UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)

Scuola di Dottorato

Ingegneria dei Sistemi, Informatica, Matematica e Ricerca Operativa (ISIMR)

Indirizzo

Ingegneria dei Sistemi ed Informatica

CICLO

XXVI

MOBILE ROBOTS: LOCALIZATION AND MAPPING ALGORITHMS

Settore Scientifico Disciplinare ING-INF/04

Direttore: Ch.mo Prof. Sergio Greco

Firma Sergio Greco

Supervisori: Ch.mo Prof. Pietro Muraca

Firma

Ch.mo Prof. Emanuele Garone (Universitè Libre de Bruxelles)

Firma

Dottorando: Dott. Luigi D'Alfonso

Firma Luigi D'Alfonso

A Maria Emilia, Papà, Mamma e Dik,
voi tutti neanche immaginate quanto siate importanti per me.

Abstract

In recent years, mobile robots start to be very often used in various applications involving home automation, planetary exploration, regions surveillance, rescue missions, ruins exploration. In all these fields, to accomplish its tasks, a mobile robot needs to **navigate** into an environment facing the localization, mapping and path planning problems. In this thesis a set of new algorithms to solve the mobile robots localization and mapping problems are proposed.

The aim is to provide an **accurate mobile robot estimated pose** and to build a **reliable map** for the environment where the robot moves, ensuring **algorithms computational costs low enough** so that the algorithms can be used in *real time*, while the robot is moving. The localization problem is faced in static and dynamic contexts, assuming the environment surrounding the robot completely known, partially known or totally unknown. In the static context, localization algorithms based on the use of cameras and inertial measurement units are proposed. In the dynamic context, the mobile robots localization problem is solved by developing a set of new Kalman filter versions. About the mapping problem, two novel mapping models are defined. These models are used along with the proposed localization techniques to develop three algorithms able to solve the Simultaneous Localization and Mapping (SLAM) problem.

All the proposed solutions are tested through numerical simulations and experimental tests in a real environment and using a real mobile robot. The results show the effectiveness of the proposed algorithms, encouraging further researches.

Arcavacata di Rende (CS), November, 2013 *Luigi D'Alfonso*

Abstract (Italian translation)

Negli ultimi anni, i robot mobili vengono adoperati sempre più spesso in un sempre maggiore numero di applicazioni inerenti la domotica, l'esplorazione di pianeti, la sorveglianza di luoghi, le missioni di salvataggio, l'esplorazione di rovine. In tutti questi campi, per portare a termine i propri compiti, un robot mobile deve **navigare** all'interno di un ambiente, affrontando i problemi di localizzazione, ricostruzione dell'ambiente (mapping) e pianificazione di traiettorie. Nella presente tesi sono proposti nuovi algoritmi atti a risolvere i problemi di localizzazione di robot mobili e ricostruzione dell'ambiente in cui essi si muovono.

Obiettivo di tali algoritmi è fornire una **stima accurata** della posizione del robot mobile ed una **mappa affidabile** dell'ambiente in cui esso si muove; il tutto garantendo **costi computazionali degli algoritmi sufficientemente bassi** da permettere l'utlizzo degli stessi in *tempo reale*, mentre il robot è in movimento. La localizzazione del robot viene effettuata sia in situazioni statiche che in situazioni dinamiche, assumendo l'ambiente circostante noto, parzialmente noto oppure completamente ignoto. Nel contesto statico, viene proposto un algoritmo di localizzazione basato sull'uso di telecamere e sensori inerziali. Nel contesto dinamico, la localizzazione viene effettuata mediante nuove versioni del filtro di Kalman. Per quanto riguarda la ricostruzione dell'ambiente circostante il robot, vengono presentati due nuovi modelli di mapping. Tali modelli vengono quindi adoperati, assieme agli algoritmi di localizzazione proposti, al fine di sviluppare tre nuove tecniche di risoluzione del problema della localizzazione e ricostruzione simultanee (SLAM).

Tutte le soluzioni proposte sono testate attraverso simulazioni numeriche ed esperimenti reali in un ambiente sperimentale ed adoperando un vero e proprio robot mobile. I risultati ottenuti mostrano l'efficacia delle soluzioni proposte, incoraggiandone sviluppi futuri.

Arcavacata di Rende (CS), November, 2013 *Luigi D'Alfonso*

Contents

List of Figures

List of Tables

Introduction

This thesis focuses on the main theoretical and methodological issues related to the mobile robots navigation. In recent years, due to the continuously increasing use of mobile robots in various applications, all the problems related to this research field have received very considerable attention. Whatever is the mobile robots application goal, the robot has to face three main problems which can be summarized by the following three well known questions:

- Where the robot is?
- Where the robot should go?
- How the robot should arrive in its goal location?

This thesis focuses on the first two questions trying to give them a reliable answer. In particular the **mobile robots localization problem** and the **Simultaneous Localization and Mapping (SLAM) problem** will be discussed. A set of solutions to both the problems will be proposed in this dissertation. The aim of the thesis is to solve the localization and mapping problems using all the available information in the best possible way and requiring as few *a priori* assumptions as possible.

History and main topics

A mobile robot is an automatic machine that is capable of movement in a given environment. The history of mobile robots starts in 1939-1945, during the World War II, when the first mobile robots emerged as a result of technical advances on a number of relatively new research fields like computer science and cybernetics. The first mobile robots examples were mostly flying bombs like smart bombs (the V1 and V2 rockets) that only detonate within a certain range of the target using a guiding systems and a radar control.

A second mobile robot example appears in 1948 when W. Grey Walter builds Elmer and Elsie, two autonomous robots called *Machina Speculatrix*. These robots were able to explore their environment thanks to a very simple

but efficient strategy: they were both equipped with a light sensor and if they found a light source they would move towards it, avoiding or moving obstacles on their way. These robots demonstrated that complex behaviors could arise from a simple design. The main disadvantage about Elmer and Elsie is related to the light need; the two robots were completely unable to explore dark rooms. Starting from the idea of overcoming this issue, in 1961 the Johns Hopkins University develops Beast, that is a mobile robot which moves around using a sonar to detect its surrounding environment. Thanks to the use of sonar sensors, dark rooms could be explored with no problems. In 1969, Mowbot (see Figure I.1), the first robot that would automatically mow the lawn was developed; it was the first example of mobile robots application in home automation.

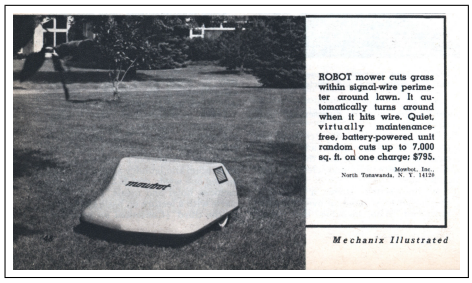


Fig. I.1: Mowbot, the first mobile robot in home automation

In 1970 the Stanford Cart line follower was developed, it was a mobile robot able to follow a white line, using a camera to see. It was radio linked to a large mainframe that made the calculations and it represents the first example of the vision based mobile robots. At about the same time, in 1966-1972, the Stanford Research Institute built Shakey the Robot (Figure I.2) a robot named after its jerky motion.

Shakey had a camera, a rangefinder, bump sensors and a radio link. Shakey was the first robot that could reason about its actions. This means that Shakey could be given very general commands, and that the robot would figure out the necessary steps to accomplish the given task. In the same year, the Soviet Union explores the surface of the Moon with Lunokhod 1, a lunar rover representing the first example of mobile robots space application. After six years, in 1976, the NASA sends two unmanned spacecrafts to Mars.

In the 1980s the interest in robots rises, resulting in robots that could be purchased for home use. These robots served entertainment or educational purposes. Examples include the RB5X, which still exists today and the HERO series. Until the '80s, the robot localization problem and the robot surrounding environment mapping problem have been always faced separately. During the 1986 IEEE Robotics and Automation Conference, held

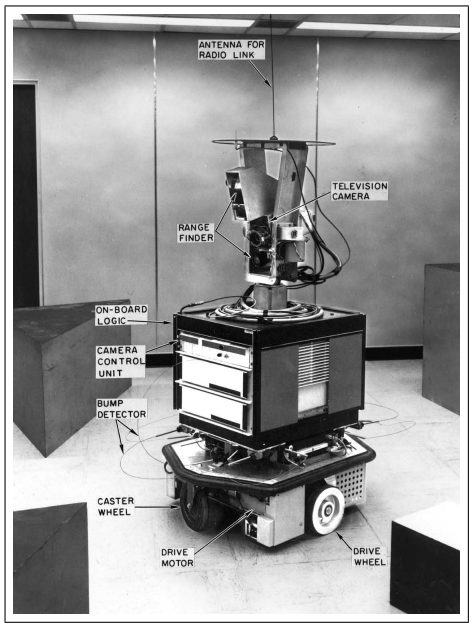


Fig. I.2: Shakey the robot

in San Francisco, a large number of researches as Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte had been looking at simultaneously applying estimation-theoretic methods to both mapping and localization problems. Over the course of the conference many paper table cloths and napkins were filled with long discussions about consistent mapping. The final result was a recognition that consistent probabilistic mapping was a fundamental problem in robotics with major conceptual and computational issues that needed to be addressed. Over the next few years a number of key papers were produced facing the Simultaneous Localization and Mapping problem (SLAM). The first paper about SLAM was written by Smith and Cheeseman [38], the authors describe the SLAM framework and lay down the foundations of SLAM research. After the '80s, robot navigation received considerable attention by the researchers due to the always increasing interest in mobile robots. In the 1990s Joseph Engelberger, father of the industrial robotic arm, works with colleagues to design the first commercially available autonomous mobile hospital robots, sold by Helpmate. In the same year, the US Department of Defense founds the MDARS-I project, based on the Cybermotion indoor security robot and André Guignard and Francesco Mondada developed Khepera, an autonomous small mobile robot intended for research activities. In 1993-1994 Dante I and Dante II were developed by Carnegie Mellon University. Both of them were walking robots used to explore live volcanoes. In 1995 the Pioneer programmable mobile robot becomes commercially available at an affordable

price, enabling a widespread increase in robotics research. In the same years, NASA sends the Mars Pathfinder with its rover Sojourner to Mars. The rover explores the surface, commanded from earth. Sojourner was equipped with a hazard avoidance system which enabled Sojourner to autonomously find its way through unknown martian terrain. In 2002 appears Roomba, a domestic autonomous mobile robot that cleans the floor and in 2003, Axxon Robotics purchases Intellibot, manufacturer of a line of commercial robots that scrub, vacuum, and sweep floors in hospitals, office buildings and other commercial buildings. Floor care robots from Intellibot Robotics LLC operate completely autonomously, mapping their environment and using an array of sensors for navigation and obstacle avoidance. In 2010, the Multi Autonomous Ground-robotic International Challenge has teams of autonomous vehicles which map a large dynamic urban environment, identify and track humans and avoid hostile objects.

Nowadays mobile robots are used in various applications and they are becoming a part of the human life. In particular, robot floor cleaners are available at low prices and many families use them everyday. Robot navigation is becoming one of the most important research field due to the high number of possible applications both in everyday life (home automation) and in research fields (planetary exploration, regions surveillance, rescue missions, ruins exploration).

For any mobile device, whatever is the robot application, the ability to navigate in an environment represents a crucial requirement. Robot navigation means the robot's ability to determine its own position in its reference frame (localization) and then to plan a path to a goal location (path planning). Moreover, to navigate in its environment, the robot or any other mobile device requires a representation (a map) of the environment and the ability to interpret this representation (mapping).

Mobile robots localization

The aim of the mobile robots localization problem is to use all the available information, by sensors and by a-priori knowledge on the environment where the robot moves, to localize the robot. This is probably one of the main problems in mobile robotics. To plan a path for a mobile robot or to map the environment surrounding the robot, it is mandatory to known **where the robot is or is assumed to be** in the environment.
More formally, the problem can be stated as follows:

> *assume the mobile robot modeled as a set of nonlinear equations*
>
> $$\dot{x}(t) = f(x(t), u(t))$$
>
> *where $x(t)$ is the robot state, consisting of robot position and orientation, and $u(t)$ is a vector containing actuators inputs. Let the robot sensors measurements be modeled as*

$$y(t) = h(x(t))$$

The goal is to obtain a reliable estimation, $\hat{x}(t)$, of the robot pose (position and orientation) $x(t)$, using all the available information from sensors ($y(t)$) and actuators ($u(t)$).

In a schematic point of view, the aim is to develop a localization module able to provide the estimated robot pose starting from the robot inputs and outputs, as shown in Figure I.3.

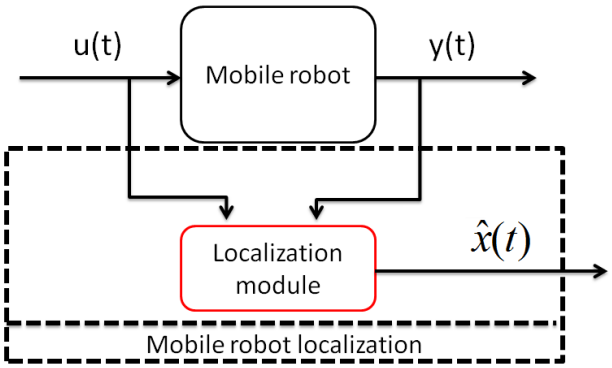


Fig. I.3: Mobile robot localization scheme

This problem deals with a very big amount of applications. For example, in the cited home automation contexts, the cleaner robots need to localize themselves into the room they have to clean. Thanks to this localization, these robots are able to define which floor parts have been cleaned and which ones have not been visited. Whatever is the application, if an exploration task has to be accomplished then it is required to solve at least the localization problem.

Path planning

Given a mobile robot, modeled as

$$\dot{x}(t) = f(x(t), u(t)),$$

placed in an initial configuration $x(t_s) = x_s$, the goal of the path planning problem is to move the robot from x_s to a desired configuration $x(t_f) = x_f$ following a desired trajectory $\chi(t), t \in [t_s, t_f]$. More formally, the path planning strategies aim to find the control moves, $u(t), t \in [t_s, t_f]$ which let the robot arrive in x_f, starting from x_s and following as best as possible the trajectory $\chi(t)$.

In a schematic point of view, the aim is to develop a path planning module able to provide the appropriate control moves to the robot, as shown in Figure I.4.

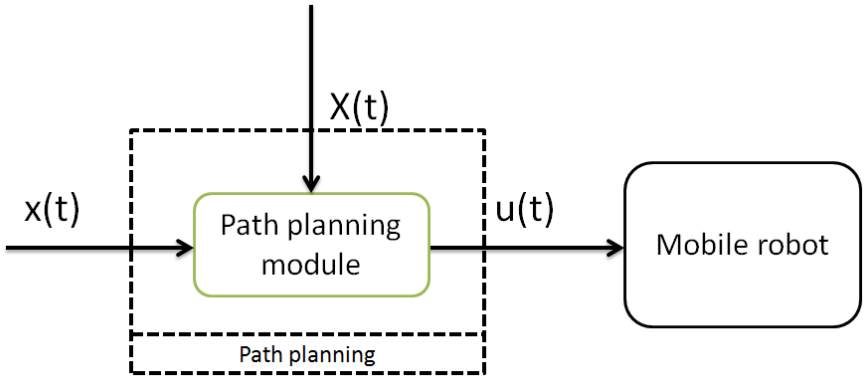


Fig. I.4: Path planning scheme

Obviously, the path planning module needs to know the actual robot pose $x(t)$ and thus the obtained control moves are a function of the desired trajectory $\chi(t)$ and of the robot pose $x(t)$.

In most applications, the actual robot pose is not directly provided by robot sensors and it has to be obtained through a localization technique. The global scheme can be summarized as shown in Figure I.5. Just to give an example of path planning applications, thinking at planetary exploration tasks, the mobile robots which have to move on an unknown planet have to detect their surrounding environment, they have to localize themselves into this environment and then they have to plan a path to perform in order to explore the region in the most efficient way. Also in this case, whatever is the application, the path planning task has to be accomplished in all the situations concerning regions exploration, navigation or inspection.

Environment mapping

The environment mapping is a task to perform in parallel with the localization task and the path planning task. The aim is to build a robot surrounding environment map as more as possible accurate and reliable. This task can be accomplished in both indoor and outdoor environments and the provided map represents a necessary information to solve the localization problem and the path planning problem.

In particular in all the environments where the GPS can not be used (indoor environment, forests, etc...), if no preliminary assumptions on the

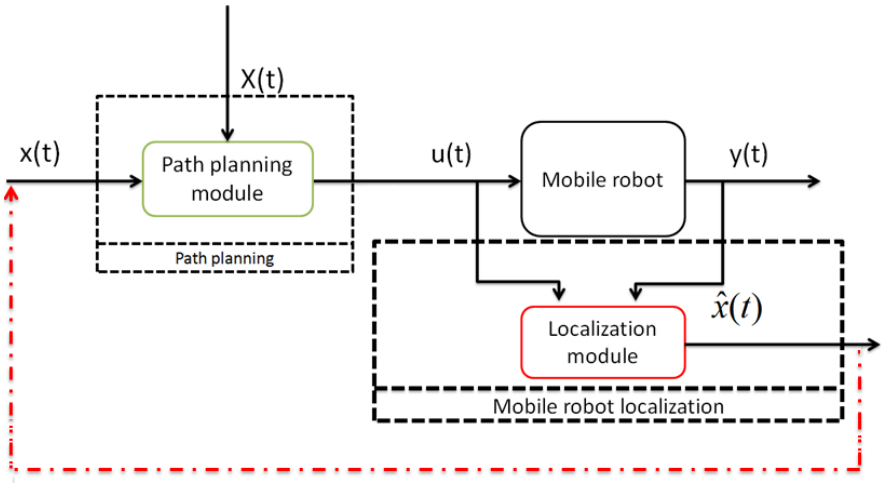


Fig. I.5: Path planning + localization scheme

environment structure is done, then the localization task can be accomplished only if the robot builds an environment map ant it localizes itself within this map.

Also for what regards the path planning, to avoid collisions with walls or obstacles, adapting the trajectory $\chi(t)$ to the environment, an environment map is required. More formally, the environment mapping problem is the problem of modeling the robot surrounding environment and of estimating its model parameters. In a schematic point of view, a mapping algorithm can be seen as a module which uses the sensors measurements to provide a structure containing all the information required to build the environment map (see Figure I.6).

As shown in Figure I.6, to obtain an environment map, the actual robot pose is needed and, as in the path planning case, a robot pose estimation can be use instead of real robot pose. This estimation is provided by the localization module; the overall scheme is depicted in Figure I.7.

If all the modules in Figure I.7 are correctly developed, than the resulting overall algorithm can be used to answer to the three main questions:

- Where the robot is? → localization task and mapping task
- Where the robot should go? → localization task and mapping task
- How the robot should arrive in its goal location? → mapping task and path planning task

In particular, the loop formed by the localization module, which uses the map provided by the mapping module and from the mapping module, which uses the robot pose estimation to obtain the map, is at the basis of the so

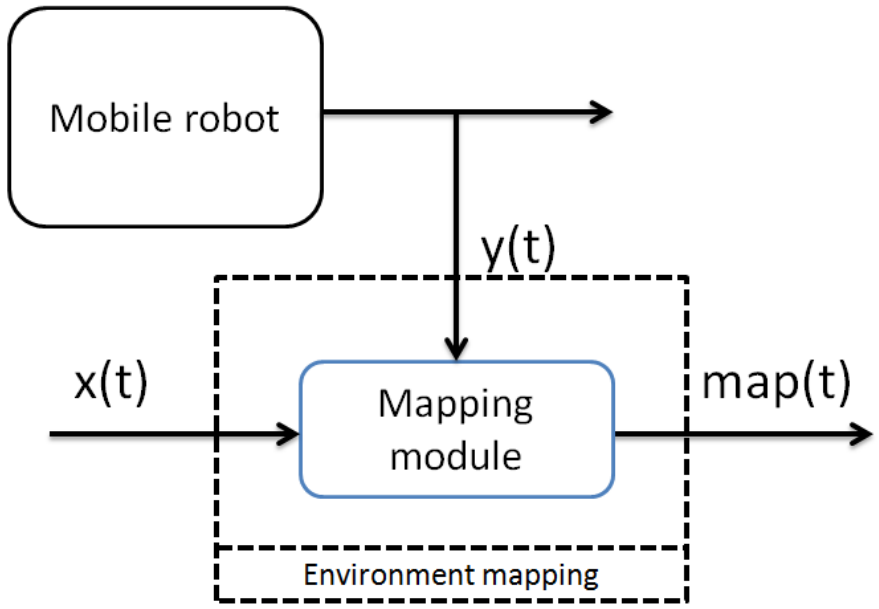


Fig. I.6: Environment mapping scheme

called Simultaneous Localization and Mapping (SLAM) problem (see Figure I.8).

The mobile robots SLAM problem can be seen as an extension to the localization problem. In the SLAM problem the robot localizes itself into the environment where it moves and it simultaneously builds a map of this environment. In mobile robotics, the SLAM problem is considered as ***chicken and egg*** problem due to the very strong correlation between the mapping task and the localization task: **the robot, to localize itself within its environment, needs to build a map of this environment but this map can be built only if the robot position is known**.

The SLAM problem is probably the mobile robotics topic which finds the bigger amount of applications in various fields. Especially in recent years, mobile robots started to be used in place of humans to accomplish difficult or dangerous tasks. For example:

- rescue applications, in which team of mobile robots are used to search and rescue people in disaster regions;
- surveillance applications, which regards the use of a single mobile robot or of a team of robots to watch regions of interest;
- exploration applications, in which mobile robots are used to build a map of a place in an autonomous way, without any required help by humans.

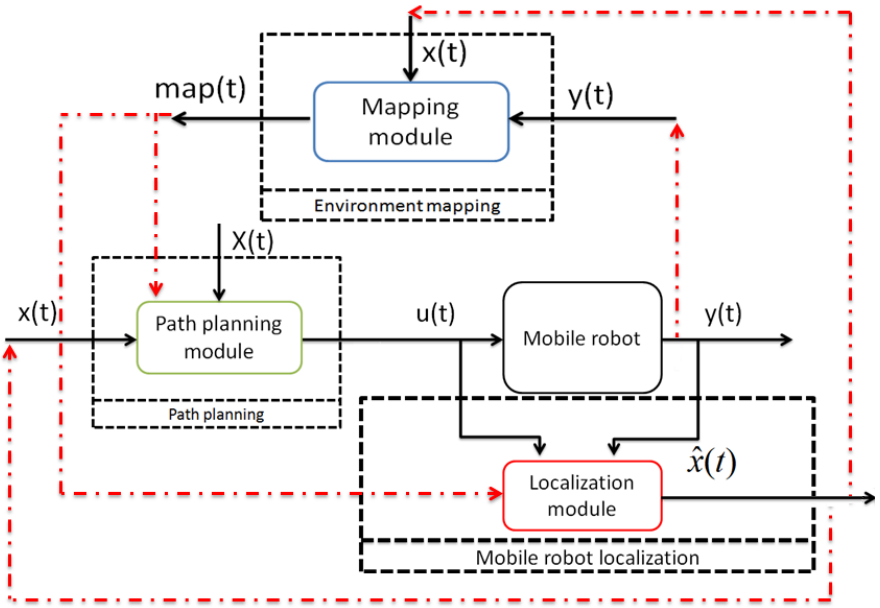


Fig. I.7: Localization + Path planning + Mapping scheme

In all these applications the SLAM problem has to be solved and the resulting performance are completely related to the SLAM algorithm performance.

Personal contribution

The most of the localization, mapping and SLAM techniques proposed in the literature are based on at least some assumptions on the robot surrounding environment and very often this environment is modeled in a very approximated way. Moreover, all the SLAM algorithms deal with the trade off between localization/mapping accuracy and overall algorithm computational cost and the proposed SLAM solutions are usually based on a massive use of the available sensors, neglecting sensors batteries saving.

The aim of the present thesis is to develop a set of localization, mapping and SLAM techniques which have to:

1. be based on very few and weak assumptions on the mobile robot surrounding environment;
2. provide as accurate as possible localization and mapping results;
3. be fast enough to allow their use in real applications;
4. be as cheap as possible in terms of used sensors (number of sensors and sensors types very influence this point);

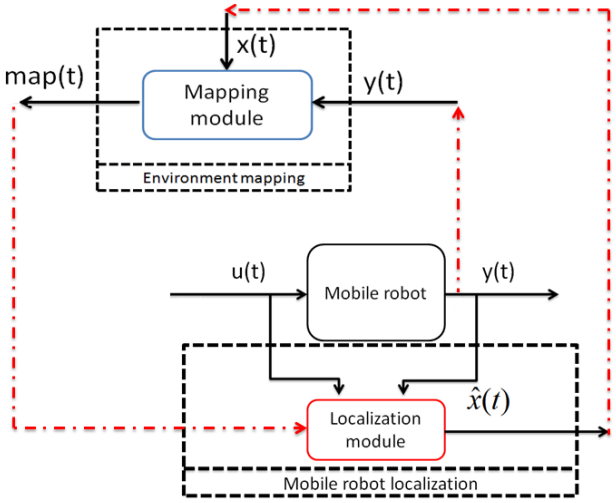


Fig. I.8: Localization + Mapping scheme → SLAM scheme

5. make use of the available sensors in the most possible efficient way in order to save sensors batteries;
6. be developed in a general form in order to allow the use of each technique in various frameworks and using various sensors types.

In this context, a set of new techniques to solve the localization problem, the mapping problem and the SLAM problem, will be described in the following Chapters. All these solutions have been developed trying, step by step, to accomplish all the previously described goals. In particular, as a first step, some techniques to localize the robot on the basis of a complete knowledge about the robot environment will be shown. As a second step, the required information about the environment will be reduced and a set of solutions based on very weak assumptions on the robot working framework will be proposed. As a final step, these techniques will be improved in order to obtain acceptable computational costs. All the obtained solutions will be tested through numerical simulations and real experiments.

Some of the results presented in the following of this thesis have been published and presented in international journals and during International Conferences. More precisely, regarding the localization problem, it has been faced in [74, 5, 26, 25, 13, 68, 77, 67, 79]; mapping problem solutions and SLAM problem solutions have been shown in [43, 44, 75, 76, 78] .

Thesis overview

The dissertation is organized as follows:

- **Chapter** 1 focuses on the mobile robot localization and mapping framework description. The mathematical model of a mobile robot moving in a planar environment is described and the most common sensors used to solve the localization and mapping problems are discussed.
- **Chapter** 2 describes all the developed mobile robots localization techniques. Starting from the simpler localization problem in a perfectly known environment, facing then the problem in a partially known environment and, finally, proposing solutions in a completely unknown environment.
- **Chapter** 3 contains an accurate description of all the developed SLAM solutions. Three main solutions will be shown: a first, very simple, one which is fast in providing mapping and localization results but it yields to approximate environment maps; a second solution which provides an accurate environment description but it requires high computational costs; finally, a third solution, which can be seen as an improvement of the latter two ones, yielding to mapping results as good as the ones obtained using the second technique but with a computational cost as low as the one required by the first technique.
- **Chapter** 4 focuses on a set of rules and policies to optimally use the available sensors in order to decrease their energy consumptions without affecting their contribution to solve the localization, mapping and SLAM problems.
- **Chapter** 5 shows a set of alternative mobile robots localization algorithms based on the use of cameras and Inertial Measurement Units. Differently from what is shown in Chapter 1, these techniques are based only on sensors characteristics, they do not use information about the robot model and they have been developed to work only in static context or during very slow robot movements.
- **Chapter** 6 shows all the numerical simulations and real experiments performed to properly test the algorithms described in the first 5 Chapters.

Finally, conclusions, showing the achieved goals and describing some possible future investigations, are drawn.

1

Localization and Mapping framework

This Chapter focuses on the main features about the framework in which the localization and mapping problems will be solved. The following discussion can be seen as a setup description common to both the second chapter and the third chapter. The mathematical model of a mobile robot moving in a planar environment will be described and the most common sensors used to solve the localization and mapping problems will be discussed.

1.1 Introduction

Starting from the scheme shown in Figure I.7, in this thesis the attention will be paid on the localization and mapping modules design, assuming to be able to properly control the robot. Looking at the literature about localization and mapping problems, many works can be found facing the problem in different ways and using various sensors types.

In particular, the literature shows three main topics about localization and mapping algorithms:

1. used sensors
2. assumptions on the robot surrounding environment
3. algorithms computational cost

In the next Sections, the mobile robot model will be described and a little description about the above three localization and mapping topics will be provided.

1.2 Robot model

In the present thesis, the case of a mobile robot placed in indoor planar environments will be considered. Once the environment reference frame has been

chosen, the mobile robot, due to the environment planarity, can be characterized using only two variables to describe the robot position and one variable to describe the robot orientation. Whatever is the mobile robot type, it has three degree of freedom (d.o.f.), described as depicted in Figure 1.1.

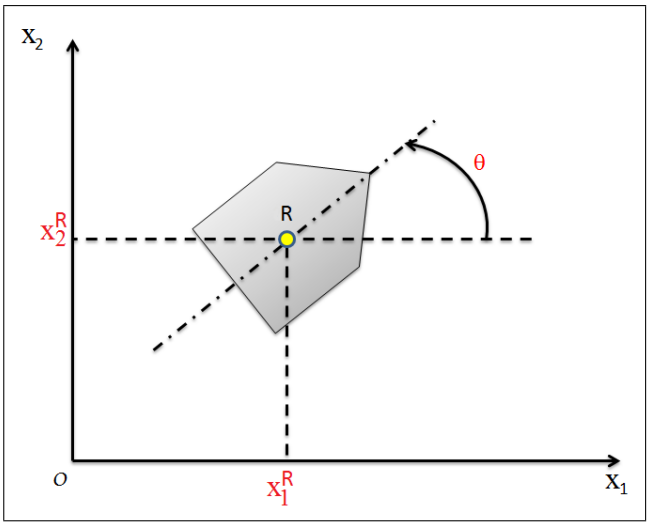


Fig. 1.1: Mobile robot d.o.f.

Let the following notation be used:

- O_{x_1,x_2} is the chosen absolute environment reference frame.
- R is the mobile robot center of gravity.
- (x_1^R, x_2^R) are the mobile robot center of gravity coordinates.
- θ is the orientation of the robot w.r.t. the x_1-axis, considered positive in counterclockwise.

In mobile robots applications only the robot kinematic model is typically used thanks to the very common assumptions of (1) very slow robot dynamics w.r.t. the robot motor dynamics and (2) very low robot accelerations. The mobile robot motors can thus be considered as static systems and the entire robot dynamics can be neglected. In this context the robot center of gravity R can be denoted as **robot center** and the mobile robot shown in Figure 1.1 can be described by only three differential equations, one equation for each degree of freedom.

More precisely, the following non linear differential equations can be defined

$$\begin{cases} \dot{x}_1^R(t) = f_1(x_1^R(t), x_2^R(t), \theta(t), u(t)) \\ \dot{x}_2^R(t) = f_2(x_1^R(t), x_2^R(t), \theta(t), u(t)) \\ \dot{\theta}(t) \quad = f_\theta(x_1^R(t), x_2^R(t), \theta(t), u(t)) \end{cases} \tag{1.1}$$

where $u(t)$ is an array containing all the robot inputs.

Defining the robot state $x(t) = [x_1^R(t)\ ,\ x_2^R(t)\ ,\ \theta(t)]^T$, the equations (1.1) are summarized by

$$\dot{x}(t) = f(x(t), u(t)). \tag{1.2}$$

The above equations can be discretized using the Euler forward method [80, 81], obtaining:

$$x_{k+1} = \phi(x_k, u_k) = x_k + Tf(x_k, u_k) \tag{1.3}$$

where

- T is the sampling period.
- x_k, u_k are the robot state and the robot inputs vector at time $t_k = kT$[1].
- the function $f(\cdot, \cdot)$ is the state update function shown in (1.2).

In the following of this thesis, the mobile robot will be always modeled as a set of non linear difference equations affected by noise:

$$x_{k+1} = \phi(x_k, u_k) + w_k \tag{1.4}$$

where $w_k = [w_{1,k},\ w_{2,k},\ w_{\theta,k}]^T$ is a Gaussian noise, denoted as **process noise**, with zero mean and covariance matrix W. This process noise takes into account unmodeled dynamics, friction, wheels slipping and also, if the case, external disturbances (such as wind).

1.3 Sensors used in mobile robotics applications

The localization and mapping algorithms are very influenced by the used sensors types and the same algorithm can be very efficient using some sensors but it can yield to poor performance using different ones. In the literature, the most *popular* sensors, used to solve the localization problem and the mapping problem, are ultrasonic sensors, laser sensors and cameras.

1.3.1 Ultrasonic Sensors

Ultrasonic sensors are widely used in many applications thanks to their simplicity, availability, and low cost. Such sensors measure (within tolerance) the distance to the surface intercepted by their beam. These sensors are the best sensors for detecting liquids, clear objects, or irregularly shaped objects. They work on a principle similar to radar which evaluate attributes of a target by

[1] In the next Chapters and Sections it will be denoted the time step as $k \in \mathbb{N}$ neglecting the T dependencies. This notation is justified by the isomorphism between the set $\mathbb{N}$ and the set of time steps $\mathbb{T} = \{t \in \mathbb{R} : t = kT\ \&\ k \in \mathbb{N}\}$

Fig. 1.2: Ultrasonic sensor example

interpreting the echoes from radio waves. Ultrasonic sensors generate high frequency sound waves and evaluate the received back echo. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object. Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18.000 hertz, by turning electrical energy into sound, then, upon receiving the echo, the sensors turn the sound waves into electrical energy which can be measured and displayed.

The sensors achieved performance is very affected by the detected objects surface, material density and material consistency. Moreover, looking at a typical ultrasonic beam shape, depicted in Figure 1.3, a set of troubles related to the use of sonar sensors emerge.

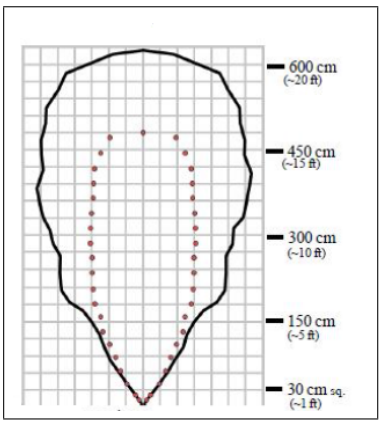


Fig. 1.3: Ultrasonic sensor beam shape

In a nominal point of view, an ultrasonic sensor should measure the distance from an object in front of the sensor. However, due to the sensor beam shape, the obtained measurement can be related to an object which is in the sensor shape but not in front of the sensor itself. This situation is depicted in Figure 1.4

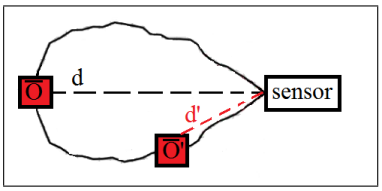


Fig. 1.4: Error in sensor measurement

If the sensor beam shape was a straight line then the obtained measurement would represent the distance d to the object $\overline{O}$. However, due to the beam shape, the object $\overline{O}'$ could be detected and the obtained measurement will be d'. In this situation the measured distance is d' while the nominal expected measured distance is $d(\neq d')$, thus the measurement provided by the ultrasonic sensor is erroneous w.r.t the nominal distance.

Moreover also when no "fake "obstacles (like $\overline{O}'$) are in the sensor shape, the measurement provided by an ultrasonic sensor can be erroneous due to sensors physical characteristics. As remarked in [1] and [2], the measurements provided by ultrasonic sensors are really influenced by the incidence angle between the sensor beam and the intercepted surface. Take in consideration the situation shown in Figure 1.5. When the sensor axis is orthogonal to a flat surface, the measurement provided is the true range, within tolerance, to the surface. However, the measurement error can be larger when the beam strikes a surface at a different **incidence angle**, γ. Consider an ultrasonic sensor, S, which provides the distance, y, from a surface U, as depicted in Figure 1.5.

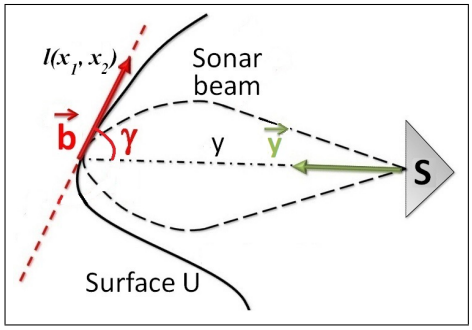


Fig. 1.5: Incidence angle

Let $l(x_1, x_2)$ be the tangent line to the surface boundaries in the intersection point, A, between the incidence surface and the sensor axis. Defining $\overrightarrow{y}$ as the sensor axis unit vector and $\overrightarrow{b}$ as the $l(x_1, x_2)$ unit vector, the incidence angle γ is defined as

$$\gamma = \arccos(\overrightarrow{y} \cdot \overrightarrow{b}) \tag{1.5}$$

The more the incidence angle is near to $\frac{\pi}{2} rad$, the better the measurement provided by the ultrasonic sensor is.

In conclusion the measurement provided by an ultrasonic sensor can be modeled as a function

$$y = y(d, \gamma, v) \tag{1.6}$$

where

- d is the nominal distance from the sensor to the incidence surface of the detected object;
- γ is the incidence angle between the incidence surface and the sensor axis;
- v is a noise used to model the standard measurement error.

Please note that the troublesome situation depicted in Figure 1.4 is modeled using v while the situation shown in Figure 1.5 is modeled using γ. Moreover due to v and γ the obtained measurement y is always $y \neq d$.

In the literature ultrasonic sensors are widely used to solve the localization problem and the mapping problem. Just a few examples can be found in [32, 31, 33].

1.3.2 Laser sensors

A laser sensor is a device which uses a laser beam to determine the distance to an object. The most common form of laser sensors operates on the time of flight principle by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender.

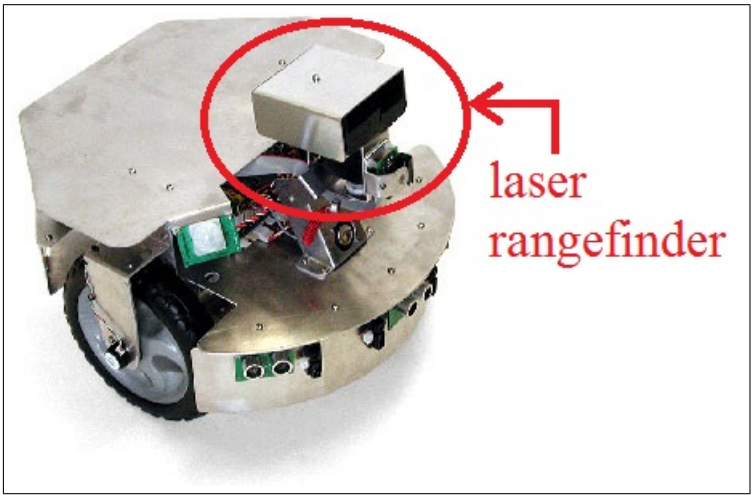


Fig. 1.6: Laser rangefinder placed on a mobile robot

A laser rangefinder provides a set of measurements related to the entire sensor surrounding environment, as shown in Figure 1.7. Laser rangefinders can be "easily"used to obtain an environment map and a robot position estimation since these sensors provide a large number of measurements and these measurements are very accurate.

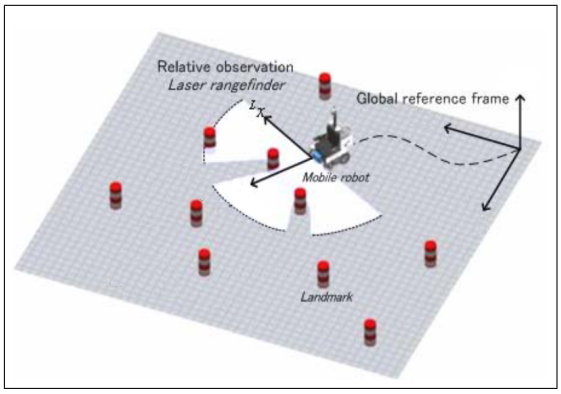


Fig. 1.7: Laser rangefinder output example

Regarding the localization and mapping problem using laser sensors, concrete examples can be found in [6, 40].

1.3.3 Cameras

A camera is a device that records images that may be photographs or moving images such as videos or movies. The term camera comes from the word *camera obscura* (Latin for "dark chamber"): an early mechanism for projecting images; the modern camera evolved from the camera obscura.

Cameras may work with the light of the visible spectrum or with other portions of the electromagnetic spectrum. A camera generally consists of an enclosed hollow with an opening (aperture) at one end, for light to enter, and a recording or viewing surface for capturing the light at the other end. A majority of cameras have a lens positioned in front of the camera's opening to gather the incoming light and focus all or part of the image on the recording surface. The diameter of the aperture is often controlled by a diaphragm mechanism, but some cameras have a fixed-size aperture. Most cameras use an electronic image sensor to store photographs on a flash memory. Other cameras, especially the majority of cameras from the 20th century, use photographic film.

It has long been known that a simple pin-hole is able to create a perfect inverted image on the wall of a darkened room, as shown in Figure 1.9.

Fig. 1.8: Mobile robot equipped with a camera

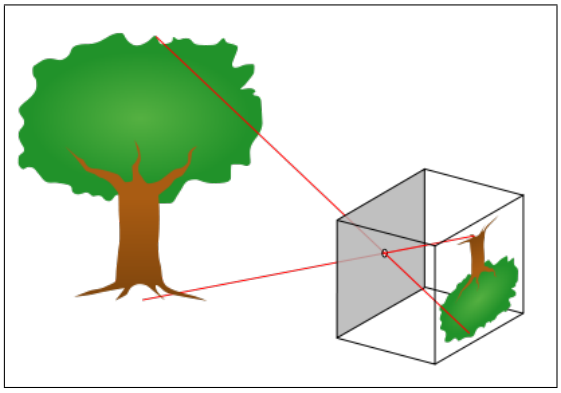


Fig. 1.9: Pinhole model example

A digital camera is similar in principle; a glass or plastic lens forms an image on the surface of a semiconductor chip with an array of light sensitive devices to convert light to a digital image. The process of image formation, in an eye or in a camera, involves a projection of the 3-dimensional world onto a 2-dimensional surface. The depth information is lost and starting from the image it is not possible to tell whether it is of **a large object in the distance** or **a smaller closer object**. This transformation from the 3-dimensional world onto the 2-dimensional one is known as **perspective projection** and it will be discussed in Chapter 5.

About using cameras to solve the localization and mapping problems, examples can be found in [41, 42].

1.4 Assumptions on the robot surrounding environment

For what concerns the assumptions on the robot surrounding environment, most of the previously cited works assume to have at least some information about the environment. For example in [25] and [5] the localization task is accomplished assuming the robot placed in a totally known rectangular environment. In [36] the environment is assumed to be partially known while in [37], the robot is placed in an environment the boundaries of which are assumed to be orthogonal-parallel lines.

As a general rule, the more the assumptions on the robot environment are strong, the simpler will be to solve the localization task and the mapping task.

1.5 Algorithms computational cost

In a computational point of view, it is mandatory for a localization algorithm and for a mapping algorithm to satisfy time constraints since information provided by these algorithms is typically used to compute the output of a control law designed to make the robot follows a given trajectory or completes a given task (see the path planning part in the scheme depicted in Figure I.7).

The localization problem when a map of the environment is available has been solved before with efficient algorithms [45, 46]. Similarly, there are well proven and efficient techniques for the generation of environment maps using observations obtained from known locations [47]. However, localizing the robot in a totally unknown environment or simultaneously provide a robot pose estimation and an environment map is a more challenging problem and optimal SLAM approaches, based on Bayesian filtering, could be extremely expensive making them difficult to apply in real time.

In this context, in [44] the proposed robot surrounding environment model is very accurate and versatile but it requires a very high computational effort and, as a consequence, the resulting SLAM algorithm is not feasible in real time applications. On the contrary, the algorithm shown in [43] is based on a more approximate environment model but it is very computationally efficient.

As a general rule, the more the provided mapping and localization results are accurate, the higher the computational costs due to these algorithms are.

2

Mobile Robots Localization

This Chapter describes a set of new mobile robots localization techniques. Starting from the simpler localization problem in a perfectly known environment, the localization problem will be faced in a partially known environment and, finally, in a completely unknown environment.

2.1 Introduction

The mobile robots localization problem is the problem of localizing a robot within its environment. It finds applications in all the mobile robots contexts: home automation applications, planetary exploration, rescue missions, surveillance tasks require to know the robot pose (position and orientation). A very few robotics frameworks allow to directly know the robot pose from sensors. For examples, outdoor mobile robotics applications use a GPS but the measurements provided by this sensor are always noisy and very often they are too approximative to be directly used to localize the robot in an accurate way. To overcome this problem, these measurements are **fused** with other measurements from other sensors to improve the localization results.

In indoor environments, the GPS cannot be used and thus the localization task has to be solved using other sensors types. In the following, the robot will be assumed to be equipped with distance sensors able to provide measurements about the distance of the robot from the sensors detected objects. These sensors could be, for example, ultrasonic sensors, laser sensors or properly adapted cameras. The goal of this chapter is to find, describe and develop solutions to the Mobile Robots Localization Problem in indoor planar environments. More formally:

> *given a mobile robot, described by the state equation* (1.4) *and equipped with a set of distance sensors, the goal is to localize the robot within its environment, providing an estimate of the robot position and orientation (**robot pose**).*

2.2 Proposed Solutions

Robot localization methods can be classified into two main groups [9]: (1) relative positioning methods, and (2) global or absolute positioning methods. The first group (also called dead-reckoning) achieves positioning by odometry, which consists of counting the number of robot wheels revolutions to compute the offset relative to a known initial position. Odometry uses the robot model to estimate the robot movements starting from the robot inputs and it is very accurate for small offsets but it is not sufficiently accurate for modeling bigger offsets, because of the unbounded accumulation of errors over time (due to wheel slippage, imprecision in the wheel circumference, or wheel inter axis). Furthermore odometry needs an initial position and fails when the robot is *waken-up* (after a forced reset for example) or is raised and dropped somewhere, since the reference position is unknown or modified.

Due to the above described reasons, a global positioning system is thus required to recalibrate the robot position periodically. There are essentially two kinds of global positioning systems:

1. techniques based only on the sensors measurements: triangulation or trilateration;
2. techniques based on the sensors measurements and on the robot model: Kalman filter based techniques.

Regarding the first family of methods: the triangulation is the geometrical process of determining the location of a point by measuring angles to it from known points; the trilateration methods involve the determination of absolute or relative locations of points by distance measurements. Because of the large variety of angle measurement systems, triangulation has emerged as a widely used, robust, accurate, and flexible technique [11]. Various triangulation algorithms may be found in the literature [10, 12, 11]; both triangulation and trilateration methods yield to the robot position estimation at time $t_k = kT$ using only measurements provided at step k. Thanks to this property, these methods are usually less computationally onerous than the techniques based on sensors measurements and robot model. However due to the lack of knowledge about the measurements *history*, the triangulation and trilateration techniques may be more affected by measurements noise.

In this thesis, all the proposed mobile robots localization solutions are related to the use of the Kalman filter theory [82, 23]. Unlike triangulation and trilateration methods, the Kalman filter uses the entire available information on acquired measurements until time t_k to obtain the pose estimation at step k. In particular the Kalman filter theory is widely used in robotics applications, concrete examples can be found in [3, 5, 13].

2.3 Kalman Filter and its nonlinear extensions

The Kalman filter is an optimal state estimator for a linear model influenced by zero-mean Gaussian noise. Consider the discrete-time linear time-invariant system

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k \quad\;\; = Cx_k + v_k \end{cases} \tag{2.1}$$

where

- $x_k \in \mathbb{R}^n$ is the system state vector.
- $u_k \in \mathbb{R}^m$ is the system input.
- $y_k \in \mathbb{R}^p$ is the system output as measured by sensors.
- $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{p\times n}$ are the system dynamic matrix, input matrix and output matrix respectively; more formally, the matrix A describes the dynamics of the system, that is, how the states evolve with time; the matrix B describes how the inputs are coupled to the system states and the matrix C describes how the system states are mapped to the observed outputs.
- $w_k \sim N(\emptyset, W)$ is the process noise and it is a Gaussian noise with zero-mean and covariance matrix $W \in \mathbb{R}^{n\times n}$. This Gaussian disturbance takes into account for unmodeled dynamics and external disturbances on the state evolution.
- $v_k \sim N(\emptyset, V)$ is the system measurement noise and it is a Gaussian noise with zero-mean and covariance matrix $V \in \mathbb{R}^{p\times p}$. It models the measurements noise due to the imperfections in sensors measurements model. v_k and w_k are assumed to be **statistically uncorrelated**.

The general problem that the Kalman filters aims to solve is:

> *given a model of the system (A, B, C, V, W), the known inputs $u_k, k \leq \overline{k}$, and the noisy sensors measurements $y_k, k \leq \overline{k}$, estimate the state of the system $x_{\overline{k}}$ at step $\overline{k}$.*

For example, in a robotic localization context, x_k is the unknown pose of the robot, u_k contains the commands sent to the robot motors and y_k is a vector containing the measurements provided by robot sensors. The Kalman filter is an optimal estimator for the case where the process and measurement noises are zero-mean Gaussian noises. The filter consists in two main steps. The first step is a prediction of the state based on the previous state and on the inputs that were applied.

$$\begin{aligned} \hat{x}_{k+1|k} &= A\hat{x}_{k|k} + Bu_k \\ P_{k+1|k} &= AP_{k|k}A^T + W \end{aligned} \tag{2.2}$$

where

- $\hat{x}_{k+1|k}$ is the **prediction** of the state x_{k+1} starting from all the available information at step k.
- $\hat{x}_{k|k}$ is the **estimate** of the system state x_k, at step k, based on all the available information at step k.
- $P_{k+1|k}$ is the covariance matrix of the prediction error $e_{k+1|k} = x_{k+1} - \hat{x}_{k+1|k}$.
- $P_{k|k}$ is the covariance matrix of the estimation error $e_{k|k} = x_k - \hat{x}_{k|k}$.

This is an open-loop step and its accuracy depends completely on the quality of the model A and B and on the ability to measure the inputs u_k. To improve the accuracy performance the sensors measurements information is introduced using the so called **innovation** term

$$\nu_{k+1} = y_{k+1} - C\hat{x}_{k+1|k} \tag{2.3}$$

which is the difference between what the sensors measure (y_{k+1}) and what the sensors are predicted to measure ($C\hat{x}_{k+1|k}$). A part of this difference will be due to the noise in the sensors (the measurement noise) but the remaining discrepancy indicates that the predicted state was in error and does not properly explain the sensors observations.

At this point, the second step of the Kalman filter, the update step, uses the Kalman gain

$$L_{k+1} = P_{k+1|k}C^T(CP_{k+1|k}C^T + V)^{-1} \tag{2.4}$$

to map the innovation into a correction for the predicted state, optimally tweaking[1] the estimate based on what the sensors have observed. The resulting state estimation is

$$\begin{aligned} \hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + L_{k+1}\nu_{k+1} \\ P_{k+1|k+1} &= P_{k+1|k} - L_{k+1}CP_{k+1|k} \end{aligned} \tag{2.5}$$

The term $(CP_{k+1|k}C^T + V)$ is the estimated covariance of the innovation and comes from the uncertainty in the state and the measurement noise covariance. If the innovation has high uncertainty in relation to some states, this will be reflected in the Kalman gain which will make correspondingly small adjustment to those states.

The above equations constitute the classical Kalman filter which is widely used in applications from aerospace to econometrics. The filter has a number of important properties. Firstly it is recursive, the output of one iteration is the input to the next. Secondly, it is asynchronous: at a particular iteration if no sensors information is available, the prediction step is just performed

[1] The Kalman gain is the gain which minimizes the covariance matrix $P_{k+1|k+1}$ and it is computed as

$$L_{k+1} = \arg\min_{L_{k+1}} P_{k+1|k+1}$$

with no update step. In the case that there are different sensors, each with its own C, and different sample rates, the update step is just applied using the appropriate y and C.

The filter must be initialized with a reasonable value of $\hat{x}_{0|0}$ and $P_{0|0}$. More precisely, $\hat{x}_{0|0}$ has to be chosen such that $\hat{x}_{0|0} = \mathbb{E}[x_0]$, where $\mathbb{E}[\cdot]$ is the expected value function. The filter also requires the best possible estimates of the covariance of the process and measurement noises (W and V respectively).

2.3.1 The Kalman Predictor

In the literature, as an alternative to the described Kalman filter, there are also works on the use of the so called Kalman Predictor (see [82]) which consists in a simpler algorithm than the standard Kalman filter since it is based only on the prediction step. More formally, while the standard Kalman filter provides the **estimate** $\hat{x}_{k|k}$ of the state x_k, the Kalman predictor yields only the **prediction** $\hat{x}_{k|k-1}$ of this state. In a mathematical point of view, the filter equations are

Kalman Predictor

$$
\begin{aligned}
L_k &= AP_{k|k-1}C^T(CP_{k|k-1}C^T + V)^{-1} \\
\hat{x}_{k+1|k} &= A\hat{x}_{k|k-1} + Bu_k + L_k(y_k - C\hat{x}_{k|k-1}) \\
P_{k+1|k} &= (A - L_kC)P_{k|k-1}(A - L_kC)^T + W + L_kVL_k
\end{aligned}
$$

where $P_{k|k-1}$ is the prediction error covariance matrix related to the prediction error $e_{k|k-1} = x_k - \hat{x}_{k|k-1}$.

It has been proved that the Kalman predictor has the same properties of the standard Kalman filter. Obviously, in a computational point of view the predictor is less onerous than the filter since the estimation step is not performed. However due the estimation step loss, the prediction thanks to the Kalman predictor is usually worse than the estimation obtained by the Kalman filter.

2.3.2 Kalman Filter non linear extensions

Now consider the case where the system is not linear and it is described by the equations

$$
\begin{cases}
x_{k+1} = \phi(x_k, u_k) + w_k \\
\\
y_k = h(x_k) + v_k
\end{cases}
\tag{2.6}
$$

The function $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the new state in terms of the previous state and of the system inputs. The function $h : \mathbb{R}^n \to \mathbb{R}^p$ maps the state vector to the sensors measurements.

At this point, starting from equations (2.6) various extensions to the standard Kalman filter have been developed. In this thesis two of these extensions will be described and used.

The Extended Kalman Filter

The Extended Kalman Filter (EKF) has been used for many years to estimate the state of nonlinear stochastic systems with noisy measurements, and it has been probably the first concrete application of Kalman work on filtering [17]. The filter is based on the linearization of the nonlinear maps (ϕ, h) of (2.6) around the estimated trajectory, and on the assumption that the initial state and measurement noises are Gaussian and uncorrelated each other. From the computational point of view the EKF is simply a time-varying Kalman filter where the dynamic and output matrices are given by

$$A_k = \left.\frac{\partial \phi(x, u_k)}{\partial x}\right|_{x=\hat{x}_{k|k}}, \quad C_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_{k|k-1}}, \tag{2.7}$$

and its output is a sequence of state estimates $\hat{x}_{k|k}$ and matrices $P_{k|k}$.

Starting from the given $(\hat{x}_{0|0}, P_{0|0})$, the Extended Kalman filter steps are [23]:

Extended Kalman Filter

$$\begin{aligned}
\hat{x}_{k+1|k} &= \phi(\hat{x}_{k|k}, u_k) \\
P_{k+1|k} &= A_k P_{k|k} A_k^T + W \\
L_{k+1} &= P_{k+1|k} C_{k+1}^T (C_{k+1} P_{k+1|k} C_{k+1}^T + V)^{-1} \\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + L_{k+1}(y_{k+1} - h(\hat{x}_{k+1|k})) \\
P_{k+1|k+1} &= P_{k+1|k} - L_{k+1} C_{k+1} P_{k+1|k}
\end{aligned}$$

where $\hat{x}_{k+1|k}$ represents the estimate of x_{k+1} *before* getting the observation y_{k+1} (prediction), and $\hat{x}_{k+1|k+1}$ represents the estimate *after* getting that observation.

In this non-linear context, differently from the linear case, $P_{k|k}$ is only an approximation (because of the linearization) of the estimation error covariance matrix.

The Unscented Kalman Filter

The Unscented Kalman Filter (UKF) has been developed to overcome two main problems of the EKF:

1. the poor approximation properties of the first order approximation;

2. the requirement for the noises to be Gaussian [18, 19].

The basic idea behind the UKF is to find a transformation that allows to approximate the mean and covariance of a random vector, of length n, when it is transformed by a nonlinear map. As a first step, a set of $2\,n+1$ points, called σ-points, are obtained from the original random vector. As a second step, these σ-points are transformed by the nonlinear map and finally starting from the mean and variance of the transformed σ-points, an approximation of the original random vector mean and variance is obtained. Refer to [19, 20] for the theoretical aspects. Regarding the filter approximating properties, it has been shown [19] that, while the EKF state estimate is accurate to the first order, the UKF estimate is accurate to the *third* order in the case of Gaussian noises. The covariance estimate also is accurate to the first order for the EKF, and to the second order for the UKF. The so-called NonAugmented version of the Unscented Kalman filter, which is suitable for additive noises, is here reported. The description follows partially the one given in [21], with some modifications to make it more compact and suitable for a MATLAB [22] implementation.

Unscented Kalman Filter – NonAugmented version

At each step, starting from $\hat{x}_{0|0}$ e $P_{0|0}$, do

1. Compute $B_{k|k} = \sqrt{(n+\lambda)P_{k|k}}$, i.e, the scaled square root of matrix $P_{k|k}$
2. Compute the σ-points matrix

$$\chi_{k|k} = [\hat{x}_{k|k} \quad \hat{x}_{k|k} + B_{k|k} \quad \hat{x}_{k|k} - B_{k|k}] \in \mathbb{R}^{n\times(2n+1)}$$

   There (and in the next equations) the sum of a vector plus a matrix is intended as summing the vector to all the column of the matrix (*à la* MATLAB)
3. Transform the σ-points matrix (columnwise)

$$\chi^*_{k+1|k} = \phi(\chi_{k|k}, u_k)$$

4. Compute the a-priori statistics

$$\begin{aligned}\hat{x}_{k+1|k} &= \chi^*_{k+1|k}\, R^m \\ P_{k+1|k} &= (\chi^*_{k+1|k} - \hat{x}_{k+1|k}) R^c (\chi^*_{k+1|k} - \hat{x}_{k+1|k})^T + W\end{aligned}$$

5. Compute the new σ-points

$$\begin{aligned}B_{k+1|k} &= \sqrt{(n+\lambda)P_{k+1|k}} \\ \chi_{k+1|k} &= [\hat{x}_{k+1|k} \;\; \hat{x}_{k+1|k} + B_{k+1|k} \;\; \hat{x}_{k+1|k} - B_{k+1|k}]\end{aligned}$$

6. Compute the predicted output

$$\begin{aligned}\Gamma_{k+1|k} &= h(\chi_{k+1|k})\\ \hat{y}_{k+1|k} &= \Gamma_{k+1|k}\, R^m\end{aligned}$$

7. Compute the Kalman gain

$$\begin{aligned}P_{yy} &= (\Gamma_{k+1|k} - \hat{y}_{k+1|k})R^c(\Gamma_{k+1|k} - \hat{y}_{k+1|k})^T + V\\ P_{xy} &= (\chi_{k+1|k} - \hat{x}_{k+1|k})R^c(\Gamma_{k+1|k} - \hat{y}_{k+1|k})^T\\ L_{k+1} &= P_{xy}P_{yy}^{-1}\end{aligned}$$

8. Compute the a-posteriori statistics

$$\begin{aligned}\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + L_{k+1}(y_{k+1} - \hat{y}_{k+1|k})\\ P_{k+1|k+1} &= P_{k+1|k} - L_{k+1}P_{yy}K_{k+1}^T\end{aligned}$$

The filter's parameters and weighs are [18]:

$$\begin{aligned}R^m &= [R_1^m \cdots R_{2n+1}^m]^T\\ R^c &= \text{diag}\{R_1^c, \ldots, R_{2n+1}^c\}\end{aligned}$$

where

$$\begin{aligned}R_1^m &= \frac{\lambda}{n+\lambda},\ R_1^c = \frac{\lambda}{n+\lambda} + 1 + \beta - \alpha^2\\ R_j^m &= R_j^c = \frac{\lambda}{2\,(n+\lambda)},\ j = 2, \ldots, 2n+1.\end{aligned}$$

and

$$\begin{aligned}&\alpha = 0.001,\ \beta = 2,\ \kappa = 3 - n\\ &\lambda = \alpha^2(n+\kappa) - n\end{aligned}$$

According to [18], the choice $\beta = 2$ minimizes the error in the fourth-order moment of the a-posteriori covariance when the random vector is Gaussian.

Please note that to use the Unscented Kalman filter **no function linearization is required**. The UKF is then less influenced by model error than the EKF and the estimation provided by the UKF is more accurate than the one obtained using the EKF.

2.4 Mobile robots localization in a perfectly known environment

In this Section a solution for the mobile robots localization in a perfectly known environment will be proposed. Assume to have a mobile robot placed

in a rectangular environment the boundaries of which are perfectly known. The robot is assumed to be equipped with n_S distance sensors, placed in known locations w.r.t. the robot center and denoted by $S_i, i = 1, \ldots, n_S$. Figure 2.1 shows the described framework for the case $n_S = 5$.

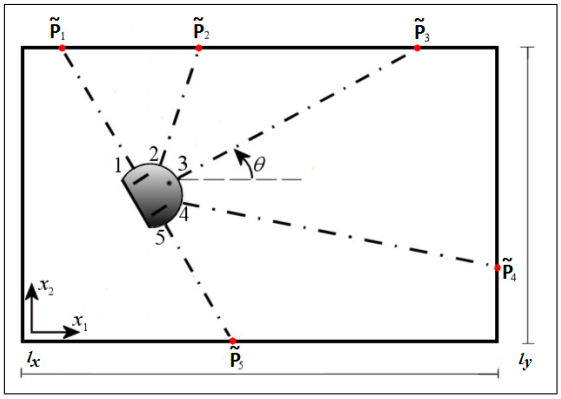


Fig. 2.1: Perfectly known rectangular environment

Assume an absolute reference frame placed in the down-left corner of the rectangular environment. Let l_x, l_y be the environment width and length and let $\alpha_i, i = 1, \ldots, 5$ denote the orientation of the sensors with respect to robot axis (orthogonal to wheels axes). The proposed mobile robot localization algorithm is based on the use of an Extended Kalman filter (EKF) or of an Unscented Kalman filter (UKF). The filters are based on the model (1.4) for what regards the robot evolution and, for what concerns the output model, the function $h(\cdot)$ in the equations (2.6) is used. Since the sensors are placed on the robot, the measurements provided by these sensors are related to the environment shape and thus the output equation has to take into account the environment.

Each distance based sensor S_i provides the distance of the robot center from one point on the environment boundaries, denoted by $\tilde{P}_i$. Starting from the framework depicted in Figure 2.1 (for the case $n_S = 5$), the distances $y_{i,k} = h_i(x_k)$, $i = 1, \ldots, n_S$, measured by each distance sensor S_i, are given by

$$
\begin{aligned}
h_i(x_k) &= \sqrt{(\tan^2(\theta_k + \alpha_i) + 1)(x^R_{1,k})^2} && \text{if } p_i \in e_1,\\
h_i(x_k) &= \sqrt{(\tan^2(\theta_k + \alpha_i) + 1)(x^R_{1,k} - l_x)^2} && \text{if } p_i \in e_2,\\
h_i(x_k) &= \sqrt{(\tfrac{1}{\tan^2(\theta_k + \alpha_i)} + 1)(x^R_{2,k})^2} && \text{if } p_i \in e_3,\\
h_i(x_k) &= \sqrt{(\tfrac{1}{\tan^2(\theta_k + \alpha_i)} + 1)(x^R_{2,k} - l_y)^2} && \text{if } p_i \in e_4,
\end{aligned}
$$

where $e_i, i = 1, \ldots, 4$ denotes the edges of the field, the equations of which are:

$$e_1 : x_1 = 0;\ e_2 : x_1 = l_x$$
$$e_3 : x_2 = 0;\ e_4 : x_2 = l_y$$

These relationships are specific to the described robot environment and define the robot model output equation. In a more compact form, the above relationships can be written as

$$y_k = h(x_k, l_x, l_y) + v_k. \tag{2.8}$$

As expected, the model output equation is a function of the environment parameters l_x, l_y and of the robot state x_k. The vector v_k collects the sensor noises, also assumed Gaussian, zero-mean, with covariance matrix V, and uncorrelated with the process noise w_k.

At this point, using equations (1.4) and (2.8), the Extended Kalman filter and the Unscented Kalman filter presented in Section 2.3.2 can be applied. Thanks to this filter, the robot pose can be estimated and the mobile robots localization problem is solved.

2.5 Mobile robots localization in a partially known environment

The previously proposed solution can be applied in a few situations due to the very strong assumptions on the environment boundaries. In this section the obtained results for a rectangular environment will be extended to a more general scenario.

Assume that the robot is moving in an unknown workspace and assume to model the environment boundaries using a set of segments, each of them intersecting at least one point on the boundaries (see Figure 2.2).
Let O_{x_1,x_2} be the chosen absolute reference frame and let
$\{B_i = (B_1^i, B_2^i), i = 1, \ldots, n_B\}$ be a set of environment boundaries points, the coordinates of which are assumed to be known. The environment is approximated by the segments from B_i to $B_{i+1}, i = 1, \ldots, n_B$ and by the segment starting from B_{n_B} and ending in B_1[2]. Consider the same mobile robot used in the previous Section. The robot is equipped with n_S distance sensors (see Figure 2.2 for the case $n_S = 5$).

Each sensor provides the distance y_i between the robot center, $P = (x_1^R, x_2^R)$, and one point on the surrounding environment, denoted by $\tilde{P}_i = (\tilde{x}_1^i, \tilde{x}_2^i)$:

[2] Please note that if the environment contains obstacles, points on the obstacles boundaries are required too. For example, if two obstacles are placed into the environment, three points sets have to be used: the first one related to the environment, the second one and the third one related to the obstacles. Only segments related to points in the same set are used.

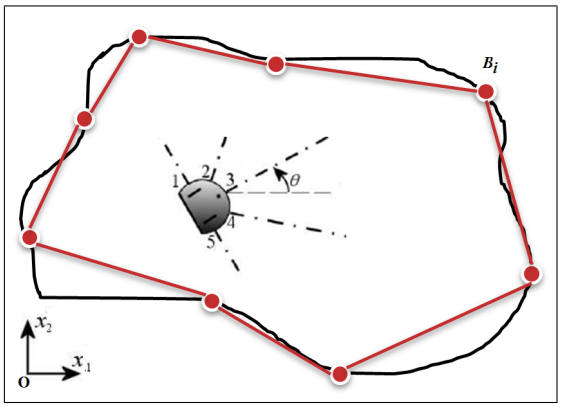


Fig. 2.2: Partially known rectangular environment

$$y_i = \sqrt{(x_1^R - \tilde{x}_1^i)^2 + (x_2^R - \tilde{x}_2^i)^2}. \tag{2.9}$$

The above measurement is approximated by the distance between P and the intersection point, $\bar{P}_i = (\bar{x}_1^i, \bar{x}_2^i)$, between the S_i sensor axis and one of the segments used to model the boundaries (see Figure 2.3).

Denoting the S_i sensor axis by $x_2 = a_i x_1 + q_i$, and the detected segment axis by $x_2 = c_i x_1 + s_i$, the intersection $\bar{P}_i$ is

$$\bar{x}_1^i = \frac{s_i - q_i}{a_i - c_i}, \qquad \bar{x}_2^i = \frac{a_i s_i - c_i q_i}{a_i - c_i}. \tag{2.10}$$

Finally the distance between P and $\tilde{P}_i$ is approximated by:

$$\eta_i = \sqrt{(x_1^R - \bar{x}_i^i)^2 + (x_2^R - \bar{x}_2^i)^2} \approx y_i. \tag{2.11}$$

Figure 2.3 shows in details the proposed observation model for the case $S_i = S_3$.

The S_i axis parameters are given by:

$$a_i = \tan(\theta + \alpha_i),\ q_i = x_2^R - a_i x_1^R, \tag{2.12}$$

where θ is the robot heading. Using (2.10) and (2.12) within (2.11), a distance function η_i depending only on the robot state and on the segment parameters, (s_i, c_i), can be obtained

$$\eta_i = h((x_1^R, x_2^R, \theta), (s_i, c_i)),\ i = 1, \ldots, n_S.$$

These relationships allow to define the robot model output equation:

$$y_k \approx h(x_k, (\bar{s}_k, \bar{c}_k)) + v_k, \tag{2.13}$$

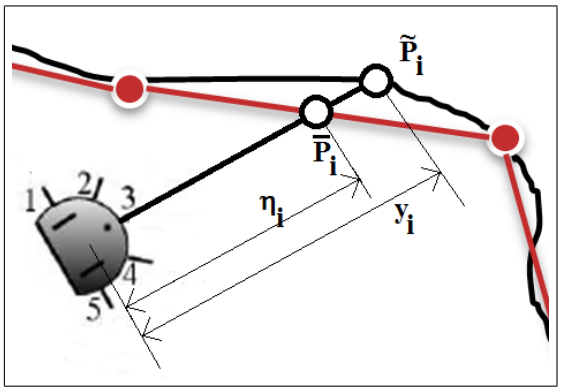


Fig. 2.3: Measurement model

where $x_k = [x^R_{1,k}\ x^R_{2,k}\ \theta_k]^T$ is the robot state at time $t_k = kT$ and the vector v_k collects the sensor noises, again assumed Gaussian and zero-mean, with covariance matrix V and uncorrelated with w_k; the vectors $\bar{s}_k$ and $\bar{c}_k$ contain the parameters (s, c) of all the segments, hit by at least one of the robot sensors, at time t_k. Obviously, these parameters are related to the points $\{B_i = (B^i_1, B^i_2), i = 1, \ldots, n_B\}$.

Using this observation model, a **S**egment based **E**xtended **K**alman **F**ilter (**SEKF**) (or a Segment based Unscented Kalman Filter (SUKF)) can be used to face the localization problem. Starting from the equation (2.13), the linearized output matrix C_k is

$$C_k = \begin{bmatrix} C_1 \\ \vdots \\ C_{n_S} \end{bmatrix}$$

where each row $C_i = [C_{i,1},\ C_{i,2},\ C_{i,3}]$ is a three element array the entries of which are

$$C_{i,1} = \frac{\partial \eta_i}{\partial x^R_1} = \frac{c_i\left(2s_i - 2x^R_2 - \tan(\theta+\alpha_i)^2 + c_i\left(2x^R_1 + \tan(\theta+\alpha_i)\right)\right)}{2(c_i - \tan(\theta+\alpha_i))^2\sqrt{x^R_2 + \frac{(s_i + c_i x^R_1 - x^R_2)^2}{(c_i - \tan(\theta+\alpha_i))^2} + \frac{s_i\tan(\theta+\alpha_i) + c_i(-x^R_2 + x^R_1\tan(\theta+\alpha_i))}{c_i - \tan(\theta+\alpha_i)}}}$$

$$C_{i,2} = \frac{\partial \eta_i}{\partial x^R_2} = \frac{-2s_i + 2x^R_2 + \tan(\theta+\alpha_i)^2 - c_i\left(2x^R_1 + \tan(\theta+\alpha_i)\right)}{2(c_i - \tan(\theta+\alpha_i))^2\sqrt{x^R_2 + \frac{(s_i + c_i x^R_1 - x^R_2)^2}{(c_i - \tan(\theta+\alpha_i))^2} + \frac{s_i\tan(\theta+\alpha_i) + c_i(-x^R_2 + x^R_1\tan(\theta+\alpha_i))}{c_i - \tan(\theta+\alpha_i)}}}$$

$$C_{i,3} = \frac{\partial \eta_i}{\partial \theta} = \frac{\sec(\theta+\alpha_i)^3\left(\cos(\theta+\alpha_i)c_i^2 + c_i\left(-\sin(\theta+\alpha_i) + 2\cos(\theta+\alpha_i)x^R_1\right) + 2\cos(\theta+\alpha_i)\left(s_i - x^R_2\right)\right)\left(s_i + c_i x^R_1 - x^R_2\right)}{2(c_i - \tan(\theta+\alpha_i))^3\sqrt{x^R_2 + \frac{(s_i + c_i x^R_1 - x^R_2)^2}{(c_i - \tan(\theta+\alpha_i))^2} + \frac{s_i\tan(\theta+\alpha_i) + c_i(-x^R_2 + x^R_1\tan(\theta+\alpha_i))}{c_i - \tan(\theta+\alpha_i)}}}$$

At this point, a standard Extended Kalman filter can be used to localize the robot.

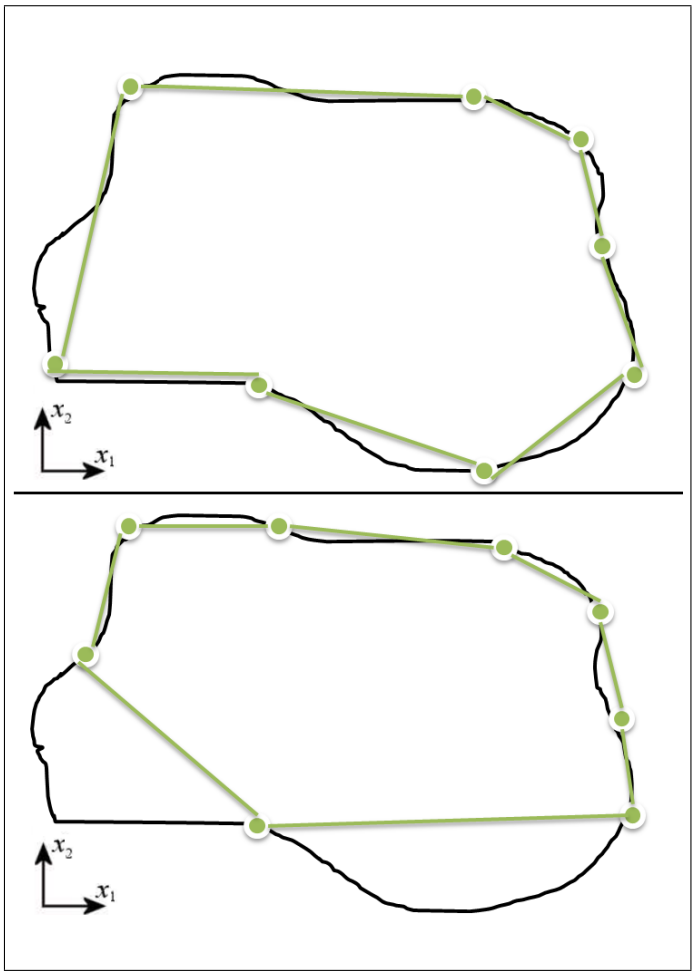


Fig. 2.4: Properly chosen points B_i (UP), bad chosen points B_i (DOWN)

In conclusion, when the environment is partially known, the localization problem can be faced thanks to an EKF based on the model related to the environment boundaries. It is not important to have a very precise model, it is important that at least the main boundaries characteristics are modeled. For example, using the proposed segment based model, it is mandatory to correctly choose the points B_i. Take in consideration the environment shown in Figure 2.4. In the first case, the points B_i have been chosen properly while in the second case the approximation is not good and thus the proposed segment

based filter will fail in the robot localization. To overcome these difficulties, in the following Section a new localization technique will be proposed based on a totally unknown environment and with no assumptions about the environment boundaries.

2.6 Mobile robots localization in a totally unknown environment

The observation structure (2.13) is well-posed only if $(\bar{s}_k, \bar{c}_k)$ are known. In the segment based EKF these parameters are assumed to be known (thanks to the points B_i). On the contrary, if no assumption is done on the robot surrounding environment, then an estimation of the parameters $(\bar{s}_k, \bar{c}_k)$, namely $(\hat{\bar{s}}_k, \hat{\bar{c}}_k)$, has to be found. To this end, the **Neighbors Based Algorithm (NBA)** has been devised.

2.6.1 Neighbors Based Algorithm

The NBA is based on the main idea that information on the whole environment is not needed to localize the robot. Only the environment parts that currently interact with the robot sensors are needed to estimate the current robot position and orientation. Starting from this idea, thanks to the **proximity** between acquired measurements over time, it is possible to estimate the environment parameters needed, at time t_k, to model the sensors output.

More precisely, two points P_1, P_2 are defined as *neighbors* if $||P_1 - P_2|| < R_{NBA}$, where $R_{NBA} > 0$ is an algorithm parameter. Moreover, for a given set of points $\mathcal{A}$ and a point $P \in \mathcal{A}$, the following set-valued *closeness function* $\mathcal{N}$ has been defined

$$\mathcal{B} = \mathcal{N}(P, \mathcal{A}, R_{NBA}) = \{P_i \in \mathcal{A} : ||P_i - P|| < R_{NBA}\}$$

that provides the subset $\mathcal{B}$, containing the points in A which are neighbors, in a radius R_{NBA}, of P.

Once a new measurement $y_{i,k}$ has been acquired by sensor S_i at step k, given the actual robot state estimate $\hat{x}_k$, an approximation, $P^*_{i,k}$, of the environment point $\tilde{P}_{i,k}$ hit by the sensor axis, is computed as:

$$\begin{aligned} P^*_{i,1,k} &= \hat{x}^R_{1,k} + y_{i,k}\cos(\hat{\theta}_k + \alpha_i) \\ P^*_{i,2,k} &= \hat{x}^R_{2,k} + y_{i,k}\sin(\hat{\theta}_k + \alpha_i) \\ P^*_{i,k} &= (P^*_{i,1,k}, P^*_{i,2,k}). \end{aligned} \tag{2.14}$$

This approximation differs from the actual point $\tilde{P}_{i,k}$ because of both the estimation and the measurement errors. An example of the above approximation is shown in Figure 2.5.

Given the set of environment detected points

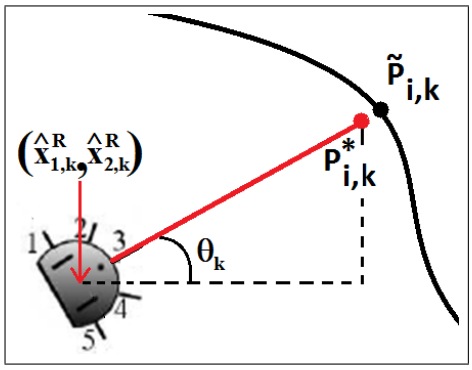


Fig. 2.5: $P^*_{i,k}$ approximating the point $\tilde{P}_{i,k}$

$$\mathcal{M}_k = \{P^*_{i,j}, i = 1, \ldots, n_S,\ j = 0, \ldots, k\}, \tag{2.15}$$

the NBA computes the closeness function of $P^*_{i,k}$ on the set $\mathcal{M}_k$ (see the second picture in Figure 2.6). At this point, the LMS function is used:

$$[b_0, b_1, \ldots, b_z] = LMS(\mathcal{A}, z). \tag{2.16}$$

This function computes the parameters $[b_0, b_1, \ldots, b_z]$ of the z-th order polynomial which best approximates, in a Least Mean Square (LMS) sense, the points contained in the set $\mathcal{A}$. The NBA computes the LMS line approximating the $P^*_{i,k}$-neighbors (see the third picture in Figure 2.6):

$$(\hat{s}_k, \hat{c}_k) = \text{LMS}(\mathcal{N}(P^*_{i,k}, \mathcal{M}_k, R_{NBA}), 1)$$

and the resulting values $\hat{s}_k$, $\hat{c}_k$ are used to approximate the segment detected by the i-th sensor.

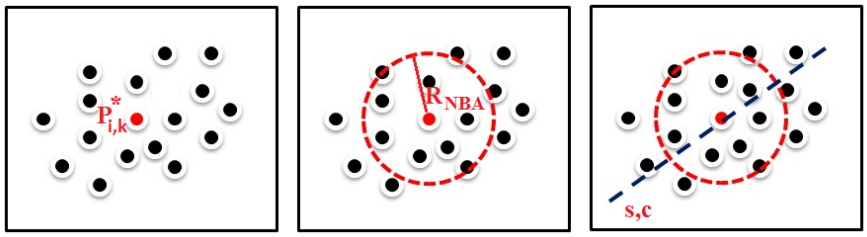


Fig. 2.6: First picture: $P^*_{i,k}$ and previously detected points; second picture: closeness function on $P^*_{i,k}$; third picture: LMS line approximation

In conclusion, the NBA algorithm can be summarized as follows:

Neighbors Based Algorithm

---

for each step k, given $\hat{x}^R_{1,k}, \hat{x}^R_{2,k}, \hat{\theta}_k, \mathcal{M}_{k-1}$, **do**

1. **for each sensor** S_i
   - acquire a measurement $y_{i,k}$ from the sensor S_i
   - compute $P^*_{i,k}$ using (2.14)

   end
2. $\mathcal{M}_k = \mathcal{M}_{k-1} \cup \{\cup_{\text{for each } S_i}\{P^*_{i,k}\}\}$
3. **for each** S_i
   - $(\hat{s}_i, \hat{c}_i) = \text{LMS}(\mathcal{N}(P^*_{i,k}, \mathcal{M}_k, R_{NBA}), 1)$

   end

end

From now on, the NBA algorithm will be used as a function

$$(\hat{\bar{s}}_k, \hat{\bar{c}}_k) = \text{NBA}(\hat{x}_k, y_k)$$

where $(\hat{\bar{s}}_k, \hat{\bar{c}}_k)$ can be used to approximate the vectors $(\bar{s}_k, \bar{c}_k)$ in the output function (2.13).

Using the NBA into the standard Extended Kalman fitler, the mobile robot localization in a totally unknown environment can be faced. The final algorithm is

---

Neighbors based Extended Kalman Filter

$$\begin{aligned}
&\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k)\\
&P_{k+1|k} = A_k P_{k|k} A_k^T + W\\
&K_{k+1} = P_{k+1|k} C_{k+1}^T (C_{k+1} P_{k+1|k} C_{k+1}^T + V)^{-1}\\
&(\hat{\bar{s}}_{k+1}, \hat{\bar{c}}_{k+1}) = \text{NBA}(\hat{x}_{k+1|k}, y_{k+1})\\
&\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - h(\hat{x}_{k+1|k}, (\hat{\bar{s}}_{k+1}, \hat{\bar{c}}_{k+1})))\\
&P_{k+1|k+1} = P_{k+1|k} - K_{k+1} C_{k+1} P_{k+1|k}
\end{aligned}$$

Following the same lines, a Neigbors based Unscented Kalman Filter (NUKF) can be easily stated.

The main drawback of the proposed NEKF is related to the initial condition influence on the resulting estimation performance. Since no preliminary information on the surrounding environment is required, if the initial condition is too wrong the resulting estimation can be affected by a bias w.r.t. the

real robot state. However, this bias problem is probably unavoidable if no assumptions are made on the robot environment.

The above considerations suggest that at least some information on the robot surrounding environment is needed to overcome the NEKF algorithm problems. In the next Section, a new algorithm, developed to overcome the above drawbacks, and based on a small amount of information about the robot surrounding environment, will be described.

2.7 Sensor fusion for the Mobile Robots Localization Problem: the MKF algorithm

All the proposed mobile robots localization techniques are based on the sensor fusion approach. The sensor fusion is the process of combing information from a number of different sources to provide a robust and complete description of a process of interest. In other words, starting from a set of noisy measurements, the sensor fusion techniques are used to obtain a process description as less influenced as possible by measurements noise.

In a control system various sensors types are used to provide as more information as possible and to ensure application robustness; sensors are used to monitor various aspects of the same system (e.g. speed, position, power, temperature, etc...). This information is always influenced by noise due to physical sensors' characteristics or to working environment features. The more the number of sensors providing the same measurement is high, the higher is the probability of extracting the real information from these measurements neglecting the noise effects.

In the mobile robots localization context, as previously shown, the Kalman filter theory can be suitably adapted to solve the localization problem both in a perfectly known environment [25] and in a totally unknown environment [13]. Instead of using robot on board distance sensors to solve the localization problem, a possible alternative consists in using a set of **out of board** distance sensors, placed in **known locations** in the robot environment. Concrete examples can be found in [3, 14, 15, 16]. Using external distance sensors, the key idea is to estimate the robot pose by means of the robot model and of the information about the distances from the robot to the sensors. In this configuration, only the information about the external sensors locations is required. Solutions based on the use of only on board sensors or only out of board sensors have both advantages and drawbacks. In particular, using external distance sensors, good results can be obtained on the robot position estimation, while the orientation estimation can be quite unreliable due to the absence of a real angular information in the sensors measurements; on the contrary, the on board sensors information can be directly related to the robot heading, as shown in the NEKF section (see 2.6.1) and therefore the heading estimation is usually better than the one obtained using external sensors. As a drawback, if on board sensors are used in an unknown environment, models

for the measurements and for the environment have to be built and these models can be very influenced by the robot initial condition estimation.

The described situation is very common in control systems. As previously remarked, various kinds of sensors are used to take information about the system state variables and not all of these sensors can estimate all the state variables with the same performance. For example there could be a first subset of sensors able to provide information only about some of the state variables, while a second subset could be better to estimate the remaining state variables. As a consequence, using only the first or the second subset always yields to state estimation troubles.

In this Section a new sensor fusion technique will be shown based on the idea of emphasizing the qualities and overcoming the defects of each used sensor. Instead of fusing the entire information from all the available sensors, which could result in a very high computational costs if the number of sensors is high, **two different filters** will be used to estimate the same state variables. The first filter will be based only on a first sensors subset's measurements while the second one will be based only on the remaining sensors' measurements. The two obtained state estimates will be then suitably combined. In this way, the computational cost will be quite the same of using a single filter but the global performance should be enhanced. Starting from these ideas, a **Mixed Kalman filter (MKF)** is proposed and tested in a mobile robot application.

In the literature many works can be found about the use of two Kalman filters, however, to the best of the author's knowledge, each filter deals with the estimate of different variables. For example, in [4, 24], the authors use a two stage Kalman filter in order to simultaneously estimate the model parameters and the system's state, but the first filter is used to estimate only the model parameters while the second one provides only an estimation of the system's state.

2.7.1 Mixed Kalman Fiter for LTI systems

Assume to have a linear time-invariant discrete system with two output equations:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ \\ y_k^1 = C^1 x_k + v_k^1 \\ y_k^2 = C^2 x_k + v_k^2 \end{cases} \tag{2.17}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $y_k^1 \in \mathbb{R}^p$ is the first output, $y_k^2 \in \mathbb{R}^q$ is the second output and the noises w_k, v_k^1, v_k^2 are zero-mean uncorrelated Gaussian noises modeling the process noise, the measurement noise on y_k^1 and the measurement noise on y_k^2 respectively. Let W, V^1 and V^2 be the covariance matrices of the noises w_k, v_k^1 and v_k^2. These matrices will be assumed to be known. Moreover an initial prediction of the state $\hat{x}_{0|-1}$ and its related prediction error covariance matrix $P_{0|-1}$ are assumed to be known.

To estimate the system state a possible solution is to use a standard Kalman filter based on the predictor structure (see Section 2.3.1):

Kalman Filter (predictor structure)

$$
\begin{aligned}
L_k &= AP_{k|k-1}C^T(CP_{k|k-1}C^T + V)^{-1} \\
\hat{x}_{k+1|k} &= A\hat{x}_{k|k-1} + Bu_k + L_k(y_k - C\hat{x}_{k|k-1}) \\
P_{k+1|k} &= (A - L_kC)P_{k|k-1}(A - L_kC)^T + W + L_kVL_k
\end{aligned}
$$

where $\hat{x}_{k+1|k}$ is the prediction of the model state at step $k+1$ given all the information available at step k, and

$$
y_k = \begin{bmatrix} y_k^1 \\ y_k^2 \end{bmatrix}, C = \begin{bmatrix} C^1 \\ C^2 \end{bmatrix}, V = \begin{bmatrix} V^1 & \emptyset \\ \emptyset & V^2 \end{bmatrix}
$$

are used for the filter evolution. The matrix $\emptyset$ is a null matrix of appropriate size.

Using a filter based on both the output equations, the computational cost and the memory requirements could be more expensive than using two filters, one for each output equation, y_k^1 or y_k^2.

Following this idea, assume to have a Kalman filter KF^1 based only on y_k^1, C^1, V^1 and a Kalman filter KF^2 based on y_k^2, C^2, V^2. Let $\hat{x}_{k|k-1}^1$ and $\hat{x}_{k|k-1}^2$ be the state predictions provided by KF^1 and KF^2 respectively.
Since two filters are available and each of them provides the prediction of the same state variables, a possible way to improve the prediction performance without affecting computational costs is to suitably combine the two predictions at each step. More precisely, consider the following convex combination

$$
\tilde{x}_{k|k-1} = M_\alpha \hat{x}_{k|k-1}^1 + (I - M_\alpha)\hat{x}_{k|k-1}^2 \tag{2.18}
$$

where $M_\alpha = diag([\alpha_1 \ \alpha_2 \ \dots \ \alpha_n])$ and $\alpha_i \in [0,1], i = 1, \dots, n$.

Using at each step the above combination in the recursive equation of the filter the following algorithm can be stated:

Mixed Kalman Filter (MKF)

$$
\begin{aligned}
\hat{x}_{k+1|k}^1 &= A\tilde{x}_{k|k-1} + Bu_k + L_k^1(y_k^1 - C^1\tilde{x}_{k|k-1}) \\
\hat{x}_{k+1|k}^2 &= A\tilde{x}_{k|k-1} + Bu_k + L_k^2(y_k^2 - C^2\tilde{x}_{k|k-1}) \\
\tilde{x}_{k+1|k} &= M_\alpha \hat{x}_{k+1|k}^1 + (I - M_\alpha)\hat{x}_{k+1|k}^2 \\
\tilde{P}_{k+1|k} &= \Phi(\tilde{P}_{k|k-1})
\end{aligned}
$$

where

- $\tilde{x}_{k+1|k}$ is the MKF predicted state, the initial condition of which is $\tilde{x}_{0|-1} = \hat{x}_{0|-1}$;
- $\tilde{P}_{k+1|k}$ is the prediction error covariance matrix related to $\tilde{x}_{k+1|k} - x_{k+1}$. The initial condition of this covariance matrix is $\tilde{P}_{0|-1} = P_{0|-1}$;
- L_k^1 and L_k^2 are two gains the values of which have to be properly chosen to minimize the prediction error covariance matrix $\tilde{P}_{k+1|k}$;
- the function $\Phi(\cdot)$ describes the evolution of the covariance matrix $\tilde{P}_{k+1|k}$.

Figure 2.7 shows the overall prediction scheme in the proposed configuration.

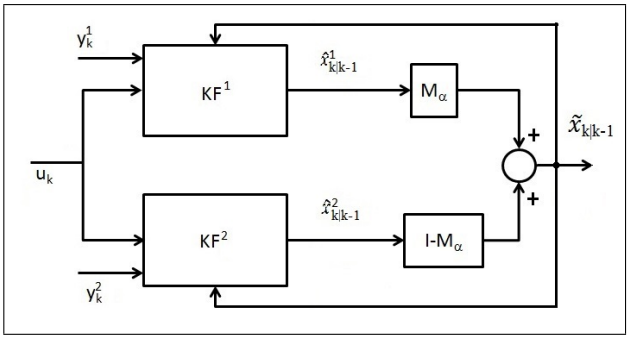


Fig. 2.7: Convex combination based prediction scheme

The more the value of M_α tends to the identity matrix, the more $\tilde{x}_{k|k-1}$ tends to $\hat{x}^1_{k|k-1}$. If M_α tends to the null matrix then $\tilde{x}_{k|k-1}$ tends to $\hat{x}^2_{k|k-1}$. In the next Subsections the function $\Phi(\cdot)$ and the optimal values for the gains L_k^1 and L_k^2 will be found.

Error covariance matrix evolution

Let $e_{k+1|k} = \tilde{x}_{k+1|k} - x_{k+1}$ be the prediction error and let $\tilde{M}_\alpha = (I - M_\alpha)$. Using the MKF equations along with (2.17) the prediction error evolution is:

$$e_{k+1|k} = \tilde{x}_{k+1|k} - x_{k+1} = M_\alpha \hat{x}^1_{k+1|k} + \tilde{M}_\alpha \hat{x}^2_{k+1|k} - x_{k+1} =$$

$$= M_\alpha[(A - L_k^1 C^1)\tilde{x}_{k|k-1} + Bu_k + L_k^1 y_k^1] + \tilde{M}_\alpha[(A - L_k^2 C^2)\tilde{x}_{k|k-1} + Bu_k + L_k^2 y_k^2] - (Ax_k + Bu_k + w_k) =$$

$$= (M_\alpha(A - L_k^1 C^1) + \tilde{M}_\alpha(A - L_k^2 C^2))(\tilde{x}_{k|k-1} - x_k) + M_\alpha L_k^1 v_k^1 + \tilde{M}_\alpha L_k^2 v_k^2 - w_k =$$

$$= (M_\alpha(A - L_k^1 C^1) + \tilde{M}_\alpha(A - L_k^2 C^2))e_{k|k-1} + M_\alpha L_k^1 v_k^1 + \tilde{M}_\alpha L_k^2 v_k^2 - w_k$$

and since $M_\alpha + \tilde{M}_\alpha = I$, the prediction error is

$$e_{k+1|k} = (A - (M_\alpha L_k^1 C^1 + \tilde{M}_\alpha L_k^2 C^2))e_{k|k-1} + M_\alpha L_k^1 v_k^1 + \tilde{M}_\alpha L_k^2 v_k^2 - w_k \tag{2.19}$$

At this point the function $\Phi(\cdot)$ can be found evaluating the $e_{k+1|k}$ covariance matrix. Let $\Delta = (A - (M_\alpha L_k^1 C^1 + \tilde{M}_\alpha L_k^2 C^2))$ and let $\mathbb{E}[\cdot]$ be the expected value function. Assuming that there is no correlation between the measurements noises v_k^1, v_k^2, the process noise w_k and the prediction error $e_{k|k-1}$, the prediction error covariance matrix evolution is

$$\tilde{P}_{k+1|k} = \mathbb{E}[(e_{k+1|k} - \mathbb{E}[e_{k+1|k}])(e_{k+1|k} - \mathbb{E}[e_{k+1|k}])^T] = \mathbb{E}[e_{k+1|k} e_{k+1|k}^T] =$$

$$= \mathbb{E}[(\Delta e_{k|k-1} + M_\alpha L_k^1 v_k^1 + \tilde{M}_\alpha L_k^2 v_k^2 - w_k)(\Delta e_{k|k-1} + M_\alpha L_k^1 v_k^1 + \tilde{M}_\alpha L_k^2 v_k^2 - w_k)^T] =$$

$$= \mathbb{E}[\Delta e_{k|k-1} e_{k|k-1}^T \Delta^T + M_\alpha L_k^1 v_k^1 v_k^{1^T} L_k^{1^T} M_\alpha^T + \tilde{M}_\alpha L_k^2 v_k^2 v_k^{2^T} L_k^{2^T} \tilde{M}_\alpha^T + w_k w_k^T]$$

and finally

$$\tilde{P}_{k+1|k} = \Phi(\tilde{P}_{k|k-1}) = \Delta \tilde{P}_{k|k-1} \Delta^T + M_\alpha L_k^1 V^1 L_k^{1^T} M_\alpha^T + \tilde{M}_\alpha L_k^2 V^2 L_k^{2^T} \tilde{M}_\alpha^T + W \tag{2.20}$$

Please note that, since $\tilde{P}_{k|k-1}$ is a positive semi-definite matrix ($\tilde{P}_{k|k-1} \geq 0$), then $\tilde{P}_{k+1|k}$ will be a positive semi-definite matrix ($\tilde{P}_{k+1|k} \geq 0$) too.

Optimal filter gains

Once the function $\Phi(\cdot)$ has been found, the optimal values for the gains L_k^1 and L_k^2 are computed by minimizing the covariance matrix $\tilde{P}_{k+1|k}$. The $trace\{\tilde{P}_{k+1|k}\}$ has been chosen as the minimization index and the following optimization problem has been stated:

$$\begin{cases} (L_k^1, L_k^2) = \arg \min\limits_{L_k^1, L_k^2} trace\{\tilde{P}_{k+1|k}\} \\ \\ \text{subject to} \\ \tilde{P}_{k+1|k} = \Phi(\tilde{P}_{k|k-1}) \end{cases} \tag{2.21}$$

To solve the above optimization problem, let $\tilde{L}_k^1 = M_\alpha L_k^1$ and $\tilde{L}_k^2 = \tilde{M}_\alpha L_k^2$. Using the trace properties the optimization index can be written as

$$trace\{\tilde{P}_{k+1|k}\} = trace\{\Phi(\tilde{P}_{k|k-1})\} =$$

$$trace\{A\tilde{P}_{k|k-1}A^T + W + \tilde{L}_k^1(C^1 \tilde{P}_{k|k-1} C^{1^T} + V^1)\tilde{L}_k^{1^T} + \tilde{L}_k^2(C^2 \tilde{P}_{k|k-1} C^{2^T} + V^2)\tilde{L}_k^{2^T} + 2\tilde{L}_k^2 C^2 \tilde{P}_{k|k-1} C^{1^T} \tilde{L}_k^{1^T} - 2A\tilde{P}_{k|k-1}(C^{1^T} \tilde{L}_k^{1^T} + C^{2^T} \tilde{L}_k^{2^T})\}$$

At this point the derivatives of the above index w.r.t $\tilde{L}_k^1$ and $\tilde{L}_k^2$ are

$$\frac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^1} = 2\tilde{L}_k^1(C^1\tilde{P}_{k|k-1}{C^1}^T + V^1) - 2A\tilde{P}_{k|k-1}{C^1}^T + 2\tilde{L}_k^2 C^2\tilde{P}_{k|k-1}{C^1}^T \tag{2.22}$$

and

$$\frac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^2} = 2\tilde{L}_k^2(C^2\tilde{P}_{k|k-1}{C^2}^T + V^2) - 2A\tilde{P}_{k|k-1}{C^2}^T + 2\tilde{L}_k^1 C^1\tilde{P}_{k|k-1}{C^2}^T \tag{2.23}$$

To find the minimum values of the optimization index, the following equations are solved

$$\begin{cases} \dfrac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^1} = \emptyset \\ \\ \dfrac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^2} = \emptyset \end{cases}$$

yielding to

$$L_k^2 = f_{L^2}(\tilde{P}_{k|k-1}) = \tilde{M}_\alpha^{-1}(A\tilde{P}_{k|k-1}{C^2}^T - A\tilde{P}_{k|k-1}{C^1}^T(C^1\tilde{P}_{k|k-1}{C^1}^T + V^1)^{-1}C^1\tilde{P}_{k|k-1}{C^2}^T)\Gamma^{-1}$$

$$L_k^1 = f_{L^1}(\tilde{P}_{k|k-1}) = M_\alpha^{-1}(A\tilde{P}_{k|k-1}{C^1}^T - \tilde{M}_\alpha C^2\tilde{P}_{k|k-1}{C^1}^T)(C^1\tilde{P}_{k|k-1}{C^1}^T + V^1)^{-1}$$

where

$$\Gamma = (C^2\tilde{P}_{k|k-1}{C^2}^T + V^2 - C^2\tilde{P}_{k|k-1}{C^1}^T(C^1\tilde{P}_{k|k-1}{C^1}^T + V^1)^{-1}C^1\tilde{P}_{k|k-1}{C^2}^T) \tag{2.24}$$

To ensure that the equations (2.24) are related to a minimum point of the optimization index $trace\{\tilde{P}_{k+1|k}\}$, the Hessian matrix of this index has to be studied. The Hessian matrix is

$$H = \begin{bmatrix} C^1\tilde{P}_{k|k-1}{C^1}^T + V^1 & C^1\tilde{P}_{k|k-1}{C^2}^T \\ C^2\tilde{P}_{k|k-1}{C^1}^T & C^2\tilde{P}_{k|k-1}{C^2}^T + V^2 \end{bmatrix} \tag{2.25}$$

Assuming $q = p = 1$, that is y_k^1 has the same size of y_k^2 and they are scalar values, the determinant of the Hessian matrix is

$$D = \det\{H\} = (C^1\tilde{P}_{k|k-1}{C^1}^T + V^1)(C^2\tilde{P}_{k|k-1}{C^2}^T + V^2) - C^1\tilde{P}_{k|k-1}{C^2}^T C^2\tilde{P}_{k|k-1}{C^1}^T =$$

$$C^1\tilde{P}_{k|k-1}{C^1}^T C^2\tilde{P}_{k|k-1}{C^2}^T + V^1C^2\tilde{P}_{k|k-1}{C^2}^T + C^1\tilde{P}_{k|k-1}{C^1}^T V^2 + V^1V^2 - C^1\tilde{P}_{k|k-1}{C^2}^T C^2\tilde{P}_{k|k-1}{C^1}^T.$$

Since $\tilde{P}_{k|k-1}$ is a symmetric positive semi-definite matrix, it is possible to deduce that

$$C^1\tilde{P}_{k|k-1}{C^2}^T C^2\tilde{P}_{k|k-1}C^1 = C^1\tilde{P}_{k|k-1}{C^2}^T C^1\tilde{P}_{k|k-1}{C^2}^T = C^1\tilde{P}_{k|k-1}{C^1}^T C^2\tilde{P}_{k|k-1}{C^2}^T$$

and thus

$$D = \det\{H\} = V^1 C^2 \tilde{P}_{k|k-1} C^{2^T} + C^1 \tilde{P}_{k|k-1} C^{1^T} V^2 + V^1 V^2 \tag{2.26}$$

Since V^1 and V^2 are positive scalar values and $\tilde{P}_{k|k-1} \geq 0$, the Hessian matrix determinant is $D > 0$ and the equations (2.24) define a minimum point for the optimization problem (2.21).

Remark 1 The obtained result about the gains optimality is valid only if the system (2.17) has two scalar outputs. Otherwise, there is no assurance about the optimality of the gains defined by equations (2.24). Further studies are in progress to validate the obtained results also when the system has not only two scalar outputs. Moreover, as it will be shown in Section 6.6.5, numerical and experimental tests have been performed using a system with two non scalar outputs and very good results have been obtained.

Remark 2 The equations (2.20) and (2.24) become the standard Kalman filter equations if $M_\alpha = I$ or $M_\alpha = \emptyset$.

As the obtained equations show, the optimal gains values and the error covariance matrix evolution are influenced by M_α, therefore it is very important to properly choose the values of $\alpha_i, i = 1, \ldots, n$ to obtain the best state prediction results. Choosing these values depends on the sensors related to each output equation. For example, if the sensors related to y_k^1 give less information about the j-th state variable than the sensors related to y_k^2, α_j has to be $\alpha_j \to 0$.

2.7.2 Mixed Kalman Filter for mobile robots localization

To adapt the Mixed Kalman filter framework to the mobile robot localization problem, the two output equations have to be defined. The first output equation is related to the robot on board sensors and it has been defined in the previous Sections (see (2.13)):

$$r_k \approx h(x_k, (\bar{s}_k, \bar{c}_k)) + v_k^1. \tag{2.27}$$

Please note that in the following, the on board sensors output will be denoted as r_k and v_k^1 will denote the on board sensors measurement noise; the symbol y_k will be instead used to denote the entire system output, including both the on board sensors outputs and the out of board sensors outputs.

The second output equation is related to a set of q external distance sensors placed in known locations in the environment. These sensors measure the distances between each out of board sensor F_i, $i = 1, \ldots, q$, located in (F_1^i, F_2^i), and the robot center $P = (x_1^R, x_2^R)$.
The second output equation is then

$$d_i = h^2((x_1^R, x_2^R), (F_1^i, F_2^i)) = \sqrt{(x_1^R - F_1^i)^2 + (x_2^R - F_2^i)^2}$$
$$i = 1, \ldots, q$$

and it will be written in compact form as

$$d_k = h^2(x_k, \bar{F}) + v_k^2 \tag{2.28}$$

where $\bar{F}$ contains the parameters $F_i, i = 1, \ldots, q$ and the vector v_k^2 collects the out of board sensors noises, also assumed Gaussian, zero-mean and uncorrelated with $w_k = [w_k^x, w_k^y, w_k^\theta]^T$ and v_k^1.

Using the on board sensors measurements, a NEKF algorithm (see Section 2.6) based on the Kalman predictor structure (see Section 2.3.1) can be stated. For what concerns the out of board sensors, an Extended Kalman filter algorithm, based on the output equation (2.28) and on the Kalman predictor algorithm, can be built. The resulting filter will be denoted as Out of board sensors based Extended Kalman Filter (OEKF).

Thanks to the use of out of board sensors, the OEKF performance are quite influenced by the initial condition error. However, due to the absence of a real angular information, the estimate on the robot heading can be very noisy and inaccurate. Otherwise the NEKF algorithm, as shown in [13], is able to provide a good heading estimation but the state estimation is very affected by initial condition errors.

In conclusion, using (2.27) as the first output equation, (2.28) as the second output equation, the NEKF as the first filter, the OEKF as the second one, the resulting Mixed Extended Kalman filter (MEKF) algorithm is

---

Mixed Extended Kalman Filter

---

first filter update:NEKF

$$L_k^1 = f_{L^1}(\tilde{P}_{k|k-1})$$
$$(\hat{\tilde{s}}_k, \hat{\tilde{c}}_k) = \mathrm{NBA}(\tilde{x}_{k|k-1}, r_k)$$
$$\hat{x}_{k+1|k}^1 = \phi(\tilde{x}_{k|k-1}, u_k) + L_k^1(r_k - h(\tilde{x}_{k|k-1}, (\hat{\tilde{s}}_k, \hat{\tilde{c}}_k)))$$

second filter update:OEKF

$$L_k^2 = f_{L^2}(\tilde{P}_{k|k-1})$$
$$\hat{x}_{k+1|k}^2 = \phi(\tilde{x}_{k|k-1}, u_k) + L_k^2(d_k - h^2(\tilde{x}_{k|k-1}, \overline{F}))$$

MEKF step

$$\tilde{x}_{k+1|k} = M_\alpha \hat{x}_{k+1|k}^1 + (I - M_\alpha)\hat{x}_{k+1|k}^2$$
$$\tilde{P}_{k+1|k} = \Phi(\tilde{P}_{k|k-1})$$

---

where

- L_k^1 and L_k^2 are computed using C_k^1, C_k^2, V^1, V^2, respectively and

$$A_k = \left.\frac{\partial \phi(x,u_k)}{\partial x}\right|_{x=\tilde{x}_{k|k-1}}, C_k^1 = \left.\frac{\partial \mathbf{h(x)}}{\partial \mathbf{x}}\right|_{x=\tilde{x}_{k|k-1}}, C_k^2 = \left.\frac{\partial \mathbf{h^2(x)}}{\partial \mathbf{x}}\right|_{x=\tilde{x}_{k|k-1}}; \tag{2.29}$$

- V^1 and V^2 are the covariance matrices related to the on board and to the out of board sensors' measurements noises;
- $\phi(\cdot,\cdot)$ is the robot model function;
- the MEKF initial conditions are $\tilde{x}_{0|-1}$ and $\tilde{P}_{0|-1}$ and they are assumed to be known;

In the MEKF algorithm, both NEKF and OEKF are used in parallel but, at each step, the filters recursive equations are updated through a convex combination of the filters' predictions. Please note that the MEKF algorithm does not make use of a single EKF based on the entire output $y_k = [r_k \; d_k]^T$ but it uses both filters in a parallel way. Thanks to the MEKF algorithm, the memory requirements and the computational cost of the filter are quite the same of using only the NEKF or only the OEKF while the estimation performance should be enhanced w.r.t these two filters, especially if $\alpha_1, \alpha_2, \ldots, \alpha_n$ are correctly chosen.

2.8 Chapter Conclusions

In this chapter the proposed solutions to the mobile robots localization problem have been shown. The discussed topics have shown how it is not possible to find a technique useful in all the situations and in all the contexts. Depending on the available sensors and on the available information on the robot surrounding environment, different solutions can be adopted.

Moreover each solution has drawbacks and advantages and, in particular, if no assumption is done on the robot environment, at least the estimation bias problem is unavoidable.

In Chapter 5, other localization techniques will be shown. In contrast to the algorithms proposed in this Chapter, the solutions described in Chapter 5 will be based on the use of cameras and Inertial Measurement Units (IMUs) with no robot model information required.

3

Simultaneous Localization and Mapping (SLAM) problem for mobile robots

3.1 Introduction

In the second chapter of this dissertation, the mobile robots localization problem has been described and faced. In particular in Section 2.6 the localization problem in a totally unknown environment has been solved. The above situation is very common in a big amount of robotics applications. Just to give a few examples, in mobile robots rescue missions or in planetary exploration missions, the robots usually do not have information on the environment where they move. In these contexts, and in many other ones, the **environment mapping** problem has to be solved.

Mapping is the problem of integrating the information gathered with the robot's sensors into a given environment representation. This problem can be described by the question **"What does the world look like?"**. Key points in mapping are the environment representation and the sensors data interpretation.

When a robot moves in an unknown environment, it could be very interesting to **simultaneously** localize the robot and obtain an estimation of the robot surrounding environment. The **S**imultaneous **L**ocalization **A**nd **M**apping (**SLAM**) problem deals with these two main tasks: (1) **robot localization** and (2) **environment mapping**.

The genesis of the SLAM problem occurred at the 1986 IEEE Robotics and Automation Conference held in San Francisco where a number of researchers had been looking at applying estimation-theoretic methods to mapping and localization problems. Among these researchers, Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte introduced for the first time the Simultaneous Localization and Mapping (SLAM) problem.

In mobile robotics, this problem is considered as *chicken and egg* problem. To know where the robot is, a good environment mapping is needed but to have a reliable environment mapping, the robot's position and orientation should be estimated as best as possible.

3.2 SLAM state of art

The literature shows various solutions to the SLAM problem [6, 7, 8, 32, 31, 33] using various sensors types, e.g. wheel encoders, laser range sensors, sonar range sensors and cameras. Each SLAM algorithm is very influenced by the used sensors types and the algorithm performance can be very high thanks to appropriate sensors but they can drastically degenerate using different ones. Probably the most used sensors types in SLAM applications are laser sensors and cameras. Regarding the SLAM problem using laser sensors, in [6] a laser sensor based SLAM algorithm for mobile robots in outdoor environments has been developed solving the problem through an Extended Kalman filter (EKF) based on an unknown robot initial position. In [40] the authors propose a SLAM algorithm which obtains one estimate of the robot pose from a dead-reckoning scheme and another one from a laser scan matching algorithm. The fusion of the two estimates is then done through a covariance intersection filter. In [7] starting from laser sensors measurements, a weighted least square fitting is used to extract certain two-dimensional environment in order to solve the SLAM problem.
About SLAM using cameras, in [41] the proposed SLAM algorithm runs an EKF based on a monocular camera. The algorithm uses an object recognition thread which detects objects by looking for geometric compatibility and correspondences in the environment. When an object is recognized, it is inserted in the SLAM map, being its position measured and hence refined by the SLAM algorithm in subsequent camera frames. In [42] the authors present a robust simultaneous localization and mapping algorithm based on a single camera catadioptric stereo system composed of vertically aligned two hyperboloidal mirrors and a CCD camera. The single camera catadioptric stereo system gives the 3D landmarks locations. In [8] a novel method of mobile robot simultaneous localization and mapping using the Rao-Blackwellised particle filter (RBPF) for monocular vision-based autonomous robot in unknown indoor environment is proposed. The particle filter is combined with Unscented Kalman filter (UKF) and the landmark position estimation is implemented through the unscented transform.

Laser sensors and cameras probably represent the best way to face the SLAM problem but they both have some drawbacks. Laser sensors can be very expensive while cameras provide a large amount of information and it may be difficult and computationally onerous to extract this information starting from the camera image. An alternative way to solve the SLAM problem consists in the use of sonar sensors. In spite of their lower accuracy w.r.t. laser sensors, the sonar sensors are often used because they are less expensive than laser scanners and range cameras, their use is computationally cheap and they work well also in dark or transparent environments, where cameras usually fail.

The literature shows many SLAM solutions based on the use of ultrasonic sensors. In [31] Tard´os et al. describe a technique to face the SLAM problem using standard sonar sensors and the Hough transform [34] to detect corners,

edges and walls into the environment starting from acquired sonar data. In [36] a range sonar array based SLAM for autonomous underwater robots is presented, the authors propose an EKF based solution and show its effectiveness through experimental studies on the P-SURO AUV. In [35] a pose-based algorithm to solve the SLAM problem for an autonomous underwater vehicle is proposed. A probabilistic scan matching technique using range scans gathered from a mechanical scanning imaging sonar is used along with two Extended Kalman filters, one for the estimation of the local path traveled by the robot and the second one to estimate and keep the registered SLAM landmarks. In [33] the authors use sonar measurements along with a particle filter to solve the SLAM problem for mobile robots in non-static environments.

Very often the SLAM algorithms assume to have at least some *a priori* information about the environment (for example in [36]) or they assume to model the environment in a very approximated way: in [37] the authors assume the robot placed in an environment modeled as a set of orthogonal-parallel lines.

As previously remarked, in a computational point of view, it is mandatory for a SLAM algorithm to be as cheap as possible in order to satisfy time constraints so that the algorithm can be used to help robot control systems (see Figure I.7). since information provided by SLAM algorithms is typically used to compute the output of a control law designed, for example, to make the robot follows a given trajectory or completes a given task. Developing a computationally cheap SLAM algorithm and obtaining an accurate environment mapping are two conflicting goals. The more the map provided by SLAM is accurate, the higher will be the SLAM algorithm effort to obtain and manipulate this map.

In the present thesis, three new distance sensors based SLAM solutions will be proposed. The first one is computationally cheap but the resulting mapping performance can be very poor; the second one has been developed looking at a very accurate environment mapping, but it requires a high computational cost. Finally, the third one has been developed to be computationally less onerous than the second one but providing mapping results better than the ones obtained using the firs technique. In other words, looking at the trade off between mapping performance and computational costs, the first two techniques are on opposite sides while the third one is in the middle of them, trying to achieve a good compromise between costs and performance.

In the three techniques, the robot is assumed to be equipped with n_S distance sensors and the sensors will be denoted as $S_i, i = 1, \ldots, n_S$.

3.3 SLAM problem description

The goal of the SLAM process is to use the environment to update the robot position and to use the robot position to better detect the robot surrounding environment. Since the robot model is often erroneous, the only information

provided by this model is not sufficient to solve the SLAM problem and the main idea is to use information provided by sensors to properly correct the robot pose estimation simultaneously detecting the surrounding environment boundaries. This is accomplished by extracting features from the environment and **re-observing** these features when the robot moves around. Very often a Kalman filter is the heart of the SLAM process. The filter is responsible for updating where the robot thinks it is, based on the environment extracted features, commonly called **landmarks**. The Kalman filter keeps track of an estimate of the uncertainty in the robots position and also of the uncertainty in the landmarks the robot has detected in the environment.

The SLAM process consists of a 5 main steps, an outline of this process is depicted in Figure 3.1.

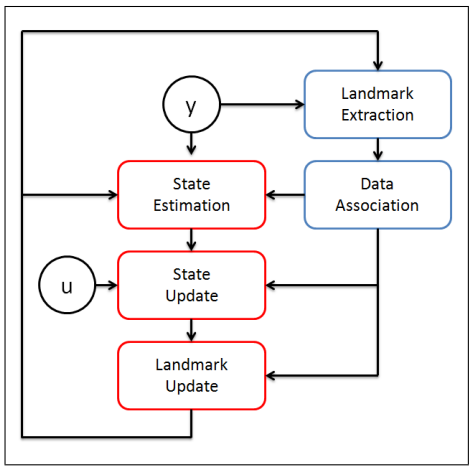


Fig. 3.1: SLAM main steps

The acquired measurements y_k and the actual robot pose prediction $\hat{x}_{k|k-1}$ are used to perform the data association and landmark extraction processes yielding to the actual environment mapping. Starting from this map and from the model inputs u_k, the state estimation, state update and landmark update processes provide the robot pose estimation $\hat{x}_{k|k}$ and prediction $\hat{x}_{k+1|k}$ and properly update the environment map.

In the following Sections, the model (1.4) will be used as the robot model and the robot will be assumed to be equipped with a set of n_S on board distance sensors.

Just to give an example of how a SLAM algorithm works, indicating the robot as a triangle, the landmarks as stars and sensor measurements as lightnings, the main SLAM steps, at each algorithm iteration, are:

1. The robot initially measures, using its sensors, the location of the landmarks, see Figure 3.2.

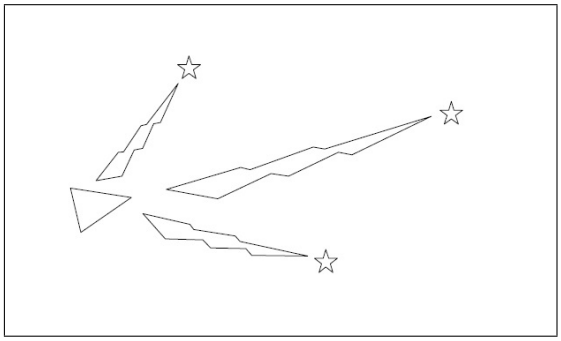


Fig. 3.2: SLAM first step

2. The robot moves depending on its wheels velocities inputs and the model provides a robot pose estimation (see Figure 3.3).

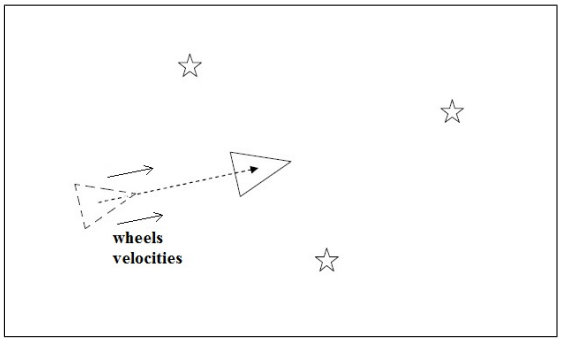


Fig. 3.3: SLAM second step

3. The robot once again measures the location of the landmarks using its distance sensors. Depending on how the measured landmarks match with the estimated landmarks positions (computed using the robot pose estimation), the algorithm modifies the robot estimated pose (see Figure 3.4).

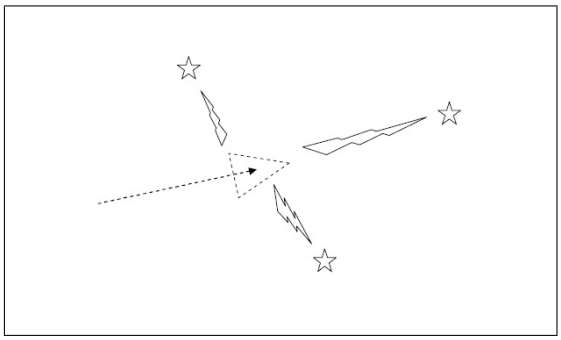


Fig. 3.4: SLAM third step

4. As the robot believes more its sensors than its model, it now uses the information gained about where the landmarks actually are to determine where it is (the location the robot originally thought it was at, is illustrated by the dashed triangle in Figure 3.5).

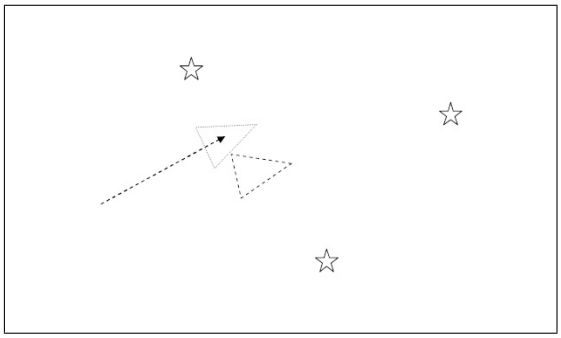


Fig. 3.5: SLAM fourth step

5. At this point the SLAM process restarts from step 1.

In the following Subsections, the main topics about the SLAM algorithms will be discussed and described.

3.3.1 Landmarks: features and properties

Landmarks are environment parts which can be easily observed, re-observed and distinguished from the other environment portions. The landmarks are used by the robot to find out where it is (to localize itself). One way to imagine how this process works for the robot is to picture a human blindfolded in a room. If he moves around blindfolded, the human may reach out and touch

objects or hug walls so that he does not get lost. Characteristic things such as that felt by touching a doorframe may help the human in establishing an estimate of where he is. Distance based sensors represent the robots feeling of touch.

Landmarks should be **re-observable** by allowing them for example to be viewed (detected) from different positions. Landmarks should be **unique** so that they can be easily identified from one time step to another time step without mixing them up.

The landmark choice is a crucial point in SLAM algorithms. If the chosen landmarks are such that the number of landmarks in the environment is very low, the robot may have to spend extended time without enough visible landmarks and, as a consequence, the robot may then get lost. On the contrary, if a too high number of landmarks can be detected in the environment, the SLAM algorithm computational cost and the algorithm memory requirements become too onerous.
Moreover, the landmarks should be **stationary**. For instance, using a person as a landmark does not represent a good choice. The reason for this criterion is fairly straightforward: if the landmark is not always in the same place how can the robot know, given this landmark, in which place it is?

In summary, the key points about good landmarks are:

- landmarks should be easily re-observable;
- individual landmarks should be distinguishable from each other;
- the landmarks number in the environment has to bee neither too low nor too high;
- landmarks should be stationary.

3.3.2 Landmark Extraction

Once the landmarks have been chosen, the next step is to somehow reliably extract them from the robot sensors measurements. There are multiple ways to perform the landmark extraction process and the way to chose largely depends on what types of landmarks are attempted extracted as well as what sensors are used.

Using distance based sensors, the landmark extraction process can be accomplished, for example, thanks to the RANSAC method [48].

3.3.3 Data Association

The problem of data association is related to obtain matching between currently observed landmarks and previously observed ones.
The following example can be helpful to illustrate what is meant:

> Consider a human in a room and let a chair be a landmark. The human sees a specific chair and then he leaves the room and, at some later point, subsequently returns to the room. If the human sees a chair in

the room and he says that it is the same chair previously seen then he has implicitly associated this chair to the old chair. This process may seem simple but the data association is a difficult task to do well. Imagine the room had two chairs that looked practically identical, one on the left side of the room and the second one on the right side. When the human subsequently returns to the room, he might not be able to distinguish accurately which of the two chairs were which of the chairs he originally saw (as they all look the same). In this context, the best bet is to say that the one on the left must be the one previously seen to the left, and the one to the right must be the one previously seen on the right.

In practice the following problems can arise during the data association process:

- The landmarks could not be re-observed at each time step.
- A landmark might be observed but it could be not possible to ever see it again.
- Currently detected landmarks could be wrongly associated to previously observed ones.

As stated in the landmarks description Section, it should be easy to re-observe landmarks and thus the above first two cases are not acceptable for a *good* landmark. In other words, if the first two situations occur, then not reliable landmarks have been chosen. The last problem is related to a wrong landmark association and it can be devastating as it means the robot will think it is somewhere different from where it actually is. To avoid these problems, a typically used policy is based on a landmarks database definition. The database is set up to store previously seen landmarks and it is usually initially empty. Each SLAM algorithm defines a set of rules to state if a new observed object has to be associated to a landmark or it has to be discarded. For example, a typically used rule states that a detected object cannot be considered as a worthwhile landmark unless it has been seen at least N times.

3.3.4 State estimation, Landmark and state update

The last three SLAM steps are usually performed thanks to an Extended Kalman Filter. More precisely, all the proposed SLAM algorithms in this thesis are based on a properly adapted EKF. As described in Section 2.3.2, the EKF is typically used only to obtain a model state estimation. In other words, the EKF, in its traditional formulation, does not deal with the map update and estimation.
In the SLAM context, the EKF is *modified* in order to simultaneously provide landmark update, robot pose estimation and update.
The main idea is to define an **augmented** state which contains the robot pose along with the landmarks describing data. More precisely, starting from the

system model (2.6), and denoting as x_k the robot pose and as $\mathbb{E}_k = \{l_k^i, i = 1, \ldots, n_k\}$ the estimated environment map at step k, where $l_k^i, i = 1, \ldots, n_k$ are the n_k landmarks currently detected, the following augmented sate is defined:

$$X_k = [x_k^T,\ l_k^1, \ldots, l_k^{n_k}]^T = [x_k^T,\ \mathbb{E}_k^T]^T$$

and a new system (robot and environment) model, based on the augmented state, is obtained using the following state update function

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ l_{k+1}^1 \\ \vdots \\ l_{k+1}^{n_k} \end{bmatrix} = \mathcal{F}\left(\begin{bmatrix} x_k \\ l_k^1 \\ \vdots \\ l_k^{n_k} \end{bmatrix}, u_k \right) + \begin{bmatrix} w_k \\ \emptyset \\ \vdots \\ \emptyset \end{bmatrix} = \begin{bmatrix} \phi(x_k, u_k) + w_k \\ l_k^1 \\ \vdots \\ l_k^{n_k} \end{bmatrix} \tag{3.1}$$

that is the standard state update function $\phi(\cdot, \cdot)$ (see equation (1.4)) for the robot pose evolution and a constant function for the landmarks evolution, stating that the landmarks data are not influenced by state noise nor by the control inputs (landmarks have to be **stationary**).
Depending on the used sensors, also the output equation is properly modified to explicitly show the relationships between the sensors measurements and the detected landmarks. The resulting output equation is

$$y_k = h(x_k, \mathbb{E}_k) + v_k = h(X_k) + v_k \tag{3.2}$$

Defining an Extended Kalman filter based on the new augmented state, the obtained estimation will be related to both the robot pose and the landmarks data and it will be used for the SLAM state estimation, state update and landmark update steps. More precisely, once the augmented model (equations (3.1) and (3.2)) has been defined, also the system and filter matrices are suitably modified to deal with the augmented state X_k.

The resulting Extended Kalman filter algorithm suitably adapted to the augmented state is

SLAM algorithm based on EKF

EKF prediction step

$$\begin{aligned}
&\hat{X}_{k+1|k} = \mathcal{F}(\hat{X}_{k|k}, u_k)\\
&\mathcal{P}_{k+1|k} = \mathcal{A}_k \mathcal{P}_{k|k} \mathcal{A}_k^T + \mathcal{W}
\end{aligned}$$

SLAM step

$$\begin{aligned}
&\mathbb{E}_{k+1} = \mathcal{L}(\mathbb{E}_k, y_{k+1}, \hat{X}_{k+1|k})\\
&(\hat{X}^*_{k+1|k}, \mathcal{P}^*_{k+1|k}) = \mathcal{U}(\hat{X}_{k+1|k}, \mathcal{P}_{k+1|k}, \mathbb{E}_k, \mathbb{E}_{k+1})\\
&\hat{X}_{k+1|k} = \hat{X}^*_{k+1|k}\\
&\mathcal{P}_{k+1|k} = \mathcal{P}^*_{k+1|k}
\end{aligned} \tag{3.3}$$

EKF estimation step

$$\begin{aligned}
&K_{k+1} = \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T (\mathcal{C}_{k+1} \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T + V)^{-1}\\
&\hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_{k+1}(y_{k+1} - h(\hat{X}_{k+1|k}))\\
&\mathcal{P}_{k+1|k+1} = \mathcal{P}_{k+1|k} - K_{k+1} \mathcal{C}_{k+1} \mathcal{P}_{k+1|k}
\end{aligned}$$

Here $\mathcal{A}_k$, $\mathcal{C}_k$, $\mathcal{W}$ and $\mathcal{P}_k$ are the dynamic matrix, the output matrix, the process noise covariance matrix and the estimation error covariance matrix respectively. All these matrices are suitably adapted to the augmented state. The algorithm initial conditions are $\mathbb{E}_0 = \{\emptyset\}$, $\hat{X}_{0|0} = [\hat{x}_{0|0}]$.

More precisely, the dynamic and output matrices are given by

$$\mathcal{A}_k = \left.\frac{\partial \mathcal{F}(X_k, u_k)}{\partial X_k}\right|_{X_k = \hat{X}_{k|k}}, \quad \mathcal{C}_k = \left.\frac{\partial h(X_k)}{\partial X_k}\right|_{X_k = \hat{X}_{k|k-1}}; \tag{3.4}$$

the process noise covariance matrix $\mathcal{W}$ is

$$\mathcal{W} = \begin{bmatrix} W & \emptyset \\ \emptyset & \emptyset \end{bmatrix}.$$

This matrix contains the robot pose process noise covariance matrix W and a null matrix, $\emptyset$, of appropriate size, related to the landmarks because they are assumed to be not influenced by process noise.

Finally, indicating with $\mathcal{P}_k$ the covariance matrix related to the augmented estimation error $X_k - \hat{X}_k$, also this matrix has to be adapted to the augmented state. Let $\hat{x}_k$ be the state x_k estimation and let $\hat{l}_k^i$ be the landmark l_k^i estimation, assuming $n = n_k$ for the sake of simplify notation, the estimation error covariance matrix related to the estimation error $X_k - \hat{X}_k$ results in

$$\mathcal{P}_k = \begin{bmatrix} P_k & P(\hat{x}_k, \hat{l}_k^1) & \dots & P(\hat{x}_k, \hat{l}_k^n) \\ P(\hat{l}_k^1, \hat{x}_k) & P(\hat{l}_k^1, \hat{l}_k^1) & \dots & P(\hat{l}_k^1, \hat{l}_k^n) \\ \vdots & & \ddots & \\ P(\hat{l}_k^n, \hat{x}_k) & P(\hat{l}_k^n, \hat{l}_k^1) & \dots & P(\hat{l}_k^n, \hat{l}_k^n) \end{bmatrix}$$

where $P(a, b)$ is the covariance matrix between the variables a and b. As initial condition, the initial estimation error covariance matrix is $\mathcal{P}_{0|0} = P_{0|0}$.
In the equations (3.3), the $\mathcal{L}$ function contains all the operations required by the landmark extraction and data association processes while the $\mathcal{U}$ function, called update function, aims to suitably adapt the augmented state and the filter matrices to the results provided by the $\mathcal{L}$-function. The dimensions of the augmented state X_k and of the matrices $\mathcal{A}_k, \mathcal{C}_k, \mathcal{P}_k, \mathcal{W}$ are time varying since the number, $n = n_k$, of landmarks used to map the environment can change during the SLAM process due to the landmark extraction and data association processes.

For each change in the modeling landmarks set $\mathbb{E}_k$, there is a change also in the estimation error covariance matrix and in X_k. More precisely, if a new landmark l_k^r comes out from the landmark extraction process, then the augmented state becomes $X_k = [X_k^T,\ l_k^r]^T$ and a new row and a new column will be added to the estimation error covariance matrix:

$$\tilde{\mathcal{P}}_k = \begin{bmatrix} & & P(x_k, \hat{l}_k^r) \\ & \mathcal{P}_k & P(\hat{l}_k^1, \hat{l}_k^r) \\ & & \vdots \\ P(\hat{x}_k, \hat{l}_k^r) & P(\hat{l}_k^1, \hat{l}_{k)}^r & \dots & P(\hat{l}_k^r, \hat{l}_k^r) \end{bmatrix}; \mathcal{P}_k = \tilde{\mathcal{P}}_k \tag{3.5}$$

In the same way, if one of the modeling landmarks is deleted by the landmark extraction process, the related entries in the augmented state and in the matrix $\mathcal{P}_k$ will be deleted.
Following the above update rules, the update function

$$(X_k^*, \mathcal{P}_k^*) = \mathcal{U}(X_k, \mathcal{P}_k, \mathbb{E}_{k-1}, \mathbb{E}_k)$$

properly modifies the augmented state and the estimation error covariance matrix according to the variations in the environment mapping from $\mathbb{E}_{k-1}$ to $\mathbb{E}_k$. The function outputs are then used as the new augmented state, $X_k = X_k^*$, and as the new estimation error covariance matrix, $\mathcal{P}_k = \mathcal{P}_k^*$. In the following Sections, three new SLAM techniques will be proposed defining the related landmarks and all the operations required by the $\mathcal{L}$-function (landmark extraction and data association function) and the $\mathcal{U}$-function (update function).

3.3.5 SLAM tasks

In summary, the 5 SLAM steps can be seen as parts of three main SLAM tasks:

1. Update the current state estimation using the model data.
2. Take sensors measurements and use these measurements to extract new landmarks or to associate the obtained data to previously detected landmarks; this task will be denoted as landmark extraction and data association task.
3. Update the current landmarks and robot pose estimation using an Extended Kalman filter; this task will be denoted as the state estimation, landmarks and state update task.

The first task is very easy and it can be performed using the robot model and the measured robot control inputs.

During the second task the re-observed landmarks and the new detected landmarks are considered. If new landmarks are detected, their describing data are added to the augmented state and to the filter and model matrices (thanks to the $\mathcal{U}$-function. For what regards the re-observed landmarks, using the robot estimated positions and the acquired sensors measurements, it is possible to estimate the landmarks expected positions. The difference between the re-observed landmarks detected positions and the expected landmarks positions represents the **landmarks innovation**. The total innovation is formed by the standard robot estimation innovation and by the landmarks innovation. This total innovation is basically the difference between the estimated robot position/landmarks positions and the actual robot position/landmarks positions, based on what the robot is able to measure. This difference will be used in the EKF to perform the third SLAM task.

3.4 Segment based SLAM (SbSLAM)

The first proposed SLAM technique has been developed thinking at the simplest way to approximate whatever curve: a set of segments. The environment will be modeled as **a set of segments such that each of them intersects at least at one point on the environment boundaries** (see Figure 3.6 for the case $n_S = 5$). In other words, the environment will be approximated by a set of properly chosen segments which works like an "envelope" for the environment boundaries.
Each sensor S_i provides, at each step k, the distance from the robot center, denoted by $P = (x^R_{1,k}, x^R_{2,k})$, to one point on the environment boundaries, denoted by $\tilde{P}_i = (\tilde{x}_1, \tilde{x}_2)$.
Considering the model used for the environment, the measurement provided by the i-th sensor, S_i, can be obtained using the output equation (2.13) proposed in section 2.5.

3.4.1 Landmark extraction and Data association

Starting from the segment based model for the robot surrounding environment, the proposed Segment based SLAM algorithm uses landmarks, l^i_k,

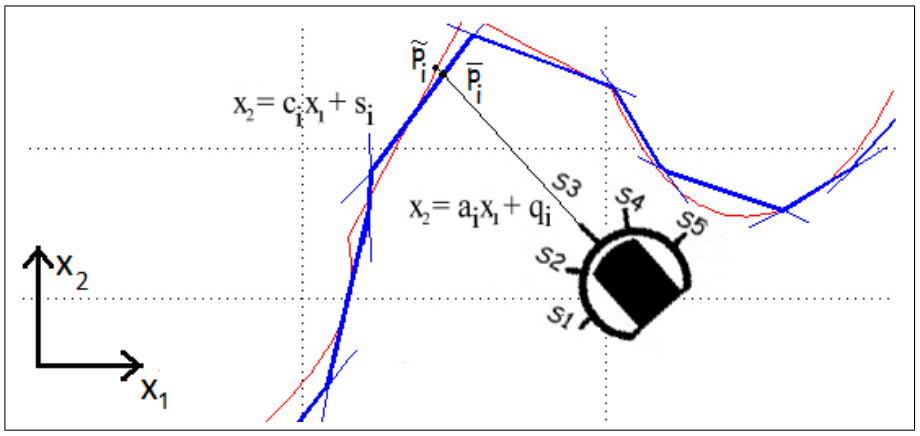


Fig. 3.6: Real environment (red line), segment based environment boundaries approximation (blue line)

formed by the starting and ending points of each modeling segment. The main idea behind the landmark extraction and data association algorithm is to use the currently acquired measurements to update the actual environment mapping by finding new surrounding environment modeling segments or by improving the previously obtained ones. The following notation will be used:

- $\mathbb{E}_k = [p_{k,1}, p_{k,2}, \ldots, p_{k,n_k}]$ is an array of n_k *sorted* points. In the following, an array of points will be considered sorted if each couple $(p_{k,i}, p_{k,i+1})$, of consecutive points, represents the starting and ending points of one of the environment modeling segments.
- $\mathcal{E}_k$ is a set containing all the points in $\mathbb{E}_k$.
- Π_k is the set of currently acquired environment points. Using equations (2.14), the state estimation $\hat{x}_k$ along with the measurements y_k provided by the on board distance sensors, an approximation of the environment points, $\pi_k^i, i = 1, \ldots, n_S$, detected by the sensors, can be obtained.

The goal of the landmark extraction and data association process is to find $\mathbb{E}_k$ starting from the previous environment mapping $\mathbb{E}_{k-1}$, the predicted robot state $\hat{x}_{k|k-1}$ and the acquired measurements y_k. To accomplish this task the main idea is to try to compute a new landmark related to each point in Π_k. The obtained n_S landmarks are then suitably added to $\mathbb{E}_{k-1}$ yielding to $\mathbb{E}_k$. More precisely, at each step k, given the prediction $\hat{x}_{k|k-1}$ and the acquired measurements y_k, the set Π_k is computed. For each point $\pi_k^i \in \Pi_k$ the set of points close, in a neighborhood $\delta > 0$ (δ is one of the algorithm parameters), to π_k^i is obtained using the closeness function defined in Section 2.6.1:

$$\Omega_i = \mathcal{N}(\pi_k^i, \{\mathcal{E}_{k-1} \bigcup \Pi_k\}, \delta).$$

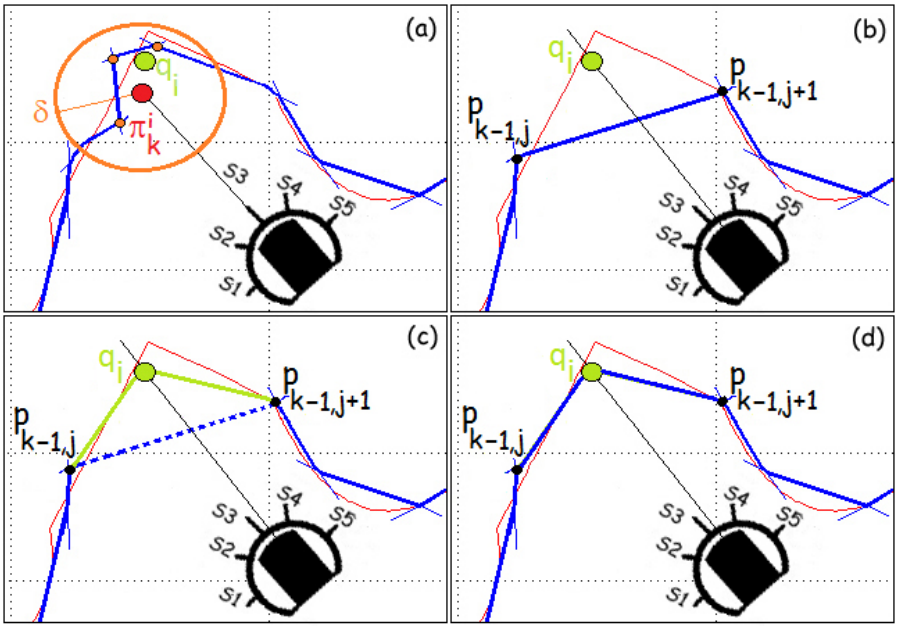


Fig. 3.7: (a): virtual point q_i computation, the orange points and the red point are in Ω_i; (b) segment b_{k-1} computation; (c) and (d): virtual point insertion

For each set Ω_i a virtual point q_i is computed through a weighted mean of the points contained in Ω_i (see Figure 3.7(a)):

$$q_i = g_x(\hat{x}_{k|k-1}, y_k) = \frac{1}{M_i} \sum_{p_j \in \Omega_i} m_j p_j,\ M_i = \sum_{j=1}^{|\Omega_i|} m_j \tag{3.6}$$

where $|\Omega_i|$ is the number of elements contained in Ω_i, $m_i = 1$ if p_i is one of the currently acquired points ($p_i \in \Pi_i$), $m_i = n_i$ if p_i is one of the previously obtained virtual points, computed using n_i points.
Please note that the proposed weighted mean is a function of the robot predicted pose $\hat{x}_{k|k-1}$ and of the currently acquired measurements y_k since each point into Π_k is computed using (2.14) and thus the function $g_x(\hat{x}_{k|k-1}, y_k)$ can be obtained by replacing (2.14) with (3.6).
The proposed weighted mean is such that the more the number of points represented by a previously computed virtual point is big, the bigger this point influence will be on the new virtual point q_i. At the end, the number of points described by the new virtual point q_i is M_i and it will be related to the new virtual point for future algorithm executions. For each set Ω_i there will be a new virtual point q_i and each of these points will be related to a different number of mapped points $M_i, i = , \ldots, n_S$. Once all the new virtual points have been obtained, each set of points Ω_i can be removed from

$\mathbb{E}_{k-1}$. The new array $\mathbb{E}_{k-1}$ can be computed as the sorted array (following the previously described sorting definition) containing the points in

$$\tilde{\mathcal{E}}_{k-1} = \mathcal{E}_{k-1} \setminus (\bigcup_{i=1}^{n_S} \Omega_i), \quad \mathcal{E}_{k-1} = \tilde{\mathcal{E}}_{k-1}.$$

Figure 3.7(a) shows $\mathbb{E}_{k-1}$ before the Ω_i deleting process while Figure 3.7(b) shows the resulting $\mathbb{E}_{k-1}$ after the Ω_i deleting process. As shown in these Figures, after the Ω_i deleting process all the segments related to starting and ending points in Ω_i are removed.
At this point, the new virtual points $q_i, i = 1, \ldots, n_S$ have to be inserted into the vector $\mathbb{E}_{k-1}$. As previously remarked, this vector contains the sorted sequence of starting points and ending points of each of the environment modeling segments. To insert a new virtual point q_i into $\mathbb{E}_{k-1}$ without infringing its ordering, a possible solution consists in three main steps:

1. the line $\overline{r}$ between the robot position and the virtual point is computed;
2. the subvector e_{k-1} of $\mathbb{E}_{k-1}$ containing all the segments in $\mathbb{E}_{k-1}$ intercepted by $\overline{r}$, and in front of the robot, is obtained;
3. the segment b_{k-1} contained in e_{k-1} and closest to the robot along the line $\overline{r}$ is computed and let $p_{k-1,j}, p_{k-1,j+1}$ be the segment starting and ending points respectively (see Figure 3.7(b)).

To ensure the array $\mathbb{E}_{k-1}$ is sorted after the point q_i is added, this point has to be inserted between the starting point and the ending point of b_{k-1}. Therefore when a new virtual point q_i is inserted into $\mathbb{E}_{k-1}$, one of the previously obtained segments is deleted and substituted by two new segments, the first one ends in q_i while the second one starts from q_i. Figures 3.7(c) and 3.7(d) show an example of virtual point insertion into the landmarks array $\mathbb{E}_{k-1}$, in particular Figure 3.7(d) shows the resulting environment mapping after the new point insertion.

Note that after the Ω_i deleting process, the resulting intermediate mapping (blue line in Figure 3.7(b)) is usually worse than the previous one (blue line in Figure 3.7(a)) but adding the obtained virtual point q_i, the final mapping (blue line in Figure 3.7(d)) is better and it contains a lower number of segments than the starting mapping (blue line in Figure 3.7(a)). Moreover comparing the segments related to the points contained into Ω_i (orange points in Figure 3.7(a)) with the segments related to q_i (Figure 3.7(d)), the second ones are less influenced by noise than the first ones thanks to the equation (3.6) which can be seen as a noise filtering operation.
As a drawback, using the described insertion strategy, segments with very short length could be chosen after the new virtual point insertion; moreover starting from a single segment, two new segments are always obtained and, as a consequence, the number of segments could increase step by step with a resulting high computational cost required by the algorithm. To face these

problems, a **minimum acceptable segment length**, $\sigma > 0$, has been defined and if both new segments have a length less or equal than σ, they are discarded.

Let $\mathbb{E}_{k-1} = [p_{k-1,1}, \ldots, \mathbf{p_{k-1,j}}, \mathbf{p_{k-1,j+1}}, \ldots, p_{k-1,n_{k-1}}]$ be the array containing the landmarks, and let $\mathbf{q_i}$ be a virtual point to insert into this array. If $||\mathbf{p_{k-1,j}} - \mathbf{q_i}|| > \sigma$ or $||\mathbf{q_i} - \mathbf{p_{k-1,j+1}}|| > \sigma$ then the resulting sorted array will be

$$\mathbb{E}_{k-1} = \left[p_{k-1,1}, \ldots, \mathbf{p_{k-1,j}}, \mathbf{q_i}, \mathbf{p_{k-1,j+1}}, \ldots, p_{k-1,n_{k-1}}\right]$$

where the points $\mathbf{p_{k-1,j}}, \mathbf{p_{k-1,j+1}}$ are computed using the previously described strategy. The related set of elements will be

$$\tilde{\mathcal{E}}_{k-1} = \mathcal{E}_{k-1} \bigcup \{\mathbf{q_i}\}, \quad \mathcal{E}_{k-1} = \tilde{\mathcal{E}}_{k-1}.$$

Otherwise, if the minimum segments length condition is not satisfied, the new virtual point is not added and the array $\mathbb{E}_{k-1}$ and the set $\mathcal{E}_{k-1}$ do not change.

Note that the parameter σ has to be accurately chosen depending on the desired performance. If $\sigma \to 0$ then very short segments are accepted resulting in a more precise environment estimation but also in a high computational cost. On the contrary, if a high σ value is chosen then the algorithm computational cost will be low but the resulting estimation and mapping performance can be very poor. After all the new virtual points have been considered, the obtained array $\mathbb{E}_{k-1}$ and its related set of points $\mathcal{E}_{k-1}$ can be considered as the updated environment map, thus $\mathbb{E}_k = \mathbb{E}_{k-1}$ and $\mathcal{E}_k = \mathcal{E}_{k-1}$. The proposed landmark extraction and data association process can be summarized by the landmark extraction and data association function

$$\mathbb{E}_k = \mathcal{L}(\mathbb{E}_{k-1}, y_k, \hat{x}_{k|k-1})$$

which can be used in the equations (3.3).

3.4.2 State estimation, State and Landmark Update

Using the proposed segment based mapping technique, the augmented state results in

$$X_k = [x_k^T \ \mathbb{E}_k^T]^T = [x_k^T, p_{k,1}, p_{k,2}, \ldots, p_{k,n_k}]^T$$

and it contains the robot pose x_k and all the starting and ending points of the environment modeling segments. Please note that each point $p_{k,i}$ is represented by its x_1 and x_2 coordinates and thus the dimension of the augmented state is $3 + 2 \times n_k$ and it is a time varying dimension.
The model output equation (2.13) is a function of both the robot state and the environment modeling segments. Since the parameters (s, c) of each segment can be related to its starting and ending points, the output function can be seen as a function of the augmented state X_k:

$$y_k = h(x_k, (\overline{s}_k, \overline{c}_k)) + v_k = h(X_k) + v_k.$$

The output matrix will be

$$\mathcal{C}_k = \begin{bmatrix} C_{1,k} & \dfrac{\partial h_1}{\partial p_{k,1}} & \dots & \dfrac{\partial h_1}{\partial p_{k,n_k}} \\ \vdots & & \vdots & \\ C_{n_S,k} & \dfrac{\partial h_{n_S}}{\partial p_{k,1}} & \dots & \dfrac{\partial h_{n_S}}{\partial p_{k,n_k}} \end{bmatrix}$$

where n_S is the robot number of sensors, $C_{i,k}$ is the i-th row of the C_k matrix computed as $C_k = \left.\frac{\partial h(x_k)}{\partial x_k}\right|_{x_k=\hat{x}_{k|k-1}}$, h_i is the function $h(X_k)$ related to the i-th sensor and $\dfrac{\partial h_i}{\partial p_{k,i}}$ is a two dimensional array containing the partial derivatives of h_i w.r.t the $p_{k,i}$ coordinates.
Due to the landmark extraction process, new points can be inserted into the current mapping $\mathbb{E}_k$ or old points can be removed from it.
Let $\mathbb{E}_{k-1} = [p_{k-1,1}, \dots, p_{k-1,j}, p_{k-1,j+1}, \dots, p_{k-1,n_{k-1}}]$ and assume that $\mathcal{L}(\cdot)$ extracts the point $\mathbf{p}$ and returns

$$\mathbb{E}_k = [p_{k-1,1}, \dots, p_{k-1,j}, \mathbf{p}, p_{k-1,j+1}, \dots, p_{k-1,n_{k-1}}].$$

The augmented state becomes $X_k = [x_k^T,\ \mathbb{E}_k^T]^T$ and, as shown in [39], the estimation error covariance matrix related to the augmented state after the new landmark insertion is

$$\tilde{\mathcal{P}}_k = G_x \mathcal{P}_k G_x^T + G_y V G_y^T\ ,\ \mathcal{P}_k = \tilde{\mathcal{P}}_k. \tag{3.7}$$

The matrix G_x is

$$G_x = \left[\begin{array}{cccccc} I_3 & \emptyset & \dots & \emptyset & \emptyset & \dots\ \emptyset \\ \emptyset & I_{2,1} & \dots & \emptyset & \emptyset & \dots\ \emptyset \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \emptyset & \emptyset & \dots & I_{2,j} & \emptyset & \dots\ \emptyset \\ \hline \dfrac{\partial \mathbf{g_x}}{\partial \mathbf{x_k}} & \emptyset & \dots & \emptyset & \emptyset & \dots\ \emptyset \\ \hline \emptyset & \emptyset & \dots & \emptyset & I_{2,j+1} & \dots\ \emptyset \\ \vdots & \vdots & & \vdots & \vdots & \ddots\ \vdots \\ \emptyset & \emptyset & \dots & \emptyset & \emptyset & \dots\ I_{2,n_k} \end{array}\right]$$

and $I_{2,j}$ is the identity matrix of order two related to the j-th landmark while $\emptyset$ is a null matrix of appropriate size.
The matrix G_y is

$$G_y = \left[\emptyset\ \dots\ \emptyset \left| \frac{\partial \mathbf{g_x}}{\partial \mathbf{y_k}} \right| \emptyset\ \dots\ \emptyset\right]^T$$

and $\frac{\partial g_x}{\partial y_k}$ is the $j+1$-th row of this matrix. In the above matrices the function g_x is the landmark extraction function (3.6). For what regards the landmark elimination, if the landmark extraction process deletes a point p_i from $\mathbb{E}_k$, then the corresponding entries in the augmented state and in the matrix $\mathcal{P}_k$ have to be removed. Following the above update rules, the update function

$$(X_k^*, \mathcal{P}_k^*) = \mathcal{U}(X_k, \mathcal{P}_k, \mathbb{E}_{k-1}, \mathbb{E}_k)$$

can be defined and used into the equations (3.3).

3.4.3 Remarks

1. The proposed segment based SLAM algorithm represents a consistent and useful technique since it has been developed looking for a very simple but efficient strategy in terms of the algorithm computational requirements. In particular, the landmark extraction process can be calibrated, depending on the chosen σ value, in order to favor the low computational cost constraint (high σ values) or to obtain better mapping results (low σ values).
2. For what regards the chosen landmarks, the modeling segments' starting and ending points comply with the mandatory landmarks properties described in Section 3.3.1. In particular starting and ending points are:
   - easily re-observable since their related segment maps an entire environment part which can also be very also large;
   - completely distinguishable each other thanks to their coordinates and to the related segment mathematical characterization ((s, c) parameters);
   - stationary.

   Moreover, thanks to the use of the σ parameters, the number of segment, and thus of landmarks, can be controlled to not be too high nor too low.
3. The technique is based on very simple geometric considerations starting from the sensors acquired measurements. Two main drawbacks can be stated about this technique. First of all, the mapping algorithm provides a set of segments connected each other. If the robot has detected and mapped two environment boundaries portions, mapping them using the segments b_1 and b_2, but it has no information about the portion in the middle of them, the above portion is mapped, **in a very poor way**, by the segment connecting the b_1 ending point and the b_2 starting point. As a second drawback, the resulting segment based map can be a very poor approximation of the robot surrounding environment due to the low *versatility* of each segment. In other words, the segment based mapping is very cheap thanks to the low required information to map the environment using a set of segments but due to the low information contained in each segment, the obtained mapping performance can be very poor.

3.5 Polynomial based SLAM

The other two proposed mapping techniques have been developed looking at a more powerful mapping model w.r.t. the previously described segment based one. The environment will be modeled as a set of m-th order polynomials where m is chosen *a-priori* (see Figure 3.8 for the case $n_S = 5$). Starting from this model, at each step k, the environment is described by a set of n_k polynomials, $\mathbb{E}_k = \{z_k^j\}_{j=1}^{n_k}$. Again, the obtained map works like an envelope for the environment boundaries.

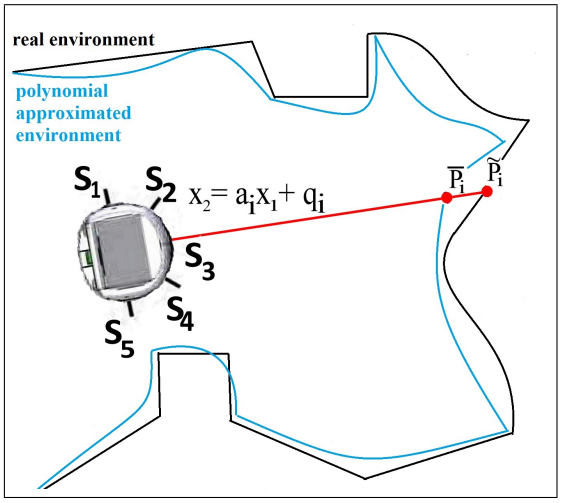


Fig. 3.8: Real environment (black line), environment approximating polynomials (light blue line)

Each sensor S_i provides, at each time step k, the distance from the robot center $P = (x_{1,k}^R, x_{2,k}^R)$, to one point on the environment boundaries, $\tilde{P}_i = (\tilde{x}_1, \tilde{x}_2)$. Considering the model used for the environment, the measurement provided by the i-th sensor, S_i, is approximated, as shown in Figure 3.8 for the case $S_i = S_3$ and $n_S = 5$, by the distance from P to the intersection point, denoted by $\overline{P}_i = (\overline{x}_1, \overline{x}_2)$, between the axis of the sensor S_i and one of the environment modeling polynomials.
Marking the axis of the sensor S_i as $x_2 = a_i x_1 + q_i$ and denoting the intercepted polynomial as z^x, the coordinates $\overline{x}_1, \overline{x}_2$ of the intersection point between the sensor axis and z^x are a function of the polynomial coefficients and of the sensor axis parameters. The measurement $y_{i,k}$, provided by sensor S_i at time step k, can be modeled as

$$y_{i,k} = \sqrt{(x_{1,k}^R - \tilde{x}_1)^2 + (x_{2,k}^R - \tilde{x}_2)^2} \tag{3.8}$$

and approximated by

$$y_{i,k} \approx \sqrt{(x_{1,k}^R - \overline{x}_1)^2 + (x_{2,k}^R - \overline{x}_2)^2}. \tag{3.9}$$

By replacing (3.9) with (2.12), an observation function, $y_{i,k}$, depending only on the robot state and on the polynomial z_x can be obtained as

$$y_{i,k} \approx h((x_{1,k}^R, x_{2,k}^R, \theta_k), z^x)), \quad i = 1, \ldots, n_S$$

These relationships define the robot model output equation and they will be written in the more compact form

$$y_k \approx h(x_k, \overline{z}_k) + v_k, \tag{3.10}$$

where the dimension of vector y_k is n_S, $\overline{z}_k$ collects the polynomials z^x intercepted by each sensor axis at step k and v_k is a Gaussian noise assumed uncorrelated with the process noise w_k (see equation (1.4)) and used to model the measurement noise.

The goal of the present work is to estimate the position and orientation of the robot, $x_k = [x_{1,k}^R \quad x_{2,k}^R \quad \theta_k]^T$, and to simultaneously find the best polynomial based approximation of the robot surrounding environment, $\mathbb{E}_k = \{z_k^j\}_{j=1}^{n_k}$. To achieve this goal, two polynomial based SLAM algorithms will be described in the next Sections. The first one has been developed looking at high mapping performance but it results in a computationally very onerous SLAM technique; it will be denoted as Polynomial based SLAM (PbSLAM). The second proposed polynomial based SLAM technique tries to achieve mapping performance as good as the ones obtained by the PbSLAM but requiring lower computational costs; this technique will be indicated as Efficient Polynomial based SLAM (EPbSLAM).

3.5.1 PbSLAM: Landmark extraction and Data association

The proposed Polynomial based SLAM technique uses polynomials as landmarks. Indicating with $\{b_{d,k}^j\}_{d=0}^m$ the polynomial coefficients, each landmark l_k^j will be $l_k^j = z_k^j = \{b_{d,k}^j\}_{d=0}^m$.
At each step k let $\mathcal{M}_k$ be the set of all the acquired environment points (computed using equations (2.14), (2.15), the state prediction $\hat{x}_{k|k-1}$ and the measurements y_k provided by the on board distance based sensors) until step k; the main idea behind the proposed landmark extraction and data association algorithm is to approximate the environment boundaries by clustering the set $\mathcal{M}_k$ into n_k subsets and associating a polynomial to each cluster. More precisely, $\mathcal{M}_k$ will be partitioned into $\mathcal{B}_k = \{B_k^j, j = 1, \ldots, n_k\}$, a polynomial z_k^j will be related to each set $B_k^j \in \mathcal{B}_k$ and the final resulting environment map will be formed by the polynomials set $\mathbb{E}_k$ and by the points partitioning $\mathcal{B}_k$.

First of all, given a set of points $\mathcal{A}$, a point $P = (x_1^p, x_2^p)$ and a polynomial z, the following functions have been defined:

- $ELMS(z, \mathcal{A})$ which computes the least mean square error due to the approximation of each point in $\mathcal{A}$ using z;
- the set division function $\mathcal{D}(\mathcal{A}, P, z)$ which returns the partition $(\mathcal{A}_1, \mathcal{A}_2)$ of $\tilde{\mathcal{A}} = \{\mathcal{A} \bigcup \{P\}\}$, such that

$$\begin{aligned} &\text{If } z \text{ is a } x_1\text{1-variate polynomial} \\ &\mathcal{A}_1 = \{P_j = (x_1^j, x_2^j) \in \tilde{\mathcal{A}} : x_1^j \leq x_1^p\}; \\ \\ &\text{else if } z \text{ is a } x_2\text{-variate polynomial} \\ &\mathcal{A}_1 = \{P_j = (x_1^j, x_2^j) \in \tilde{\mathcal{A}} : x_2^j \leq x_2^p\} \end{aligned}$$

  and $\mathcal{A}_2 = \tilde{\mathcal{A}} \setminus \mathcal{A}_1$.

The landmark extraction and data association process starts from the current partition $\mathcal{B}_{k-1} = \{B_{k-1}^i, i = 1, \ldots, n_{k-1}\}$ of $\mathcal{M}_{k-1}$ and from the related set of landmarks $\mathbb{E}_{k-1} = \{z_{k-1}^i, i = 1, \ldots, n_{k-1}\}$ and properly modifies this partition and this polynomials set $\mathbb{E}_{k-1}$ to include the currently acquired environment boundaries points π_k^i (computed using (2.14)), in order to obtain $\mathcal{B}_k$ and $\mathbb{E}_k$.

For each point π_k^i the data association step is performed; the main idea behind this process is closeness between acquired environment points. Two points P_1, P_2 are defined *neighbors* iff they satisfy the closeness condition defined in Section 2.6.1.

Using the set-valued *closeness function* $\mathcal{N}$ (defined in Section 2.6.1), the point π_k^i will be associated to the cluster B_{k-1}^j and, consequently, to the landmark z_{k-1}^j, such that

$$B_{k-1}^j = \max_{B_{k-1}^r \in \mathcal{B}_{k-1}} \{|\mathcal{N}(\pi_k^i, B_{k-1}^r, R)|\}. \tag{3.11}$$

B_{k-1}^j is the cluster which contains the biggest number neighbors, in a radius $R > 0$, of π_k^i.

If there is not a cluster which can be associated to π_k^i, that is $|\mathcal{N}(\pi_k^i, B_{k-1}^r, R)| = 0, \forall B_{k-1}^r \in \mathcal{B}_{k-1}$, then a new cluster has to be created containing only the point π_k^i. Therefore, at the end the clusters and environment update will be

$$\mathcal{B}_k = \mathcal{B}_{k-1} \bigcup \{\pi_k^i\}; \quad \mathbb{E}_k = \mathbb{E}_{k-1}$$

Otherwise, the approximation error due the use of z_{k-1}^j to model π_k^i is computed as $\varepsilon = ELMS(z_{k-1}^j, \{\pi_k^i\})$. If this error is lower than a defined threshold $\varepsilon_{TH} > 0$, which is one of the algorithm parameter, then

$$B_{k-1}^j = B_{k-1}^j \bigcup \{\pi_k^i\}; \quad \mathcal{B}_k = \mathcal{B}_{k-1}; \quad \mathbb{E}_k = \mathbb{E}_{k-1}$$

Else if $\varepsilon > \varepsilon_{TH}$ then the total approximation error on B_{k-1}^j using z_{k-1}^j is computed as

$$\varepsilon_a = ELMS(z_{k-1}^j, B_{k-1}^j \bigcup \{\pi_k^i\}).$$

At this point, a new m-th order polynomial is obtained using the points in $\{B_{k-1}^j \bigcup \{\pi_k^i\}\}$:

$$z_b = LMS(B_{k-1}^j \bigcup \{\pi_k^i\}, m)$$

and the related approximation error is

$$\varepsilon_b = ELMS(z_b, B_{k-1}^j \bigcup \{\pi_k^i\}).$$

Finally, the set B_{k-1}^j is partitioned into (B_c, B_d) as

$$(B_c, B_d) = \mathcal{D}(B_{k-1}^j \bigcup \{\pi_k^i\}, \pi_k^i, z_{k-1}^j)$$

and the related polynomials are

$$z_c = LMS(B_c, m),\ z_d = LMS(B_d, m)$$

and

$$\varepsilon_c = ELMS(z_c, B_c),\ \varepsilon_d = ELMS(z_d, B_d),$$
$$\varepsilon_e = \xi(\varepsilon_c + \varepsilon_d)$$

where $\xi > 0$ is one of the algorithm parameters and ε_e is considered as the approximation error due to the use of z_c and z_d as environment modeling polynomials.
At this point:

- if $\min(\varepsilon_a, \varepsilon_b, \varepsilon_e) = \varepsilon_a$ then the polynomial z_{k-1}^j is used to approximate the points in $B_{k-1}^j \bigcup \{\pi_k^i\}$ and

$$B_{k-1}^j = B_{k-1}^j \bigcup \{\pi_k^i\};\ \ \mathcal{B}_k = \mathcal{B}_{k-1};\ \ \mathbb{E}_k = \mathbb{E}_{k-1}$$

- otherwise if $\min(\varepsilon_a, \varepsilon_b, \varepsilon_e) = \varepsilon_b$ then the polynomial z_b is used to approximate the points in $B_{k-1}^j \bigcup \{\pi_k^i\}$ and

$$B_{k-1}^j = B_{k-1}^j \bigcup \{\pi_k^i\};$$
$$\mathcal{B}_k = \mathcal{B}_{k-1};$$
$$\mathbb{E}_k = \{\mathbb{E}_{k-1} \setminus \{z_{k-1}^j\}\} \bigcup \{z_b\}$$

- else if $\min(\varepsilon_a, \varepsilon_b, \varepsilon_e) = \varepsilon_e$, then the best possible choice is to divide the cluster B_{k-1}^j and to use the two obtained polynomials z_c, z_d instead of z_{k-1}^j:

$$\mathcal{B}_k = \{\mathcal{B}_{k-1} \setminus B_{k-1}^j\} \bigcup \{B_c, B_d\};$$
$$\mathbb{E}_k = \{\mathbb{E}_{k-1} \setminus \{z_{k-1}^j\}\} \bigcup \{z_c, z_d\}$$

The ξ parameter is related to the number of clusters that will be found at the end of the landmark extraction process and **the bigger the value of ξ is, the lower the number of clusters will be**.

Obviously, following only the previous steps, the landmark extraction process may yield to a continuously increasing number of clusters and polynomials. To avoid a too big number of clusters, achieving better computational performance, every $K_s \in \mathbb{N}$ steps a unification process is executed. A cluster B_k^i is defined *close* to a cluster B_k^j if it has at least N_p points which are neighbors to at least one point in B_k^j. Using the above definition, during the unification process, given each possible couple of close clusters B_k^i, B_k^j and the related polynomials z_k^i, z_k^j and approximation errors $\varepsilon_i, \varepsilon_j$, the polynomial $z = LMS(B_k^i \bigcup B_k^j, m)$ is computed along with the related approximation error $\varepsilon = ELMS(z, B_k^i \bigcup B_k^j)$. At this point given the algorithm parameter $\rho > 0$, if $\varepsilon < \rho(\varepsilon_i + \varepsilon_j)$ then the partition is modified as

$$\begin{aligned} \mathcal{B}_k &= \mathcal{B}_k \setminus \{B_k^i, B_k^j\} \bigcup \{B_k^i \bigcup B_k^j\}; \\ \mathbb{E}_k &= \mathbb{E}_k \setminus \{z_k^i, z_k^j\} \bigcup \{z\} \end{aligned}$$

The bigger the value of ρ is, the bigger will be the effect of the unification process over the landmark extraction process.
As in the Segment based SLAM, the entire landmark extraction and data association process is summarized by a function

$$(\mathcal{B}_k, \mathbb{E}_k) = \mathcal{L}(\mathcal{B}_{k-1}, \mathbb{E}_{k-1}, y_k, \hat{x}_{k|k-1})$$

which can be used into the equations (3.3).

3.5.2 PbSLAM: State estimation, State and Landmark Update

The following augmented state is defined

$$X_k = [x_k^T,\ b_{m,k}^1, \ldots, b_{0,k}^1\ , \ldots,\ b_{m,k}^{n_k}, \ldots, b_{0,k}^{n_k}]^T = [x_k^T,\ z_k^1, \ldots, z_k^{n_k}]^T$$

Where $l_k^j = z_k^j = [b_{m,k}^j, \ldots, b_{0,k}^j]$. This augmented state contains the robot pose and all the $m+1$ coefficients $\{b_{q,k}^j\}_{q=0}^m$ of the j-th m-th order polynomial $z_k^j, j = 1, \ldots, n_k$.
The output equation (3.10) is a function of both the robot state and the environment modeling polynomials, therefore it can be seen as a function of the augmented state X_k:

$$y_k = h(x_k, \overline{z}_k) + v_k, = h(X_k) + v_k$$

The output matrix will be

$$\mathcal{C}_k = \begin{bmatrix} C_{1,k} & \frac{\partial h_1}{\partial z_k^1} & \cdots & \frac{\partial h_1}{\partial z_k^{n_k}} \\ \vdots & & \vdots & \\ C_{n_S,k} & \frac{\partial h_{n_S}}{\partial z_k^1} & \cdots & \frac{\partial h_{n_S}}{\partial z_k^{n_k}} \end{bmatrix}$$

where $C_{i,k}$ is the i-th row of the C_k matrix (computed as $C_k = \frac{\partial h(x_k)}{\partial x_k}\Big|_{x_k=\hat{x}_{k|k-1}}$), h_i is the function $h(X_k)$ related to the i-th sensor and $\frac{\partial h_i}{\partial z_k^i} = [\frac{\partial h_i}{\partial b_k^{m,i}} \ \cdots \ \frac{\partial h_i}{\partial b_k^{0,i}}]$.

For what regards the estimation error covariance matrix $\mathcal{P}_k$, when a new landmark is extracted, this matrix is modified using the equation (3.5). As a heuristic each new extracted landmark is assumed to be not correlated with the past ones and the landmarks are assumed to be not correlated with the state estimate. That is, given $q \neq j$:

$$P(\hat{l}_k^q, \hat{l}_k^q) = \emptyset;\ P(\hat{l}_k^q, \hat{x}_k) = \emptyset.$$

where $\emptyset$ is a null matrix of appropriate size. Each new landmark estimation error covariance matrix is initialized as

$$P(\hat{l}_k^q, \hat{l}_k^q) = I_{m+1} \times P_{landmark};$$

where $P_{landmark} > 0$ is one of the algorithm parameters and it represents the landmark initial estimation error covariance.

On the basis of the above update rules, the following update function is defined

$$(X_k^*, \mathcal{P}_k^*) = \mathcal{U}(X_k, \mathcal{P}_k, \mathbb{E}_{k-1}, \mathbb{E}_k)$$

and it can be used into the equation (3.3) along with the previously defined PbSLAM landmark extraction and data association function; the resulting formulation represent the Polynomial based SLAM algorithm.

3.5.3 PbSLAM remarks

1. As in the case of the Segment based mapping, the proposed Polynomial based SLAM technique is based on geometric considerations starting from the acquired measurements. The main difference w.r.t the SbSLAM is related to the higher amount of information provided by polynomials instead of segments. This additional information ensures to better map the environments boundaries since the polynomial based environment model can be easily adapted to whatever indoor environment.
2. As a drawback, the computational cost due to the landmark extraction and data association process can be very high and it can increase as the simulation/experiment goes on. As the simulation time grows, the number of acquired points grows and, as a consequence, the computational

cost due to the points clustering procedure through equation (3.11) and $ELMS(\cdot)$ function evaluation can become unacceptable with a resulting too high computation time. In these situations, the proposed polynomial based solution can not be used in real experiments due to possible time constraints violation.

3.5.4 EPbSLAM: Landmark extraction and Data association

As previously remarked, in a computational point of view, the proposed PbSLAM algorithm can be very onerous. The PbSLAM *weak point* is the landmark extraction and data association process which is based on a points clustering procedure. To overcome this defect, in the following, a new polynomial based SLAM technique will be described; this algorithm has been developed looking for a more efficient technique and for a lower computational cost w.r.t. the PbSLAM requirements. The goal is to obtain mapping results as good as in the PbSLAM case but avoiding the points clustering operation.

As in the PbSLAM case, the proposed efficient Polynomial based SLAM algorithm uses a set of n_k m-th order polynomials $\mathbb{E}_k = \{z_k^j, j = 1, ..., n_k\}$ to approximate the environment boundaries. Given a polynomial z_k^j and a point q, the following notation will be used:

- the polynomial is represented by a set of coefficients, $\{b_{m,k}^j,\ b_{m-1,k}^j,\ \cdots,\ b_{1,k}^j,\ c_{0,k}^j\}$, where $\{b_{q,k}^j\}_{q=1}^m\}$ are related to the polynomial shape while $c_{0,k}^j$ is related to the polynomial position and it will be denoted as the **polynomial position coefficient** in the next Sections. For example, a x_1-variate polynomial, z_k^j, will be modeled as

$$x_2 = z_k^j(x_1) = b_{m,k}^j(x_1)^m + b_{m-1,k}^j(x_1)^{m-1} + \cdots + b_{1,k}^j(x_1) + c_{0,k}^j$$

- $x_1^c(q, z_k^j)$ returns the x_1 coordinate of q if z_k^j is a x_1-variate polynomial, otherwise it returns the x_2 coordinate of q. Following the same lines, $x_2^c(q, z_k^j)$ returns the x_2 coordinate of q if z_k^j is a x_1-variate polynomial, else it returns the x_1 coordinate of q;
- $\mathcal{Q}_{bad}(z_k^j)$ is a set of points badly approximated by z_k^j. This points set is defined as:

$$\mathcal{Q}_{bad}(z_k^j) = \{Q_m : \rho_m < |x_2^c(Q_m, z_k^j) - z_k^j(x_1^c(Q_m, z_k^j))| \leq \sigma_m\}$$

  where $\sigma_m > \rho_m > 0$ are fixed thresholds representing the maximum allowed approximation error and Q_m is one of the points that should be approximated by z_k^j; Fig. 3.9 shows an example of the above points set;
- $q_s^j = (x_1^s, x_2^s)$,$q_e^j = (x_1^e, x_2^e)$ are the starting point and ending point of the environment boundaries section approximated by z_k^j.
- $\mathcal{Z}_{k-1} = \{Z_{k-1}^j, j = 1, \ldots, n_z\}$ is a set of points clusters. Each cluster contains a set of environment boundaries points which have not been approximated by any polynomial yet.

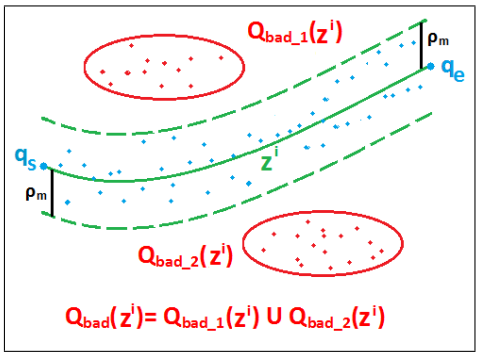


Fig. 3.9: An environment boundaries section

At each step k, using the set of currently acquired environment boundaries points $\pi_k^i, i = 1, \ldots, n_S$, computed through equation (2.14), the landmark extraction and data association process modifies the polynomials set $\mathbb{E}_{k-1}$ and the set of points clusters $\mathcal{Z}_{k-1}$ into $\mathbb{E}_k$ and $\mathcal{Z}_k$.

For each acquired point π_k^i, the first step is to compute the set of mapping polynomials which could be used to approximate the point:

$$\mathbb{E}_k^* = \{z_k^j \in \mathbb{E}_k : x_1^c(q_s^j, z_k^j) \leq x_1^c(\pi_k^i, z_k^j) \leq x_1^c(q_e^j, z_k^j)\}$$

If $\mathbb{E}_k^* = \{\emptyset\}$ the point π_k^i can not be approximated by the polynomials in $\mathbb{E}_k$ and it is then inserted into the cluster $Z_{k-1}^j \in \mathcal{Z}_{k-1}$ containing the highest number of points neighbors, in a radius R, to π_k^i. If there is not a cluster satisfying this neighborhood condition, a new cluster is created containing only the point π_k^i:

$$\tilde{\mathcal{Z}}_{k-1} = \mathcal{Z}_{k-1} \bigcup \{\pi_k^i\};\ \mathcal{Z}_{k-1} = \tilde{\mathcal{Z}}_{k-1}.$$

If $\mathbb{E}_k^* \neq \{\emptyset\}$ then the following optimization problem is solved:

$$z_k^* = \arg \min_{z_k^j \in \mathbb{E}_k^*} |x_2^c(\pi_k^i, z_k^j) - z_k^j(x_1^c(\pi_k^i, z_k^j))|. \tag{3.12}$$

Three cases may occur. The first one is related to

$$|x_2^c(\pi_k^i, z_k^*) - z_k^*(x_1^c(\pi_k^i, z_k^*))| \leq \rho_m;$$

in this situation, z_k^* is good to approximate π_k^i and no further actions are performed.
The second case occurs when

$$\rho_m < |x_2^c(\pi_k^i, z_k^*) - z_k^*(x_1^c(\pi_k^i, z_k^*))| \leq \sigma_m,$$

where $\sigma_m > \rho_m$ is another given threshold. In this situation, the approximation error is assumed to be only due to the low accuracy of the polynomial coefficients. The point π_k^i is then stored in $\mathcal{Q}_{bad}(z_k^*)$
The third possible case occurs when the approximation error related to π_k^i using z_k^* is too high, i.e.

$$|x_2^c(\pi_k^i, z_k^*) - z_k^*(x_1^c(\pi_k^i, z_k^*))| > \sigma_m.$$

To face this situation, the point π_k^i is inserted into one of the clusters in $\mathcal{Z}_{k-1}$, following the same lines used in the case $\mathbb{E}_k^* = \{\emptyset\}$.
The landmark extraction and data association process could modify the mapping polynomials in the second and in the third case. More precisely:

- in the second case, after the insertion of a point π_k^i into the set $Q_{bad}(z)$, if the number of points in the above set becomes greater than a fixed threshold ε_M, then the polynomial z is modified trying to better adapt z to the badly approximated points contained in $\mathcal{Q}_{bad}(z)$. A set of points on z is computed as

$$\begin{array}{l} \mathcal{R} = \{Q^a : Q^a = (x_1^a, z(x_1^a)) \ , \ x_1^s \leq x_1^a \leq x_1^e, \\ a = 1, \ldots, n_a \ , x_1^1 = x_1^s, x_1^{n_a} = x_1^e\} \\ \text{if } z \text{ is a } x_1\text{-variate polynomial,} \\ \\ \mathcal{R} = \{Q^a : Q^a = (z(x_2^a), x_2^a) \ , \ x_2^s \leq x_2^a \leq x_2^e, \\ a = 1, \ldots, n_a \ , x_2^1 = x_2^s, x * n_{a2} = x_2^e\} \\ \text{otherwise;} \end{array} \tag{3.13}$$

  where $x_1^a, a = 1, \ldots, n_a$ and $x_2^a, a = 1, \ldots, n_a$ are $n_a > 0$ equally spaced values on the x_1-axis and on the x_2-axis respectively.
  At this point, a new polynomial $\underline{z}$ is computed by minimizing the least mean square error due to the use of the polynomial $\underline{z}$ in approximating the points contained in $\mathcal{Q}_{bad}(z) \bigcup \mathcal{R}$:

$$\underline{z} = LMS(\mathcal{Q}_{bad}(z) \bigcup \mathcal{R}, m).$$

  The resulting new polynomial $\underline{z}$ will be used to map the environment and will substitute the polynomial z. More precisely

$$\tilde{\mathbb{E}}_{k-1} = \{\mathbb{E}_{k-1} \bigcup \{\underline{z}\}\} \setminus \{z\}; \ \mathbb{E}_{k-1} = \tilde{\mathbb{E}}_{k-1}$$

  and the related set of badly approximated points will be $\mathcal{Q}_{bad}(\underline{z}) = \{\emptyset\}$.
- In the third case ($|x_2^c(\pi_k^i, z_k^*) - z_k^*(x_1^c(\pi_k^i, z_k^*))| > \sigma_m$), after the insertion of a point π_k^i into a cluster Z_{k-1}^j, if the number of points in the cluster becomes greater than ε_M, then the cluster is removed from $\mathcal{Z}_{k-1}$

$$\tilde{\mathcal{Z}}_{k-1} = \mathcal{Z}_{k-1} \setminus Z_k^j; \ \mathcal{Z}_{k-1} = \tilde{\mathcal{Z}}_{k-1},$$

and a new polynomial z is computed by minimizing the least mean square error over the points in Z_k^j

$$z = LMS(Z_k^j, m).$$

The obtained polynomial is then inserted into the landmarks set

$$\tilde{\mathbb{E}}_{k-1} = \mathbb{E}_{k-1} \bigcup \{z\};\ \mathbb{E}_{k-1} = \tilde{\mathbb{E}}_{k-1}.$$

It is to be specified that, if a new polynomial z^2 approximates or includes an environment part that has been already mapped by another polynomial $z^1 \in \mathbb{E}_{k-1}$, then a third polynomial z^3 is computed by minimizing the least mean square error over the points in $\mathcal{R}_1 \bigcup \mathcal{R}_2$ where $\mathcal{R}_i, i = 1, 2$ are obtained as shown in (3.13), for the polynomials $z^i, i = 1, 2$. The resulting landmarks set will be

$$\tilde{\mathbb{E}}_{k-1} = \{\mathbb{E}_{k-1} \bigcup \{z^3\}\} \setminus \{z^1\};\ \mathbb{E}_{k-1} = \tilde{\mathbb{E}}_{k-1}$$

An example of the above polynomials merge is depicted in Figure 3.10.

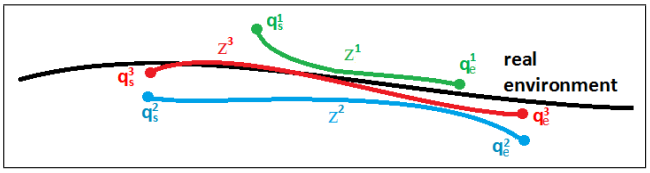


Fig. 3.10: Overlapping polynomials

In conclusion, after all the currently acquired points $\{\pi_k^i, i = 1, \ldots, n_S\}$ have been used, the obtained landmarks array $\mathbb{E}_{k-1}$ and the clusters set $\mathcal{Z}_{k-1}$ become the updated environment map: $\mathbb{E}_k = \mathbb{E}_{k-1}$ and $\mathcal{Z}_k = \mathcal{Z}_{k-1}$.
As in the previous cases, the entire landmark extraction and data association process can be summarized by

$$(\mathcal{L}_k, \mathbb{E}_k) = \mathcal{L}(\mathcal{L}_{k-1}, \mathbb{E}_{k-1}, y_k, \hat{x}_{k|k-1})$$

and this function has to be used into the equations (3.3).

3.5.5 EPbSLAM: State estimation, state and landmark update

In contrast to the PbSLAM algorithm, in the Efficient Polynomial based SLAM the following augmented state has been defined

$$X_k = [x_k^T,\ c_{0,k}^1, \ldots, c_{0,k}^{n_k}]^T$$

containing the robot state and only the position coefficient, $c_{0,k}^j$, of each of the polynomials $z_k^j \in \mathbb{E}_k$. The main idea behind this choice is that, using

distance sensors, the most logic interpretation of the Kalman filter innovation term regarding the mapping polynomials is that, depending on the innovation term, each polynomial has to be moved towards or away from the robot center maintaining its shape (coefficients $\{b^j_{q,k}\}_{q=1}^m$) unchanged. In other words, the landmark extraction process is used to obtain each polynomial shape while the Kalman filter is used to properly move this shape towards or away from the robot. In the landmark extraction process, each landmark is characterized by all the polynomial coefficients while only the polynomial position coefficients are involved into the Kalman filter: $l^j_k = c^j_{0,k}$.

About the output equation (3.10), as in the PbSLAM case, it can be seen as a function of both the robot state and the environment modeling polynomials, therefore it can be written as a function of the augmented state X_k too:

$$y_k = h(x_k, \overline{z}_k) + v_k, = h(X_k) + v_k$$

The output matrix will be

$$\mathcal{C}_k = \begin{bmatrix} C_{1,k} & \frac{\partial h_1}{\partial c^1_{0,k}} & \cdots & \frac{\partial h_1}{\partial c^{n_k}_{0,k}} \\ \vdots & & \vdots & \\ C_{n_S,k} & \frac{\partial h_{n_S}}{\partial c^1_{0,k}} & \cdots & \frac{\partial h_{n_S}}{\partial c^{n_k}_{0,k}} \end{bmatrix}$$

where $C_{i,k}$ is the i-th row of $C_k = \left.\frac{\partial h(x_k)}{\partial x_k}\right|_{x_k=\hat{x}_{k|k-1}}$ and h_i is the function $h(X_k)$ related to the i-th sensor.
After the landmark extraction process for each change in the modeling polynomials set $\mathbb{E}_k$, there is a change also in the estimation error covariance matrix and in X_k. More precisely, if a new polynomial $z^r_k = [b^r_{m,k}, \ldots, b^r_{1,k}, c^r_{0,k}]$ comes out from the landmark extraction process, then the augmented state becomes $X_k = [X^T_k,\ c^r_{0,k}]^T$ and a new row and a new column will be added to the estimation error covariance matrix as shown in (3.5).

As in the PbSLAM case, as a heuristic, each new extracted landmark is assumed to be not correlated with the past ones and the landmarks are assumed not correlated with the state estimate. That is, given $q \neq j$:

$$P(\hat{c}^q_{0,k}, \hat{c}^j_{0,k}) = 0;\ P(\hat{c}^q_{0,k}, \hat{x}_k) = \emptyset.$$

where $\emptyset$ is a null matrix of appropriate size. Each new landmark estimation error covariance matrix is initialized as

$$P(\hat{c}^q_{0,k}, \hat{c}^q_{0,k}) = P_{landmark};$$

where $P_{landmark} > 0$ is the landmark initial estimation error covariance.

Once again, following the above update rules, the update function is defined as:

$$(X_k^*, \mathcal{P}_k^*) = \mathcal{U}(X_k, \mathcal{P}_k, \mathbb{E}_{k-1}, \mathbb{E}_k).$$

and this function has to be used into the equations (3.3) along with the previously defined EPbSLAM landmark extraction and data association function; the resulting formulation represent the Efficient Polynomial based SLAM algorithm.

3.5.6 EPbSLAM remarks

1. The proposed efficient polynomial based SLAM does not require any points clustering since it is assumed that the information contained into the polynomial representing a cluster is sufficient to model the cluster itself. More precisely, when a new polynomial has to be computed using the points in $Q_{bad}(z)$, these points are fused with a set of points computed on z (see equation (3.13)). As a heuristic, the points computed using equation (3.13) are assumed to be a reliable sample for all the points approximated by z. Thanks to this assumption, the proposed mapping algorithm does not require to use, at each step k, all the points in $\mathcal{M}_k$ and thus the algorithm memory and computational costs are lower than the costs to perform the polynomial based mapping required by the PbSLAM algorithm.
2. However, as a drawback, given a boundaries portion approximated by a polynomial z, if this polynomial is not good enough to approximate the points related to the above portion, using the equation (3.13) instead of clustering the points in $\mathcal{M}_k$ may yield to poor environment approximation.
3. When a point π_k^i has to be inserted into one of the clusters contained in $\mathcal{Z}_{k-1}$, to find the correct cluster, Z_{k-1}^j, the equation (3.11) is used as in the PbSLAM case. However, in the EPbSLAM case, this equation is computed on a lower number of points since the cardinality of each cluster contained in $\mathcal{Z}_{k-1}$ is usually lower than the cardinality of each cluster in $\mathcal{B}_{k-1}$. The key point is that while in the PbSLAM case, the clusters contain all the acquired environment points, in the EPbSLAM case they contain only the not approximated points and once a cluster cardinality becomes greater than a given threshold ϵ_M, the approximating polynomial is computed and **the cluster is removed from** $\mathcal{Z}_{k-1}$. As a consequence, the number of points contained into a cluster Z_{k-1}^j **can not continuously increase**, while, in the PbSLAM case, the number of points contained into the clusters continuously increase as the simulation/experiment goes on.

In conclusion, the proposed efficient polynomial based mapping is faster and computationally cheaper than the polynomial based SLAM but the resulting mapping performance may be worse.

3.6 Chapter Conclusions

In this chapter the proposed solutions to the mobile robots simultaneous localization and mapping problem have been shown. The discussed topics have shown how changing the chosen landmarks and the technique to obtain them, the resulting mapping and computational performance can vary. In particular, three SLAM techniques have been proposed:

1. SbSLAM with expected low computational cost but sometimes poor mapping performance;
2. PbSLAM with expected high computational cost and high mapping performance;
3. EPbSLAM with expected good computational cost and mapping performance comparable to the PbSLAM;

In Chapter 6 all the above considerations will be proved through numerical simulations and experimental tests.

4

Sensors Switching Logics

To accomplish the mobile robots localization task (in the mobile robots localization problems and in the SLAM problems), it seems intuitive that **the more sensors you use, the better estimate you get**. However, especially using some sensors types, there are reasons that suggest not too exceed with their number or use. The most important is that sensors employ robot's batteries, and an intensive use reduces the robot autonomy. This first feature can be critical especially in planetary exploration or rescue applications, where changing robot batteries during the robot work can be hard or impossible.

Moreover there are also situations where multiple sensors cannot operate simultaneously, for example when they use the same frequency band [29]. When the mobile robots localization algorithm is not run on board the robot, which is the most frequent case, one more issue is bandwidth consumption and possible collisions when transferring measurements from the sensors to the algorithm. In these situations, an excessive use of sensors will result in higher energy consumption.

In order to optimize the sensors use, a possible way is to develop an appropriate **sensors switching policy**. In this way, only a part of all the available sensors will be used, and the energy consumption will be consequently reduced. However, this will result in a lower accuracy of the robot pose estimation.

Due to the above considerations the problem is related to finding the **"best"** sequence of activation of a small (fixed) part of the available sensors to obtain the **"best" (in some statistical sense)** robot pose estimation. Within this framework, in this Chapter, two new sensors switching policies are proposed and described. The first one is based on the statistical Kalman filter properties while the second one is related to the sonar sensor physical characteristics.

4.1 Problem Statement

Consider a physical system described by the following nonlinear discrete model (see equations (2.6)):

$$\begin{cases} x_{k+1} = \phi(x_k, u_k) + w_k \\ y_k = \eta(x_k) + v_k \end{cases} \tag{4.1}$$

Where $x_k \in \mathbb{R}^n$ is the model state vector, $u_k \in \mathbb{R}^m$ is the model input vector, $y_k \in \mathbb{R}^p$ is the model output vector and v_k, w_k are the measurement noise and the process noise respectively (see Section 2.3 for more details). In this framework, the physical system has p usable sensors to detect its state and its surrounding environment.

In a standard configuration, at each step k, the entire information provided by these sensors is used and thus each sensor consumes energy to perform its measurements. Moreover, using all the available sensors at the same time, each sensor can influence the measurements provided by the other ones with a resulting noise increment in the obtained measurements.

For example, consider a set of 3 ultrasonic sensors, $S_i, i = 1, 2, 3$, placed as shown in Figure 4.1.

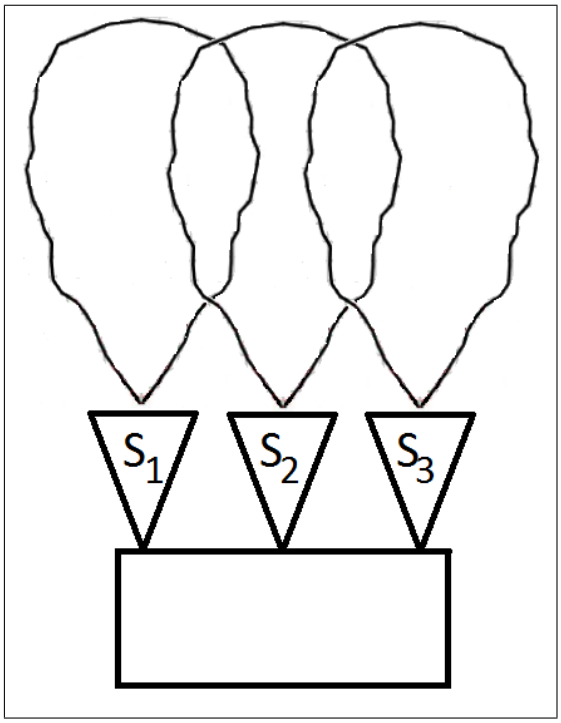


Fig. 4.1: Problems using multiple sensors simultaneously

Considering the ultrasonic sensor beam shape described in Section 1.3.1, the resulting sensors beams will intersect and the resulting measurements can be badly influenced by this intersection phenomenon.
The above motivations are only some of the possible examples about troubles when using multiple sensors simultaneously. To overtake these problems, a possible solution is to **activate only a part of the available sensors at each step** k.

Considering the situation depicted in Figure 4.1, there are two possible activation sequences not affected by any sensors beams intersection:

- at each step k only one of the available sensors is activated, for instance:
  1. $k = 1$, S_1 is used and $S_{2,3}$ are not active
  2. $k = 2$, S_2 is used and $S_{1,3}$ are not active
  3. $k = 3$, S_3 is used and $S_{1,2}$ are not active
  4. $k = 4$, S_1 is used and $S_{2,3}$ are not active
  5. and so on.
- the system switches between two sensors configurations, depending on the time step: a first configuration in which only S_2 is activated while $S_{1,3}$ are not active and a second configuration using S_1 and S_3 while S_2 is not active (see Figure 4.2 for more details).

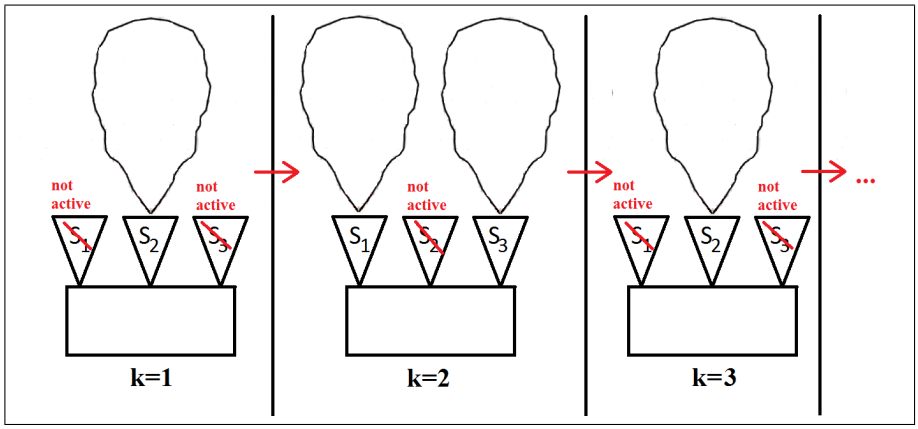


Fig. 4.2: Sensors beams intersection avoidance thanks to the use of a switching rule

If one of the two above described switching policies is used, the sonar sensors beams shown in Figure 4.1 do not intersect and, as a consequence, there are no problems related to the mutual influence between the used sensors.

More formally the sensors switching problem can be formulated as:

At each instant, among the p available sensors, choose q out of them in such a way that a given cost index is minimized.
In a mathematical point of view, at step k, given the set of available sensors $\mathcal{S} = \{S_i\}_{i=1}^{p}$, *find the subset* $\mathcal{Q}_k^* \subset \mathcal{S}$ *such that*

$$\begin{cases} \mathcal{Q}_k^* = \arg \min\limits_{\mathcal{Q} \in \mathcal{S}_q^2} J_s(\mathcal{Q}, k) \\ s.c. \\ \mathcal{S}_q^2 = \{\mathcal{Q} \in \mathcal{S}^2 : |\mathcal{Q}| = q\} \end{cases} \tag{4.2}$$

where

- $\mathcal{S}_q^2$ *is the subset of the power set* $\mathcal{S}^2$ *containing only the sets with q elements;*
- $J_s(\mathcal{Q}, k)$ *is a cost index which has to be minimized and it is a function of the used sensors and of the step k.*

Obviously, depending on the chosen optimization index, the resulting optimal sensors switching policy, $\{\mathcal{Q}_k^*\}_{k=1,2,3,\ldots}$, changes; it is therefore very important to accurately choose the cost function J_s.

Looking at the mobile robot localization problem, if at each instant q out of p sensors are activated, the resulting output equation is a q-valued equation $h(\cdot)$ and the related output matrix is a $q \times n$ matrix, $C_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_{k|k-1}}$.

Various choices of the activation sensors sequence consequently return different estimates of the robot state x_k, whatever is the chosen localization algorithm. Moreover, as far as a switching policy may be good, using a small fraction, q, of the p available sensors will obviously have an impact on the estimate. In conclusion, the goal is to find **the switching policy which provides the best robot pose estimation, in the application of interest, ensuring a correct use of the available sensors**. This problem results in finding the appropriate cost index $J_s(\mathcal{Q}, k)$ depending on the considered application. As a mandatory constraint, the cost function J_s has to be properly chosen in order to avoid a too big degradation on the estimation performance.

Please note that finding the best sensors activation sequence could result in a problem of combinatorial complexity and therefore, to avoid too high computational costs, heuristics have to be defined to look for an activation sequence approximating the best one as more as possible.

In this dissertation two possible switching rules will be described.

4.2 Switching policy based on the observations effect maximization

The first proposed switching policy is based on the **maximization** of the current observations effect on the estimation. The main idea is to chose, at

each instant, q out of the available p sensors, in such a way that the effect of the current observation on the estimate is maximized.

Let the chosen robot localization algorithm be related to the use of the Kalman filter theory (see Section 2.3).

In a linear estimation problem, a meaningful quality measurement of the estimate is the trace of the estimation error covariance matrix $P_{k|k}$, i.e., the mean square estimation error. A simple way to evaluate the effect of the current observation y_k is to consider the trace of the difference between $P_{k|k-1}$ and $P_{k|k}$ (i.e. the *a-priori* and *a-posteriori* estimation errors).

This is correct under the additional assumption that all the state variables are coherent, i.e., they share the same measurement units. If that is not the case, as in the mobile robot localization problem (the robot position coordinates and the robot orientation angle do not share the same measurement units), a *weighted trace* has to be used and the weights both make the sum coherent and also can be chosen to emphasize some eigenvalues of $P_{k|k}$ with respect to the others.

Although in a nonlinear filtering problem the matrix $P_{k|k}$ only is an approximation of the estimation error covariance, its weighted trace has been chosen as the optimization criterion, by analogy with the linear case, and because it is simple to be computed, as it will be shown.

Finally, the switching rule criterion will be

> *At each instant, among the p sensors, choose q of them in such a way that the trace of $J_k = P_{k|k-1} - P_{k|k}$ is maximized. That is $J_s = -trace(J_k)$ in the optimization problem* (4.2).

The proposed approach of maximizing, at each step, the J_k trace has been introduced for the first time in [30], for the state estimation of linear systems using a Kalman filter, and it has been here adapted to the non linear mobile robots localization framework.

The proposed criterion can be applied to all the filtering algorithms proposed in the Chapters 2 and 3, of this dissertation. Although there is no guarantee that this choice may take to the optimal switching sequence, as a heuristic, this criterion will be used to try to improve the localization algorithms performance.

The described localization algorithms are essentially based on the EKF or on the UKF and thus, in the following Subsections, the proposed switching rule will be adapted to these two Kalman filter extensions.

4.2.1 Trace criterion for the Extended Kalman Filter

The difference $J_k = P_{k|k-1} - P_{k|k}$, the trace of which has to be maximized, is simply

$$J_k = P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k^T + V)^{-1} C_k P_{k|k-1}.$$

changing the subset of used sensors, the C_k matrix changes. Therefore the maximization variable is the matrix C_k.

When a small fraction q of sensors is used, the computational cost of this maximization is low, because of many common terms and symmetries, and it is suitable to be done online. The hypothesis that q is small is intrinsic in the sensors switching problem, hence the best trace is easily computed by enumeration of the $\binom{p}{q}$ possible C_k matrices.

4.2.2 Trace criterion for the Unscented Kalman Filter

For the unscented filter the difference $J_k = P_{k|k-1} - P_{k|k}$ amounts to

$$J_k = K_k P_{yy} K_k^T.$$

Here the computation is a little more complex; let Q_k be a $q \times p$ matrix where the j-th entry of each row is one if the j-th sensor is active, and zero otherwise, and define

$$\xi_k = (\chi_{k|k-1} - \hat{x}_{k|k-1})R^c, \quad \psi_k = Q_k(\Gamma_{k|k-1} - \hat{y}_{k|k-1}).$$

Then the value of the matrix J_k can be written as

$$J_k = \xi_k \psi_k^T \left(\xi_k \, R^c \psi_k^T + Q_k V Q_k^T\right)^{-1} \psi_k \xi_k^T,$$

the optimization variable is the matrix Q_k and again, being R^c diagonal and Q_k sparse, and because of the symmetry, the computational cost of the trace index is low.

It is worth noticing that the trace criterion becomes more meaningful when the UKF is used because the approximation of $P_{k|k}$ is correct to the second order.

4.3 Switching policy based on the sensors incidence angles

The second proposed switching policy is based on the use of sonar sensors.in particular, the policy is based on the sonar sensors model described in Section 1.3.1. The main idea is to chose among the p available sensors, the subset of q ones related to the best incidence angles.

Given the set of available sensors $\{S_i, i = 1, \ldots, p\}$, each of them is related to an incidence angle γ_i. The more the incidence angle γ_i is near to $\frac{\pi}{2} rad$, the better the measurement provided by the ultrasonic sensor is. Using (1.5) the following incidence parameter can be defined:

$$r = \cos\gamma = \overrightarrow{y} \cdot \overrightarrow{b} \tag{4.3}$$

since the incidence angle is $\gamma \in (0, \frac{\pi}{2}] rad$, the incidence parameter r is $r \in [0, 1)$. For $r \to 0$, the sensor axis becomes orthogonal to the incidence surface; for $r \to 1$ the sensor axis becomes parallel to the incidence surface.

At this point, each sensor S_i can be related to an incidence parameter r_i and the q sensors related to the lower values of r_i will be chosen.
To take in consideration the influence of the incidence angle into the measurement model related to an ultrasonic sensor, a possible way is to model the measurement provided by such sensor as

$$y_M = y + \Xi(r) \tag{4.4}$$

where y is the real distance between the sensor and the incidence surface (see Figure 1.5), y_M is the measurement provided by the ultrasonic sensor and $\Xi(r)$ is a Gaussian noise of zero mean and covariance matrix $V(r)$ related to the parameter r. Such noise takes into account the incidence angle influence on the measurement y_M along with the measurement noise. In order to satisfy the described characteristics about the incidence angle and its influence on the sensor measurements, the incidence function $\Xi(r)$ has to be chosen such that:

1. its standard deviation is monotonic increasing in r;
2. $V(0)$ has to be equal to the nominal covariance of the used sensor that is the covariance of the measurement noise when the incidence angle is $\gamma = \frac{\pi}{2} rad$.

If the environment is perfectly known, starting from the sensor axis equations (2.12) and using the incidence angle equation (1.5) and the incidence parameter equation (4.3), it is possible to obtain r_i for each sensor S_i. Otherwise, if no assumption on the environment is made and the environment is totally unknown, the incidence angles are not known *a-priori*. In this situation, to use the proposed sensors switching rule an estimate of the incidence angles, $\{\hat{\gamma}_i, i = 1, \ldots, p\}$, or, which is completely equivalent, of the associated incidence parameters $\{\hat{r}_i, i = 1, \ldots, p\}$, has to be found.

As explained in Section 2.6, if the environment is not known, the mobile robots localization task can be accomplished using the proposed Neighbors based Algorithm (NBA) (see Section 2.6.1); the information provided by the NBA can be used to obtain the incidence parameters estimations.

More precisely, marking the axis of the sensor S_i as $x_2 = a_i x_1 + q_i$ and ignoring the translation q_i, a vector on such line can be parametrized as

$$y_i(\lambda) = [\lambda \ , \ a_i\lambda]^T \ , \ \lambda \in \mathbb{R}$$

thus, the unit vector, $\overrightarrow{y}_i$, related to the sensor's axis is

$$\overrightarrow{y}_i = \frac{y_i(\lambda)}{||y_i(\lambda)||}$$

Using the current estimate $\hat{x}_k$, an estimation of a_i can be found as $\hat{a}_i = \tan(\hat{\theta}_k + \alpha_i)$ (see equation (2.12)). Thus an estimation of $y_i(\lambda)$ is $\hat{y}_i(\lambda) = [\lambda \ , \ \hat{a}_i\lambda]^T$ and finally the estimated unit vector is

$$\hat{\overrightarrow{y}}_i = \frac{\hat{\overrightarrow{y}}_i(\lambda)}{||\hat{\overrightarrow{y}}_i(\lambda)||} \tag{4.5}$$

The Neighbors based Algorithm provides a set of lines, $\{(\hat{s}_i, \hat{c}_i)\}$, related to an approximation of the parameters of the segments intercepted by the used sensors' axes. The line (s_i, c_i) is related to the sensor S_i and can be assumed as an estimation of the tangent line, $l_i(x_1, x_2)$, to the boundaries of the environment part intercepted by such sensor's axis. Therefore, for each used sensor S_i, an approximation of the unit vector $\overrightarrow{b}_i$ of the line ($l_i(x_1, x_2)$ in Figure 1.3) can be found as

$$\hat{\overrightarrow{b}}_i = \frac{\hat{b}_i(\lambda)}{||\hat{b}_i(\lambda)||} \tag{4.6}$$

where $\hat{b}_i(\lambda)$ is the approximation of the parametrized vector on the line $l_i(x_1, x_2)$, that is $\hat{b}_i(\lambda) = [\lambda \ , \ \hat{c}_i\lambda]^T$

At this point, using the equations (4.5) and (4.6) along with the equation (4.3), it is possible to obtain an approximation of the incidence parameter related to each sensor S_i as

$$\hat{r}_i = \hat{\overrightarrow{y}}_i \cdot \hat{\overrightarrow{b}}_i \tag{4.7}$$

Thanks to the above equation, at each step k the following switching algorithm can be performed:

Sensors Switching Algorithm

Given $(\hat{\bar{s}}_k, \hat{\bar{c}}_k) = NBA(\hat{x}_k, y_k)$, do

1. evaluate $\hat{\overrightarrow{y}}_i, i = 1, \ldots, p$, using (4.5)
2. evaluate $\hat{\overrightarrow{b}}_i, i = 1, \ldots, p$, using (4.6)
3. evaluate $\hat{r}_i = \hat{\overrightarrow{y}}_i \cdot \hat{\overrightarrow{b}}_i, i = 1, \ldots, p$, using (4.7)
4. $\mathcal{Q}_k^* = \mathcal{MIN}(\{\hat{r}_i, i = 1, \ldots, j\}, q)$

where $\hat{x}_k$ is the actual robot pose estimation, y_k contains the measurements provided by robot sensors and the function $\mathcal{MIN}$ is defined as:

Given a set of variables $\{z_i\}, i = 1, \ldots, m$, *and* $z \in \mathbb{N}$ *such that* $z \leq m$,

$$\mathcal{MIN}(\{z_i, i = 1, \ldots, m\}, z)$$

returns the indexes related to the z *variables with lower value than the others.*

Looking at the optimization problem (4.2), the proposed switching rule is related to the following cost function

$$J_s(\mathcal{Q}, k) = \sum_{S_i \in \mathcal{Q}} \hat{r}_i(S_i)$$

where $\hat{r}_i(S_i)$ is the estimation of the incidence parameter r_i related to the sensor S_i.

The first proposed switching logic is based on choosing the subset of sensors the measurements of which have maximum effect on the current robot pose estimation and this policy has a combinatorial computational cost on the number, p, of available sensors. On the contrary, the incidence angle based policy has a polynomial cost on the number, p, of sensors. Moreover, using the proposed policy along with the NEKF algorithm, the computational cost due to the use of the switching rule is very low since the parameters $(\hat{\bar{s}}_k, \hat{\bar{c}}_k)$ obtained by the filtering algorithm can be used by the switching policy too. In this situation, the additional cost due to the switching policy, in the worst case, is only related to sorting the elements $\hat{r}_i, i = 1, \ldots, p$, and it is $O(p \log p)$. This cost is less than the computational cost due to the NEKF and thus, in this case, the additional computational effort due to the use of the proposed switching rule is negligible w.r.t. the filtering algorithm.

4.4 Applications and Considerations

Depending on the desired performance and on the computational constraints, an infinite number of sensors switching rules can be stated, each one with its advantages and its drawbacks. Whatever is the chosen switching rule, all the mobile robots localization algorithms, described in Chapters 2 and 3 of this dissertation, can be suitably adapted in order to properly switch among the available sensors.

Please note that in principle q could be a time varying value, $q = q(k)$, but in this dissertation only constant q switching rules have been developed. Further studies are in progress to look for time varying q switching policies.

5

Perspective n Point using Inertial Measurement Units

In this Chapter a set of alternative mobile robots localization algorithms will be shown. The described techniques are based on the simultaneous use of vision and inertial sensors. Differently from what has been shown in Chapter 2, these techniques are based only on sensors characteristics and they do not require information about the robot model. However, as a drawback, the proposed localization methods have been developed under the assumption of working in a static context or during very slow robot movements.

5.1 Problem Description

Given a camera and an object in the field of view of the camera the goal of the present chapter is to develop suitable algorithms to find the relative position and orientation between the object and the camera. First of all, a brief introduction about the standard camera model is required.

5.1.1 Camera Pinhole Model

The main idea behind the pinhole model is to project a 3 dimensional world into a 2 dimensional plane, called image plane. The resulting projection represents the image provided by the camera. To model the camera behavior, the most common and used model in computer vision is **the central perspective imaging model**.

Let C be the camera focus and $z = f$ be the distance from C to the image plane. The rays converge on the origin of the camera frame C and a non-inverted image is projected onto the image plane. As shown in Figure 5.1, using similar triangles, given a point at the world coordinates $P = (X, Y, Z)$, this point is projected into the image plane point $p = (x, y)$ by

$$x = f\frac{X}{Z},\ y = f\frac{Y}{Z}$$

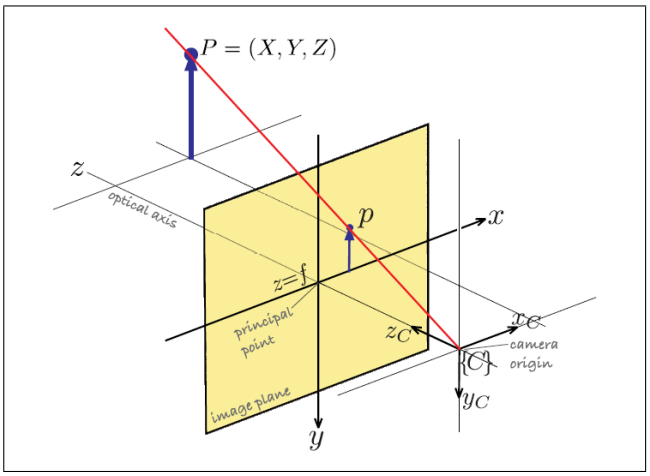


Fig. 5.1: Pinhole model example

The above equations represent a projective transformation, or more specifically a perspective projection, from the world to the image plane. The perspective projection has the following characteristics:

- It performs a mapping from 3-dimensional space to the 2-dimensional image plane: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$.
- Straight lines in the world are projected to straight lines on the image plane.
- Parallel lines in the world are projected to lines that intersect at a vanishing point; in drawing, this effect is known as foreshortening. Lines in the plane parallel to the image plane represent an exception since they do not converge.
- Conics in the world are projected to conics on the image plane. For example, a circle is projected as a circle or an ellipse.
- The mapping is not one-to-one and a unique inverse does not exist. That is, given the image point $p = (x, y)$, it is not possible to uniquely determine the world point $P = (X, Y, Z)$ which has been projected into p. All that can be said is that the world point lies somewhere along the projecting ray through the camera focus C and the image point p, as shown in Figure 5.1.
- The transformation is not conformal: it does not preserve shape since internal angles are not preserved.

Due to the above properties, if no information about the object is provided, using only one camera, it is not possible to infer information about the relative position and orientation between the camera and an object in the field of view of the camera. As described in 1.3.3, the process of image formation, in an eye or in a camera, involves a projection of the 3-dimensional world onto a

2-dimensional surface. The depth information is lost and starting from the image it is not possible to tell whether it is of a large object in the distance or a smaller closer object. To overcome this problem a first solution is to use two cameras instead of a single one and to consequently find the relative position and orientation between the camera and the object by solving a stere ovision problem (see [72] for more details about stereo vision).

However, as it will be shown in the next Sections, if at least some information about the object is known, the proposed localization problem may be solved using a single camera. More precisely, in the following the perspective n point (PnP) will be discussed. A new approach to solve this problem will be proposed by means of **inertial measurement units** used to help the vision system solve the PnP.

5.2 PnP History

In the 80's Fischler and Bolles [48] introduced for the first time the **Perspective-n-Point (PnP) problem**, also known as pose estimation. They summirize the problem as follows:

> *Given the relative spatial locations of n 'control points' (landmarks), and given the angle to every pair of control points from an additional point called the Center of Perspective (C_P), find the lengths of the line segments joining C_P to each of the control points.*

In other words, the problem is that of determining the relative position and orientation of an object with respect to a camera by exploiting the image provided by the camera and the knowledge of a feature of n points placed on the object (see Figure 5.5). This problem finds application in several fields, such as computer vision [58], computer animation [57], photogrammetry [59] and robotics [60] [61] [62].

5.2.1 Classical *PnP* Problem

Several solutions to the PnP problem have been proposed in the literature. From the theoretical viewpoint, it has been proved that the smallest number of points which yield to a finite number of solutions for this problem is $n = 3$ [48], since the P2P problem ($n = 2$), in its classical formulation, admits infinite solutions. Moreover, as proved in [53], the smallest number of feature points ensuring a unique solution to the PnP problem in all possible object configurations is $n = 4$, under the assumption they are coplanar and no more than two of them lie on a single line. In [49] a complete analysis of the P3P problem is provided. There, the authors show that this problem has at most four solutions and, in some situations, it can have a unique solution. In [50] a survey of the main methods to approach the P3P problem is provided.

The P4P problem has been faced in many ways. Fischer and Bolles [48] introduced the RANSAC algorithm, which faces the problem by first solving the P3P problem for any groups of 3 points of the feature and then making the intersection of their solutions. Rivers et al. [51] propose an approach based on the solutions of a set of six quadratic equations with four unknowns.

Even if using $n = 4$ correspondences between feature points and pixels is sufficient to obtain a unique solution for the PnP problem, a larger set of points, say $n > 4$, makes the solution more robust with respect to the measurement noise. For this reason, many works in the literature focus on finding algorithms to solve the generic PnP problem for $n > 4$. In [52], authors consider triplets of points among the n available correspondences, and for each of them derive fourth degree polynomials in the unknowns of the problem. Then, they rearrange such polynomials in a matrix form and use the singular value decomposition to estimate the unknown values and solve the PnP problem.

In [70], authors propose a non-iterative solution to the PnP problem, the computational complexity of which grows linearly with n. This method is applicable for $n \geq 4$ and handles properly both planar and non-planar configurations. The main idea behind this method is to express the n feature points as a weighted sum of four virtual control points in the camera reference frame. The control points can be estimated in $O(n)$ time by expressing their coordinates as weighted sum of the eigenvectors of a 12×12 matrix and solving a small number of quadratic equations to pick the right weights.

The aforementioned methods are classified as non iterative methods as they yield to the solution of the PnP problem in a closed form, without generating a sequence of improving sub-optimal estimates as in the case of the iterative methods. Among the iterative methods, it is relevant to cite the one proposed in [69], which is one of the fastest and most accurate. This method is based on the minimization of an error index in the 3D space, optimizing alternatively on the relative position and orientation unknowns. This algorithm has a very fast convergence time, but it can get stuck in local minima, depending on its initial guess.

In the classical PnP problem the only available information is the one provided by the camera and the feature. However, in many cases, such as in robotics applications, additional sensors are available and might provide further useful information to enhance the pose estimation. For instance, mobile robots and cameras are often equipped with **Inertial Measurement Units (IMUs)** that, in the static configuration, are able to measure the gravity vector in their own reference frame through accelerometers, and the magnetic field by means of magnetometers.

5.2.2 Inertial Measurement Units (IMUs)

An **I**nertial **M**easurement **U**nit (Figure 5.2) (IMU), is an electronic device that measures the velocity, orientation, and gravitational forces applied on

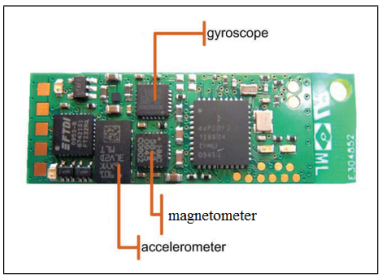


Fig. 5.2: Inertial Measurement Unit example

the sensor itself. These measurements are obtained using a combination of accelerometers, gyroscopes and magnetometers.
The IMUs are the main components of inertial navigation systems used in aircrafts, mobile robots, aerial robots, spacecrafts, watercrafts and guided missiles. These sensors are typically used, for example, to maneuver unmanned aerial vehicles (UAVs), satellites and landers.
An IMU works by detecting the current rate of acceleration using one or more accelerometers, and detects changes in rotational attributes like pitch, roll and yaw, using one or more gyroscopes and one magnetometer. Therefore using an IMU it is possible to obtain information on the **roll-pitch-yaw (RPY)** angles related to the sensor orientation.
Assume to have an IMU placed on an object as shown in Figure 5.4(a) and let $O_{X',Y',Z'}$ be the IMU reference frame and $O_{X,Y,Z}$ be the absolute reference frame (North-East-Down reference Frame[1], see Figure 5.3).
The sensor detects to two main accelerations: an acceleration $\overrightarrow{a}$ due to input forces and an acceleration $\overrightarrow{g}$ due to earth's gravitational field, as shown in Figure 5.4(c). The IMU provides a measurement related to the total object acceleration, $\overrightarrow{a} + \overrightarrow{g}$, the coordinates of which are in the IMU reference frame. Please note that if the object is static then $\overrightarrow{a} = \overrightarrow{0}$ and the only acceleration on the object is the gravitational acceleration $\overrightarrow{g}$. Therefore, in a static context, an IMU directly provides measurements about the gravity vector $\overrightarrow{g}$. Moreover if the object speed variations are slow enough, that is $\overrightarrow{a} \approx 0$, the measurements provided by the IMUs can be seen as an approximation of the gravitational acceleration $\overrightarrow{g}$.

[1] The **N**orth-**E**ast-**D**own (NED) reference frame (see Figure 5.3, also known as local tangent plane (LTP), is a geographical coordinate frame, for representing state vectors, that is commonly used in aviation. It consists of three numbers: one represents the position along the northern axis, one along the eastern axis, and one represents vertical position. Down is chosen as opposed to up in order to comply with the right-hand rule and it represents the gravity direction.

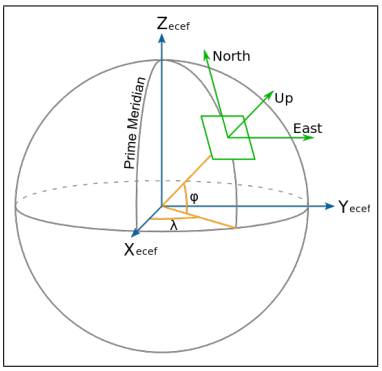
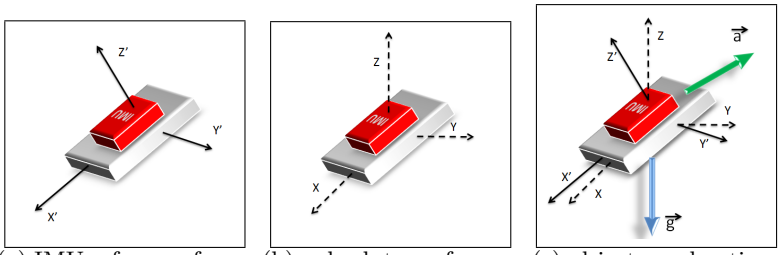


Fig. 5.3: North East Down reference frame



(a) IMU reference frame (b) absolute reference frame (c) object accelerations

Fig. 5.4: IMU, reference frames and object accelerations

5.2.3 *PnP* Problem and IMUs

In most of the current applications (see, *e.g.*, [55, 56]), information coming from the camera system and from the other sensors are gathered separately and then fused *a posteriori*. For instance, as for the dynamic case, in [55] authors present an extended Kalman filter for precisely determining the unknown transformation between a camera and an IMU. In [56], a tight coupling of IMU and camera is achieved by an error-state extended Kalman filter which uses each visually tracked feature contribute as an individual measurement for its update.

The aim of this dissertation is to use the data elaborated from the IMUs to 'help' the vision system solve the PnP problem. In particular, the problem will be faced in a static configuration where, due to the high resolution of the IMUs' accelerometers, the proposed approach is expected not only to simplify the solution of the PnP problem, but also to yield to a more accurate pose estimation.

Only a few recent works using this approach have been presented in the literature. In [54], the authors use the roll, pitch and yaw angles provided by

the IMU to compute the translation vector between the feature and a fixed camera with known position and orientation. In the same paper it is also shown that, if the whole object attitude is known, the P2P problem admits a unique solution whenever the two points on the image are distinct. However, it is worth mentioning that the results obtained in this paper, although interesting from the scientific viewpoint, are quite fragile w.r.t. pixel noise. Moreover, the information about the attitude is computed by the IMU using both the accelerometers and the compass, the latter of which, as already remarked, can be quite noisy and unreliable. In [63], the authors assume to know the coordinates of two points in the absolute reference frame and, using the roll and pitch provided by an IMU mounted on the camera, they solve a P2P problem on reconstructing the absolute camera pose. The yaw angle provided by the IMU is discarded due to the low accuracy of the compass.

In the present dissertation a more general scenario is considered. No assumption on the absolute coordinates of either the camera or the object is made and both the observed object and the camera are assumed to be equipped with IMUs. Several results obtained for this configuration will be presented, both in the case 1) the information of the magnetometer is used or 2) only inclinometers are considered.

In particular, first a closed-form solution for the P2P Problem with IMUs (i.e. using both inclinometers and magnetometers) will be proposed. This solution considerably improves the robustness to pixel noise w.r.t. the algorithm proposed in [54]. Then, the above solution will be extended to the general PnP Problem with IMUs.

Moreover, a way to solve the P2P problem using only the information given by the inclinometers will be provided. It will be proved that such a modified P2P problem always gives 2 solutions, except when the 2 points are seen as one by the camera (in this case, infinite solutions can be found). To solve the ambiguity between the possible two solutions of the P2P, a geometrical test is given, that allows to obtain a unique solution for the P2P problem in roughly one half of the cases. Moreover, it will be remarked that singular configurations and ambiguities can be avoided by using a feature of 3 non-collinear points.

Finally, the PnP problem using only inclinometers will be studied. In particular, the PnP problem will be reformulated as an optimization problem where the objective function is a least mean square index. This choice allows to solve the PnP problem by solving a single variable optimization problem.

5.3 Problem statement

Assume a camera and an object in the field of view of the camera. Two reference frames are defined: the camera reference frame O_{xyz}, the origin of which is in the camera focus, and the object reference frame O'_{uvw}. It is assumed that the object is provided with a feature composed of n points, the coordinates of which are known "a priori" in the object reference frame. The object

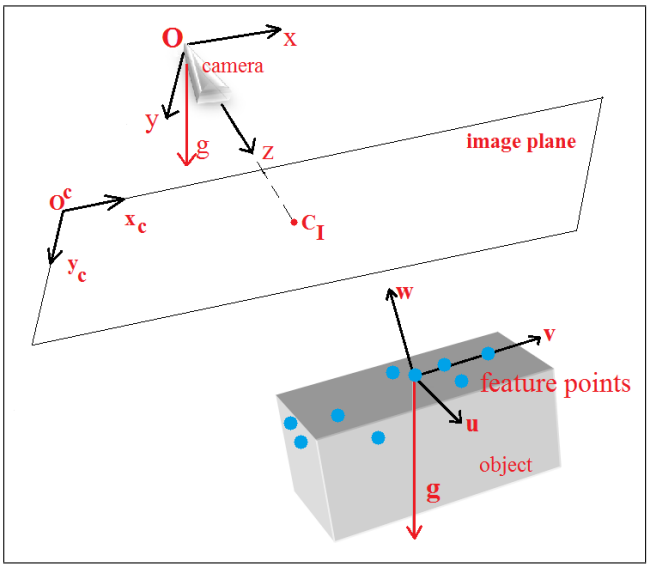


Fig. 5.5: Camera, image and object reference frames

and the camera are equipped with an IMU capable of measuring the gravity unit vector[2] and the earth magnetic field unit vector: $\hat{g}_o = [g_u, g_v, g_w]$, $\hat{m}_o = [m_u, m_v, m_w]$ in the object reference frame, and $\hat{g}_c = [g_x, g_y, g_z]$, $\hat{m}_c = [m_x, m_y, m_z]$ in the camera reference frame, respectively. It is assumed that the distance between the camera and the object is negligible w.r.t. the earth radius, hence $\hat{g}_c, \hat{m}_c$ and $\hat{g}_o, \hat{m}_o$ represent the same couple of vectors in two different coordinate frames. Furthermore, the camera is assumed to have no distortion and to be characterized by the focal length f and the distance per pixel dpx.

The image reference frame is denoted by $O^c_{x_c y_c}$, the image plane is $z = f$ and the image center is $C_I = (x_{C_I}, y_{C_I})$. The overall scenario is depicted in Figure 5.5.

Goal of this dissertation is to solve the PnP problem using information provided by both the IMUs and the camera. The aim is to obtain the transformation matrix from the object reference frame to the camera reference frame:

$$R_t = \left[\begin{array}{ccc|c} & R & & t \\ 0 & 0 & 0 & 1 \end{array}\right],$$

where R is the relative rotation matrix and $t = [t_x, t_y, t_z]^T$ is the translation vector.

[2] Clearly, using an IMU also the magnitude of the gravity and earth magnetic field vectors is measured. However, for the purposes of this work only the vectors' directions are of interest.

5.4 PnP Problem using IMUs: proposed solutions

Using the information provided by the IMUs, the whole rotation matrix from the object reference frame to the camera reference frame can be estimated. In this context, solving the PnP problem translates into determining the translation vector t between the two reference frames. In the following, a new solution to the P2P problem with IMUs will be presented. Aftewards, this solution will be extended to the more general PnP problem with IMUs.

5.4.1 P2P Problem with known Rotation Matrix

The P2P problem is the problem of estimating the relative pose of the object in the camera reference frame when the object contains a feature of two distinct points $A = (A_u, A_v, A_w)$ and $B = (B_u, B_v, B_w)$, the coordinates of which are known in the object reference frame. The information provided by the camera can be modeled using the pinhole model [72] as follows. Let $K = (K_u, K_v, K_w)$ be a point in the object reference frame. Using the transformation matrix R_t, the coordinates of K in the camera reference frame can be written as

$$P_K = [K_x, K_y, K_z]^T = RK + t, \tag{5.1}$$

and the coordinates of pixel P_K^* related to P_K in the image are defined as

$$x_K = \tilde{f}\,\frac{K_x}{K_z} + x_{C_I},\; y_K = \tilde{f}\,\frac{K_y}{K_z} + y_{C_I}, \tag{5.2}$$

where $\tilde{f} = f/dpx$ is the camera focal length measured in pixels.

Using the above equations it is possible to compute the pixels $P_A^* = (x_A, y_A)$ and $P_B^* = (x_B, y_B)$, where points A and B are projected in the image plane. On defining $\tilde{P}_A = P_A^*/\tilde{f}$, $\tilde{P}_B = P_B^*/\tilde{f}$, from equations (5.2) and (5.1) it is possible to obtain

$$\begin{aligned}
\tilde{P}_A = (\tilde{x}_A, \tilde{y}_A) &= \left(\frac{r_1 A + t_x}{r_3 A + t_z}, \frac{r_2 A + t_y}{r_3 A + t_z}\right), \\
\tilde{P}_B = (\tilde{x}_B, \tilde{y}_B) &= \left(\frac{r_1 B + t_x}{r_3 B + t_z}, \frac{r_2 B + t_y}{r_3 B + t_z}\right),
\end{aligned} \tag{5.3}$$

where r_i is the i-th row of matrix R.

Since all the information from the IMUs is used, the whole rotation matrix R is known. As a consequence to solve the P2P problem simplifies into the problem of estimating the translation vector t. A possible way to find t is to use the results presented in [54]. There, the authors solve equations (5.3) with respect to t_x, t_y, t_z, obtaining the following solution:

$$\begin{aligned}
t_x &= A_z \tilde{x}_A - r_1 A \\
t_y &= A_z \tilde{y}_A - r_2 A \\
t_z &= A_z - r_3 A
\end{aligned}, \tag{5.4}$$

where A_z is computed as follows:

$$\textbf{if } \tilde{x}_B \neq \tilde{x}_A \textbf{ then } A_z = \frac{(r_1 - \tilde{x}_B r_3)\,(A - B)}{\tilde{x}_A - \tilde{x}_B}$$
$$\textbf{otherwise } A_z = \frac{(r_2 - \tilde{y}_B r_3)(A - B)}{\tilde{y}_A - \tilde{y}_B}.$$

Although the above equations represent a formally correct solution, they are very sensitive to pixel noise. To mitigate this problem, a possible solution is to compute the translation vector $\hat{t}$ which minimizes the following index

$$\mathbb{E} = ||\hat{P}_A(\ \hat{t}\ ) - \tilde{P}_A||^2 + ||\hat{P}_B(\ \hat{t}\ ) - \tilde{P}_B||^2 \tag{5.5}$$

where $\hat{P}_A(\ \hat{t}\ )$ and $\hat{P}_B(\ \hat{t}\ )$ are the pixels in which points A and B are projected using the rotation matrix provided by the IMUs, the translation vector $\hat{t}$ and equation (5.2).

The above index represents the reprojection error due to the use of the rotation matrix R and the translation vector $\hat{t}$. This index is widely used in the literature to test the performance of PnP solutions (see [54], [63], [70]). It is important to highlight that the resulting optimization problem must be solved through numerical methods and may require a high computational effort. To overcome this difficulty, in the present thesis an alternative least means square index is proposed. The starting point is that equations (5.3) can be rewritten as the following set of linear equations

$$\begin{array}{rcl} -t_x + t_z\tilde{x}_A &=& r_1 A - r_3 A\tilde{x}_A \\ -t_y + t_z\tilde{y}_A &=& r_2 A - r_3 A\tilde{y}_A \\ -t_x + t_z\tilde{x}_B &=& r_1 B - r_3 B\tilde{x}_B \\ -t_y + t_z\tilde{y}_B &=& r_2 B - r_3 B\tilde{y}_B \end{array}, \tag{5.6}$$

which can be written in matrix form as follows

$$Q\,t = s, \tag{5.7}$$

where

$$Q = \begin{bmatrix} -1 & 0 & \tilde{x}_A \\ 0 & -1 & \tilde{y}_A \\ -1 & 0 & \tilde{x}_B \\ 0 & -1 & \tilde{y}_B \end{bmatrix}; \quad s = \begin{bmatrix} r_1 A - r_3 A\tilde{x}_A \\ r_2 A - r_3 A\tilde{y}_A \\ r_1 B - r_3 B\tilde{x}_B \\ r_2 B - r_3 B\tilde{y}_B \end{bmatrix}. \tag{5.8}$$

An index alternative to (5.5) is defined as

$$\mathbb{E}_2 = ||Q\,t - s||^2, \tag{5.9}$$

which is the least mean square index over equations (5.6). The main advantage of using the index (5.9) rather than (5.5) is that, as well known, the optimal

solution of this LMS problem can be computed by using the pseudo-inverse matrix of Q as follows:

$$\hat{t} = Q^{\dagger} s. \tag{5.10}$$

where $\hat{t}$ is the optimal LMS approximation of the translation vector t. Clearly, this solution is more robust with respect to pixel noise than the one provided by (5.4).

To verify the robustness of the proposed solution with respect to pixel noise, a set of 10 000 numerical tests have been performed, using a feature of $n = 2$ points placed in $A = [0, 0, 0]^T m$ and $B = [0.1, 0.1, 0]^T m$ in the object reference frame, and assuming a zero mean Gaussian noise with standard deviation of $\sigma = 5$ pixels on the pixels provided by the camera. For each test, the actual transformation matrix has been chosen randomly. The estimation of the translation vector computed using the expression (5.10) here proposed has been contrasted with the one proposed in [54] by means of the relative error

$$\varepsilon_t = \frac{||t - \hat{t}||}{||t||}.$$

Figure 5.6 shows the results of some of the performed tests.

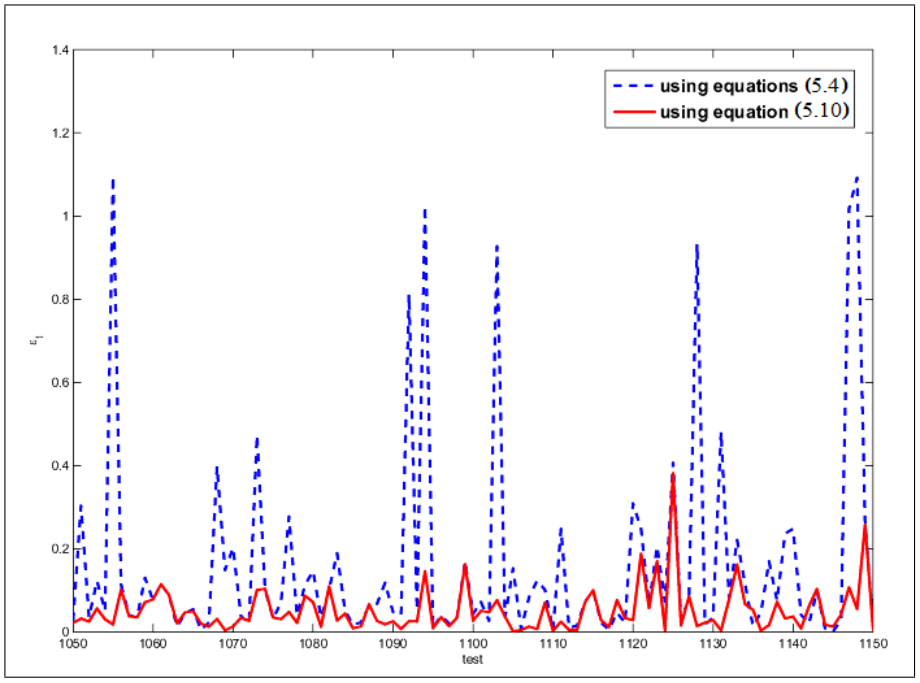


Fig. 5.6: Index ε_t in some of the performed tests

As expected, using the pseudo-inverse, the estimation error ε_t is generically smaller than the one obtained using equations (5.4) of [54], and provides an average value of 0.0691 (7% error) against an average value of 0.5029 (50% error) for the estimation method proposed in [54].

Remark 5.1. **1** Interestingly enough, the index $\mathbb{E}$ and the index $\mathbb{E}_2$ are related as $\mathbb{E} = ||M||^2\mathbb{E}_2$, where

$$M = \begin{bmatrix} \frac{1}{r_3 A + t_z} & 0 & 0 & 0 \\ 0 & \frac{1}{r_3 A + t_z} & 0 & 0 \\ 0 & 0 & \frac{1}{r_3 B + t_z} & 0 \\ 0 & 0 & 0 & \frac{1}{r_3 B + t_z} \end{bmatrix}.$$

For this reason, and under appropriate assumptions (*e.g.*, $t_z \gg r_3(A-B)$), the optimal value for the index $\mathbb{E}_2$ can be taken as a good approximation of the optimal value for the index $\mathbb{E}$.

Remark 5.2. **2** The index $\mathbb{E}$ is the most used in the literature when it comes to pose estimation, it is worth to compare it with $\mathbb{E}_2$, Figure 5.7 shows the ε_t index, in some of the performed 10 000 tests, computed using the translation vector obtained by equation (5.10) and the translation vector resulting from the index $\mathbb{E}$ minimization.

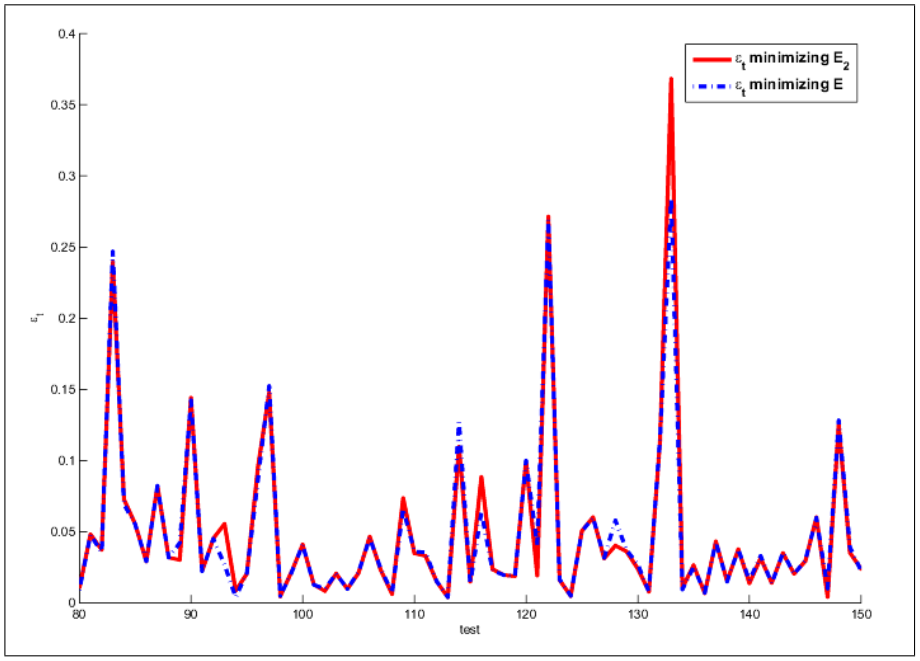


Fig. 5.7: Index ε_t minimizing $\mathbb{E}$ and minimizing $\mathbb{E}_2$

Results show no evidence of any real estimation advantages in using $\mathbb{E}$ and $\mathbb{E}_2$. Conversely, as shown in Figure 5.8, the use of $\mathbb{E}_2$ clearly outperforms the use of $\mathbb{E}$ in terms of computational time.

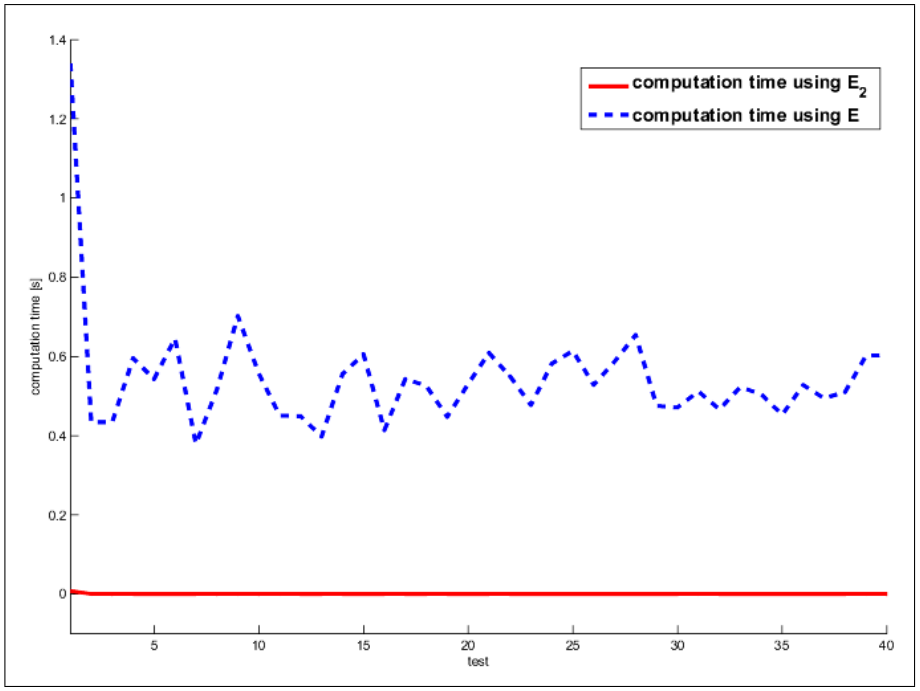


Fig. 5.8: Computation times required to minimize w.r.t. $\mathbb{E}$ or w.r.t. $\mathbb{E}_2$

The averaged computation time[3], over the performed 10 000 tests, required to solve the P2P problem is $0.0021s$ and $0.5563s$ minimizing w.r.t. $\mathbb{E}_2$ and $\mathbb{E}$ respectively.

5.4.2 PnP Problem with known Rotation Matrix

In this section the results obtained for the P2P problem will be extended to the PnP problem in the general case $n \geq 3$. To avoid pathological situations where all the points of the feature $P_i = (P_{u,i}, P_{v,i}, P_{w,i}), i = 1, \ldots, n$ are seen by the camera as a single pixel, it will be hereafter assumed that at least three points of the feature are not collinear.

As in the P2P case, if the rotation matrix R between the object and the camera reference frames is provided by IMUs, the PnP problem can be

[3] The computation times have been obtained using Matlab R2012 running on an Intel(R) Core(Tm) i7 CPU Q720 processor.

reformulated as a set of linear equations and LMS solution can be found using the pseudo-inverse method.

More formally, given a set of n correspondences between feature points and pixels, the Q matrix and the s vector will contain $2n$ rows, two rows for each correspondence. Given a generic pixel $\tilde{P}_i = (\tilde{x}_i, \tilde{y}_i)$ and the related point $P_i = (P_{u,i}, P_{v,i}, P_{w,i})$ in the object reference frame, the Q matrix and the s vector will contain the following two rows, respectively:

$$\begin{aligned} Q_{i,1} &= [-1,\ 0, \tilde{x}_i]\ s_{i,1} = r_1 P_i - r_3 P_i \tilde{x}_i, \\ Q_{i,2} &= [0,\ -1, \tilde{y}_i]\ s_{i,2} = r_2 P_i - r_3 P_i \tilde{y}_i, \end{aligned}$$

thus the Q matrix and the s vector will have the following form:

$$Q = \begin{bmatrix} Q_{1,1} \\ Q_{1,2} \\ \vdots \\ Q_{n,1} \\ Q_{n,2} \end{bmatrix}, \quad s = \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ \vdots \\ s_{n,1} \\ s_{n,2} \end{bmatrix}. \tag{5.11}$$

To obtain the PnP problem solution, the pseudo-inverse method can be used:

$$\hat{t} = Q^{\dagger} s.$$

Remark 5.3. Please note that the dimension of the matrix Q grows only linearly with respect to the number n of points. Moreover, note that in the very common case where the features points are known a priori, the pseudo-inverse matrix $Q^{\dagger}$ can be pre-computed offline in a symbolical form, translating the PnP problem in a mere product between a matrix and a vector.

5.5 PnP Problem using accelerometers only

In the previous Section, the rotation matrix provided by the IMUs is computed using both the magnetometers and the accelerometers. However, due to the low reliability of the compass, the information about the magnetic field can be quite noisy and can yield to unreliable attitude values. In these cases, a possible choice is to neglect the measurements provided by the magnetometer and use only the inclinometers. To this end, the first step will be to parametrize the transformation matrix R_t with respect to the neglected data. Then, using this new transformation matrix, a method to tackle the PnP problem will be detailed.

5.5.1 Rotation matrix parametrization

In a static context, accelerometers measure the gravity vector in their reference frames. To obtain the parametrized transformation matrix using only this

information and neglecting the magnetometers measurements, the first step is to find the parametrization of the rotation matrix R w.r.t. the missing data.

The idea is the following: let a certain reference frame h. If the rotation matrix R_c^h from this reference frame to the camera reference frame, and the rotation matrix R_o^h from the h reference frame to the object reference frame were known, then the rotation matrix R would be

$$R = R_c^o = R_c^h [R_o^h]^T. \tag{5.12}$$

The above relationship is true whatever the h reference frame is.

As well known, to obtain the rotation matrix from a reference frame to another, the canonical basis of the first frame has to be represented in the second one, assuming no translation. Since information on the gravity unit vector in the object and in the camera reference frames are available in the IMUs, it may be convenient to define an "intermediate" reference frame h defined as an artificial NED reference frame where the Down-vector is the gravity vector and the North-vector (and consequently the East-vector) are chosen in an arbitrary way. Using this artificial NED reference frame, the rotation matrices R_c^h and R_o^h can be found. Matrix R will be finally obtained by equation (5.12).

The first column of R_c^h is the unit vector $\hat{g}_c$. To complete this matrix an artificial North unit vector $\hat{m}_c$ in the camera reference frame is arbitrarily chosen. This unit vector must lie on the plane orthogonal to the gravity unit vector $\hat{g}_c$.

For simplicity the following unit vector is here chosen[4]:

$$\hat{m}_c = \left[\frac{g_z}{\sqrt{g_x^2 + g_z^2}} \ , \ 0 \ , \ \frac{-g_x}{\sqrt{g_x^2 + g_z^2}} \right]^T , \tag{5.13}$$

which will be the second column of R_c^h. The third column of R_c^h will be

$$\hat{n}_c = \hat{g}_c \times \hat{m}_c = \left[-\frac{g_x g_y}{\sqrt{g_x^2 + g_z^2}} \ , \ \sqrt{g_x^2 + g_z^2} \ , \ -\frac{g_y g_z}{\sqrt{g_x^2 + g_z^2}} \right]^T , \tag{5.14}$$

therefore the rotation matrix from the artificial NED reference frame to the camera reference frame is

$$R_c^h = [\hat{g}_c \ \ \hat{m}_c \ \ \hat{n}_c] \, . \tag{5.15}$$

Next step is to build the rotation matrix R_o^h from the object reference frame to the h reference frame. Following the same lines of what done with R_c^h, the first column of R_o^h is $\hat{g}_o$. The second column should contain $\hat{m}_o$, that is the artificial North unit vector in the object reference frame. However, the

[4] Please note that this choice is completely arbitrary. If $g_x = g_z = 0$, which implies that the gravity unit vector lies on the Y-axis, it is always possible to choose another unit vector orthogonal to $\hat{g}_c$.

coordinates of this vector in the object reference frame are unknown. The only information available is that it has to lie on the plane orthogonal to $\hat{g}_o$ and it must be a unitary vector. As a consequence, from the geometrical point of view, this vector will lie on the intersection between the plane orthogonal to $\hat{g}_o$ and a sphere with unit ray, as shown in Figure 5.9.

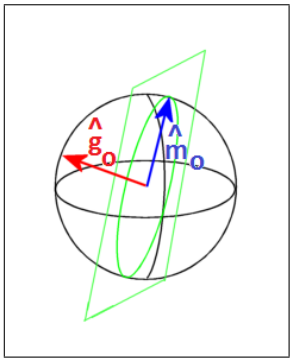


Fig. 5.9: Unit vectors $\hat{g}_o$ and $\hat{m}_o$

This allows to introduce the following parametrization for $\hat{m}_o$

$$\hat{m}_o = \hat{m}_o(\alpha) = \hat{m}_1 \sin\alpha + \hat{m}_2 \cos\alpha, \tag{5.16}$$

where $\{\hat{m}_1, \hat{m}_2\}$ is an orthonormal basis for the plane orthogonal to $\hat{g}_o$ and α is an unknown angle characterizing the artificial North vector orientation. For simplicity $\hat{m}_1$ has been chosen as[5]

$$\hat{m}_1 = \left[\frac{g_w}{\sqrt{g_u^2 + g_w^2}},\ 0,\ \frac{-g_u}{\sqrt{g_u^2 + g_w^2}}\right]^T$$

and thus $\hat{m}_2$ as

$$\hat{m}_2 = \hat{g}_o \times \hat{m}_1 = \left[-\frac{g_u g_v}{\sqrt{g_u^2 + g_w^2}},\ \sqrt{g_u^2 + g_w^2},\ -\frac{g_v g_w}{\sqrt{g_u^2 + g_w^2}}\right]^T.$$

The third vector $\hat{n}_o(\alpha)$ is

$$\hat{n}_o(\alpha) = \hat{g}_o \times \hat{m}_o(\alpha) =$$
$$\left[-\frac{g_w \cos\alpha + g_u g_v \sin\alpha}{\sqrt{g_u^2 + g_w^2}}, \sqrt{g_u^2 + g_w^2}\sin\alpha, \frac{g_u \cos\alpha - g_v g_w \sin\alpha}{\sqrt{g_u^2 + g_w^2}}\right]^T. \tag{5.17}$$

Finally, the rotation matrix from the artificial NED reference frame to the object reference frame is

[5] As in the equation (5.13), if $g_u = g_w = 0$, another unit vector orthogonal to $\hat{g}_o$ can be chosen, obtaining similar results.

$$R_o^h(\alpha) = [\hat{g}_o \;\; \hat{m}_o(\alpha) \;\; \hat{n}_o(\alpha)]\,. \tag{5.18}$$

Using $R_o^h(\alpha)$ and R_c^h, the rotation matrix $R_c^o = R_c^o(\alpha)$ is

$$R_c^o = R_c^o(\alpha) = R_c^h[R_o^h(\alpha)]^T = R(\alpha) \tag{5.19}$$

and the whole transformation matrix will be a function of α and t

$$R_t(\alpha, t) = \left[\begin{array}{cc|c} & R(\alpha) & t \\ 0\;0 & 0 & 1 \end{array}\right], \tag{5.20}$$

where α is one of the unknowns of the problem.

5.5.2 P2P using accelerometers only

If the parametrized matrix $R(\alpha)$ is used instead of R in (5.3), a system with four unknowns: t_x, t_y, t_z, α, is obtained. The P2P problem can be solved by computing first the value of α and then obtaining the rotation matrix $R(\alpha)$ using (5.19); finally, t can be determined using (5.10).

For what regards the computation of the α angle, under the assumption $\Delta_y = \tilde{y}_A - \tilde{y}_B \neq 0$ and $\Delta_x = \tilde{x}_A - \tilde{x}_B \neq 0$, using equations (5.3) the following equation can be found:

$$a \sin\alpha + b \cos\alpha + c = 0 \tag{5.21}$$

where

$$\begin{array}{l} a = a(A, B, \tilde{x}_A, \tilde{y}_A, \tilde{x}_B, \tilde{y}_B, \hat{g}_c, \hat{g}_o), \\ b = b(A, B, \tilde{x}_A, \tilde{y}_A, \tilde{x}_B, \tilde{y}_B, \hat{g}_c, \hat{g}_o), \\ c = c(A, B, \tilde{x}_A, \tilde{y}_A, \tilde{x}_B, \tilde{y}_B, \hat{g}_c, \hat{g}_o) \end{array}$$

are scalar constants that can be computed in closed form on the basis of the measured data (the details can be found in the chapter appendix). Equation (5.21) can be solved using the relation

$$a \sin\alpha + b \cos\alpha = M \cos(\alpha - \beta),$$

where

$$M = \sqrt{a^2 + b^2},\;\; \beta = \arctan\left(\frac{a}{b}\right).$$

Finally:

$$\alpha = \arccos\left(-\frac{c}{M}\right) + \beta. \tag{5.22}$$

Note that equation (5.22) has two solutions, denoted hereafter as $\alpha_i, i = 1, 2$.

Once the α angle and therefore the $R(\alpha)$ matrix are computed, the translation vector t can be obtained using equation (5.10). Given a rotation matrix R, this equation has a unique solution for the associated translation vector t. Then, since there are two possible values for α, *i.e.*, α_1 and α_2, there are also

two possible rotation matrices, R_1 and R_2, respectively, and for each of them, a translation vector, t_1 and t_2.

The couples (R_1, t_1) and (R_2, t_2) are both possible solutions for the P2P problem in the case $\Delta_y \neq 0$ and $\Delta_x \neq 0$.

There are three pixel configurations in which the assumption $\Delta_y \neq 0$ and $\Delta_x \neq 0$ does not hold, that have to be analyzed separately:

1. $\Delta_x = 0$ and $\Delta_y \neq 0$
2. $\Delta_x \neq 0$ and $\Delta_y = 0$
3. $\Delta_x = 0$ and $\Delta_y = 0$

The first configuration may be solved using the first and the third equation in (5.3), which for $\tilde{x}_A = \tilde{x}_B$ allows us to obtain

$$a_1 \sin\alpha + b_1 \cos\alpha + c_1 = 0, \tag{5.23}$$

where

$$\begin{aligned} a_1 &= a_1(A, B, \tilde{x}_A, \hat{g}_c, \hat{g}_o), \\ b_1 &= b_1(A, B, \tilde{x}_A, \hat{g}_c, \hat{g}_o), \\ c_1 &= c_1(A, B, \tilde{x}_A, \hat{g}_c, \hat{g}_o), \end{aligned}$$

are scalar constants that can be computed in closed form (see the chapter appendix).

Similarly, the second configuration may be faced using the second and the fourth equation of (5.3), that for $\tilde{y}_A = \tilde{y}_B$ give

$$a_2 \sin\alpha + b_2 \cos\alpha + c_2 = 0, \tag{5.24}$$

where

$$\begin{aligned} a_2 &= a_2(A, B, \tilde{y}_A, \hat{g}_c, \hat{g}_o), \\ b_2 &= b_2(A, B, \tilde{y}_A, \hat{g}_c, \hat{g}_o), \\ c_2 &= c_2(A, B, \tilde{y}_A, \hat{g}_c, \hat{g}_o), \end{aligned}$$

are scalar constants that can be computed in closed form (see the chapter appendix).

In these two configurations there are two possible solutions for α and then two possible solutions for the P2P problem.

The third pixel configuration is shown in Figure 5.10, and it corresponds to the case where the two feature points and the origin of the camera reference frame lie on the same line. Using this information the line unit vector can be inferred. Depending on the verse of this unit vector there are two possible values for α. However, if it is assumed to be able to distinguish if the pixel seen in the image is $\tilde{P}_A$ or $\tilde{P}_B$ (for example using different colors for the feature points), the correct unit vector can be found. At this point, if the line vector is not aligned with the gravity vector, it is possible to infer the α angle using information on the two unit vectors. As a consequence, there is only one solution for the rotation matrix. Otherwise, if the line and the gravity vectors are aligned, as shown in Figure 5.11, then the information on the line vector is

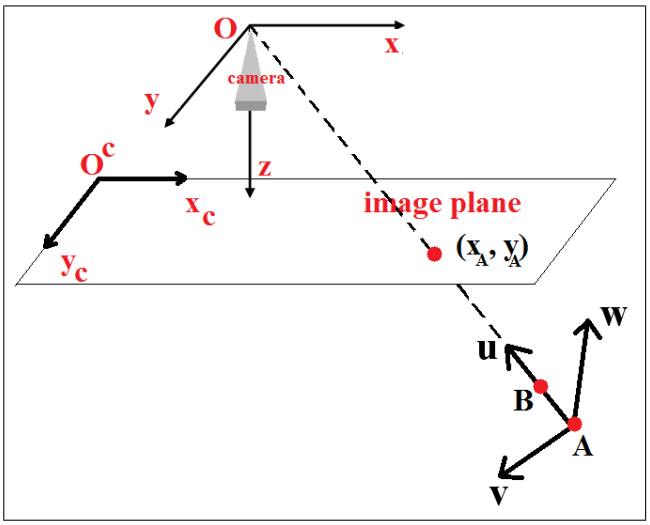


Fig. 5.10: P2P, $\Delta_x = 0$ and $\Delta_y = 0$

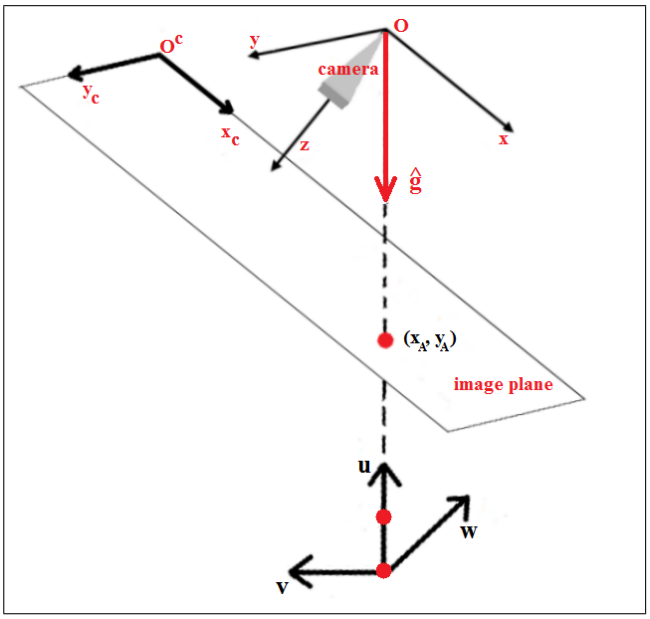


Fig. 5.11: P2P, $\Delta_x = 0$ and $\Delta_y = 0$: line unit vector aligned with gravity unit vector

the same provided by the IMUs and it is not possible to find a unique solution for the orientation.

In this configuration there are ∞^1 possible solutions for the rotation matrix. However, it should be noticed that the case $\Delta_x = 0$ and $\Delta_y = 0$ is a singular configuration where the two pixels $\tilde{P}_A$ and $\tilde{P}_B$ are the same. Since the camera sees a single point, no information on the translation vector can be obtained. As there is no possibility to infer the distance between the two reference frames, there are ∞^1 solutions for the translation.

By the above analysis the following lemma can be stated:

Lemma 1 *Using the image provided by the camera and measurements of the gravity vectors provided by the IMUs placed on the camera and on the object, the P2P problem yields to:*

- *two solutions for orientation and translation when $\Delta_x \neq 0$ or $\Delta_y \neq 0$;*
- *a unique solution[6] for orientation and an infinite number of solutions for translation when $\tilde{P}_A = \tilde{P}_B$ and the line between the two feature points and O is not aligned with the gravity vector;*
- *an infinite number of solutions for orientation and for translation when $\tilde{P}_A = \tilde{P}_B$ and the line between the two feature points and O is aligned with the gravity vector.*

Hereafter the focus will be placed on the case $\Delta_x \neq 0$ or $\Delta_y \neq 0$. It should be remarked that although the P2P problem with accelerometers admits two solutions, in the general case, in some situations one of these solutions has no physical meaning and can be discarded. This situation occurs when one of the two computed transformation matrices maps some of the points of the P2P outside of the camera view.

The camera view, as shown in Figure 5.12, can be defined as the conic combination of vectors v_1, v_2, v_3, v_4 pointing from the camera focus to the image corners. As a consequence a point P expressed w.r.t. the camera frame will be in the camera view if and only if it can be represented as a conic combination of v_1, v_2, v_3, v_4. The latter is equivalent to say that P can be expressed as a conic combination of either v_1, v_2, v_3 or v_1, v_3, v_4, *i.e.*:

$$\exists\, k_1, k_2, k_3 \geq 0 : [v_1\ v_2\ v_3] \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = P$$

or

$$\exists\, k_1, k_3, k_4 \geq 0 : [v_1\ v_3\ v_4] \begin{bmatrix} k_1 \\ k_3 \\ k_4 \end{bmatrix} = P$$

Being v_1, v_2, v_3 and v_1, v_3, v_4 two sets of 3 conical independent vectors of $\mathbb{R}^3$, the latter condition is equivalent to test that:

[6] This is true under the realistic assumption that it is possible to recognize which of the two point is nearer to the camera, otherwise the possible orientations are two.

$$[v_1\ v_2\ v_3]^{-1}\ P \geq 0 \tag{5.25}$$

or

$$[v_1\ v_3\ v_4]^{-1}\ P \geq 0 \tag{5.26}$$

where '> 0' has to be meant elementwise (each entry is greater or equal to zero). Notice that since matrices $[v_1\ v_2\ v_3]$ and $[v_1\ v_3\ v_4]$ only depend on the camera characteristics, they do not change in time and their inverse can be computed a priori. To test if a given transformation matrix R_t has a physical meaning the following algorithm can be used

Algorithm 1: P2P Solution Feasibility Test

1 **compute** $P_A = RA + t$ and $P_B = RB + t$
2 **check** that Eq. (5.25) or Eq. (5.26) are satisfied to ensure that P_A and P_B are in the view of the camera
3 **if** one of the points is not in the view, **then** discard R_t

To evaluate the number of cases where the proposed test succeeds in eliminating the P2P ambiguity, a set of 125 000 randomly generated numerical simulations has been performed. In each test, both the configuration of the object and of the camera were randomly chosen. In 47% of the tests only one of the obtained two solutions to the P2P problem has a physical meaning. In these tests, thanks to the proposed field of view test, the wrong solution has been discarded.

5.5.3 P3P problem using accelerometers only

To solve the ambiguities w.r.t. the two solutions of the P2P problem, a possible way is to use a third feature point $C = (C_u, C_v, C_w)$. The only assumption on the resulting P3P problem is that points A, B and C must be not collinear.

The first main difference with respect to the P2P case is that, being the three points non collinear, it is not possible to see a unique pixel in the image provided by the camera, since it is impossible to have the three points and the origin of the camera reference frame on the same line.

Without loss of generality, let us assume that A and B are not aligned with the focus of the camera. The main steps to solve the P3P problem are:

1. solve the P2P problem related to the points A and B;
2. use the third point, C, and its related pixel in the image, $\tilde{P}_C$, to choose the solution of the P3P problem between the solutions of the P2P problem related to A and B.

More precisely, assuming $\Delta_x \neq 0$ or $\Delta_y \neq 0$, first of all a P2P problem with points A, B and pixels $\tilde{P}_A, \tilde{P}_B$ is solved by using the method presented in the

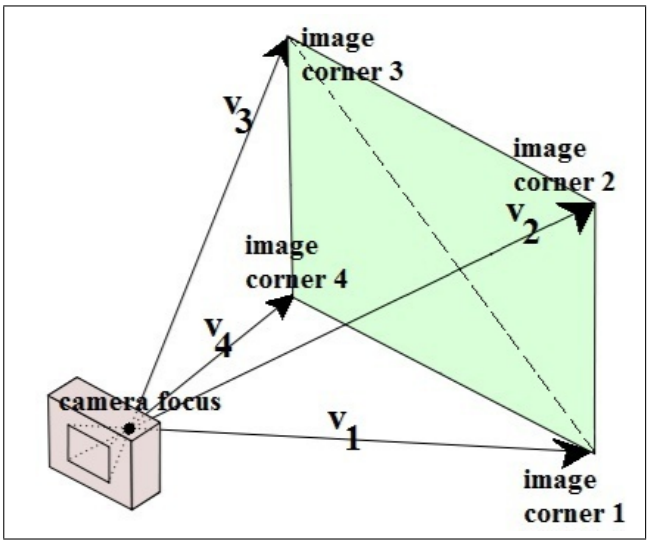


Fig. 5.12: Camera field of view as conic combination

previous sections. By doing so two possible solutions, (R_1, t_1) and (R_2, t_2) are obtained, and the one which better projects the point C into the pixel $\tilde{P}_C$ is chosen.

More formally, using equations (5.2), the pixels $\tilde{P}_{C,1}$ and $\tilde{P}_{C,2}$, which project the point C using (R_1, t_1) and (R_2, t_2) respectively, are computed. If $||\tilde{P}_{C,1} - \tilde{P}_C|| < ||\tilde{P}_{C,2} - \tilde{P}_C||$, then (R_1, t_1) is chosen as the solution of the P3P problem, otherwise (R_2, t_2) is chosen.

Remark 5.4. Please note that, compared to other algorithms based on the solution of the P3P and P4P problems, the algorithm proposed in this dissertation is typically faster, since it is based on the solution of a simple P2P problem and a few further tests to resolve the ambiguity between its two solutions. Finally, note that to increase the robustness w.r.t. pixel noise, it may be convenient to perform the P2P on the couple of points showing the maximal distance in the image plane.

5.5.4 PnP problem using accelerometers only

Solving the PnP problem using only information provided by accelerometers translates in the problem of finding the angle α of the parametrized rotation matrix $R(\alpha)$ and the translation vector $t = [t_x, t_y, t_z]^T$ between the camera and the object.

Given the set of feature points $\{P_i\}_{1=1}^n$ in the object reference frame, and the related set of pixels $\{\tilde{P}_i\}_{i=1}^n$, the following index is defined

$$J(\alpha, t_x, t_y, t_z) = \sum_{i=1}^{n} ||\hat{P}_i(\alpha, [t_x, t_y, t_z]^T) - \tilde{P}_i||^2 \tag{5.27}$$

where $\hat{P}_i(\alpha, [t_x, t_y, t_z]^T)$ is the pixel in which the feature point P_i is mapped using the transformation matrix $R_t(\alpha, [t_x, t_y, t_z]^T)$.

The above index represents the total quadratic reprojection error due to the use of the transformation matrix $R_t(\alpha, [t_x, t_y, t_z]^T)$. Using this index, the following optimization problem can be formulated:

$$\{\alpha^*, t_x^*, t_y^*, t_z^*\} = \arg \min_{\alpha, t_x, t_y, t_z} J(\alpha, t_x, t_y, t_z) \tag{5.28}$$

Once the optimal solution $\alpha^*, t_x^*, t_y^*, t_z^*$ has been found, the transformation matrix $R_t(\alpha^*, [t_x^*, t_y^*, t_z^*]^T)$ will be the solution of the PnP problem.

As explained in Section 5.5.2, to find an optimal solution for the reprojection index may be difficult and computationally onerous. Again, to overcome this problem, in this dissertation the use of LMS as an alternative cost function is proposed:

$$J_2(\alpha, t_x, t_y, t_z) = ||Q(\alpha)t - s(\alpha)||^2 \tag{5.29}$$

where $Q(\alpha)$ and $s(\alpha)$ are the Q matrix and the s vector obtained in (5.11) when the rotation matrix is $R = R(\alpha)$. Using this function, the optimization problem associated to the solution of the PnP problem becomes

$$\{\alpha^*, t_x^*, t_y^*, t_z^*\} = \arg \min_{\alpha, t_x, t_y, t_z} J_2(\alpha, t_x, t_y, t_z) \tag{5.30}$$

Interestingly enough, this four variables optimization problem can be reformulated without loss of generality as an equivalent single variable optimization problem. In fact, if the angle α is fixed, the translation vector $\hat{t}$ minimizing the cost function admits the closed-form solution (5.10). Hence, $\hat{t}$ can be rewritten as a function of α, i.e. $\hat{t}(\alpha)$, and (24) can be equivalently reformulated as

$$\alpha^* = \arg \min_{\alpha} J_2(\alpha, \hat{t}_x(\alpha), \hat{t}_y(\alpha), \hat{t}_z(\alpha)) \tag{5.31}$$

where $\left[\hat{t}_x(\alpha), \hat{t}_y(\alpha), \hat{t}_z(\alpha)\right]^T = Q(\alpha)^\dagger s(\alpha)$. Clearly, the optimal solution of (5.31) will be the optimal solution for Problem (5.30) too.

The main advantage of this reformulation is that the optimization problem (5.31) is a single variable unconstrained optimization problem the solution of which is computationally less cumbersome that the one of (5.30). Several efficient unconstrained optimization algorithms exist in the literature that can be used to solve the optimization problem (5.31), *e.g.*, the Nelder-Mead method [73].

Please note that most of these algorithms are iterative and require an initial guess. So, to solve (5.31), a good initialization value α_0 is required. To find a good initial guess, some heuristic solution must be computed. A first possible heuristic consists of solving a P3P problem on three arbitrary chosen feature points. Another more performing heuristic will be presented in the next subsection.

5.5.5 Weighted mean initialization for the PnP problem

In this subsection a possible alternative way to initialize the optimization problem (5.31) is described. The main idea is to solve ρ P3P problems obtained by choosing arbitrarily ρ subset of 3 non collinear feature points among the n available.

At this point, let $(\tilde{\alpha}_j, \tilde{t}_j), j = 1, \ldots, \rho$ be the ρ solutions of these P3P problems, the following weights can be then computed

$$w_j = \frac{\left(\frac{1}{\sum_{i=1}^{n} ||P_{x,i} - \hat{P}_{x,i}(\tilde{\alpha}_j, \tilde{t}_j)||^2} \right)^2}{\sum_{j=1}^{\rho} \left(\frac{1}{\sum_{i=1}^{n} ||P_{x,i} - \hat{P}_{x,i}(\tilde{\alpha}_j, \tilde{t}_j)||^2} \right)^2} \tag{5.32}$$

where $\hat{P}_{x,i}(\tilde{\alpha}_j, \tilde{t}_j)$ is the pixel computed using the transformation matrix $R_t(\tilde{\alpha}_j, \tilde{t}_j)$.

By using these weights, the following estimation of α can be obtained

$$\hat{\alpha} = \sum_{j=1}^{\rho} w_j \tilde{\alpha}_j \tag{5.33}$$

which can be used as initialization value, $\alpha_0 = \tilde{\alpha}$, for the optimization problem (5.31). Using the proposed weights $w_i, i = 1, \ldots, \rho$, an angle α_j will give higher contribute to the final $\hat{\alpha}$ if its transformation matrix $R_t(\tilde{\alpha}_j, \tilde{t}_j)$ has a lower quadratic re-projection error over the n correspondences.

The proposed heuristic can be summarized by the following algorithm:

Algorithm 2: PnP Problem Initialization Algorithm

1 choose ρ subsets of 3 not collinear feature points;
2 solve ρ P3P problems over the ρ subsets and find $(\tilde{\alpha}_j, \tilde{t}_j), j = 1, \ldots, \rho$;
3 evaluate $w_j, j = 1, \ldots, \rho$ using (5.32);
4 evaluate $\hat{\alpha}$ using (5.33).

5.6 Chapter Conclusions

In this Chapter the problem of estimating the relative orientation and position between a camera and an object has been considered. It is assumed

that both the camera and the object are equipped with an IMU and that the object contains a feature of n points the position of which in the object reference frame is known *a priori*. Using the image provided by the camera and the information measured by the IMUs, two new algorithms to solve the PnP problem have been presented. The first covers the case all the information provided by the IMU is used. The second algorithm covers the case only the measurements provided by inclinometers are used and the ones from the magnetometer are discarded due to their unreliability. It has been shown that if the entire information provided by IMUs is used, then the resulting PnP problem solution is more affected by noise than the solution obtained using only cameras and inclinometers. In Chapter 6, numerical simulations and experimental tests will show the effectiveness of the proposed solution for the PnP problem.

Appendix

Equation (5.21) assuming $\tilde{x}_A - \tilde{x}_B \neq 0$ and $\tilde{y}_A - \tilde{y}_B \neq 0$, from the first and the third equation in (5.3) t_x and t_z may be written as

$$
\begin{aligned}
t_z = & \tfrac{1}{(\tilde{x}_A - \tilde{x}_B)\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}[\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)} \\
& \times (g_u(B_u(\tilde{x}_B g_z - g_x) + A_u(g_x - \tilde{x}_A g_z)) \\
& + g_v(B_v(\tilde{x}_B g_z - g_x) + A_v(g_x - \tilde{x}_A g_z))) \\
& + (g_u g_v(B_u(g_x\tilde{x}_B + g_z) - A_u(g_x\tilde{x}_A + g_z)) + g_u^2(A_v(g_x\tilde{x}_A + g_z) \\
& - B_v(g_x\tilde{x}_B + g_z) + g_w g_y(A_u(g_x - \tilde{x}_A g_z) - B_u(g_x - \tilde{x}_B g_z))) \\
& + g_w(A_v g_w(g_x\tilde{x}_A + g_z) - B_v g_w(g_x\tilde{x}_B + g_z) \\
& + g_y(g_v^2 + g_w^2)(B_u(\tilde{x}_B g_z - g_x) + A_u(g_x - \tilde{x}_A g_z)))) \cos\alpha \\
& + (g_y(g_u^2 + g_w^2)(A_v(\tilde{x}_A g_z - g_x) + B_v(g_x - \tilde{x}_B g_z)) \\
& + A_u(g_w(g_x\tilde{x}_A + g_z) + g_u g_v g_y(g_x - \tilde{x}_A g_z)) \\
& - B_u(g_w(g_x\tilde{x}_B + g_z) + g_u g_v g_y(g_x - \tilde{x}_B g_z))) \sin\alpha]
\end{aligned}
$$

$$
\begin{aligned}
t_x = & \tfrac{1}{(\tilde{x}_A - \tilde{x}_B)\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}[\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)} \\
& \times (g_u(A_u\tilde{x}_B(g_x - \tilde{x}_A g_z) + B_u\tilde{x}_A(\tilde{x}_B g_z - g_x)) \\
& + g_v(A_v\tilde{x}_B(g_x - \tilde{x}_A g_z) + B_v\tilde{x}_A(\tilde{x}_B g_z - g_x))) \\
& + (g_u g_v(-A_u\tilde{x}_B(g_x\tilde{x}_A + g_z) + B_u\tilde{x}_A(g_x\tilde{x}_B + g_z)) \\
& + g_u^2(A_v\tilde{x}_B(g_x\tilde{x}_A + g_z) - B_v\tilde{x}_A(g_x\tilde{x}_B + g_z) \\
& + g_w g_y(B_u\tilde{x}_A(\tilde{x}_B g_z - g_x) + A_u\tilde{x}_B(g_x - \tilde{x}_A g_z)) \\
& + g_w(A_v g_w\tilde{x}_B(g_x\tilde{x}_A + g_z) - B_v g_w\tilde{x}_A(g_x\tilde{x}_B + g_z) \\
& + g_y(g_v^2 + g_w^2)(A_u\tilde{x}_b(g_x - \tilde{x}_A g_z) + B_u\tilde{x}_A(\tilde{x}_B g_z - g_x)))) \cos\alpha \\
& + (A_u\tilde{x}_B(g_w(g_x\tilde{x}_A + g_z) + g_u g_v g_y(g_x - \tilde{x}_A g_z)) \\
& + (g_u^2 + g_w^2)g_y(A_v\tilde{x}_B(-g_x + \tilde{x}_A g_z) + B_v(g_x - \tilde{x}_B g_z)) \\
& - B_u\tilde{x}_A(g_w(g_x\tilde{x}_B + g_z) + g_u g_v g_y(g_x - \tilde{x}_b g_z))) \sin\alpha]
\end{aligned}
$$

while from the second and the fourth equation in (5.3) t_y and t_z are

$$
\begin{aligned}
t_z = & \tfrac{1}{(\tilde{y}_A - \tilde{y}_B)\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}[\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)} \\
& \times (g_u(B_u(\tilde{y}_B g_z - g_y) + A_u(g_y - \tilde{y}_A g_z)) \\
& + g_v(B_v(\tilde{y}_B g_z - g_y) + A_v(g_y - \tilde{y}_A g_z))) \\
& + (g_u g_v g_x(-A_u\tilde{y}_A + B_u\tilde{y}_B) + g_u^2(g_x(A_v\tilde{y}_A - B_v\tilde{y}_B) + \\
& + B_u g_w(g_x^2 + g_y\tilde{y}_B g_z + g_z^2) - A_u g_w(g_x^2 + g_z(g_y\tilde{y}_A + g_z))) \\
& + g_w(g_w g_x(A_v\tilde{y}_A - B_v\tilde{y}_B) - A_u(g_v^2 + g_w^2)(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) \\
& + B_u(g_v^2 + g_w^2)(g_x^2 + g_z(g_y\tilde{y}_B + g_z)))) \cos\alpha + (g_w(g_x(A_u\tilde{y}_A - B_u\tilde{y}_B) \\
& - B_v g_w(g_x^2 + g_y\tilde{y}_B g_z + g_z^2) + A_v g_w(g_x^2 + g_z(g_y\tilde{y}_A + g_z))) \\
& + g_u g_v(-A_u(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) + B_u(g_x^2 + g_z(g_y\tilde{y}_B + g_z))) \\
& + g_u^2(A_v(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) - B_v(g_x^2 + g_z(g_y\tilde{y}_B + g_z)))) \sin\alpha]
\end{aligned}
$$

$$
\begin{aligned}
t_y = {} & \tfrac{1}{(\tilde{y}_A-\tilde{y}_B)\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}[\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)} \\
& \times (g_u(A_u\tilde{y}_B(g_y-\tilde{y}_Ag_z)+B_u(-g_y\tilde{y}_A+\tilde{y}_A\tilde{y}_Bg_z))+g_v(A_v\tilde{y}_B(g_y-\tilde{y}_Ag_z) \\
& + B_v(-g_y\tilde{y}_A+\tilde{y}_A\tilde{y}_Bg_z)))+(g_u(-A_u+B_u)g_vg_x\tilde{y}_AB_v \\
& + g_u^2(B_ug_w\tilde{y}_A(g_x^2+g_z(g_y\tilde{y}_B+g_z))-\tilde{y}_B((-A_v+B_v)g_wg_x\tilde{y}_A \\
& + A_u(g_v^2+g_w^2)(g_x^2+g_z(g_y\tilde{y}_A+g_z)))))\cos\alpha \\
& + (g_ug_v(-A_u\tilde{y}_B(g_x^2+g_y\tilde{y}_Ag_z+g_z^2)+B_u\tilde{y}_A(g_x^2+g_y\tilde{y}_Bg_z+g_z^2)) \\
& + g_u^2(A_v\tilde{y}_B(g_x^2+g_y\tilde{y}_Ag_z+g_z^2)-B_v\tilde{y}_A(g_x^2+g_y\tilde{y}_Bg_z+g_z^2)) \\
& + g_w(-B_vg_w\tilde{y}_A(g_x^2+g_z(g_y\tilde{y}_B+g_z)) \\
& + \tilde{y}_B((A_u-B_u)g_x\tilde{y}_A+A_vg_w(g_x^2+g_y\tilde{y}_Ag_z+g_z^2))))\sin\alpha]
\end{aligned}
$$

By equating the two values of t_z we obtain

$$a\sin\alpha + b\cos\alpha + c = 0$$

where

$$
\begin{aligned}
a = {} & +\tfrac{1}{\tilde{x}_A-\tilde{x}_B}[(g_u^2+g_v^2)g_y(-A_vg_x+B_vg_x+A_v\tilde{x}_Ag_z-B_v\tilde{x}_Bg_z) \\
& + A_u(g_w(g_x\tilde{x}_A+g_z)+g_ug_vg_y(g_x-\tilde{x}_Ag_z)) \\
& - B_u(g_w(g_x\tilde{x}_B+g_z)+g_ug_vg_y(g_x-\tilde{x}_Bg_z))] \\
& - \tfrac{1}{\tilde{y}_A-\tilde{y}_B}[g_w(g_x(A_u\tilde{y}_A-B_u\tilde{y}_B)-B_vg_w(g_x^2+g_y\tilde{y}_Bg_z+g_z^2) \\
& + A_vg_w(g_x^2+g_z(g_y\tilde{y}_A+g_z)))+g_ug_v(-A_u(g_x^2+g_z(g_y\tilde{y}_A+g_z)) \\
& + B_v(g_x^2+g_z(g_y\tilde{y}_B+g_z)))+g_u^2(A_v(g_x^2+g_z(g_y\tilde{y}_A+g_z)) \\
& - B_v(g_x^2+g_z(g_y\tilde{y}_B+g_z)))]
\end{aligned}
$$

$$
\begin{aligned}
b = {} & +\tfrac{1}{\tilde{x}_A-\tilde{x}_B}[g_ug_v(-A_u(g_x\tilde{x}_A+g_z)+B_u(g_xB_v+g_z)) \\
& + g_u^2(A_v(g_x\tilde{x}_A+g_z)-B_v(g_x\tilde{x}_B+g_z) \\
& + g_wg_y(A_ug_x-B_ug_x-A_u\tilde{x}_Ag_z+B_u\tilde{x}_Bg_z)) \\
& + g_w(A_vg_w(g_x\tilde{x}_A+g_z)-B_vg_w(g_x\tilde{x}_B+g_z) \\
& + g_y(g_v^2+g_w^2)(-B_ug_x+B_u\tilde{x}_Bg_z+A_u(g_x-\tilde{x}_Ag_z)))] \\
& - \tfrac{1}{\tilde{y}_A-\tilde{y}_B}[g_ug_vg_x(-A_u\tilde{y}_A+B_u\tilde{y}_B)+g_u^2(g_x(A_v\tilde{y}_A-B_v\tilde{y}_B) \\
& + B_ug_w(g_x^2+g_y\tilde{y}_Bg_z+g_z^2)-A_u(g_v^2+g_w^2)(g_x^2+g_z(g_y\tilde{y}_A+g_z)) \\
& + B_u(g_v^2+g_w^2)(g_x^2+g_z(g_y\tilde{y}_B+g_z)))]
\end{aligned}
$$

$$
\begin{aligned}
c = {} & +\tfrac{\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}{\tilde{x}_A-\tilde{x}_B}[g_u(-B_ug_x+B_u\tilde{x}_Bg_z+A_u(g_x-\tilde{x}_Ag_z)) \\
& + g_v(-B_vg_x+B_v\tilde{x}_Bg_z+A_v(g_x-\tilde{x}_Ag_z))] \\
& - \tfrac{\sqrt{(g_u^2+g_w^2)(g_x^2+g_z^2)}}{\tilde{y}_A-\tilde{y}_B}[g_u(-B_ug_y+B_u\tilde{y}_Bg_z+A_u(g_y-\tilde{y}_Ag_z)) \\
& + g_v(-B_vg_y+B_v\tilde{y}_Bg_z+A_v(g_y-\tilde{y}_Ag_z))]
\end{aligned}
$$

Equation (5.21) assuming $\Delta_x = 0$ and $\Delta_y \neq 0$.

$$
\begin{aligned}
a_1 = &- g_y(A_v - B_v)(g_u^2 + g_w^2)(g_x - \tilde{x}_A g_z) + A_u(g_w(g_x\tilde{x}_A + g_z) + g_u g_v g_y(g_x - \tilde{x}_A g_z)) \\
&- B_u(g_w(x\tilde{x}_A + g_z) + g_u g_v g_y(g_x - \tilde{x}_A g_z))
\end{aligned}
$$

$$
\begin{aligned}
b_1 = &- g_u g_v(A_u - B_u)(g_x\tilde{x}_A + g_z) \\
&+ g_u^2(A_v(g_x\tilde{x}_A + g_z) - B_v(g_x\tilde{x}_A + g_z) + (A_u - B_u)g_w g_y(g_x - \tilde{x}_A g_z)) \\
&+ g_w(A_v g_w(g_x\tilde{x}_A + g_z) - B_v g_w(g_x\tilde{x}_A + g_z) + g_y(A_u - B_u)(g_v^2 + g_w^2)(g_x - \tilde{x}_A g_z))
\end{aligned}
$$

$$
c_1 = + g_u(A_u - B_u) + g_v(A_v - B_v)(g_x - \tilde{x}_A g_z)\sqrt{(g_u^2 + g_w^2)(g_x^2 + g_z^2)}
$$

Equation (5.21) assuming $\Delta_x \neq 0$ and $\Delta_y = 0$.

$$
\begin{aligned}
a_2 = &- g_u g_v(A_u - B_u)(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) + g_u^2(A_v - B_v)(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) \\
&+ g_w((A_u - B_u)g_x\tilde{y}_A - B_v g_w(g_x^2 + g_y\tilde{y}_A g_z + g_z^2) + A_v g_w(g_x^2 + g_z(g_y\tilde{y}_A + g_z)))
\end{aligned}
$$

$$
\begin{aligned}
b_2 = &+ g_u g_v g_x\tilde{y}_A(-A_u + B_u) \\
&+ g_u^2((A_v - B_v)g_x\tilde{y}_A + B_u g_w(g_x^2 + g_y\tilde{y}_A g_z + g_z^2) - A_u g_w(g_x^2 + g_z(g_y\tilde{y}_A + g_z))) \\
&+ g_w((A_v - B_v)g_w g_x\tilde{y}_A - A_u(g_v^2 + g_w^2)(g_x^2 + g_z(g_y\tilde{y}_A + g_z)) \\
&+ B_u(g_v^2 + g_w^2)(g_x^2 + g_z(g_y\tilde{y}_A + g_z)))
\end{aligned}
$$

$$
c_2 = + g_u(A_u - B_u) + g_v(A_v - B_v)(g_y - \tilde{y}_A g_z)\sqrt{(g_u^2 + g_w^2)(g_x^2 + g_z^2)}
$$

6

Numerical and Experimental Results

In this Chapter, all the previously proposed techniques will be tested through numerical simulations and real experiments. First of all, the robot model will be described, then the proposed simulative and real environments will be discussed. A set of performance indexes will be defined and the set of performed simulations and experiments will be detailed showing the obtained results.

6.1 Robot Model

The mobile robots localization, mapping and SLAM techniques proposed in the previous Chapters can be adapted to whatever mobile robot type.

Whatever is the mobile robot type, assuming it moves on a plane, its model can be surely described using a set of equations like (1.2). All the algorithms described in the present thesis have been developed in a general framework, starting from the equation (1.2) and thus the specific mobile robot model choice does not influence any mathematical step in the algorithms developing. In the simulative and experimental framework a differential drive robot has been chosen.

A differential drive robot is a mobile robot the movement of which is based on **two separately driven wheels** placed on either side of the robot body. The robot can change its direction by varying the relative rate of rotation of its wheels. To balance the robot, an additional castor wheel is added, as shown in Figure 6.1.

If both the wheels are driven with the same speed, the robot will go in a straight line. If the wheels are driven with equal speed but in opposite directions, the robot will rotate about the central point of the axis. Otherwise, the robot will perform a curved trajectory depending on the values of the wheels speeds. A differentially steered robot is similar to the differential gears used in automobiles where both the wheels can have different rates of rotations. Unlike the differential gearing system, a differentially steered system will have

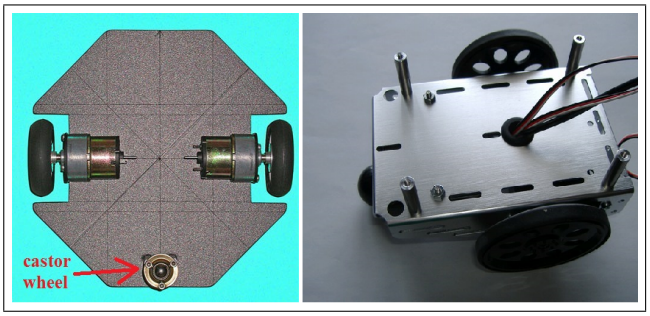


Fig. 6.1: Differential Drive Robot examples

both the wheels powered. Differential wheeled robots are used extensively in robotics, since their motion is easy to program and it can be well controlled. More formally this robot type can be described as shown in Figure 6.2.

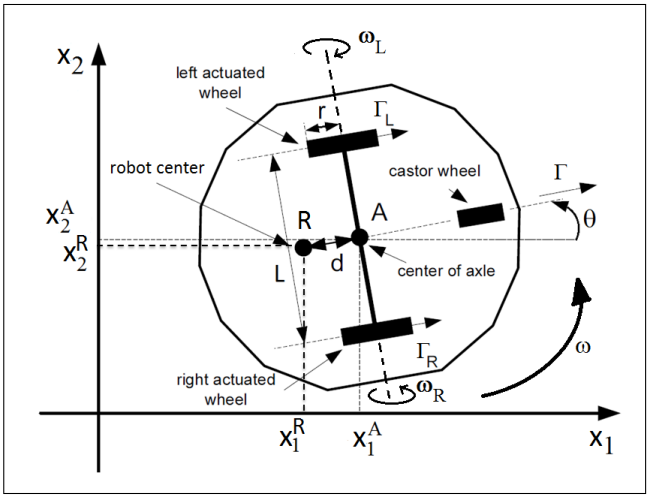


Fig. 6.2: Differential Drive Robot kinematic model

where:

- r is the wheels radius;
- L is the axle length between drive wheels;
- $A(t) = (x_1^A(t), x_2^A(t))$ is the center of axle between drive wheels;
- $R(t) = (x_1^R(t), x_2^R(t))$ is the robot center of gravity, hereafter denoted as robot center since a kinematic model is considered;
- $\theta(t)$ is robot orientation w.r.t. robot axis, orthogonal to wheels axis;

- $d = ||A(t) - R(t)||$ is the distance between the robot center and the center of axle;
- $\omega_R(t), \omega_L(t)$ are the right and left angular velocities respectively;
- $\Gamma(t)$ is the point $A(t)$ linear velocity;
- $\Gamma_R(t), \Gamma_L(t)$ are the right wheel and left wheel linear velocities respectively;
- $\omega(t)$ is the mobile robot rotational speed.

Starting from Figure 6.2 and neglecting the time dependencies for the sake of simplify notation, the velocity of the point A can be described by

$$\dot{x}_1^A = \Gamma \cos\theta, \dot{x}_2^A = \Gamma \sin\theta$$

while the robot rotational velocity is defined as

$$\dot{\theta} = \omega$$

The linear velocity Γ and the rotational speed ω are related to the wheels angular velocities by the following relationships:

$$\Gamma = r\frac{\omega_R + \omega_L}{2}, \omega = r\frac{\omega_R - \omega_L}{L}.$$

Using the above equations the kinematic model for a differential drive robot results in

$$\begin{bmatrix} \dot{x}_1^A \\ \dot{x}_2^A \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\frac{\omega_R + \omega_L}{2}\cos\theta \\ r\frac{\omega_R + \omega_L}{2}\sin\theta \\ r\frac{\omega_R - \omega_L}{L} \end{bmatrix} \tag{6.1}$$

At this point the velocity of the robot center R can be obtained as

$$\begin{aligned} \dot{x}_1^R &= \dot{x}_1^A + \dot{\theta} d \sin\theta \\ \dot{x}_2^R &= \dot{x}_2^A - \dot{\theta} d \cos\theta \end{aligned} \tag{6.2}$$

In the following, it will be assumed, without loss of generality, that the robot center is coincident with the axle center, that is $A \equiv R$ and $d = 0$. Defining the robot state $x(t) = [x_1^R(t),\ x_2^R(t),\ \theta(t)]$, the robot inputs vector $u = [\omega_R(t),\ \omega_L(t)]$ and neglecting the time dependencies, the differential drive model is

$$\dot{x} = \begin{bmatrix} \dot{x}_1^R \\ \dot{x}_2^R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\frac{\omega_R + \omega_L}{2}\cos\theta \\ r\frac{\omega_R + \omega_L}{2}\sin\theta \\ r\frac{\omega_R - \omega_L}{L} \end{bmatrix} = f\left(\begin{bmatrix} x_1^R \\ x_2^R \\ \theta \end{bmatrix}, \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}\right) = f(x,u) \tag{6.3}$$

The equations (6.3) show how the differential drive kinematic model can be adapted to the general framework defined by the equation (1.2).

6.1.1 Differential Drive Discrete model equations

The model (6.3) discretized using (1.3) is described by the following equations

$$\begin{cases} x^R_{1,k+1} = x^R_{1,k} + \Gamma_k\, T \cos(\theta_{k+1}) \\ x^R_{2,k+1} = x^R_{2,k} + \Gamma_k\, T \sin(\theta_{k+1}) \\ \theta_{k+1} = \theta_k + \Delta_k, \end{cases} \tag{6.4}$$

where:

- $(x^R_{1,k}, x^R_{2,k})$ is the robot center position at time $t_k = kT$;
- θ_k is the angle between the robot axle and the x_1-axis, at time t_k;
- $\omega_{L,k}$ and $\omega_{R,k}$ are the wheels angular velocities;
- $\Delta_k = r\,\dfrac{(\omega_{L,k} - \omega_{R,k})T}{L}$ is the rotation within $[t_{k-1}, t_k]$
- $T = t_k - t_{k-1}$ is the sampling period.

The above model does not take into account unmodeled dynamics, friction, wheels sleeping and external disturbances (such as wind). To consider the influence of these phenomena on the robot evolution and to not complicate the robot model, as described in Section 2.3, a process noise $w_k = [w_{1,k}, w_{2,k}, w_{\theta,k}]^T$ is added to the robot evolution. The resulting model is

$$x_{k+1} = \phi(x_k, u_k) + w_k \tag{6.5}$$

Where the $\phi(\cdot,\cdot)$ function consists in the equations (6.4) and it represents the robot evolution discrete model obtained thanks to the Euler approximation of the $f(\cdot,\cdot)$ function (see equation (1.3)).

6.1.2 Robot Khepera III

In the experimental framework a Khepera III mobile robot [27] has been used. This mobile robot, shown in Figure 6.3 has been chosen because of its small dimensions, which make it suitable for indoor experiments, and for its capability to be programmed using different operating systems and languages. Its communication module allows the robot to be monitored and controlled by a Personal Computer using the Bluetooth protocol. The robot is equipped with $n_S = 5$ on board ultrasonic sensors able to detect in a range from $20cm$ to $4m$.

In the experimental tests, all the proposed localization, mapping and SLAM algorithms have been implemented in Matlab code. The robot input velocities are computed using Matlab and they are sent to the robot through a bluetooth communication; using the same channel, all the robot sensors acquired measurements have been sent to Matlab to use them into the described localization and mapping algorithms.

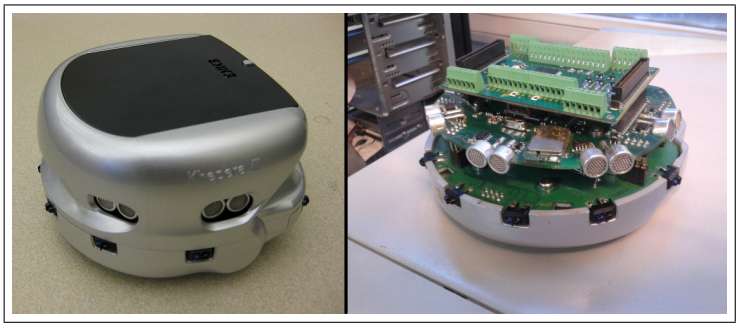


Fig. 6.3: Robot Khepera III equipped with upper body (on the left) and not equipped with upper body (on the right)

6.2 Robot surrounding environments

In this Section, the environments used during the simulation and experimental tests will be shown. To properly test all the proposed algorithms, two simulative environments and one real environment have been made.

6.2.1 Simulative environments

Single room environment

The first simulative environment (see Figure 6.4) aims to represent a single room in which the robot is placed. The room walls have various shapes and the room covered surface is about $2m^2$.

Three-rooms environment

The second simulative environment (see Figure 6.5) aims to represent three rooms connected by corridors. The robot starts its path into one of the rooms or into one of the corridors and then it moves into the entire environment. Also in this case, the room walls have various shapes; the environment covered surface is about $8m^2$.

6.2.2 Experimental environment

The experimental environment is a rectangular field of $1.30m \times 1.00m$ containing two boxes as shown in Figure 6.6.
The experimental environment boundaries are shown in Figure 6.7.

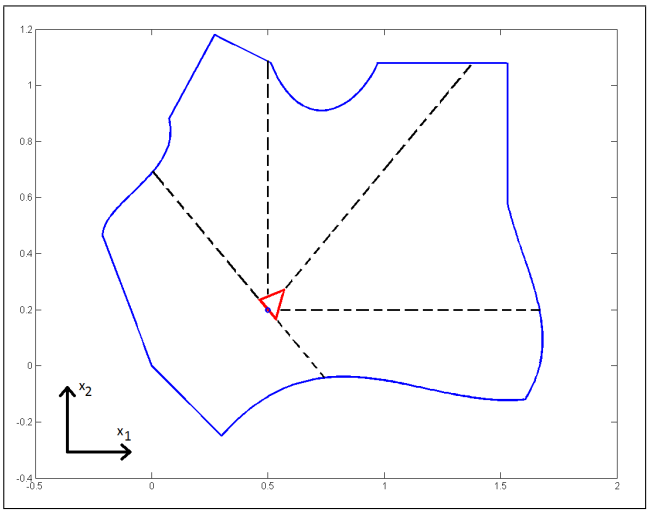


Fig. 6.4: Simulative environment: single room. The red triangle represents the robot while the blue circle is robot center coordinates. The black dashed lines are the robot on board distance sensors measurements

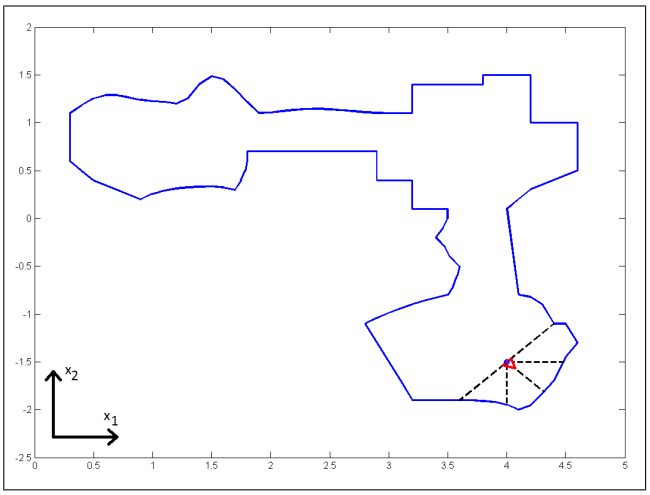


Fig. 6.5: Simulative environment: three-rooms. The red triangle represents the robot while the blue circle is robot center coordinates. The black dashed lines are the robot on board distance sensors measurements

6.3 Robot planned trajectories

As remarked into the Introduction, this dissertation does not aim to solve the path planning problem. The robot control inputs are assumed to be precomputed so that the robot follows the desired trajectories avoiding the obstacles

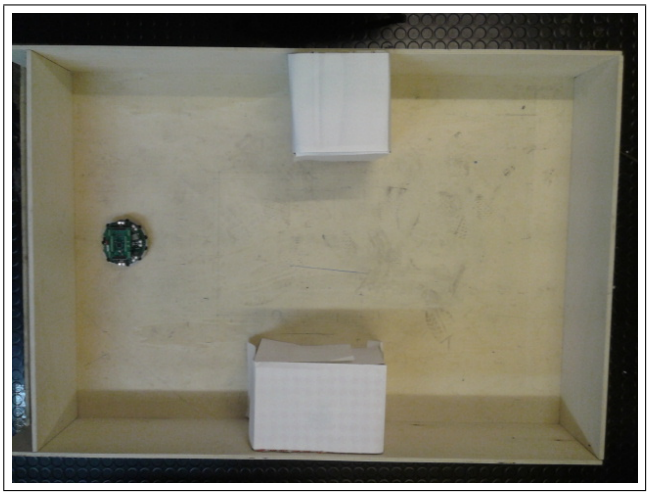


Fig. 6.6: Experimental environment. The robot Khepera III has been placed into the environment.

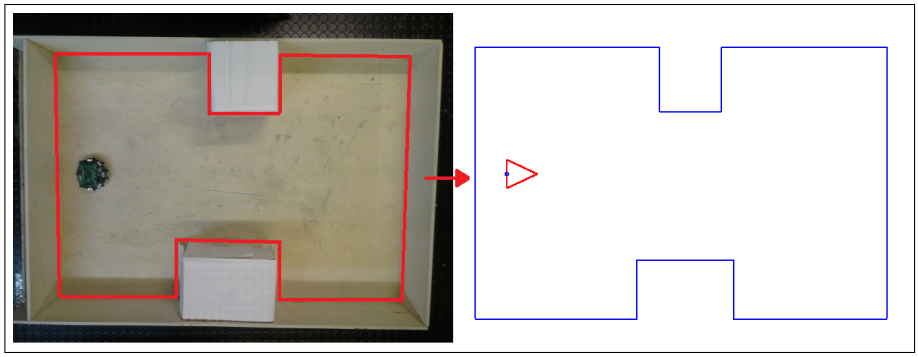


Fig. 6.7: Experimental environment boundaries

and the walls into the environment. More precisely a set of trajectories have been planned and used into the previously described environments.

6.3.1 Rectangle trajectory

The first trajectory planned is shown in Figure 6.8. It is a rectangle traveled by the robot starting from $(0.50, 0.40)m$ with zero heading, passing by $(1.00, 0.40)m$, $(1.00, 0.60)m$, $(0.50, 0.60)m$ and coming back to $(0.50, 0.40)m$. The entire path is performed in $k_f = 120$ steps.

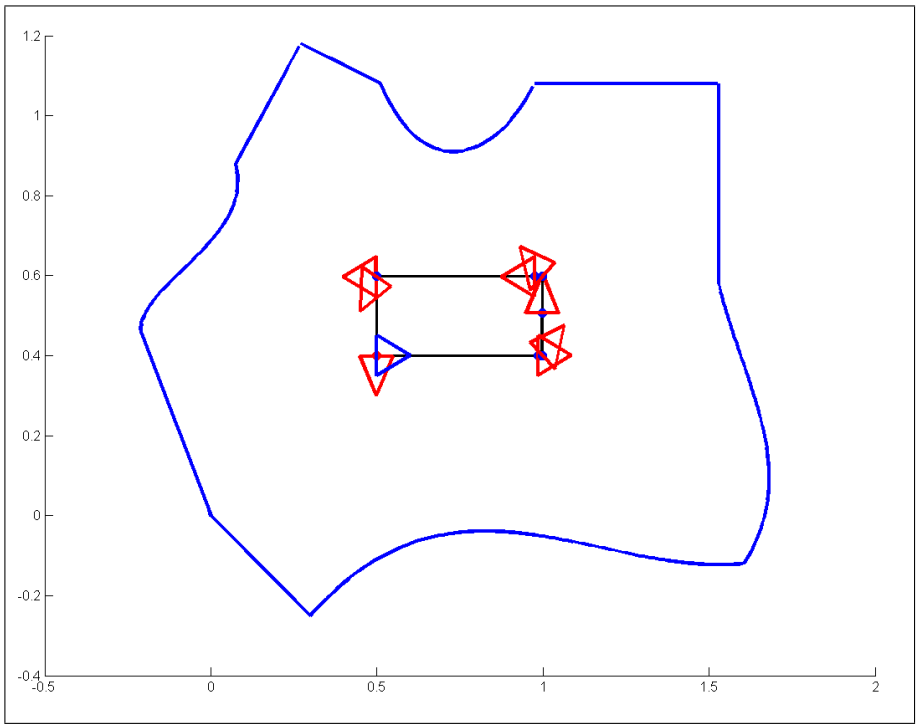


Fig. 6.8: Rectangle trajectory into the single room simulative environment. The blue triangle is robot initial pose

6.3.2 *I*-like trajectory

The second trajectory planned is shown in Figure 6.9. It is an I-like trajectory, starting from $(0.26, 0.50)m$ with zero heading, passing by $(0.26, 0.40)m$, $(0.26, 0.50)m$, $(1.10, 0.50)m$, $(1.10, 0.40)m$, $(1.10, 0.60)m$, $(1.10, 0.50)m$, $(0.26, 0.50)m$, $(0.26, 0.60)m$ and coming back to $(0.26, 0.50)m$. The entire path is performed in $k_f = 200$ steps.

6.3.3 *L*-like trajectory

In the three-rooms simulated environment, the planned robot trajectory starts from $(0.5, 0.9)m$, with zero heading, and passes by $(3.9, 0.9)m$, $(3.9, -1.6)m$, $(3.7, -1.6)$, $(3.7, 0.8)$ and $(0.5, 0.8)$. The overall path is performed in $k_f = 500$ steps, it is shown in Figure 6.10 and it will be hereafter denoted as L-like trajectory.

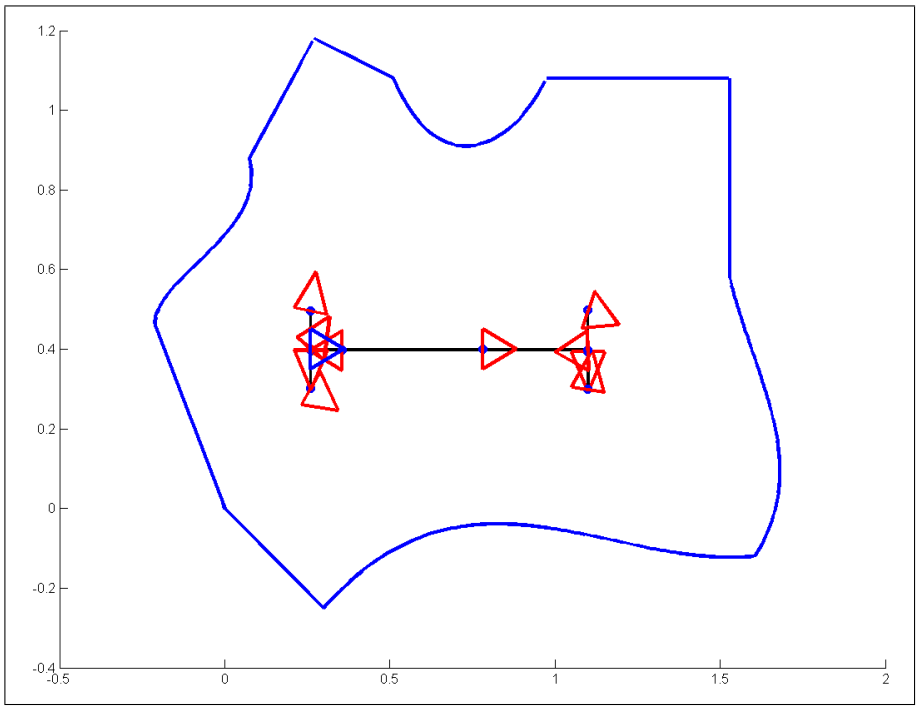


Fig. 6.9: I-like trajectory into the single room simulative environment. The blue triangle is robot initial pose

6.4 Robot and Kalman Filter parameters

The following parameters will be used for the mobile robot parameters model (see equations (6.4) and (6.5)) and for the Kalman filter matrices:

- $n_S = p = 5$ on board sensors placed as depicted in Fig. 6.11;
- $L = 0.09m$, $r = 0.0205m$ for the robot;
- Sampling period $T = 1s$;
- $V = 0.05^2 I_{n_S}$, where I_{n_S} is an identity matrix of order n_S. Each on board sensor measurement is then affected by a noise with standard deviation of $0.05m$;
- $W = diag\{0.01^2, 0.01^2, 0.0017^2\}$: a standard deviation of $0.01m$ on $w_{1,k}$ and $w_{2,k}$, a standard deviation of 0.1°on $w_{\theta,k}$;
- $\tilde{P}_{0|0} = diag\{0.05^2, 0.05^2, 0.0873^2\}$: a standard deviation of $0.05m$ on robot initial estimation position error and a standard deviation of 5° on robot initial estimation heading error.

The following UKF weights will be used:

$$\alpha = 0.001,\ \beta = 2,\ \kappa = 3 - n,$$
$$\lambda = \alpha^2(n + \kappa) - n.$$

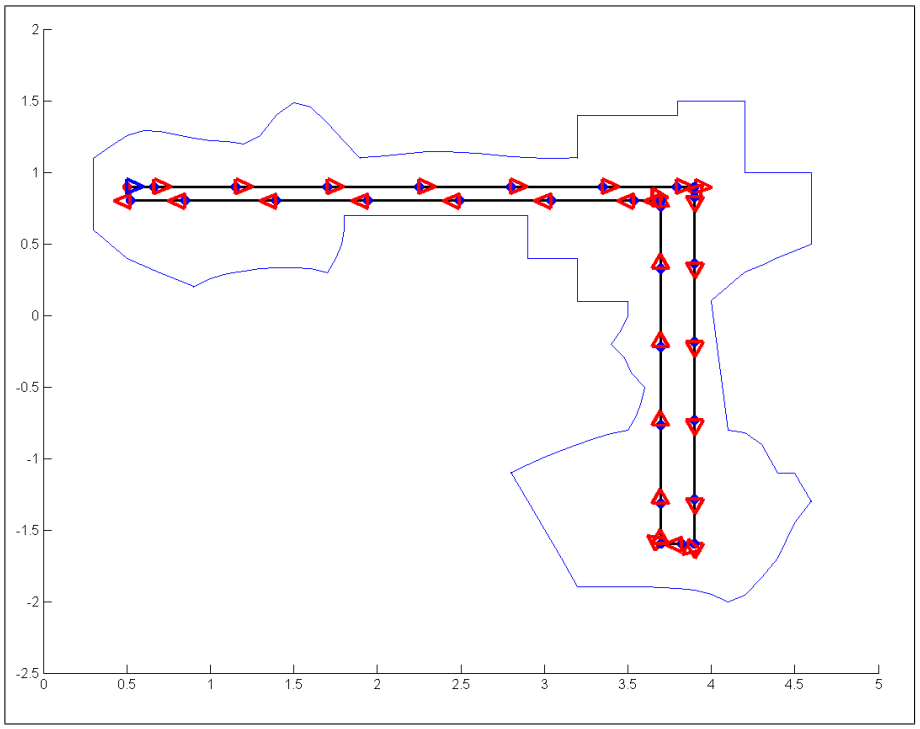


Fig. 6.10: L-like trajectory into the three-rooms simulative environment. The blue triangle is robot initial pose

According to [18], the choice $\beta = 2$ minimizes the error in the fourth-order moment of the a-posteriori covariance when the random vector is Gaussian. Furthermore:

$$R_1^m = \frac{\lambda}{n+\lambda},\ R_1^c = \frac{\lambda}{n+\lambda} + 1 + \beta - \alpha^2$$

$$R_j^m = R_j^c = \frac{\lambda}{2\,(n+\lambda)},\ j = 2, \ldots, 2n+1.$$

For what regards the mobile robot parameters (wheels radius, wheels distance, on board sensors locations), the Khepera III parameters have been used. About the filter initial condition $\hat{x}_{0|0}$, it has been randomly generated on the basis of the statistical properties imposed by $P_{0|0}$.

6.5 Performance Indexes

To evaluate the proposed algorithms performance, three indexes have been defined.

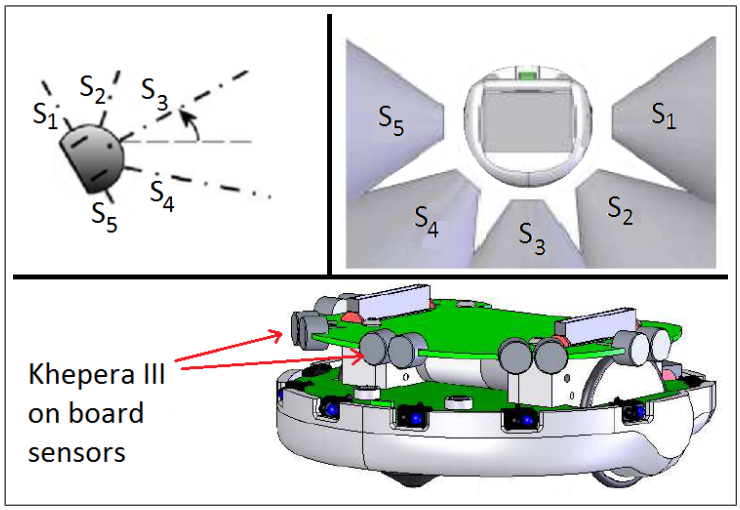


Fig. 6.11: Mobile robot on board sensors locations

Estimation/Localization index

To evaluate the localization performance of the mobile robots localization and SLAM algorithms, the following index has been used

$$\varepsilon = \frac{1}{k_f + 1} \frac{2\varepsilon_p + \varepsilon_o}{3} \times 100 \tag{6.6}$$

where

$$\varepsilon_p = \sum_{k=0}^{k_f} \frac{\left\| \begin{bmatrix} x^R_{1,k} \\ x^R_{2,k} \end{bmatrix} - \begin{bmatrix} \hat{x}^R_{1,k|k} \\ \hat{x}^R_{2,k|k} \end{bmatrix} \right\|}{\left\| \begin{bmatrix} x^R_{1,k} \\ x^R_{2,k} \end{bmatrix} \right\|}$$

is related to robot position estimation error, while

$$\varepsilon_o = \sum_{k=0}^{k_f} \left\| \begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} - \begin{bmatrix} \cos(\hat{\theta}_{k|k}) \\ \sin(\hat{\theta}_{k|k}) \end{bmatrix} \right\|$$

is related to robot orientation estimation error.
k_f is the number of steps in which each experiment/simulation is performed and $||\cdot||$ is the euclidean norm. The above index represents the percentage relative localization error between the estimated robot pose $\hat{x}_{k|k}$ and the real robot pose x_k.

Following the same lines, the estimation performance of the PnP problem solutions (proposed in Chapter 5) has been evaluated by the index

$$\varepsilon = \frac{||R_t - \hat{R}_t||}{||R_t||} \tag{6.7}$$

where $||\cdot||$ is the euclidean norm, R_t is the real transformation matrix between the object and the camera while $\hat{R}_t$ is the estimated transformation matrix. This index represents the relative estimation error between the estimated transformation matrix and the real one.

Mapping index

To evaluate the quality of the environment mapping algorithms a further index γ has been used. The index is evaluated as follows: all the proposed mapping algorithms provide a set of landmarks approximating environment portions. Given the i-th landmark, a set of n_i points, $P^{i,j}, j = 1, \ldots, n_i$, equally spaced by an amount equal to 0.01, is computed on the landmark surface (e.g. on the polynomial shape if the landmark is a polynomial). For each point $P^{i,j}$, the distance $d^{i,j}$ between this point and its nearest real environment boundaries point is computed yielding to the index

$$\gamma = \frac{1}{n_k} \sum_{i=1}^{n_k} \gamma_i \tag{6.8}$$

where $\gamma_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d^{i,j}$ and n_k is the number of landmarks contained into $\mathbb{E}_k$ (see Chapter 3 for more details).

Computation time index

The third index is used to evaluate the time performance and the required computational effort, of the proposed algorithms. It is defined as

$$\tau = \frac{1}{k_f + 1} \sum_{k=0}^{k_f} \tau_k \tag{6.9}$$

where τ_k is the execution time of the algorithm of interest at step k. This index represents the algorithm averaged execution time.

Please note that the computed τ index values can not be considered in an absolute point of view since they are obviously influenced by the used CPU and by the used operative system. However these values can be read as an indication of the algorithm time performance and they provide very important information in a relative point of view. Neglecting a common bias affecting all the τ indexes values (due to CPU and operating system), considerations about the fastest or the slowest algorithm can be extracted from these indexes values.

6.6 Mobile robots localization results

In this Section the numerical and experimental results about the mobile robot localization techniques proposed in Chapter 2 will be shown.

6.6.1 Mobile robots localization in a perfectly known environment

Assuming the robot placed in the environment shown in Figure 2.1, with $l_x = 1.5m$ and $l_y = 1m$, two sets of 100 real experiments have been performed. In the first set the robot follows the rectangle trajectory while in the second set it performs the I-like trajectory.

Fig. 6.12: Real perfectly known rectangular environment

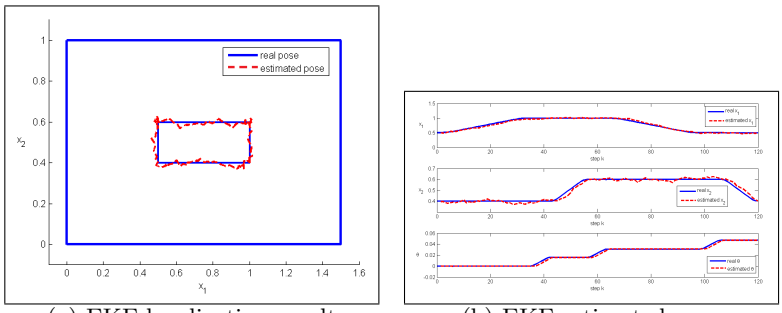
Table 6.1 shows the results using the Extended Kalman filter and the Unscented Kalman filter in terms of the averaged localization performance index ε and of the averaged computational cost index τ, over the performed tests in the real experimental framework.

Table 6.1: ε index and τ index - perfectly known environment - real framework experiments

	EKF		UKF	
Trajectory	averaged ε	averaged τ	averaged ε	averaged τ
Rectangle	1.8%	0.077s	1.7%	0.25s
I-like	2.2%	0.075s	4%	0.23s

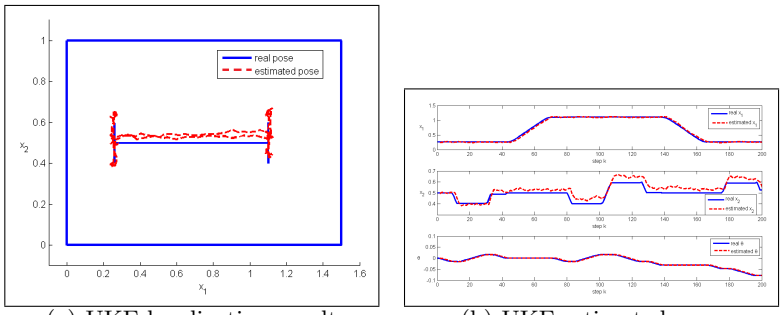
Figures 6.13(a), 6.13(b), 6.14(a), 6.14(b) show a localization example in the experimental framework.

The most evident consideration which comes from the analysis of the results shown in Table 6.1 is that the Extended Kalman filter and the Unscented



(a) EKF localization result (b) EKF estimated pose

Fig. 6.13: Experimental result using the EKF, rectangle trajectory



(a) UKF localization result (b) UKF estimated pose

Fig. 6.14: Experimental result using the UKF, I-like trajectory

Kalman filter perform comparably well in reconstructing the robot pose. This is quite surprising, as the estimation properties of the UKF are superior to those of the EKF (as shown in [19]), and it is probably due to the very smooth nonlinearities in the differential drive robot model; these nonlinearities are not bad enough to highlight any substantial difference between the UKF performance and the EKF performance.

Localization using the Switching logic based on the Observations Effect Maximization

In the following, the mobile robot localization results, in a perfectly known environment, using the observations effect maximization switching rule proposed in Chapter 4, along with the Extended Kalman filter and the Unscented Kalman filter, will be shown. Once again 100 real experiments have been performed and three kinds of estimation algorithms have been used:

- Estimation algorithm (EKF/UKF) using a single sensor kept fixed along all the path. This case will be referred as $S_j - filter$ for the case where the j-th sensor is used.

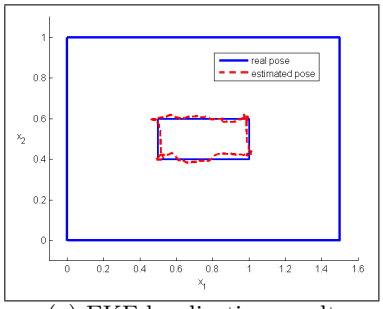
- Estimation algorithm (EKF/UKF) using the proposed observations effect maximization switching rule. This case will be referred as *Switching-filter*.
- Traditional estimation algorithm (EKF/UKF) based on the use of all the $n_S = 5$ available sensors. This case will be referred as *Standard filter*.

Table 6.2 shows the results in the performed experiments.

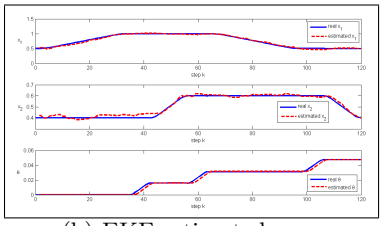
Table 6.2: ε index and τ index - perfectly known environment - real framework experiments

		EKF		UKF	
Trajectory	Filter	averaged ε	averaged τ	averaged ε	averaged τ
Rectangle	$s_1 - filter$	2.5%	0.075s	2.9%	0.23s
	$s_2 - filter$	3%	0.074s	3%	0.23s
	$s_3 - filter$	2.7%	0.072s	2.8%	0.22s
	$s_4 - filter$	2.9%	0.074s	3.1%	0.24s
	$s_5 - filter$	2.45%	0.075s	2.8%	0.23s
	$Switching - filter$	2.4%	0.09s	2.7%	0.26s
	$Standard\ filter$	1.8%	0.077s	1.7%	0.25s
I-like	$s_1 - filter$	3.2%	0.075s	4.3%	0.24s
	$s_2 - filter$	3.5%	0.074s	4.4%	0.24s
	$s_3 - filter$	3%	0.072s	4.3%	0.23s
	$s_4 - filter$	3.4%	0.074s	4.6%	0.25s
	$s_5 - filter$	3.3%	0.075s	4.5%	0.23s
	$Switching - filter$	2.9%	0.09s	4.2%	0.27s
	$Standard\ filter$	2.2%	0.075s	4%	0.23s

Figures 6.15(a), 6.15(b), 6.16(a), 6.16(b) show a localization example in the experimental framework.

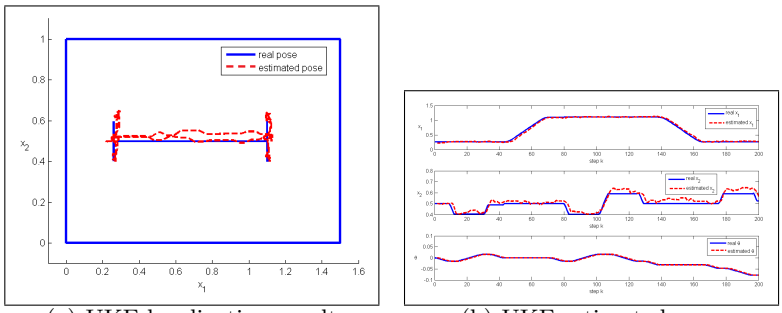


(a) EKF localization result



(b) EKF estimated pose

Fig. 6.15: Experimental result using the EKF along with the observations effect maximization switching logic, rectangle trajectory



(a) UKF localization result (b) UKF estimated pose

Fig. 6.16: Experimental result using the UKF along with the Observations effect maximization switching logic, I-like trajectory

The proposed sensors switching policy shows to be effective for both filters, always giving a value of the index ε which is better than the ones given by any fixed sensor, and close to the one obtained when all sensors are used.

6.6.2 Mobile robots localization in a totally unknown environment

Assuming the robot placed in the environment shown in Figure 6.4, two sets of 500 numerical simulations have been performed. In the first set the robot follows the rectangle trajectory while in the second set it performs the I-like trajectory. Moreover 100 real experiments have been performed placing the robot Khepera III into the unknown rectangular environment shown in Figure 6.6 and letting it perform the I-like trajectory.

In both the numerical simulations and the experimental tests the Neighbors based Extended Kalman filter (NEKF) and the Neighbors based Unscented Kalman filter (NUKF) (see Section 2.6.1 for more details about the neighbors based algorithm used into the proposed modified Extended and Unscented Kalman filters) have been used to compare their performance in terms of ε and τ indexes. The Neighbors based algorithm parameter R_{NBA} has been chosen as $R_{NBA} = 0.1m$.

Tables 6.3 and 6.4 show the obtained averaged indexes ε and τ, over the performed tests, using the NEKF and the NUKF in the simulative case and in the real experimental framework respectively.

Table 6.3: ε index and τ index - totally unknown environment - simulative framework case

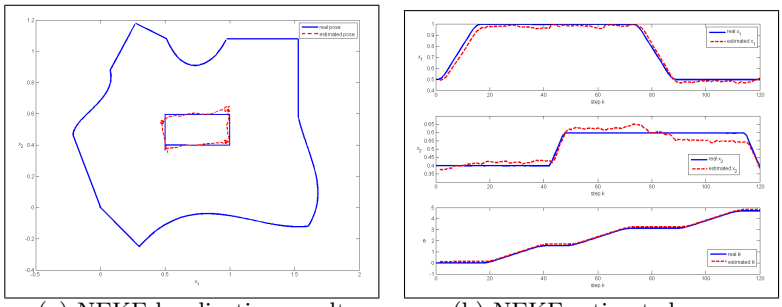
	NEKF		NUKF	
Trajectory	averaged ε	averaged τ	averaged ε	averaged τ
Rectangle	7%	0.014s	7%	0.09s
I-like	7.5%	0.0145s	7%	0.09s

Table 6.4: ε index and τ index - totally unknown environment - real framework case

	NEKF		NUKF	
Trajectory	averaged ε	averaged τ	averaged ε	averaged τ
I-like	4%	0.05s	10%	0.13s

Tables 6.3 and 6.4 show that the proposed Neighbors based algorithm allows to obtain good localization results also with no assumptions on the robot surrounding environment and with no knowledge about the environment shape. Moreover, as in the case the environment is perfectly known, performed simulations/experiments show that the EKF and the UKF perform comparably well in reconstructing the robot pose also in their neighbors based versions.

Figures 6.17(a), 6.17(b), 6.18(a), 6.18(b) show a localization example in the simulative framework while Figures 6.19(a), 6.19(b), 6.20(a), 6.20(b) show a localization example in the real framework.



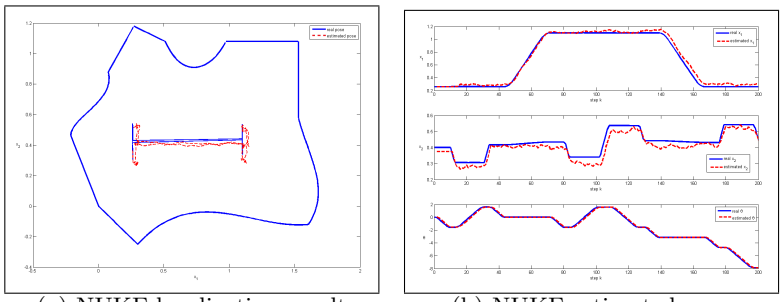
(a) NEKF localization result (b) NEKF estimated pose

Fig. 6.17: Numerical result using the NEKF, rectangle trajectory

Localization using the Switching logic based on the Observations Effect Maximization

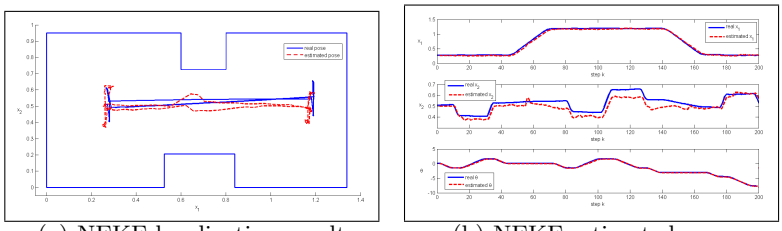
The mobile robot localization problem in a totally unknown environment will be now faced using the observations effect maximization switching rule proposed in Chapter 4 along with the NEKF and the NUKF. Once again 500 numerical simulations and 100 real experiments have been performed and three kinds of estimation algorithms have been used:

- Estimation algorithm (NEKF/NUKF) using a single sensor kept fixed along all the path. This case will be referred as $S_j - filter$ for the case where the j-th sensor is used.



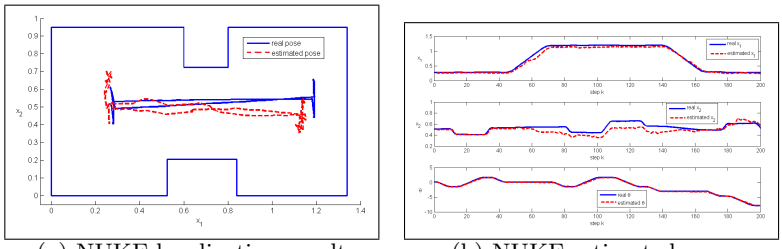
(a) NUKF localization result (b) NUKF estimated pose

Fig. 6.18: Numerical result using the NUKF, I-like trajectory



(a) NEKF localization result (b) NEKF estimated pose

Fig. 6.19: Experimental result using the NEKF, I-like trajectory



(a) NUKF localization result (b) NUKF estimated pose

Fig. 6.20: Experimental result using the NUKF, I-like trajectory

- Estimation algorithm (NEKF/NUKF) using at each step only one sensor, choosing it through the proposed observations effect maximization switching rule. This case will be referred as *Switching-filter*.
- Traditional estimation algorithm (NEKF/NUKF) based on the use of all the $n_S = 5$ available sensors. This case will be referred as *Standard filter*.

Tables 6.5 and 6.6 show the results in the performed simulations and experiments.

Also in this case, the proposed sensors switching policy shows to be effective for both filters, always giving a value of the index ε better than the

Table 6.5: ε index and τ index - totally unknown environment - simulative framework case

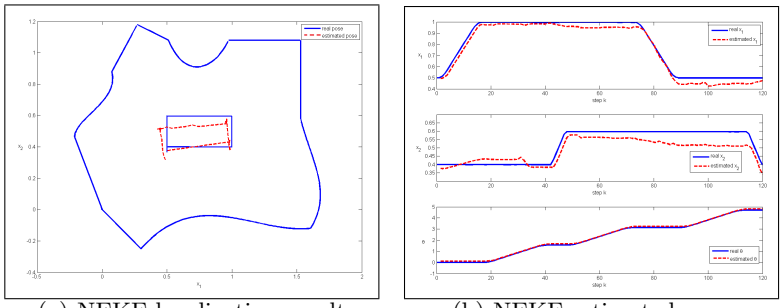
		NEKF		NUKF	
Trajectory	Filter	averaged ε	averaged τ	averaged ε	averaged τ
Rectangle	$s_1 - filter$	9%	0.0115s	8%	0.083s
	$s_2 - filter$	8%	0.0115s	10%	0.085s
	$s_3 - filter$	10%	0.0114s	11%	0.082s
	$s_4 - filter$	8%	0.0113s	16%	0.084s
	$s_5 - filter$	10%	0.0113s	12%	0.087s
	$Switching - filter$	7.8%	0.013s%	7.5%	0.09s
	$Standard\ filter$	7%	0.014s	7%	0.09s
I-like	$s_1 - filter$	10%	0.0113s	8%	0.083s
	$s_2 - filter$	8.5%	0.0113s	10%	0.085s
	$s_3 - filter$	9%	0.0114s	11%	0.082s
	$s_4 - filter$	9%	0.0113s	16%	0.084s
	$s_5 - filter$	10%	0.0113s	12%	0.087s
	$Switching - filter$	8%	0.013s%	7.5%	0.09s
	$Standard\ filter$	7.5%	0.0145s	7%	0.09s

Table 6.6: ε index and τ index - totally unknown environment - real framework case

		NEKF		NUKF	
Trajectory	Filter	averaged ε	averaged τ	averaged ε	averaged τ
I-like	$s_1 - filter$	9%	0.03s	12%	0.05ss
	$s_2 - filter$	10%	0.03s	11.5%	0.05s
	$s_3 - filter$	8%	0.03s	12%	0.07s
	$s_4 - filter$	10%	0.02s	12%	0.08s
	$s_5 - filter$	7%	0.03s	13%	0.09s
	$Switching - filter$	5%	0.04s%	11%	0.11s
	$Standard\ filter$	4%	0.05s	10%	0.13s

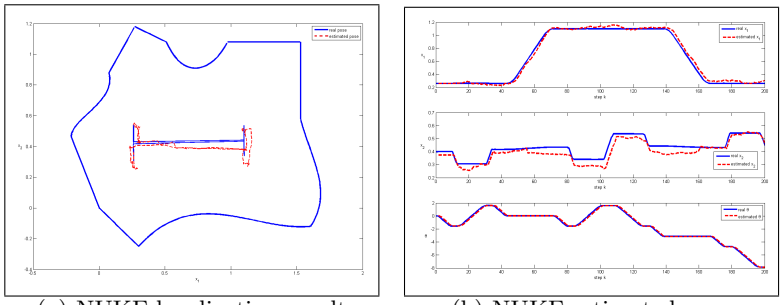
ones given by any fixed sensor and close to the one obtained when all sensors are used. Looking at the time performance, as shown in Table 6.5, the averaged computation time required by the proposed switching strategy is a little higher than the time required by the single fixed sensor filtering strategies due to the computations required by the switching policy. However, as shown by τ index values, all the computation times are of the same order of magnitude. In conclusion, numerical simulations and real experiments show the effectiveness of the proposed switching rule since it provides estimation results close to the results due to the use of all the available sensors but with a more efficient sensors use and with very low additional computation effort.

Figures 6.21(a), 6.21(b), 6.22(a), 6.22(b) show a localization example in the simulative framework while Figures 6.23(a), 6.23(b), 6.24(a), 6.24(b) show a localization example in the real framework.



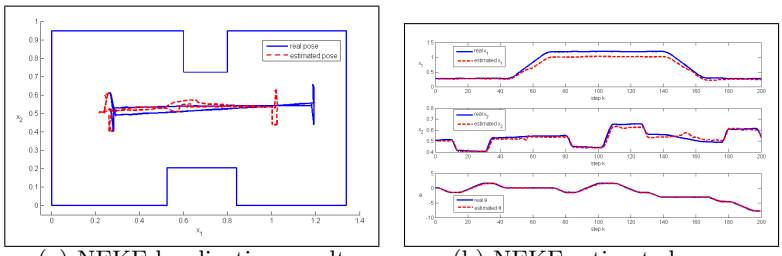
(a) NEKF localization result (b) NEKF estimated pose

Fig. 6.21: Numerical result using the NEKF along with the observations effect maximization switching logic, rectangle trajectory



(a) NUKF localization result (b) NUKF estimated pose

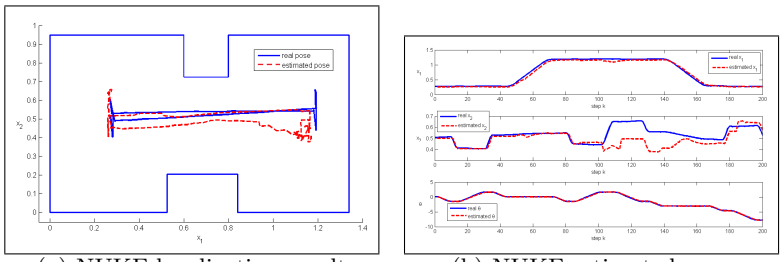
Fig. 6.22: Numerical result using the NUKF along with the observations effect maximization switching logic, I-like trajectory



(a) NEKF localization result (b) NEKF estimated pose

Fig. 6.23: Experimental result using the NEKF along with the observations effect maximization switching logic, I-like trajectory

Remark In conclusion, both in the case the environment is perfectly known or it is totally unknown, using the Extended Kalman filter (EKF or NEKF) or the Unscented Kalman filter (UKF or NUKF), the same localization performance are obtained. As a consequence, in the following, only one of the



(a) NUKF localization result (b) NUKF estimated pose

Fig. 6.24: Experimental result using the NUKF along with the observations effect maximization switching logic, I-like trajectory

two Kalman filter extensions will be used assuming that using the other one, analogous results would be obtained. With no loss of generality, the Extended Kalman filter (EKF or NEKF) will be hereafter used.

Moreover only the case the environment is totally unknown will be considered since it is the more realistic case in mobile robots applications and since the Neighbors based algorithm shows to be effective enough to allow mobile robot localization also with no information about the robot surrounding environment.

6.6.3 Localization using the Incidence Angle based Switching logic

In this Section the performance of the Neighbors based Extended Kalman filter will be tested using the incidence angle based switching rule proposed in Chapter 4. The above switching rule can be used if the robot is equipped with ultrasonic sensors. Since this is the case of the robot Khepera III, the incidence angle based switching policy can be tested.
First of all, the sensor model proposed in (4.4) has been validated. To this end, the noise covariance function $V(r)$ has been estimated through a series of experiments, each one related to a different value of the incidence angle, from 90° to 60°. For each angle value, a set of 1000 measurements has been taken by the S_3 sensor on the Khepera III, and the corresponding standard deviation has been estimated. The obtained results are shown in Figure 6.25. Using the obtained data, the standard deviation of the incidence function $\Xi(r)$ can be estimated through a LMS approximation of the values by a polynomial curve. As it can be seen, such standard deviation function can be well approximated by a second order polynomial function. Such function is monotonic increasing in r and thus the experimental data confirm the correctness of the model (4.4) used for the ultrasonic sensors.

At this point, to evaluate the performance of the proposed sensors switching rule two sets of 100 real experiments have been performed placing the Khepera III into the environment shown in Figure 6.6. The robot performed the I-like trajectory and the rectangle trajectory.

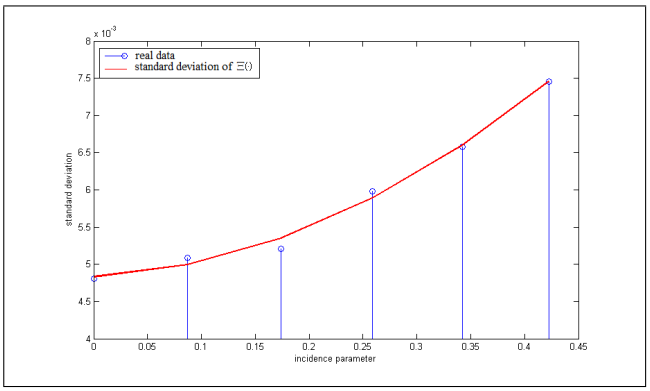


Fig. 6.25: Standard deviation function estimation

In each experiment, three different filters have been tested. The first one uses two sensors S_i, S_j, keeping them fixed along the path; this test has been repeated for all the $\binom{5}{2} = 10$ possible sensors configurations. The second one uses two sensors ($q = 2$) out of the five available, switching between sensors using the incidence angle based switching policy shown in Section 4.3. The third one uses all five sensors together.

In Table 6.7 the results are reported in terms of the ε index; the values are averaged over the 100 experiments. In this table:

- z_i indicates a generic pair of sensors among the $\binom{5}{2}$ available;
- Switching refers to the proposed switching policy with $q = 2$;
- All refers to the case where all sensors are used together-

Table 6.7: Averaged ε index over the performed 100 experiments

Trajectory	***I*-like**	**Rectangle**
Switching	8%	12%
z_1	10%	14%
z_2	21%	15%
z_3	13%	14%
z_4	17%	17%
z_5	16%	13%
z_6	10%	13%
z_7	15%	17%
z_8	16%	15%
z_9	21%	17%
z_{10}	12%	13%
All	4%	10%

As shown by the results, the proposed incidence angle based switching rule provides very good estimation results for both the trajectories yielding to a very low estimation error ε. Moreover the averaged tau index, over the performed experiments, using the z_i based localization algorithm is $\tau = 0.03s$ for the rectangle trajectory and for the I-like trajectory while the index value using the incidence angle based switching rule is $\tau = 0.035s$ and using all the sensors it is $\tau = 0.036s$, for both trajectories. The above results show the computational efficiency of the proposed switching rule since the filter based on the above rule does not require a too high additional computation time w.r.t. the filter based on a fixed set of used sensors (z_i filters).

To further test the proposed switching rule, a set of 1000 numerical simulations, based on the same parameters and trajectories used in the experimental setting, has been performed. Simulation tests show the effectiveness of the proposed switching policy. In the performed numerical tests, the filter based on the switching policy and $q = 2$ sensors performed always better than each other possible fixed pair of sensors.

6.6.4 Conclusions about switching logics

As shown in the previous Sections, using the proposed switching logics, the localization results do not degrade w.r.t using all the available sensors. As a consequence, if the application requirements impose an energy saving use of the available sensors, using one of the two proposed sensors switching rules could help efficiently solving the problem. The used switching rule has to be chosen depending on the used localization algorithm, on the available computation power and on the used sensors types. As previously remarked, if a neighbors based localization algorithm (NEKF or NUKF) is used starting from sensors measurements, the additional computational effort due to the use of the incidence angle based switching rule is very low and acceptable also if the available computational power is low too.

If the used localization algorithm is not neighbors based, than information on the incidence angle has to be extracted starting from the available information about the robot surrounding environment. This step could result in higher computational costs w.r.t using the observations effect maximization switching rule, especially if the number of available sensors is low enough. In this case, observations effect maximization switching rule could be used along with the chosen localization algorithm. Moreover, if the available sensors are not sonar sensors, the incidence angle based switching rule can not be used since it is based on the sonar sensors physical features.

In conclusion, depending on the application, one of the two switching rules can be used with no appreciable degradation on the localization results. Due to the above considerations, in the following the localization module will always be implemented and tested using all the available sensors understanding that, if required, one of the proposed switching rules could be used instead.

6.6.5 Localization using a Mixed Kalman Filter

In this Section the localization performance of the Mixed Extended Kalman filter (MEKF) proposed in Section 2.7.2 will be tested placing the robot in the real environment shown in Figure 6.6 and acquiring measurements from the Khepera III on board sensors and from $q = 4$ out of board sensors placed in the corners of the environment bounds. More precisely the mobile robot has been equipped with an ultrasonic transmitter and four ultrasonic receivers have been placed on the environment boundaries corners (as shown in [28] and in Figure 6.26). The measurement noise covariance matrix related to the out of board sensors is $V^2 = 0.02^2 I_4$.

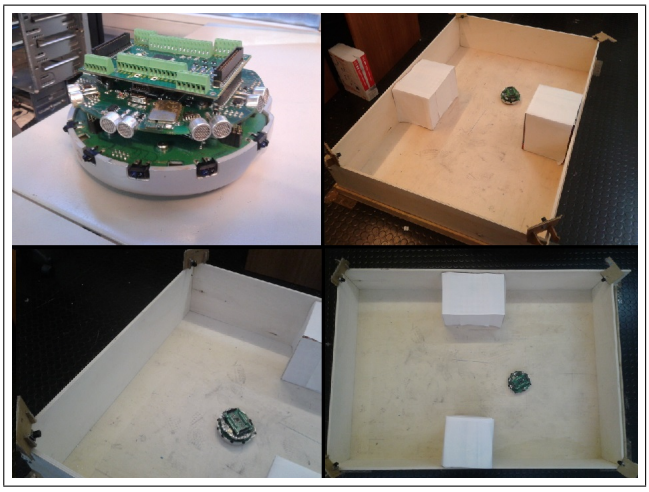


Fig. 6.26: Experimental framework with out of board sensors

The following filters have been tested: an Extended Kalman filter (EKF) based on the entire output formed by on board and out of board sensors measurements; the proposed Mixed Extended Kalman filter (MEKF) with various values of $M_\alpha = diag\{\alpha_1, \alpha_2, \alpha_\theta\}$; the Neighbors based EKF (NEKF) and the Out of board sensors based EKF (OEKF).

In each experiment, the robot performs the I-like trajectory and Table 6.8 shows the averaged ε values over the 50 performed experiments.

As shown by Table 6.8, the proposed $MEKF$ algorithm performance is satisfactory, whatever is the M_α value. In particular, using $M_\alpha = diag\{0.1,\ 0.1,\ 0.9\}$, that is a mixed filter essentially based on the OEKF for the position prediction and on the NEKF for the heading prediction, the proposed $MEKF$ algorithm performs better than the other filters (EKF, OEKF, NEKF). This result is consistent with the previously described NEKF and OEKF properties. Since the NEKF is better than the OEKF to

Table 6.8: averaged ε index, over the 50 experiments

filter	ε
MEKF $M_\alpha = diag\{0.1, 0.1, 0.1\}$	9.02%
MEKF $M_\alpha = diag\{0.5, 0.5, 0.5\}$	9%
MEKF $M_\alpha = diag\{0.9, 0.9, 0.9\}$	9.5%
MEKF $M_\alpha = diag\{0.9, 0.9, 0.1\}$	12%
MEKF $M_\alpha = diag\{0.1, 0.1, 0.9\}$	8.8%
NEKF	11.2%
OEKF	9.3%
EKF	8.9%

predict robot heading, while the OEKF is more reliable than the NEKF w.r.t. the robot position prediction, using a low value of α_1, α_2 and a high value of α_θ the MEKF performance increases.

A typical result of the MEKF algorithm, using $M_\alpha = diag\{0.1, 0.1, 0.9\}$, is shown in Figure 6.34.

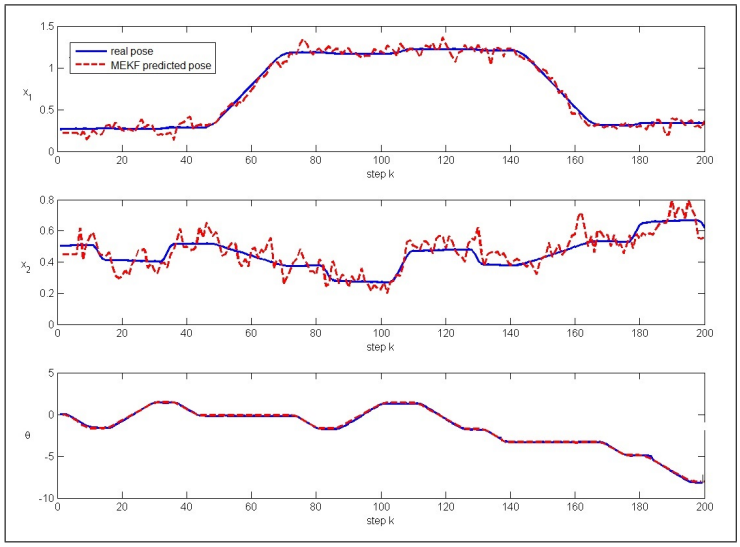


Fig. 6.27: MEKF experimental results using $M_\alpha = diag\{0.1, 0.1, 0.9\}$

6.6.6 Conclusions about localization

In conclusion, all the proposed mobile robot localization techniques have shown to be effective in estimating the robot pose. The techniques have been tested in both numerical and experimental settings and the results are very

satisfactory and encouraging both if the robot surrounding environment is perfectly known or if it is totally unknown. In the next Sections, some of the proposed mobile robot localization algorithms will be used as the localization module in the Simultaneous Localization and Mapping scheme shown in Figure I.7.

6.7 SLAM results

In this Section the numerical and experimental results about the three SLAM techniques proposed in Chapter 3 will be shown. The following numerical simulations and experimental tests have been performed:

- 100 numerical simulations placing the robot into the single room environment shown in Figure 6.4 and letting it perform the rectangle trajectory;
- 100 numerical simulations placing the robot into the three-rooms environment shown in Figure 6.5 and letting it perform the L-like trajectory;
- 50 real experiments placing the robot Khepera III into the real framework shown in Figure 6.6 and letting it perform the I-like trajectory.

6.7.1 Segment based SLAM (SbSLAM) results

The SbSLAM algorithm proposed in Section 3.4 changes its properties depending on its two main parameters δ and σ. In all the performed simulations the SbSLAM parameter $\delta = 0.1$ has been used and the SLAM algorithm performance evolution depending on the σ parameter has been studied. To this end, for each simulation, four possible values of the SbSLAM parameter σ have been tested: 0.06, 0.08, 0.1, 0.12. The obtained averaged ϵ, γ, τ indexes over the 100 simulations are shown in Tables 6.9 and 6.10 for the single room simulations and for the three-rooms simulations respectively. To clearly show the σ parameter influence on the algorithm performance, the values contained in Table 6.10 are also depicted in Figure 6.28[1].

Table 6.9: Averaged indexes over the 100 performed simulations in the single room environment

index	$\sigma = 0.06$	$\sigma = 0.08$	$\sigma = 0.1$	$\sigma = 0.12$
ε	0.79%	0.85%	4.8%	9.1%
γ	4.2%	4.6%	4.7%	6.7%
τ	$0.47s$	$0.35s$	$0.21s$	$0.17s$

[1] Analogous results would be obtained if the values in Table 6.9 were plotted.

Table 6.10: Averaged indexes over the 100 performed simulations in the three-rooms environment

index	$\sigma = 0.06$	$\sigma = 0.08$	$\sigma = 0.1$	$\sigma = 0.12$
ε	1.09%	1.21%	1.46%	4.16%
γ	5.76%	6.15%	7.83%	11.68%
τ	$1.35s$	$0.96s$	$0.85s$	$0.71s$

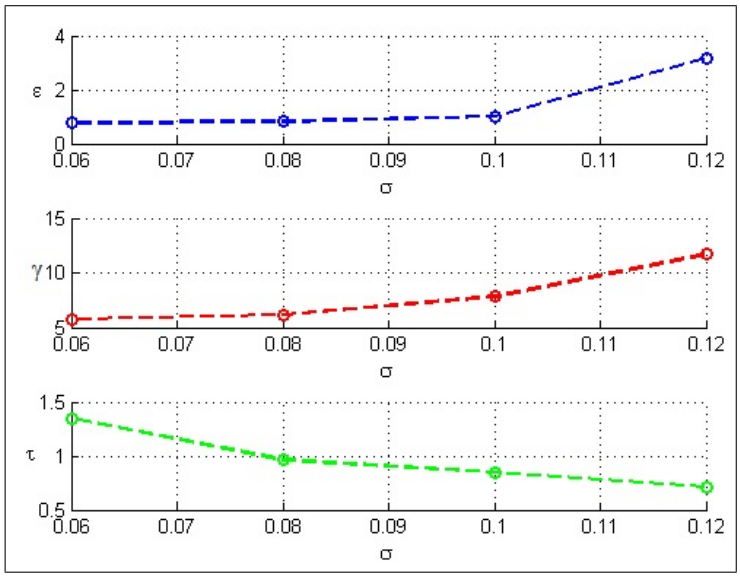
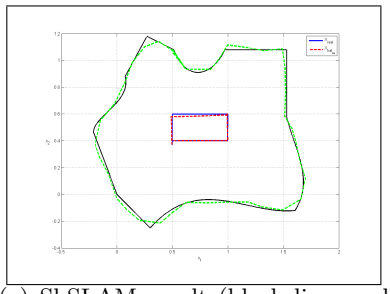
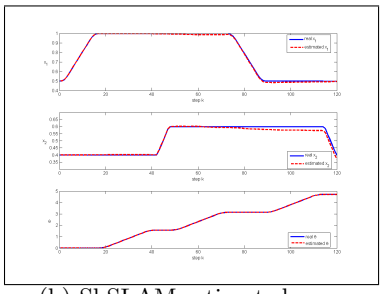


Fig. 6.28: $\varepsilon, \gamma, \tau$ evolution changing σ values

Tables 6.9 and 6.10 show that the performance of the proposed Segment based SLAM algorithm is satisfying using all the tested σ values since the algorithm yields to low localization and mapping errors. Moreover, choosing $\sigma \geq 0.08$, τ index is always such that the algorithm can be used during the robot evolution with no delays troubles since the algorithm execution time is lower than the chosen sampling time $T = 1s$. As shown in Figure 6.28, when σ tends to 0, the estimation and mapping errors decrease and τ index grows; as σ grows, the estimation and mapping performance are deteriorated while the time performance increases. A typical result of the SLAM algorithm, using $\sigma = 0.08$, is depicted in Figures 6.29 and 6.33 for the single room simulations and for the three-rooms simulation respectively.
In the experimental setup $\sigma = 0.08$ has been used for the SbSLAM landmark extraction and data association step. The obtained averaged indexes over the performed 50 real experiments are: $\varepsilon = 5.1\%$, $\gamma = 12\%$, $\tau = 0.41s$ and also in this case they are very encouraging and the time constraints are satisfied. A

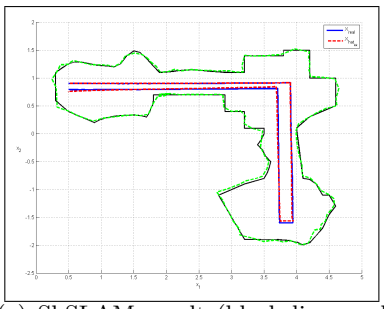


(a) SbSLAM result (black line: real environment, green line: estimated environment)

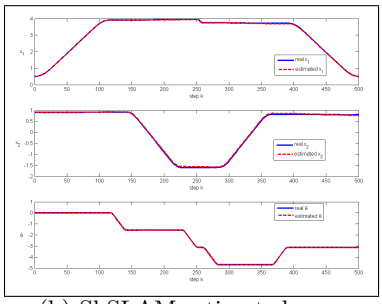


(b) SbSLAM estimated pose

Fig. 6.29: SbSLAM simulation result (single room, $\sigma = 0.08$)



(a) SbSLAM result (black line: real environment, green line: estimated environment)



(b) SbSLAM estimated pose

Fig. 6.30: SbSLAM simulation result (three-rooms, $\sigma = 0.08$)

typical result of the SLAM algorithm in a real experiment is shown in Figure 6.34.

6.7.2 Polynomial based SLAM (PbSLAM) results

The polynomial based SLAM algorithm described in Section 3.5.1 has been tested using $\xi = \frac{1}{3}$, $\epsilon_{TH} = 0.05$, $\rho = \frac{1}{2}$, $m = 3$, $K_s = 10$, $R = 0.1m$, $N_p = 15$ for the landmark extraction and data association steps.

The obtained averaged $\varepsilon, \tau, \gamma$ indexes over the 100 numerical simulations are $\varepsilon = 3.5\%$, $\gamma = 3\%$, $\tau = 5.1s$ and $\varepsilon = 3.4\%$, $\gamma = 3\%$, $\tau = 8.65s$, for the rectangular trajectory in the single room environment and for the L-like trajectory into the three-rooms environment respectively. A typical result of the SLAM algorithm is depicted in Figures 6.32 and 6.33.

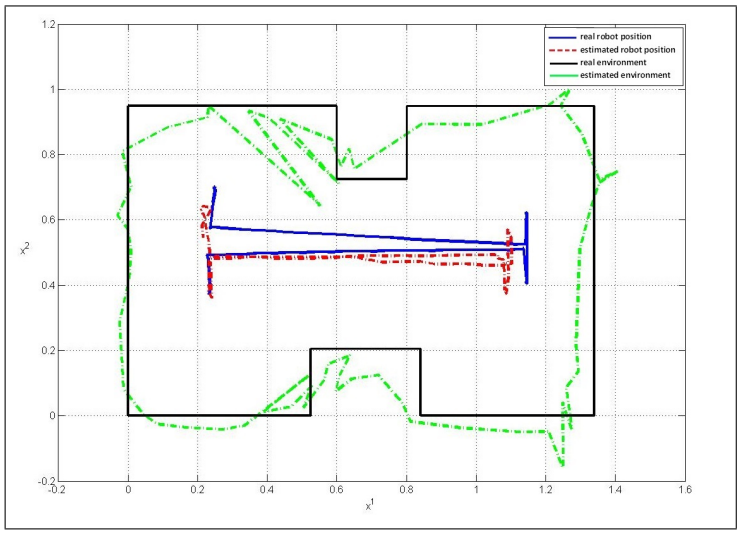
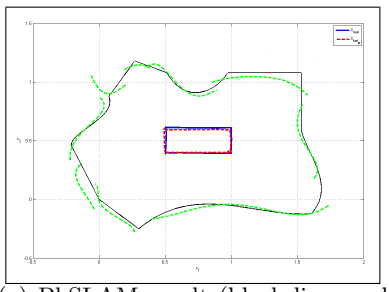
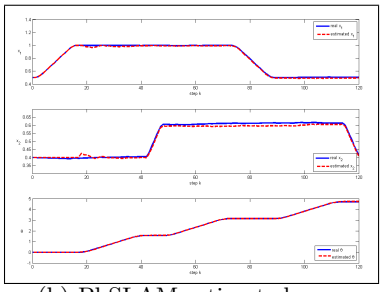


Fig. 6.31: Segment based SLAM results in a real experiment



(a) PbSLAM result (black line: real environment, green line: estimated environment)

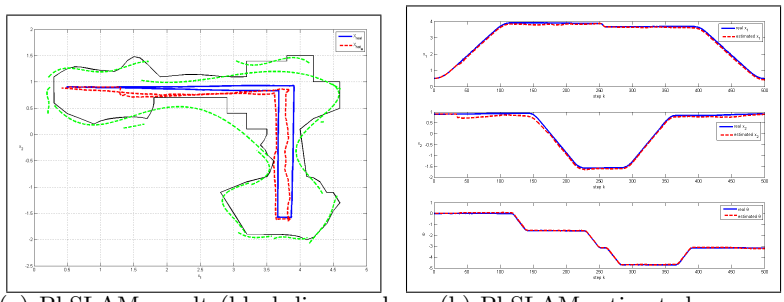


(b) PbSLAM estimated pose

Fig. 6.32: PbSLAM simulation result (single room)

In the experimental framework, the obtained averaged indexes over the 50 experiments are $\varepsilon = 18\%$, $\gamma = 5\%$, $\tau = 5s$. A typical result of the SLAM algorithm is depicted in Figure 6.34.

As shown by the numerical and experimental results, the proposed polynomial based mapping is really consistent with the real environment boundaries. Comparing the indexes values with the results by the SbSLAM, the PbSLAM algorithm yields to better mapping results but it requires an averaged computation time one order of magnitude greater than the one required by the SbSLAM.



(a) PbSLAM result (black line: real environment, green line: estimated environment)

(b) PbSLAM estimated pose

Fig. 6.33: PbSLAM simulation result (three-rooms)

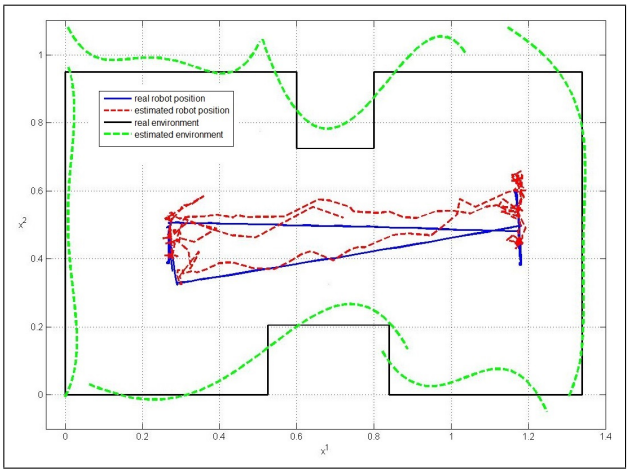


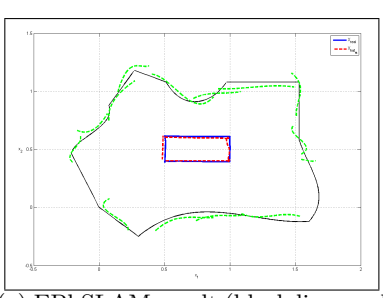
Fig. 6.34: PbSLAM real result, black line: real environment, green line: estimated environment

6.7.3 Efficient Polynomial based SLAM (EPbSLAM) results

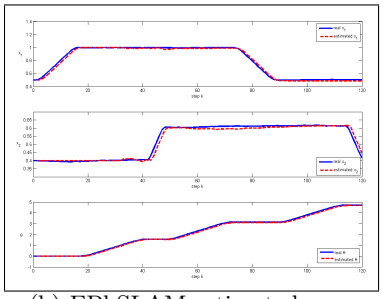
The efficient polynomial based SLAM algorithm described in Section 3.5.4 has been tested using $\rho_m = 0.05m$, $\sigma_m = 0.25m$, $\varepsilon_M = 10$, $\varepsilon_z = 20$, $n_a = \varepsilon_M$, $m = 3$, $R = 0.1m$ for the landmark extraction and data association steps.

The averaged performance indexes over the simulation tests are $\varepsilon = 3\%$, $\gamma = 3.5\%$, $\tau = 0.12s$ and $\varepsilon = 2.5\%$, $\gamma = 5\%$, $\tau = 0.58s$ for the single room case and for the three-rooms case respectively.
A typical result of the SLAM algorithm is depicted in Figures 6.35 and 6.36 .

In the experimental setup the obtained averaged indexes, over the 50 experiments, are $\varepsilon = 11\%$, $\gamma = 8\%$ and $\tau = 0.68s$. A typical result of the EPbSLAM algorithm is depicted in Figure 6.37.

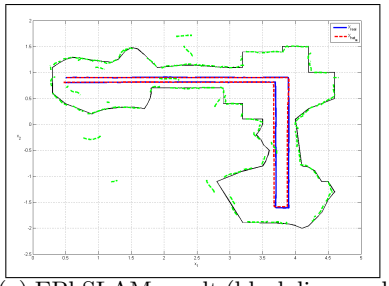


(a) EPbSLAM result (black line: real environment, green line: estimated environment)

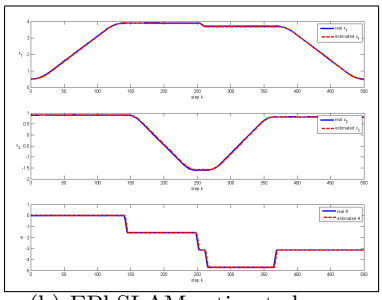


(b) EPbSLAM estimated pose

Fig. 6.35: EPbSLAM simulation result (single room)



(a) EPbSLAM result (black line: real environment, green line: estimated environment)



(b) EPbSLAM estimated pose

Fig. 6.36: EPbSLAM simulation result (three-rooms)

As shown by the numerical and experimental tests, the proposed EPbSLAM is effective in localizing the robot and building a map of its surrounding environment requiring a very low computation time.

6.7.4 Conclusions about SLAM

Numerical simulations and experimental tests have shown the effectiveness of the proposed three SLAM algorithms in facing both the mobile robot localization problem and the surrounding environment mapping problem. All the proposed SLAM techniques provide very satisfactory results in terms of the mapping error index γ and localization error index ε.
Tables 6.11, 6.12 and 6.13 summarize the results obtained using the three proposed SLAM techniques.

As shown in Tables 6.11, 6.12 and 6.13, looking at the localization results (index ε), the SbSLAM yields to the lowest index values, however this algo-

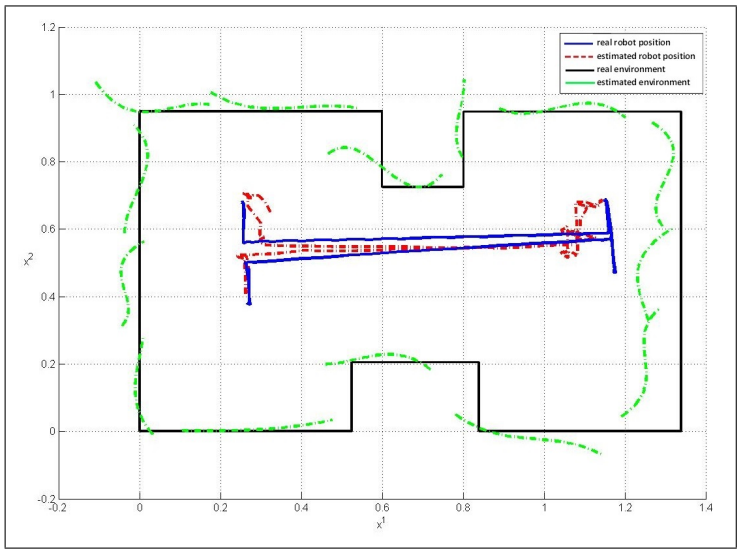


Fig. 6.37: Real result using the EPbSLAM algorithm

Table 6.11: Averaged indexes over the 100 performed simulations in the single room environment

	ε	γ	τ
$SbSLAM(\sigma = 0.08)$	0.8%	4.6%	$0.35s$
$PbSLAM$	3.5%	3%	$5.1s$
$EPbSLAM$	3%	3.5%	$0.12s$

Table 6.12: Averaged indexes over the 100 performed simulations in the three-rooms environment

	ε	γ	τ
$SbSLAM(\sigma = 0.08)$	1%	6.15%	$0.96s$
$PbSLAM$	3.4%	3%	$8.65s$
$EPbSLAM$	2.5%	5%	$0.58s$

rithm yields to the worse mapping performance. The PbSLAM shows to be the most effective in mapping the robot environment but its required computation time is one order of magnitude bigger than the ones required by the other two algorithms. In particular, both the SbSLAM and the EPbSLAM algorithms can be used in *real time*, in the proposed experimental framework, since their averaged computation times are lower than the chosen sampling time $T = 1s$.

Table 6.13: Averaged indexes over the 50 performed experiments

	ε	γ	τ
$SbSLAM(\sigma = 0.08)$	5.1%	12%	$0.41s$
$PbSLAM$	18%	5%	$5s$
$EPbSLAM$	11%	8%	$0.68s$

The localization results using the EPbSLAM algorithm are quite the same of the results obtained thanks to the SbSLAM algorithm, but the proposed efficient polynomial based SLAM is better than the SbSLAM in terms of the mapping results. In conclusion:

- the SbSLAM algorithm provides the best localization results, it is very computationally efficient but it provides the worse mapping results due to the use of a very simple mapping model (segment based).
- the PbSLAM algorithm is the best algorithm in terms of mapping performance but it can not be used in real time during robot motion due to its required high computational cost; more precisely, the high versatility of the points clustering based polynomial mapping model requires high computational costs to find and update the environment map;
- the EPbSLAM algorithm is not the best in terms of mapping or localization performance but it provides good results requiring low computational cost. This algorithm has proven to be a right compromise between the high mapping performance of the PbSLAM and the high computation time performance of the SbSLAM.

The above considerations are summarized in Table 6.14 where the pointing up arrows represent optimal results, the pointing right-up arrows stay for very good results, the pointing right arrows denote good results and the pointing down arrows represent bad results.

Table 6.14: Overall results provided by SbSLAM, PbSLAM and EPbSLAM

algorithm	localization	mapping	time
SbSLAM	↑	→	↑
PbSLAM	↑	↑	↓
EPbSLAM	↑	↗	↗

6.8 PnP problem using IMUs

To assess the performance of the proposed algorithm, several numerical simulations and real experiments have been performed. The following three algorithms have been compared:

1. the method (5.31) proposed in this paper, implemented through the Nelder-Mead algorithm (Matlab *fminsearch*) and hereafter denoted as α-PnP;
2. the efficient PnP algorithm proposed in [70], denoted as e-PnP, in the implementation provided by the *Machine Vision Toolbox for MATLAB* [71].
3. the PnP with IMUs method proposed in Section 5.4.2. In this implementation, the whole rotation matrix is provided by IMUs while the translation vector is computed using (5.10). Hereafter this algorithm will be denoted as IMU-PnP.

To compare the results obtained by the three algorithms the relative error index defined by 6.7 has been used.

6.8.1 Numerical Simulations

To evaluate the performance of the proposed solutions to the PnP problem, a set of simulations have been performed. In all the simulations, different features with points randomly generated in a box $[-0.2, 0.2]m \times [-0.2, 0.2]m \times [-0.2, 0.2]m$ have been used. To ensure the existence of at least three non collinear feature points, the first three points of each feature are always $P_1 = (0, 0, 0)m$, $P_2 = (0.1, 0.1, 0)m$ and $P_3 = (0.1, 0, 0)m$. For each test, the relative rotation and translation between the camera and the object reference frames have been randomly generated in a box $[-0.5, 0.5]m \times [-0.5, 0.5]m \times [0.5, 2.5]m$.

In all numerical tests, a camera with $\tilde{f} = 800$ pixels, resolution $[640 \times 480]$ and center $C_I = (320, 240)$ has been simulated.
The data acquired by the IMUs and the image provided by the camera are affected by Gaussian noise with zero mean and standard deviation of:

- $\sigma_g = I_3 * 0.01m$ for $\hat{g}_c$ and $\hat{g}_o$;
- $\sigma_{pixel} = 5$ pixels in the image;
- $\sigma_{mag} = 8^\circ$ on the measurements provided by the magnetometer.

The performance of the algorithms have been tested with different values of the number n of feature points, starting from $n = 4$ up to $n = 200$. Each set of feature points has been tested in 300 different configurations. Figure 6.38 shows the averaged value of the ε index as n changes.

Note that the α-PnP algorithm proposed in this paper performs better than the e-PnP and the IMU-PnP algorithms, yielding always to a lower value of the ε index. Moreover note that, after a certain number of points (about 40), to add further points to the feature does not increase significantly the quality of the solution.

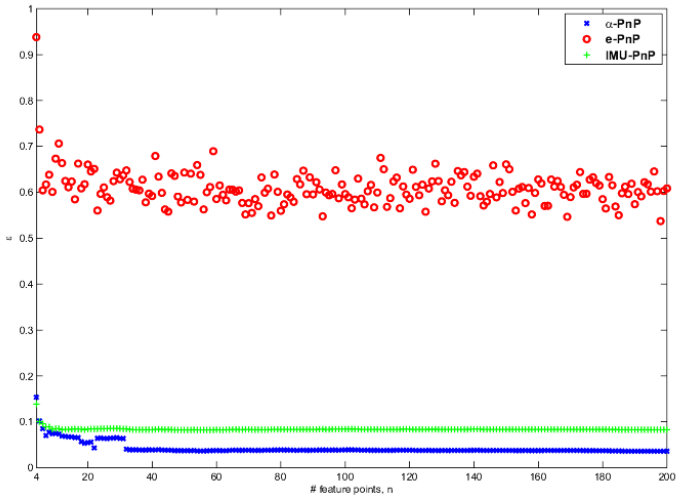


Fig. 6.38: Averaged ε index over 300 configurations with $n = 4, 5, \ldots, 200$ feature points

6.8.2 Real Experiments

To experimentally test the proposed PnP solutions, the following experimental setting has been used:

- A Logitech C310 HD webcam with resolution 1280×960 [64]. The intrinsic parameters of the camera have been estimated using the Camera Calibration Toolbox [65].
- Two ArduIMU V3 Inertial Measurement Units [66].
- The four squares feature shown in Figure 6.39. Each corner of the squares can be used as a feature point.

A set of experiments has been performed using the following procedure

1. the camera and the object have been placed in an unknown configuration, with the object in the field of view of the camera, and an estimation $\hat{R}_{t,1}$ of the actual transformation matrix $R_{t,1}$ has been computed using the α-PnP, the IMU-PnP and the e-PnP algorithms.
2. The object has been rotated and translated of a known transformation matrix $R_{t,2}$.
3. An estimation of $R_{t,2}$, namely $\hat{R}_{t,2}$, has been obtained.
4. An estimation of the displacement matrix between the two configurations has been computed as $\hat{R}^2_{t,1} = (\hat{R}_{t,1})^{-1}\hat{R}_{t,2}$.
5. The estimation $\hat{R}^2_{t,1}$ has been compared with the real $R^2_{t,1}$.

Figure 6.40 shows two pictures taken during the experiments. The perfor-

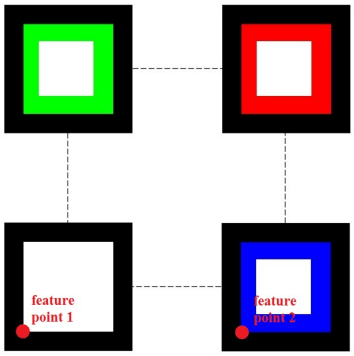


Fig. 6.39: The four squares feature

Fig. 6.40: Pictures taken by the camera in the configuration $R_{t,1}$ (right) and $R_{t,2}$ (left)

mance of the above algorithms have been contrasted through the proposed index ε using $n = 4, \ldots, 16$ feature points. A set of 42 experiments has been performed changing the relative rotation and translation between the camera and the object. The ε index has been computed over the obtained $R^2_{t,1}$ and $\hat{R}^2_{t,1}$. The resulting averaged ε index is $\varepsilon = 29\%$ for the e-PnP algorithm, $\varepsilon = 34\%$ for the IMU-PnP algorithm and $\varepsilon = 24\%$ for the α-PnP algorithm. Again the α-PnP algorithm presented in this paper performs in the average better than the e-PnP and the IMU-PnP algorithms.

Conclusions

In this thesis a set of new algorithms to solve the mobile robots localization and mapping problems have been proposed.

All the proposed algorithms have been developed looking at obtaining good localization/mapping results and simultaneously minimizing the required computational costs, so that the proposed solutions can be used in *real time* during the robot motion.

Main results

Localization

The localization problem has been faced in the static and in dynamic contexts. In the static contexts, the problem has been solved by means of a camera and two inertial measurement units. More precisely, the mobile robot localization problem has been translated into a Perspective-n-Point problem and the inertial measurement units have been used to help the vision system solve the PnP problem. In the dynamic contexts the Kalman filter theory has been suitably adapted to the mobile robot localization problem in the cases: (1) the robot surrounding environment is perfectly known; (2) only a few information is known about the robot environment and (3) no information is available about the environment.

In particular, in the case the environment is unknown, a measurements closeness strategy has been developed resulting in the Neighbors based Algorithm (NBA) and in the Neighbors based Extended and Unscented Kalman filters (NEKF and NUKF). All these solutions have shown to be effective in localizing the robot with no required assumptions about its surrounding environment.

Finally, strategies to localize the robot efficiently using the available sensors have been proposed. Two sensors switching logics to allow robot batteries

saving have been described and tested, showing that, depending on the application requirements, good localization results may be obtained using only a part of the available sensors. Moreover, a new sensor fusing technique has been developed looking at emphasizing the sensors qualities simultaneously overcoming their defects. The resulting Mixed Kalman filter has been studied, in terms of its properties, and tested in numerical and experimental ways, showing its effectiveness.

Mapping and SLAM

Three new mapping strategies have been proposed based on two new mapping models. The first model is a segment based model and aims to map the robot surrounding environment by means of a set of segments forming an *environment envelope*. The second model is based on a polynomial approximation of the environment boundaries. The segment based model is computationally very efficient but it could provides poor mapping performance. On the contrary, the polynomial based model is versatile in mapping the environment, yielding to good mapping results, but it could require high computational costs to obtain the environment map. These two models clearly show the trade off between map accuracy and time to obtain it. A set of heuristics has been developed to make the polynomial mapping model computationally cheaper with no excessively affecting its mapping performance. The resulting segment based mapping algorithm (first model), polynomial based mapping algorithm (second model) and efficient polynomial based mapping algorithm (second model using heuristics) have been then used to develop three SLAM techniques: Segment based SLAM, Polynomial based SLAM and Efficient Polynomial based SLAM. These three techniques have been tested through numerical simulations and real experiments showing their effectiveness in localizing the mobile robot and simultaneously providing a reliable environment mapping. In particular the results have shown that the EPbSLAM mapping performance are quite the same of the PbSLAM technique but the required computational cost is drastically reduced.

Applications

The proposed algorithms can be suitably used in all the mobile robots applications involving a robot navigation. In particular, depending on the requirements about energy saving, on the localization and mapping performance, on the available computational power, one of the proposed algorithms could be better than the remaining ones. Expertise can help to choose the appropriate solution.

Regarding the PnP problem solution using IMUs, it can be used in mobile robots applications or also in cameras systems calibration where using IMUs, the calibration performance can be improved by comparing the results by cameras with the results by cameras and IMUs.

Future research directions

The numerical and experimental results provided in this thesis are really encouraging and the proposed approaches deserve further investigations. In particular, the following future research directions have to be pointed out:

- facing the path planning problem so that the mobile robot exploration problem (localization, mapping and path planning) can be faced;
- providing further experimental results using a bigger environment and testing all the proposed localization and SLAM algorithms;
- extending the Mixed Kalman filter gains optimality proof to the case the measurements are not scalar; studying the filter stability properties;
- facing the SLAM problem for a team of mobile robots by properly extending the proposed single robot SLAM algorithms;
- providing and testing new mapping models;
- testing the proposed switching rules in the SLAM algorithms;
- extending the provided PnP algorithms to more dynamical scenarios.

References

1. M. Drumheller, "Mobile robot localization using sonar", *IEEE Trans Pattern Anal Mach Intell*, 1987 Feb ;9(2):325-32
2. Q. Yang, K. Yuan, J. Li, H. Wang, "A histogram sensor fusion method for mobile robots", *International Conference on Information Acquisition*, pp. 339-343, 21-25 June 2004
3. P. Muraca, P. Pugliese, G.Rocca, "An Extended Kalman Filter for the state estimation of a mobile robot from intermittent measurements", *16th Mediterranean Conference on Control and Automation* June 2008, pp. 1850-1855
4. T. Nelson, E. A. Wan, "A Two-Observation Kalman Framework for Maximum-Likelihood Modeling of Noisy Time Series", *The 1998 IEEE International Joint Conference on Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence*, pp.2489–2494, May, 1998.
5. L. D'Alfonso, W. Lucia, P. Muraca, P. Pugliese, "Filters for Mobile Robots: EKF, UKF and Sensor Switching - Experimental Results", *9th IEEE International Conference on Control and Automation, IEEE ICCA'11*, Santiago, Chile, 2011
6. Y.Misono, Y.Goto, Y.Tarutoko, K.Kobayashi, K.Watanabe, "Development of Laser Rangefinder-based SLAM Algorithm for Mobile Robot Navigation", *SICE Annual Conference 2007*, Sept.17-20, Japan 2007
7. S.Fu;H. Liu;L.Gao;Y.Gai, "SLAM for mobile robots using laser range finder and monocular vision", *14th International Conference on Mechatronics and Machine Vision in Practice M2VIP 2007*, pp.91,96, 4-6 Dec. 2007
8. L.Maohai, H.Bingrong, L.Ronghua, "Novel Mobile Robot Simultaneous Loclization and Mapping Using Rao-Blackwellised Particle Filter", *International Journal of Advanced Robotic Systems*, Vol. 3, No. 3, pp. 231-238, 2006
9. [J. Borenstein, H. Everett, L. Feng, D. Wehe, "Mobile Robot Positioning - Sensors and Techniques", *Journal of Robotic Systems*, Vol. 14, No. 4, pp. 231-249, 1997
10. M. Betke, L. Gurvits, "Mobile Robot Localization Using Landmarks", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, pp.251-263, 1997
11. J. Esteves, A. Carvalho, C. Couto, "Generalized Geometric Triangulation Algorithm for Mobile Robot Absolute Self-Localization", *International Symposium on Industrial Electronics (ISIE)*, 2003

12. E.Z. Casanova, S.D. Quijada, J.G. Garcia-Bermejo, J.R. Gonzàlez, "A New Beacon-based System for the Localization of Moving Objects", *IEEE International Conference on Mechatronics and Machine Vision in Practice*, 2002
13. G.Cotugno, L. D'Alfonso, P. Muraca, P. Pugliese, "A new Extended Kalman Filter based on actual local information for mobile robots", *9th European Workshop on Advanced Control and Diagnosis, ACD 2011*, Budapest, Hungary, November 17-18, 2011
14. Hyeonwoo Cho, Sang-Woo Kim, "Mobile Robot Localization Using Biased Chirp-Spread-Spectrum Ranging", *IEEE Transactions on Industrial Electronics*, vol.57, no.8, pp.2826,2835, Aug. 2010
15. Soonshin Han, HyungSoo Lim, JangMyung Lee, "An Efficient Localization Scheme for a Differential-Driving Mobile Robot Based on RFID System", *IEEE Transactions on Industrial Electronics*, vol.54, no.6, pp.3362,3369, Dec. 2007
16. Hung-Hsing Lin, Ching-Chih Tsai, Jui-Cheng Hsu, "Ultrasonic Localization and Pose Tracking of an Autonomous Mobile Robot via Fuzzy Adaptive Extended Information Filtering", *IEEE Transactions on Instrumentation and Measurement*, vol.57, no.9, pp.2024,2034, Sept. 2008
17. M. S. Grewal, A. P. Andrews, *Kalman Filtering: Theory And Practice Using MATLAB*, 3rd ed., Wiley, 2008
18. S. J. Julier, J. K. Uhlmann, "Unscented filtering and nonlinear estimation", *Proc. of the IEEE*, 92(3):401–422, 2004
19. E. Wan, R. van der Merwe, "The unscented Kalman filter", in *Kalman Filtering and Neural Networks*, S. Haykin, Ed., Wiley, 2001
20. Y. Wu, D. Hu, M. Wu, X. Hu, "Unscented Kalman filtering for additive noise case: augmented versus nonaugmented", *IEEE Signal Processing Letters*, 12(5):357–360, 2005
21. Yi Cao, "Learning the Unscented Kalman Filter" [Online], available: http://www.mathworks.com/matlabcentral/fileexchange/18217
22. The Mathworks, Inc., *MATLAB User's Guide*, Natick, MA, 1996
23. B.D.O. Anderson, J.B. Moore, *Optimal Filtering*, Prentice-Hall, 1979
24. F. Pengpai, S. Li-fen, W. Bing, and W. Wei, "Particle Filter-Weight Estimation and Dual Particle Filter", *International Workshop on Intelligent Systems and Applications*, pp.1–4, 2009.
25. V. Crugliano, L. D'Alfonso, P. Muraca, P. Pugliese, "Experimental results of a sensor switching policy for mobile robots", *18th Mediterranean Conference on Control and Automation (MED) 2010*, pp.581,585, 23-25 June 2010
26. L. D'Alfonso, A. Grano, P. Muraca, P. Pugliese, "Mobile robot localization in an unknown environment using sonar sensors and an incidence angle based sensors switching policy - Experimental results", *10th IEEE International Conference on Control and Automation*, Hangzhou, China, June 12-14, 2013
27. K-TEAM Corporation [Online]. Available: http://www.k-team.com
28. C. Hsu, H. Chen, C. Lai, "An Improved Ultrasonic-Based Localization Using Reflection Method", *2009 International Asia Conference on Informatics in Control, Automation and Robotics*
29. V. Gupta, T. Chung, B. Hassibi, R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and dynamic sensor coverage", *Automatica*, vol. 2, pp. 251–260, 2006.
30. L. Carotenuto, P. Muraca, G. Raiconi, "On the optimal design of the output transformation for discrete-time linear systems", *J. Optim. Theory Appl.*, 68(1):1–18, 1991

31. J. D. Tard´os, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data", *IJRR*, vol. 21, no. 4, pp. 311-330, Apr. 2002
32. A.K.Pandey, K.M. Krishna, H.Hexmoor, "Feature Chain Based Occupancy Grid SLAM for Robots Equipped with Sonar Sensors", *International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS) 2007*, April 30 2007-May 3 2007, pp. 283-288, Waltham Massachusetts
33. J.Lee, C.Kim, W.K.Chung, "Robust RBPF-SLAM using Sonar Sensors in Non-Static Environments", *2010 IEEE International Conference on Robotics and Automation*, May 3-8, 2010, Anchorage, Alaska, USA
34. R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures", *Comm. ACM*, vol. 15, no. 1, pp. 11-15, Jan. 1972
35. A.Mallios, P.Ridao, D.Ribas, E.Hern, "Probabilistic Sonar Scan Matching SLAM for Underwater Environment", *IEEE OCEANS 2010*, Sydney, 2010
36. J.Li, M.Lee, S.Park, J.Kim, "Range sonar array based SLAM for P-SURO AUV in a partially known environment", *9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 353-354, November 26-29, Korea, 2012
37. T.N.Yap Jr, C.R.Shelton, "SLAM in Large Indoor Environments with Low-Cost, Noisy, and Sparse Sonars", *2009 IEEE International Conference on Robotics and Automation*, May 12-17, 2009, Japan
38. R. C. Smith, P. Cheeseman, "On the representation and estimation of spatial uncertainty", *IJRR*, vol. 5, no. 4, pp. 56-68, 1986
39. L.Pedraza, D. Rodriguez-Losada, F. Matia; G. Dissanayake; J.V. Miro, "Extending the Limits of Feature-Based SLAM With B-Splines", *IEEE Transactions on Robotics*, vol.25, no.2, pp.353,366, April 2009
40. G.C. Anousaki; K.J. Kyriakopoulos, "Simultaneous localization and map building of skid-steered robots", *IEEE Robotics & Automation Magazine*, vol.14, no.1, pp.79,89, March 2007
41. J. Civera; D. Galvez-Lopez; L. Riazuelo; J.D. Tardos; J. M M Montiel, "Towards semantic SLAM using a monocular camera", *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.1277,1284, 25-30 Sept. 2011
42. Kim Jungho; Yoon Kuk-Jin; Kim Jun-Sik; Kweon Inso, "Visual SLAM by Single-Camera Catadioptric Stereo", *International Joint Conference SICE-ICASE 2006*, pp.2005,2009, 18-21 Oct. 2006
43. L. D'Alfonso, A. Griffo, P. Muraca, P. Pugliese, "A SLAM algorithm for indoor mobile robot localization using an Extended Kalman Filter and a segment based environment mapping", *16th International Conference on Advanced Robotics, ICAR 2013*, Montevideo, Uruguay, November 25-29, 2013
44. L. D'Alfonso, A. Grano, P. Muraca, P. Pugliese, "A polynomial based SLAM algorithm for Mobile Robots using ultrasonic sensors - Experimental Results", *16th International Conference on Advanced Robotics, ICAR 2013*, Montevideo, Uruguay, November 25-29, 2013
45. H. Durrant-Whyte, "An Autonomous Guided Vehicle for Cargo Handling Applications", *Int. Journal of Robotics Research*, 15(5), pp. 407-441, Oct. 1996
46. W. Burgard, A. B. Cremers, D. Fox, D. Ahnel, G. Lakemeyer, D. Schulz, W. Steiner and S. Thrun, "Experiences with an interactive museum tour-guide robot", *Artificial Intelligence*, 114(1-2), pp.3-55, Jan. 1999.

47. A. Elfes, “Using occupancy grids for mobile robot perception and navigation”, *Computer*, 22(6), pp. 46- 57, Jun. 1989.
48. M.A. Fischler, R.C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”, *Commun. ACM*, vol. 24, pp. 381-395, 1981.
49. X. Gao, X. Hou, J. Tang, H. Cheng, “Complete solution classification for the perspective-three-point problem”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 930-943, 2003.
50. R.M. Haralick, D. Lee, K. Ottenburg, M. Nölle, “Review and analysis of solutions of the three point perspective pose estimation problem”, *Int. J. Comput. Vision*, vol. 13, pp. 331-356, 1994.
51. P. Rives, P. Bouthèmy, B. Prasada, E. Dubois, “Recovering the orientation and the position of a rigid body in space from a single view”, Tecnical report, INRS-Tèlècommunications, 3, Place du Commerce, Ile-des-Soeurs, Verdun, Quebec, Canada, 1981.
52. L. Quan, Z. Lan, “Linear N-point camera pose determination”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 774-780, 1999.
53. M.A. Abidi, T. Chandra, “A new efficient and direct solution for pose estimation using quadrangular targets; algorithm and evaluation”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, pp. 534-538, 1995.
54. L. Merckel, T. Nishida, “Evaluation of a method to solve the perspective-two-point problem using a three-axis orientation sensor”, *Proc. of the IEEE CIT 2008*, Sydney, Australia, 2008, pp. 862-867.
55. F.M. Mirzaei, S.I. Roumeliotis, “A Kalman filter-based algorithm for IMU-camera calibration: observability analysis and performance evaluation”, *IEEE Trans. Robot.*, vol. 24, pp. 1143-1156, 2008.
56. T. Oskiper, S. Samarasekera, R. Kumar, “Tightly-coupled robust vision aided inertial navigation algorithm for augmented reality using monocular camera and IMU”, *Proc. of the IEEE ISMAR 2011*, Basel, Switzerland, 2011, pp. 255-256.
57. C. Su, Y. Xu, H. Li, and S. Liu, “Application of Wu’s method in computer animation”, *Proc. of the fifth Int. Conf. on CAD/CG*, HongKong, China, 1997, pp. 211-215.
58. R. Horaud, B. Conio, O. Leboulleux, “An analytic solution for the perspective 4-Point problem”, *Comput. Vision Graph.*, vol. 47, pp. 33-44, 1989.
59. J.S.C. Yuan, “A general photogrammetric method for determining object position and orientation”, *IEEE Trans. Robot. Autom.*, vol. 5, pp. 129-142, 1989.
60. G. Chen, D. Xu, P. Yang, “High precision pose measurement for humanoid robot based on PnP and OI algorithms”, *Proc. of the IEEE ROBIO 2010*, Tianjin, China, 2010, pp. 620-624.
61. Q. Wang, X. Zhang, D. Xu, “Human behavior imitation for a robot to play table tennis”, *Proc. of the CCDC 2012*, Taiyuan, China, 2012, pp. 1482-1487.
62. Q. Wang, X. Zhang, D. Xu, “On pose recovery for generalized visual sensors”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 848-861, 2004.
63. Z. Kukelova, M. Bujnak, T. Pajdla, “Closed-form solutions to the minimal absolute pose problems with known vertical direction”, *Proc. of the ACCV 2010*, Queenstown, NZ, 2010, pp. 216-229.
64. http://www.logitech.com
65. http://www.vision.caltech.edu/bouguetj/calib_doc

66. http://code.google.com/p/ardu-imu/wiki/HomePage
67. L. D'Alfonso, E. Garone, P. Muraca, P. Pugliese, "P3P and P2P Problems with known camera and object vertical directions", *Proc. of the 21st IEEE Mediterranean Conference on Control and Automation (MED '13)*, Crete, Greece, 2013, pp. 444-451.
68. L. D'Alfonso, E. Garone, P. Muraca, P. Pugliese, "On the use of the inclinomenters in the PnP Problem", *Proc. of the IEEE 12th biannual European Control Conference (ECC '13)*, Zurich, Switzerland, 2013, pp. 4112-4117.
69. C.P. Lu, G.D. Hager, E. Mjolsness, "Fast and globally convergent pose estimation from video images", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 610-622, 2000.
70. V. Lepetit, F. Moreno-Noguer, P. Fua, "EPnP: an accurate $O(n)$ solution to the PnP problem", *Int. J. Comput. Vis.*, vol. 81, pp. 155-166, 2009.
71. http://petercorke.com/Machine_Vision_Toolbox.html
72. P. Corke, *Robotics, Vision and Control. Fundamental Algorithms in Matlab*, Springer, 2011.
73. J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions", *SIAM J. Optim.*, vol. 9, pp. 112-147, 1998.
74. L. D'Alfonso, A. Grano, P. Muraca, P. Pugliese, "Sensor fusion and surrounding environment mapping for a mobile robot using a Mixed Extended Kalman Filter", *Proc. of the 10th IEEE International Conference on Control and Automation, Hangzhou, China, June 12-14, 2013*
75. "Sensor fusing using a convex combination of two Kalman filters - Experimental results", *16th International Conference on Advanced Robotics, ICAR 2013*, Montevideo, Uruguay, November 25-29, 2013
76. G. Cotugno, L. D'Alfonso, W. Lucia, P. Muraca, P. Pugliese, "Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction", *Proc. of the 21st IEEE Mediterranean Conference on Control and Automation (MED '13)*, Crete, Greece, 2013
77. L. D'Alfonso, E. Garone, P. Muraca, P. Pugliese, "On the use of Inertial Measurements Units in the PnP Problem", *submitted to IEEE tran. Robotics*
78. L. D'Alfonso, A. Grano, P. Muraca, P. Pugliese, "An efficient polynomial based SLAM algorithm for mobile robots in an unknown indoor environment", *submitted to the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, Hong Kong, China, 2014.
79. L. D'Alfonso, E. Garone, P. Muraca, P. Pugliese, "On the use of IMUs in the PnP Problem", *submitted to the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, Hong Kong, China, 2014.
80. K. Ogata, *Discrete-Time Control Systems*, Prentice-Hall, 1987
81. Gene F. Franklin, Michael L. Workman, and Dave Powell, *Digital Control of Dynamic Systems*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997
82. Jazwinsky, *Stochastic Processes and Filtering Theory*, Mathematics in science and engineering, volume 64

Acknowledgements

Ed ecco arrivato il momento dei ringraziamenti. È notte fonda mentre scrivo e chiedo venia in anticipo per eventuali errori di battitura. Non sono abituato a scrivere pagine di ringraziamenti. Ai tempi delle mie prime due lauree infatti, i ringraziamenti non andavano molto di moda. Cercherò tuttavia di adattarmi a questa usanza e, auspicabilmente, proverò a non essere troppo prolisso (cosa che in genere non mi riesce tanto facilmente).

Il mio primo ringraziamento va al mio papà. Grazie papà per i tuoi abbracci quando mi vieni ad aspettare di fronte alla porta dell'ascensore ogni venerdì sera; grazie per le tue raccomandazioni (anche se non sembra, le ascolto sempre), grazie per il tuo sguardo sempre rivolto su di me. Grazie per avermi insegnato che nella vita la cosa più importante è essere buoni con chi si ama, e tu lo sei sempre con me. Spero un giorno di poter diventare come te.

Grazie anche a te, mamma. Se riesco a fare nottate lavorando, se non demordo mai, affrontando tutto a testa alta, è merito tuo e dei tuoi insegnamenti. E di questo ti ringrazio. Grazie perché quando mi vedi stanco per il troppo lavoro, nonostante tu voglia dirmi di riposare, ti trattieni perché mi conosci, sai come sono fatto e sai che non riuscirei a restare in pace con me stesso se non mi impegnassi fino alla fine, provando sempre a dare il massimo. Grazie mamma, ovunque io sia, c'è sempre un filo che mi lega a te.

Grazie al mio amico Dik, grazie per le tue feste quando mi vedi arrivare, per la tua dolcezza, per il tuo abbaiare piano piano ai piedi della mia poltrona quando vuoi un po' di coccole, grazie per la tua compagnia mentre sono al computer. Grazie amico mio.

Grazie a zia Rosanna e a zia Adele che vedo sempre come delle seconde mamme. Grazie a zio Salvatore: mi accompagni sempre in ogni mio viaggio e so di poter contare sempre su di te, grazie zio. Grazie a zia Lina, so che chiedi sempre di me e ti sento sempre vicina.

Grazie a Gaetano, Maria, Andrea, Toniuccia e Fefo: siete come fratelli e sorelle per me.

Grazie a Giada, Francesca, Alessandro e Rodolfo (in ordine di età, poiché altro ordinamento non potrebbe esistere), voi siete piccoli e non lo sapete, ma i vostri sorrisi, le vostre smorfie, ritratte in tutte le foto che vi ho scattato e che vi continuo a scattare, mi aiutano a ridere e mi danno forza in ogni momento. Grazie piccolini miei.

Grazie ai fratelli che ho scelto (e se li ho scelti, un motivo ci sarà), agli amici di una vita, grazie a Giova, Tato, Victor e al mio avvocato Andrea (Fefo, ci saresti anche tu, ma ti ho inglobato nei cugini, solito conflitto di interessi).

Amici miei grazie per le risate, le chiacchiere, i discorsi seri, le telefonate, le avventure vissute, i consigli, grazie grazie di tutto. Grazie anche a Deni, che mi sopporta nel mio caos ed evita che io e Tato andiamo divergendo con i nostri voli pindarici.

Grazie agli amici dell'università: Walter, Antonio, Stefano M., Stefano (piccolo), Giuseppe, Spillo, Felice, Matteo, Francesco, Simona. Le pause passate a scherzare con tutti voi (in Italia e in Belgio) mi hanno sempre aiutato a ripartire con più carica.

Grazie al Prof. Emanuele Garone, per l'amicizia dimostratami e per tutto quel che ha fatto per me mentre ero a Bruxelles.

Grazie a tutti i Professori del terzo e quarto piano del cubo 42C, per avermi accolto a braccia aperte fin dal primo giorno in cui ho iniziato il dottorato. Grazie al Prof. Famularo per le belle chiacchierate a base di manga e anime. Dulcis in fundo, grazie al Prof. Muraca e al Prof. Pugliese per i consigli sempre utili e per l'aiuto datomi in qualunque momento.

Grazie al Signor Aldo e alla Signora Gina, per avermi accolto nella loro splendida famiglia e per avermi fatto sentire da subito parte integrante di essa.

E infine, grazie a te, amore mio, che starai leggendo pronta a piangere come tuo solito (cerca di evitare dai). Come ho sempre detto, tu sei il vero delta di differenza nella mia vita. I tuoi sorrisi illuminano le mie giornate, il tuo sguardo mi riempie il cuore di calore, persino le tue sgridate sono oramai parte integrante di me. Tutto ciò mi accompagna in ogni mio giorno di lavoro. Tu sei quanto di più importante io abbia. Da quando sei entrata nella mia vita, tutto per me è cambiato in meglio. Da te apprendo ogni giorno nuovi insegnamenti e per te io cerco, ogni giorno, di diventare un uomo migliore rispetto a quello che ero il giorno prima. Spero così, un giorno, di poter essere alla tua altezza. Spero di poter pensare, un giorno, di meritarti. Per ora, posso solo ringraziare il destino per aver fatto sì che tu entrassi nella mia vita. Ancora mi emoziono quando penso alle prime nostre chiacchierate e ai primi nostri scambi di sguardi. Grazie amore mio, grazie perché mi sei sempre vicina in ogni mio gesto e in ogni mio respiro. Grazie per avermi accompagnato durante tutto il mio dottorato. Grazie per avermi sopportato, so che non è stato facile, e supportato ogni giorno. Pensare al futuro è più semplice da quando so che tu sarai al mio fianco, a stringere la mia mano. Ogni frase mi pare superflua, potrei continuare a scrivere pagine e pagine ringraziandoti ma ciò non basterebbe comunque a rendere l'idea di quanto questo mio GRAZIE venga direttamente dalla parte più profonda del mio cuore, dove tu oramai vivi da un bel pò. Grazie mia bella, grazie di esistere e di essere al mio fianco. Ogni mio traguardo è nulla in confronto alla gioia che provo quando tu mi sorridi. Ti amo, sempre e per sempre, sempre e per sempre.

A tutti voi, GRAZIE. Con il vostro aiuto punterò a raggiungere sempre nuovi traguardi.

Rende, 29/11/2013