a particular behaviour which must be predefined [Arkin (1998)]. In some sense, it can be understood as a deterministic final state automaton, when the initial state automatically executes the next step.

The reactive paradigm is inspired by the fact that some insects and other simple animals exhibit intelligent behaviour virtually without brain [Faigl (2015)]. Furthermore, there still exists the framing problem, that is, in which way to define the mapping and recognition of the surrounding world in an appropriate computational way.

2.2.3. The hybrid paradigm

As it is mentioned in the previous sections, both hierarchical and reactive paradigms specify the strict setting of the sense, plan and act primitives, which does not work in numerous situations. On the one hand, the amount of time needed for sensing and planing in the hierarchical paradigm supposes a risk and does not match the expectations of many domains. On the other hand, the reactive paradigm omits the necessary planning that is attached to almost any experiment, and hence, its limitations are apparent. For these reasons, a new paradigm, namely, the **Hybrid paradigm**, is defined. It gathers the positive aspects of the two formerly defined paradigms. Hence, a new organization of the three main primitives emerges.

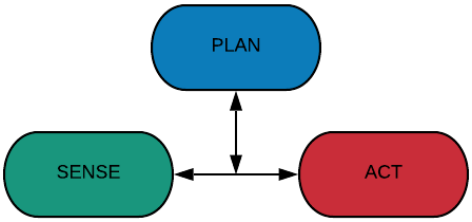

Figure 2.4: Order of the main primitives in the hybrid paradigm.

Figure 2.4 illustrates the insertion of the planning primitive in the reactive approach, represented in the figure 2.3. The innovation comes in the first stage of planning using a global world model. The next phase is a pair of sense-act reactive control, which executes the instructions to accomplish the goal. In this respect, the problem of not having information and planning stages in the reactive paradigm solves by the introduction of the planing primitive. The information acquired in this phase is beneficial for the rest of the operation.

Even though the planning phase stills requires a considerable time window, the real improvement of the hierarchical paradigm comes by not having to plan every instruction of the goal every time a new, unexpected element enters the world. Thus, the goal can be achieved in less time and, most likely, using fewer resources. Moreover, the different events that may occur during the experiment can be understood as subtasks and solved with the minimal interference on the main assignment. The last has big implications from a computational point of view, since solving different situations, such as dead reckoning or obstacle-avoidance, can be implemented and executed in different threats, allowing better performance and resource preservation.

Presently, most of the robot-related projects use the hybrid paradigm or a customized version of it.

2.3. SLAM

In robotics, Simultaneous Localization and Mapping (SLAM) is an essential area of study. It comprises constructing a map of the surroundings of a robot without having a definite knowledge about its position or the position of the surrounding objects. In the specialized literature, it is defined as the chicken-egg problem, since the localization of the features requires the position of the agent, while the localization of landmarks can be of great significance for a more exact calculation of the robot position.

The problem of localizing the robot is named Odometry. For legged or wheeled robots, a wheel slippage or an error in the gyroscope of certain servo may complicate the task of predicting the disposition. Hence, the question "Where am I?" is answered from a functional point of view rather than a perfectionist one. That is because aiming to construct a perfect map is usually an unfeasible task due to many erroneously involved components existing even in the simple experiments [Siegwart et al. (2011)].

The localization calculations could be solved easily by attaching a Global Positioning

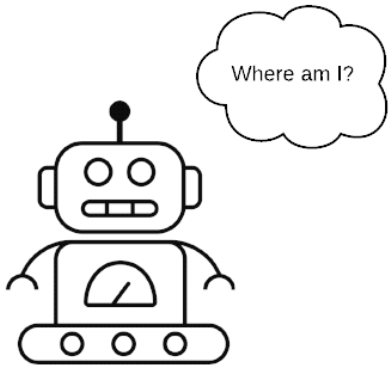

Figure 2.5: "Where am I?" [Pixabay (2018)]

System (GPS) to the agent. Even though the localization provided by such a device might still be faulty, eventually it is reasonably precise and is used in substantial projects related to aviation and autonomous cars.

Despite the fact of the existence of the global position system that gives us a position of reference within the world, two shortcomings may arise.

- Firstly, many experiments are executed in an indoor environment, meaning the information from the GPS might not be available.
- Secondly, and more importantly, is that in SLAM, the global position is just a component of the localization step. Moreover, the relative position to a certain object or person is required.

As a result of the need of the relative position, the robot must use its sensors to calculate the position of the surrounding objects. When relying on the sensors and the effectors to perform a certain task in an unknown world, three main problems, specified in the following, arise:

Noisy sensors - The information, provided by a sensor, for example, a laser rangefinder, is usually affected by many environmental aspects, such as lighting, building materials, reflection or others. Although the errors are present in all the existent hardware, the development of prediction, filtering and smoothing techniques may help to reduce the interference of different noises.

Sensor aliasing - The sensor aliasing problem usually occurs when the information provided from a particular sensor is not representative of an individual situation. For instance, when a navigating across a dark room, a robot can receive the perception of not moving because the different light signals are indistinguishable from each other. The fact that two or more signals can not be differentiated increases the uncertainty of the surroundings. This problem is partially solved by techniques such as **sensor fusion**. Thus, combining the events extracted from different sensors may provide a way to differentiate the overall situation of the agent.

Effector noise - When giving *orders* to the robot, there always exists a certain noise associated with the performance of the effector. It provokes, for example, that an expected new position of the robotic arm might not be the obtained one. The inaccurate outcomes occurred in calculations, produced by both sensor and effector noise, are noticeable in **Odometry**.

2.3.1. Types of SLAM

In the literature, we can differentiate two types of SLAM, namely, offline and online.

- In the online SLAM, the map the robot builds of the world is created simultaneously with the navigational process.
- In the offline SLAM, the mapping process is performed afterwards. The information collected by the sensors along the navigation process, is analyzed and transformed into the desired map.

In our case, the SLAM procedure chosen to represent the map is the offline method. Thus, first we gather information and then we create the surrounding map.

2.3.2. Odometry

The word "Odometry" references the calculations done on the data provided by the sensors of the robot in order to obtain an estimation of the robot position [Shen et al. (2011)]. The considered sensors are, among others, range sensors and wheel encoders. Aspects, such as the diameters of a wheel, velocity, and range of gyroscope determinate the values of a particular data state that gives information about the robot position. However, this information is subject

to specific errors and imperfections associated with the specifications of the robot. Therefore, it is crucial to have a good insight of the agent being used, which means, gather all the information provided by the manufacturers about the robot itself and the expected error of the sensors.

In real life experiments, each robot is different due to imperfections that may appear during the usage.

It is evident from all mentioned above that Odometry is not an easy task. It requires competence in mathematical modelling of the relation between the effectors and the actual movement of the robot, as well as the elimination of possible outliers present in the data that can alter the perception and results of the model entirely. It becomes even more complicated when the data from the sensors about the surroundings is very noisy or does not exist at all. Such calculation of the position of the robot, considering only the velocity and angle of bearing from the wheels is named **dead-reckoning**.

2.4. Mathematical aspects

The previous section. In this section, some of the mathematical procedures related to estimation are presented. Later on, the parameters and modules of these approaches will be adapted for the specific problem of this project.

2.4.1. Least squares regression

The most straightforward way of fitting an equation given specific points is the **Least Square Regression**. This procedure allows us to fit an equation to given points by minimizing the geometrical position between them and the possible solution. In a more formal definition, the least square regression tries to adjust the parameters of a specific function in order to fit the data set best [Menon (2018)]. The following definitions are to clarify some notions related to this topic.

Definition 1. Residual. The residual is the subtract between the actual value of the function f and the values predicted by the model. The residual obtained in a certain domain represents the fitting accuracy of a model.

$$e_i = \overline{y}_i - f(x_i, \beta). \tag{2.1}$$

where y is the dependent variable variable, x is the dependent one.

The least square method will try to find the best parameters for the fitting equation by minimizing the square of the residual.

$$S = \sum_{i=1}^{n} {e_i}^2. \tag{2.2}$$

A good basic example of this is the linear approximation. Giving certain points described in the Cartesian coordinates, as shown in figure 2.6, it is relatively easy to draw a line "by eye" that represents the values of the predicted function that fits the model.

Definition 2. Straight Line equation. The straight-line equation is given by the product of the slope (determined by (m)) with the change in x, plus the value of y when x is zero (determined by (b)).

$$y = mx + b \tag{2.3}$$

As we can observe, having two points (x_1, y_1) and (x_2, y_2), we can obtain the values of b and m from the equation 2.3.

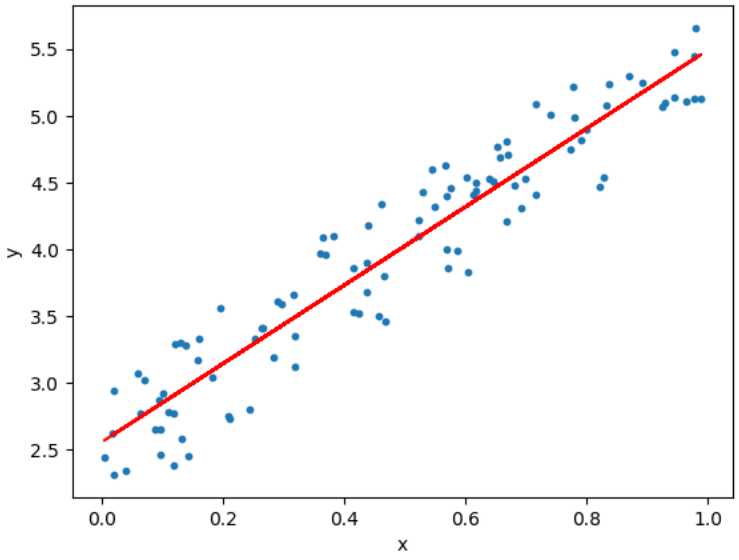

Figure 2.6: Linear least squares. The dots are the data and the output is the line fitted by the model.

The figure 2.6 shows particular points from which we fit the line equation 2.3. By using the least square method, the residual in this case is defined as the difference between

the left- and right-hand sides of the points applied to the equation of the line $y = mx + b$, being m and b the parameters of the model. Thus, the sum S from equation 2.2, applied to the points (x_1, y_1) and (x_2, y_2) is:

$$S(m, b) = [y_1 - (x_1 * m + b)]^2 + [y_2 - (x_2 * m + b)]^2 \tag{2.4}$$

The next step is to minimize the equation by calculating the partial derivatives $\frac{\partial S}{\partial m}$ and $\frac{\partial S}{\partial b}$ and setting them to zero. Obtaining a system of two equations with two unknowns (namely, m and b), parameters which define the estimated line.

Despite being simple, the least square method is widely used in many data science projects. Also, it is easy to generalize to other models as we will see in section 4.4.2, with circle fitting applying a least square method.

2.4.2. The Kalman Filter

Among all the approximation methods used in robotics and data science, the **Kalman filter** is one of the most relevant. As it is explained in [Ogorek (2019)], there have been many papers related to this topic that go from a superficial explanation of what the Kalman filter is to a more abstract in-depth mathematical description. In the following chapter, we present the fundamental though thorough ideas of the Kalman filter in order to provide a better comprehension of the topic. Since the groundwork of this report is based on the Kalman filter and its extended version, the notions defined in the following are essential.

To be able to proceed with the Kalman approach, we need to clarify some concepts [Lacy and Thacke (1998)]. First of all, we introduce the notion of the **mean square error** associated with a signal:

$$y_i = a_i x_i + n_i \tag{2.5}$$

where y_i is the observed signal, a_i is a gain term, x_i is the information bearing signal, and n_i is the noise.

The goal of this process is to estimate x_i. We define an **error function** by the difference between $\hat{x}_i$, which represents the estimation and x_i:

$$f(e_i) = f(\hat{x}_i - x_i) \tag{2.6}$$

In the following, we introduce some definitions to simplify the further understanding of the concepts.

Definition 3. Monotonic Function. A monotonic function is a function which is either entirely non-increasing or entirely non-decreasing. In other words, a function is monotonic if its first derivative (which does need to be continuous) does not change its sign [Stover (1999)].

We emphasise that the error function should be both positive and monotonic. The square of the function $f(e_i) = f(\hat{x}_i - x_i)^2$ follows the constraint.

Definition 4. Mean Squared Error (MSE). The mean squared error is the sum of squared errors divided by number (n) of predictions:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2.7}$$

The Kalman filter is used to estimate values of a certain variable in a specific system influenced by noise. First, a state vector $\mathbf{x}$ of the process is defined;this vector represents the value of the model at a certain step i [Rlabbe]. The next element necessary for defining the filter is the transition model, which represents the evolution of the state vector $\mathbf{x}$ throughout the process:

$$\mathbf{x}_{i+1} = \mathbf{F} \cdot \mathbf{x}_i + \mathbf{w}_i \tag{2.8}$$

where $\mathbf{F}$ is a state transition matrix, and $\mathbf{w}$ is the associated noise.

Following with the description, we define a measurement vector $\mathbf{z_t}$ that gathers the values obtained at a certain time. The basis concept of the Kalman Filter is the combination between the theoretical model and the observations. The connection between the measurement and the state vector is provided by a relation $\mathbf{H}$, added a certain typical measurement error $\mathbf{v}$:

$$\mathbf{z}_i = \mathbf{H} \cdot \mathbf{x}_i + \mathbf{v}_i \tag{2.9}$$

The Kalman filter intends to minimize the mean squared error associated with both the measurement and to the transition of state vectors $\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}$. For modelling such errors, it is necessary to use Gaussian distributions.

$$\mathrm{Q}_i : \mathbf{w}_i \sim \mathcal{N}(0, \mathbf{Q}_i) \tag{2.10}$$

$$\mathrm{R}_i : \mathbf{v}_i \sim \mathcal{N}(0, \mathbf{R}_i) \tag{2.11}$$

In the Kalman filter, the measurement noise R is supposed to be a Gaussian *white noise* [Wiki (2019)]. However, many dynamical filters do not follow this constraint.

The next essential element for describing the Kalman Filter is the covariance matrix $\mathbf{P}$. By minimizing this matrix, the filter accomplishes the estimation of the parameters. The covariance matrix can be expressed as:

$$\mathbf{P}_i = E\left[(\mathbf{x}_i - \hat{\mathbf{x}}_i)(\mathbf{x}_i - \hat{\mathbf{x}}_i)^T\right] \tag{2.12}$$

As explained before, the Kalman Filter requires to minimize the MSE. In order to achieve this we need a coefficient that minimizes the the trace of the covariance matrix. Thus, we define the coefficient K, namely the Kalman Gain:

$$\mathbf{K}_i = \mathbf{P}_{i-1}\mathbf{H}_i^\top \mathbf{S}_i^{-1} \tag{2.13}$$

The Kalman Gain works as a weighting for the regressor with the smallest error.

The last essential element to define is the control input $\mathbf{u}$. The process takes advantage of known control inputs, in order to improve the prediction of the next state of vector $\mathbf{x}$.

The Kalman filter is a recursive method based on two phases: predicting the next state of the system 2.8 and the update phase that improves the current estimate vector with a control input. A more detailed explanation of each step follows.

- The **Predict step** uses the state estimated vector of the previous step $i-1$ to produce an estimation on the current step i:

$$\hat{\mathbf{x}}_i = \mathbf{F}_i\hat{\mathbf{x}}_{i-1} + \mathbf{B}_i\mathbf{u}_i \tag{2.14}$$

The equation 2.14 is derived from the equation 2.8, adding the matrix $\mathbf{B}$, that is applied to the control input $\mathbf{u}$. The last will improve the prediction of the state estimator.

The error covariance is also predicted as follows:

$$\mathbf{P}_i = \mathbf{F}_i \mathbf{P}_{i-1} \mathbf{F}_i^\top + \mathbf{Q}_i \tag{2.15}$$

- The **Update step** produces both posterior values of the state estimated vector and the covariance matrix by using what is called the Kalman gain K. For this step, the following procedure is applied:

  1. Firstly, the measurement residual $\mathbf{y}_i$ is calculated as:

     $$\tilde{\mathbf{y}}_i = \mathbf{z}_i - \mathbf{H}_i \hat{\mathbf{x}}_{i-1} \tag{2.16}$$

     In this step we subtract from the measurement $\mathbf{z}_i$ the expected value $\mathbf{H}_i \hat{\mathbf{x}}_{i|i-1}$. In other words, we calculate the fit of the function, or how certain the predicted value is.

  2. Secondly, the innovation covariance $\mathbf{S}_i$ is obtained as follows:

     $$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i-1} \mathbf{H}_i^\top + \mathbf{R}_i \tag{2.17}$$

  3. In the next step, we calculate the Kalman gain $\mathbf{K}_i$. There are different approaches to calculate what will determinate the change of the state estimate vector, and after all, the fit of the model. The following one expresses the generic form of the Kalman gain:

     $$\mathbf{K}_i = \mathbf{P}_{i-1} \mathbf{H}_i^\top \mathbf{S}_i^{-1} \tag{2.18}$$

  4. Having the Kalman gain $\mathbf{K}_i$ and the fit of the model $\mathbf{y}_i$, we can update the state estimate vector:

     $$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{i-1} + \mathbf{K}_i \tilde{\mathbf{y}}_i \tag{2.19}$$

  5. Finally, we update the error covariance matrix $\mathbf{P}_i$:

     $$\mathbf{P}_i = (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i) \, \mathbf{P}_{i-1} \tag{2.20}$$

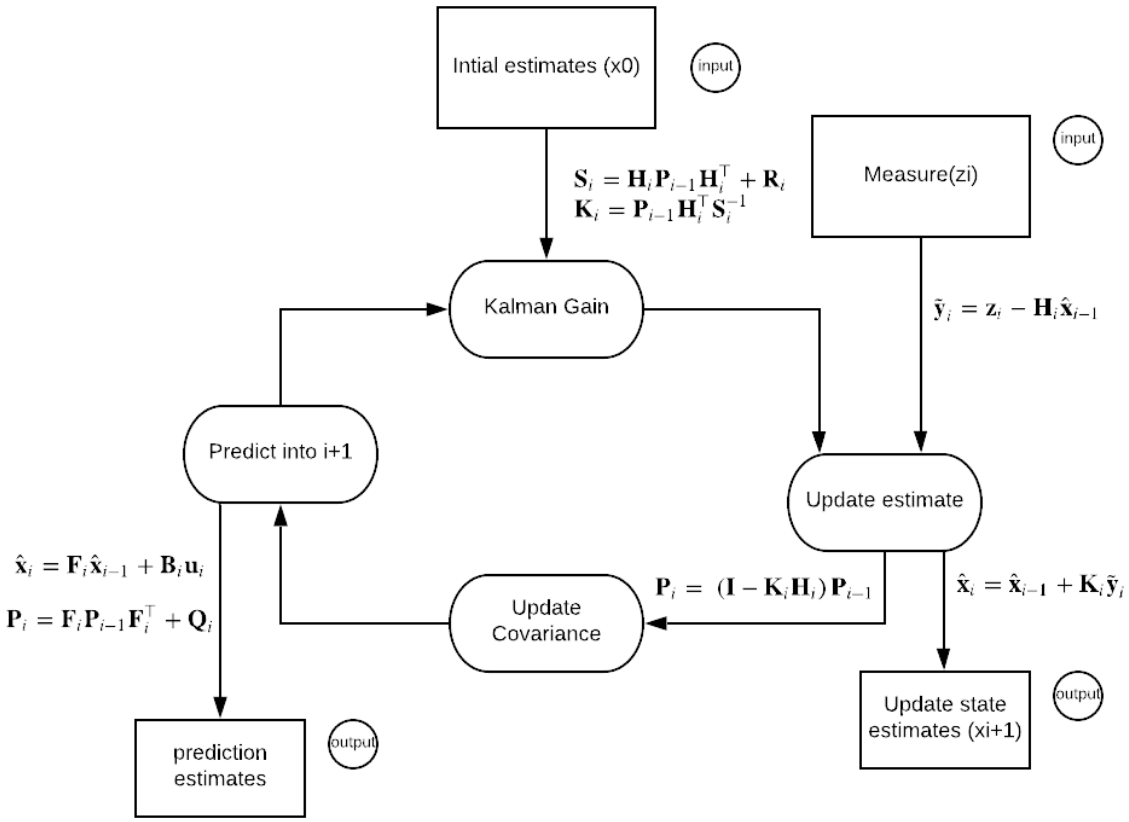

Figure 2.7: The Kalman filter recursive method for dynamic systems. First the transition model is applied. Second, the update phase adapts the values considering the measurement function.

The figure 2.7 shows a diagram with distinct steps. As we can see, the recursive estimate requires an initial value to start the predict-update step process. It means that we need to have an initial state estimated vector value $\mathbf{x}_0$, as well as define properly the initial covariance matrix $\mathbf{P}_0$. Other elements, such as the error covariance matrix $\mathbf{Q}_i$ and $\mathbf{R}_i$, have to be estimated properly in order to obtain the desired results of the model fitting.

There are many adjustments that can be done to the predict-update process. Subsequently, we will see that each experiment requires to find a proper description of the different elements that are parts of the model. Not every formula provided in its generic form works for every case, as we will notice in the circle fitting example, section 4.6, or the tracking the position of a robot, section 5.3.3, a important adaption has to be made to the Kalman filter in pursuance of better results.

2.4.3. The Extended Kalman filter

The Kalman filter brings good results within linear models. However, many models are based non-linear equations. In this respect, a new approach was presented, so that one may use the Kalman filter for a non-linear domain. The **Extended Kalman filter (EKF)** uses an extension in the Taylor Series of the the transition matrices $\mathbf{F}$ and $\mathbf{H}$ from equations 2.8 and 2.9. Thus, assuming the non-linearities in the dynamic and the observation model are smooth, we can estimate the next value of the state vector and therefore propose a new method for these non-linear applications [Terejanu (2005)].

Once again, we define the equations for the estate estimated vector and the measurement model:

$$\mathbf{x}_{i+1} = \mathbf{F}\mathbf{x}_i + \mathbf{w}_i \tag{2.21}$$

$$\mathbf{z}_i = \mathbf{H}\mathbf{x}_i + \mathbf{v}_i \tag{2.22}$$

The EKF linearizes an estimate of the current mean and covariance. It uses multivariate Taylor Series expansions to linearize the model. In consequence, the functions that define the transition and measurement model do not need to be linear; however, they must be differentiable [Kalman].

Unlike what happens with the Kalman filter, in its extended version the functions $\mathbf{F}$ and $\mathbf{H}$ can not be applied directly to the error covariance matrix $\mathbf{P}$. Instead, we compute the Jacobian (see definition 5) [Morrell (1997)].

$$\boldsymbol{F}_i = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{i-1}, \boldsymbol{u}_i} \quad \boldsymbol{H}_i = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_i}$$

Definition 5. Jacobian. The Jacobian is a determinant, which is defined for a finite number of functions of the same number of variables and in which each row consists of the first partial derivatives of the same function for each of the variables [Webster (2016)].

For example, consider the function $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2$, with $(x, y) \mapsto (f_1(x, y), f_2(x, y))$, where $f_1(x, y) = x^3 y$ and $f_2(x, y) = 5x + \sin y$. The Jacobian function is defined as:

$$\mathbf{J}_{\mathrm{f}}(x, y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 3xy & x^3 \\ 5 & \cos y \end{bmatrix} \tag{2.23}$$

As well as in the Kalman filter, the EKF is a recursive method based on the pair predict-update.

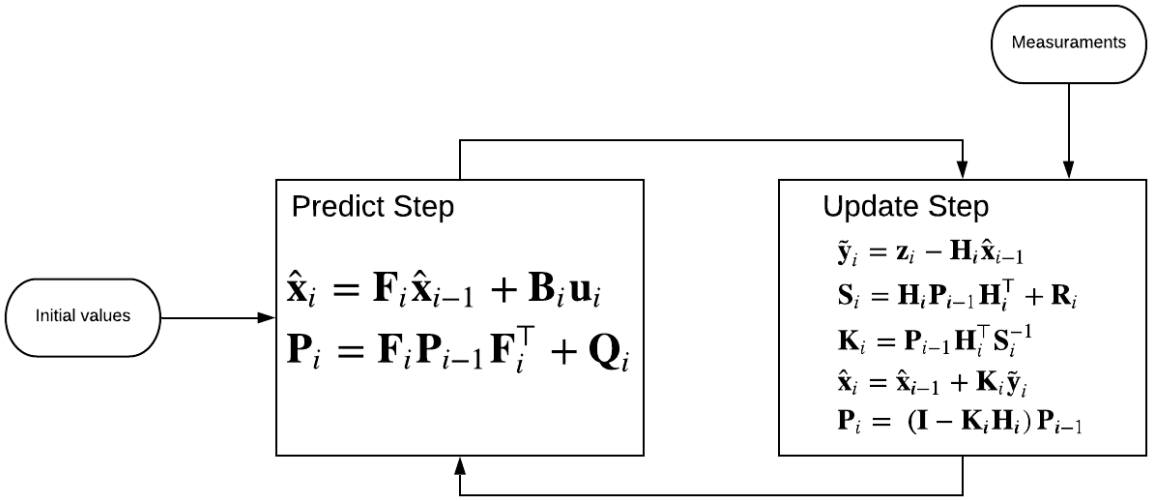

Figure 2.8: Extended Kalman Filter predict-update steps. The equations are based on the explanations made in previous paragraphs.

The Extended Kalman Filter is the most essential part of this paper. The most relevant steps, used in the localization and mapping processes proposed in this experiment, relate to this technique. In fact, the EKF is considered the standard of nonlinear state estimation in navigation systems [Wan (2006)].

Chapter 3

State of the art

This paper, to a certain extent, is a continuation of the work of [Muñoz Bañón (2016)] and [Cuadra Troncoso (2011)]. As explained in the introduction, we aim to provide accurate detection and representation of circular segments using a laser range finder attached to a robot. [Cuadra Troncoso (2011)] proposes an approach for lineal segments using the Kalman filter that adapts very well to the expectations. In reality, robots should also be able to detect and estimate circular segments such as column or trees.

In this chapter, we introduce the geometry aspects of a circle, followed by an explanation of the modelling of the Kalman Filter proposed in [Muñoz Bañón (2016)]. More precisely, the extension to the EKF will be needed since the circle is defined in two dimensions as a non-linear function. At the end of the chapter, the problems encountered in the developing and modelling will be commented, with the hope of describing how difficult the definition and representation of circular segments can turn to be, despite how simple can seem from a naive perspective.

3.1. The circle

In this project, we aim to provide an accurate description and modelling of the circular segments. In order to accomplish that, we require to have a solid knowledge of the mathematical description of the circle.

The circle is defined by every point that has equal distance from a specified central

point, defined as (x_0, y_0), in Cartesian coordinates [DODGSON (1896)]. It is described by the Pythagoras distance of two points, being this distance r (radius):

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{3.1}$$

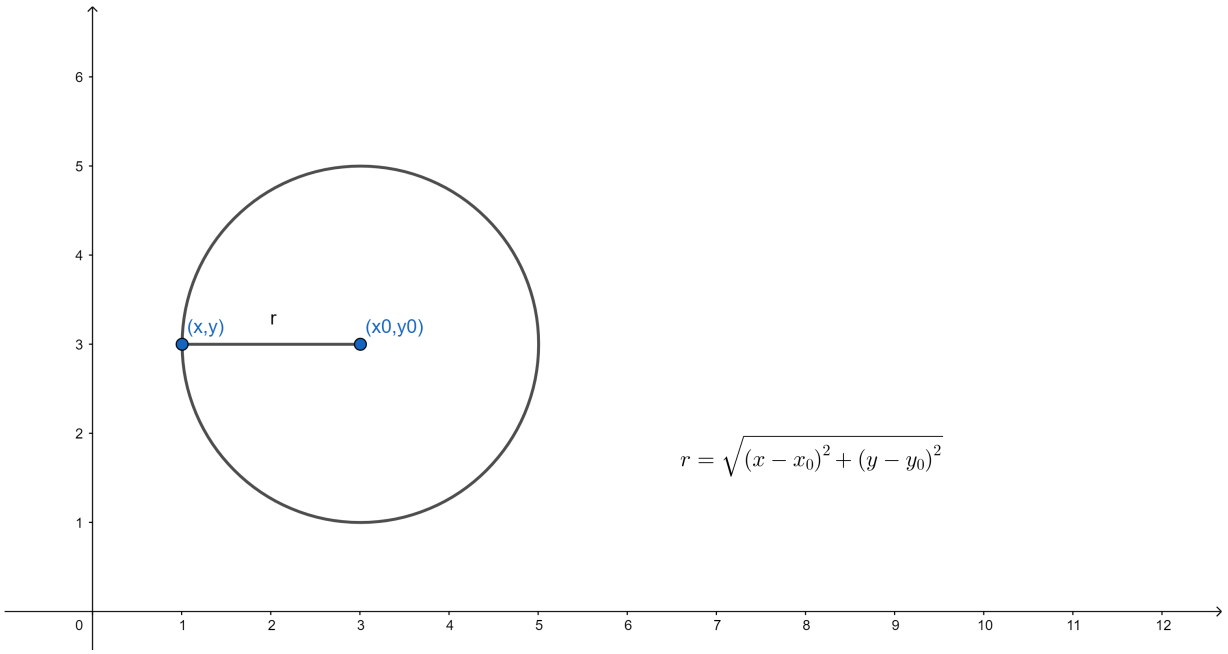

Figure 3.1: Circle in Cartesian coordinates.

In equation 3.1, the correlation between the three parameters r, x_0, y_0 might be advantageous from a descriptive point of view. However, modelling the parameters for fitting the circle might become more difficult when there exists a strong correlation between these parameters. In other words, when trying to fit these parameters, the error from one of the axes may affect the estimated value of the other two parameters. For this reason, we need a new description of the circle that allows as to get a lower interdependence. Moreover, the readings from the laser rangefinder are obtained as a polar distribution. Following the Pythagoras theorem, we obtain the polar representation of the circle as follows:

$$r(\phi) = \rho \cdot \cos(\Phi - \Theta)^2 \pm \sqrt{R^2 - \rho^2 \cdot \sin(\Phi - \Theta)^2} \tag{3.2}$$

where:

- ρ determinate the distances from the centre of the circle to the origin of coordinates.

- θ Is the angle between the centre of the circle and the origin.
- R is the radius of the circle.
- r is the distance of a certain point of the circle to the origin.
- ϕ is the clockwise angle from the x-axis.

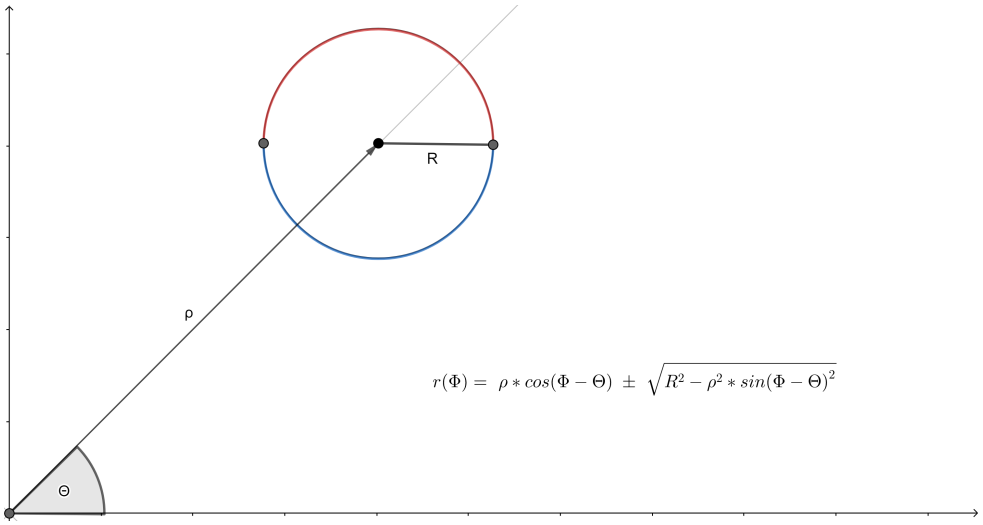

Figure 3.2: Circle in Polar coordinates. Differentiate concave(red) and convex(blue) parts from the origin of coordinates.

In figure 3.2, we can see the different elements defining the formula. As we can observe, it has a $\pm$ sign. Namely, the formula with a minus sign is the one that draws the convex part of the circle (blue in the figure), while using the plus sign we obtain the concave sight with regard to the origin (red in the figure).

The formula in polar coordinates improves in terms of modelling. Now, we only have three parameters that depend on r, which is a function of ϕ. In the polar coordinates, ϕ is the angle with the origin and takes value from 0 to 2π. However, this values are admissible only if the origin of coordinates is inside the circle. For this project, this has no much sense, since we are trying to estimate circular segments and the centre of origin is the robot. Admitting a circle containing the origin would mean, in other words, that the robot is in the inside of, for example, a tree or a column. Therefore, we have to provide a valid ϕ for every circle that is not containing the origin of coordinates. Given a circle, defined in polar coordinates, with a centre in ρ and with an angle θ, we must provide the angle α, which is the maximum angel that can be added or subtracted to θ to obtain the set of angles ϕ that describe the circle [David (2014)]:

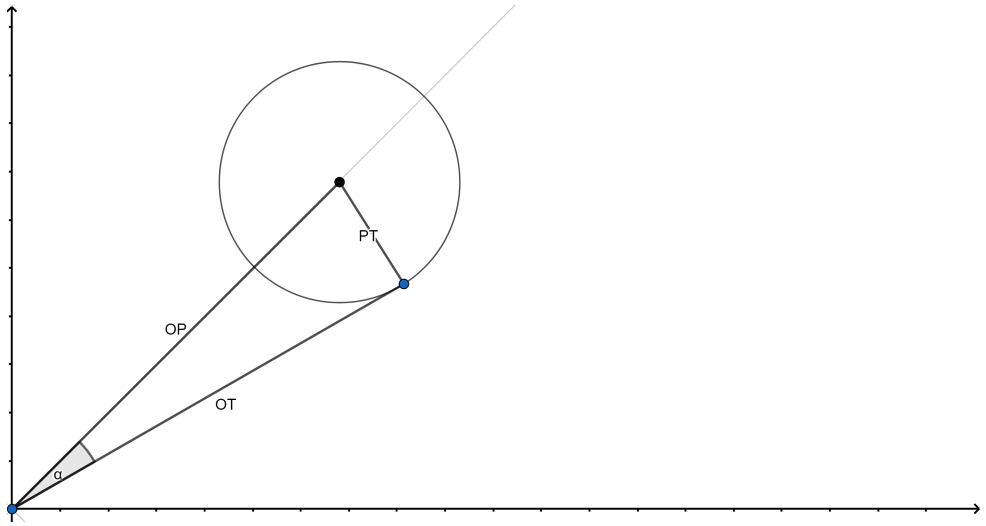

Figure 3.3: The angles α marks the limits of ϕ for the description of circle outside the origin of coordinates.

$$\theta - \alpha \leq \phi \leq \theta + \alpha$$

The angle α is calculated by drawing a segment from the origin O to the centre of the circle ρ, named as OP. Next step is to draw the tangent between O and $T \longrightarrow OT$.

From the figure 3.3, we define the angle between OP and OT:

$$\sin\alpha = \frac{PT}{OP} \longrightarrow \alpha = \arcsin\frac{R}{\rho} \tag{3.3}$$

where R and ρ denote the same as in equation 3.2.

We have formally described every circle which can be drawn in polar coordinates. However, three main complications may appear from a computational point of view:

1. The $\pm$ sign, found in the equation 3.2 has to be implemented in two different cases. In the meantime, just the minus sign will be regarded, since we consider only the convex part of the circle, this is, the one detected by the robot. It is important to notice that circular shaped walls are not considered in this project.

2. For certain values of r, the solution of the equation might be an imaginary number. The last will interfere with the estimations.

3. The last complication comes with the estimation of α, since the value of α depends

on the estimation of the parameters of the circle, an error in the estimation would add possible values to ϕ, which do not belong to this specific circle.

As commented in the introduction, the difficulty of a project usually becomes more visible when we acquire a reasonable knowledge. Later on, we see that even though the circle might have a fundamental mathematical formalization, it can be complex to design a proper model.

3.2. Circle readings

In a real experiment, the various data we obtain from the rangefinder sensor is always associated with a certain noise. Estimating segments is a task that requires a good understanding of the world in which we are working. Successfully developing a system that adapts to different situations is difficult, if not unfeasible.

The before mentioned also holds for the circle fitting problem. Many aspects must be considered when using a rangefinder (e.g., a laser), such as illumination, texture or reflection. Therefore, accomplishing an error-free approach must not be the goal. As we briefly explained in the section 2.3 (SLAM), when trying to estimate features extracted from sensor readings, the aim is to give a pragmatic description of the world rather than a perfectionist one. Following this philosophy, we are ready to introduce a first approach to the circle recursive estimation using the Kalman filter. In order to be able to do this, we have to make some assumptions.

Primarily, we will obtain simulation data provided by laser rangefinder that gives us 180 points, that is, from 0 to 179 degrees. Moreover, the unit of measure for angles is radians, so, in fact, we obtain 180 points from 0 to π.

Secondly, we assume a white noise with a Gaussian distribution. The Kalman filter presents a solution for such type of noise and has proven to solve many problems in that context [Nixon (1997)].

Figure 3.4 illustrates the readings of a circle with a radius 0.25 meters and a distance of 1 meter. Its exact representation by equation 3.2 is: $\rho = 1.5$, $\theta = 1,57$ rad, $R = 0.25$ m.

If we compare the real representation of a circle with what we see in figure 3.4, we realize that they do not look alike. That is because, on the one hand, we are just able to

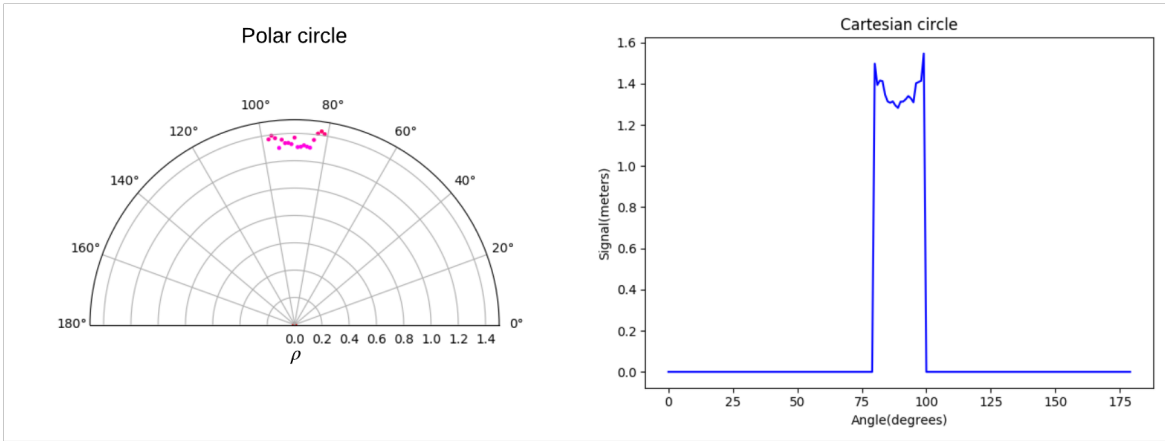

Figure 3.4: Circle readings with $\sigma = 0.02$ Gaussian noise. On the left, we have the polar representation of the data obtained by the laser. On the right, the same data in Cartesian coordinates.

access the convex part of the circle, and from the other hand, a Gaussian additive noise with a deviation $\sigma = 0.02$ is added to the signal. Thus, the distribution of the point might seem randomly situated in the coordinates.

3.3. Circle and the Kalman Filter

Having in mind how the leaser readings look like, we can start defining the different components of the Kalman Filter.

The first step to take is to design the state variable $\mathbf{x}$ in equation 2.8. Since the data provided by the laser rangefinder comes as a function of distances in degrees, the most natural choice is the polar representation. Moreover, we have already described the advantage of having the function $r(\phi)$ rather than different inputs (x, y).

Coming back to the equation 3.2, we see that we have three parameters, and these are the ones to be established as the state variables.

$$\mathbf{x} = \begin{bmatrix} \text{distance} \\ \text{angle} \\ \text{radius} \end{bmatrix} = \begin{bmatrix} \rho & \theta & R \end{bmatrix}^\top \tag{3.4}$$

The next step is to define the process model. Recalling section 2.4.2, the Kalman filter is divided into two phases. In the predict step, we predict the values the state variables will acquire. This is done by using a matrix $\mathbf{F}$ that computes the transition from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. In our case, the matrix $\mathbf{F}$ is the identity matrix.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

That is because the transition is produced by the transformation from one step to the next one of the parameters given in the formula. Once $\mathbf{x}$ and $\mathbf{F}$ are defined, we have to decide which one is the error $\mathbf{w}_i$ associated with the process model. For the same reason that the parameters are the only ones considered in the state estimate vector 3.4, there is no noise attached to the process model.

The next step is to define the covariance matrix $\mathbf{P}$. In this case, according to [Muñoz Bañón (2016)], we apply a normalization by the Gaussian error $\mathbf{P} \longrightarrow \frac{\mathbf{P}}{\sigma^2}$. It will give as a matrix $\mathbf{P}$, which is independent of the error covariance. In consequence, the error $\mathbf{Q}$, associated with the variance in the model, is equal to 1.

Once we describe the process model, the measurement model must be clarified. The equation 3.2 is non-linear, and for this reason, we are not allowed to use the standard Kalman Filter , we must to use its extension. The basic prerequisite for the Extended Kalman Filter is to be differentiable 2.4.3. Since we have chosen the polar equation of the circle, we obtain the partial derivatives in the determinant shape; this is, the Jacobian $\mathbf{H}$. In the EKF, the function that interacts with the process covariance and the Kalman gain is the Jacobian of the measurement function:

$$\boldsymbol{H}_i = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_i}$$

Considering equation 3.2 and the state model, the Jacobian is formed as follows:

$$h(\overline{\mathbf{x}}) = \rho \cdot \cos(\Phi - \Theta)^2 \pm \sqrt{R^2 - \rho^2 \cdot \sin(\Phi - \Theta)^2}$$

$$H_n = \begin{pmatrix} \frac{\partial r}{\partial \rho} \\ \frac{\partial r}{\partial \theta} \\ \frac{\partial r}{\partial R} \end{pmatrix} \tag{3.5}$$

Now we have to obtain the partial derivatives of r by every parameter in the state vector $\mathbf{x} = (\rho\ \theta\ R)$:

$$\frac{\partial r}{\partial \rho} = \frac{\rho_{n-1} \sin^2(\phi_n - \theta_{n-1})}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}} + \cos(\phi_n - \theta_{n-1})$$

$$\frac{\partial r}{\partial \theta} = -\rho_{n-1} \sin(\phi_n - \theta_{n-1}) - \frac{\rho_{n-1} \sin(\phi_n - \theta_{n-1}) \cos(\phi_n - \theta_{n-1})}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}}$$

$$\frac{\partial r}{\partial R} = -\frac{R_{n-1}}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}}$$

Having the partial derivatives, we define $\mathbf{H}$ as:

$$\mathbf{H}_n = \begin{pmatrix} \cos(\phi_n - \theta_{n-1}) + \frac{\rho_{n-1} \sin^2(\phi_n - \theta_{n-1})}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}} \\ -\rho_{n-1} \sin(\phi_n - \theta_{n-1}) - \frac{\rho_{n-1} \sin(\phi_n - \theta_{n-1}) \cos(\phi_n - \theta_{n-1})}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}} \\ -\frac{R_{n-1}}{\sqrt{R_{n-1}^2 - \rho_{n-1}^2 \sin^2(\phi_n - \theta_{n-1})}} \end{pmatrix} \tag{3.6}$$

Those mentioned above are the necessary components required for the basic definition of the model that we use in the EKF. In [Muñoz Bañón (2016)], some modifications are proposed concerning the generic equations given in section 2.4.2, especially in the update of the covariances $\mathbf{P}$ of the Kalman Filter.

In algorithm 1, we introduce two innovations concerning the update of the covariance matrix $\mathbf{P}$ (line 15). The first one is that the identity matrix $\mathbf{I}$ is no longer used as a subtracted entity, and the second one is newly introduced lambda parameter multiplied by I_k, where k is the number of parameters of the state estimate vector, in this case, $k = 3$.

Algorithm 1 Proposed Kalman recursive function

def ekf(r_i,ϕ_i,$\mathbf{x}_{i-1}$, MSE_{i-1}):
#returns the Jacobean based on r and phi values from equation 3.6
$\mathbf{H}_i$ = Jacobean(r_i,ϕ)
calculate the residual
$\tilde{\mathbf{y}}_i = \mathbf{r}_i - H_i(x_i)$
innovation covariance
$\mathbf{S}_i = 1 + \mathbf{H}_i \mathbf{P}_{i-1} \mathbf{H}_i^\top + \mathbf{R}_i$
kalman gain
$\mathbf{K}_i = \mathbf{P}_{i-1} \mathbf{H}_i^\top \mathbf{S}_i^{-1}$
update the state estimate
$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{i-1} + \mathbf{K}_i \tilde{\mathbf{y}}_i$
modified update covariance compare with equation 2.20
$\mathbf{P}_i \leftarrow \left(\mathbf{P}_{i-1}^{-1} + \mathbf{H}_i^\top \mathbf{H}_i + \lambda \mathbf{I}_k\right)^{-1}$
Calculate the added mean square error
$MSE_I = MSE_{i-1} + MSE$

The lambda function is the component to apply the Tikhonov regularization [Tikhonov et al. (1998)]. For a deeper explanation of the search of an appropriate λ and the development of the Tikhonov regularization, it is better to refer to the master thesis [Muñoz Bañón (2016)], section 3.3.Where mentioned it as a solution to ill-conditioned regression of non-linear models.

The Extended Kalman filter is an recursive process, meaning that we need the initial values of the state vector $\mathbf{x}_0 = (\phi_0, \theta_0, R_0)$.

Another essential element is introduced to the Kalman iteration function. The estimation of the noise error adding proves to be helpful later on when we try to distinguish between different types of segments (e.g., linear or circular). For that, we add to the algorithm the calculation of the mean squared error shown in equation 2.7.

The calculation of the MSE will orientate us in every iteration. When we observe the variation in the value, we can extract some ideas on how the estimation itself is developing. For example, a growing series of MSE values will probably indicate that the estimation is ill-configured or somehow malfunctioning. It could also mean that we are trying to misplace a wall for a column or the other way around, in more practical terms, will see about this in chapter 5.

3.3.1. Student's t-test

The last important aspect of the process is to decide when the process should stop. There are two main proposed conditions, in which the iteration of the filter should come to an end:

1. The value of the error has increased several times in comparison to previous iterations. The MSE can increase due to the existence of an outlier. Having a closer look to figure 3.4, one observes that certain values, given in an erroneous context, can differ largely from the others. The probability of these values appearing one after another in a row is reasonably low. Hence, if the calculations of MSE has a growing tendency, probably we are not calculating properly where the segment ends.

2. Another condition to stop the iteration is a so-called break-point. A break-point means that, apparently, we have reached the end of the segment from one side or the other.

A common way to analyze the progress of statistics problem, where we need to estimate the parameters with small sample size and unknown variance, is the Student's t-distribution (or simply the t-distribution). However, every distribution that follows t-distribution has its peculiarities, determined by the degrees of freedom [tdi]. The number of independent observations in a set of data is called the degree of freedom; for instance, if we estimate from a single sample, the degree of freedom is $n-1$. The implicated parameters of the model play a role in the calculations.

For using the EKF together with the Student's t-distribution, we have to accept the considerations of the Gauss Markov theorem assumptions [Anderson (2018)]:

- **Linearity**: the estimated parameters must be linear.
- **Randomness**: the data must be randomly obtained from the population.
- **Non-Collinearity**: the calculated regressors are not perfectly correlated.
- **Exogenous**: the error is not correlated with the regressors.
- **Homoscedaticity**: the error variance is constant, along with the distribution of the values of the regressors.

The first condition from the list above is accomplished by the EKF when using the partial derivatives procures a linearization of the state vector. The data is extracted randomly as seeds obtained in the scale-space algorithm, section 4.3.1. The regressors in equation 3.4 are not correlated. The error in our case is related to the measurement model given by the function $r(\phi)$, and it is not dependant on the regressors. Finally, homoscedasticity is an assumption that we make for the current case of study. However, the error in real experiments has a variable distribution, provoked by the reflection of different materials the laser rangefinder might be pointing to, meaning that the more realistic approach would be considering the heteroscedasticity.

3.3.2. Comments and proposals

Now that wee have discussed the most important considerations on how the Extended Kalman filter is designed, we should consider the limitations of the subject matter that has been discussed thus far. We retake a look at some of the problems already described in order to provide different perspectives that motivate further research.

The estimation of the circle, as proven in [Muñoz Bañón (2016)], is not an easy task. Some complications, as described further in this section, have to be considered.

First of all, the appearance of imaginary numbers as a result of the calculation of the square root in the Jacobean equation 3.6. When the estimated circle does not contain the origin of coordinates, we might obtain negative numbers inside the square root. In previous projects, only the real part of the result was considered since the data set was small; therefore, rejecting some points would be unreasonable. Hence, the whole data set was included by merely removing the imaginary part of the resulting numbers. The problem arises when the imaginary numbers, resulting in these cases, are not in direct relation with the real circumference. Therefore, its inclusion in the estimation process usually leads to undesirable results.

The second problem lies in estimating circle objects that are too close to the robot. When the robot is close to such an object (e.g., a column), the seed of data extracted from the laser rangefinder might be excessively concentrated in an interjection of the segment, resulting in not accurately defined radius of the circle (occasionally, the segment might event be approximated to a straight line). This issue is further discussed in the following chapter.

The distinction between different segments, be it linear or circular, is profoundly influenced by the position of the robot. Moreover, it is related to the idea of break-points.

The proper detection of each segment boundaries is essential. Errors in estimation can include outliers in the data set.

We propose another approach, where we try to analyze the circle fitting problem from a more pragmatic point of view. Since the main problem seems to be related to the small representation of the segments, by adding movement to the robot we obtain more features that represent the same circle, allowing us to mix and construct larger-scale data sets, in many cases improving the estimation of the circle. Naturally, handling the movement comes with certain disadvantages, which have to be analyzed. For that, we use some SLAM techniques which help to construct maps. All of the above serves as a guide to introduce a circle as a new geometrical figure.

Chapter 4

Circle fitting

In the previous chapter, we have described the Extended Kalman filter iteration process. However, for the first step of the process, we should set the parameters of the state estimate vector. In this chapter, we explain the mechanism of extracting the data set from a circle and, later on, obtaining the initial parameters from the noisy signal. The main idea is to be able to acquire different features or segments that represent the same circle in order to improve the estimation. Finally, we propose a new design of the Extended Kalman Filter.

4.1. Robot and sensor

The porpoise of this project is to take a step forward the description of circles since they appear in many robotics experiments. Although the most realistic situation would be using a real robot, due to the low budget and time window limit, these experiments use a simulator, specifically *V-REP* by [CoopeliaRobotics]. This simulator is suitable for our research due to three main reasons. Firstly, it comes with different models, particularly: robots, static objects, and materials. Secondly, each of these modules can have associated scripts that allow us to control the module independently from the others, which helps to separate the navigation from the sensor reading process. Lastly, and most importantly, *V-REP* provides a beneficial API that allows us to accede the simulation from other programming languages, in this case, *Python*.

Let us now make an introduction to the robot that we are going to use in the experiments, namely, the **Pioneer P3DX**. In the literature, the Pioneer robot is widely used.

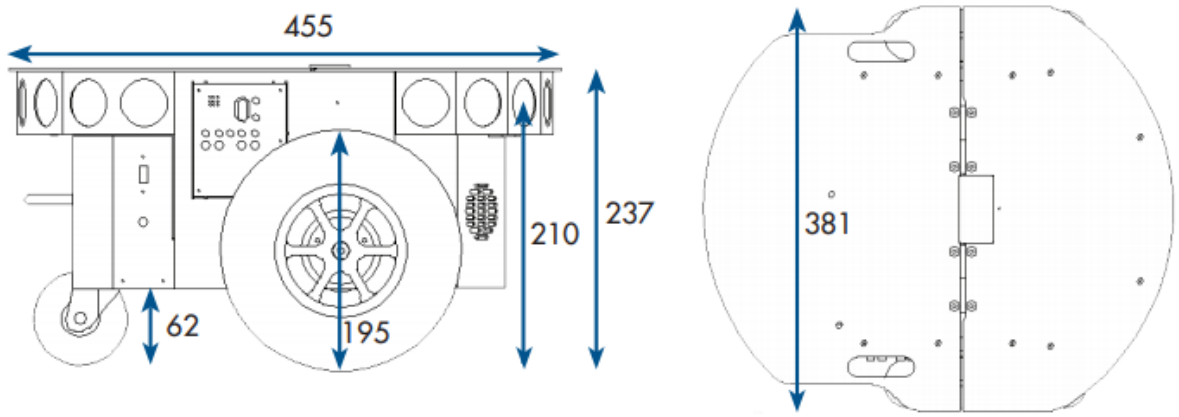

Figure 4.1: Pioneer 3-DX schemes and dimensions. The figure shows measure of important components, such as the wheels and the axis of the robot.

It is composed of two lateral wheels that allow a smooth movement and change in direction. The Pioneer research robots are the world's most popular intelligent mobile robots for education and research purposes. Their versatility, reliability and durability have made them the preferred platform for advanced intelligent robotics [MobileRobots].

The dimensions of the robot, shown in figure 4.1, play a significant role in the next chapter, where we enter the explanation of localization and odometry.

The usability and easy access to the robot, plus the fact that the simulator V-REP already includes a representation of a module of the robot, are the reasons that determine this choice. The robot also includes SONAR rangefinders and sockets on the top in order to connect, in this case, the laser beam, on which readings we will base the circle fitting.

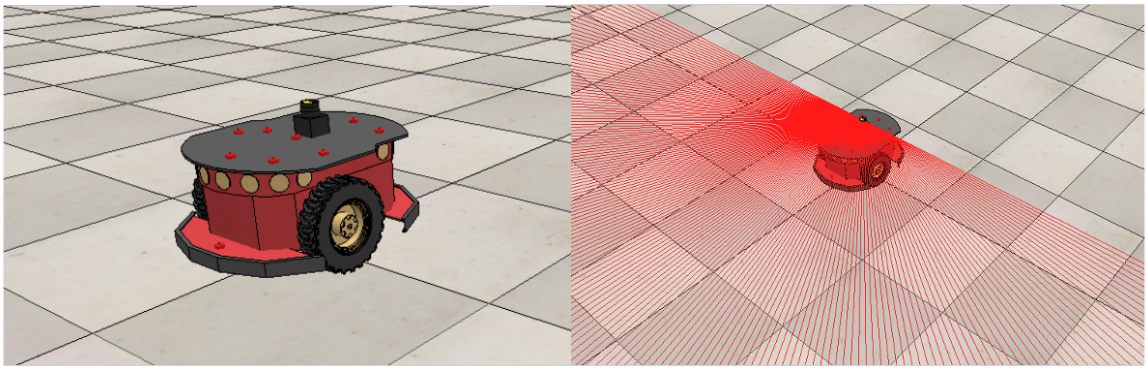

Figure 4.2: Pioneer 3-DX with an Hokuyo Laser attached.

4.1.1. Laser rangefinder

The laser we use is the Hokuyo UST-30LX Scanning Laser Rangefinder, attached to the top of the robot. In [Dekan and Duchon], the author provides a probabilistic model for a laser range finder. In the experiments of this project, we use the same laser.

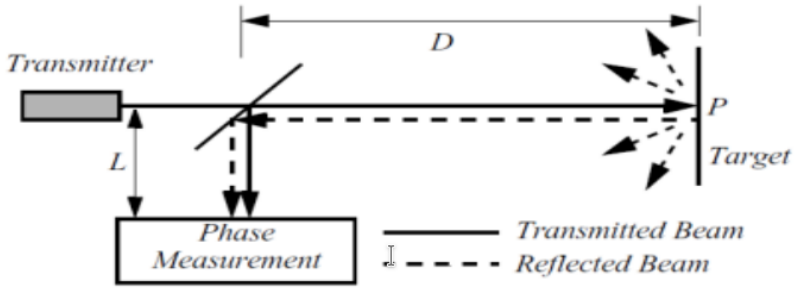

Figure 4.3: Scheme of the phase shift calculation by the laser rangefinder [Siegwart et al. (2011)].

A laser rangefinder usually operates by the principle of the time of flight. It measures the phase shift from the sent to the received signal [Siegwart et al. (2011)]. For obtaining the phase shift, the laser sends a light beam towards the environment. When the laser beam hits a certain point P, the light reflects if the surface thickness is greater than the wavelength. The reflection is, therefore, isotropic. It is important to notice that the calculation of the distance depends on the material, due to the thickness of the surface [Dekan and Duchon]. In our case, we consider the specifications for the Hokuyo sensor provided by the manufacturer. The specifications of the laser in figure 4.4 show that the accuracy from 0.1m to 10m has a deviation error of 30mm under 3000lx White Kent Sheet. In this project, we use the approximation made by [Dekan and Duchon] of concrete materials. Hence, we consider that the detected objects are walls or columns built of these materials.

The accurate description, provided in this article, and the fact that is it already implemented as a model in the simulator we use, are the two reasons why we choose this laser rangefinder. However, a particular modification of the simulator is needed to adjust the errors and be able to obtain the 180 measures of the laser, which is originally set to an angle of 270 degrees.

Detection Range	Guaranteed Range: 0.1 ~ 30m (White Kent Sheet) Maximum Range : 0.1 ~ 60m
Detection Object	Minimum detectable width at 10m : 130mm (Vary with distance)
Accuracy	Under 3000lx : White Kent Sheet: ±30mm*¹ (0.1m to 10m) Under 100000lx : White Kent Sheet: ±50mm*¹ (0.1m to 10m)
Measurement Resolution	1mm

Figure 4.4: The specifications provided by the manufacturer of the Hokuyo UTM-30LX [Hokuyo (2007). Different errors expected considering the distance from the objects.]

4.2. The first experiment

In order to provide a more practical explanation of the calculations and the implementation of the Kalman filter, a basic experiment is conducted. We define a simple case, particularly, an unmovable robot facing a column with an angle of 90 degrees.

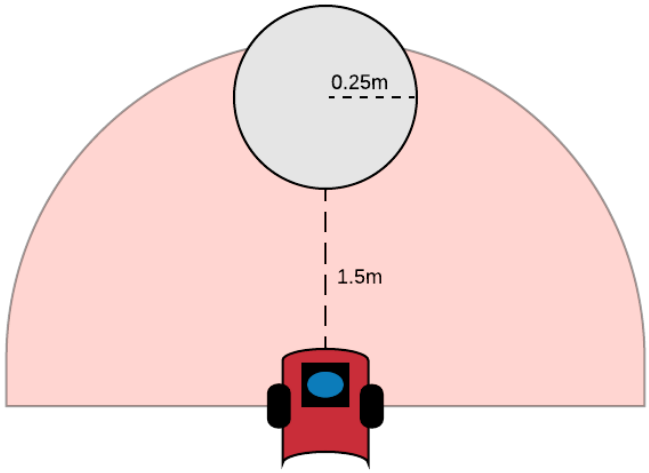

Figure 4.5: The robot pointing to a column in the first experiment. The column is one and half meters away from the robot.

As we can see in figure 4.5, the column is $1,5$ meters away from the robot and has a radius of 0.25 meters. Using the parameters from equation 3.2, we obtain $\rho = 1.5$m,

$\theta = \frac{\pi}{2}$rad, and $R = 0.25$m. Notice that all the measures given in this paper are in meters, with the exception of the angles that are given in radians.

The laser readings, given by the simulator in this situation, are shown in figure 4.6 below.

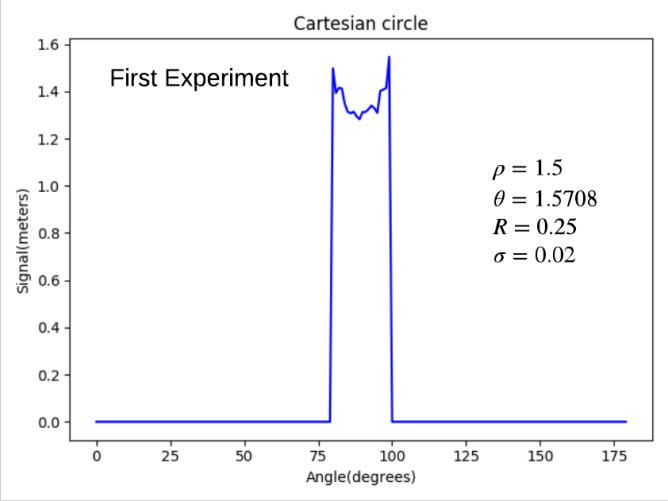

Figure 4.6: The readings provided by the simulator for the first experiment.

This experiment serves as a first demonstration of the application of steps to run the Kalman filter. Considering the difficulty of the process, it is recommended to review this base case later, when we examine many-corners cases. The philosophy lies in hardening the problem once we obtain expected results at each level of difficulty.

4.3. Signal clustering using *scale-space*

To be able to start the Kalman recursive process of a segment, we need to be confident in the extraction of the so-call seeds. In Cuadra Troncoso (2011), the author uses an Artificial Vision technique, in order to obtain a proper division of the segments that composed each reading. The algorithm provides initial data, named seed, from which we can initialize the Kalman estimation. Thus, providing us with the initial data, essential to get the initialize the parameters of the circle. Therefore, we need a method to extract the limits of every segment, existing in the function $r(\phi)$, that is, the polar representation of laser readings. In this project, the applied algorithm for extracting the seed information is based on the explanations from [Vincken et al. (1997)].

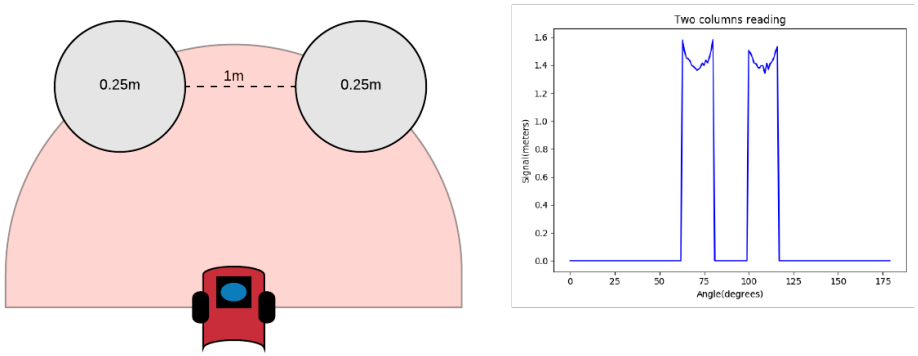

Figure 4.7: Example for scale-space, set-up with two columns

Let us see an example that might clarify the matter. Assume a case of finding two columns located in front of the robot, as in figure 4.7. We need to determine that there are two segments to be analyzed and obtain data from both of them. The difficulty of application of this procedure grows with the number of segments and their proximity to each other. Another essential issue to comment is the possibility of an appearance of seeds in between the two segments, meaning that the initialization of the filter is incorrect, hence, producing not desirable results.

4.3.1. Introducing scale-space

We need to find a perspective from which the data obtained in the laser rangefinder is identified as a set of segments. In [Witkin (1984)], the author provides a method, in which, given a 1D signal, by applying convolution to it we obtain a function of two dimensions $F(x,t)$, representing a surface in which the signal variations appear as zeros in the first derivatives, $\frac{\partial^k F}{\partial \varphi^k}$ of the function F. In other words, the fundamental idea of this algorithm is to parameterize the function space with a single parameter so that one can localize or select interesting variations of the parameter. The convulsion, based on Gaussians, provides an approach where the different scales do not add new extremum to the function F; instead, the parameter variations become more workable by the elimination of other aspects that might be noisy [Kumar (2018)].

Moreover, scale-space allows us to implement feature analysis without having any prior information about where the appropriate scale to extract useful information for a given data set is [Lacy and Thacke (1998)].

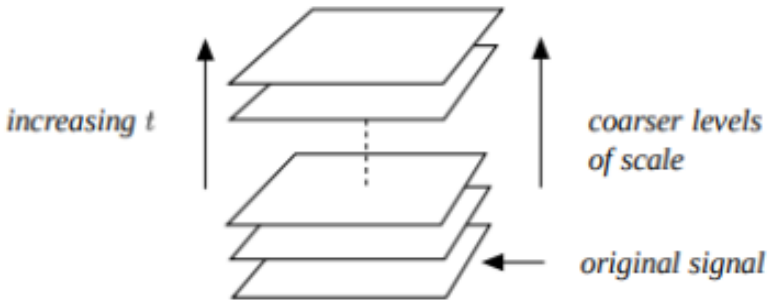

Figure 4.8: The original signal is derived in different levels in order to represent it in different levels of scale. The different layers maintain the characteristics that are fundamental for the segmentation.

4.3.2. Process description

Formally, given a 1D function $f : \mathbb{R} \to \mathbb{R}$, its scale-space representation is given by:

$$L(x;t) = \int_{\xi \in \mathbb{R}} f(x-\xi) g(\xi) d\xi$$

where g denotes the Gaussian kernel with a variance t that refers to the scale parameter:

$$g(x;t) = \frac{1}{(2\pi\sigma^2)^{D/2}} e^{-\left(x_1^2+\cdots+x_D^2\right)/2t} \tag{4.1}$$

By applying the smoothing, using the Gaussian kernel, we obtain a parametric representation of the specific signal. The segmentation strategy follows four steps:

1. **Blurring**: The first step applies the Gaussian kernel from equation 4.1, creating the different scales. This process, known as convolution, provides us with a smoother representation of the signal. It is essential to follow the principle **scale invariance**. Thus, the kernel that is used might not affect the characteristics of the signal in the consecutive scales.

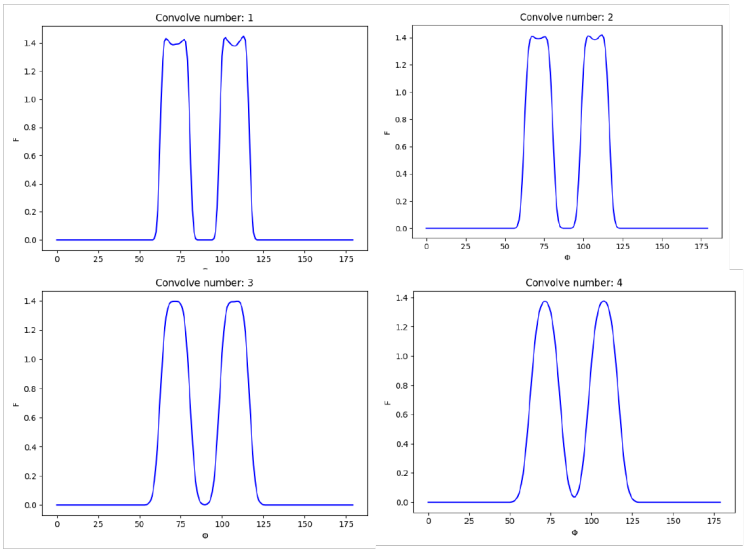

Figure 4.9: The convolved signal from figure 4.7.

2. **Linking**: The next step of the process is to create connections between the voxels of the signal that are present at higher levels of scale. Putting together twin voxels from levels n and $n+1$ is necessary for the determination of characteristics that stay in the scale invariance. By linking all the voxels of the signal, we create a segmentation tree. The linkage criterion follows an affectiveness approach. The dependency of the affection between a child and a parent voxel is based on the Gaussian. Hence, the parents are selected by the level of affection to a possible child, where this affection is measured by:

$$\text{affection} = \mathcal{D} \cdot \frac{\sum_{i=1}^{N} w_i \cdot C_i}{\sum_{i=1}^{N} w_i}, \text{ with } C_i \in [0,1]$$

where D is the distance and C_i's are individual linking components.

3. **Root labeling**: Once the connections between the voxels are created, we can proceed with the segmentation. Weakly linked voxels, as well as the ones on the top level of the segmentation tree, are labelled as roots, meaning that a voxel, selected from the scale tree, represents a single segment at a basic level. Labelling the voxel as root is usually completed by choosing a heuristic criterion, such as the ground intensity variance.

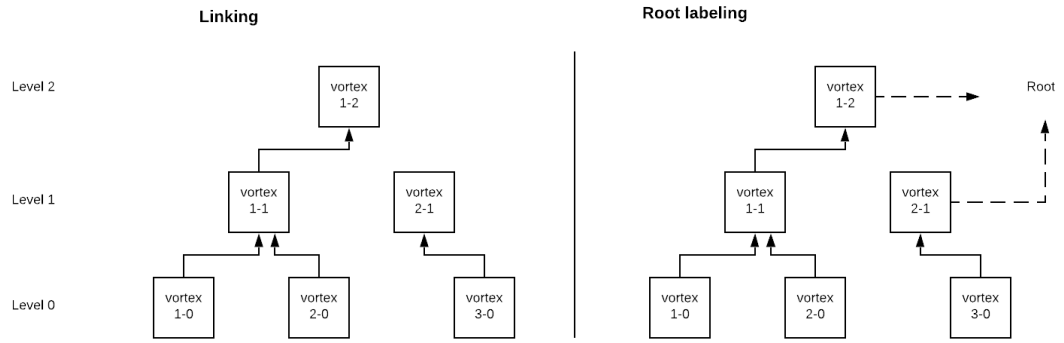

Figure 4.10: a) The linking step, where different voxels are connected. b) Voxels with a weak linkage on top-most level are set as root.

4. **Downward projection**: Once we have obtained the segmentation tree, we can project the connections onto the the signal. The linkages are followed downwards in order to relate the root voxels. Furthermore, all the ground voxels, connected to a common root, are set to be a single segment. In figure 4.10, voxels 1-0, 2-0 and 1-1 are together in the same segment.

For a deeper description of the scale-space process applied to the signal refer to Vincken et al. (1997), where the reader can find a more exhaustive explanation of the concepts commented above.

The scale-space process has proved a good performance in the experiments of this project. However, adding complexity to the world could create errors that have not been observed in this case.

The initialization provided by this algorithm is of great importance since it is the first step to analyze the signal. Hence, errors in the calculations would most likely lead to the wrong estimation of the segments present in the readings.

4.4. Parameters initialization

After we have obtained the first points representing a segment, we can continue with the initialization of the Kalman recursive process. The results of applying the scale-space

algorithm to the data from figure 4.11, in which we base the first experiment, are shown in figure 4.11 below.

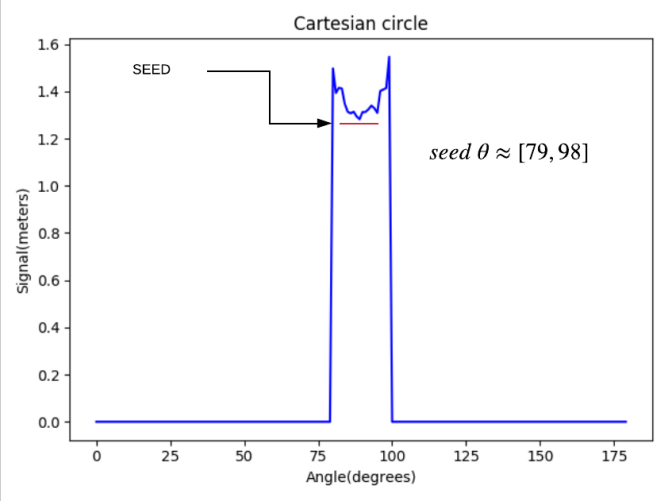

Figure 4.11: Seed obtained from scale-space process applied on the laser data of the first experiment.

The points that we obtain are approximately between 1.39 and 1.72 radians. This approximation covers almost the whole circle from the data set, which gives a good impression on the segmentation executed by the scale-space approach. The next step is to initialize the parameters of the circle.

As it has already been mentioned in section 3.3, the initial parameters ρ, θ, and R of the described model have to be set, to start the iterations of the filter.

4.4.1. Geometrical approach

The most intuitive mathematical way of fitting a circle given certain points is to use its Cartesian equation and create a system of equations to find the unknowns [ja72]. Looking back to section 3.1 of the previous chapter, we recall the equation 3.1 of a circle in Cartesian coordinates:

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

The equation has three unknowns, therefore, we need at least three points described

by $(x, y) \rightarrow (x_1, y_1), (x_2, y_2), (x_3, y_3)$. Having the three points, we define the system of equations:

$$\begin{aligned} (x_1 - x_0)^2 + (y_1 - y_0)^2 - r^2 = 0 \\ (x_2 - x_0)^2 + (y_2 - y_0)^2 - r^2 = 0 \\ (x_3 - x_0)^2 + (y_3 - y_0)^2 - r^2 = 0 \end{aligned} \tag{4.2}$$

Let us recover the points from the seed obtained in the previous section. And apply those to the system of equations to see in a practical example, based on the experiment described in section 4.5.

However, before, we must consider the fact that the values, determining the seed, are in Polar coordinates. It exhibits the first disadvantage of this method. From a mathematical perspective, the transformation between coordinates (e.g., Cartesian to Polar), might not imply a significant difference. However, in computational terms, this transformation can have a notable impact on performance. Having the aforementioned in mind, we consider three points from the seed: we take the initial value, the last one, and the one in the middle.

$$\text{First point} \rightarrow r(1.3962) = 1.49 \longrightarrow (x_1, y_1) = (0.26, 1.46)$$

$$\text{Middle point} \rightarrow r(1.55) = 1.28 \longrightarrow (x_2, y_2) = (0.026, 1.27)$$

$$\text{Last point} \rightarrow r(1.72) = 1.54 \longrightarrow (x_3, y_3) = (-0.22, 1.52)$$

The transformations above are computed by using the Polar to Cartesian convertion as follows:

$$x = r \cdot \cos\phi$$

$$y = r \cdot \sin\phi$$

In the following, we substitute the corresponding values from equation 4.2:

$$\begin{aligned} (0.26 - x_0)^2 + (1.46 - y_0)^2 - r^2 = 0 \\ (0.026 - x_0)^2 + (1.27 - y_0)^2 - r^2 = 0 \\ (-0.22 - x_0)^2 + (1.52 - y_0)^2 - r^2 = 0 \end{aligned}$$

By subtracting the first equation from the second one and from the third one, we obtain a system of two lineal equations:

$$-0.47x_0 + 0.38y_0 - 0.76 = 0$$

$$0.96x_0 - 0.12y_0 + 0.161 = 0$$

Therefore, we have found the coordinates of the center of the circle as:

$$(x_0, y_0) = (0.16, 1.76)$$

By equation 3.1 we calculate the radius using the first point considered:

$$r = \sqrt{(0.26 - 0.16)^2 + (1.46 - 1.69)^2} = 0.31$$

Recall that the circle described in section 4.5 in Cartesian coordinates has center coordinates $(x_c, y_c) = (0, 1.5)$ and radius $= 0.25$. We observe that the obtained results differ from the expected ones. By assuming that we have more than three points in the seed, we try to improve the initial estimation using an approach based on the calculations done above.

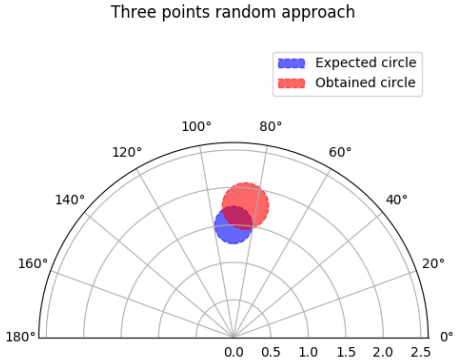

Figure 4.12: Comparison of an expected circle(blue) and obtained one(red) by three randomly selected points.

The error in the initial estimation is related to two main factors. The first one is that the points contained in the seed have an associated error, therefore, if we would have

no error and the description of the three points would be exact, we would obtain the exact parameters that define the circle. The second one comes with the computational application; namely, when representing geometrical points at the programming language level, we operate with floating-point numbers, known for having limitations in terms of truncation and exact calculus.

Following the idea proposed by [Muñoz Bañón (2016)] and inspired by the calculations of lineal segments by [Cuadra Troncoso (2011)], we make use of the advantage of having seeds of more than three points. By having a larger data set, we compare the estimations done by combinations of different individuals of the seed. Throughout experimentation with different circles, this method will prove to be more reliable. However, the existence of outliers is still a concern.

Algorithm 2 Circle initial estimation algorithm based on three points calculations

```
estimate_circle_three_points(seed: [()]):
# Create an array of choices with every point from the seed
choices = [seed]
# For every combination of three extract three random points and remove from the choices
list
for all elements in choices do
  firstPoint = random(choices)
  choices.remove(firstPoint)
  secondPoint = random(choices)...
  # Calculate the estimation based on three points extracted and add to list
  estimation = estimate(firstPoint, secondPoint, thirdPoint)
end for
return mean(estimation)
```

Algorithm 2 shows a high-level implementation of the method for initializing the parameters of the circle. First, we create a list of possible points, obtained from the seed: we extract the points as triplets and remove them from the original set so that they are not reused. Then, for the calculation of parameters x_{center} and y_{center}, using the function *estimate*, we extract the unknowns from the system of equations 4.2, as follows:

$$x_c = \frac{\left(x_2^2 - x_1^2 + y_2^2 - y_1^2\right)(y_2 - y_3) - \left(x_2^2 - x_3^2 + y_2^2 - y_3^2\right)(y_2 - y_1)}{2\left((x_3 - x_2)(y_2 - y_1) - (x_1 - x_2)(y_2 - y_3)\right)}$$

$$y_c = \frac{x_2^2 - x_1^2 + y_2^2 - y_1^2 + 2a(x_1 - x_2)}{2(y_2 - y_1)}$$

Then, the radius is calculated by equation 3.1 given the first point of the triplet of points defined above. Finally, we return the means of arrays formed by the parameters estimated in each iteration. The results of the first experiment, as displayed in figure 4.13, improves in comparison with what we had in figure 4.12. By computing the functions, given the seed extracted in the scale-space phase, we obtain:

$$\rho = 1.54,\ \theta = 1.6,\ R = 0.21$$

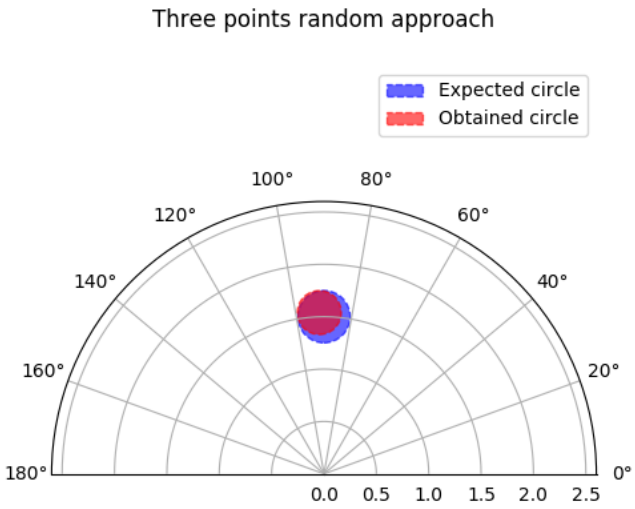

Figure 4.13: Comparison of expected circle and obtained one by function in algorithm 2.

The main advantage of the function proposed here is that we can execute the iterative process as many times as we want. Hence, we can obtain approximations by doing as many combinations as needed. Nevertheless, what this function does not have considered is the existence of outliers, that can lead to unexpected results. It is crucial to have a value, representing the confidence that we have on the fitting process. For instance, in the next section, we present the linear square regression method, where the *residual* can be an indicator of how good the model fits.

4.4.2. Least-squares Circle Fitting

The method described in the previous section does not deal with the presence of outliers. The estimation, based on randomly chosen points, could be interpreted as a hard-coding function. Ideally, supposing that we obtain the points with no associated error, the geometrical approximation would be satisfactory for estimating the parameters of the circle. On the contrary, we contemplate an experimental situation with the need for a more statistical approximation. In [Bullock (2017)] the author proposes a method for estimating the circle parameters in $\mathbb{R}^2$.

Formally, given a set of points in $\mathbb{R}^2 \rightarrow \{(x_i, y_i) \,|0 \leq i < N\}$, we look for the circle that fits the best in the points. We define x and y:

$$\overline{x} = \frac{1}{N}\sum_i x_i \quad \text{and} \quad \overline{y} = \frac{1}{N}\sum_i y_i$$

Also, we define $u_i = x_i - \overline{x}$ and $v_i = y_i - \overline{y}$. The problem is, after solving the first in the coordinates (u, v), transform it to (x, y).

The circle we want to estimate has center (u_c, v_c) and radius R. Following the philosophy of the least-squares algorithm, we minimize the square residual given by:

$$S = \sum_i \left(g\left(u_i, v_i\right)\right)^2$$

where $g(u, v) = (u - u_c)^2 + (v - v_c)^2 - \alpha \rightarrow \alpha = R^2$.

In order to accomplish the minimization, we have to obtain the partial derivatives of $S(\alpha, u_c, v_c)$:

$$\begin{aligned}\frac{\partial S}{\partial \alpha} &= 2\sum_i g\left(u_i, v_i\right)\frac{\partial g}{\partial \alpha}\left(u_i, v_i\right) \\ &= -2\sum_i g\left(u_i, v_i\right)\end{aligned}$$

The partial derivative of α, $\dfrac{\partial S}{\partial \alpha}$, equals to 0 if and only if:

$$\sum_i g\left(u_i, v_i\right) = 0 \tag{4.3}$$

Continuing with the partial derivative u_c, $\dfrac{\partial S}{\partial u_c}$:

$$\begin{aligned}
\frac{\partial S}{\partial u_c} &= 2\sum_i g\left(u_i, v_i\right) \frac{\partial g}{\partial u_c}\left(u_i, v_i\right) \\
&= 2\sum_i g\left(u_i, v_i\right) 2\left(u_i - u_c\right)(-1) \\
&= -4\sum_i \left(u_i - u_c\right) g\left(u_i, v_i\right) \\
&= -4\sum_i u_i g\left(u_i, v_i\right) + 4u_c \underbrace{\sum_i g\left(u_i, v_i\right)}_{=0 \text{ by equation } 4.3}
\end{aligned}$$

Thus, by equation 4.3, the derivative $\dfrac{\partial S}{\partial u_c} = 0$ if and only if:

$$\sum_i u_i g\left(u_i, v_i\right) = 0 \tag{4.4}$$

Similarly, $\dfrac{\partial S}{\partial v_c} = 0$ if and only if:

$$\sum_i v_i g\left(u_i, v_i\right) = 0 \tag{4.5}$$

Now, by expanding equation 4.4, we have:

$$\sum_i u_i \left[u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 - \alpha\right] = 0$$

We define $S_u = \sum_i u_i$, $S_{uu} = \sum_i u_i^2$, etc. and substitute in the above:

$$S_{uuu} - 2u_c S_{uu} + u_c^2 S_u + S_{uvv} - 2v_c S_{uv} + v_c^2 S_u - \alpha S_u = 0$$

Having $S_u = 0$, we simplify the previous equation to:

$$u_c S_{uu} + v_c S_{uv} = \frac{1}{2}\left(S_{uuu} + S_{uvv}\right) \tag{4.6}$$

By doing the same with equation 4.5, we obtain:

$$u_c S_{uv} + v_c S_{vv} = \frac{1}{2}\left(S_{vvv} + S_{vuu}\right) \tag{4.7}$$

Solving equations 4.6 and 4.7, we obtain (u_c, v_c) and, by expanding equation 4.3, we obtain the radius:

$$\sum_i \left[u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 - \alpha\right] = 0$$

$$(x_c, y_c) = (u_c, v_c) + (\overline{x}, \overline{y})$$

Having $S_u = 0$ and $S_v = 0$:

$$\alpha = u_c^2 + v_c^2 + \frac{S_{uu} + S_{vv}}{N} \tag{4.8}$$

where $R = \sqrt{\alpha}$.

The mathematical description of this method might seem complicated. However, in a higher-level description, we provide a model formula of the circle by equation 3.1 and then obtain the partial derivatives. Thus, when the derivatives are equal to 0, we minimize the square residual and find the estimation of the parameters.

The algorithm 3 shows that the implementation of this method is straightforward and, as observed through experiments, its performance is better than in the geometrical proposal (indicated in algorithm 2) in terms of time complexity. One notable disadvantage of this method is that the presence of outliers, due to the error associated with the signal, can provoke that the linear system (line 13) does not have a solution. In this case, we catch the exception and return a big symbolic value as residue. Thus, we clarify that the estimation has not obtained a solution.

$$\text{residue} = \sum_i (R_i - \overline{R_i})^2 \tag{4.9}$$

Algorithm 3 Circle estimation based on least-squares.

1: estimate_circle_least_squares($\mathbf{x}$, $\mathbf{y}$):
2: # Calculate mean of x and y
3: $\overline{x}$ = mean($\mathbf{x}$)
4: $\overline{y}$ = mean($\mathbf{y}$)
5: # Calculate u and v
6: $u = x - \overline{y}$
7: $v = y - \overline{y}$
   # Calculate the S components
8: Suv = $\sum(u * v)$
9: Suu = $\sum(u^2)$
10: Svv = $\sum(v^2)$
11: Suuv = $\sum(u^2 * v)$
12: ...
   # Solve system of equations of s components based on equations 4.6 and 4.7.
13: u_c, $v_c = solve\_equation(S...)$
14: $x_{center} = \overline{x} + u_c$
15: $y_{center} = \overline{y} + v_c$
16: $\mathbf{radius} = \sqrt{(\mathbf{x} - x_{center}^2) + (\mathbf{y} - y_{center}^2)}$
17: $\overline{radius} = mean(radius)$
18: $residue = \sum(\mathbf{radius}_i - \overline{radius})^2$
19: **return** ρ, θ, $\overline{radius}$, residue

Nevertheless, we define an indicator of the performance of the estimation, the so-called residue, calculated by equation 4.9. It is given by the distance from each point to the estimated centre, extracted from the mean of the distance from each point to the centre (line 18). In the code it makes us realize whether any outlier has diverged the estimation. Thus, a small residue would indicate a good estimation, and a big one would indicate that the estimation has been not prosperous. The terms big and small are, of course, relative and require a research for each particular experiment.

Having a fitting indicator represents a significant advantage in comparison to the geometrical method.

Let us consider an example with some points extracted from the seed, from section 4.2, and show the process of the previously shown calculations. The number of points N can be as large as we want.

From table 4.1 we have the averages $\overline{x} = -0.066$ and $\overline{y} = 1.39$. Also, we have $S_{uu} = 0.15$, $S_{uv} = -0.013$, $S_{uuu} = 0.0075$, $S_{vvv} = 0.00046$, $S_{vuu} = 0.009$, $S_{uvv} = -0.0038$.

ρ	θ	x_i	y_i	u_i	v_i
1.49	82	0.20	1.47	0.26	0.08
1.30	87	0.06	1.29	0.126	-0.1
1.33	95	-0.11	1.32	-0.04	-0.07
1.40	98	-0.19	1.38	-0.124	-0.01
1.54	101	-0.29	1.51	-0.224	0.12

Table 4.1: Least-Square Example table showing randomly picked points from the seed obtained in the first experiment, section 4.2.

Hence, from equations 4.6 and 4.7, we obtain the linear system as follows:

$$\begin{bmatrix} 0.15 & -0.013 \\ -0.036 & 0.0048 \end{bmatrix} \begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} 0.0036 \\ 0.14 \end{bmatrix} \tag{4.10}$$

Thus, we have $(x_c, y_c) = (-0.029, 1.54)$, and from equation 4.8 we calculate that $R = \sqrt{\alpha} = 0.24$. As we see, with such a small amount of points we obtain the estimation that is quite precise.

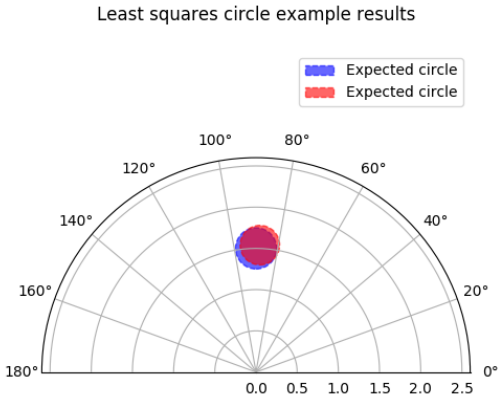

Figure 4.14: Comparison of expected circle and obtained one by least square method. The blue circle is the expected one, whereas the red circle is the obtained one.

The election of one of the two above proposed methods, namely, the geometrical approach and the least-squares circle fitting, is an essential step towards the further process of estimating circles. As we see later, the initial estimation affects the further estimation of the Extended Kalman filter significantly. To this end, in the next section, we present the initial estimation by both methods of four different circles. It might be representative when we apply the SLAM approach in the following chapter 5.

4.5. Initial estimation comparison

Each experiment shown in this section uses the approaches presented in the previous two sections, particularly, the geometrical approach and the least-squares circle fitting. For every circle proposed, there are 400 repetitions, and the results of the real values are compared. Moreover, the error added to the material has a normal distribution with $\sigma = 0.02meters$.

The first circle we consider is the one from the example of the first experiment, presented in figure 4.5. As we recall, the real parameters are: $\rho = 1.5$, $\theta = 1.5758$, and $R = 0.25$. The results are shown in table 4.2:

	$\Delta\rho$	$\Delta\theta$	ΔR
Geometrical (Min)	0.000264	0.001419	0.0028
Geometrical (Max)	1.10	0.93	0.62
Geometrical (Avg)	0.12	0.066	0.078
Least Squares (Min)	0.00024	0.00063	0.00024
Least Squares (Max)	0.0625	0.015	0.04422
Least Squares (Avg)	0.027	0.008	0.018

Table 4.2: Results for first experiment using both initial parameters estimation approaches. The factor $\Delta\rho$ indicates the error of the parameter rho, where as the $\Delta\theta$ indicates the error in of parameter θ and so the $\Delta radius$ the error of the radius. The section geometrical shows the results of applying the algorithm 2. The leas squares sections shows the results of applying the algorithm 3.

In table 4.2, showing the results of the first experiment, we already notice that for the minimum in the estimation, the error is more or less similar for both approaches, with a slight difference in favour of the least-squares method. However, in the least-squares method, the maximum gets reduced in a significant way. Hence, so does the average, proving that in this case, this approach is much more consistent.

The second circle we consider has different parameters with respect to the angle θ, specifically, the parameters are: $\rho = 1.41$, $\theta = 0.78$, and $R = 0.25$.

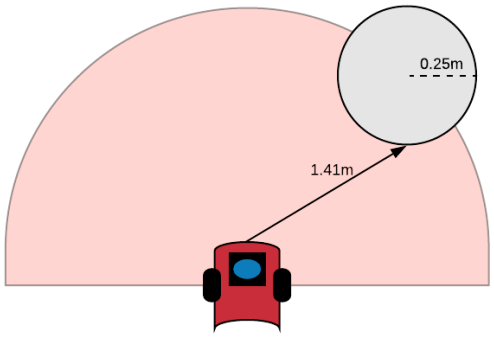

Figure 4.15: Schema of the second circle estimated

	$\Delta\rho$	$\Delta\theta$	ΔR
Geometrical (Min)	0.0019	0.0007	0.00039
Geometrical (Max)	1.25	0.95	1.29
Geometrical (Avg)	0.16	0.089	0.10
Least Squares (Min)	0.00061	0.0009	0.000240.011
Least Squares (Max)	0.090	0.022	0.096
Least Squares (Avg)	0.036	0.015	0.06

Table 4.3: Results for second circle estimated.

The results, presented in table 4.5, follow the same considerations as in the previous experiment. The maximums in the geometrical approach grow slightly due to the size of the seed provided by the scale-space. In the case of the first experiment, we have obtained a larger data set than in the second one.

In the third experiment, we also account those considerations. At the same time, we try to update the radius of the circle in order to obtain a better understanding of what happens when we experiment with the sizes. Therefore, we define the parameters of the third circle as follows: $\rho = 1.00$, $\theta = 0.0$, and $R = 1.00$.

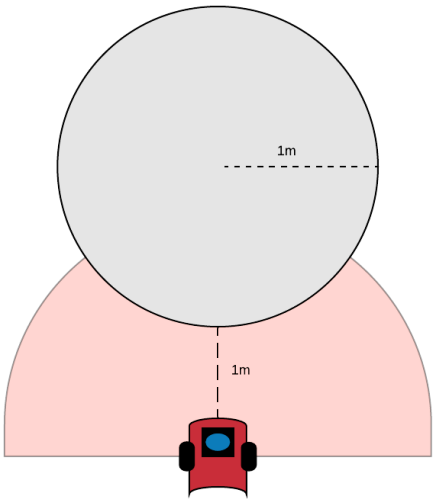

Figure 4.16: Schema of the third circle estimated.

In this case, in the geometrical approach, the maximum error grows. The existence of larger data in the seed means more probability of finding outliers that can deteriorate the estimation process. On the contrary, the estimation based on least-squares seems to provide better results, most likely because of the larger data set.

	$\Delta\rho$	$\Delta\theta$	ΔR
Geometrical (Min)	0.0009	0.00014	0.00028
Geometrical (Max)	2.46	0.92	3.33
Geometrical (Avg)	0.40	0.21	0.31
Least Squares (Min)	0.0000132	0.00138	0.024
Least Squares (Max)	0.07	0.012	0.10
Least Squares (Avg)	0.018	0.007	0.0638

Table 4.4: Results of the third experiment.

In the last, fourth experiment, we reduce the size of the circle and place it further from the robot. This experiment aims to see how the initialization behaves with a much smaller data set. The parameters of this column with respect to the robot are: $\rho = 4$, $\theta = 1.57$, and $R = 0.25$.

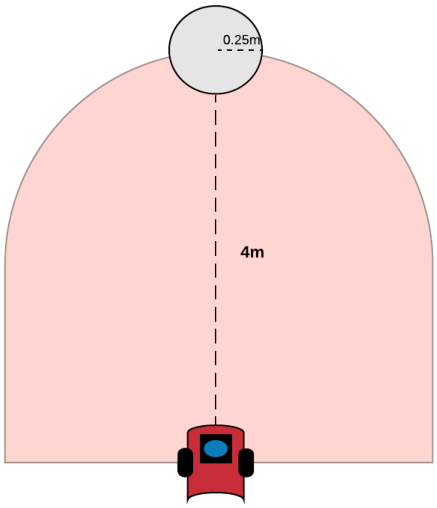

Figure 4.17: Schema of the fourth circle estimated.

The results are provided in table 4.5. The outcomes given by the geometrical proposal are far from desirable; the small data set can not give a right prediction of the distance and radius in the maximal error case. On the contrary, the least square method, even though it increases is minimums and maximums, still provides a better estimation.

	$\Delta\rho$	$\Delta\theta$	ΔR
Geometrical (Min)	0.0002	0.00004	0.00035
Geometrical (Max)	7.001	1	9.003
Geometrical (Avg)	0.37	0.06	0.39
Least Squares (Min)	0.000037	0.00130.0039	0.00010
Least Squares (Max)	0.2269	0.0207	0.14
Least Squares (Avg)	0.05	0.01	0.022

Table 4.5: Results of the fourth experiment.

As we have seen from the four experiments, presented in this section, the least-squares circle fitting copes better with the primary estimation task. There are two reasons to choose this method as the main approach for the initialization of the parameters. The first is the high performance and easy implementation, shown in algorithm 3. The second reason comes with the stability, despite the lower minimums provided by the geometrical approach in the experiments above, the least-squares fitting shows a better average and does not get to estimate such significant errors on either large or small data sets.

4.6. Extended Kalman Filter

In previous sections, we have defined the initialization of the algorithm. We have seen that the error and different aspects such as the position or the size affect directly on the estimation process. In section 3.3 we described a first implementation of the Kalman Filter proposed by Muñoz Bañón (2016). In this section, we will present a new proposal for the design of the Kalman Filter. This new proposal is inspired by the idea of using Kalman Filter for localization of the robot in Choset (2016) and Stachniss (2015). This implementation of the Kalman Filter imitates a localization problem on the data obtained from the laser rangefinder. In this respect, we understand the input $r(\phi)$ as the erroneous position to correct. While applying the filter, we adapt the parameters of the circle to a more consistent approximation with the received measures. In other words, the different points will be treated as if it were the robot that draws the circular trajectory, and this points where a GPS gives the position.

4.6.1. Kalman Filter design

Using words, it seems difficult to understand, that is why in figure 4.18 we show a visualization of the state of the Kalman Filter in progressive steps. In this figure we visualize the estimation of the circle described by the parameters $\rho = 1$, $\theta = 1,57$ and $radius = 0.25$.

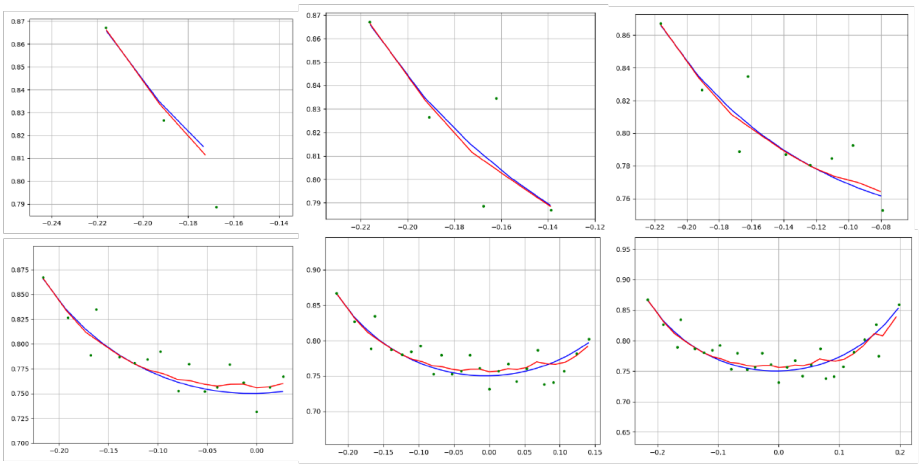

Figure 4.18: A visualization of the the different iterations occurring in the Kalman Filter. The blue line is the real circle, the red line is the estimation given by the algorithm. The green dots represent the inputs from the laser.

Figure 4.18 shows two crucial things to be noticed. The first one is that the size of the data set is essential when adjusting the parameters. Thus, a small data set will complicate the estimation process. The last reminds to the machine learning algorithms when more data for training might probably lead to better results. The second aspect to consider is the dependence of the filter with the input data from the laser(green dots). If we take a closer at image 3, we see that a bit more deviated point produces a significant change in the estimation, in this image and the following. Although, bit by bit, the estimation stabilizes at its last steps. Hence, the presence of outliers must be examined, to the point that we must ask ourselves if it is worth to discard these points. However, let us start with the formal definition of the Kalman filter that leads to this visualization. To accomplish this, we need to defined the different parts explained in section 2.4.2.

State variables, as we did before, the first step when implementing the Kalman Filter is to define which ones will be the state variables, this is, the state estimate vector:

$$\mathbf{x} = \begin{bmatrix} \mathsf{r} \\ \rho \\ \theta \\ \mathsf{R} \end{bmatrix} = \begin{bmatrix} r\rho\theta R \end{bmatrix}^\top \tag{4.11}$$

where r is the distance with the detected point at each iteration and the others at the polar parameters of the circle given in 3.2.

The process model, the transition function that defines how the dynamic system should change from one step to the next one, is the identity matrix of four elements. The parameters in the transition define the change. So we obtain the identity matrix.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The next step is to define the **measurement model**, the function h that calculates the expected values based on the estimated parameters follows the formula 3.2.

The Jacobian proposed is the same as in equation 3.6. Adding of course, a 1 at the beginning that is to represent the input r:

$$\mathbf{H}_n = \begin{pmatrix} 1 \\ \cos(\phi_n - \theta_{n-1}) + \frac{\rho_{n-1}\sin^2(\phi_n-\theta_{n-1})}{\sqrt{R_{n-1}^2-\rho_{n-1}^2\sin^2(\phi_n-\theta_{n-1})}} \\ -\rho_{n-1}\sin(\phi_n - \theta_{n-1}) - \frac{\rho_{n-1}\sin(\phi_n-\theta_{n-1})\cos(\phi_n-\theta_{n-1})}{\sqrt{R_{n-1}^2-\rho_{n-1}^2\sin^2(\phi_n-\theta_{n-1})}} \\ -\frac{R_{n-1}}{\sqrt{R_{n-1}^2-\rho_{n-1}^2\sin^2(\phi_n-\theta_{n-1})}} \end{pmatrix} \tag{4.12}$$

We must remember that the measurement has to be defined by the partial derivatives in order to be able to linearize the model; this is precisely the innovation proposed on the Extended Kalman Filter.

It only affects the r parameter in the state vector 4.11. Adding the noise into the matrix, $\mathbf{Q}$ smooths up the estimation of the parameter r and hence the estimation of other parameters [Wakar (2016)]. Finally, the assumed error on the initialization shown by the initial covariance matrix P_0 is the identity matrix, and we do not assume an error since the model is given directly by the parameters which are not correlated. Since we have assumed an error of $\sigma = 0.02$ in the inputs provided by the laser rangefinder, the noise associate to the measurement is:

$$\mathbf{Q} = \begin{bmatrix} \sigma_{\text{range}}^2 \end{bmatrix} = \begin{bmatrix} 0.02^2 \end{bmatrix}$$

and initial P_0:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now that we have defined the different components necessary for the estimation, we can show the process of the Extended Kalman Filter. The implementation is based on some examples of robot localization from literature, more specifically in [balzer82 (2018)].

As we can see in algorithm 4 the covariance calculation represents the main update. The other equations are similar to the generic ones shown in the last chapter. The new covariance matrix update proves to be more numerically stable than the generic on what observed through different experiments.

Algorithm 4 Kalman recursive function based on navigational problems.

def ekf(r_i,ϕ_i,$\mathbf{x}_{i-1}$):
#returns the Jacobean based on r and phi values from equation 3.6
$\mathbf{H}_i$ = Jacobean(r_i,ϕ)
calculate the residual
$\tilde{\mathbf{y}}_i = \mathbf{r}_i - H_i(x_i)$ # innovation covariance
$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i-1} \mathbf{H}_i^\top + \mathbf{R}_i$
kalman gain
$\mathbf{K}_i = \mathbf{P}_{i-1} \mathbf{H}_i^\top \mathbf{S}_i^{-1}$
update the state estimate
$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{i-1} + \mathbf{K}_i \tilde{\mathbf{y}}_i$
modified update covariance compare with equation 2.20
$\mathbf{P}_i \leftarrow \Big(\mathbf{I} - \mathbf{KH}) * \mathbf{P}_{i-1} * (\mathbf{I} - \mathbf{KH}^\top)\Big)$

4.6.2. Results

If we remember the generic description of the Extended Kalman Filter, specifically equation 2.14, we see that there is a component base on the control input u. One essential problem we have in what respects the circle fitting is that we do not possess any control input since the parameters are strictly defined from the model. During the project many approximations have been made trying to palliate this problem. A complication on the initialization of the parameters will lead to a presumably bad estimation of the circle. Figure 4.18 shows the estimation when the initial parameters are accurate with an error of ± 0.1(threshold observed after various experiments), this is 10 cm. However, when the error in the initialization grows, the Kalman estimation tends to worsen the results. Let us see what happens when we initialize the algorithm used to visualize figure 2.14 with a bad initial estimation. This means that if we have a real representation by parameters $\rho = 1$ $\theta = 1.5758$ and $R = 0.25$, we take an illustrative not correct initial estimation of $\rho = 0.87$ $\theta = 1.4$ and $R = 0.20$.

Figure 4.19 shows that the initial estimation tumbles a bit around the real circle(red line vs blue line). However, it stabilizes in the last image. The last suggests that if we would have more points provided by the readings and the seed, probably the estimation could be better. In many cases, the complex values, which result from the model equation 3.2, have no relation to the real circle.

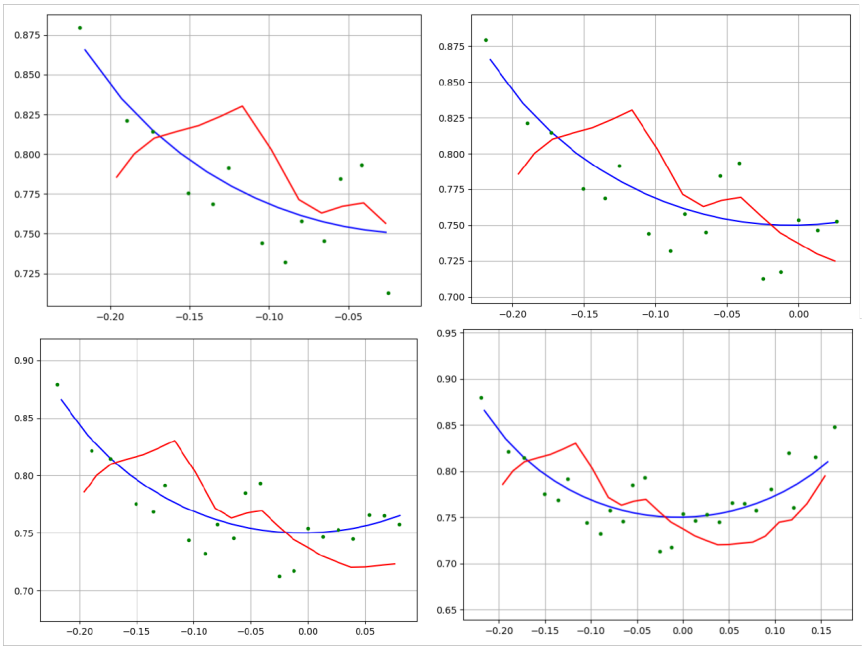

Figure 4.19: A visualization of the Kalman Filter with a wrong initialization

.

	$\Delta\rho$	$\Delta\theta$	ΔR
Least Squares (Min)	0.001	0.00002	0.0014
Least Squares (Max)	0.058	0.020	0.048
Least Squares (Average)	0.027	0.08	0.018
Kalman(Min)	0.0007	0.007	0.0013
Kalman(Max)	0.0083	0.040	0.12
Kalman (Average)	0.023	0.014	0.019

Table 4.6: Results of applying the Kalman Filter on first experiment 4.5. The section on the top shows the calculations performed by the leas-squares method. The section below the results obtained in the Kalman estimation.

Table 4.6 seems to indicate what explained in previous paragraphs. As already said, it seems the Kalman Filter works properly within a certain threshold of the initial estimation. If the initial estimation does not provide a correct approximation, the filter does not provide correct results. If, on the contrary, the initial estimation correctly approximates the real circle, the filter does not represent a real improvement. Lastly, there are some intermediate values of the initial estimation, where the Kalman Filter tends to improve the initial estimation. The three left experiments illustrate the same tendency.

	$\Delta\rho$	$\Delta\theta$	ΔR
Least Squares (Min)	0.003	0.000020	0.00060
Least Squares (Max)	0.076	0.027	0.092
Least Squares (Average)	0.036	0.08	0.06
Kalman(Min)	0.000222	0.006	0.0311
Kalman(Max)	0.067	0.1	0.099
Kalman (Average)	0.040	0.02	0.03

Table 4.7: Results for the second experiment

	$\Delta\rho$	$\Delta\theta$	ΔR
Least Squares (Min)	0.00014	0.00675	0.02917
Least Squares (Max)	0.05	0.0270	0.095
Least Squares (Average)	0.017	0.013	0.062
Kalman(Min)	0.00023	0.006	0.0311
Kalman(Max)	0.07	0.10.017	0.0990
Kalman (Average)	0.002	0.020.013	0.06

Table 4.8: Results for the third experiment.

	$\Delta\rho$	$\Delta\theta$	ΔR
Least Squares (Min)	0.0006	0.01121	0.0007
Least Squares (Max)	0.19	0.025	0.12
Least Squares (Average)	0.06	0.016	0.017
Kalman(Min)	0.0009	0.01	0.00005
Kalman(Max)	0.19	0.025	0.12
Kalman (Average)	0.059	0.016	0.017

Table 4.9: Results for the forth experiment.

Figure 4.20 shows the tendency of the errors in the parameters of the circle, when the initial estimation has an error of approximately 0.1 for each parameter. As explained before, the Kalman Filter tends to improve the solution when the initial estimation is not so precise. On the contrary, when the initial estimation improves the kalman filter tends to worsen the results, see figure 4.21.

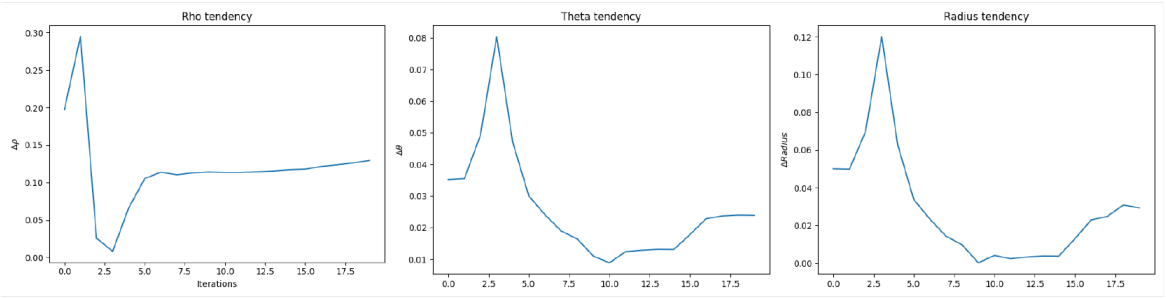

Figure 4.20: Visualization of good tendency of the parameters during kalman iteration, when the initialization has an error of approximately ± 0.1. The left image shows the tendency follow by parameter ρ, the image in the middle the parameter θ and the last image parameter $radius$

.

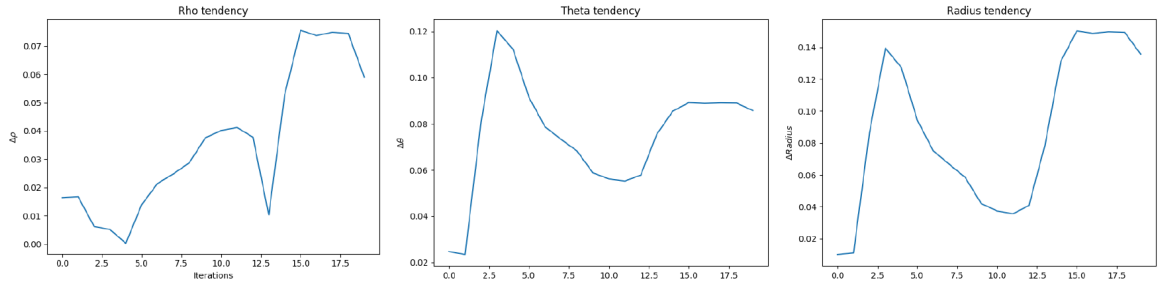

Figure 4.21: Visualization of a bad tendency of the parameters during Kalman iteration. When initialization has precise.

.

4.6.3. Analysis

There are four significant outcomes that we can extract from the analysis performed on different experiments. These outcomes serve as a summary of the ideas commented in this chapter, and also they might be useful for future research.

- The Kalman Filter is a recursive process that requires an initialization of the parameters that form its model. This initialization has a crucial influence on the final results obtained by the filter. In this chapter, we have provided a method, which average errors show that it procures reasonable stability.

- In many applications the Kalman Filter obtains data as a control input, and this is useful to provide stabilization of the filtering process. Without a control input, the filter depends primarily on the measurement function; this provokes that the presence of outliers might worsen the estimation considerably. A possible solution for this is to use more than one sensor and apply data fusion techniques in order to palliate the effect of the noisy inputs with another source of measurement.

- The detection of breakpoints, this is , the limits of the circular segments is a crucial step. The presence of data, which does not belong to the segment we are estimating, worsens the results of the application.

- The last and most important aspect refers to the equation used by the model. The polar representation of the circle contains a square root that, on some occasions, gives a complex number as a solution. This issue has two undesirable effects on the performance of the filter. The first one is that, in some cases, the calculations based on complex numbers has its limitations from a computational point of view. Moreover, the apparition of complex values can lead to indeterminacy in the calculations of the covariance matrix. The solution we have provided for this is, as commented, to modify the generic formula which updates this matrix.

The last point commented may have a solution with a profound modification of the filter design. Thus, updating the model and basing it on the Cartesian representation can represent a solution [Nixon (1993)]. Another possible approach is to use the Unscented Kalman Filter. The last shows a more stable approximation in many examples of the literature. We have tried to provide a solution using these methods. However, the mathematical knowledge required proved too high.

Now that we have analyzed the performance of the Kalman Filter. We can take a step forward and consider what would happen if we allow the robot to move. That is the reason why in the next chapter, we present two experiments where circular objects, located around the scene, need to be estimated by the robot.

Chapter 5

SLAM based Circle Fitting

In the previous chapter, we have seen an approximation of how we can perform the estimation of a circle. However, until now in the experiments, we have considered only a static situation, where the robot does not move, and there is solely one column. In real life, we want to be able to approximate circular elements, such as columns and trees, in a landscape environment. Aiming to follow the philosophy of adding more complexity in each step, in this chapter, we show how does the algorithm, presented in the previous chapter, behave in a SLAM problem. First of all, we present the scenes or worlds for the experiments. Second, we explain the navigation algorithm used for the robot to move around and explore, following by an explanation of the two methods used for estimating the position of the robot. The next step is the most representative one: we proposed a process for estimating and creating a map composed by circular objects. Finally, we show the results of the estimation for the two scenes proposed.

5.1. Scenes disposition

The scenes, proposed in this chapter, are built in the simulator V-REP. This simulator provides an intuitive user interface. Moreover, it allows us to create different basic shapes, e.g. cylinders, from scratch.

At first, we consider an outdoor scene. The need for estimating circular segments appears in situations where we have trees of different sizes. Not only terrestrial robots but also aerial ones find themselves in the circumstances, where recognizing specific patterns in

the readings is necessary. Luckily, in this experiment, we will consider only the two dimensions, x and y, of the Cartesian system, as we have been doing throughout the paper.

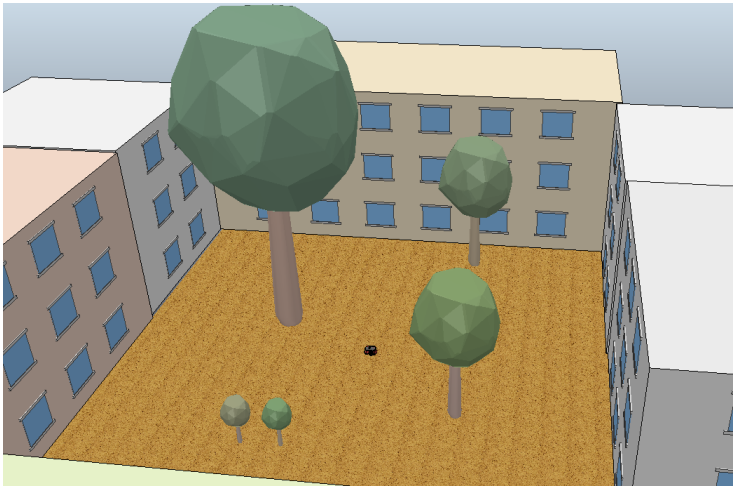

Figure 5.1: The robot in the backyard with a tree set up. The different trees are located to cover some observed corner cases.

The figure 5.1 shows a simple disposition of trees in the landscape. Nevertheless, the trees are distributed in a way that corner cases, appeared during the design process, can be tested. The attentive reader might have realized that the size of trunks does not change much from one tree to another. The reason we do not add thick trees is clarified later in section 5.4.4. Although, in real situations, broad circular segments should be expected. A necessary characteristic of trees model provided by V-REP is that the trees do not draw a perfect circular shape at the bottom of there trunks, which has to be considered since the data obtained by the laser rangefinder does not represent a real circle.

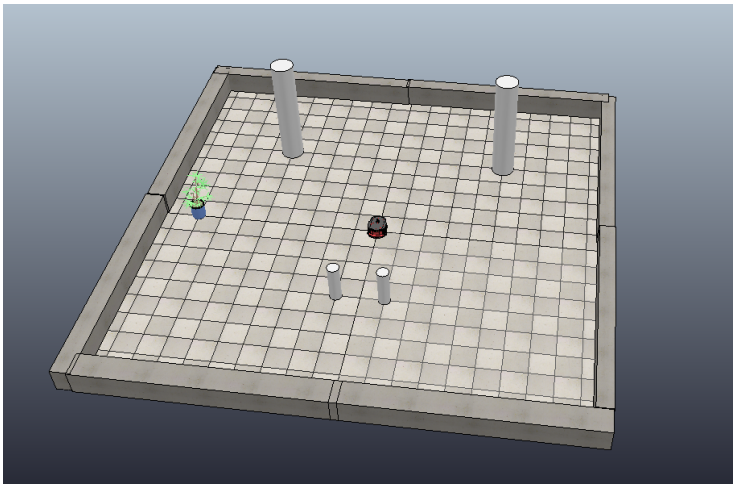

Figure 5.2: The robot in the office with a column set up.

The next scene, shown in figure 5.2, represents an indoors office with a column set up. On the contrary of what happened with the trees, the column models are primitive shapes based on cylinders [coo (2009)]. It means that the error, associated with the readings, is not so high, hence, we expect better approximation. Once again, the presented columns and an indoor plant are not disposed randomly but have a meaning, related to the classification of features and breakpoints detection, that are discussed further in this chapter. Although the second scene is not a realistic setup for an office, adding more rectangular elements, such as tables or chairs, leads to a problem of splite classification.

5.2. Navigation

Once we have described the disposition of both scenes, we explain the algorithm we use in order to make the robot explore the surroundings. The aim of this experiment is not the navigation algorithm that rules the direction of the robot. However, we need a simple navigational model that can avoid the obstacles and wander around in order to collect the different features to be analyzed. In this respect, in this chapter, we consider two navigation approaches.

5.2.1. Braitenberg algorithm

The first algorithm we consider to make the robot move around is an algorithm that in the simplest form is purely reactive (see section 2.2.2). This algorithm is designed by the neuroscientist Valentino Braitenberg, and it is based entirely on the reactive paradigm. This approach is used for obstacle avoidance. The input from the rangefinder sensors affects the movement of the agent directly [Gochev and Nadzinski (2014)].

Formally, we describe an array of weights, with the length of the array of sensors we have. The measured distance from each sensor affects the motor velocity proportionally. Hence, as the distance is getting smaller with respect to a specific obstacle, the wheel on the same side accelerates depending on the weights, assigned in order to avoid the obstacle. The velocity of each motor is updated by the following equation:

$$\overline{v}_m = \mathbf{v}_m + w_c \cdot \overline{d}$$

where $\mathbf{w}_c$ is the weight coefficient assigned to the sensor that provides normalized data, $\overline{\mathbf{d}}$ and $\mathbf{v}_m$ references the velocity of the motor we update.

In our case, the robot PIONEER P3DX has by default sixteen ultrasonic sensors. Therefore, we need an array of sixteen weights for each of two engines situated in the left and right wheels. The value, given to this array of weights, determinates how slow or fast the avoidance is. Moreover, we define a maximum detection distance for the robot not to seem oscillatory.

The application of the Braitenberg algorithm has the advantage of being easily programmed and lightweight. On the other side, the disadvantage is that the robot randomly wanders around avoiding objects.

5.2.2. Graph based approach

The Braitenberg algorithm works correctly for obstacle avoidance. However, it does not set a specific path for the robot. We need the robot to move in a more determinate way.. Thus we indicate the robot which parts of the world it has to visit.

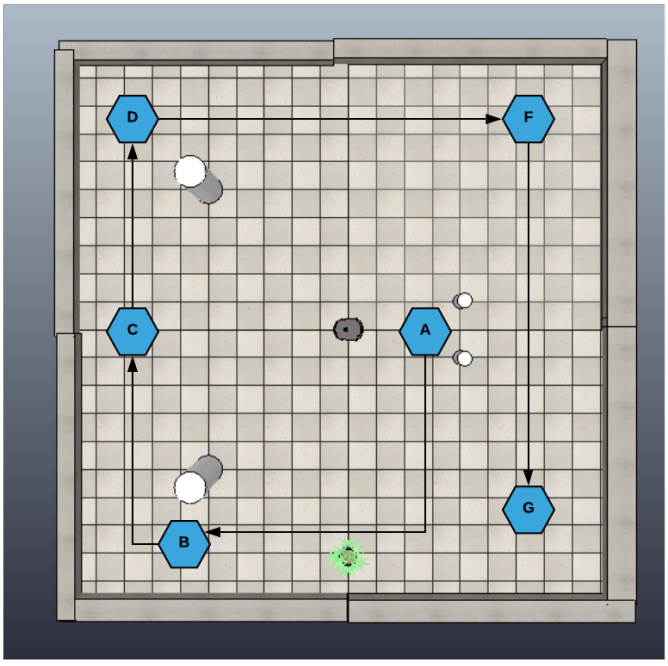

Figure 5.3: The graph representing the path the robot follows during the indoor experiment. The robot follows the alphabetical order of the nodes.

In figure 5.3 we see the navigational path the robot takes when the experiment is running. A graph is defined with different nodes in alphabetical order. The robot will follow those nodes accomplishing a successful overview of the scene. Of course in order to obtain this graph we need previous knowledge and the close world assumption, which exits in the hierarchical paradigm.

In order to accomplish the movement of the robot from one node to another we need three essential functions.

1. A function that returns to the robot distance to the next node. By the distance, we mean the euclidean distance between the point (node) and the robot location:

$$\text{distance} = \sqrt{(\text{node}_x - \text{robot}_x)^2 + (\text{node}_y - \text{robot}_y)^2}$$

   The positions of the robot in this and the next function are obtained from the simulator and are absolute, meaning that there is no error associated to both measures, i.e. the position of the robot and the node.

2. The angle θ the robot has with respect to the node is obtained as:

$$\theta = \arctan \cdot 2(\frac{\text{node}_y - \text{robot}_y}{\text{node}_x - \text{robot}_x})$$

   The calculation uses an extended function of the trigonometrical inverse for the tangent called *atan2* [architectpianist].

3. A function that relates the angle with the velocity of the wheels is proposed in the following. We need to accelerate one of the wheels in order to make the robot orientate directly to the node. Once the angle they form is between certain threshold, 0 to 20 degrees in this case, we set the two wheels to the same velocity. Thus, the robot will move directly towards the node.

Although the proposed navigation system is straightforward, the aim of this paper it is not to study the different navigation models that could be applied. We have implemented a potential field based algorithm [Siddiqui (2018)]. However, the performance requires by this algorithm interfered with more important aspects of the experiment.

Nonetheless, the combination of the graph approach with the Braitenberg obstacle avoidance is sufficient for the goals of the experiment. This combination sets the navigation approach into the hybrid paradigm from section 2.2.3.

5.3. Localization

Once we have set up experiments with a moving robot, as the next step, we move on to the first part of the SLAM, particularly, the localization. In this section, we show three techniques to obtain the absolute position of the robot. The first one is based on the Odometry, explained in section 2.3.2. The second one tries to solve the errors of the first approach by adding a GPS. Finally, the last approach we discuss is the localization for indoors experiments with the Extended Kalman filter.

5.3.1. Dead reckoning

The first approach proposed for localizing the robot is based on the information obtained by the encoders of the wheels [Maderia Cardoso (2017)]. Each wheel has an encoder that can be used to extract its velocity by the next equation:

$$V = \frac{\Delta\theta}{\Delta\,\mathrm{time}} \cdot R \tag{5.1}$$

where $\Delta\theta$ is the increment in the angular position of the encoder, $\Delta time$ is the time increment, and R is the radius of the wheels. In our case $R = 0,0975$m, as determined in figure 4.1.

Once we obtain the velocity of both wheels of the robot, we can calculate its linear and angular velocity by the following equations:

$$V = \frac{V_r + V_l}{2}$$
$$\omega = \frac{V_r - V_l}{D}$$

where V_r and V_l are the velocity of the left and right wheel respectively, and D is the axis of the robot. In our case $D = 0,381$m (see figure 4.1).

In this respect, we define the state vector of the robot given by its position in the Cartesian coordinates and its bearing:

$$\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{5.2}$$

Now we have to relate the information about the velocity obtained in equation 5.1 with the position and bearing angle.

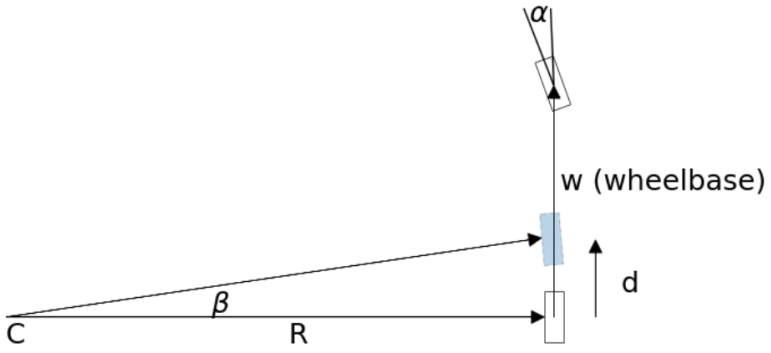

Figure 5.4: The disposition of the wheels showing mathematical angles involved [Wakar (2016)]. Alpha is the angle the wheel draws when turning.

In figure 5.4 we see a front wheel bearing with an angle α. Having θ, which is the robot absolute orientation, we can define the position x and y before the robot turns as:

$$\begin{aligned} x_{t-1} &= x - R\sin(\theta) \\ y_{t-1} &= y + R\cos(\theta) \end{aligned}$$

And after the robot turns, the new state vector is given by:

$$\begin{aligned} x_t &= x_{t-1} + \left(\Delta s \cdot \cos\left(\theta_{t-1} + \tfrac{\Delta\theta}{2}\right)\right) \\ y_t &= y_{t-1} + \left(\Delta s \cdot \sin\left(\theta_{t-1} + \tfrac{\Delta\theta}{2}\right)\right) \\ \theta &= \theta_{t-1} + \Delta\theta \end{aligned} \tag{5.3}$$

where the component Δs is:

$$\Delta s = \frac{V_r \cdot \Delta t + V_l \cdot \Delta t}{2} \tag{5.4}$$

and $\Delta\theta$:

$$\Delta s = \frac{V_r \cdot \Delta t + V_l \cdot \Delta t}{2 \cdot \mathrm{robot}_{axis}} \tag{5.5}$$

In both equations above, Δt is the increment of time between one calculation and

the next. In our experiments $\Delta t = 0,1$ seconds.

Having the motion equations defined, we can already develop the algorithm that will give us the consecutive positions of the robot. The implementation has to have in account that the transformation of angles might be incorrect. A critical aspect from equation 5.1 is that the sign of the velocity depends directly on the increment of the angle of the encoder. If the absolute angle is greater than π, then the encoder has changed the spin direction (it occurs when the robot is turning).

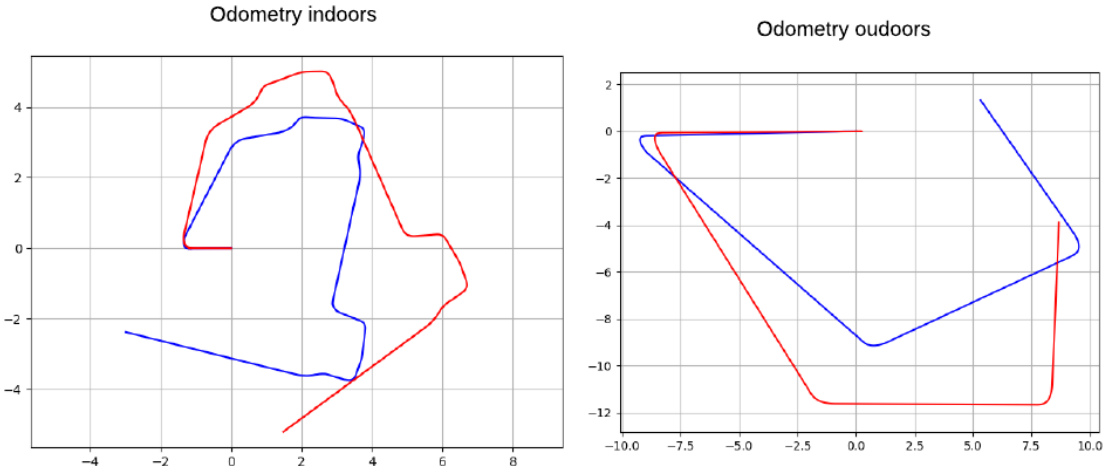

Figure 5.5: The results of dead reckoning for indoor scene(left) and outdoor scene(right). The blue line shows the real path travelled by the robot. The red line the path estimated with dead-reckoning.

Figure 5.5 shows the results of applying the previous calculations to the scenes described in section 5.1. The blue line represents the real path travelled by the robot, and the red one is the path travelled according to the dead reckoning. The robot has travelled for thirty seconds, considering $\Delta t = 0.1$.

The problem of this approach is that the small errors accumulate in each step of the execution become larger and larger. Since no control input whatsoever is given, the error continues to grow until the end of the execution. The error in odometric calculations can occur due to many factors. In this experiment, we have no bumps or discontinuities on the floors, but there is an error associated with the encoder position estimation.

5.3.2. GPS based Extended Kalman filter

Continuing with the localization of the robot, we present a solution for the outdoors experiment. In this section, we explain the modelling using the Extended Kalman filter, from which we normalize and filter the noisy data, obtained from a GPS so that we can extract a more accurate position of the robot. This approach, described in [Sakai (2019)], adapts well to the needs and description of the outdoors experiment.

Let us start by defining the state vector that represents the estimation progress in each iteration of the filter. We do this by adding the velocity to the equation 5.2:

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & v_t \end{bmatrix}^T \tag{5.6}$$

The next step is to define the motion model, which describes the transition from one position to the next one:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$$

With respect to $\mathbf{u}_t$, we have the control input:

$$\mathbf{u}_t = [v_t, \omega_t]$$

where v is the speed of the robot provided by the speed sensor and the gyro sensor that gives us information about the orientation (this value could also be provided by the odometry).

The matrix B that relates the control input with the state transition is defined as:

$$\mathbf{B} = \begin{bmatrix} \cos(\phi)dt & 0 \\ \sin(\phi)dt & 0 \\ 0 & dt \\ 1 & 0 \end{bmatrix}$$

Now, as we explained in section 2.4.2, the EKF linearizes the non-lineal equations by calculating the Jacobian matrix:

$$\mathbf{J_F} = \begin{bmatrix} \frac{dx}{dx} & \frac{dx}{dy} & \frac{dx}{d\phi} & \frac{dx}{dv} \\ \frac{dy}{dx} & \frac{dy}{dy} & \frac{dy}{d\phi} & \frac{dy}{dv} \\ \frac{d\phi}{dx} & \frac{d\phi}{dy} & \frac{d\phi}{d\phi} & \frac{d\phi}{dv} \\ \frac{dv}{dx} & \frac{dv}{dy} & \frac{dv}{d\phi} & \frac{dv}{dv} \end{bmatrix}$$

One may notice that each (i, j) component of the matrix J_F describes the partial derivative of the components of equation 5.6. The result is then:

$$\mathbf{J}_F = \begin{bmatrix} 1 & 0 & -v\sin(\phi)dt & \cos(\phi)dt \\ 0 & 1 & v\cos(\phi)dt & \sin(\phi)dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To predict the covariance $\mathbf{P}$, we use the generic equation 2.15.

Now let us define the observation model. As we have already mentioned, the measurement is guided by a GPS which provides us with an input:

$$\mathbf{z}_t = [x_t, y_t]$$

Having the aforementioned in mind, we define the observation equation:

$$\mathbf{z}_t = H\mathbf{x}_t$$

where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The reason we define the matrix H in this way is transparent; since we get direct input of the state vector values x and y, the function returns these same values. Therefore, we set to 1 the positions of these variables in the matrix H.

Again, we have to calculate the Jacobian defined by the partial derivatives:

$$\mathbf{J}_H = \begin{bmatrix} \frac{dx}{dx} & \frac{dx}{dy} & \frac{dx}{d\phi} & \frac{dx}{dv} \\ \frac{dy}{dx} & \frac{dy}{dy} & \frac{dy}{d\phi} & \frac{dy}{dv} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Now, having defined the three main components of the filter, we need to provide the noise matrices $\mathbf{R}$ and $\mathbf{Q}$. The error matrix $\mathbf{Q}$ associated to the prediction of the covariance matrix $\mathbf{P}$ in our case is:

$$\mathbf{Q} = \begin{bmatrix} 0.4 & 0.4 & 0.01 & 1 \end{bmatrix}^T$$

Each defined value matches the expected error of the state estimate vector in equation 5.6. We take the matrix R as :

$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}^2$$

Finally,as we have completed defining the components of the Extended Kalman filter. The recursive process follows the generic equations and phases shown in figure 2.8. We initialize the filter with the initial position of the robot: $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & \pi & 0 \end{bmatrix}^T$ and the initial covariance matrix $\mathbf{P}_0$, which is a 4×4 size identity matrix.

Figure 5.6 shows the path travelled by the robot calculated and filtered by the Kalman filter designed above. The ellipse drawn at the end of the path is the **covariance ellipse**. As explained in [Spruyt (2015)], it allows us to visualize a $2D$ confidence interval. In other words, it shows the region, where a major percentage of the values can be drawn by the underlying Gaussian distribution.

The different noise distributions, shown in figure 5.6 are added to remark how important is to define the noise matrices $\mathbf{R}$ and $\mathbf{Q}$ for each experiment. We will comment more on the importance of tweaking the small components of the filter in the conclusive chapter.

If we compare figures 5.6 and 5.5, we see that the approximation of the first one is better (even when the simulated error is higher). However, GPS approximation is not exactly realistic. In many cases, the GPS connection signal is weak or not available at all. This example shows an easy approximation of one of the most common usages of the Kalman Filter, that is the GPS error adjustment [Katkov (2018)].

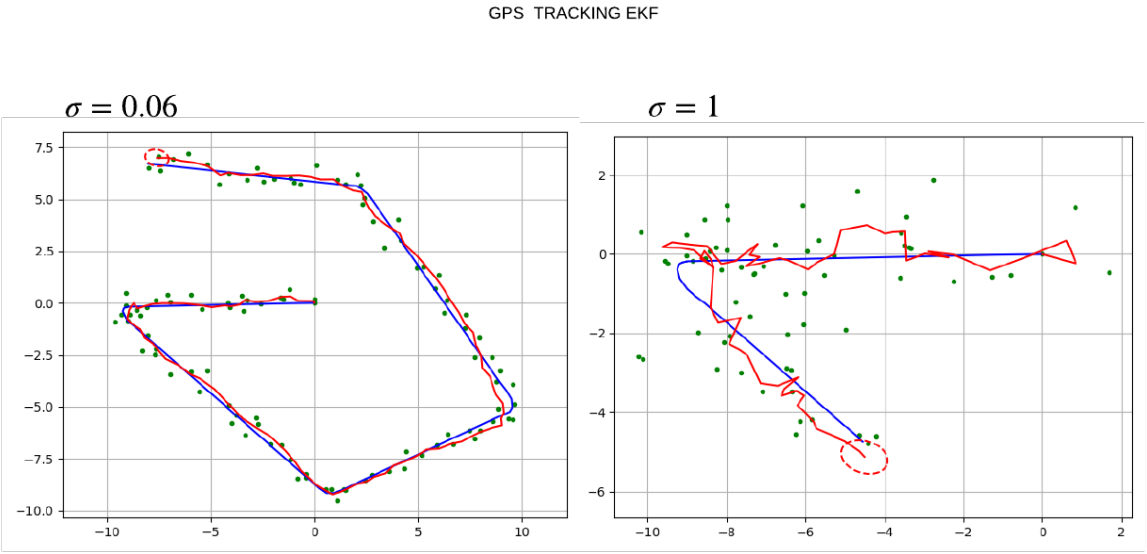

Figure 5.6: The results of GPS for the outdoors scene. The blue line representing the real travelled path. The red line represents the estimation done by the Kalman Filter. The green dots are the data provided by the GPS. In the left image the GPS has an error with a deviation $\sigma = 0,06$. In the right image this deviation is increased to $\sigma = 1$.

5.3.3. Odometry based Extended Kalman filter

In the previous section, we assumed having a GPS chip inserted in the robot. Although this is imaginable in outdoors experiments, the indoors experiments are usually based on odometric calculations. Also, in the previous section, we have seen that Kalman is an efficient approach when combining sensor data (the velocity and orientation with GPS information). In this section, we take a step further to a more realistic approach. We estimate the path travelled by the robot indoors using just one landmark. A landmark is a known object in the experiment that, when detected by some robot sensor, can be used as measurement input to improve the odometric calculations. We follow the explanations from [Pinheiro and Cardozo (2014)] to design and use an Extended Kalman filter that, based on the odometry and the position of the landmark, estimates the travelled path.

Again, let us start with the definition of the state vector. We use the same vector as in equation 5.3. We can model the robot movement as a nonlinear motion model with associated noise.

$$\mathbf{x}_{i+1} = \mathbf{F}\mathbf{x}_i + \mathbf{w}_i$$

To extend this, we have to recover the equations 5.3 from dead reckoning.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_i = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \Delta s \cdot \cos\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) & \Delta s \cdot \sin\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) & \Delta\theta \end{bmatrix} \tag{5.7}$$

For designing a non-linear model, compute the Jacobian of the equation 5.7:

$$\mathbf{F} = \frac{\partial f(x,u)}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s_t \sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{\partial_t}\right) \\ 0 & 1 & \Delta s_t \cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}$$

where, Δs and $\Delta\theta$ denote the same as in equations 5.4 and 5.5, respectively. As for now, we observe that the design of the system is based on the dead-reckoning formulations.

We define the the control input, in this case given by the vector u, as:

$$\mathbf{u} = \begin{bmatrix} v \\ \alpha \end{bmatrix}$$

where v is the velocity of each wheel and α is the robot steering. Both are needed for the calculation of $\Delta\theta$ and Δs.

We have defined how we predict the state estimate vector. In the next step we describe how we update the covariance matrix P:

$$\overline{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{V}\mathbf{M}\mathbf{V}^T \tag{5.8}$$

In equation 5.8 we have a new set up comparing to the equation 2.15. It is because our motion model is non linear, so we linearize $\mathbf{F}$ by providing the Jacobian with respect to the vector $\mathbf{u}$:

$$\mathbf{V} = \frac{\partial f(x,u)}{\partial u} \begin{bmatrix} \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \alpha} \\ \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \alpha} \\ \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \alpha} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}\cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) - \frac{\Delta st}{2Robot_{axis}}\sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) & \frac{1}{2}\cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) + \frac{\Delta st}{2Robot_{axis}}\sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) \\ \frac{1}{2}\sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) + \frac{\Delta st}{2Robot_{axis}}\cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) & \frac{1}{2}\sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) - \frac{\Delta st}{2Robot_{axis}}\cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right) \\ \frac{1}{2Robot_{axis}} & -\frac{1}{2Robot_{axis}} \end{bmatrix}$$

Now we define the matrix $\mathbf{M}$ that describes the noise associated to the control input in equation 5.8:

$$\mathbf{M} = \begin{bmatrix} \sigma_{vel}^2 & 0 \\ 0 & \sigma_{\alpha}^2 \end{bmatrix}$$

We have defined the system model, that will serve for the predicting phase of the filter. Taking a closer look, we see that the system model is based on the dead reckoning information. For now, we have not added a real improvement to the design, except the noise matrix.

In this respect, we need to design the measurement model. As it has been already said, the measurement model works by detecting a group of one or more landmarks that can be used to reference and update the knowledge of the robot position. In this experiment, we will use just one landmark, which seems to be enough for a correct estimation of the travelled path. This assumption holds because of the small size of the room shown in figure 5.2. Moreover, the fact of just having one landmark means we have less complexity and needless computational performance.

The measurement input is given by the range and baring from the landmark, thus:

$$\begin{aligned} \mathbf{z} &= h(\overline{\mathbf{x}}, \mathbf{p}) \quad + \mathcal{N}(0, R) \\ &= \begin{bmatrix} \sqrt{(L_x - x)^2 + (L_y - y)^2} \\ \arctan 2\left(\frac{L_y - y}{L_x - x}\right) - \theta \end{bmatrix} \end{aligned} \tag{5.9}$$

where L_x and L_y are the Cartesian coordinates of the landmark detected by the sensors. The first row calculates the range and the second the bearing from the robot to the landmark. Since we are in non-linear model, we need to compute the Jacobian of the function above:

$$\mathbf{J}_h = \begin{bmatrix} -\frac{L_x - x_t}{\sqrt{(L_x-x_t)^2+(L_y-y_t)^2}} & -\frac{L_y - y_t}{\sqrt{(L_x-x_t)^2+(L_y-y_t)^2}} & 0 \\ \frac{L_y - y_t}{(L_x-x_t)^2+(L_y-y_t)^2} & -\frac{L_x - x_t}{(L_x-x_t)^2+(L_y-y_t)^2} & -1 \end{bmatrix}$$

The robot will follow the path based on the odometric parameters of the system model. However, once a landmark is detected, the state estimate will be updated accordingly to its relative position to the landmark. The error matrix R associated to the measurement model, this is, the laser readings is in our case the same supposed in previous experiments:

$$\mathbf{R} = \begin{bmatrix} \sigma^2_{range} & 0 \\ 0 & \sigma^2_{bearing} \end{bmatrix} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.01 \end{bmatrix}$$

The error is associated with the range captured by the sensors and the bearing of the detection. Having defined the necessary components, we start the process shown in figure 5.7.

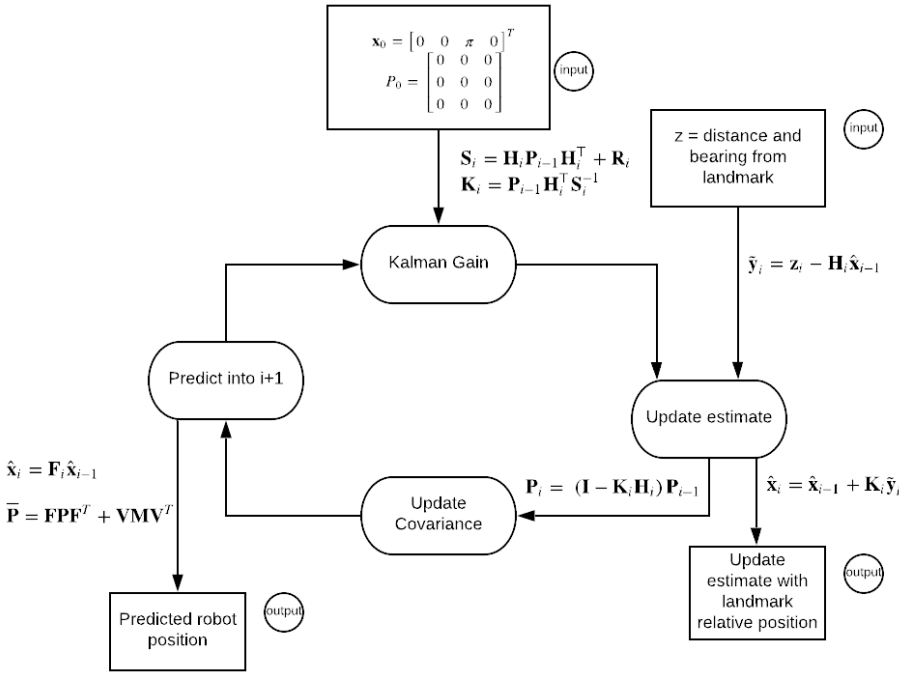

Figure 5.7: Process for the odometry and landmarks based Kalman Filter.

The figure 5.7 shows that we initialize the process by the initial position of the robot, given as a vector $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & \pi & 0 \end{bmatrix}^T$. The resulting path of applying the estimation to the indoor scene is shown in figure 5.8.

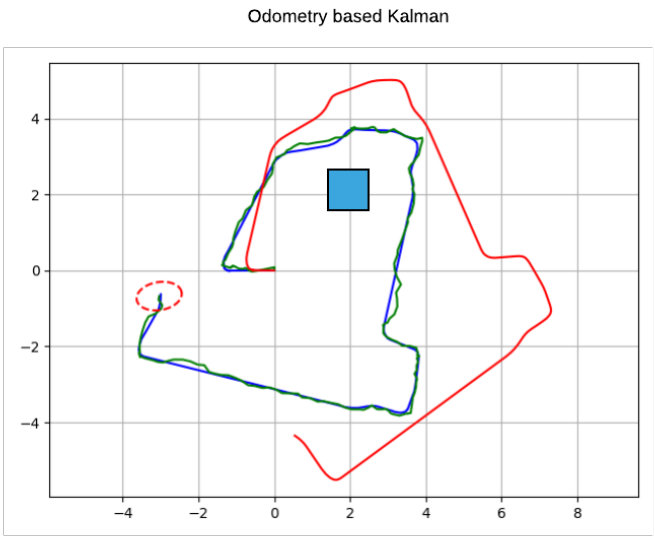

Figure 5.8: Visualization of the estimation of the path taken by the robot in this last Kalman approach. The blue line, as before, represents the real path travelled by the robot, the green line is the Kalman estimation of the travelled path, and the red line represents the odometric calculations. The blue square is the position of the landmark $(L_x = 2, L_y = 2)$.

In our case, the landmark is virtual, meaning we give simulated values in the function z of the measurement model. The reason for this is that we do not want to trouble the feature detection process described in the next section.

The estimation shown in the figure is more trustworthy, comparing with the odometry estimation. For unknown reasons, in some executions, the estimation goes badly, this is most lightly related to a not consistent definition of the error matrices $\mathbf{R}$ and $\mathbf{Q}$. Another problem that has been noticed is the high requirements for performance. Sometimes the simple fact of adding the plot function produces interferences with the calculations of the algorithm, provoking errors and delays in the plotting of the different estimations.

We have shown in this section approaches to be able to accede a more certain robot position. Nevertheless, in the real world, outside a simulator, we would most likely get worse estimations than shown in the last sections.

5.4. Mapping

We continue with the SLAM approach. Now having obtained the robot position, we start building the map. As we explained in section 2.3.1, the method followed to create the map is offline. Thus, during the navigation we collect the data that at the end of it will serve as as source of information to create the map. In this paper, we focus on the detection of circular segments. It means we assume a simple environment, such as the one described in section 5.1. The interference of other geometrical objects is not considered here and is left as a proposition for future work, meaning that besides the walls (lineal segments) we only have circular objects, such as columns, trees, and plants. In this section, we imitate the machine learning process in order to extract the circular segments from the laser rangefinder data. For doing this, we follow the steps of the context recognition chain inspired by the one proposed in [Holzl (2017)]:

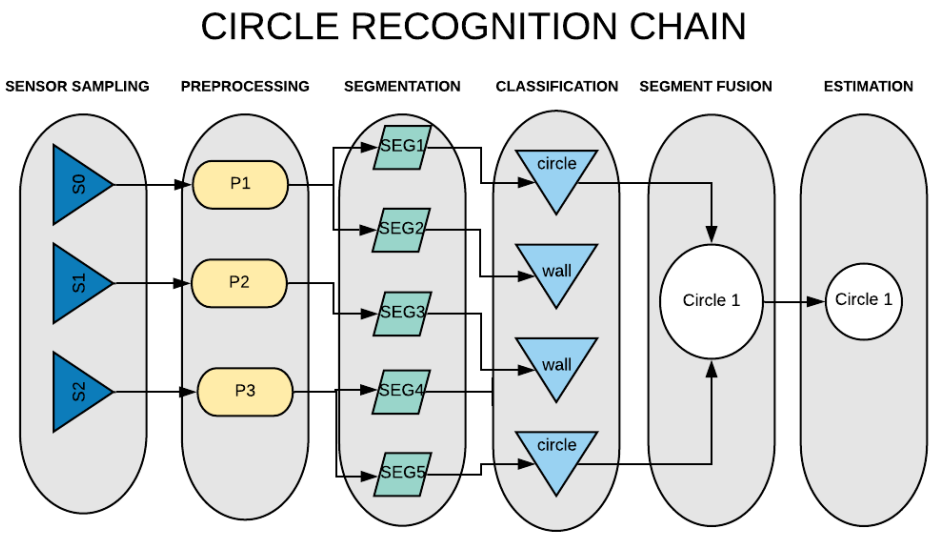

Figure 5.9: The different steps for obtaining the circular objects from the surroundings. First during the navigation the information from the sensors is stored. Second, this data is segmented and preprocessed in order to normalize it. Third, we classify the segments into linear or circular segments. Last, we join the segments that belong to the same circular object and apply the estimation.

5.4.1. Sensor sampling

The situation basis for both experiments is that the robot moves around, and with a certain periodicity, the laser reads data and saves it into an array. In this case, we assume that the time interval in which each sampling is taken is one second. It is for the reason that, if we take the same interval as in the localization process $\Delta t = 0.1$, the performance and the amount of memory needed is too large. Thus, we define an array of samplings $\mathbf{S}$ that contains every lecture the robot has made during its wandering.

$$\mathbf{S} = [s_1, s_2 ... s_n] \tag{5.10}$$

Every sampling is obtained as form of the function $r(\phi)$, where r is the distance to the object and ϕ is the angle between each laser ray and the robot from 0 to π (180 degrees). Note that when the laser detects no object at a particular deviation, it assigns a zero to that specific angle.

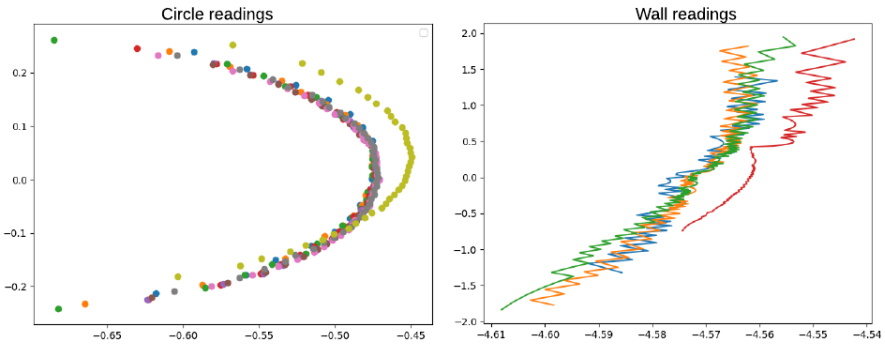

Figure 5.10: Multiple readings of the same column (left), multiple readings of the same wall (right).

On figure 5.10, the different samplings are shown in order to help to visualize the concepts above. On the left-hand side, different readings of the same column are presented; each reading in time is depicted with a different colour. On the right-hand side, the same holds for the walls. Each of the colours arrays of points is saved as a sampling in equation 5.10.

5.4.2. Prepocessing

In this section, we transform the array of sampling from the previous section, so it makes sense from an absolute perspective. Each sampling is saved as the distance from the robot at the moment the laser performs the reading, meaning that the samplings are relative to the robot position. In order to proceed to the next step, we need to calculate the absolute position of each of the positions in the sampling. For doing it, we need to use a simple vector adding.

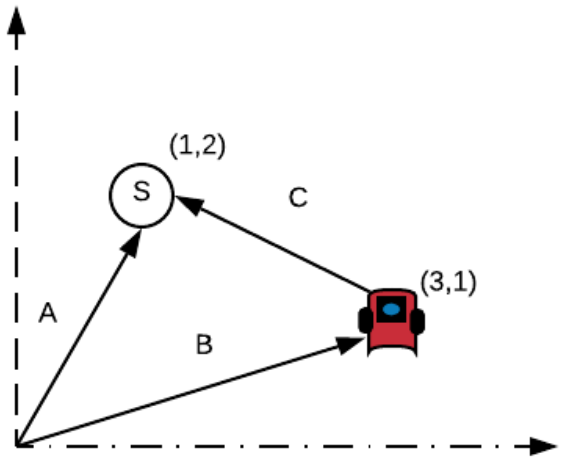

Figure 5.11: Example of calculation of the absolute position of the sampling s. The robot is in position $(x = 3, y = 1)$, where as the position of the sampling is $(x = 1, y = 2)$.

To help the explanations, figure 5.11 shows a simple example of a disposition when a laser reading is captured. The position of the robot is denominated by vector $\vec{b} = (x_r = 3, y_r = 1)$, whereas the relative position of the object captured is $\vec{c} = (x_r = -2, y_r = 1)$. To calculate the absolute position of the sample we add the $\vec{b}$ and $\vec{c}$.

$$\vec{a} = \vec{b} + \vec{c} = (3, 1) + (-2, 1) = (x_s = 1, y_s = 2)$$

As we can see, we obtain the position of the sample from the origin of coordinates. This process is repeated for every point in the sample s_t, so we get an array $\mathbf{P}$ of segments observed from the origin of coordinates. The reason we want a common point of reference is to determinate which segments belong together to describe a unique object.

5.4.3. Segmentation

Before starting the segment classification, we need to realize that in each lecture, a not unique object is captured. On the contrary, the typical situation is that more than one object is detected. In parallel with the absolute position calculation, we need to find the breakpoints that define each segment per lecture. To accomplish this, we define a simple situation.

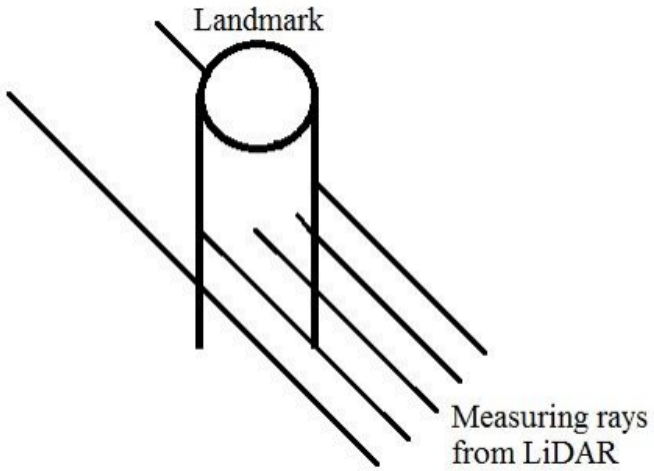

Figure 5.12: Laser pointing to specific landmark during the navigation. The rays that reflect on the landmark provoke a change in the data obtained by the laser. Thus, a big change is notices in the function $r(\phi)$. [Qader (2018)]

Figure 5.12 shows the laser rays pointing to a specific object or landmark. As we can see, the leftmost ray extends itself longer than the ones that reflect on the landmark. This approach aims to calculate which rays belong to each of the objects captured in one reading. One possibility would be using an extension of the scale-space algorithm described in section 4.3.1. However, we do not posses the time required to adapt the algorithm to this specific usage.

Having the function r_ϕ from the laser readings, we can calculate the first derivative $r'(\phi)$. By definition, the derivative is a measure of the rate at which the value of the function changes with respect to the change of the function variable; in our case, the change in the value $r(\phi)$ with respect to ϕ. If we look again at figure 5.12, we should expect that the leftmost ray and the rightmost ray produce a big change in the value of r, which provokes a pick in the first derivative that can be understood as a breakpoint. Therefore, the segments are defined by the values in between two breakpoints.

Let us illustrate this with a real example, extracted from the indoors experiment, precisely, the situation at time 0, shown in figure 5.13. Here, the robot receives readings from two lateral walls, two columns, and a part of the wall in front of it.

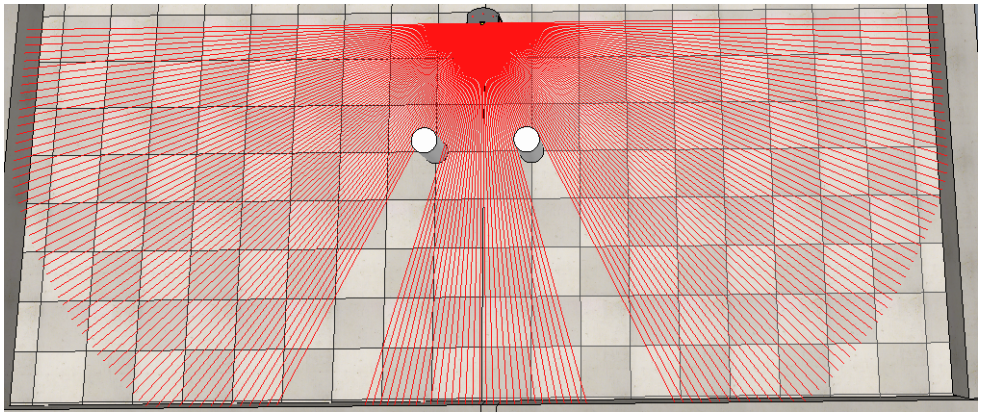

Figure 5.13: The robot pointing to two columns and behind walls.

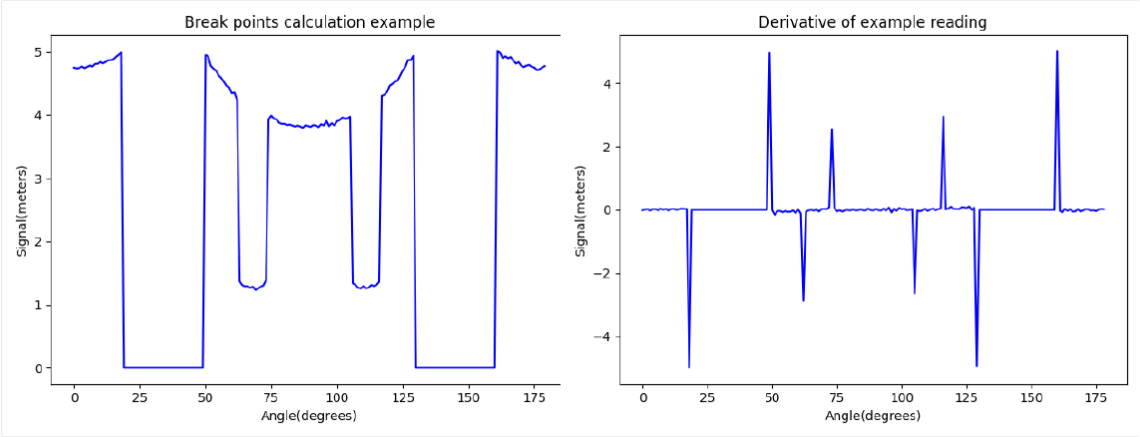

Figure 5.14: Readings in illustrative situation (left) and first derivative of function $r(\phi)$ for these readings (right).

The figure 5.14 shows on the left-hand side the readings obtained by the laser in the situation described above, and on the right-hand side, the calculation of the first derivative of the function. As we observe, there exist eight maximums and minimums that divide the readings into nine segments. The segments, which value is zero, in this case, the second and the eighth, correspond to a non-detection zone. The rest of the segments represent with regard to the robot: the right wall, the right column, the wall in-between columns, the left column and the left lateral wall.

A noticeable difficulty of this method is the calculations of maximums and minimums of the first derivative. To accomplish this, we need to compare each value of the function $r'(\phi)$:

$$|r'(\phi)_i - r'(\phi)_{i-1}| > \Delta c$$

where $r'(\phi)_i$ and $r'(\phi)_{i-1}$ are consecutive values in the derivative, and Δc is a threshold value. We need the threshold because the change in the function r is also subjected to noise, meaning that a big interference in the signal received by the laser could cause a pick in the first derivative and be understood as a breakpoint.

Figure 5.15 shows the first derivative with circular blue markers on the detected breakpoints.

Another corner case detected during the experiments is when the relative distance between two different objects and the robot is relatively similar. In this case, only a small change is noticed; thus, the breakpoints are not captured by the procedure of finding minima and maxima. We talk more about it in the conclusive chapter.

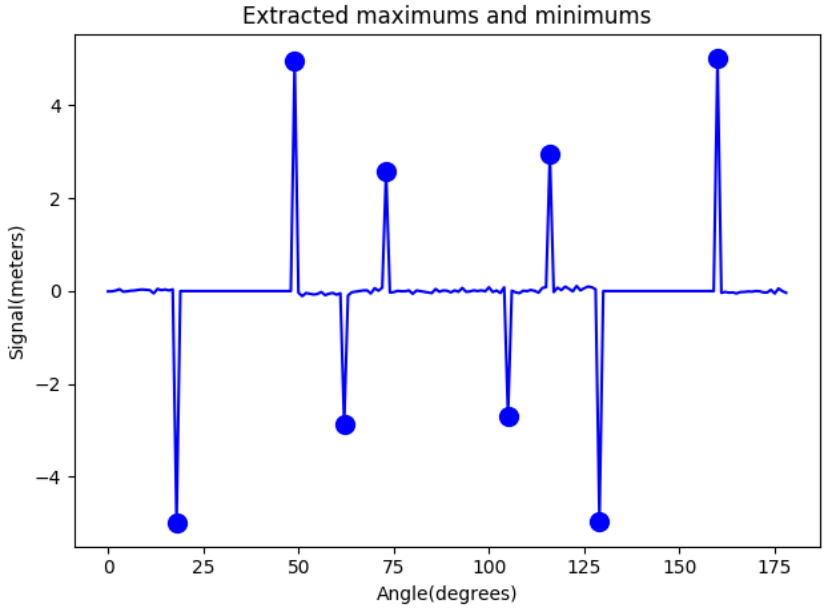

Figure 5.15: Detected breakpoints by the maxima an minima procedure. Each minimum and maximum describes the beginning and end of a specific segment.

5.4.4. Feature classification

Once we have obtained the breakpoints for every segment in the array of samplings 5.10, we partition the readings into segments and perform the calculations to obtain the absolute position. From this, we obtain an array of features $\mathbf{F}$, where each f_i represents a segment:

$$\mathbf{F} = [f_1, f_2 ... f_n] \tag{5.11}$$

It is crucial to notice, that the features with a small length and the ones compose just by zero are discarded. In the first case, because really small features probably come from an erroneous estimation of the breakpoints and most likely do not grant relevant information. In the second case, the features compose just by 0 represents the part of the readings when no object was detected. This can be understood as part of the preprocessing.

Once we have the segments, we proceed to one of the most complex tasks of this process, namely, the classification of segments as linear or circular. It is tough to accomplish as in many situations the read data from a circle can be easily understood as a linear segment and the other way around. As an illustration, think that if we look from a short distance at a large column, it will seem that we are facing a wall, since the part of the column that we are able to see does not denote a significant circular deviation. In a big scale, the same happens with the Earth, which looks flat from a human's perspective.

Consider an example illustrated in figure 5.16. We observe a situation when the robot is pointing closely at a wide column. The segment that represents the column in the figure is the one in-between the two blue dots. It is essential to notice how similar the segments on the top right and left representing walls are with the center bottom one representing the column.

The next step is to define a procedure where we decide whether the data refers to a linear or a circular segment. A possible way of doing it would be to recover the explanation from section 2.4.1, where we defined a residual that can give as an indicator of how the estimation of linear fitting went. By applying this procedure to the data shown in figure 5.16, we obtain a residual value of $R_{linear} = 0.05$. On the other hand, we can use the same data to try to estimate a circle by the method explained in section 4.4.2. Then, the residue is defined as the sum of the distances of each point to the estimated centre of the circumference. In this case, the result of the circle estimation residue is $R_{circular} = 0.1$.

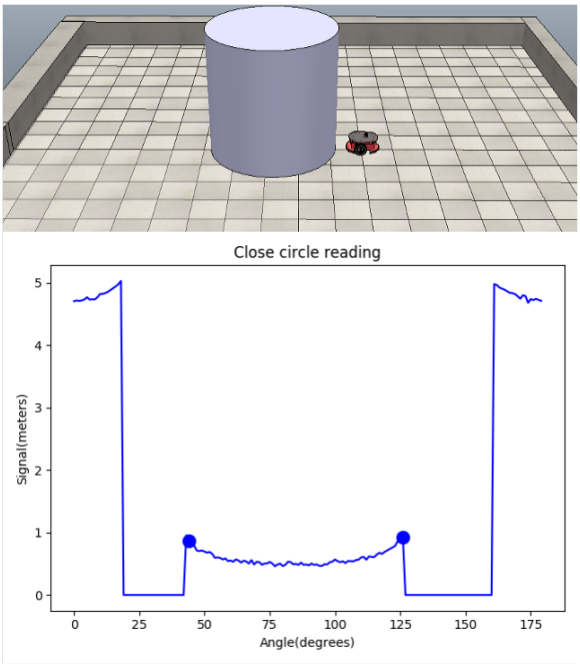

Figure 5.16: The robot closely facing a big column(top) and the readings obtained.

We could discuss whether comparing these results makes sense. However, this is not possible since the values of each estimation do not have an actual relationship among them. The fact that, in this case, the residue for the circular estimation is greater than for the wall fitting is not relevant, because it does not provide useful information about a generalization of the experiment.

The other approach proposed in [Muñoz Bañón (2016)], is to observe the deviation of the value of the MSE error described in the Kalman filter section. By comparing this deviation on circular and linear segments, we can discern between both. However, after experimenting with this approach, it was not possible to find the proper aspects to discern between the two. Furthermore, this method can not be applied if we have more than two types of segments.

It is because of this and the time window that we decide to define a much simpler approach. We observe the values of the least squared circle fitting in the experiments. From these values, we define a threshold. All the segments, which circular estimation is above the defined threshold are considered linear segments and the ones under the threshold are considered circular segments. Moreover, we define the maximum radius the column in our experiments can have. The linear segments can be predicted as large-sized circumferences. By limiting the size, we discard many of erroneous estimations.

5.4.4.1. Comments on implementation

The code implemented for feature classification deserves a comment. Even though we have not been able to provide a stable method for discerning between the types of segments, a design based on object-oriented programming is proposed.

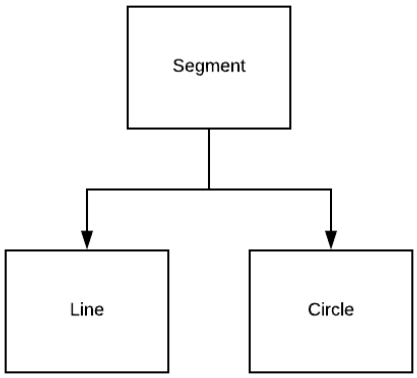

Figure 5.17: Class diagram abstracting the type of segments.

The figure 5.17 shows a diagram of classes. The super-class segment contains common attributes and operations. For example, the common attributes are the values representing the points of the segments. The operations can be general ones, such as converting the coordinates. Each of the sub-classes has its methods. In the case of the Line, it is the linear regression, whereas the class Circle is estimated by the Kalman Filter.

We point out some advantages of using an objected-oriented description of the segments. The first one is that it can be easily extended, meaning that we can add other types of geometrical figures. Each one of the sub-classes can provide its estimation methods. Another advantage is the use of comparators in the classes; for example, we can define whether two segments are equal or not. From the maintainability and extensibility point of view, defining the different segments as classes is preferable and can help to improve the classification in future projects. At some point.

One other possible solution would be to use the lineal segments estimation method developed by [Cuadra Troncoso (2011)]. Once we put the two methods together it might be that we could find a significant feature for discerning between the two types of segments.

5.4.5. Segment fusion

In the segment classification phase, we obtained two different arrays of features: one containing what we have classified as lines and the other what are thought to be circular segments. In this section, we explain how to decide which of these segments belong together (meaning that the segments were captured at different times but belong to the same object).

In order to be able to match different circular segments into one unique circular object, we have to define what we understand as different columns. We assume that every circular object in the experiment is separated from the other by a minimum distance. Thus, the circles do not intersect. For every circular segment obtained in the classification phase, we apply the least square estimation explained in section 4.4.2. Even though the initial estimation is not so precise, it is enough to decide whether two or more of the obtained segments represent one circle.

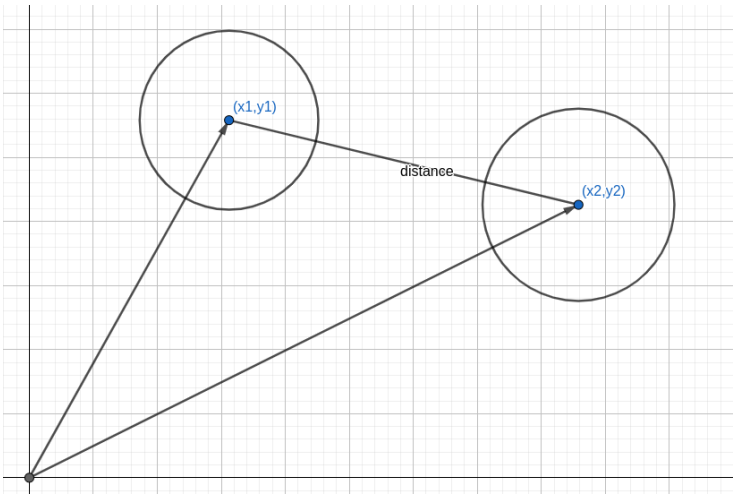

Figure 5.18: Comparison of two circles. The distance between centers determines whether they intersect or not.

For two circles C_1 and C_2, extracted from different segments, we say that they represent the same circle if:

$$R_{C1} + R_{C2} \leq \sqrt{(x_{C1} - x_{C2}^2) - (y_{C1} - y_{C2})^2}$$

If this condition is satisfied, we say that the segments from which the circles where estimated belong to the same object. In this case, we fusion the segments by creating a new instance of the class Circle and add the values of both. Finally have an array of circular segments

representing the detected circles in the experiment. Remember that for all this method, there is an error involved, meaning that the process is not deterministic. Thus, different experiments have different results. A segment that belongs to the same circle may be considered as different, hence producing a result with a greater number of circles than actually existing.

5.4.6. Final estimation

We have obtained the array of segments composed by one segment per circular object in the scene. We are ready to apply the estimation measures that we see necessary to estimate the circles. In this case we apply the Kalman Filter explained in section 4.6 for each circular segment obtained from the previous phase. However, one consideration must be made in this section we applied the Kalman Filter to the convex part of the circle. If we remember the definition of the circumference that we provided in the equation 3.2, we see that for defining the convex part we took the minus sign of the equation. The values obtain from the plus sign will give us the concave part of the circle. Hence, we need to update the design of the Kalman Filter. Let us see it again to have a better view. In figure 5.19 the red segment is drawn by the plus sign version of the formula.

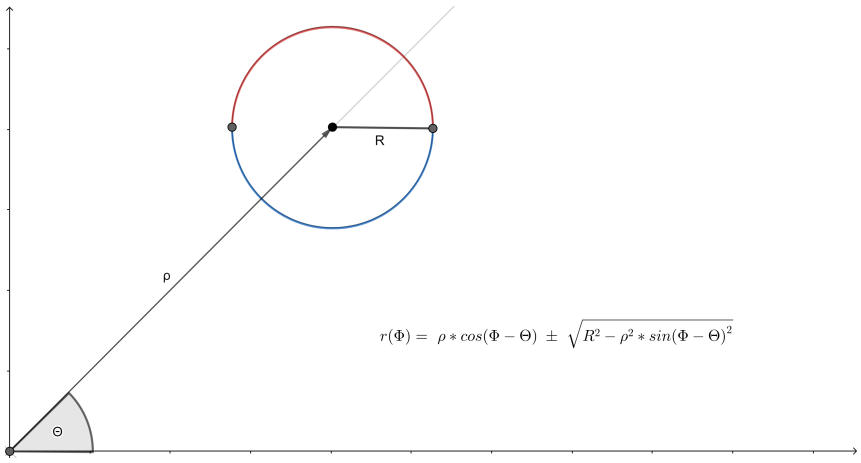

Figure 5.19: Circle in Polar coordinates.

We update the design of the Kalman Filter by updating its measurement function h. This function now will consider that if the measurement received is higher than the estimated center of the circle then this points belongs to the concave part. If the data received is lower then we understand that is a point that belongs to the convex part. Because of stability reasons all the points that are close to the distance ρ are discarded. This is because the noise can alter this points that are near the two tangents of the circle with the origin.

Furthermore, it has been observed that the initial estimation of the circle improves when adding also data from the concave part. However, this is not the same for the Kalman Filter in every case. Notice that some times adding more data means raising the probability of having outliers.

Now that we have updated the Kalman Filter we apply the estimation process and obtain the circular objects around the scenes.

5.4.7. Results

5.4.7.1. Results indoors scene

For the indoors scene we have a run time of 60 seconds. This time is more than enough to capture different features that represent the circles. the localization algorithm used is the one based in the GPS, section 5.3.2.

In the indoors scenes we have distributed the columns and the plant so they cover some of the corner cases explained in previous sections. From the one side the plant is really close to the wall which difficulties the estimation of breakpoints. From the other side we have two small columns close to each other. Figure 5.20 identifies every circular object for the indoors experiment. The results in one of the best cases can be visualized in right side of the figure. The red circles represent the real columns and the blue circles represent the estimated ones. There are some gaps in the walls, mainly caused because the laser did not detect them.

Another factor can be that as a criteria the linear segments that are not over a certain threshold by the residue calculated in the linear regression process are not plotted. Table 5.4.7.1 shows the results of the error for each one of the circular segments.

	$\Delta\rho$	$\Delta\theta$	ΔR
Column 1	0.022	0.004	0.019
Column 2	0.010	0.008	0.02
Column 3	0.05	0.018	0.027
Column 4	0.074	0.0085	0.005
Column 5	0.008	0.009	0.003

Table 5.1: Results for each circular object of the indoors scene. The errors are given with respect to the polar parameters of the circle in equation 3.2.

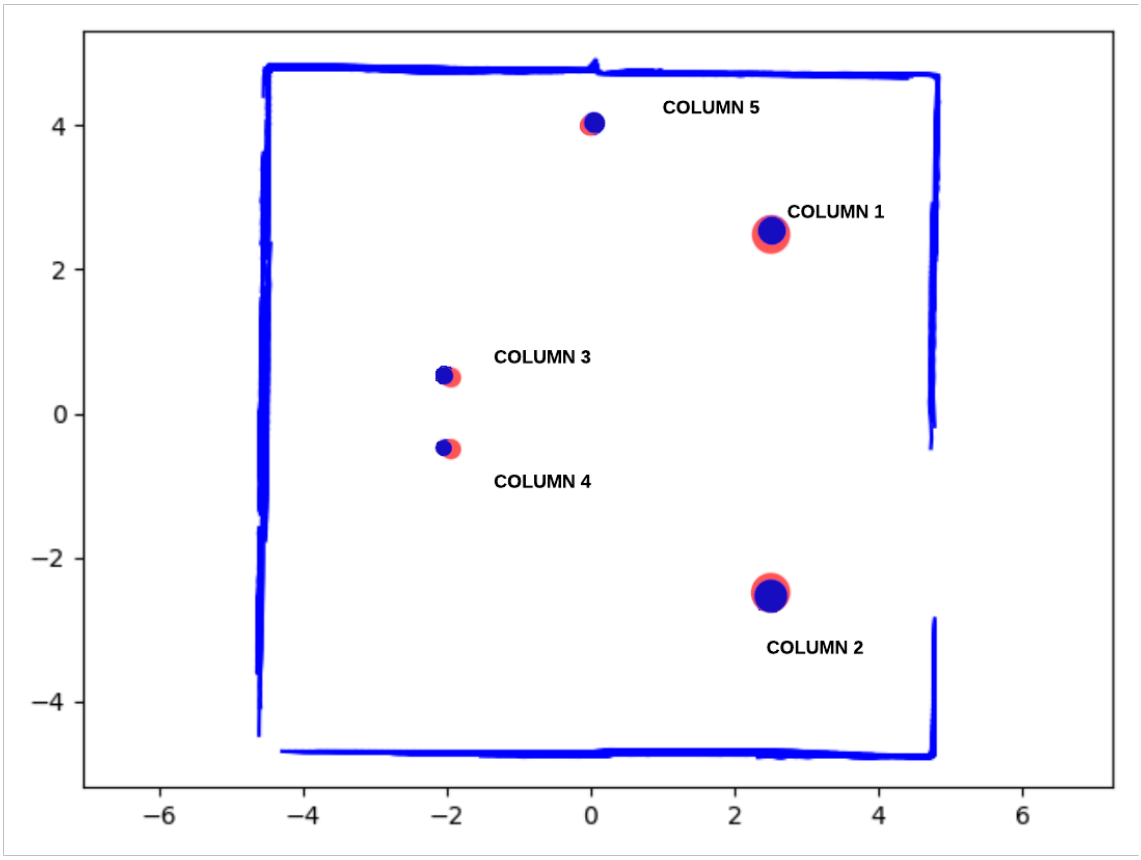

Figure 5.20: Identification of the circles for the indoor experiment and good case results. The blue circles represent the real position of the columns and the red ones the estimation after applying the mapping chain.

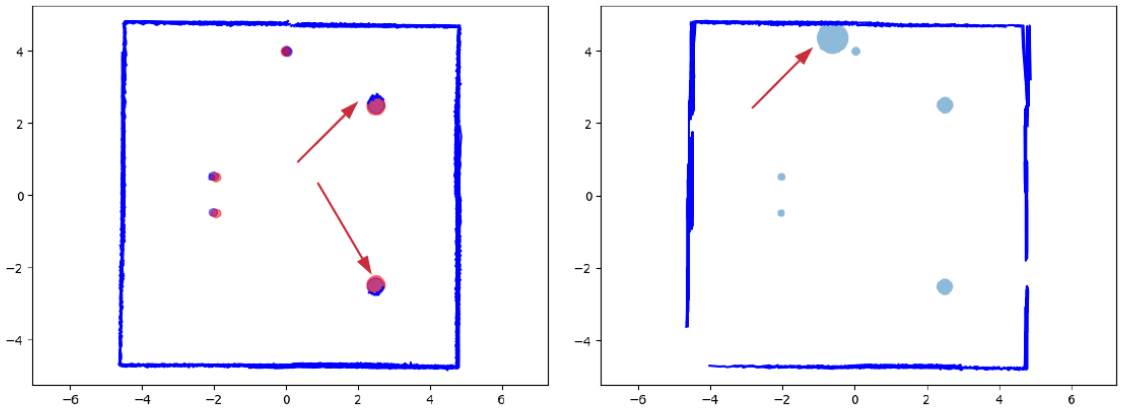

Figure 5.21: Typical errors in experiment associated to a bad classification. On the left the circular segments have been take as linear ones. On the right, the wall section together with the plant is understood as a big circular object.

In other cases, the most common errors in estimation are related to the classification process. Either one circular segment is take as an linear one or the other way around. Two examples can be seen in figure 5.21. In the left hand side image we see two segments in the big columns that have been understood as lines. With respect to the image in the right hand side, what happened, most likely, is that the breakpoints calculation failed. From this failure we get a segment that includes part of the circular object and the wall. The result being an estimate column that does not exist.

5.4.7.2. Results outdoors scene

The outdoor scene is a bit different. Where as in the indoor experiment the columns and the plant described a perfect circular object, in the outdoor experiment the trees do not possess a circular trunk. Hence, the classification is harder because the shape of the trees draws a linear segment in one of its sides. The simulation time for this experiment is larger, 180 seconds, because there is more terrain to cover.

Figure 5.22 shows the estimation on one good case. The problems explained in previous section, derived from the classification and the calculations of breakpoints are the same here. In some occasions, we get lineal segments where the trees are supposed to be. Furthermore, many times, we do not obtain the trees as circular segments. This is due to the linear side of the tree that worsens the circular estimation.

	$\Delta\rho$	$\Delta\theta$	ΔR
Tree 1	0.04	0.0012	0.51
Tree 2	0.054	0.01	0.19
Tree 3	0.023	0.011	0.066
Tree 4	0.032	0.007	0.05
Tree 5	0.008	0.0036	0.011

Table 5.2: Result errors for outdoors estimation.

As we can see in table 5.2 the estimation of the radius of the big trees is not so precise(radius of tree 1 and tree 2). The reason for this is the linear section that the draws the bottom of the tree. Also could be derived of the problem of big circles. The estimation has been made by extracting the localization from the simulator. The experiments with the different localization explained in section 5.3 have been tried, however the error in the orientation of the robot is too high to provide a good estimation of the circular objects.

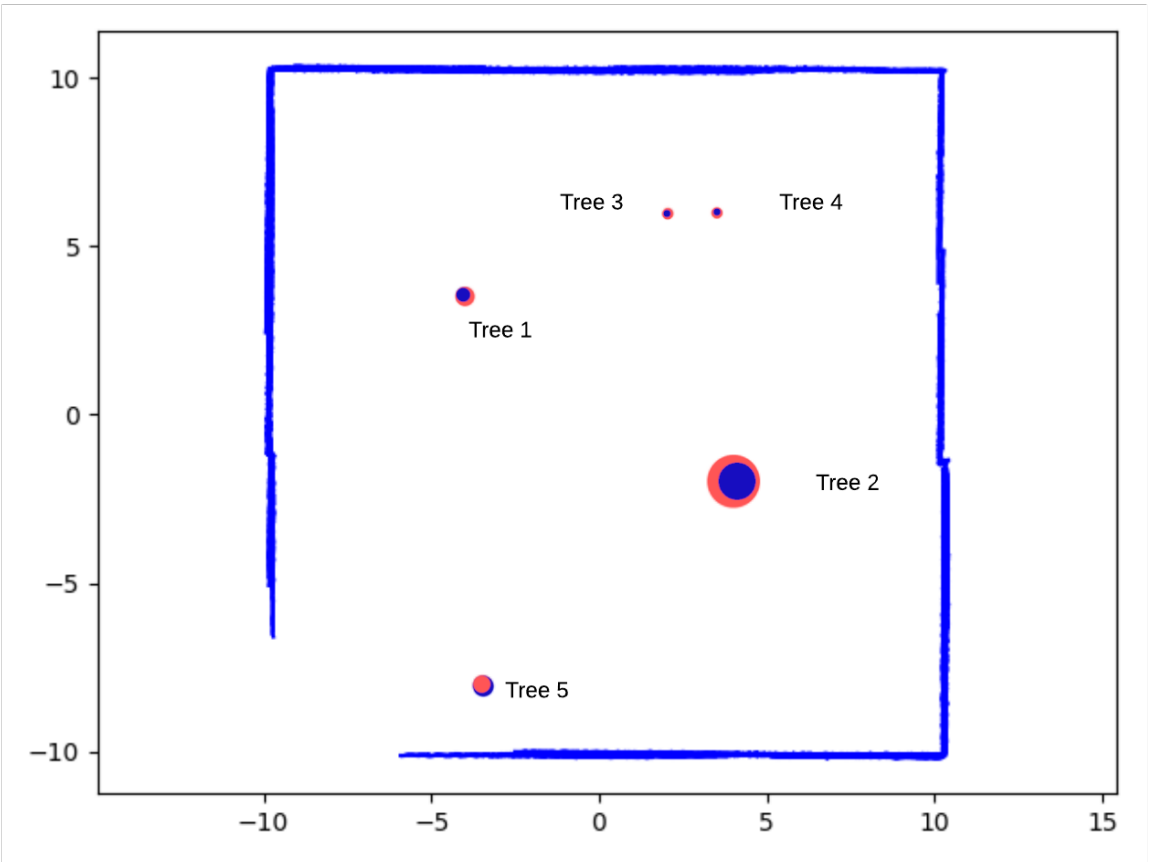

Figure 5.22: Identification of the trees outdoors and the results in a good case. The red circles represent the real position of the columns and the blue ones the estimation after applying the mapping chain.

Chapter 6

Conclusion and outlook for future work

6.1. Conclusion

To enable a mobile robot to navigate in an unknown environment, and at the same time, calculating its location and building a map of the surroundings is not an easy task. Every step of the process opens new questions, which require answers. Furthermore, composing the estimation of the surroundings with circular objects does not make the task easier.

There are two main ingredients needed for a partly successful accomplishment of this task. The first one is to have a competitive mathematical knowledge of the problem. We have seen that what is a basic geometrical shape that everyone has met since primary school, can turn to be difficult from a statistical point of view. The second essential factor is the understanding of the different aspects related to the design of a context recognition process.

In this project, we have overcome these difficulties and provided an extension to a relatively new concept, that is the estimation of circular segments with the Kalman Filter. By researching, we have found a better initialization method. The least-squares circle fitting explained in section 4.4.2, proves to be more numerically stable. Thus, even when there is the presence of outliers and noise-related factors, the maximum errors satisfy the expectations.

With this respect, we have applied the knowledge and conclusions from work by [Muñoz Bañón (2016)] to design what seems to be a more stable solution of the Kalman Extended Filter. The research done for applying localization method using this kind of filter has served as inspiration for the circle fitting approach.

What at the beginning of research seems to be a *easy* problem, turns into a complicated question with many factors to analyze. The presence of many parameters of the model and constraints to avoid the sensor error influence difficulties and enlarges the experimentation time. When dealing with Artificial Intelligence related projects, the existence of magic numbers is ubiquitous. Magic numbers are a nickname given by the machine learning communities to those static numbers assigned as a threshold. The value of these numbers improves the results, although the real reason behind it, requires a parallel study.

Continuing with this idea, we can think of this project as a tree, branched by every new question. It is the explanations we give to this question, and there answers what suppose a big challenge.

Despite that, the results of the Extended Kalman Filter for circle fitting have not been as exact as expected from the beginning. With these results, we could and have done a step forward into a SLAM problem. As described in the literature, the most basic localization and mapping problem can turn out to be very complicated. For designing a new model, we have to make some underlying assumptions and try to reduce the complexity of what a real experiment would suppose. In this sense, the time window and the budget of the system play a significant role. The models used in the simulator, both robot and laser rangefinder, are highly costly. Nevertheless, the simulator approximates the first corner cases that we could find in reality.

On the one hand, we have used the Kalman Filter to design two localization approaches. The first approximation based in a GPS(section 5.3.2), proves itself to work relatively good. However, the assumption of having access to a GPS signal cannot apply for every situation. Therefore, another approach is proposed by combining the odometric data with the detection of landmark, in section 5.3.3. This two implementations, together with the circle fitting approach, prove that the Kalman Filter is a powerful technique. However, the real complexity lies in the tweaking of the parameters, for instance, the error matrices associated with the process.

Finally, we have provided an innovative context recognition chain in section 5.4. This mapping chain, although only based in circular segments, can be extended for other geometrical figures. Both the design and the implementation lead to good maintainability.

The mapping process is an excellent example of the study of Machine Learning Problems. Each step of the process opens new questions, namely the segment classification. In this project, this part has not matched the expectations. Nevertheless, the implementation

proposed in section 5.4.4.1 has paved the path for future research. Consequently, the results obtained in the estimation of the circular object represent a necessary abstraction of a real problem from the one side, the results when estimating columns match the expectations, where we have errors of just some centimetres. From the other side, when using the tree models that are not precisely circular, the results are not so exact.

To sum up, the context recognition problem is an exciting field. The more one learns about the problem. The more one realizes how difficult it is. Thus, during the process of research, the expectations are lowered, sometimes causing a sensation of disappointment. However, by overcoming this, there are moments when the knowledge acquired through investigation becomes practical and helps solve some of the problems. It is at this moment, when one realizes how fascinating the Artificial Intelligence can become.

6.2. Future work

There are many aspects of this project that are worth researching. In this last section, we briefly comment on four of them. The ones that deserve more attention.

- The main one is to keep researching on the design of a better Kalman Filter. It has been tried during this project to use the Unscented Kalman Filter. However, the mathematical competence it requires is too high.
- The second line of study proposed derives from this project, in my opinion, should be to study the classification of different segments. Statistical algorithms used in Machine Learning could provide an interesting solution.
- The third one is related to the SLAM problem. Even though in this project, we have designed the two parts that compose this problem separately, namely localization and mapping, once we put them together, the errors do not match the expectations.
- Finally, in this same line of study, to improve the Kalman Filter based on the odometry information, section 5.3.3. Despite accomplishing good results some times, many times due to unknown reasons, the estimation goes badly.

There are many more topics that can be researched. After all, as we explained in the introductory chapter, the focus of the study in Artificial Intelligence should be to provide more realistic solutions one bit at a time.

Bibliography

Students t distribution. URL https://stattrek.com/probability-distributions/t-distribution.aspx.

Primitive shapes, 2009. URL http://www.coppeliarobotics.com/helpFiles/en/primitiveShapes.htm.

Shakey: the world's first mobile, intelligent robot, May 2015. URL http://robotglobe.org/shakey-the-worlds-first-mobile-intelligent-robot/.

Stanford research institute problem solver, Mar 2019. URL https://en.wikipedia.org/wiki/Stanford_Research_Institute_Problem_Solver.

Patricia Anderson. The gauss-markov theorem: Study guide., May 2018. URL http://www.dartmouth.edu/~econ20pa/StudyGuide1.doc.

architectpianist. Finding the angle between two points. Mathematics Stack Exchange. URL https://math.stackexchange.com/q/1201367. URL:https://math.stackexchange.com/q/1201367 (version: 2015-03-22).

Ronald C. Arkin. *Behavior-Based Robotics*. Jan 1998. URL https://www.biblio.com/9780262011655.

balzer82. balzer82/kalman, 2018. URL https://github.com/balzer82/Kalman/blob/master/Extended-Kalman-Filter-CHCV.ipynb.

Dominic B.O Boesl. 4 robotic revolutions - proposing a holistic phase model describing future disruptions in the evolution of robotics and automation and the rise of a new generation 'r' of robotic natives, Oct 2016. URL http://www.roboticgovernance.com/wp-content/uploads/2017/01/1285.pdf.

Randy Bullock. Least-squares circle fit, Feb 2017. URL https://dtcenter.org/met/users/docs/write_ups/circle_fit.pdf.

Howie Choset. Localization, mapping, slam and the kalman filter according to george, 2016. URL https://www.cs.cmu.edu/~motionplanning/lecture/Chap8-Kalman-Mapping_howie.pdf.

CoopeliaRobotics. Vrep. URL http://www.coppeliarobotics.com/.

CS-Oswego. Robotic paradigms, 2019. URL http://www.cs.oswego.edu/~odendahl.

Jose Manuel Cuadra Troncoso. Modelado adaptativo del medio para la navegación de robots autónomos utilizando algoritmos basados en el centro de áreas. 2011.

José Manuel Cuadra Troncoso, José Ramón Alvárez, Israel Navarro, Félix de la Paz, and Raúl Arnau. Consistent robot localization using polar scan matching based on kalman segmentation, Jan 2015.

David. Plotting polar equations of circles not centered at (0, 0). Mathematics Stack Exchange, 2014. URL https://math.stackexchange.com/q/737380. URL:https://math.stackexchange.com/q/737380 (version: 2014-04-02).

Martin Dekan and Frantisiek Duchon. Probabilistic model laser rangefinder. URL http://www.magnanimitas.cz/ADALTA/0102/papers/J_dekan.pdf.

C.L DODGSON. *12*. OXFORD UNIVERSITY, 1896.

Etimology. Robot etimology. 2012. URL https://www.etymonline.com/word/robot. Accesed May 2019.

Jan Faigl. Robotic paradigms and control architectures, 2015. URL https://cw.fel.cvut.cz/old/_media/courses/b4m36uir/lectures/b4m36uir-lec02-slides.pdf.

Ivan Gochev and Gorjan Nadzinski. Path planning and collision avoidance regime for a multi-agent system in industrial robotics, 2014. URL https://pdfs.semanticscholar.org/69fa/f19aaf406796377141ed6ba76a4dd6641f23.pdf.

Hokuyo. Scanning laser range finder utm-30lx/ln, 2007. URL http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/UTM-30LX_spec.pdf.

Dr. Gerold Holzl. Context aware system design principles, 2017.

IFR. Industrial robot sales increase worldwide by 31 percent. 2017. URL https://ifr.org/ifr-press-releases. Accesed May 2019.

ja72. Get the equation of a circle when given 3 points. Mathematics Stack Exchange. URL https://math.stackexchange.com/q/213670. URL:https://math.stackexchange.com/q/213670 (version: 2012-10-14).

R.E Kalman. A new approach to linear filtering and prediction problems. *A New Approach to Linear Filtering and Prediction Problems1*. doi: 10.1109/9780470544334.ch9.

Oleg Katkov. Reduce gps data error on android with kalman filter and accelerometer, Jun 2018. URL https://blog.maddevs.io/reduce-gps-data-error-on-android-with-kalman-filter-and-accelerometer.

Pramukta Kumar. Recipe: Laplacian scale space representations with opencv, Feb 2018. URL https://medium.com/@pramukta/recipe-laplacian-scale-space-representations-with-opencv-250b88cc8c80.

Tony Lacy and N.A. Thacke. *Tutorial: the Kalman Filter*. MIT, 1998.

Luisa Maderia Cardoso. Mo810 - trabalho 2, 2017. URL https://github.com/luwood/MO810-vrep-python.

Adasrsh Menon. Linear regression using least squares, Sep 2018. URL https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570.

Adebts MobileRobots. P i o n e e r 3 - d x. URL https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf.

Darryl Morrell. Extended kalman filter lecture notes, 1997. URL https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/kalman/ekf_lecture_notes.pdf.

Robin Murphy. *Introduction to AI robotics*. The MIT Press, 2018.

Miguel Angel Muñoz Bañón. Segmentación de contornos circulares a partir de mediciones tomadas con láser. 2016.

Mark S. Nixon. Circle extraction via least squares and the kalman filter. In Dmitry Chetverikov and Walter G. Kropatsch, editors, *Computer Analysis of Images and Patterns*, pages 199–207, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-540-47980-2.

rk S Nixon. Circle extraction via least squares and the kalman filter, 1997. URL https://eprints.soton.ac.uk/250337/1/kalman.pdf.

Ben Ogorek. Yet another kalman filter explanation article, May 2019. URL https://towardsdatascience.com/yet-another-kalman-filter-explanation-article-be0264d99937.

Oxford Dictionary. Robot definition. 2019. URL https://www.oxfordlearnersdictionaries.com/definition/english/robot. Accesed May 2019.

Paulo Pinheiro and Eleri Cardozo. Kalman filter for mobile robot localization, May 2014. URL https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxwYXVsb3BpbmV8Z3g6NWExMGI2MmZhZmViMzg3.

Pixabay. Free image on pixabay - robot, technology, robotic, machine, 2018. URL https://pixabay.com/vectors/robot-technology-robotic-machine-3422113/.

Hisham Abdel Qader. Extended kalman filter slam implementation for a differential robot with lidar, 2018. URL https://pdfs.semanticscholar.org/d07b/02282e2d1494a55165ed07e4c6db3dcf0f1f.pdf.

Rlabbe. rlabbe/kalman-and-bayesian-filters-in-python. URL https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/07-Kalman-Filter-Math.ipynb.

Atsushi Sakai. Extended kalman filter localization, Jan 2019. URL https://github.com/AtsushiSakai/PythonRobotics/blob/master/Localization/extended_kalman_filter/extended_kalman_filter_localization.ipynb.

Jinglin Shen, David Tick, and Nicholas Gans. Localization through fusion of discrete and continuous epipolar geometry with wheel and imu odometry. *Proceedings of the 2011 American Control Conference*, 2011. doi: $10.1109/\mathrm{acc}.2011.5990946$.

Rymsha Siddiqui. Path planning using potential field algorithm, Jul 2018. URL https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08.

Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT Press, 2011.

Vincent Spruyt. How to draw an error ellipse representing the covariance matrix?, Mar 2015. URL https://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/.

Cyill Stachniss. Robot mapping ekf slam, 2015. URL http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam04-ekf-slam.pdf.

Christopher Stover. Monotonic function, 1999. URL http://mathworld.wolfram.com/MonotonicFunction.html.

Gabriel A. Terejanu. Extended kalman filter tutorial, 2005. URL https://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf.

A. N. (Andrei Nikolaevich) Tikhonov, A. S Leonov, and A. G Yagola. *Nonlinear ill-posed problems*. Champan and Hall, 1998. ISBN 0412790203 (v. 2). Includes index.

Tractica. Consumer attitudes about household robots. 2016. URL https://www.tractica.com/robotics/consumer-attitudes-about-household-robots/. Accesed May 2019.

VexRobotics. What is the engineering design process?, 2019. URL https://curriculum.vexrobotics.com/curriculum/intro-to-engineering/what-is-the-engineering-design-process.html.

Koen L. Vincken, André S. E. Koster, and Max A. Viergever. Probabilistic multiscale image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:109–120, 1997.

Rumman Wakar. rlabbe/kalman-and-bayesian-filters-in-python, 2016. URL https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/11-Extended-Kalman-Filters.ipynb.

E. Wan. Sigma-point filters: An overview with applications to integrated navigation and vision assisted control. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 201–202, Sep. 2006. doi: 10.1109/NSSPW.2006.4378854.

Merriam Webster. Jacobian, 2016. URL https://www.merriam-webster.com/dictionary/Jacobian.

Wiki. White noise, Apr 2019. URL https://en.wikipedia.org/wiki/White_noise.

Andrew P. Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP*, 1984.

Glossary

behaviour mapping of sensory inputs to pattern of motor action. 12

Closed World assumption The world model contains everything the robot needs to know. 11

isotropic (of an object or substance) having a physical property which has the same value when measured in different directions. 39

Paradigm a philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated [CS-Oswego (2019)]. 9

parameterize Describe or represent a function in terms of a parameter or parameters. 42

regressor Regressor is the independent variable in a regression equation. 35