UNIVERSITA' DELLA CALABRIA
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Scuola di dottorato **"Bernardino Telesio"**

Indirizzo: **Progetto RISPEISE - Domotica**
Con il contributo di

Fondo Sociale Europeo

HEALTH MONITORING OF BUILDINGS: METHODOLOGIES, TOOLS AND PRACTICAL APPLICATIONS

Settore scientifico disciplinare **ING-INF/04**

Direttore: Ch.mo Prof. Roberto Bartolino [illegible]

Supervisori

Ch.mo Prof. Raffaele Zinno [illegible]

Ch.mo Prof. Pietro Muraca [illegible]

Ch.mo Prof. Enrico Natalizio - UTC Compiègne, Laboratoire Heudiasyc (FR) [illegible]

Dottorando Antonio Grano
Antonio Grano

Abstract

Ensuring a high level security to buildings is an important task which involves all the building life, from the design to the daily service. The building security systems may answer to two different kinds of need: "prevention" and "care". Prevention is related to all studies and activities aimed to improve building security under construction; we could mention structural engineering, which is focused on developing mathematical models for characterizing structural behaviour. On the other side, care includes all the activities performed on the structure under regular service, e.g. design of dynamic dampers and all what is used as support to the main structure.

Computer science and electronic engineering can be useful in both cases. Computer and other sensors systems can be used as useful supports to acquire and analyze data for preliminary diagnosis and in-situ investigations. Moreover, the high level achieved by technology nowadays allows to develop also low-cost systems, while keeping a good accuracy, thus letting the user a larger variety of tools to choose, depending on the needs.

The present work is aimed at developing low-cost systems and methodologies for improving building security. The focus will be on both prevention and care.

Some systems will be proposed to be used for preliminary analysis, in particular for displacements monitoring during tests on shaking-table, an important class of experiments performed to study the behaviour of a structure under the action of a seismic force.

Other applications will focus more on monitoring in-situ; in particular, mobile robots will be used to develop solutions to a common problem in context of building security, which is environment exploration expecially in hostile situations. In robotics, such a problem is denoted with "SLAM" (Simultaneous Localization and Mapping) and consists in localizing a mobile robot while reconstructing a map of the surrounding environment, where the robot moves.

Detailed reports will be provided, showing the performed tests on each proposed solution. As it will be seen, the obtained results show the validity of the implemented techniques and open interesting scenarios for future research.

Sommario

Garantire un elevato livello di sicurezza per gli edifici è un compito importante che coinvolge tutta la vita dell'edificio, dalla progettazione al servizio quotidiano. I sistemi di sicurezza dell'edificio possono rispondere a due diversi tipi di necessità: "prevenzione" e "cura". La prevenzione è legata a tutti gli studi e le attività volte a migliorare la sicurezza degli edifici in costruzione; potremmo citare l'ingegneria strutturale, che si concentra sullo sviluppo di modelli matematici per la caratterizzazione di comportamento strutturale. Dall'altra parte, la cura comprende tutte le attività svolte sulla struttura in regolare servizio, ad esempio, progettazione di smorzatori dinamici e tutto ciò che viene utilizzato come supporto alla struttura principale.

L'informatica e l'ingegneria elettronica possono essere utili in entrambi i casi. I computer e altri sistemi di sensori possono essere utilizzati come supporti utili per acquisire e analizzare dati per diagnosi preliminari e indagini in-situ. Inoltre, l'elevato livello raggiunto dalla tecnologia al giorno d'oggi consente di sviluppare anche sistemi a basso costo, pur mantenendo una buona precisione, permettendo così all'utente una grande varietà di strumenti da scegliere, a seconda delle esigenze.

Il presente lavoro si propone di sviluppare sistemi a basso costo e metodologie per migliorare la sicurezza degli edifici. L'attenzione sarà focalizzata sia sulla prevenzione sia sulla cura.

Verranno proposti alcuni sistemi da utilizzare per analisi preliminari, in particolare per il monitoraggio degli spostamenti durante le prove su tavola vibrante, un'importante classe di esperimenti eseguiti per studiare il comportamento di una struttura sotto l'azione di una forza sismica.

Altre applicazioni si concentreranno di più sul monitoraggio in situ; in particolare, saranno utilizzati robot mobili per sviluppare soluzioni ad un problema comune nel contesto della sicurezza degli edifici, che è l'esplorazione dell'ambiente soprattutto in situazioni ostili. In robotica, tale problema viene indicato con SLAM (Simultaneous Localization and Mapping) e consiste nella localizzazione di un robot mobile in contemporanea con la ricostruzione di una mappa dell'ambiente circostante, in cui il robot si muove.

Verranno forniti dei report dettagliati, mostrando i test effettuati su ogni soluzione proposta.

Come si vedrà, i risultati ottenuti mostrano la validità delle tecniche messe in atto e lasciano aperti scenari interessanti per la ricerca futura.

Contents

List of Figures

List of Tables

Introduction

The first example of building automation dates back to 1891, when a business man from Wisconsin, William Penn Powers, founded the "Power regulator company", which made temperature regulators. In 1907, for the first time, an automatic ai conditioning system was installed in a hotel in Chicago and since then, there have been more and more buildings with the need for energy control systems. In 1930, at the "4th exposure of decorative and industrial arts" in Monza (Italy) SCAEM presented the "Electric house" [1], designed by 4 italian architects Figini, Pollini, Fratta and Libera; it was a prototype of house for holydays, provided with many electrical devices, but it was destined to become more and more popular (see Figure I.1). Another prototype was

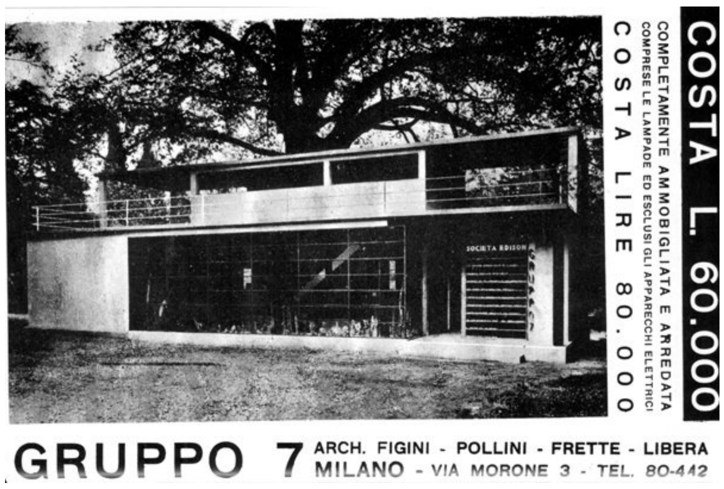

COSTA L. 60.000
COMPLETAMENTE AMMOBIGLIATA E ARREDATA
COMPRESE LE LAMPADE ED ESCLUSI GLI APPARECCHI ELETTRICI
COSTA LIRE 80.000
SOCIETÀ EDISON
GRUPPO 7
ARCH. FIGINI - POLLINI - FRETTE - LIBERA
MILANO - VIA MORONE 3 - TEL. 80-442

Fig. I.1. The "Electric house"

presented by R.B.Fuller during the 30s; it was totally built with prefabricated materials and it was rested on a rotating basis.

In U.S., the "System 320", produced during the 50s by a group of electrical engineers was the first control systems for energy management was produced, which used liquid crystal displays to show; during the same period, more precisely in 1956, Peter and Alison Smithson presented "The future house", totally based on nuclear energy, at an exhibition in London; later, in 1966, Jim Sutherland, an engineer in collaboration with the Westinghouse Corporation, produced the "ECHO IV" (Electronic Computing House Operator), which implemented control function for managing energy and some daily activities like shopping list. In 1970 the "Pico Electronics" was born, an industry which has produced X10, one of the most used communication standards in domotics. Another important project is a propotype designed by Alex Pixe in 1972, a self-sufficient house powered by renewable energy; it was destined to emphasize the need for an autonomous control system of energy.

However, the development of autonomous houses had great growth with the advent of computer technology. Among the 70's and 80's, the "Ahwatukee house" and the "DORIS" project were the first cases where the control of security, environmental comfort and alarm systems was based on microchips. In Italy, in 1983, Ugo La Pietra presented the "Telematic House", where comfort and lighting were managed by computer based systems and advanced telecommunication tools. In those years, however, the most advenced domotic technologies were presented in Japan by the Ken Sakamura's "THRON House"; they already provided most of the currently used technologies.

Home automation had a rapid growth from the end of 80s; on January 14 and 15, 1988, in Paris, the "Premire confrence europenne sur l'habitat intelligent", while in 1989, the "Maison Domotique Panorama" was presented in Lione; it has been the inspiration for many exhibition models. In those years they started to look more at real systems instead of futurible prototypes; thus, many systems were proposed: "Securisan" system in France was used in a lot of civil houses, while the "ESPIRIT (European Strategic Program for Research and Development)" was one of the first big projects for home automation. Some other examples were "Butibus" system in France, "Eibus" system in Germany. In Italy, the first energy management systems ("Ariston" by Merloni Elettrodomestici and "Isi" by Ave) appeared in 1985: they were based on a telephone able to manage also electrical devices. After that, different projects were proposed: "Intelligent" by Beghelli was a radio waves based system, easy to install and to use, Olivetti proposed solutions based on wireless communication while Innovatech started to use touch screen displays an BTicino produced "MyHome", an advanced home automation system for centralized control. Moreover, in the 90s, the CENELEC set up the CT 105, a committee dedicated to fix normatives for home automation systems development.

It is possible to distinguish 4 different fields in home automation:

1. Environment management system: this area includes illumination, air conditioning and all what is related to environment comfort;
2. Building security: alarm systems, support systems for people with disabilities, fire protection, flood protection, protection from gas leaks etc.
3. Communication systems;
4. Electrical devices management systems.

Home automation, with particular reference to the second of the above mentioned fields, is also one of the strategic areas of the RISPEISE (International Network for Exchanging Good Practices in Building Safe and Seismically Sustainable) project, within which this work is framed. This project (see Figure I.2) has been developed at the SmartLab (Structural Monitoring Advanced materials Rehabilitation Testing Laboratory) in the University of Calabria; it has the strategic aim of creating a network of people with different expertises in the field of building safe and seismically sustainable(see Figure I.3). Within this framework, the aim of the present work is to develop some strategies for health monitoring of building, in order to use modern technology to help people in safely enjoying daily environments. The idea is to use some low cost technologies in order to find easily usable solution to common health monitoring problems.

The new contribution of this thesis lies in two different areas related to health monitoring of buildings:

1. Systems for structural tests before putting a building in normal service (prevention);
2. Systems for monitoring buildings already in service, in order to improve people safety and building security (care).

For what concerns the first point, the idea is to develop some sensor systems alternative to classical sensing techniques, to be used to perform preliminary laboratory experiments, useful to test materials and building metodologies; this activity can help engineers and construction companies to improve the product security. However, structural health monitoring has not to be limited to the "prevention" aspect, but it is very important also for in-service structures; there are some natural disasters (e.g. Chernobyl-1986 and Fukushima-2011, see Figures I.4,I.5) that clearly emphasize the need for systems able to help people in accomplishing dangerosu tasks. In this context, an effective choice can be the use of mobile robots. Indeed, the design of a static and wired system can be useful but it usually requires space and it can be not so easy to change its features to meet eventual new requirements. On the contrary, mobile robots usually occupy a relatively little space, as well as they can be equipped with a of different sensors and tools and can be moved easily to be adapted to different scenarios and execute various tasks; they can explore an environment totally autonomously, protecting in this way people from exploring dangerous places.

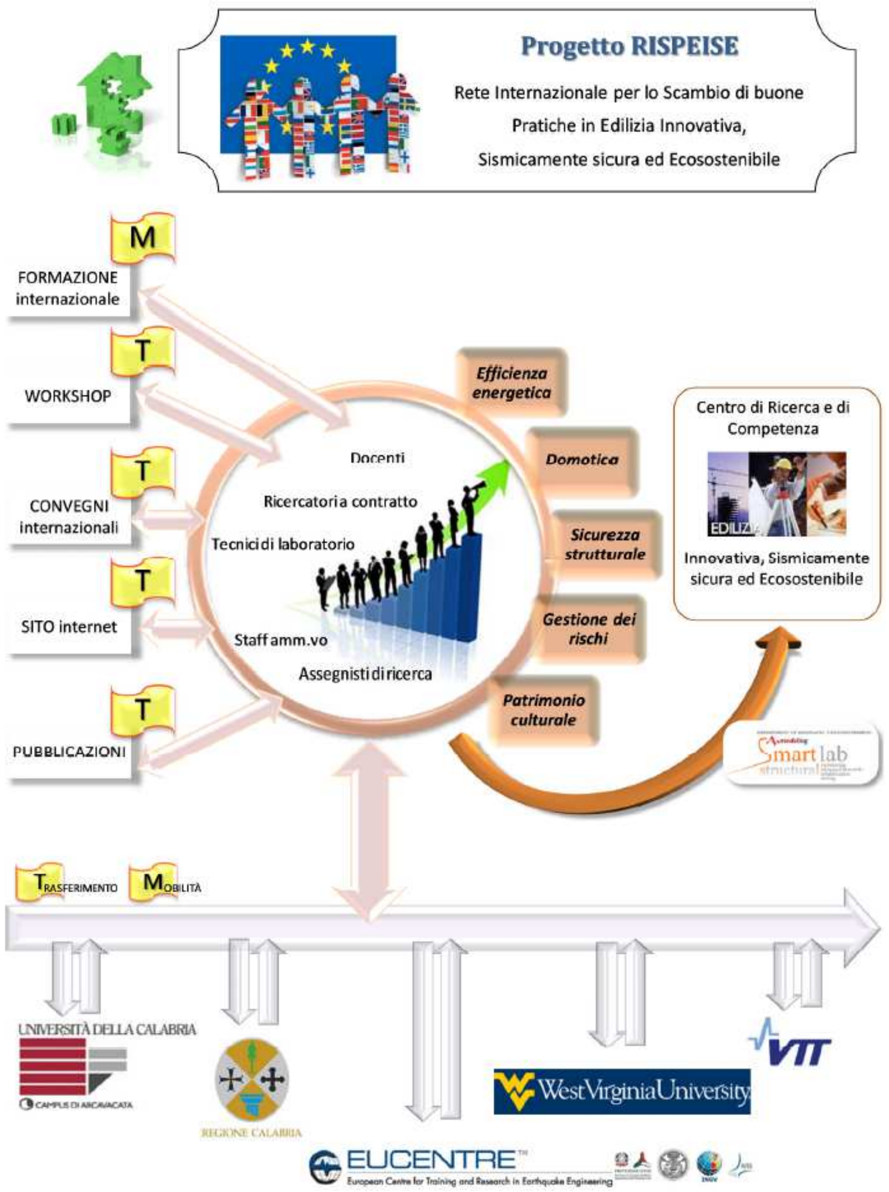


Fig. I.2. The "RISPEISE project"

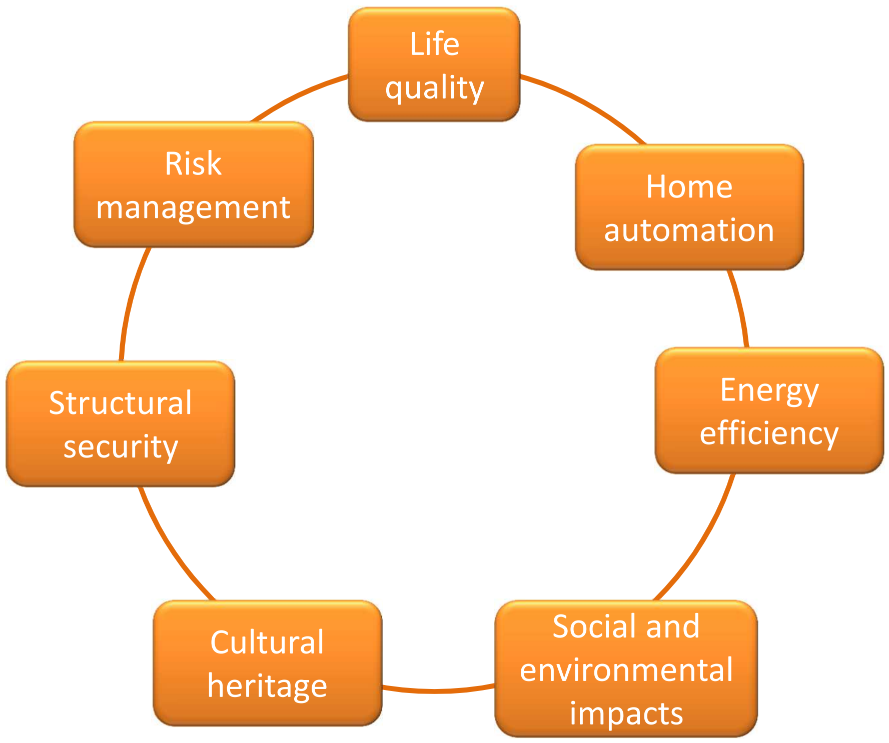


Fig. I.3. The "Scientific fields in the RISPEISE project"

The thesis has the following structure:

- **Chapter 1** is dedicated to robotics and SLAM problem: an introduction to the problem is given, then some solutions are proposed to solve different problems in a SLAM context; mathematical models, methodologies and performed experiments are described in detail, to conclude with final remarks and discussion about possible improvements to the implemented techniques;
- **Chapter 2** presents an optical displacements monitoring systems, alternative to the classical sensors; the reasons for such a design are related to security and will be explained in detail in the chapter;
- **Chapter 3** presents a new compression data technique, starting from data acquired by the same kind of sensors used in **Chapter 2**, in order to efficiently use the available bandwidth for transmitting large amounts of acquired data.

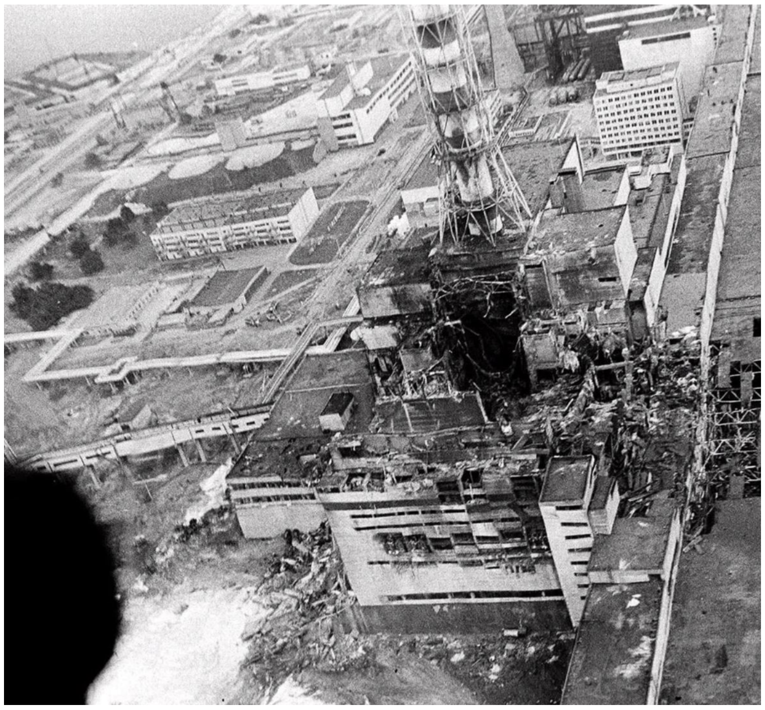


Fig. I.4. The Chernobyl disaster - 1986

Fig. I.5. The Fukushima accident - 2011

Part I

Indoor environments exploration

1

SLAM (Simultaneous Localization and Mapping) algorithms for mobile robots in indoor environments

1.1 A little history of mobile robots

The general task of a mobile robot is to follow a predefined path in an environment, in order to reach a destination point and accomplish an assigned task. Each robot is usually equipped with various kinds of sensors for explorind the environment and detecting eventual obstacles.

The first mobile robots were designed for military purposes during the World War II and consisted in flying bombs and rockets equipped with very simple automatic control systems.

After the War, robots started to be used also for other tasks; in 1948, in order to study some animals habits, W.Grey Walter deisgned a robot able to navigate in an environment using a light sensor to detects the objects. The dependency of such a robot from the environment illumination had been eliminated at the John Hopkins University, which proposed "Beast", a robot able to explore the environment using a ultrasonic sensor; moreover, "Beast" was able also to find a power source to recharge its battery, if needed. In 1969 the first example of mobile robot for home automation applications was presented (see Figure 1.1: it was able to automatically mow the lawn, while in 1970 Lester Donald Ernest, an american computer scientist, tried to modify an existing prototype in order to create a robot based on visual perception. In the following years, the Stanfor research Institute presented "Shakey", a robot with a more complex equipment; it had a camera, a rangefinder, bump sensors and a radio link; moreover, it was the first robot able to autonomously analize general commands and divide them into basic actions, through the use of different kinds of sensors. During the 80s the first robotic vehicles were developed at the Bundeswehr University of Munich and at the Hughes Research Laboratory. During th 90s, many different solutions were proposed. It can be mentioned:

- commercially hospital robots;
- robots for research purposes ([2]);

mowbot
the world's first truly automatic lawnmower
June 1969 107

Until you see MOWBOT, you'll never believe it. Turn it on. Electronically-controlled MOWBOT mows and steers itself around your lawn until the grass is completely cut. Leaves flowers, trees, and shrubs untouched.

Safe? Sure. Double rotary blades won't throw grass, stones, or foreign objects. And the MOWBOT mower stops automatically at the touch of any obstruction.

Rechargeable electric power makes MOWBOT a good neighbor. No noise. No fumes. So, if you have nothing to do while MOWBOT mows your lawn, relax; you'll think of something!

If your favorite lawn and garden center or hardware store doesn't yet carry MOWBOT, please write for the name of your nearest distributor. **MOWBOT INC., 9 Hackett Drive, Tonawanda, N.Y. 14150**

Fig. 1.1. The Mowbot

- robots for exploring live volcanoes (Dante I [3] and Dante II [4]);
- autonomous robots for traffic; with guests on board, the twin robot vehicles VaMP and VITA-2 of Daimler-Benz and Ernst Dickmanns of UniBwM drive autonomously more than one thousand kilometers on a Paris three-lane highway in standard heavy traffic at speeds up to 130 km/h(see Figure [5]). Similarly, in 1995, robot cars were improved by Erns Dickmanns, using active vision for a better environment perception, especially in case of scene changing.

In 2001, starting from studying the insects behaviour, swarm robotics start to be widespread; group of robots start to be used in order to execute more complex operations.

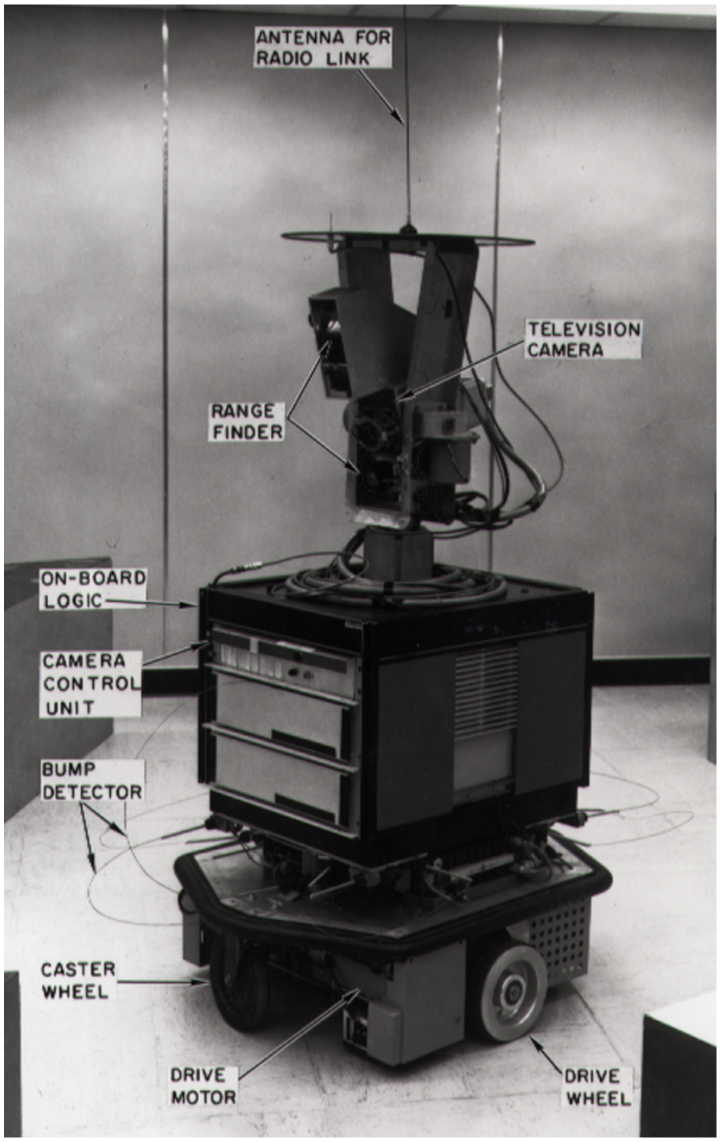


Fig. 1.2. Shakey the robot

However, robot motion does not involve only a pure path planning problem; indeed, the predefined path may be not accurately followed due to inaccuracy of the adopted control law, which depend on different factors:

1. bad calibration of the low-level control system; to control effectively the input signals, a PID algorithm or an optimal control algorithm can be used, but their calibration has to carefully take into accont the model disturbances;
2. inaccurate command signals; at a higher level, the system control consists in computing, at each step, the input commands for robot's actuators. In this case, the control reliability depends on the knowledge of robot position and orientation and of the environment; if the environment and the obstacle placements are perfectly known, it is quite easy to design a high level control for the robot; on the contrary, inaccurate hypothesis on the environment structure and the obstacles placement can be a significant errors source for robot motion, as well as they may complicate the control design.

Thus, it is quite clear that path planning cannot avoid dealing with localization and environment mapping, in order to optimize the robot path and reduce energy consumption and walking time (see Figure 1.3). In the present work, the focus will be on the second point. In particuar, we will try to develop algorithm for reconstructing robot position and orientation as well as methods for Simultaneous Localization And Mapping Problem.

1.2 SLAM: State of the art

The Simultaneous Localization And Mapping (SLAM) problem has been introduced for the first time in the 80's by Smith and Cheeseman [6] and it has received very considerable attention in recent years. This problem involves the parallel estimation of a mobile robot position and orientation and of the environment where the robot moves. The SLAM problem is considered as *chicken and egg* problem in mobile robotics and it can be hard to find a reliable solution to this problem. The main difficulty about SLAM lies in the strong correlation between the mapping task and the localization task: the robot needs to build a map of its surrounding environment and to simultaneously localize itself within this map.

In the literature, the SLAM problem has been faced using various sensors types, e.g. wheel encoders, laser range sensors, sonar range sensors and cameras. Each SLAM algorithm is really 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. All these sensors allow the robot to observe (though only partially and inexactly) itself and the world where it moves. In [7] a laser sensor based SLAM algorithm for mobile robots in outdoor environments has been developed. The authors solve the problem

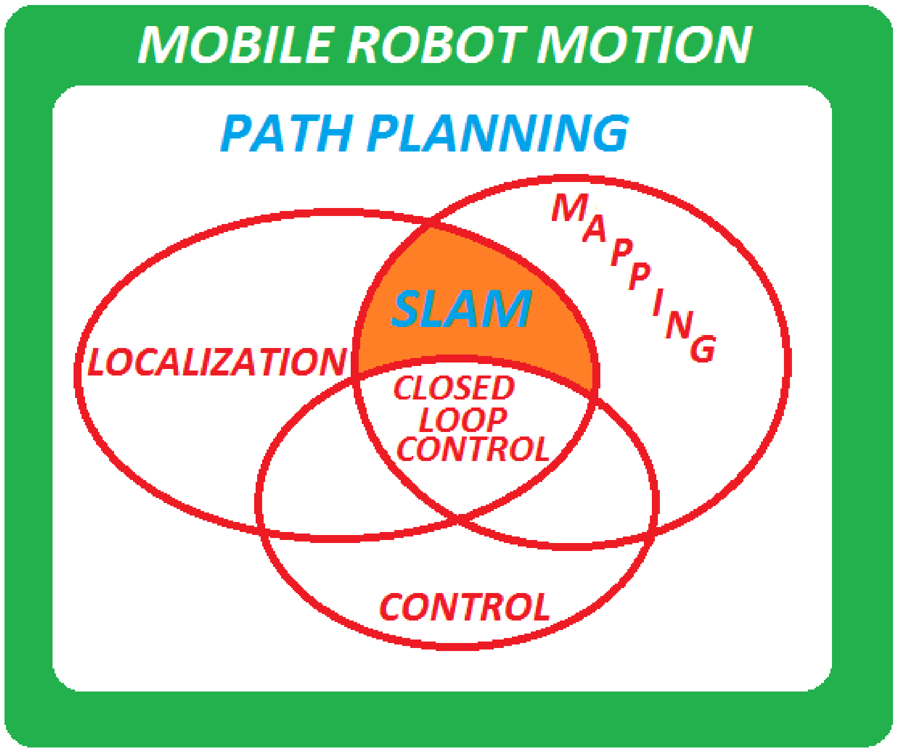


Fig. 1.3. Robot motion scheme

starting from an unknown robot initial position and using information from laser sensors through an Extended Kalman filter (EKF).In [8], an adaptive Occupancy Grid Mapping is presented, while FastSLAM(see [9]) is used to create and update a set of weighted particles representing the robot pose; at each step, laser measurements are acquired to update the grid resolution. More precisely, a tree-structure is used: each cell is splitted into two children cells if the acquired measurements contain conflicting information about its occupation; on the other side, two children cells with a common parent are merged if the standard deviation of their probability values is less than some predefined threshold and their mean is within a threshold distance from 0 (certainly free) or 1 (certainly occupied), or if all probability values are greater than a predefined value or lower than another predefined value.

In [10] the authors present a method to perform SLAM using the Rao-Blackwellised particle filter (RBPF) for monocular vision-based autonomous robot in an unknown indoor environment. In this work the particle filter is combined with an Unscented Kalman filter (UKF) and the environment mapping is implemented through the unscented transform. To solve the SLAM problem a monocular CCD camera, mounted on the robot, is used. This camera tracks a set of landmarks to obtain the environment mapping. Particle filter and vision are used also in [11], but a new probabilistic approach is used for landmarks position estimation instead of the unscented transform is proposed. In particular, at each step, the measurements likelihood is built as a convolution of two Gaussian distributions; the first one is the current estimate of the landmark position, while the other one is a Gaussian with zero mean and covariance matrix equal to the sensor model covariance, which is a given algorithm parameter.

Laser sensors and cameras are probably the best sensors used to solve the SLAM problem. However, these sensors present some drawbacks. Laser rangefinders are very accurate sensors but they can be very expensive. Cameras provide a large amount of information but it may be difficult to extract this information starting from the camera image and moreover the detection ability of a camera can be very influenced by the environment structure (e.g. dark or transparent environment). Sonar sensors represent one of the possible alternatives to laser sensors and cameras, these sensors are widely used in many applications for several reasons: they are less expensive than laser scanners and range cameras and they work well also in dark or transparent environments where cameras fail.

Many works can be found in the literature facing the SLAM problem using ultrasonic sensors. In [12] the SLAM problem has been solved using ultrasonic sensors and mapping the environment as a set of lines. In [13] Tardos et al. describe a technique for SLAM problem using standard sonar sensors and the Hough transform [14] to detect corners, edges and walls into the environment starting from the acquired sonar data. In [15] the authors use sonar measurements along with a particle filter to solve the SLAM problem in non-static environments.

In most of these works ([7, 8, 10, 16]) the authors use a scanner rangefinder sensor to solve the SLAM problem and thus the proposed solutions rely on accurate and dense measurements.

1.3 Robot model

There are different kinds of ground vehicle, each one distinguished by the others depending on the number of driving wheels. Some examples of ground robot are shown in Fig. 1.4, equipped with 2 (1.4(a),1.4(b),1.4(c)) or 4 driving wheels (1.4(d),1.4(e),1.4(f)). However, to the purposes of the subject of this chapter, there is no difference among robots with different numbers of wheels; the most important thing is that the robot has to acquire measurements to be used as input data of a SLAM algorithm. In the present research, a battery-powered mobile robot with two independent driving wheels and one castor wheel, that is Khepera III (see Fig. 1.4(b)), has been used. An approximated, discrete-time model of such a robot, which neglects the motors dynamic and the frictions, is [17]:

$$\begin{cases} x^1_{k+1} = x^1_k + V_k\,T\cos(\theta_{k+1}) + w^x_k \\ x^2_{k+1} = x^2_k + V_k\,T\sin(\theta_{k+1}) + w^y_k \\ \theta_{k+1} = \theta_k + \Delta_k + w^\theta_k, \end{cases} \tag{1.1}$$

where the state of the system is:

- (x^1_k, x^2_k): the position of the robot center at time t_k,
- θ_k: the angle between the robot axle and the x^1-axis,
- $V_k = r\,(\omega^l_k + \omega^r_k)/2$: the linear velocity of the robot,
- ω^l_k and ω^r_k: the angular velocities of wheels,
- w^x_k, w^y_k, w^θ_k: zero-mean uncorrelated Gaussian noises,
- r: the radius of the wheels,
- l: the length of the axes,
- $\Delta_k = \frac{r(\omega^l_k - \omega^r_k)T}{l}$: the rotation within $[t_{k-1}, t_k]$,
- $T = t_k - t_{k-1}$: the sampling period.

The input variables of the model are the angular velocities of the wheels, denoted by ω^l_k and ω^r_k, that are usually pre-computed so that the robot follows the desired trajectories.

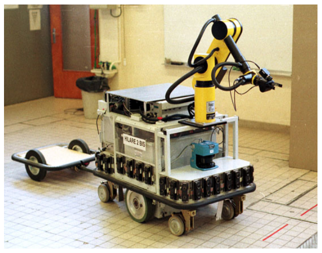
The Gaussian disturbances take into account for unmodeled dynamics, friction, wheels slipping and external disturbances. A typical representation of a differential-drive robot is shown in Figure 1.5

1.4 Model output equation and environment modeling

The robot is supposed to be equipped with five ultrasonic sensors (this is the case of the Khepera III), located as shown in Figure 1.6 and denoted by $S_i,\ i = 1, \dots, 5$. An ultrasonic sensors is a device which produces sound-waves at high frequency (usually around 40 khz) and and waits for a back echo. Then the distance is measured evaluating the elapsed time between the signal emission and the echo reception. A typical esample of this kind of sensor is represented in Figures 1.7(a). In an ideal case, the sensor should measure the distance from the point just in front of him; the problem is that the sonar beam is not a straight line (see Figure 1.7(b)), so a very common situation is represented in Figure 1.8, where the sensor detects not a real obstacle for the robot movement. However, this problem wil be faced more in detail in next sections.

1.4.1 On Board Sensors output equation

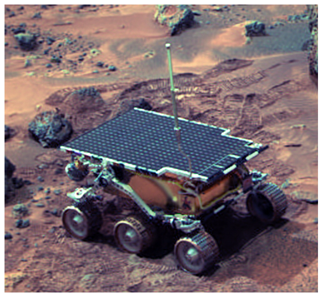
Whitout doing any restrictive hypothesis, the environment where the robot moves is usually unknown, thus to yield the output equation related to the on


(a) Hilare2bis robot

(b) Khepera III

(c) Yamabico robot

(d) B21 robot


(e) Sojourner robot

(f) Atrv robot

Fig. 1.4. Examples of mobile robots

board sensors, a model for the environment boundary is needed. The simplest choice is to model the environment by a set of segments such that each of them intersects at least one point of the boundary(see blue lines in Fig. 1.10(a)). The measurement r_i, provided by sensor S_i, is approximated, as shown in Figure 1.10(a), by the distance between P and the intersection point, denoted

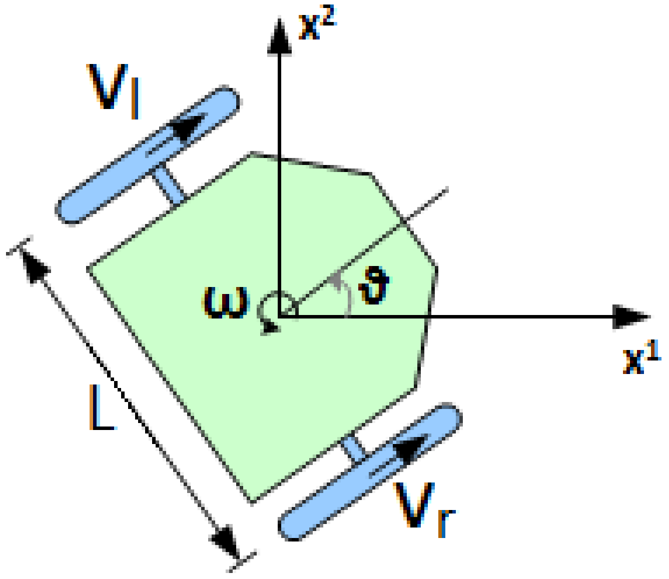


Fig. 1.5. A model of differential drive robot

by $\bar{P}_i = (\bar{x}_i^1, \bar{x}_i^2)$, between the axis of sensor S_i and one of the environment modeling segments.

Marking the axis of sensor S_i as $x^2 = a_i x^1 + q_i$, and denoting the axis of the environment modeling segment as $x^2 = c_i x^1 + s_i$, the intersection point $\bar{P}_i$ is given by

$$\bar{x}_i^1 = \frac{s_i - q_i}{a_i - c_i} \quad , \quad \bar{x}_i^2 = \frac{a_i s_i - c_i q_i}{a_i - c_i}, \tag{1.2}$$

therefore the distance between P and $\tilde{P}_i$ is approximated by the distance between P and $\bar{P}_i$, which is given by

$$\eta_i = \sqrt{(x^1 - \bar{x}_i^1)^2 + (x^2 - \bar{x}_i^2)^2}. \tag{1.3}$$

Now let γ_i be the orientation of each sensor S_i with respect to the robot axis (orthogonal to the wheel axes); the axis of sensor S_i is then given by:

$$a_i = \tan(\theta + \gamma_i),\, q_i = x^2 - a_i x^1, \tag{1.4}$$

and using (1.2) and (1.4) within (1.3) a distance function $h^{(1)}$, depending only on the robot state and segment (s_i, c_i), can be obtained

$$r_i \approx \eta_i = h^{(1)}((x^1, x^2, \theta), (s_i, c_i)),\; i = 1, \ldots, p.$$

These relationships allow to define the following output equation for the on board sensors

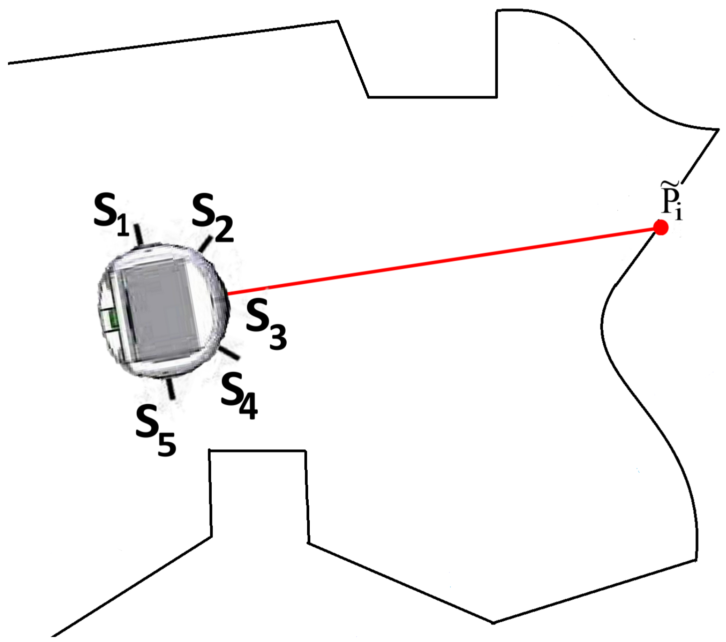


Fig. 1.6. Measurements by on board sensors

$$r_k \approx h^{(1)}(x_k, (\bar{s}_k, \bar{c}_k)) + v_k^{(1)}, \tag{1.5}$$

where $x_k = [x_k^1 \; x_k^2 \; \theta_k]^T$ is the robot state at time k and the vector $v_k^{(1)}$ collects the sensor noises, also assumed Gaussian, zero-mean and uncorrelated with $w_k = [w_{k,x} \; w_{k,y} \; w_{k,\theta}]^T$; the vectors $\bar{s}_k$ and $\bar{c}_k$ contain the parameters (s, c) of the segments hit by the p sensors at time k.

Similarly, more complex models can be used, based on n-degree polynomials (see Figure 1.10(b)); in this case, the computational load is greater, but the results accuracy can be increased. In this case, an output equation can be defined as

$$r_k \approx h^{(1)}(x_k, (\bar{p}_k) + v_k^{(1)}, \tag{1.6}$$

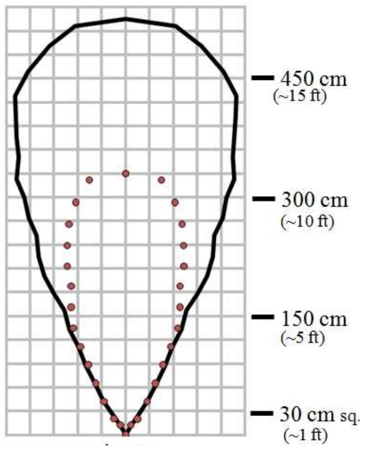
where $\bar{p}_k$ represents the polynomials intercepted by the sensors; each n-degree polynomial is defined by n+1 coefficients.

1.4.2 Out of Board Sensors output equation

As shown in Figure 1.11, the measurement provided by each out of board sensor F_i can be modeled as the distance between the point (F_i^1, F_i^2) and the center of the robot $P = (x_1, x_2)$. That is

$$d_i = h^{(2)}((x^1, x^2, \theta), (F_i^1, F_i^2)) = \sqrt{(x^1 - F_i^1)^2 + (x^2 - F_i^2)^2}$$
$$i = 1, \ldots, q$$

(a) An ultrasonic distance sensor



(b) An ultrasonic sensor beam

Fig. 1.7. Example of ultrasonic sensor

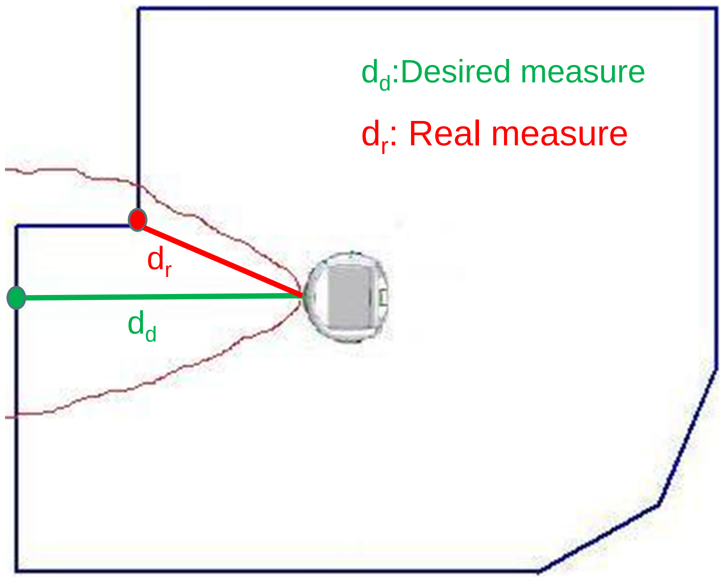


Fig. 1.8. Problems with ultrasonic measurements

which can be written in compact form as

$$d_k = h^{(2)}(x_k, \bar{F}) + v_k^{(2)} \tag{1.7}$$

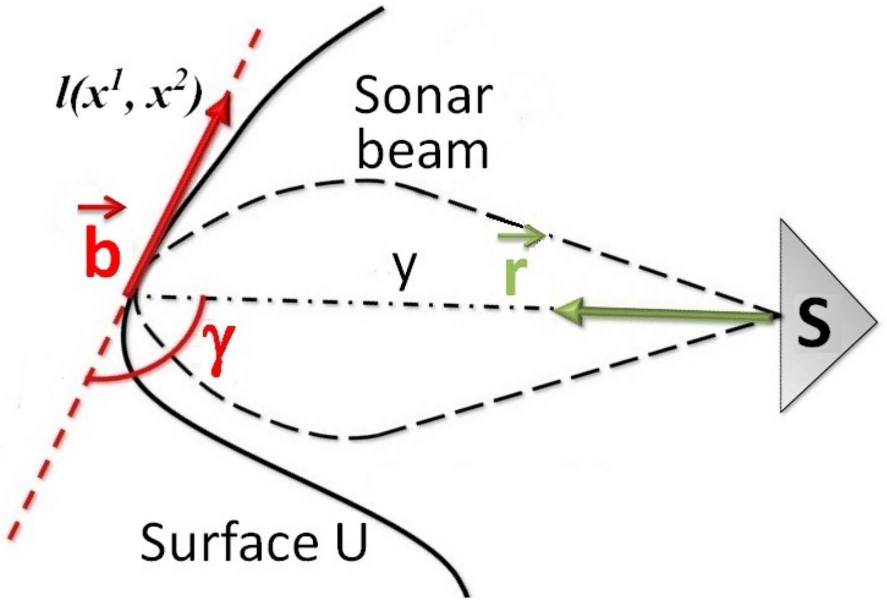
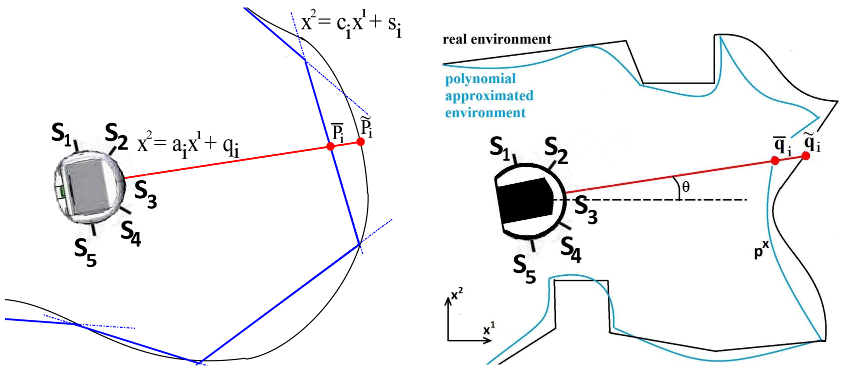


Fig. 1.9. Incidence Angle



(a) Segments based environment approximation

(b) Polynomials based environment approximation

Fig. 1.10. Environment models

where 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 $v_k^{(1)}$, while $\bar{F}$ contains the points $F_i, i = 1, \ldots, q$.

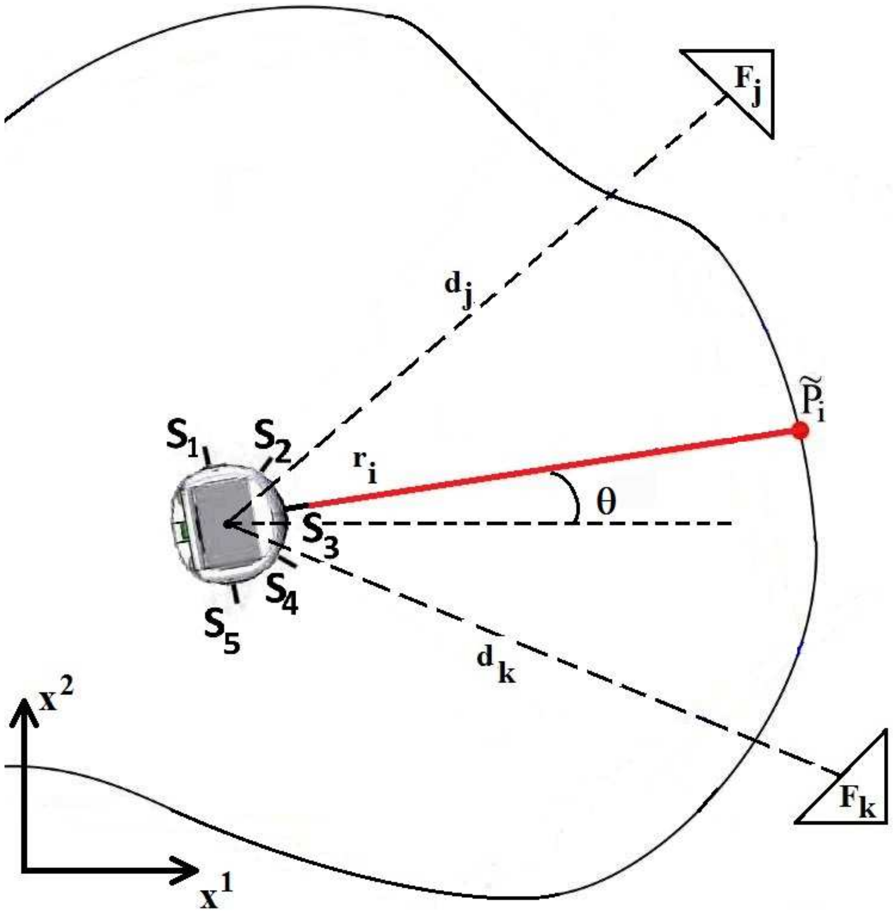


Fig. 1.11. On board and out of board sensors.

1.5 State estimation using the Kalman Filter

Kalman filter provides an optimal estimation method to approximate the state of a dynamic in case of noisy measurements. It is a dynamic system whose output is a gaussian probability density function (PDF) with zero mean and covariance matrix P. There are two fundamental hypothesis to satisfy to use Kalman Filter:

- the system to be analized has to be described by a linear model;
- the sensors model has to be a linear function of the system state.

It has been proved also that estimation techniques based on Kalman Filter fulfill accuracy and robustness criteria.

1.5.1 Linear case

Consider a discrete time, linear and time-invariant state space system:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k = Cx_k + v_k \end{cases} \tag{1.8}$$

where

- $x_k \in \mathbb{R}^n$ is the state vector;
- $u_k \in \mathbb{R}^m$ is the input vector;
- $y_k \in \mathbb{R}^p$ is the output vector;
- $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$ describe the dynamic of the system; more precisely, A describes the internal dynamic, while B models the influence of the input on the system state evolution;
- $C \in \mathbb{R}^{p\times n}$ is the output matrix, which models the relation between state and observed outputs;
- $w_k \in \mathbb{R}^n$ is the "process noise", it is used to characterize the uncertainty of the model, with particular reference to unmodeled disturbances;
- $v_k \in \mathbb{R}^p$ is the output noise and it is related expecially to intrinsic sensors characteristics.

In order to use such an estimation method, some hypothesis must be fulfilled:

- At each step, the dynamic of the system (A, B, C) is perfectly known;
- the initial state x_0 is a gaussian variable with mean equal to $\tilde{x}_0$ and covariance matrix P_0 positive semidefined;
- $w_k \sim WN(0, W^{n\times n})$, i.e. the state noise is a white noise, with zero mean and covariance matrix W known
- $v_k \sim WN(0, W^{p\times p})$, v_k is the output noise and, as w_k, is a white noise, with zero mean and covariance matrix V known
- w_k and v_k are uncorrelated
- w_k and x_k are uncorrelated

Depending on the number of used measurements, two cases can be distinguished:

- Kalman predictor, when the state estimation at step k is performed using measured inputs and outputs up to step k-1;
- Kalman filte, when the state estimation at step k is performed using all the measured inputs and outputs

1.5.1.1 Kalman predictor

A Kalman predictor has the following recursive structure:

$$\hat{x}_{k+1|k} = Ax_{k|k-1} + Bu_k + L_k[y_k - \underbrace{C\hat{x}_{k|k-1}}_{\hat{y}_{k|k-1}}] \tag{1.9}$$

where

- (A, B, C) are the already mentioned system dynamic matrix.
- L_k is the time-varying gain of a state observer.

The prediction error can be defined as

$$e_{k|k-1} \equiv x_k - \hat{x}_{k|k-1} \tag{1.10}$$

Combining the first equation of the System 1.8 and the Eq. (1.10) we obtain:

$$e_{k+1|k} \equiv A_k^c \hat{e}_{k|k-1} + B_k^c z_k \tag{1.11}$$

where

$$A_k^c \equiv A - L_k C \ , \ B_k^c \equiv [\, I - L_k \,] \ , \ z_k \equiv \begin{bmatrix} w_k \\ v_k \end{bmatrix}$$

Thus, the estimation error evolution has been modeled by another linear and time-varying dynamic system, where the noises affecting the original system are considered as inputs. The covariance matrix is defined as

$$\hat{P}_{k|k-1} \equiv E[e_{k|k-1} e_{k|k-1}^T], \hat{P}_{k|k-1} \in \mathbb{R}^{n \times n}$$

while the initial condition is:

$$E[e_{0|-1} = \emptyset_n]$$

that means the intial prediction value is the mean value of the real state. The goal is to find a proper sequence of gains, in order to minimize the prediction error. This can be done solving an optimization problem, where a quadratic form defined by the error covariance matrix is minimized

$$L_{i, i\in[0 \quad k]} = \arg \min_{L_j, j\in[0 \quad k]} \zeta^T \hat{P_{k+1|k}} \zeta$$

where $\zeta \in \mathbb{R}^n$. It can be proved that the solution is

$$L_i = A\hat{P}_{i|i-1} C^T (C\hat{P}_{i|i-1} C^T + V)^{-1} \ , \ \ i \in [1 \quad k] \tag{1.12}$$

where $\hat{P}_{i|i-1}$ is a solution of the differences Riccati's equation:

$$\hat{P}_{i+1|i} = A\hat{P}_{i|i-1} A^T + W - A\hat{P}_{i|i-1} C^T (C\hat{P}_{i|i-1} C^T + V)^{-1} C\hat{P}_{i|i-1} A \tag{1.13}$$

assuming $\hat{P}_{0|-1} = P_0$.

Under the hypothesis that

- (A, B_w) is stabilizable, where $B_w^T B_w = W$
- (A, C) is detectable

it can be proved that

- $\lim_{k\to\infty} \hat{P}_{k|k-1} = \bar{P}$, where $\bar{P}$ is the solution of the algebric Riccati's equation
$$\bar{P} = A\bar{P}A^T + W - A\bar{P}C^T (C\bar{P}C^T + V)^{-1} C\bar{P}A$$
- the predictor gain L_k converges to
$$\bar{L} = A\bar{P}C^T (C\bar{P}C^T + V)^{-1}$$
and the matrix $(A - \bar{L}C)$ is asimptotically stable.

1.5.1.2 Kalman filter

The optimal filter has also a recursive structure:

$$\hat{x}_{k|k} = A\hat{x}_{k-1|k-1} + Bu_{k-1} + K_k[y_k - \underbrace{C(A\hat{x}_{k-1|k-1} + Bu_{k-1})}_{\hat{y}_{k|k}}] \tag{1.14}$$

The gain sequence K_k is different from the gains sequence L_k calculated for the predictor. As it has been done for the prediction, the estimation error can be defined as

$$e_{k|k} \equiv x_k - \hat{x}_{k|k} \tag{1.15}$$

and the Eq. (1.14) can be written depending on this definition

$$e_{k|k} = (I - K_kC)[Ae_{k-1|k-1} + w_{k-1}] - K_kv_k \tag{1.16}$$

In thit case, the covariance matrix is updated at each step as

$$\hat{P}_{k|k} = \hat{P}_{k|k-1} - \hat{P}_{k|k-1}C^T[C\hat{P}_{k|k-1}C^T + V]^{-1}C\hat{P}_{k|k-1} \tag{1.17}$$

$$\hat{P}_{k+1|k} = A\hat{P}_{k|k}A^T + W \tag{1.18}$$

where

$$\hat{P}_{0|-1} = \hat{P}_0 \tag{1.19}$$

The connection between prediction and estimation can be defined now

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k[y_k - C\hat{x}_{k|k-1}] \tag{1.20}$$

The Eq. (1.20) states that the state prediction at atep k-1(first addend) is updated by using an optimal gain and the innovation term (second addend). Then, a new state prediction can be calculated

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{1.21}$$

By replacing the equation (1.20) and considering the structure of the predictor (Eq. (1.9)), a connection between predictor gain and filter gain can be established:

$$L_k = AK_k \tag{1.22}$$

1.5.2 EKF (Extended Kalman Filter)

As mentioned in the introduction about Kalman filter, either the system model and the sensor model must be linear to use the filter, but this is a highly restrictive hypothesis for real systems; in fact, first examples of state estimation using Kalman filter were based on a modified version of the algoruthm, that is EKF (Extended Kalman Filter, see [18]).

Consider a generic state space system

$$\begin{cases} x_{k+1} = \phi(x_k, u_k) + w_k \\ y_k = \eta(x_k) + v_k \end{cases} \tag{1.23}$$

Define an differential value for input, state and output as:

$$\begin{aligned} \Delta x_k &= x_k - x_{rk} \\ \Delta u_k &= u_k - u_{rk} \\ \Delta y_k &= y_k - y_{rk} \end{aligned}$$

Then, the Taylor expansions of the functions $\phi(\cdot,\cdot)$ e $\eta(\cdot,\cdot)$ are evaluated around the reference values x_{rk}, u_{rk} e y_{rk} and truncated to the first order:

$$\begin{aligned} \phi(x_k, u_k) &\approx \phi(x_{rk}, u_k) + \bar{A}_k[x_k - x_{rk}] + \bar{B}_k[u_k - u_{rk}] \\ \eta(x_k, u_k) &\approx \eta(x_{rk}, u_k) + \bar{C}_k[x_k - x_{rk}] \end{aligned}$$

where:

$$\begin{aligned} \bar{A}_k &= \left.\frac{\partial \phi}{\partial x}\right|_{[x_{rk}]} \\ \bar{B}_k &= \left.\frac{\partial \phi}{\partial u}\right|_{[u_{rk}]} \\ \bar{C}_k &= \left.\frac{\partial \eta}{\partial x}\right|_{[x_{rk}]} \end{aligned} \tag{1.24}$$

Then, a linear model can be defined to be used to design a predictor or a filter:

$$\begin{cases} \Delta x_{k+1} = \bar{A}_k \Delta x_k + \bar{B}_k \Delta u_k + w_k \\ \Delta y_k = \bar{C}_k \Delta x_k + v_k \end{cases} \tag{1.25}$$

The prediction algorithm for such a system is:

$$\begin{aligned} &\Delta\hat{x}_{k+1|k} = \bar{A}_k \Delta\hat{x}_{k|k-1} + \bar{L}_k[\Delta y_k - \bar{C}_k \Delta\hat{x}_{k|k-1}] \\ &L_k = \bar{A}_k \hat{P}_{k|k-1} \bar{C}_k^T [\bar{C}_k \hat{P}_{k|k-1} \bar{C}_k^T + V]^{-1} \\ &\hat{P}_{k+1|k} = \bar{A}_k \hat{P}_{k|k-1} \bar{A}_k^T + W + \\ &-\bar{A}_k \hat{P}_{k|k-1} \bar{C}_k^T [\bar{C}_k \hat{P}_{k|k-1} \bar{C}_k^T + V]^{-1} \bar{C}_k \hat{P}_{k|k-1} \bar{A}_k \end{aligned}$$

In the same way, the estimation algorithm can be written. Combining the prediction algorithm and the estimation algorithm, the fundamental steps to use an EKF can be written:

$$\hat{x}_{k|k-1} = \phi(\hat{x}_{k-1|k-1}, u_{k-1}) \tag{1.26}$$

$$\begin{aligned} \hat{P}_{k|k-1} &= \bar{A}_{k-1} \hat{P}_{k-1|k-1} \bar{A}_{k-1}^T + W \\ K_k &= \hat{P}_{k|k-1} \bar{C}_k^T [\bar{C}_k \hat{P}(k \mid k-1) \bar{C}_k^T + V]^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k[y_k - \eta(\hat{x}_{k|k-1})] \\ \hat{P}_{k|k} &= \hat{P}_{k|k-1} - K_k \bar{C}_k \hat{P}_{k|k-1} \end{aligned} \tag{1.27}$$

For what concerns the choice of the reference values x_{rk} to calculate the linearized model, two possibilities are:

- the model can be linearized around a predefined trajectory; in this way, filter gains can be calculated offline, but the estimation performances may be poor if the real state was far from the chosen trajectory;
- at each step, the model can be linearized aroun the predicted state value $\hat{x}_{k|k-1}$ (or around the estimated value $\hat{x}_{k-1|k-1}$) at the previous step; this choice is computationally more expensive but, since the prediction (and the estimation) are updated at each step, the obtained model is expected to be more accurate w.r.t the previous case.

1.5.3 Algorithm performance evaluation

To evaluate the performance of a SLAM algorithm, a simple RMS index can be defined as

$$\bar{\nu} = \frac{1}{k_f + 1} \sum_{k=0}^{k_f} ||x_k - \hat{x}_{k|k}|| \tag{1.28}$$

Another index, called NEES (Normalized Estimation Error Squared, [19]) can be used. This index is defined as

$$\bar{\mu} = \frac{1}{k_f + 1} \sum_{k=0}^{k_f} \left[x_k - \hat{x}_{k|k}\right]^T P_{k|k} \left[x_k - \hat{x}_{k|k}\right] \tag{1.29}$$

where

- x_k is the real robot state;
- $\hat{x}_{k|k}$ is the estimated robot state;
- $P_{k|k}$ is the robot state estimation error covariance matrix;
- k_f is the number of steps in which each experiment/simulation is performed.

1.6 Sensor fusing for mobile robot localization using a convex combination of two Kalman Filters

The sensor fusion is the process of combing information from a number of different sources to provide a robust and complete description of an environment or process of interest. In other words, starting from a set of noisy measurements, the sensor fusion techniques are used to obtain a better process description, less influenced by noise w.r.t. the starting one.

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 measuring the same variable is high, the higher is the probability of extracting the real information from these measurements neglecting noise effects.

Kalman filter and its nonlinear versions (e.g. the Extended Kalman filter and the Unscented Kalman filter [20],[21]) are some of the most common techniques to fuse sensors information. In the literature the Kalman filter has been widely used in various contexts. In [22] an adaptive Kalman filter is used to improve performance of an electrocardiogram (ECG) monitoring system. The authors propose the use of consecutive ECGs and of a sequential averaging filter that adaptively varies the number of complexes included in the averaging based on the characteristics of the ECG signal, in order to obtain good monitoring performance without increasing the system computational cost. In [23] a marine INS/GPS adaptive navigation system is presented and a novel Kalman filter is proposed, based on the maximum likelihood criterion for the proper computation of the filter structure. In [24] the authors propose a vision-based obstacle detection system for an helicopter flight. In this system a range map is computed using a sequence of images from a passive sensor, and an Extended Kalman filter is used to estimate range to obstacles. Other examples about sensor fusing using the Kalman filter can be found in radar tracking [25], orientation estimation [26] and in many other science fields.

The Kalman filter theory is widely used in robotics applications too. Concrete examples can be found in [27] where the Kalman filter is used to help the robot vision system and in [28], [29], [30], where the Extended and the Unscented Kalman filters are used to solve the mobile robot localization problem. The above problem has received considerable attention in the literature especially in recent years. The various proposed solutions differ each other essentially for the assumptions about the environment where the robot moves, for the used sensors types and for the algorithm chosen to solve the problem.

The Kalman filter theory can be suitably adapted to solve the localization problem in a perfectly known workspace, as shown in [31], where an Extended Kalman filter is used to estimate a differential drive robot pose (position and orientation) in a known environment. If the environment is unknown, as shown in [28], [32], [33], [34] a possible solution is to use a set of (external) out of board distance sensors placed in known locations. In this configuration, the robot position can be estimated using the information about the locations of the external sensors. Other possible solutions use distance sensors placed on the robot, as shown in [30]. Each of these approaches has advantages and drawbacks. In particular, using external sensors, good results can be obtained about the robot position estimation, while the orientation estimation can be quite unreliable; on the contrary, the on board sensors information can be directly related to the robot heading, as shown in [30] 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, the obtained localization performance 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. 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.

A possible way to overcome these problems is to fuse the entire information from all the available sensors.

In the literature many works can be found about the use of two Kalman filters, however, to the best of our knowledge, in these works each filter deals with the estimate of different variables. For example, in [35],[36], the authors use a two stage Kalman filter in order to simultaneously estimate the model parameters, thanks to the first Kalman filter, and the system's state, using the second Kalman filter.

However, fusing a large number of sensors measurements can yield to very high computational costs and difficulties to satisfy time constraints. To overtake these problems, the idea proposed in this section is to use two different filters to estimate the same state variables. The first filter is based on measurements provided by robot on board distance sensors while the second one uses out of board distance sensors measurements. The two obtained state estimates are then suitably combined to emphasize the qualities and overcome the defects of each used sensor. 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 is proposed and tested in a mobile robot application.

1.6.1 System model

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

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ \\ y_k^{(1)} = C^{(1)}x_k + v_k^{(1)} \\ y_k^{(2)} = C^{(2)}x_k + v_k^{(2)} \end{cases} \tag{1.30}$$

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. Let W, $V^{(1)}$ and $V^{(2)}$ be the covariance matrices of the noises w_k, $v_k^{(1)}$ and $v_k^{(2)}$, respectively. 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.

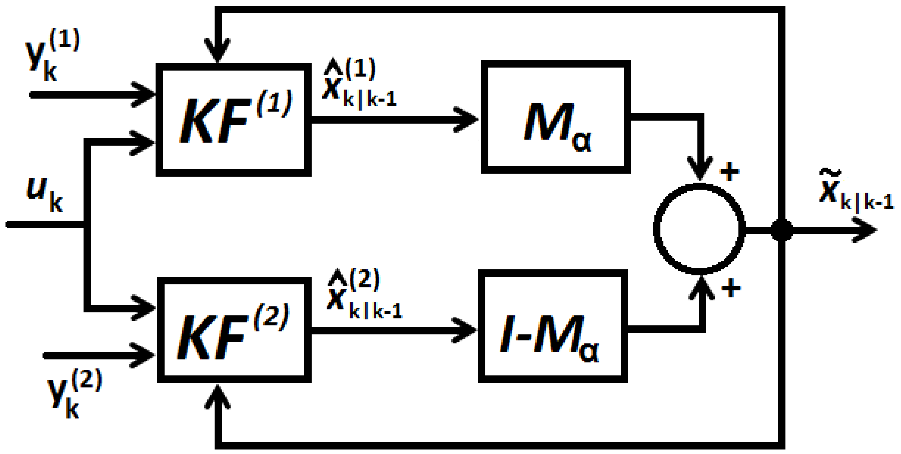


Fig. 1.12. Prediction scheme based on convex combination of Kalman filters

To estimate the system state a possible solution is to use a standard Kalman filter:

Kalman Filter

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

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 $y_k^{(2)}$.

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 a 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

$$\tilde{x}_{k|k-1} = M_\alpha \hat{x}^{(1)}_{k|k-1} + (I - M_\alpha)\hat{x}^{(2)}_{k|k-1} \tag{1.31}$$

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 filters the following algorithm can be stated:

Mixed Kalman Filter (MKF)

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

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 $\tilde{x}_{k+1|k}$. The initial condition of this covariance matrix is $\tilde{P}_{0|-1} = P_{0|-1}$;
- $L^{(1)}_k$ and $L^{(2)}_k$ are two gains which have to be chosen properly in order 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 1.12 shows the overall prediction scheme in the proposed configuration. 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 $L^{(1)}_k$ and $L^{(2)}_k$ will be found.

1.6.2 Prediction 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 (1.30) the prediction error evolution is:

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

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

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

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

and since $M_\alpha + \tilde{M}_\alpha = I$ we obtain

$$\begin{aligned} e_{k+1|k} &= (A - (M_\alpha L^{(1)}_k C^{(1)} + \tilde{M}_\alpha L^{(2)}_k C^{(2)}))e_{k|k-1} + \\ &M_\alpha L^{(1)}_k v^{(1)}_k + \tilde{M}_\alpha L^{(2)}_k v^{(2)}_k - w_k \end{aligned} \tag{1.32}$$

At this point the function $\Phi(\cdot)$ can be found evaluating the $e_{k+1|k}$ covariance matrix.

Let $\Delta = (A - (M_\alpha L^{(1)}_k C^{(1)} + \tilde{M}_\alpha L^{(2)}_k C^{(2)}))$ and $\mathbb{E}[\cdot]$ be the expected value operator. Assuming that there is no correlation between the measurements noises, the state noise and the initial prediction $\hat{x}_{0|-1}$, the prediction error covariance matrix evolution is

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

$$= \mathbb{E}[e_{k+1|k}e^T_{k+1|k}] = \mathbb{E}[(\Delta e_{k|k-1} + M_\alpha L^{(1)}_k v^{(1)}_k +$$
$$\tilde{M}_\alpha L^{(2)}_k v^{(2)}_k - w_k)(\Delta e_{k|k-1} +$$
$$M_\alpha L^{(1)}_k v^{(1)}_k + \tilde{M}_\alpha L^{(2)}_k v^{(2)}_k - w_k)^T] =$$

$$= \mathbb{E}[\Delta e_{k|k-1}e^T_{k|k-1}\Delta^T + M_\alpha L^{(1)}_k v^{(1)}_k {v^{(1)}_k}^T {L^{(1)}_k}^T M^T_\alpha +$$
$$\tilde{M}_\alpha L^{(2)}_k v^{(2)}_k {v^{(2)}_k}^T {L^{(2)}_k}^T \tilde{M}^T_\alpha + w_k w^T_k]$$

And finally

$$\begin{aligned} \tilde{P}_{k+1|k} &= \Phi(\tilde{P}_{k|k-1}) = \Delta\tilde{P}_{k|k-1}\Delta^T + M_\alpha L^{(1)}_k V^{(1)} {L^{(1)}_k}^T M^T_\alpha + \\ &\tilde{M}_\alpha L^{(2)}_k V^{(2)} {L^{(2)}_k}^T \tilde{M}^T_\alpha + W \end{aligned} \tag{1.33}$$

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 ($\tilde{P}_{k+1|k} \geq 0$).

1.6.3 Optimal filter gains

Once the function $\Phi(\cdot)$ has been found, the optimal values for the gains L^1_k and L^2_k can be computed minimizing the covariance matrix $\tilde{P}_{k+1|k}$. The

$trace\{\tilde{P}_{k+1|k}\}$ has been chosen as a minimization index obtaining the following optimization problem:

$$\begin{cases} (L_k^{(1)}, L_k^{(2)}) = \arg \min\limits_{L_k^{(1)}, L_k^{(2)}} trace\{\tilde{P}_{k+1|k}\} \\ \\ \text{subject to} \\ \tilde{P}_{k+1|k} = \Phi(\tilde{P}_{k|k-1}) \end{cases} \tag{1.34}$$

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 it is possible to rewrite the optimization index as

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

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

$$\begin{aligned} \frac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^{(1)}} = & 2\tilde{L}_k^{(1)}(C^{(1)}\tilde{P}_{k|k-1}C^{(1)^T} + V^{(1)}) - 2A\tilde{P}_{k|k-1}C^{(1)^T} + \\ & 2\tilde{L}_k^{(2)}C^{(2)}\tilde{P}_{k|k-1}C^{(1)^T} \end{aligned} \tag{1.35}$$

and

$$\begin{aligned} \frac{\delta trace\{\tilde{P}_{k+1|k}\}}{\delta \tilde{L}_k^{(2)}} = & 2\tilde{L}_k^{(2)}(C^{(2)}\tilde{P}_{k|k-1}C^{(2)^T} + V^{(2)}) - 2A\tilde{P}_{k|k-1}C^{(2)^T} + \\ & 2\tilde{L}_k^{(1)}C^{(1)}\tilde{P}_{k|k-1}C^{(2)^T} \end{aligned} \tag{1.36}$$

To find the minimum values of the optimization index, the following equations have to be solved

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

The solution of the above equations are:

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

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

where
$$\Gamma = (C^{(2)}\tilde{P}_{k|k-1}C^{(2)^T} + V^{(2)} - C^{(2)}\tilde{P}_{k|k-1}C^{(1)^T}(C^{(1)}\tilde{P}_{k|k-1}C^{(1)^T} + V^{(1)})^{-1}C^{(1)}\tilde{P}_{k|k-1}C^{(2)^T}). \tag{1.37}$$

To ensure that the equations (1.37) 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

$$H = \begin{bmatrix} C^{(1)}\tilde{P}_{k|k-1}C^{(1)^T} + V^{(1)} & C^{(1)}\tilde{P}_{k|k-1}C^{(2)^T} \\ C^{(2)}\tilde{P}_{k|k-1}C^{(1)^T} & C^{(2)}\tilde{P}_{k|k-1}C^{(2)^T} + V^{(2)} \end{bmatrix} \tag{1.38}$$

If $q = p = 1$ is assumed, then $y_k^{(1)}$ and $y_k^{(2)}$ are both scalar values and the determinant of the Hessian matrix is

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

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

Since $\tilde{P}_{k|k-1}$ is a symmetric positive semi-definite matrix we can deduce that

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

and thus

$$D = \det\{H\} = V^{(1)}C^{(2)}\tilde{P}_{k|k-1}C^{(2)^T} + C^{(1)}\tilde{P}_{k|k-1}C^{(1)^T}V^{(2)} + V^{(1)}V^{(2)} \tag{1.39}$$

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 (1.37) define a minimum point for the optimization problem (1.34).

Remark 1.1. The obtained result is valid only if the system (1.30) has two scalar outputs. Otherwise, there is no assurance about the optimality of the gains defined by equations (1.37). Further studies are in progress to validate the obtained results also for systems having two non-scalar outputs. Moreover, as it will be shown in Section 1.6.6, experimental tests have been performed using a system with two non scalar outputs and very good results have been obtained using the MKF.

Remark 1.2. The equations (1.33) and (1.37) 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 prediction 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, consider a system with two outputs, $y_k^{(1)}$ and $y_k^{(2)}$, such that the sensors providing the first output give less information on the $j-th$ state variable than the sensors providing $y_k^{(2)}$. In this situation, α_j has to be chosen to emphasize the second output neglecting the first one, that is $\alpha_j \to 0$.

1.6.4 Measurements fusing algorithm

The aim of the mobile robot localization problem is to localize the robot using all the available information by sensors and by a-priori partial knowledge of the environment where the robot moves. To solve the mobile robot localization problem, the proposed solution consists in estimating the robot pose using a Kalman filter. The mobile robot is modeled through a set of nonlinear differential equations (1.1) and therefore the standard Kalman theory can not be used. In this context, the Extended Kalman filter (EKF) has been chosen to solve the filtering problem. In particular, using the two defined output equations, (1.6) and (1.7), two filters can be stated: the Neighbours based extended Kalman filter (NEKF) related to the first output equation and the Out of board sensors based EKF (OEKF) related to the second one.

In contrast of [31], the NEKF has been developed without doing any assumption about the environment where the robot moves and only sensors' measurements are used to obtain the robot pose. The basic idea behind the algorithm is that robot state can be estimated using only information about the environment portions that interact with the robot. The above concept is the basis of the implemented *Neighbours based Algorithm(NbA)* proposed in [30], which can be summarized as follows:

Neighbours Based Algorithm

```
At each step, given $\hat{x}_k^1, \hat{x}_k^2, \hat{\theta}_k, \mathcal{M}_k, \mathcal{I}_k, \mathcal{J}_k, \{(\hat{s}_i, \hat{c}_i), i \in \mathcal{J}_k\}$ do

1. for $i \in \mathcal{I}_k$
   • acquire measure $y_{i,k}$ from sensor $S_i$
   • $q_{i,1}^* = \hat{x}_k^1 + y_{i,k} \cos(\hat{\theta}_k + \alpha_i)$
   • $q_{i,2}^* = \hat{x}_k^2 + y_{i,k} \sin(\hat{\theta}_k + \alpha_i)$
   • $q_i^* = (q_{i,1}^*, q_{i,2}^*)$
   end
```

```
2. $\mathcal{M}_{k+1} = \mathcal{M}_k \cup \{q_i^*, i \in \mathcal{I}_k\}$
3. for $i \in \mathcal{I}_k$
   • $(\hat{s}_i, \hat{c}_i) = \mathrm{LMS}(\mathcal{N}(q_i^*, \mathcal{M}_{k+1}))$
   end
4. return $(\hat{s}_i, \hat{c}_i), i \in \mathcal{J}_k$
```

where

- $\mathcal{I}_k$ is the set of indexes related to the sensors used at step k;
- $\mathcal{J}_k$ is the set of sensor indexes whose intercepted segments are needed at step k;
- $[\hat{x}_k^1 \ \hat{x}_k^2 \ \hat{\theta}_k]$ is the robot state estimation at step k;
- α_i is the orientation of each sensor S_i with respect to the robot axis;
- $\mathcal{M}_k$ is the set of previously acquired environment points;
- $(\hat{s}_i, \hat{c}_i)$ are the approximations, at step k, of the parameters of the segment intercepted by sensor S_i axis;
- $\mathcal{N}_q = \mathcal{N}(q_i^*, \mathcal{M}_{k+1}) = \{q_j \in \mathcal{A} : ||q_j - q_i^*|| < R\}$ is the subset of points of $\mathcal{A}$ which are neighbours, in a radius R, of q;
- $(\hat{s}_i, \hat{c}_i) = \mathrm{LMS}(\mathcal{N}_q)$ returns the Least mean Square approximation $(\hat{s}_i, \hat{c}_i)$ of the straight line $x^2 = c \cdot x^1 + s$ through points in $\mathcal{N}_q$

From now on we will use the NBA as a function: $(\hat{\hat{s}}, \hat{\hat{c}}) = NBA(\mathcal{I}_k, \mathcal{J}_k, k)$ where $(\hat{\hat{s}}, \hat{\hat{c}})$ represent the output of NBA at time k and for each sensor $S_i, i \in \mathcal{J}_k$.

The above mentioned algorithm is used to approximate the environment boundaries, and then the intersections between the envirnonment and the sensors' beam. The NEKF is based on a standard EKF including the NBA:

Neighbors based Extended Kalman Filter (NEKF)

$$
\begin{aligned}
&L_k = A_k P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k^T + V)^{-1} \\
&(\hat{\hat{s}}_k, \hat{\hat{c}}_k) = \mathrm{NBA}(\hat{x}_{k|k-1}, r_k) \\
&\hat{x}_{k+1|k} = f(\hat{x}_{k|k-1}, u_k) + L_k(r_k - h^1(\hat{x}_{k|k-1}, (\hat{\hat{s}}_k, \hat{\hat{c}}_k))) \\
&P_{k+1|k} = (A_k - L_k C_k) P_{k|k-1} (A_k - L_k C_k)^T + W + L_k V L_k
\end{aligned}
$$

There $\hat{x}_{k+1|k}$ is the prediction of x_{k+1} starting from all the available information at time k and

$$
A_k = \left.\frac{\partial f(x_k, u_k)}{\partial x_k}\right|_{x_k = \hat{x}_{k|k-1}}, \quad C_k = \left.\frac{\partial \mathbf{h^1(x)_k}}{\partial \mathbf{x_k}}\right|_{x_k = \hat{x}_{k|k-1}}, \tag{1.40}
$$

For what concerns the out of board sensors a standard EKF algorithm based on output Eq. (1.7) can be defined using

$$A_k = \left.\frac{\partial f(x_k, u_k)}{\partial x_k}\right|_{x_k=\hat{x}_{k|k-1}}, \quad C_k = \left.\frac{\partial \mathbf{h^{(2)}(x)_k}}{\partial \mathbf{x_k}}\right|_{x_k=\hat{x}_{k|k-1}}, \tag{1.41}$$

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 heading information, the estimate on the robot orientation can be very noisy and inaccurate. Otherwise the NEKF algorithm, as shown in [30], is able to provide a good heading estimation but the state estimation is very affected by initial condition errors.

1.6.5 Mixed Extended Kalman Filter

The proposed MKF can be suitably adapted to the problem of estimating the state of a mobile robot. To this purpose, an Extended version of KF has to be used because of the nonlinearity of the system. Using the NEKF as the first filter and the OEKF as the second filter, the overall algorithm is

Mixed Extended Kalman Filter

```
first filter update:NEKF
$L_k^{(1)} = f_{L^{(1)}}(\tilde{P}_{k|k-1})$
$(\hat{\tilde{s}}_k, \hat{\tilde{c}}_k) = \mathrm{NBA}(\tilde{x}_{k|k-1}, r_k)$
$\hat{x}_{(k+1|k)1} = f(\tilde{x}_{k|k-1}, u_k) + L_k^1(r_k - h^1(\tilde{x}_{k|k-1}, (\hat{\tilde{s}}_k, \hat{\tilde{c}}_k)))$

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

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

where $L_k^{(1)}$ and $L_k^{(2)}$ are computed using $C_k^{(1)}$, $C_k^{(2)}$, $V^{(1)}$, $V^{(2)}$, (1.40) and (1.41) respectively and $V^{(1)}$ and $V^{(2)}$ are the covariance matrices related to the on board and to the out of board sensors' measurements noises. 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 recursive equation of the filters 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 whole output $y_k = [r_k \; d_k]^T$ but it uses both filters in a parallel way. In this way, the memory requirements and the computational cost of the filter are similar to the costs related to the use of the only NEKF or OEKF.

1.6.6 Experimental results

To evaluate the performance of the proposed filters, real experiments have been performed using the following parameters:

- $p = 5$ on board sensors placed as depicted in Fig. 1.6 and $q = 4$ out of board sensors;
- $l = 0.09m$, $\sigma = 0.0205m$ for the robot;
- Sampling period $T = 1s$;
- $V^{(1)} = 0.20^2 I_5$ and $V^{(2)} = 0.02^2 I_{4\times 4}$, a standard deviation of $0.20m$ for the measurements provided by the on board sensors and a standard deviation of $0.02m$ for the measurements from the out of board sensors;
- $W = diag\{0.01^2, 0.01^2, 0.0017^2\}$: a standard deviation of $0.01m$ on w_k^x and w_k^y, a standard deviation of 0.1°on w_k^θ;
- $\tilde{P}_{0|-1} = diag\{0.05^2, 0.05^2, 0.0873^2\}$: a standard deviation of $0.05m$ on robot position and a standard deviation of 5°on robot heading;
- $R = 0.1m$ for the NBA; the initialization of the set $\mathcal{M}$ have been formed during 5 initial steps at the beginning of each experiment: during these steps measurements and points of the environment are acquired, these points will form the initial condition for $\mathcal{M}$.

The following filters have been tested: an Extended Kalman filter (EKF) based on the whole output $y_k = [y_k^{(1)}\ y_k^{(2)}]^T$; 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).

The NEES index described in Eq. (1.29) has been used for evaluating the localization performance. The proposed fusing algorithm has been tested in the real experimental framework shown in Figure 1.13 using the robot Khepera III [37], equipped with 5 ultrasonic sensors, and four 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 boundary corners (as shown in [38]). The angular velocities have been precomputed so that the Khepera III starts from $(0.26, 0.5)m$, with $\theta_0 = 0rad$, passes by $(0.26\ 0.4)m$, $(0.26\ 0.5)m$, $(1.1\ 0.5)m$, $(1.1\ 0.4)m$, $(1.1\ 0.6)m$, $(1.1\ 0.5)m$, $(0.26\ 0.5)m$, $(0.26\ 0.6)m$ and comes back to $(0.26\ 0.5)m$. The path has been performed in $k_f = 200$ steps. 50 experiments have been performed setting $\hat{x}_{0|-1} = \tilde{x}_{0|-1} = [0.22\ 0.45\ 0]^T$ as initial condition for the filers. A typical result of the MEKF algorithm, using $M_\alpha = diag\{0.1\ 0.1\ 0.9\}$, is shown in Figure 1.14.

Table 1.1 shows the averaged NEES values over the *50* performed experiments. As it can be seen, the proposed *MEKF* algorithm performs better than the other filters (EKF, OEKF, NEKF) whatever is the M_α value. In

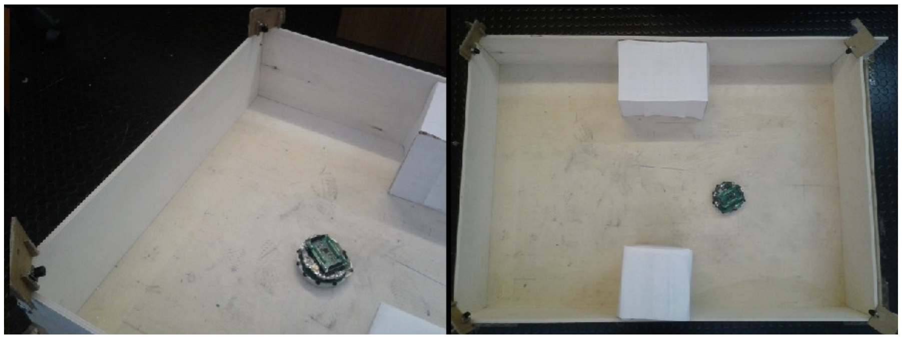


Fig. 1.13. experimental framework

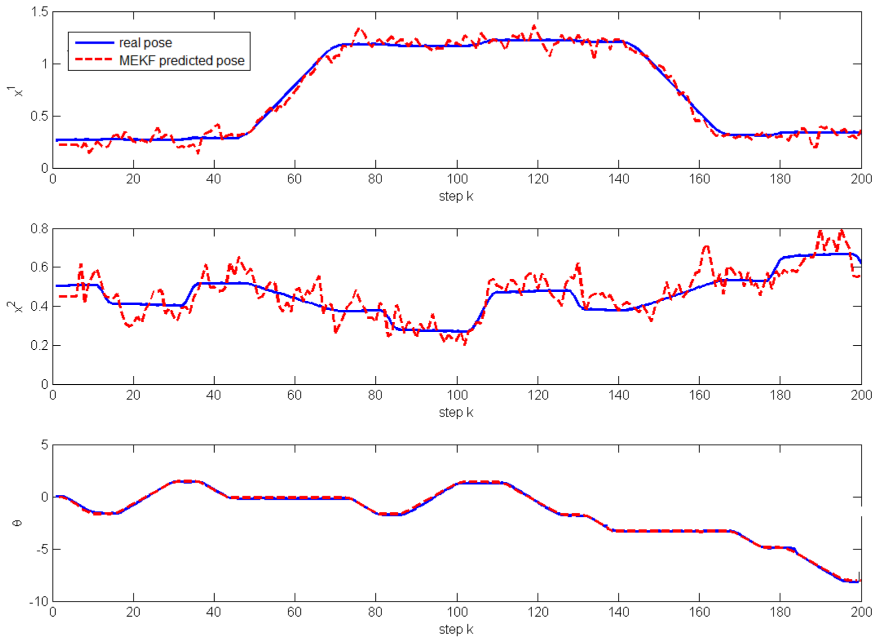


Fig. 1.14. MEKF experimental results using $M_\alpha = diag\{0.1, 0.1, 0.9\}$

particular, the best prediction performance has been obtained 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. This result is consistent with the previously described NEKF and OEKF properties. Since the NEKF is better than the OEKF to predict robot heading, while the OEKF is more reliable than the NEKF w.r.t. the robot position prediction,

Table 1.1. averaged NEES index, over the *50* experiments

Filter	ε
MEKF $M_\alpha = diag\{0.1, 0.1, 0.1\}$	$2.2967 \cdot 10^{-6}$
MEKF $M_\alpha = diag\{0.5, 0.5, 0.5\}$	$2.2552 \cdot 10^{-6}$
MEKF $M_\alpha = diag\{0.9, 0.9, 0.9\}$	$2.2739 \cdot 10^{-6}$
MEKF $M_\alpha = diag\{0.9, 0.9, 0.1\}$	$2.2803 \cdot 10^{-6}$
MEKF $M_\alpha = diag\{0.1, 0.1, 0.9\}$	$2.2101 \cdot 10^{-6}$
NEKF	$5.0482 \cdot 10^{-6}$
OEKF	$4.4899 \cdot 10^{-5}$
EKF	$2.29131 \cdot 10^{-6}$

using a low value of α_1, α_2 and a high value of α_θ the mixed filter prediction performance increases.

1.6.7 Remarks

In this section the localization of a mobile robot in an unknown environment has been faced using a new version of the Extended Kalman filter. The proposed Mixed Kalman filter is based on combining measurements provided by robot on board and out of board sensors, in order to emphasize the qualities and overcome the defects of each sensor. The prediction error covariance matrix evolution using the described MKF has been studied. Using the proposed MKF, two Kalman gains have to be computed and preliminary studies on the optimal values of these gains have been performed. The proposed fusing technique has been tested in a real experimental framework using the robot Khepera III. The algorithm has been contrasted with other Extended Kalman filters, based on: only on board sensors (NEKF), only out of board sensors (OEKF), both on board and out of board sensors (EKF). The obtained results are very encouraging. However, MEKF algorithm does not provide a complete solution to the SLAM problem, because the acquired measurements are simply clustered and used to localize the robot, without performing any environment structure reconstruction.

The complete SLAM problem will be addressed in the following section, where a detailed description of the environment mapping procedures is given.

1.7 SLAM algorithms

1.7.1 A SLAM algorithm for environmental structure mapping based on polynomials

Very often the SLAM algorithms assume to have at least some information about the environment (e.g. in [39] the environment is assumed to be partially known) or to model the environment in a very approximated way. For example in [40], the authors assume the robot placed in an environment modeled as a set of orthogonal-parallel lines.

On the contrary, in the present section, the proposed solution to the SLAM problem does not need any assumption about the robot surrounding environment. The proposed algorithm is based on a polynomial model for the surrounding environment boundaries and on an Extended Kalman filter. The basic idea behind the proposed algorithm is that robot position and orientation can be estimated using only information about the environment portions that interact with robots sensors. Moreover sparse measurements are used and the mapping problem is solved by fusing the new acquired measurements with the past environment maps. The above concept is the basis of the implemented *Polynomial based Extended Kalman Filter (PEKF)* and will be illustrated in Section 1.7.1.1 and Section 1.7.1.2.

As explained in [41] and [42] a SLAM algorithm is divided into 5 main parts: landmark extraction, data association, state estimation, state update and landmark update. Using the sonar measurements and the actual robot pose estimation, the data association and landmark extraction processes are performed and yield to the actual environment mapping. Starting from this mapping and from the model inputs, the state estimation, state update and landmark update processes provide the robot pose and update the environment map.

1.7.1.1 Landmark extraction and Data association

At each step k, using the state prediction $\hat{x}_{k|k-1}$ along with the measurements y_k provided by the on board ultrasonic distance sensors, an approximation of the intersection points between the sensors' axes and the environment boundaries, $\hat{q}_k^i, i = 1, \dots, n_S$, can be obtained by

$$\begin{aligned}
\hat{q}_k^{(i,1)} &= \hat{x}^1_{k|k-1} + y_{i,k}\cos(\hat{\theta}_{k|k-1} + \alpha_i) \\
\hat{q}_k^{(i,2)} &= \hat{x}^2_{k|k-1} + y_{i,k}\sin(\hat{\theta}_{k|k-1} + \alpha_i) \\
\hat{q}_k^i &= (\hat{P}_k^{(i,1)}, \hat{P}_k^{(i,2)})
\end{aligned} \tag{1.42}$$

where $y_{i,k}$ is the measurement provided by sensor S_i at time k. These points can be seen as an approximation of the points $\tilde{q}_i$ (see Figure 1.15) due to the estimation and measurements errors. Let $\mathcal{M}_{k-1} = \{\hat{q}_j^i, i = 1, \dots, n_S,\ j = 0, \dots, k-1\}$ be the set of all the acquired environment points until time step k. The main

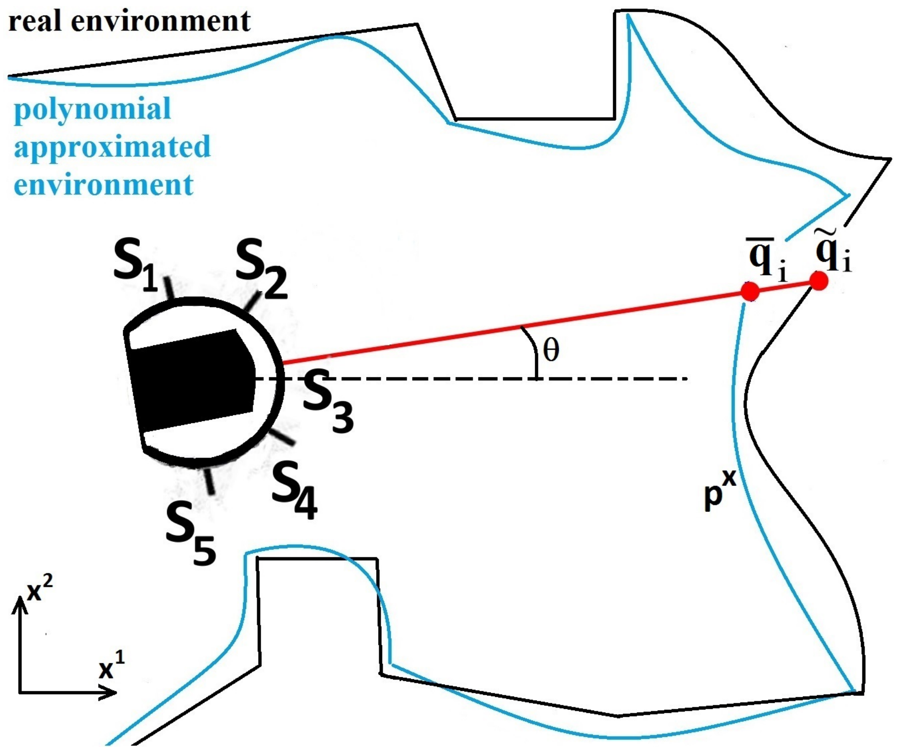


Fig. 1.15. Real environment (black line), polynomial based model environment (light blue line), robot on board sensors position

idea behind the proposed landmark extraction and data association algorithm is to approximate the environment bounds by clustering the set $\mathcal{M}_{k-1}$ into n_{k-1} subsets and associating a polynomial to each cluster. Each polynomial will be one of the SLAM landmarks.

To achieve this goal, given a set of points $\mathcal{A}$, a point $q = (x_q^1, x_q^2)$ and a polynomial p, represented by its coefficients $\{b_i\}_{i=0}^m$, three functions have been defined. The first one is $LMS(\mathcal{A}, z)$ and it computes the Least Mean Square z-th order polynomial approximating the points in $\mathcal{A}$. the second function is $ELMS(p, \mathcal{A})$ which computes the least mean square error due to the approximation of each point in $\mathcal{A}$ using p. The last function is the set division function $\mathcal{D}(\mathcal{A}, q, p)$. this function returns the partition $(\mathcal{A}_1, \mathcal{A}_2)$ of $\tilde{\mathcal{A}} = \{\mathcal{A} \bigcup \{q\}\}$, such that if p is a x^1-variate polynomial then $\mathcal{A}_1 = \{q_j = (x_j^1, x_j^2) \in \tilde{\mathcal{A}} : x_j^1 \leq x_q^1\}$, else if p is a x^2-variate polynomial, then $\mathcal{A}_1 = \{q_j = (x_j^1, x_j^2) \in \tilde{\mathcal{A}} : x_j^2 \leq x_q^2\}$. Finally $\mathcal{A}_2 = \tilde{\mathcal{A}} \setminus \mathcal{A}_1$.

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 $\mathbb{E}_{k-1} = \{p^i_{k-1}, i = 1, \dots, n_{k-1}\}$ where $p^i_{k-1} \in \mathbb{E}_{k-1}$ is the m-th order polynomial related to the set $B^i_{k-1} \in \mathcal{B}_{k-1}$. The process modifies the partition $\mathcal{B}_{k-1}$ and the polynomials set $\mathbb{E}_{k-1}$ into $\mathcal{B}_k$ and $\mathbb{E}_k$, respectively. As a first step, for each acquired measurement $y_{i,k}, i = 1, \dots, n_S$, given the current state prediction $\hat{x}_{k|k-1}$, a set of points $\Pi_k = \{\hat{P}^i_k, i = 1, \dots, n_S\}$ is computed using (1.42). Then, for each point $\hat{q}^i_k \in \Pi_k$ the data association step is performed; the main idea behind this process is closeness between acquired environment points. Two points q_1, q_2 are defined *neighbors* iff $||q_1 - q_2|| < R$, where R is a given algorithm parameter. Moreover, given a set of points $\mathcal{A}$ and a point q, the set-valued *closeness function* $\mathcal{N}$ has been defined by $\mathcal{B} = \mathcal{N}(q, \mathcal{A}) = \{q_i \in \mathcal{A} : ||q_i - q|| < R\}$, i.e. the subset of points of $\mathcal{A}$ which are neighbors, in a radius R, of q. The point $\hat{q}^i_k \in \Pi_k$ will be associated to the cluster B^j_{k-1} and, consequently, to the landmark p^j_{k-1}, such that

$$B^j_{k-1} = \max_{B^r_{k-1} \in \mathcal{B}_{k-1}} \{|\mathcal{N}(\hat{q}^i_k, B^r_{k-1})|\}$$

Thus B^j_{k-1} is the cluster which contains the biggest number of neighbours of $\hat{q}^i_k$.

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

$$\mathcal{B}_k = \mathcal{B}_{k-1} \bigcup \{\hat{q}^i_k\}; \quad \mathbb{E}_k = \mathbb{E}_{k-1}$$

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

$$B^j_{k-1} = B^j_{k-1} \bigcup \{\hat{q}^i_k\}; \quad \mathcal{B}_k = \mathcal{B}_{k-1}; \quad \mathbb{E}_k = \mathbb{E}_{k-1}.$$

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

$$\varepsilon_1 = ELMS(p^j_{k-1}, B^j_{k-1} \bigcup \{\hat{q}^i_k\}).$$

At this point, a new m-th order polynomial is obtained using the points in B^j_{k-1} along with the new acquired point:

$$p^{j,1}_{k-1} = LMS(B^j_{k-1} \bigcup \{\hat{q}^i_k\}, m)$$

and the related approximation error is

$$\varepsilon_2 = ELMS(p^{j,1}_{k-1}, B^j_{k-1} \bigcup \{\hat{q}^i_k\}).$$

Finally, the set B^j_{k-1} is partitioned into $(B^{j,1}_{k-1}, B^{j,2}_{k-1})$ as

$$(B^{j,1}_{k-1}, B^{j,2}_{k-1}) = \mathcal{D}(B^j_{k-1} \bigcup \{\hat{q}^i_k\}, \hat{q}^i_k, p^j_{k-1})$$

and the related polynomials are $p_1 = LMS(B^{j,1}_{k-1}, m)$, $p_2 = LMS(B^{j,2}_{k-1}, m)$ and

$$\begin{aligned}
\varepsilon_{3,1} &= ELMS(p_1, B^{j,1}_{k-1} \bigcup \{\hat{q}^i_k\}),\\
\varepsilon_{3,2} &= ELMS(p_2, B^{j,2}_{k-1} \bigcup \{\hat{q}^i_k\}),\\
\varepsilon_3 &= \xi(\varepsilon_{3,1} + \varepsilon_{3,2})
\end{aligned}$$

where $\xi > 0$ is one of the algorithm parameters and ε_3 is considered as the approximation error due to the use of p_1 and p_2 as environment modeling polynomials.

At this point:

- if $\min(\varepsilon_1, \varepsilon_2, \varepsilon_3) = \varepsilon_1$ then the polynomial p^j_{k-1} is the best possible approximation for the points in $B^j_{k-1} \bigcup \{\hat{q}^i_k\}$ and

$$B^j_{k-1} = B^j_{k-1} \bigcup \{\hat{q}^i_k\}; \quad \mathcal{B}_k = \mathcal{B}_{k-1}; \quad \mathbb{E}_k = \mathbb{E}_{k-1}$$

- otherwise if $\min(\varepsilon_1, \varepsilon_2, \varepsilon_3) = \varepsilon_2$ then the best possible approximation is $p^{j,1}_{k-1}$ and

$$\begin{aligned}
B^j_{k-1} &= B^j_{k-1} \bigcup \{\hat{q}^i_k\};\\
\mathcal{B}_k &= \mathcal{B}_{k-1};\\
\mathbb{E}_k &= \{\mathbb{E}_{k-1} \setminus \{p^j_{k-1}\}\} \bigcup \{p^{j,1}_{k-1}\}
\end{aligned}$$

- else if $\min(\varepsilon_1, \varepsilon_2, \varepsilon_3) = \varepsilon_3$, then the best possible choice is to divide the cluster B^j_{k-1} and to use the two obtained polynomials p_1, p_2 instead of p^j_{k-1}:

$$\begin{aligned}
\mathcal{B}_k &= \{\mathcal{B}_{k-1} \setminus B^j_{k-1}\} \bigcup \{B^{j,1}_{k-1}, B^{j,2}_{k-1}\};\\
\mathbb{E}_k &= \{\mathbb{E}_{k-1} \setminus \{p^j_{k-1}\}\} \bigcup \{p_1, p_2\}
\end{aligned}$$

The ξ parameter is related to the number of cluster that will be found by 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. During this process, given each possible couple of close clusters (two cluster are defined close if it exists at least a couple of points, one from the first cluster and the second one from the second cluster, which are neighbors in a radius R) B^i_k, B^j_k and the related polynomials p^i_k, p^j_k and approximation errors $\varepsilon_i, \varepsilon_j$, the polynomial $p = LMS(B^i_k \bigcup B^j_k, m)$ is computed along with the related approximation error $\varepsilon = ELMS(p, B^i_k \bigcup B^j_k)$. At this point given the algorithm parameter $\rho > 0$, if $\varepsilon < \rho(\varepsilon_1 + \varepsilon_2)$ then the partition is modified as

$$\begin{aligned}
\tilde{\mathcal{B}}_k &= \mathcal{B}_k \setminus \{B^i_k, B^j_k\} \bigcup \{B^i_k \bigcup B^j_k\}, \mathcal{B}_k = \tilde{\mathcal{B}}_k;\\
\tilde{\mathbb{E}}_k &= \mathbb{E}_k \setminus \{p^i_k, p^j_k\} \bigcup \{p\}, \mathbb{E}_k = \tilde{\mathbb{E}}_k
\end{aligned}$$

The bigger the value of ρ is, the bigger will be the effect of the unification process over the landmark extraction process.

In the rest of the section, the entire landmark extraction and data association process will be denoted as

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

1.7.1.2 State estimation, State and Landmark Update

To solve the SLAM problem the Extended Kalman filter has to be adapted in order to provide an estimation of both the robot state, modeled by Equations 1.1, and the environment modeling polynomials. To this end, the following augmented state is defined

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

which contains the robot state and all the $m+1$ coefficients $\{b_k^{i,j}\}_{i=0}^m$ of the j-th m-th order polynomial $p_j, j = 1, \ldots, n_k$. In the following, we will use N_k to indicate the dimension of X_k, the compact notation $X_k = [x_k^T,\ p_k^1, \ldots, p_k^{n_k}]^T$ to indicate the augmented state where $p_k^i = [b_k^{m,i}, \ldots, b_k^{0,i}]$.

The landmarks positions are assumed to be constant whatever is the control input and they are assumed to be not influenced by any process noise, therefore the state update function, the process noise covariance matrix and the dynamic matrix related to the augmented state will be

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ p_{k+1}^1 \\ \vdots \\ p_{k+1}^{n_{k+1}} \end{bmatrix} = \mathcal{F}\left(\begin{bmatrix} x_k \\ p_k^1 \\ \vdots \\ p_k^{n_k} \end{bmatrix}, u_k \right) = \begin{bmatrix} f(x_k, u_k) + w_k \\ p_k^1 \\ \vdots \\ p_k^{n_k} \end{bmatrix};$$
$$\mathcal{W} = \begin{bmatrix} W & \emptyset \\ \emptyset & \emptyset_{N_k-3} \end{bmatrix}; \mathcal{A}_k = \begin{bmatrix} A_k & \emptyset \\ \emptyset & I_{N_k-3} \end{bmatrix}$$

where I_{N_k-3} and $\emptyset_{N_k-3}$ are the identity matrix of order $N_k - 3$ and the null matrix with $(N_k - 3) \times (N_k - 3)$ elements respectively. W is the state noise covariance matrix and it is assumed known.

The output equation 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:

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

As well as for the state noise, the output noise covariance matrix, denoted by V, is assumed known. The output matrix will be

$$\mathcal{C}_k = \begin{bmatrix} C_{1,k} & \partial h_1 p_k^1 & \ldots & \partial h_1 p_k^{n_k^p} \\ \vdots & & & \vdots \\ C_{n_S,k} & \partial h_{n_S} p_k^1 & \ldots & \partial h_{n_S} p_k^{n_k} \end{bmatrix}$$

where $C_{i,k}$ is the i-th row of the C_k matrix, h_i is the function $h(X_k)$ related to the i-th sensor and $\partial h_i p_k^i = [\partial h_i b_k^{m,i} \ \dots \ \partial h_i b_k^{0,i}]$.

At this point, also the estimation error covariance matrix P_k has to be adapted to the augmented state. Indicating with $\hat{x}_k$ the estimation of the state x_k and with $\hat{p}_k^i$ the estimation of the polynomial p_k^i, the estimation error covariance matrix related to the augmented state will be

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

where $P(a, b)$ is the estimation error covariance matrix between the a and b.

Obviously, the polynomials $\{p_k^i\}_{i=1}^{n_k}$ are time varying since their coefficients are involved into the state estimation and they are updated by the landmark extraction function at each time step. Moreover also the number, n_k, of polynomials can change during the SLAM process due to the landmark extraction and data association processes and thus the dimensions of the augmented state X_k and of $\mathcal{A}_k, \mathcal{C}_k, \mathcal{P}_k, \mathcal{W}$ are time varying.

After the landmark extraction process, for each change in the modeling polynomials set $\mathbb{E}_k$, there is a change also in $\mathcal{P}_k$ and in X_k. More precisely, if a new polynomial $p_k^r = [b_k^{m,r}, \dots, b_k^{0,r}]$ comes out from the landmark extraction process, then the state becomes $X_k = [X_k^T, \ p_k^r]^T$ and the estimation error covariance matrix is

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

In the same way, if one of the polynomial is deleted by the landmark extraction process, all the related entries in the augmented states and in the matrix $\mathcal{P}_k$ are deleted. As a consequence the dimensions of the augmented state and of the filter matrices are reduced.

Following the above update rules, it is convenient to define the update function

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

which 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, $\mathcal{P}_k = \mathcal{P}_k^*$.

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. Therefore, given $i \neq j$, $P(\hat{p}_k^i, \hat{p}_k^j) = \emptyset$ and $P(\hat{p}_k^i, \hat{x}_k) = \emptyset$, where $\emptyset$ is a null matrix of appropriate size. Each new landmark estimation error covariance matrix is initialized as $P(\hat{p}_k^i, \hat{p}_k^i) = I_{m+1} \times P_{landmark}$ and $P_{landmark} > 0$ is one of the algorithm parameters and it represents the initial landmark estimation error covariance.

Finally, at each time step, using the observation structure (1.6), the landmark extraction function $\mathcal{L}(\cdot)$ and the update function $\mathcal{U}(\cdot)$, the resulting polynomial based SLAM algorithm can be summarized as:

Polynomial based Extended Kalman Filter

$$\begin{aligned}
&\hat{X}_{k+1|k} = \mathcal{F}(\hat{X}_{k|k}, u_k)\\
&\mathcal{P}_{k+1|k} = \mathcal{A}_k \mathcal{P}_{k|k} \mathcal{A}_k^T + \mathcal{W}\\
\\
&(\mathcal{B}_{k+1}, \mathbb{E}_{k+1}) = \mathcal{L}(\mathcal{B}_k, \mathbb{E}_k, y_{k+1}, \hat{X}_{k+1|k})\\
&(\hat{X}^*_{k+1|k}, \mathcal{P}^*_{k+1|k}) = \mathcal{U}(\hat{X}_{k+1|k}, \mathcal{P}_{k+1|k}, \mathbb{E}_k, \mathbb{E}_{k+1})\\
&\hat{X}_{k+1|k} = \hat{X}^*_{k+1|k}\\
&\mathcal{P}_{k+1|k} = \mathcal{P}^*_{k+1|k}\\
\\
&K_{k+1} = \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T (\mathcal{C}_{k+1} \mathcal{P}_{k+1|k} \mathcal{C}_{k+1}^T + V)^{-1}\\
&\hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_{k+1}(y_{k+1} - h(\hat{X}_{k+1|k}))\\
&\mathcal{P}_{k+1|k+1} = \mathcal{P}_{k+1|k} - K_{k+1} \mathcal{C}_{k+1} \mathcal{P}_{k+1|k}
\end{aligned}$$

where $\mathbb{E}_0 = \mathcal{B}_0 = \{\emptyset\}$, $\hat{X}_{0|0} = [\hat{x}_{0|0}]$ and $\mathcal{P}_{0|0} = P_{0|0}$ are the filter initial conditions.

1.7.1.3 Numerical and Experimental Results

To evaluate the performance of the proposed SLAM algorithm, numerical simulations and experimental tests have been performed both using the robot Khepera III model data. The following parameters have been used:

- $d = 0.09m$, $r = 0.0205m$ for the robot Khepera III;
- Sampling period $T = 1s$;
- $V = 0.05^2 I_5$ and $W = diag\{0.02^2, 0.02^2, 0.0003\}$: a standard deviation of $0.05m$ for the ultrasonic sensors measurements, a standard deviation of $0.02m$ on w_k^1, w_k^2 and a standard deviation of 1°on w_k^θ;
- $P_{0|0} = diag\{0.0025, 0.0025, 0.0305\}$: a standard deviation of $0.05m$ on robot initial position estimation error and a standard deviation of 10°on robot initial heading estimation error; $P_{landmark} = 10^{-8}$ for the initial landmark estimation error covariance;

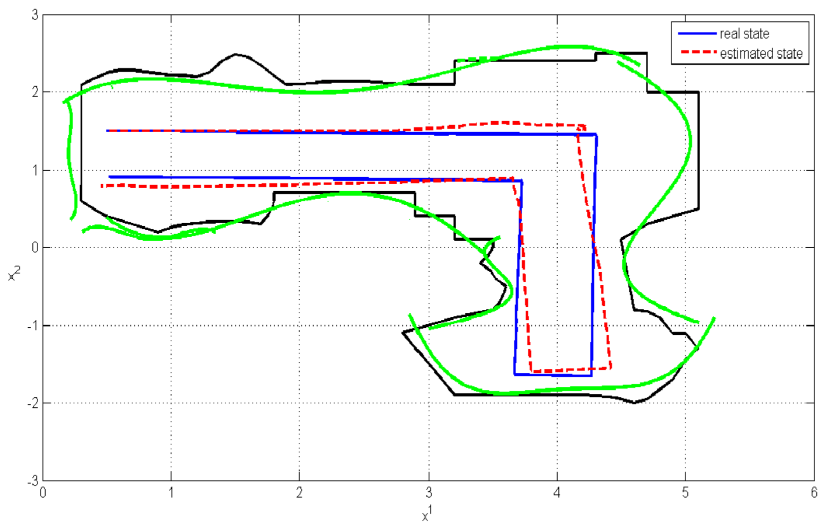


Fig. 1.16. PbSLAM simulation result: real environment in black, estimated environment in green, real trajectory in blue, estimated trajectory in dashed red

- $R = 0.1m$, $\xi = \frac{1}{3}$, $\varepsilon_{TH} = 0.05$, $\rho = \frac{1}{2}$, $m = 4$, $K_s = 10$ for the landmark extraction and data association algorithm.
- The initialization of the set $\mathcal{M}$ have been formed during 20 initial steps at the beginning of each experiment and simulation. During these steps measurements and points of the environment are acquired, and these points will form the initial condition for $\mathcal{M}$.

The NEES index (see Eq. (1.29)) has been used as localization performance algorithm.

The algorithm has been tested in the environment shown in Figure 1.16; a set of 100 simulations have been performed and the obtained averaged NEES index over the 100 simulations is $9.24 \cdot 10^{-7}$, while the RMS index is As shown in the numerical simulations and in the experimental tests, the proposed PEKF algorithm allows to obtain an accurate robot pose estimation and to simultaneously build a good environment boundaries mapping.

1.7.1.4 Remarks

In this section, a new environment model for SLAM has been developed. A set of polynomials has been as SLAM landmarks to approximate the real environment structure and an Extended Kalman filter is employed to localize a mobile robot and properly modify the landmarks structure. The proposed algorithm has been evaluated in both numerical and experimental tests providing always very encouraging estimation and mapping results.

However, the algorithm has two main limitations

- the correlation between robot state and landmarks or among new and old landmarks has not been studied yet;
- the computational cost is very high to use the algorithm for real applications.

Thus, in the next section, another SLAM algorithm will be described, keeping the same robot and environment models, but introducing some modifications to improve the computation time.

1.7.2 Heuristics for improving the computation performance of a polynomials based SLAM algorithm

The aim of the present section is to start from the same modeling approach shown in 1.7.1 and to develop a new SLAM algorithm trying to improve the computation time performance required by the PbSLAM without significantly affecting its mapping performance. Once again, an Extended Kalman Filter is used to perform the estimation process and the environment boundaries are mapped as a set of polynomials.

In particular, two landmarks types are used: shape landmarks and innovation landmarks. Given a m-th order polynomial $p^q(\xi) = \sum_{i=1}^{m} b_i^q \xi^i + c_0^q$, it is represented by a coefficient c_0^q (related to the polynomial position) and by a set of coefficients $\{b_m^q,\ b_{m-1}^q,\ \cdots,\ b_1^q\}$ related to the polynomial shape. The modeling polynomials coefficients are used as shape landmarks and, as for the algorithm presented in Section 1.7.1, they are extracted by the landmark extraction and data association steps. The innovation landmarks are the polynomials position coefficients and they are inserted into the SLAM augmented state and used into the Kalman filter steps. More precisely, an augmented state $X_k = [x_k^T,\ c_{0,k}^1, \ldots, c_{0,k}^{n_k}]^T$ is defined and estimated through the EKF. The main idea behind this choice is that, using distance measurements, the most logic interpretation of the Kalman Filter innovation term regarding the mapping polynomials is that, depending on this term, each polynomial has to be moved towards or away from the robot center, maintaining its shape unchanged. In other words, the landmark extraction process will be used to obtain each polynomial shape while the Kalman Filter will be used to properly move this shape towards or away from the robot.

1.7.2.1 Landmark extraction and Data association

The main idea behind the proposed landmark extraction and data association algorithm is to develop a fast algorithm to obtain an accurate environment approximation by storing a low amount of data and by requiring a low computational cost to elaborate them. The proposed SLAM algorithm uses a set of m-th order polynomials $p^i, i = 1, ..., n$ to approximate the environment boundaries.

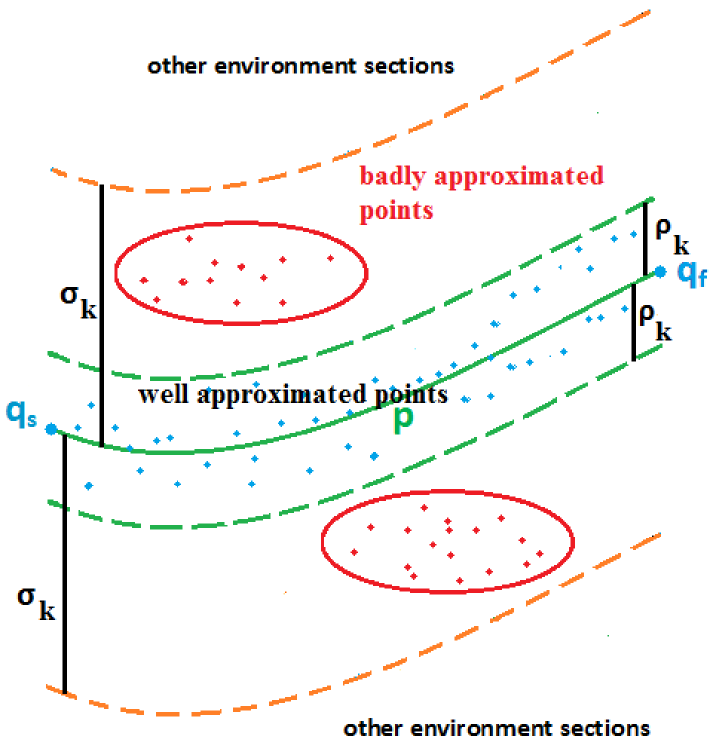

other environment sections
badly approximated points
ρ_k
q_f
ρ_k
σ_k
well approximated points
q_s
p
σ_k
other environment sections

Fig. 1.17. An environment boundaries section

Given a polynomial p^i and a point q:

- $x_c^1(q, p^i)$ returns the x^1 coordinate of q if p^i is a x^1-variate polynomial, otherwise it returns the x^2 coordinate of q. Following the same lines, $x_c^2(q, p^i)$ returns the x^2 coordinate of q if p^i is a x^1-variate polynomial, else it returns the x^1 coordinate of q;
- $\mathcal{Q}_{bad}(p^i)$ contains the points which should be approximated by p^i but their related approximation error is too high.
- q_s^i,q_e^i are the starting point and ending point of the environment boundaries section approximated by p^i.

The landmark extraction and data association processes start from an approximation of the current intersection points between the sensors' axes and the environment boundaries, $\hat{q}_k^i, i = 1, \ldots, n_S$, obtained by using the Eq. (1.42)

where $y_{i,k}$ is the measurement provided by sensor S_i at step k. These points can be seen as an approximation of the points $\tilde{q}_i$ (see Fig. 1.15) due to the estimation and measurements errors.

For each point $\hat{q}_k^i$, the data association process is performed. The first step is to define the set

$$\mathbb{E}_k^* = \{p_k^j \in \mathbb{E}_k : x_c^1(q_s^j, p_k^j) - R \leq x_c^1(\hat{q}_k^i, p_k^j) \leq x_c^1(q_e^j, p_k^j) + R\}$$

containing the candidate polynomials to approximate $\hat{q}_k^i$. R is an approximation tolerance level and it is one of the algorithm parameters. The best, in a LMS sense, approximating polynomial in $\mathbb{E}_k^*$ for the point $\hat{q}_k^i$ is then computed:

$$p_k^* = \arg \min_{p_k^j \in \mathbb{E}_k^*} (x_c^2(\hat{q}_k^i, p_k^j) - p_k^j(x_c^1(\hat{q}_k^i, p_k^j)))^2. \tag{1.43}$$

A first possible situation occurs when $\mathbb{E}_k^* = \{\emptyset\}$; in this case the point $\hat{q}_k^i$ is considered to be not approximated by the polynomials in $\mathbb{E}_k$. A set of points clusters $\mathcal{B}_{k-1} = \{B_j, j = 1, \ldots, n_b\}$ is used to face this situation: the point $\hat{q}_k^i$ is inserted into the cluster $B_j \in \mathcal{B}_{k-1}$ containing the highest number of neighbors, in a radius R, to $\hat{q}_k^i$. If there is no cluster satisfying this neighborhood condition, a new cluster is created containing only the point $\hat{q}_k^i$:

$$\tilde{\mathcal{B}}_{k-1} = \mathcal{B}_{k-1} \bigcup \{\hat{q}_k^i\};\ \mathcal{B}_{k-1} = \tilde{\mathcal{B}}_{k-1}.$$

If $\mathbb{E}_k^* \neq \{\emptyset\}$, the optimization problem (1.43) can be solved and three cases may occur. The first one is related to

$$|x_c^2(\hat{q}_k^i, p_k^*) - p_k^*(x_c^1(\hat{q}_k^i, p_k^*))| \leq \rho_k; \tag{1.44}$$

where ρ_k is an approximation quality threshold. In this first situation, p_k^* is good to approximate $\hat{q}_k^i$ and no further actions are performed. ρ_k is computed as

$$\rho_k = \underline{\rho} + (\overline{\rho} - \underline{\rho}) \frac{||\Sigma|| - ||\Sigma - P_{k|k-1}||}{||\Sigma||}$$

and $\underline{\rho}$, $\overline{\rho}$ are the threshold minimum and maximum values respectively. The main idea behind the above equation is to use a threshold linearly varying w.r.t. the prediction error covariance matrix $P_{k|k-1}$ in a range $\emptyset \leq P_{k|k-1} \leq \Sigma$ where Σ is one of the algorithm parameters. If $P_{k|k-1} > \Sigma$, then the prediction error covariance matrix is too high and the filter is considered in divergence state. The second case occurs when

$$\rho_k < |x_c^2(\hat{q}_k^i, p_k^*) - p_k^*(x_c^1(\hat{q}_k^i, p_k^*))| \leq \sigma_k, \tag{1.45}$$

where, following the same lines of ρ_k, the threshold σ_k is

$$\sigma_k = \underline{\sigma} + (\overline{\sigma} - \underline{\sigma}) \frac{||\Sigma|| - ||\Sigma - P_{k|k-1}||}{||\Sigma||}$$

and $\overline{\sigma}_k > \overline{\rho}_k$, $\underline{\sigma}_k > \underline{\rho}_k$, $(\overline{\sigma} - \underline{\sigma}) > (\overline{\rho} - \underline{\rho})$ so that $\sigma_k > \rho_k$.

In this situation, the approximation error is assumed to be only due to the low accuracy of the polynomial coefficients. The point $\hat{q}_k^i$ is then stored in $\mathcal{Q}_{bad}(p_k^*)$ and if the number of points in the above set becomes greater than a fixed threshold ε_M, then p_k^* is modified to be better adapted to the badly approximated points in $\mathcal{Q}_{bad}(p_k^*)$. Let $\mathcal{R}$ be a set of ε_M equally spaced points on p_k^*. A new m-th order polynomial p is computed by minimizing the weighted least mean square error due to the use of the polynomial p in approximating the points contained in $\mathcal{Q}_{bad}(p_k^*) \bigcup \mathcal{R}$. More precisely, the badly approximated points are weighted by $1/||P_{\bar{k}|\bar{k}-1}||$ where $P_{\bar{k}|\bar{k}-1}$ is the covariance matrix related to the predicted pose $(\hat{x}|_{\bar{k}|\bar{k}-1})$ used to compute the point $\hat{q}_{\bar{k}} \in \mathcal{Q}_{bad}(p_k^*)$ at step $\bar{k}$. The points in $\mathcal{R}$ set are weighted using the inverse covariance related to the innovation landmark relative to the polynomial to be updated. The resulting new polynomial p will substitute the polynomial p_k^* into the map:

$$\tilde{\mathbb{E}}_{k-1} = \{\mathbb{E}_{k-1} \bigcup \{p\}\} \setminus \{p_k^*\};\ \mathbb{E}_{k-1} = \tilde{\mathbb{E}}_{k-1}$$

and the related set of badly approximated points will be an empty set, $\mathcal{Q}_{bad}(p) = \emptyset$.

The third possible case occurs when the approximation error related to $\hat{q}_k^i$ using p_k^* is too high, i.e.

$$|x_c^2(\hat{q}_k^i, p_k^*) - p_k^*(x_c^1(\hat{q}_k^i, p_k^*))| > \sigma_k. \tag{1.46}$$

In this case, any of the polynomials in $\mathbb{E}_k$ is suitable to map $\hat{q}_k^i$. The point $\hat{q}_k^i$ is then inserted into one of the clusters in $\mathcal{B}_{k-1}$, following the same lines used in the case $\mathbb{E}_k^* = \{\emptyset\}$.

After the insertion of a point q into a cluster B_j, if the number of points in the cluster becomes greater than a fixed threshold ε_z, then the cluster is removed from $\mathcal{B}_{k-1}$

$$\tilde{\mathcal{B}}_{k-1} = \mathcal{B}_{k-1} \setminus B_j;\ \mathcal{B}_{k-1} = \tilde{\mathcal{B}}_{k-1},$$

and a new m-th order polynomial p is computed minimizing the weighted least mean square error over the points in B_j. All the points in B_j are wighted following the same lines of the badly approximated points. The obtained polynomial is then inserted into the landmarks set

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

In conclusion, after all the currently acquired points $\{\hat{q}_k^i, i = 1, \ldots, n_S\}$ have been used, the obtained landmarks set $\mathbb{E}_{k-1}$ and the clusters set $\mathcal{B}_{k-1}$ become the updated environment map, thus $\mathbb{E}_k = \mathbb{E}_{k-1}$ and $\mathcal{B}_k = \mathcal{B}_{k-1}$.

1.7.2.2 Polynomials merging

When a new polynomial, say it p, is extracted, if it partially or totally includes an environment part that has been already mapped by another polynomial $p_2 \in \mathbb{E}_{k-1}$, then a third polynomial is computed by merging p and p_2 to reduce the amount of data used to map the environment.

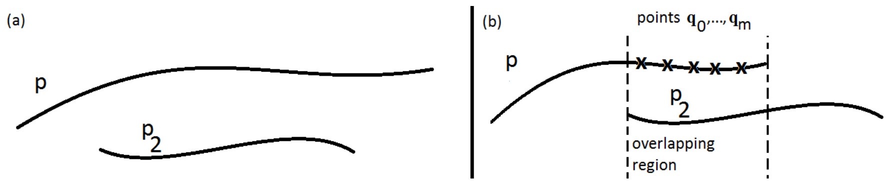


Fig. 1.18. Landmarks merging situations: (a) p_2 included into p; (b) polynomials have an overlapping region

Two cases may occur: in the first case, the extrema of p are included or completely include the extrema of p_2 (see Fig. 1.18(a)); in the second case, the two polynomials p and p_2 overlap for a sufficiently big region(see Fig. 1.18(b)). In both the cases, let $q_0, q_1, \ldots, q_m$ be $m+1$ equally spaced points placed on p in the overlapping region between p and p_2 (see Fig. 1.18(b)) and let d be the mean LMS approximation error due to the use of p_2 in mapping the q_i points; if d is lower than a threshold d_M, then p and p_2 are merged by creating a new LMS approximating polynomial starting from ε_M points equally spaced on p and ε_M points equally spaced on p_2. Also in this case a weighted LMS approximation is used and the points on p and p_2 are weighted using the inverse covariances related to their position coefficients (as in the case a polynomial has to be updated using its badly approximated points).

1.7.2.3 State estimation, State and Landmark Update

As shown in [41], when a Kalman filter is used to solve the SLAM problem, the dimensions of the augmented state X_k and of the matrices involved into the filter are time varying since the number of landmarks used to map the environment can change during the SLAM process. After the landmark extraction and data association processes, for each change in the modeling landmarks, there is a change also in the estimation error covariance matrix $\mathcal{P}_k$ and in the augmented state X_k. More precisely, for each landmark added/removed, the related position coefficient is added/removed from X_k. About the $\mathcal{P}_k$ matrix, if a landmark is removed from the map, the related rows and columns are removed from the covariance matrix; if a new landmark is inserted into the map, then, as shown by [43], the matrix becomes

$$\tilde{\mathcal{P}}_k = G_x \mathcal{P}_k G_x^T + G_y V G_y^T \ , \ \mathcal{P}_k = \tilde{\mathcal{P}}_k. \tag{1.47}$$

where G_x and G_y matrices are computed as

$$G_x = \partial g(x,y)x|_{\substack{x = \hat{x}_{k|k-1} \\ y = y_k}} \ , G_y = \partial g(x,y)y|_{\substack{x = \hat{x}_{k|k-1} \\ y = y_k}} \tag{1.48}$$

and $g(x,y)$ is the landmark generation function used to obtain the new landmark: the weighted least mean square function in the proposed SLAM method.

Using the above update rules the augmented state and the estimation error covariance matrix are properly modified according to the variations in the environment mapping.

1.7.2.4 Algorithm complexity analysis

The proposed SLAM algorithm uses a set of heuristics to improve the performances of the PbSLAM algorithm proposed in Section 1.7.1, without significantly affecting the mapping results. More precisely, the above proposed strategy is based on acquiring the detected environments boundary points (computed by Eq. (1.42)) and then clustering them, at each step k, so that the obtained clusters can be well approximated by m-th order polynomials. The cost of this procedure increases with the number of acquired points and then it grows as the simulation/experiments goes on. Instead of directly using measured points, the here proposed algorithm is based on the use of their approximating polynomial. Say $\mathcal{G}(p)$ the set of points used to generate a polynomial p. We assume that if p is chosen to map $\mathcal{G}(p)$, then it represents a good approximation for them and thus the points contained in $\mathcal{G}(p)$ may be discarded once the polynomial p is computed. When the polynomial has to be updated (i.e. $|Q_{bad}(p)| > \varepsilon_M$), the PbSLAM clustering procedure is bypassed by directly checking the mapping polynomials approximation performance (Equations (1.44), (1.45), (1.46)) and some points on each polynomial $p \in \mathbb{E}_k$ are used instead of the points in $\mathcal{G}(p)$, assuming that p is a good approximation for them.

As a consequence, while the PbSLAM algorithm complexity grows with the number of acquired measurements, the new proposed algorithm complexity grows with the number of mapping polynomials which is considerably smaller than the number of acquired measurements and it does not continuously increase as the simulation/experiment goes on. Moreover, in the PbSLAM all the acquired measured points have to be kept in memory to allow the clustering procedure; on the contrary the here proposed algorithm requires to keep in memory only the currently badly approximated points and the currently not yet approximated points (contained into $\mathcal{B}_k$); when the above points set cardinalities increase, these points are used to modify the map and then they are discarded.

Obviously, the obtained map, using the strategy proposed in this paper, is expected to have worse mapping performance w.r.t. the map obtained by PbSLAM due to the used heuristics instead of the clustering procedure but the computation times required by the first technique are expected to be very lower than the ones required by the second technique.

1.7.2.5 Numerical Results

To evaluate the performance of the proposed SLAM algorithm, numerical simulations have been performed using the a differential drive model equipped with $n_S = 5$ distance sensors. The model input variables are the wheels angular velocities ω_k^l and ω_k^r, and they have been precomputed so that the robot follows the desired trajectories.

In the performed test, the following parameters have been used:

- $L = 0.09m$, $r = 0.0205m$ for the robot;
- Sampling period $T = 1s$;
- $V = 0.05^2 I_5$ and $W = diag\{0.001^2, 0.001^2, 0.0055^2\}$;
- $\Sigma = diag\{0.5^2, 0.5^2, 0.52^2\}$;
- $\underline{\rho} = 0.1m, \overline{\rho} = 0.2m, \underline{\sigma} = 0.2m, \overline{\sigma} = 0.4m$, $R = 0.05m$, $\varepsilon_M = 5$, $\varepsilon_z = 10$, $\underline{m} = 3$, $d_M = 0.15m$ for the SLAM algorithm.

Please note that the algorithm parameters have been chosen validating the algorithm performance by simulation. Further studies are in progress to develop some heuristics usable to choose such parameters values. The proposed polynomial based SLAM algorithm, in the following denoted as $EPbSLAM$ (Efficient Polynomial based SLAM), has been contrasted with the polynomial based SLAM algorithm described in Section 1.7.1, namely $PbSLAM$, and with the EKF SLAM described in [41], denoted as $ESLAM$.

Once again the NEES (see Eq. (1.29)) index has been used as localization performance evaluation index.

The algorithm has been tested with different trajectories and differents shapes of environment and obstacles, in order to evaluate its performance in various situations.

A set of 100 experiments have been performed for each different scenario. As described in [18] the EKF may be affected by linearization problems if the initial condition is not zero. To face this problem, following the same heuristic proposed in the above cited paper, the reference frame fixed in the robot initial condition has been used as reference frame to perform the localization and mapping task; then the initial SLAM condition is $\hat{x}_{0|0} = [0\ 0\ 0]^T$ with initial covariance matrix $P_{0|0} =_{3\times3}$ The results, in terms of the averaged proposed index and of the averaged computation time[1], required by each algorithm, are shown in Tables 1.2,1.3. Typical mapping and localization results are shown in Figures 1.19,1.20, 1.16, 1.21.

Table 1.2. Trajectory 1:results over the 100 performed simulations

	μ	RMS index	mean computation time
$PbSLAM$	$9.2407 \cdot 10^-7$	0.054233	6.2812
$EPbSLAM$	$2.0043 \cdot 10^{-6}$	0.0094012	0.092746
$ESLAM$	$1.6049 \cdot 10^{-4}$	1.1687	0.0049281

[1] The algorithms computation times have been computed using Matlab R2012 running on an Intel(R) Core(TM) i7 Q720 CPU.

Table 1.3. Trajectory 2:results over the 100 performed simulations

	μ	RMS	mean computation time
PbSLAM	$1.3492 \cdot 10^{-}6$	0.06852	12.0959
EPbSLAM	$4.2509 \cdot 10^{-}6$	0.013571	0.1078
ESLAM	$1.361 \cdot 10^{-}4$	0.50721	0.011478

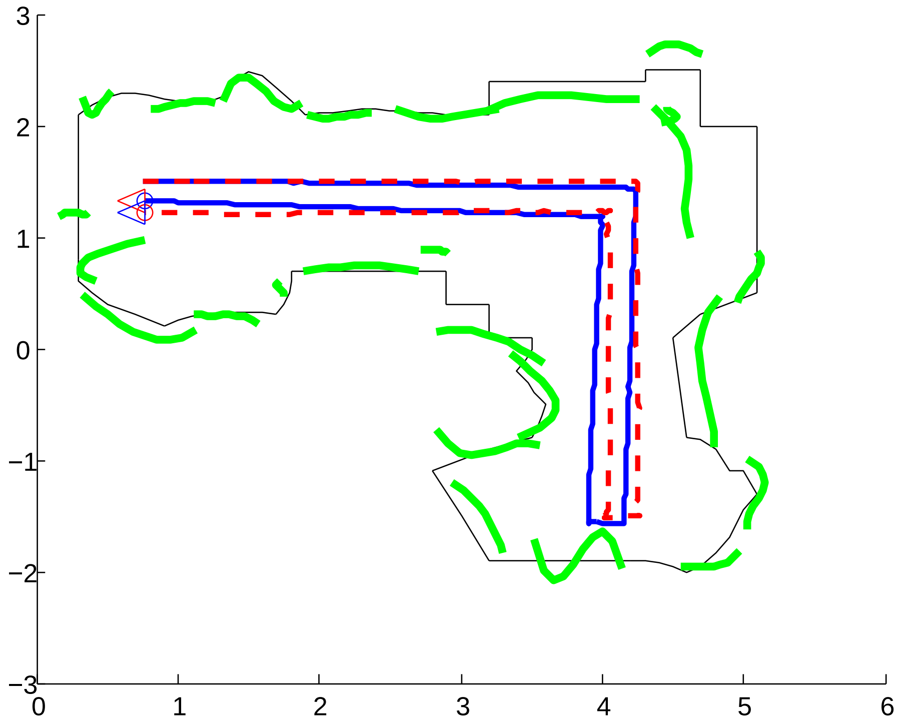


Fig. 1.19. Proposed SLAM simulation result: real environment in black, estimated environment in green, real trajectory in blue, estimated trajectory in dashed red

As shown in Tables 1.2,1.3 and in Figures 1.16, 1.19, 1.20, 1.21, the *Pb-SLAM* yield to the best mapping and localization results. However its averaged computation time is too high to use the algorithm in *real time*. The *ESLAM* algorithm is the fastest algorithm and its localization performance are satisfactory; however it provides a map containing very poor information about the environment boundaries shape. Finally, the proposed *EPbSLAM* method yields to good localization and mapping results. The *EPbSLAM* performance is comparable with the *PbSLAM* performance but the here proposed method requires a lower computation time allowing its use during robot motion.

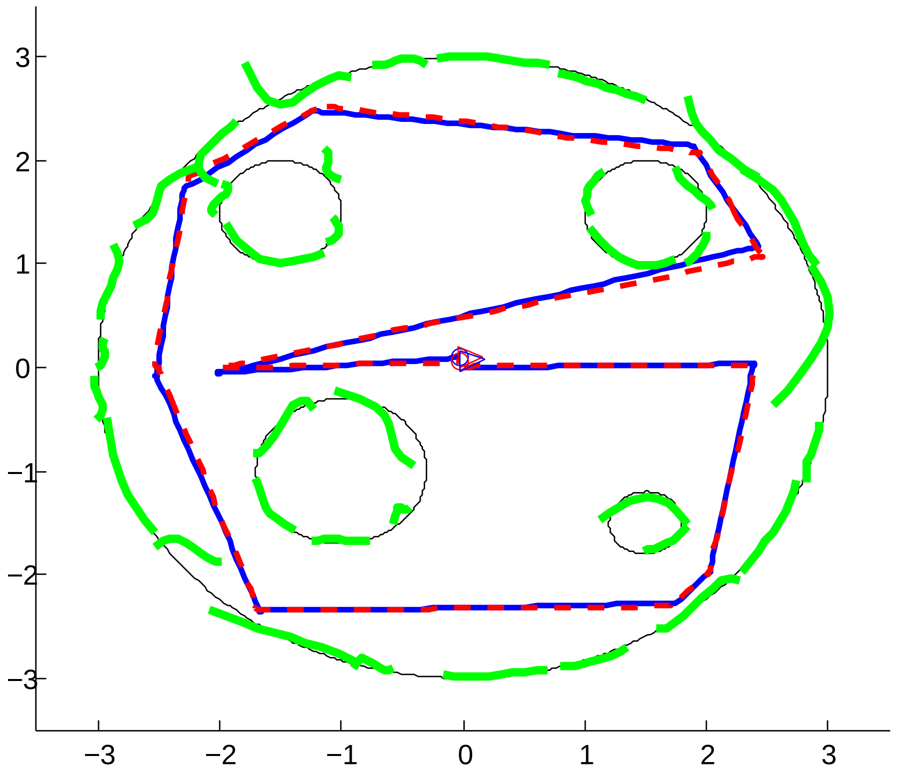


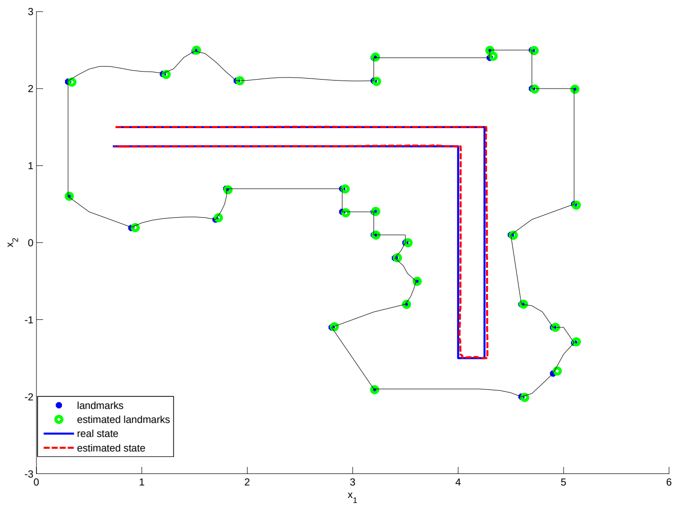
Fig. 1.20. Proposed SLAM simulation result: real environment in black, estimated environment in green, real trajectory in blue, estimated trajectory in dashed red

1.7.2.6 Remarks

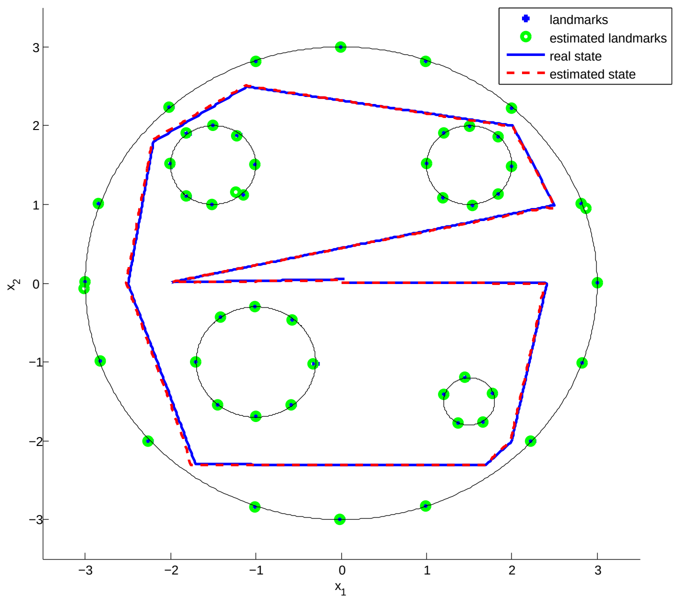
The algorithm presented in section 1.7.1 has a very good localization and mapping performances but has two main limitations: the correlation between robot state and landmarks or among new and old landmarks has not been studied, as well as it has a high computational cost and thus it is not applicable in real time.

In this section, a novel solution to the SLAM problem is proposed, looking for a faster technique. The computational load reduction is accomplished using some heuristics aimed to reduce the big amount of stored data, typical of the algorithm described in Section. Moreover, the interaction between robot state and landmarks is taken into account.

Numerical simulations have been performed showing the effectiveness of the proposed polynomial based SLAM compared with the SLAM technique shown in 1.7.1 and with the traditional SLAM technique described in [41]. The results show that the new algorithm leads to satisfactory localization and



(a) World 1



(b) World 2

Fig. 1.21. ESLAM simulation result

mapping performance while requiring a computation time low enough to use the algorithm in real time during robot motion.

1.7.3 Optimization of the EPbSLAM mapping perfrmance

1.7.3.1 Introduction

The algorithms described in Sections 1.7.1 and 1.7.2 are based on calculating an environment point for each acquired measurement. On the contrary, the idea described in the present Section is based on a different strategy, in order to improve the reliability of the points used to calculate the polinomials approximating the environment. In particular, the idea is to design a SLAM strategy based on the so called *occupancy grids.* Occupancy grid mapping is based on using a grid of equally spaced locations, each one associated to a binary value indicating the probability of finding or not an object in that location. Such occupancy grids can be defined also as *certainty grids*; in [44] it is shown that such approach is suitable to data fusion of measurements acquired by different kind of sensors, reducing the estimation inaccuracy due to measurements uncertainty. In [45] the authors propose a SLAM algorithm for dynamic environments; they use two different occupancy grids, the first one for the static parts and the second for the dynamic parts of the environment. In [46] the authors use particle weight based occupancy grids and polar scan matching to improve the estimation accuracy with respect to simply using odometry.

In this section, a new mapping approach is proposed. No hypothesis are made about the environment structure, so that the procedure can be easily used in different situations. The idea is to create a cell based covering of the environment bounds and to obtain a set of points which are expected to be on these boundaries. To this end, the probability mass function of finding an environment boundary point in a given covering cell is found by properly using the information provided by robot sensors.

The resulting mapping algorithm is directly usable in a Simultaneous Localization and Mapping (SLAM) framework or it can be used in series to an existing given SLAM algorithm to improve its performance.

The paper is organized as follows: in Section II the problem statement is described; in Section III the proposed probabilistic covering mapping algorithm is proposed and in Section IV it is inserted in a SLAM context; in Section V the resulting SLAM algorithms results are shown and in Section VI conclusions are drawn.

1.7.3.2 Problem Statement

The new environment mapping strategy is represented in Figure 1.22. In particular, the environment boundaries are covered by a set of n_k cells $\{C_i\}_{i=1}^{n_k}$, and, for each of them, a probabilistically reliable point, Q_i, is properly computed to represent the information provided by sensors measurements about that cell. Starting from these cells, an environment approximation can be built, for example using lines or polynomials, and used to model the sensors measurements as:

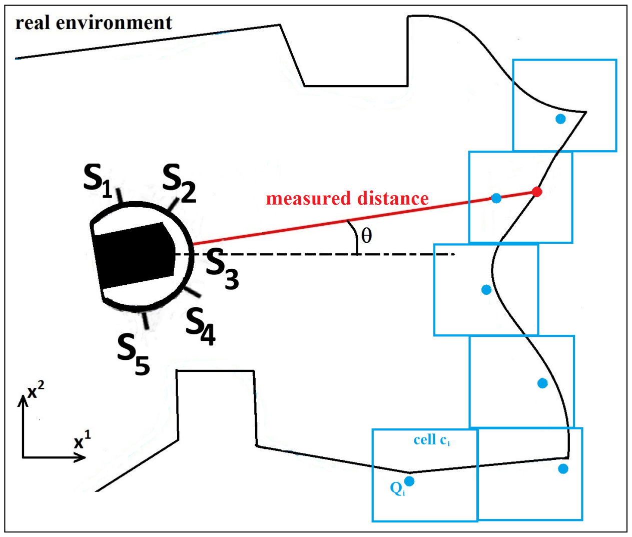


Fig. 1.22. Real environment (black line), occupancy cells (light blue line), robot on board sensors position

$$\hat{y}_k = h(x_k, \mathbb{E}(\{C_i\}_{i=1}^{n_k})) + v_k, \tag{1.49}$$

where the size of vector y_k is n_S, $\{C_i\}_{i=1}^{n_k}$ is the environment boundaries covering using cells at step k, $\mathbb{E}(\{C_i\}_{i=1}^{n_k})$ is the environment approximation and v_k is the measurement noise, assumed to be a zero mean Gaussian noise with covariance V and uncorrelated with w_k.

The goal of the present work is to estimate the state of the robot, $x_k = [x_k^1, x_k^2, \theta_k]^T$, and simultaneously find a good approximation for the environment portions detected by the robot.

1.7.3.3 Spatial Probabilistic Covering

Traditional grid-based mapping techniques build a partition of the robot surrounding environment and assign an *occupancy probability* to each cell into the grid. The environment map is made by the cells with high occupancy probability, and it is usually affected by the resolution problem due to the dimension of each cell forming the grid. On the contrary, feature based mapping techniques aim at obtaining an approximation of the environment boundaries by building an envelope of the environment shape formed by properly parameterized mathematical structures (lines [47, 48], polynomials (Sections 1.7.1,1.7.2), splines [49]). The main advantage of feature based techniques is their better mapping performance w.r.t. grid based ones, at the cost of higher computational burden.

To overcome the issues of both the feature based and the grid based techniques, we propose a new way to build a mapping of the environment. The environment is structured as a time growing set of n_k cells $C_i, i = 1, \ldots, n_k$. Once a new measurement is acquired, we use the related information to update the set of cells. Each cell C_i is represented by:

- its extreme points $x^1_{i,m}$, $x^1_{i,M}$, $x^2_{i,m}$, $x^2_{i,M}$; for the sake of simplicity, each cell is a square of size Γ;
- two probability mass functions [50], related to the x^1 axis values and the x^2 axis values into the cell;
- the most likely point in the cell, say it Q_i.

The probability mass function related to the x^1 axis gives a probability value to each value on this axis from $x^1_{i,m}$ to $x^1_{i,M}$. In the same way, the probability mass function related to the x^2 axis gives a probability value to each value between $x^2_{i,m}$ and $x^2_{i,M}$. These functions are approximated by discretizing the segment from $x^1_{i,m}$ to $x^1_{i,M}$ and the segment from $x^2_{i,m}$ to $x^2_{i,M}$.

Consider now the segment on the x^1 axis (similar considerations apply to the x^2 axis); this segment is divided into N segments of the same size $\Lambda^{i,1}_j, j = 1, \ldots, N$, such that $\Lambda^{i,1}_j$ extreme values are $\xi^1_{j,m} = x^1_{i,m} + (j-1)\gamma$ and $\xi^1_{j,M} = x^1_{i,m} + j\,\gamma$, where $\gamma = \Gamma/N$.

Let $\mathcal{P}(x^1, \Lambda^{i,1}_j)$ be the probability that at least a point on the environment boundaries is contained in the cell C_i and whose x^1 coordinate belongs to $[\xi^1_{j,m}, \xi^1_{j,M}]$. This probability is approximated, at each step k, by counting the number of environment boundary points detected by robot sensors until time k, contained into C_i and whose x_1 coordinate falls within $\Lambda^{i,1}_j$ extreme points. Let this number be $\lambda^{i,1}_j$; the probability $\mathcal{P}(x^1, \Lambda^{i,1}_j)$ is then approximated as

$$\mathcal{P}(x^1, \Lambda^{i,1}_j) = \frac{\lambda^{i,1}_j}{\sum_{j=1}^{N} \lambda^{i,1}_j}.$$

Define $\Lambda^{i,1}_*$ and $\Lambda^{i,2}_*$ by

$$\begin{aligned} \Lambda^{i,1}_* &= \max_{\Lambda^{i,1}_j \subset [x^1_{i,m}, x^1_{i,M}]} \mathcal{P}(x^1, \Lambda^{i,1}_j), \\ \Lambda^{i,2}_* &= \max_{\Lambda^{i,2}_j \subset [x^2_{i,m}, x^2_{i,M}]} \mathcal{P}(x^2, \Lambda^{i,2}_j), \end{aligned} \tag{1.50}$$

in such a way that $\Lambda^{i,1}_*$ and $\Lambda^{i,2}_*$ represent the region, on x^1 and x^2 axis respectively, of the cell C_i with the highest probability of finding an environment boundary point.

Let now Q_i be defined as the point whose coordinates are the middle values in $\Lambda^{i,1}_*$ extreme points and in $\Lambda^{i,2}_*$ extreme points;

$$Q_i = \left[\frac{\xi^1_{*,m} + \xi^1_{*,M}}{2} \;,\; \frac{\xi^2_{*,m} + \xi^2_{*,M}}{2} \right]. \tag{1.51}$$

This point is an approximation of the most likely point in C_i and it is used to represent the entire information provided by all the detected points in the cell.

At each step k, given the estimated robot state $\hat{x}_k$ and a distance measurement $y_{t,k}$ provided by sensor S_t, the related environment detected point is approximated by

$$\begin{aligned} q^1_{t,k} &= \hat{x}^1_k + y_{i,k}\cos(\hat{\theta}_k + \alpha_t), \\ q^2_{t,k} &= \hat{x}^2_k + y_{i,k}\sin(\hat{\theta}_k + \alpha_t), \\ q_{t,k} &= (q^1_{t,k}, q^2_{t,k}), \end{aligned} \tag{1.52}$$

where α_t is the sensor S_t orientation w.r.t. robot axis and it is known by robot structure. Two cases may occur: either the point is not contained in any cell C_i in the actual grid or it exists a cell C_i containing the point $q_{t,k}$. In the first case a new cell is created and centered in $q_{t,k}$. The cell extreme points will be

$$\begin{aligned} x^1_m &= q^1_{t,k} - \mathit{\Gamma}/2,\, x^1_M = q^1_{t,k} + \mathit{\Gamma}/2, \\ x^2_m &= q^2_{t,k} - \mathit{\Gamma}/2,\, x^2_M = q^2_{t,k} + \mathit{\Gamma}/2. \end{aligned} \tag{1.53}$$

The point $q_{t,k}$ is then inserted into this new cell properly updating the cell probability mass functions.

In the second case the point $q_{t,k}$ falls in an existing cell. The point is inserted in this cell and the related probabilities are updated, see Figure 1.23.

To filter outlier measurements and to ensure a large enough spatial probability for each point Q_i in each cell C_i, when a new point $q_{t,k}$ is inserted into a cell C_i, the spatial probabilities $\mathcal{P}(x^2, \mathit{\Lambda}^{i,2}_j)$ and $\mathcal{P}(x^1, \mathit{\Lambda}^{i,1}_j)$ are updated for each $j = 1, \dots, N$ and the point Q_i is updated too. This point is considered reliable enough to be used if and only if the following two conditions are satisfied:

Condition 1: $\sum_{j=1}^{N} \lambda^{i,1}_j \geq N_p$;

Condition 2: $\mathcal{P}(x^1, \mathit{\Lambda}^{i,1}_*) \geq \overline{p}$ && $\mathcal{P}(x^2, \mathit{\Lambda}^{i,2}_*) \geq \overline{p}$.

The first condition ensures that the point Q_i is not related to outlier measurements since at least N_p points have been detected into the same cell C_i. The second condition admits a point Q_i as a representation of all the points in C_i only if its coordinates are probabilistically reliable enough.

In summary, a Spatial Covering Creation algorithm can be defined, and the entire cells creation process can be seen as a single function

$$\mathrm{Q} = SCC(\mathrm{old\_Q}, \hat{x}_k, y_k).$$

providing the list, Q, of points Q_i reliable enough to approximate their related cells C_i after the sensors measurements acquisition y_k.

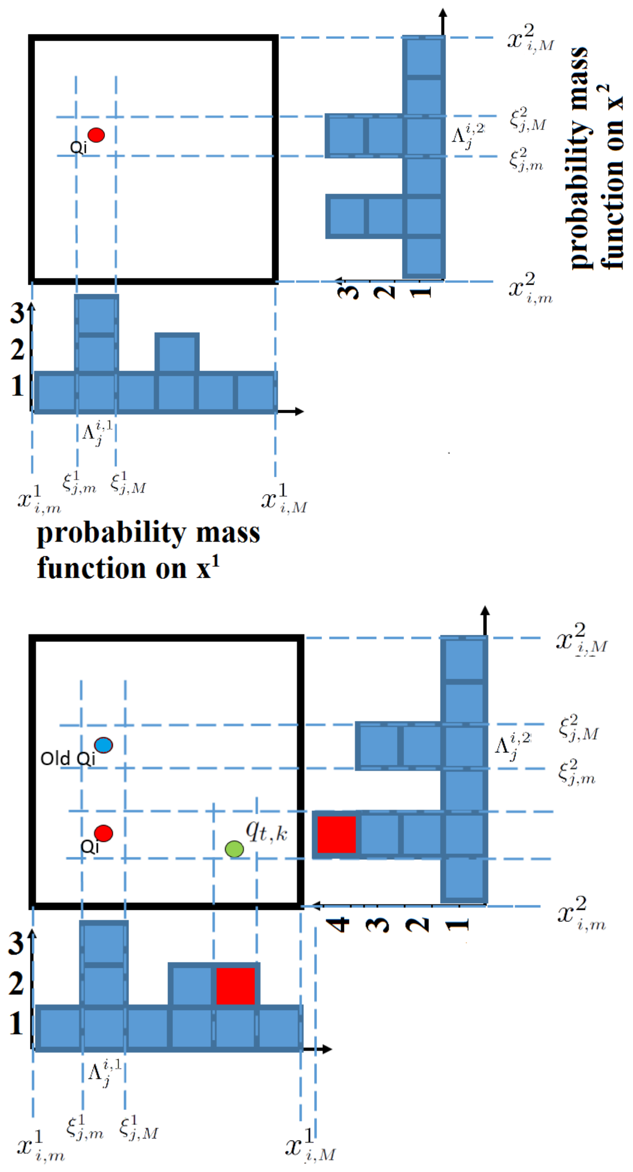


Fig. 1.23. Q_i coordinates update after $q_{t,k}$ detection. The red squares represent the updated counters $\lambda^{i,1}$ and $\lambda^{i,2}$ related to x^1 and x^2 axis respectively

Remark 1.3. **1** Note that if **Condition** 1 or **Condition** 2 are not satisfied, then the information provided by the new acquired point is inserted into the related cell C_i but the point Q_i is not used as an approximation for the cell.

Algorithm 1 Spatial Covering Creation (SCC) Algorithm

```
1: procedure SCC($G_k$,$\hat{x}_k$,$y_k$,$Q$)
2:     Parameters - $G_k$: current grid; $\hat{x}_k$: robot estimated pose; $y_k \in \mathbb{R}^{n_S}$: sensors measurements at step $k$; $Q$:list of points reliable enough to approximate their related cells.

3:     for $t \leftarrow 1, n_S$ do
4:         compute $q_{t,k}$ using (1.42)
5:         $flag \leftarrow FALSE$
6:         for $C_i$ in grid do
7:             if $x^1_{i,m} \leq q^1_{t,k} \leq x^1_{i,M}$ AND $x^2_{i,m} \leq q^2_{t,k} \leq x^2_{i,M}$ then
8:                 $flag \leftarrow TRUE$
9:                 $C \leftarrow C_i$
10:                break
11:            end if
12:        end for
13:        if flag=FALSE then
14:            create new cell $C$ centered in $q_{t,k}$ following (1.53)
15:        end if
16:        find $\Lambda^{i,1}_j$ in $[x^1_{i,m}, x^1_{i,M}]$ s.t. $\xi^1_{j,m} \leq q^1_{t,k} \leq \xi^1_{j,M}$
17:        $\lambda^{i,1}_j = \lambda^{i,1}_j + 1$
18:        find $\Lambda^{i,2}_j$ in $[x^2_{i,m}, x^2_{i,M}]$ s.t. $\xi^2_{j,m} \leq q^2_{t,k} \leq \xi^2_{j,M}$
19:        $\lambda^{i,2}_j = \lambda^{i,2}_j + 1$
20:        compute $\Lambda^{i,1}_*$ and $\Lambda^{i,2}_*$ using (1.50)
21:        update $Q_i$ using (1.51)
22:        if Condition 1 and Condition 2 are satisfied then
23:            consider $Q_i$ as a reliable approximation for $C_i$ and $Q = [Q; Q_i]$
24:        end if
25:        return updated grid and updated list $Q$ of reliable points approximating cells
26:    end for
27: end procedure
```

As a consequence, the list Q contains only information about cells C_i with a probabilistic reliable enough approximating point Q_i.

Remark 1.4. **2** Using two probability mass functions for each cell, instead of the single occupancy probability usually proposed in the literature, the information provided by each cell is expected to be more usable, flexible to the environment structure and able to provide better mapping performance.

Remark 1.5. **3** Note that the optimization problems (1.50) could have multiple solutions. In this case, to chose one of the available solutions, a possible way consists in using information provided by the covariance related to robot estimate $\hat{x}_k$. The solution related to the points acquired with more reliable robot poses is preferred to the other ones.

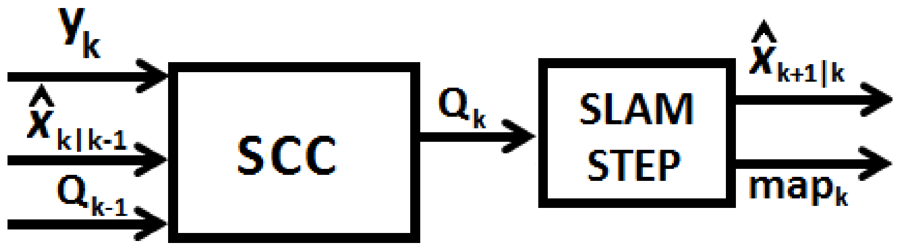


Fig. 1.24. SCC + SLAM

1.7.3.4 SLAM using Spatial Probabilistic Covering

In section 1.7.3.3, the algorithm used to create a cell based covering of the working environment has been presented. The algorithm creates and updates a set of cells and, for each of them, calculates the point with the highest probability to represent a real point of the environment boundaries. However, the result of the SCC process is a set of points which is still far from a detailed representation of the environment boundaries. Thus, the idea to effectively exploit these points consists in using them as input for a SLAM algorithm. In this paper, the algorithm presented in Section 1.7.2 is used.

The whole strategy is represented in Figure 1.24: at each step, the actual measurements and the previously computed robot state prediction are used to update the probability mass functions associated to each cell using SCC algorithm, then a detailed environment mapping is performed using a SLAM algorithm.

In the traditional SLAM algorithms, the environment mapping is updated at each step using all the acquired robot measurements. Such strategy may increase the computational cost more and more as the experiment progresses.

On the contrary, since each cell has a physical defined size and the environment to be reconstructed has limited dimensions, it is possible to obtain a mapping formed by a limited number of cells, regardless of the experiment duration and of the number of acquired measurements.

However, an important issue to be considered is the time taken by the whole procedure. In particular, the time required by the SLAM procedure, at a single step, is expected to be shorter when its input data are computed using the SCC algorithm (scheme shown in Figure 1.24) and the saved time w.r.t. using only the SLAM technique is expected to be longer than the time taken by the SCC procedure itself.

Moreover the points provided by the SCC procedure are expected to be reliable enough to ensure no mapping performance degradation using the SCC+SLAM scheme w.r.t. using only the SLAM algorithm.

1.7.3.5 Numerical Results

To evaluate the performance of the proposed mapping and SLAM strategies, numerical simulations have been performed using a differential drive model (See Section 1.3).

In the performed tests, the following parameters have been used for the robot:

- $l = 0.09$ m, $r = 0.0205$ m for the robot;
- sampling period $T = 1$ s;
- $V = 0.05^2 I_5$ and $W = \text{diag}\{0.01^2, 0.01^2, 0.0017^2\}$;

Four SLAM algorithms have been compared in 100 numerical simulations:

1. a SLAM algorithm based on the scheme shown in Figure 1.24 where the SCC algorithm is used in series with the SLAM polynomial feature based SLAM algorithm proposed in Section 1.7.2. This method will be denoted as covering/feature based SLAM (*CFSLAM*);
2. an Extended Kalman Filter based SLAM method which directly uses the proposed covering strategy for the environment mapping: each point Q_i contained in the list provided by the SCC algorithm is used as a SLAM landmark and to obtain the map, these points are properly connected through segments; the overall algorithm will be denoted as *CSLAM*;
3. the *EPbSLAM*) algorithm described in Section 1.7.2;
4. the SLAM described in [41], denoted as *ESLAM*.

In all the numerical tests, the following SCC algorithm parameters have been chosen: $\Gamma = 0.3m$, $N = 3$, $N_p = 5$, $\overline{p} = 0.1$. These algorithms parameters have been heuristically chosen. Further studies are in progress to develop some policy usable to choose such parameters values.

The algorithms have been tested in the environment shown in Fig. 1.25. The angular velocities have been precomputed so that the robot follows a trajectory starting from $(0.75, 1.5)$ m with orientation $\theta_0 = 0$ rad, and passing by $(4.25, 1.5)$ m, $(4.25, -1.5)$ m, $(4, -1.5)$ m, $(4, 1.25)$ m, $(0.75, 1.25)$ m. The path has been performed in $k_f = 1000$ steps.

As usual, the NEES (see Eq. (1.29)) index has been used as localization performance evaluation index.

All the used four SLAM methods are based on the Extended Kalman Filter (EKF) theory properly adapted to the SLAM context. The results, in terms of the averaged proposed index and of the averaged computation time[2] required by each algorithm, are shown in Table 1.4. Typical mapping and localization results are shown in Figures 1.25, 1.26, 1.27, 1.28.

As shown in Table 1.4 and in Figures 1.25, 1.26, 1.27, 1.28, the *ESLAM* algorithm is the fastest one but it provides the worst mapping and localization performance. The proposed *CSLAM* strategy yield to good computation

[2] The algorithms computation times have been computed using Matlab R2012 running on an Intel© Core™ i7 Q720 processor.

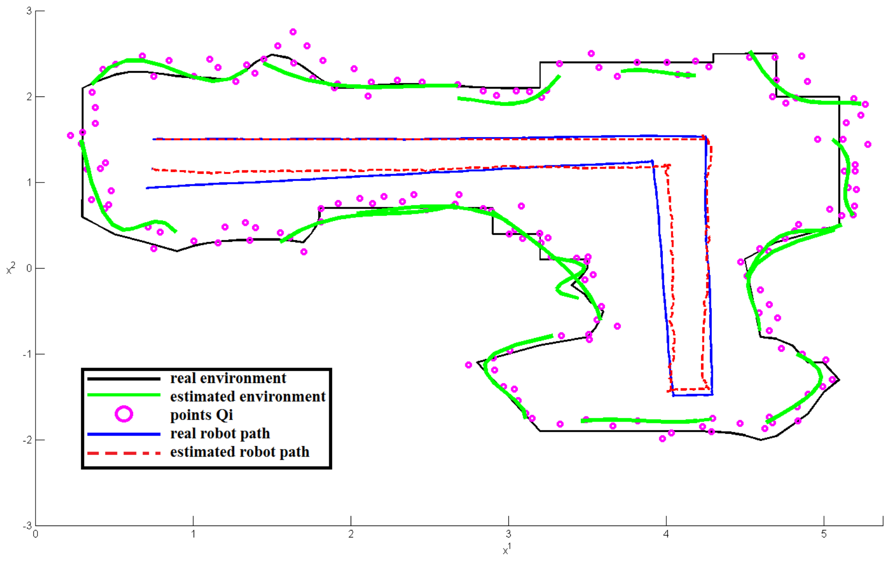


Fig. 1.25. *CFSLAM* simulation result

performance and its mapping and localization results are comparable to the ones provided by the *PbSLAM* method. As expected, the best SLAM results are obtained by using the *CFSLAM* algorithm. Note that all the methods ensure computation times low enough to be used in real application during robot motion.

1.7.3.6 Conclusions

In this work a novel solution to the mapping problem for a mobile robot moving in a an unknown indoor environment is described. The here proposed mapping technique provides a cells based covering of the environment boundaries. The main idea is to estimate the probability mass function of finding an environment boundary point in each covering cell.

Table 1.4. results over the 100 performed simulations

	$\mu \times 10^5$	mean computing time
CFSLAM	0.94	0.1 s
CSLAM	3.6	0.06 s
PbSLAM	2.27	0.47 s
ESLAM	16	0.015 s

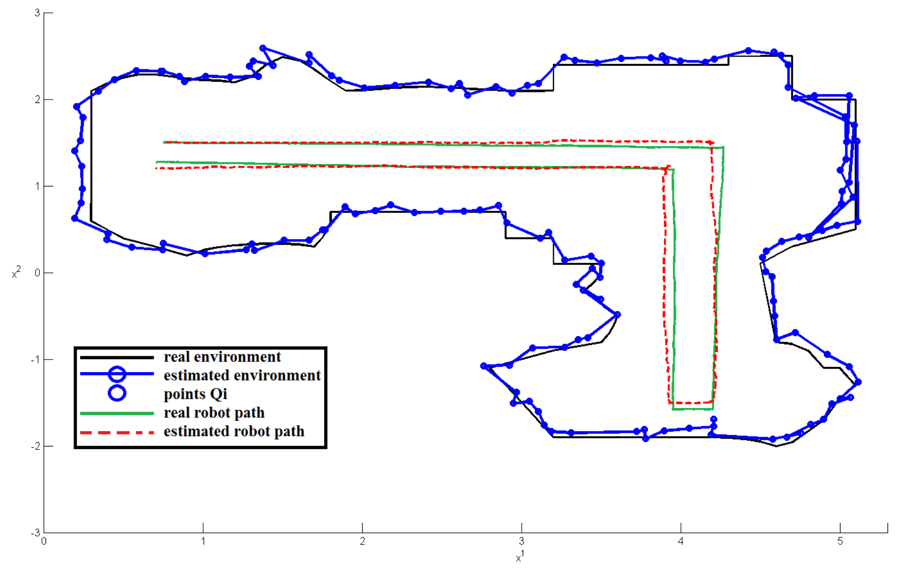


Fig. 1.26. *CSLAM* simulation result

The proposed SCC mapping algorithm can be used alone to map the environment in a SLAM method or in series to an existing given SLAM technique to improve its performance.

In this context, the *CSLAM* algorithm has been developed by using the SCC algorithm for the mapping and the *CFSLAM* algorithm has been formed by using the SCC algorithm along with an existing polynomial feature based SLAM technique(scheme shown in Figure 1.24). These two SLAM solutions have been contrasted, by numerical simulations, with the EKF SLAM technique proposed in [41] and with the PbSLAM method described in Section 1.7.2. The results have shown that the CSLAM algorithm performance are satisfactory and that using the SCC+SLAM scheme (*CFSLAM* method) leads to obtain the best mapping and localization performance requiring a computation time low enough for using the method in real applications.

1.7.4 A decentralized Polynomial based SLAM algorithm for a team of mobile robots

Until recently, most research on this topic has involved a single mobile robot; on the other hand, using a team of robots to solve the SLAM problem allows to achieve the mapping goal faster and, hopefully, with a better performance w.r.t. the single robot case ([51]).

The main advantage of the multi-robot SLAM is the data exchange between robots. When two robots meet, they can exchange the currently ac-

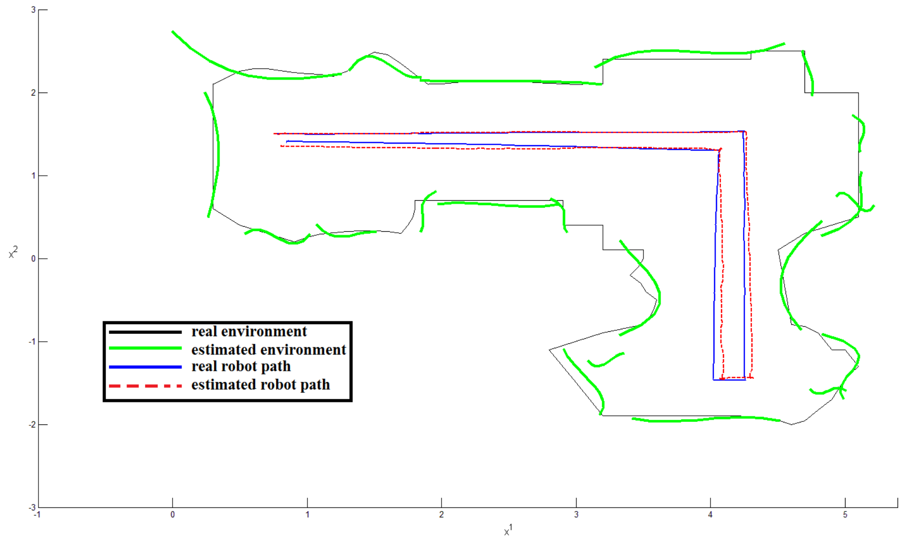


Fig. 1.27. *PbSLAM* simulation result

quired maps: if the two robots have explored and mapped different parts of the environment, after the data exchange each of them also knows the parts detected by the other. However, a team of coordinated mobile robots also introduces several sources of complexity: limited bandwidth; unreliable wireless communication channels; team coordination managing; shared map managing between robots; memory requirements (depending on the number of robots and the map size).

Multi-robot SLAM algorithms can be divided into two main groups: centralized and decentralized ones. In the first group, the main computations are performed by a central unit receiving the information acquired by the robots. The second group is related to decentralized algorithms where each robot makes its own computations and shares part of its own information only with the nearest robots. A centralized approach usually ensures better results, but if the central unit breaks, the whole algorithm fails. On the contrary, in a decentralized context, the algorithm works also if one of the teammates has a failure.

[52] proposes a decentralized SLAM algorithm combining fast maximum likelihood map growing with a Monte Carlo localizer based on particle representation. Authors make the restrictive assumptions of known relative robots initial positions and of overlap in robots maps. [53] propose a centralized structure for a team of mobile robots, which solves the mapping problem using a manifold based representation for two dimensional maps. The advan-

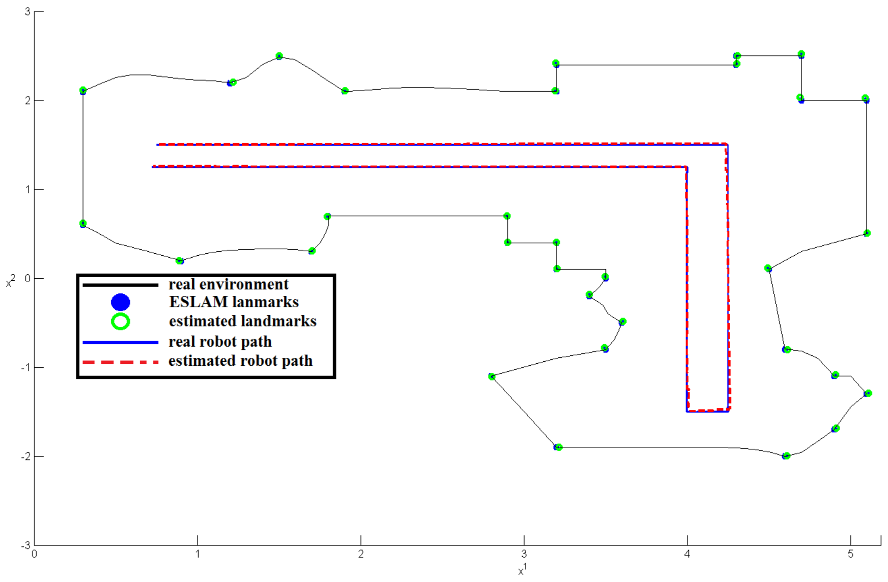


Fig. 1.28. *ESLAM* simulation result

tage is self-consistency when closing loops: maps are not affected by cross over problem.

[54] present a distributed Extended Kalman Filter (EKF) algorithm to build the local feature map for each teammate. The local maps are merged into a single global map after rendezvous events between robots. [55] propose a centralized multi-robot SLAM solution based on an EKF that estimates a state vector collecting the poses of the robots and the locations of the observed landmarks. [56] investigate the SLAM problem for a multi-robot system relaxing the assumptions made in [52] and proposing an application of Rao-Blackwellized Particle Filters for the purpose of cooperatively estimating SLAM posterior; each robot travels independently and each pair of robots exchange the acquired information once they meet.

All the cited solutions suffer of high costs in terms of energy consumption, since a large amount of information must be transferred between the robots (decentralized ap-proaches) or between the robots and the central unit (centralized approaches). Data transferring is the most energy consuming operation in a multi-robot SLAM algorithm, thus communications have to be correctly managed, and the amount of exchanged data must be kept to a minimum.

The goal of this paper is to solve the SLAM problem using a team of mobile robots with little data communications. We propose a decentralized solution based on approximating the environment boundaries by a set of polynomials.

By the simple mathematical characterization of a polynomial, exchanging such data between two robots requires a low communication cost, saving robots' batteries.

Two main parts may be detected into a multi-robot SLAM algorithm: (1) the part regarding the single robot behavior and (2) the part about the robots behavior when they meet. Regarding the first part, since the robot is equipped with its own sensors, it is able to build a map and localize itself on its own. As for the second part, the idea is to use the data exchanged between two robots once they meet and are involved into a *rendezvous* event.

Each SLAM algorithm is based on an estimation algorithm to localize the robot and simultaneously build the environment map. The most used SLAM estimator in the literature is the Extended Kalman Filter (EKF), which can be easily adapted to the SLAM context by defining an augmented state containing both the robot pose and the environment landmarks.

1.7.4.1 Single robot SLAM

The first module of the SLAM strategy proposed in this section regards the SLAM of a single robot. Using a team of robots has some advantages, as it will be shown, but it does not exclude the possibility, for each robot, to build its own map without communicating with other robots. In the present thesis, two different SLAM strategies have been proposed, the PbSLAM (see Section 1.7.1) and the EPbSLAM (see Section 1.7.2); the first one has better localization performance but requires a high computation time and cannot be used in real time, while the EPbSLAM has slightly worst localization performance but its computational cost is very low compared to the PbSLAM; thus, the EPbSLAM algorithm will be used in this section.

1.7.4.2 Data exchange process

The second part of a multi-robot SLAM algorithm is related to the interactions of robots once they meet. Let the robots poses be given in the absolute reference frame. Each robot r_j starts its path from an unknown initial position $x_{j,0}, j = 1, \ldots, N$. As a consequence, each robot assumes to start from $\hat{x}_{j,0} = [0\ 0\ 0]^T$ and uses this pose as its relative mapping and localization reference frame.

Let $R_{j,k}$ be the reference frame consistent with robot r_j at step k: it is centered on robot r_j estimated position $(\hat{x}^1_{j,k}, \hat{x}^2_{j,k})$ and its x^1 axis is rotated of an angle $\hat{\theta}_{j,k}$ w.r.t. the robot relative reference frame $R_{j,0}$. The algorithm outputs will be the robots estimated trajectories and maps, $\hat{x}_{j,k}, \mathbb{E}_{j,k}, j = 1, \ldots, N$, and they will be related to the relative robot reference frame $R_{j,0}$, and thus biased w.r.t. the absolute reference frame by $x_{j,0}$ offsets, respectively.

Each robot starts its path into the environment, taking measurements by its distance sensors and running its own SLAM algorithm. No assumptions

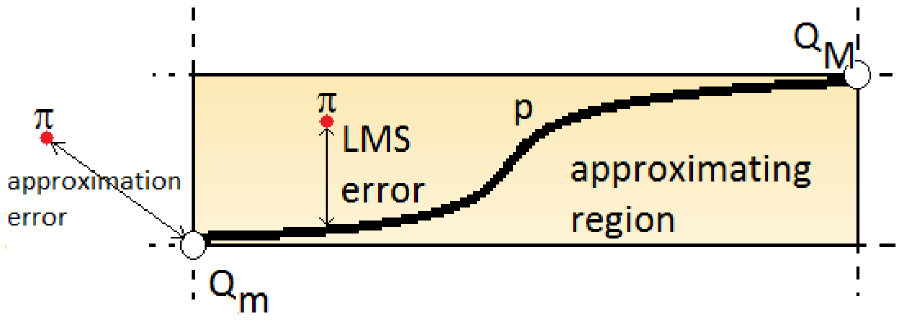


Fig. 1.29. Polynomial approximating region and approximation errors.

are made about robots relative pose; they move as they were on their own and interact only when a rendezvous occurs. A rendezvous happens when two robots r_j and r_l detect each other; in that case, the two robots establish a connection allowing r_j sending its map $\mathbb{E}_{j,\bar{k}}$ to r_l and vice versa. This event is denoted by the involved robots (r_l, r_j) and the time step it occurs, $\bar{k}$. In the following, the index l will denote the robot that sends the data, while the index j will denote the robot that receives data from r_l.

To use the map $\mathbb{E}_{l,\bar{k}}$, received by r_l, robot r_j has to rotate and translate this map w.r.t. its own reference frame $R_{j,0}$, coping with the bias due to the different reference frames. This transformation translates each polynomial in $\mathbb{E}_{l,\bar{k}}$, expressed in $R_{l,0}$, into the reference frame $R_{j,0}$. However, when a polynomial is rotated, the resulting curve is no mo-re a polynomial in the new reference frame, unless $m = 1$. To face this problem, once r_j has received the coefficients of a m-th order polynomial p from r_l, it approximates the polynomial by a piecewise linear approximation made of a set of segments $\{s_\tau, \tau = 1, \ldots, \mathcal{T}\}$.

After the segments $\{s_1, \ldots, s_\mathcal{T}\}$ have been obtained, their starting/ending points are rotated and translated to express them into the r_j reference frame. This transformation is obtained by finding the transformation matrix $T_{l0,j0}$ from the reference frame $R_{l,0}$ to the reference frame $R_{j,0}$. As shown in [56], the transformation matrix can be found using the rendezvous information:

$$T_{l0,j0} = T_{j,j0}\; T_{l,j}\; T_{l,l0}^{-1},$$

where $T_{l,l0}$ is the transformation matrix from the reference frame $R_{l,\bar{k}}$ consistent with robot r_l during the rendezvous to the reference frame $R_{l,0}$. Along the same lines, $T_{j,j0}$ is the transformation matrix from the reference frame $R_{j,\bar{k}}$ to the reference frame $R_{j,0}$. The transformation matrices are found using the information about $\hat{x}_{l,k|k}$ and $\hat{x}_{j,k|k}$, respectively. Finally, $T_{l,j}$ is the transformation matrix from $R_{l,\bar{k}}$ to $R_{j,\bar{k}}$, and it depends on the relative pose between robots:

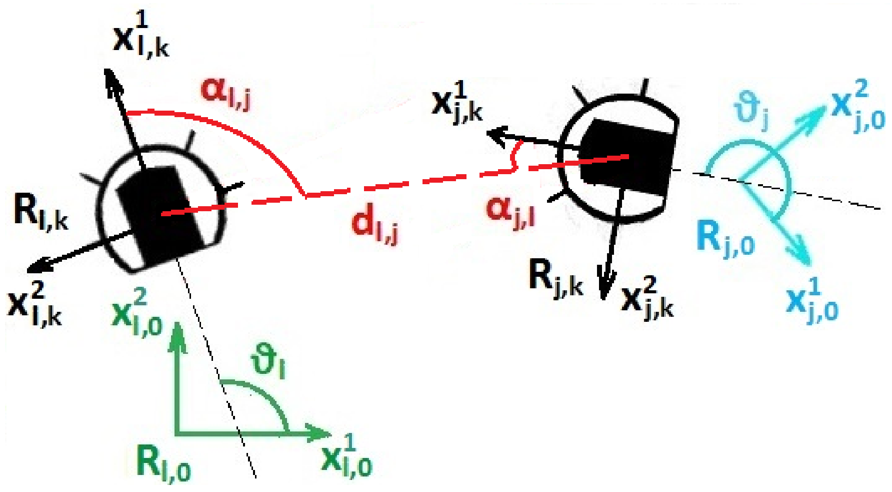


Fig. 1.30. Reference frame involved in coordinate transformation during a rendezvous events

$$T_{l,j} = \begin{bmatrix} \cos(\vartheta_{l,j}) & -\sin(\vartheta_{l,j}) & d_{l,j}\cos(\vartheta_{l,j}) \\ \sin(\vartheta_{l,j}) & \cos(\vartheta_{l,j}) & d_{l,j}\sin(\vartheta_{l,j}) \\ 0 & 0 & 1 \end{bmatrix},$$

where $d_{l,j}$ is the relative distance between robots, and $\vartheta_{l,j} = \pi + \alpha_{l,j} - \alpha_{j,l}$; angles $\alpha_{l,j}$ and $\alpha_{j,l}$ are shown in Fig. 1.30. Using $T_{l0,j0}$, the map $\mathbb{E}_{l,\overline{k}}$ is transformed into a map in the r_j reference frame. Let this map be $\mathbb{E}_{l\to j,\overline{k}}$,: robot r_j can now update its own map by adding to it $\mathbb{E}_{l\to j,\overline{k}}$.

Remark 1.6. **1** The solution here proposed only requires to transfer the environment modeling polynomials from the previous rendezvous to the current one, thus the amount of data to be transferred is extremely low. Moreover, if two robots meet more than one time, they have to exchange only the polynomials extracted after the last rendezvous.

Remark 1.7. **2** The proposed strategy uses polynomials to model the environment and then approximates them as a set of segments when they have to be rotated. An alternative approach could be the direct use of segments, as shown in [18]. However, using polynomials instead of lines is expected to yield to better mapping results, and if the segments based approximation before the rotation is sufficiently accurate, the resulting set of rotated segments should preserve the accuracy due to the polynomial approximation.

1.7.4.3 Map merging

Map merging is a problem of interest for both the single robot and the multi-robot SLAM context, because in both cases, when a new landmark p is obtained from another robot or it is extracted from current measurements, it has to be compared to previously acquired ones and eventually merged with one of them. Following the same lines as in [49], the map merging process is based on the geometrical analysis of the polynomial landmarks.

Two cases may occur: in the first case, the extrema of p are included or completely include the extrema of a mapping landmark p_2: see Fig. 1.18(a); in the second case, the two polynomials p and p_2 overlap for a sufficiently large region: see Fig. 1.18(b). In both cases, let $Q_0, Q_1, \ldots, Q_m$ be $m+1$ equally spaced points on p in the overlapping region between p and p_2: see Fig. 1.18(b). Let γ be the LMS approximation error due to using p_2 in mapping the Q_i points. If γ is lower than a threshold γ_M, then p and p_2 are merged by creating a new LMS polynomial from ε_M points equally spaced on p and ε_M points equally spaced on p_2.

To summarize, three landmark creation situations may occur: (1) a new landmark is created from measurements not mapped yet; (2) an existing landmark is modified because of a high number of badly approximated measurements; (3) a new polynomial is computed by merging two existing ones. In all the cases, a weighted LMS technique is used. The badly approximated points and the not mapped points are weighted by $1/||P_{k|k-1}||$, where $P_{k|k-1}$ is the covariance matrix related to the predicted pose used to compute that point. The points in $\mathcal{R}$ are weighted using the inverse covariance related to the innovation landmark relative to the polynomial to be updated; the same technique is used in the third case for the polynomials to be merged. As a consequence, the function $g(x, y)$ of Eq. (1.48) is the weighted least mean square function: depending on the landmark creation, its derivatives are computed taking into account polynomials rotation, translation and update.

1.7.4.4 Numerical results

To assess the performance of the proposed algorithm, numerical simulations have been performed using $N = 3$ differential drive robots, each one assumed to be equipped with five distance sensors to detect the environment and one camera to ensure other robots detection.

A set of 100 numerical simulations have been performed with the following parameters: sampling period $T = 1$ s, $m = 3$, $\underline{\rho} = 0.08$ m, $\overline{\rho} = 0.24$ m, $\underline{\sigma} = 0.1$ m, $\overline{\sigma}_m = 0.3$ m, $\varepsilon_M = 5$, $\varepsilon_Z = 10$, $K = 150$, $R = 0.35$ m, $\gamma_M = 0.35$ m. Distance sensors are assumed to be affected by a Gaussian noise with zero mean and standard deviation of 0.02 m. The process noise is assumed to have a standard deviation of 0.001 m on the position and of 0.01 degrees on the heading. A matrix $P_0 = \text{diag}\{0.05^2, 0.05^2, 0.0175^2\}$ has been used as initial estimation error covariance matrix for all the robots.

Table 1.5. Averaged indexes over 100 simulations - World 1

	robot r_1	robot r_2	robot r_3
μ_j	$3.5324 \cdot 10^{-8}$	$2.7924 \cdot 10^{-8}$	$8.8341 \cdot 10^{-9}$
ν_j	$4.3792 \cdot 10^{-3}$	$3.8437 \cdot 10^{-3}$	$1.872 \cdot 10^{-3}$ m
τ_j	0.61 s	0.66 s	0.62 s

Different environments have been tested, in order to provide a more comprehensive algorithm evaluation. Each robot moves in only a part of the environment and could not map all the environment boundaries on its own. Each path has been performed in $k_f = 200$ steps ensuring rendezvous only between robots r_2 and r_3.

To evaluate the algorithm localization performance, a NEES index slightly different from the Eq. (1.29) has been used:

$$\mu_j = \frac{1}{k_f + 1} \sum_{k=0}^{k_f} \left[x_{j,k} - \tilde{x}_{j,k|k}\right]^T P_{j,k|k} \left[x_{j,k} - \tilde{x}_{j,k|k}\right]$$

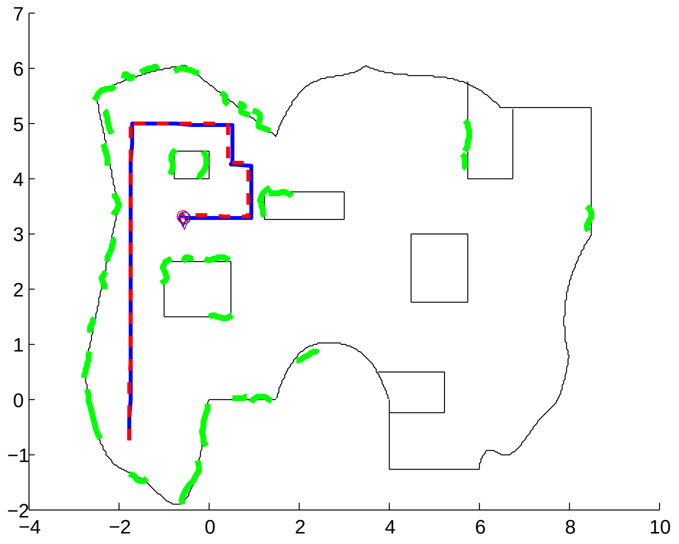
where $P_{j,k|k}$ is the robot r_j localization error covariance matrix and $\tilde{x}_{j,k|k}$ is the robot r_j estimated pose in the absolute reference frame (compensating the bias due to the use of $\hat{x}_{j,0} = [0\ 0\ 0]^T$).

Averaged results for μ_j, ν_j and the computation time[3] τ_j, over the 100 simulations are shown in Tables 1.5 and 1.6.

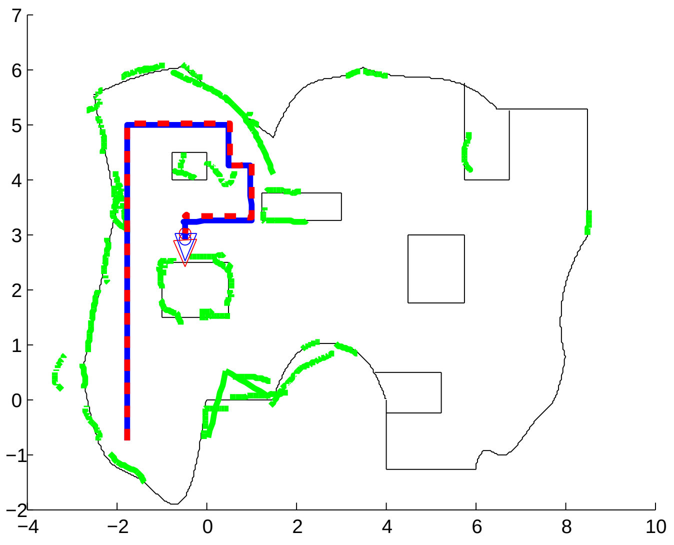
Typical results of the proposed SLAM algorithm are shown in Figs. 1.31 to 1.35 for the three robots. For each couple of figures, the upper one shows the results obtained by the robot moving independently, while the lower one shows the results obtained by the same robot, following the same trajectory, but taking into account also for data sent by other robots. In this particular experiment, the robot r_1 is not involved in any rendezvous event and, as shown in Fig. 1.31, its map is incomplete. On the contrary, using a multi-robot approach, it is possible, for each robot, to have a map of not directly detected environment sections; in this way, the time required to build a complete map of the explored environment is significantly reduced.

Table 1.5 and 1.6 show that both the error due to robots pose localization and the one related to environment mapping are very low, thus the proposed algorithm is efficient in solving the SLAM problem. Moreover, the computation times are sufficiently low w.r.t. the chosen sampling period to use the algorithm in real time during robots motion.

[3] The algorithm computation times have been computed using Matlab R2012 running on an Intel(R) Core(TM) i7 Q720 CPU



(a) Single-robot SLAM results

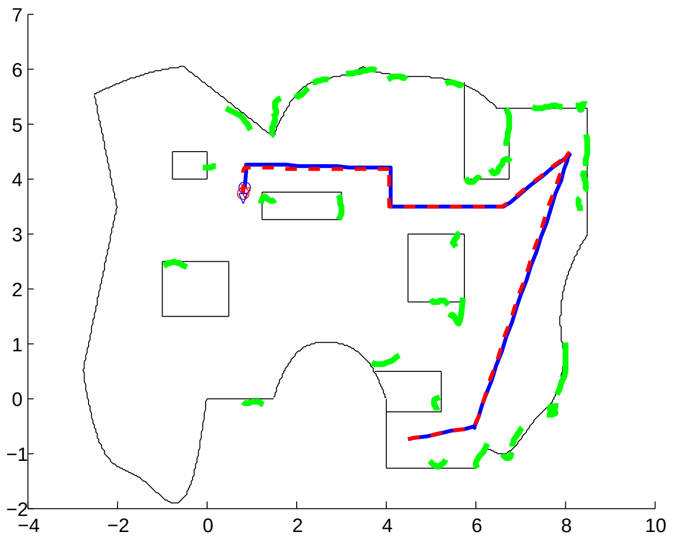


(b) Multi-robot SLAM results

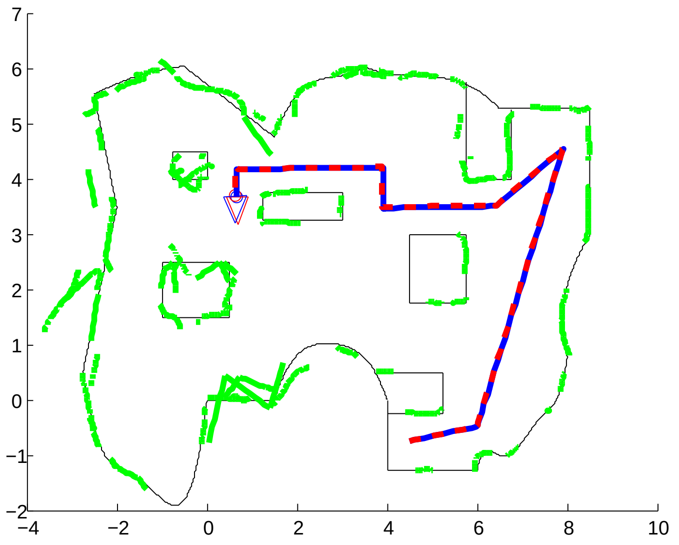
Fig. 1.31. Results obtained by the robot r_1 in the world 1; green: mapped environment; red: real environment; blue: real pose; dashed red: estimated pose.

1.7.4.5 Remarks

In this section, the SLAM problem has been faced using a team of mobile robots. Due to the good compromise between accuracy and required compu-



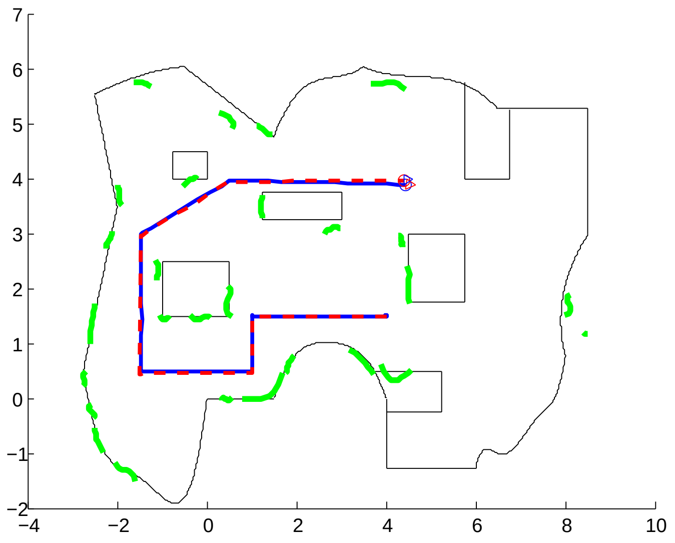
(a) Single-robot SLAM results



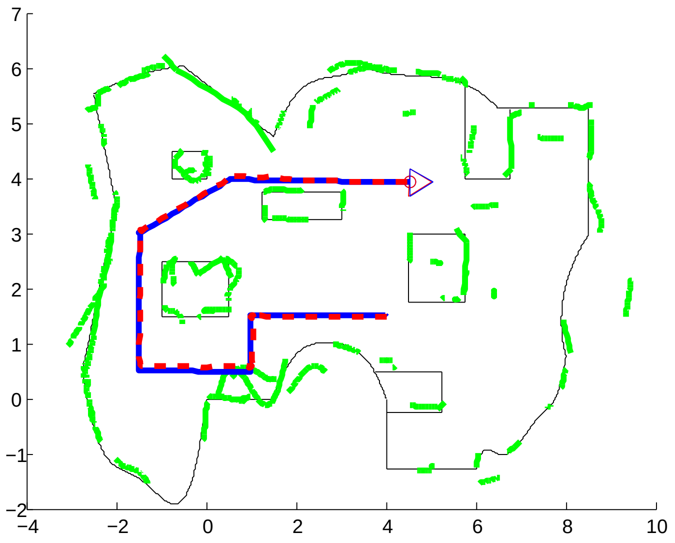
(b) Multi-robot SLAM results

Fig. 1.32. Results obtained by the robot r_2 in the world 1; green: mapped environment; red: real environment; blue: real pose; dashed red: estimated pose.

tational cost, the environment mapping method proposed in Section 1.7.2 has been used. Moreover, a key feature is that each robot in the team is able to build its own map; thus possible faults on a robot would not affect on the



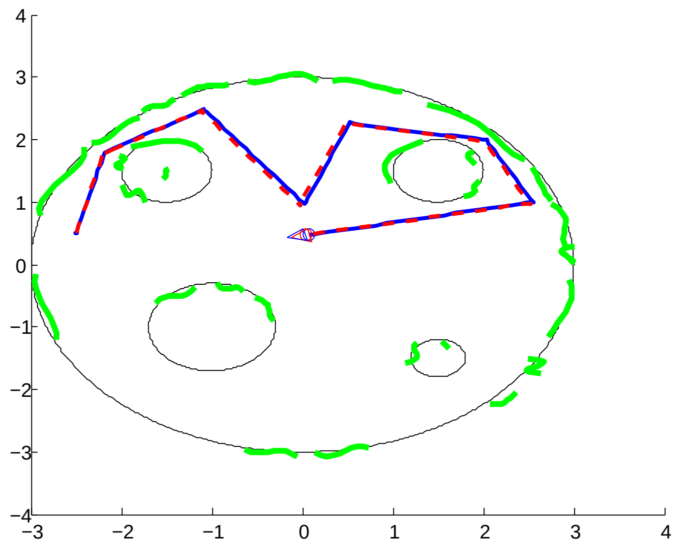
(a) Single-robot SLAM results



(b) Multi-robot SLAM results

Fig. 1.33. Results obtained by the robot r_3 in the world 1; green: mapped environment; red: real environment; blue: real pose; dashed red: estimated pose.

others. Also, when two robots are involved into a rendezvous, only the polynomials created after the previous rendezvous (if any) have to be exchanged, with a low communication effort. Once a robot has received the information


(a) Single-robot SLAM results

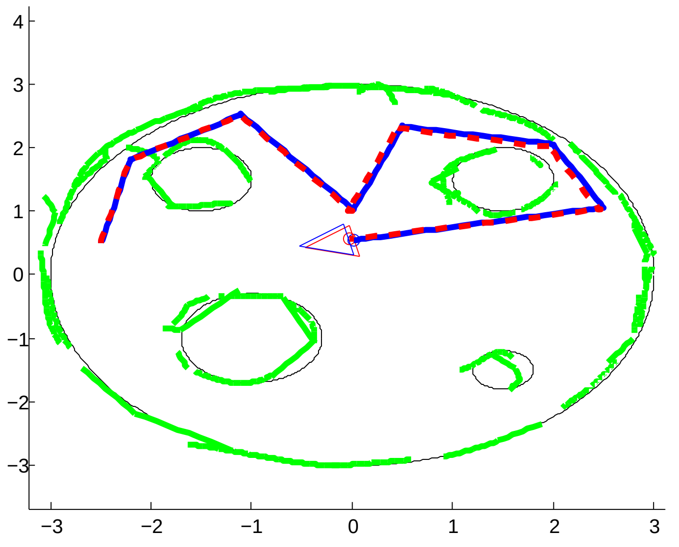
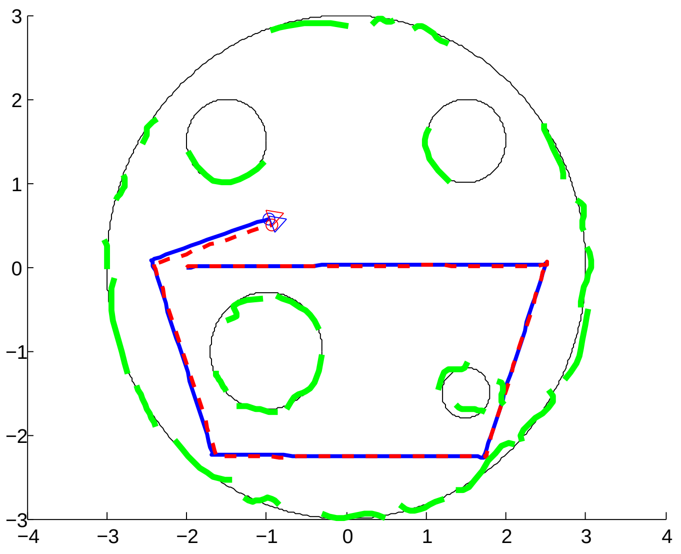
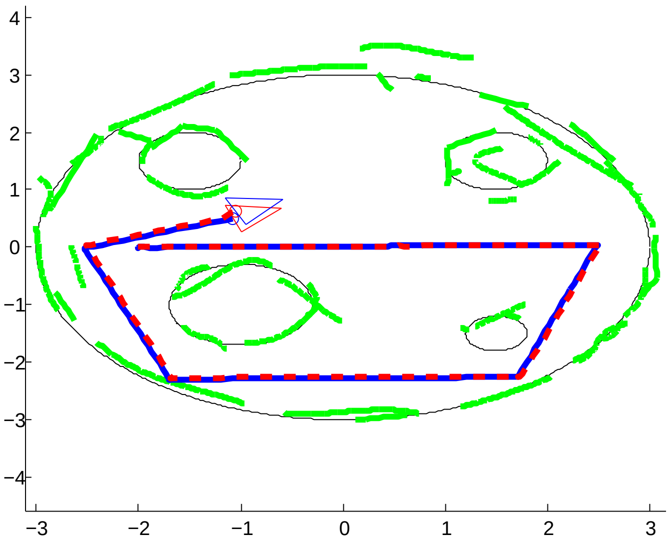

(b) Multi-robot SLAM results

Fig. 1.34. Results obtained by the robot r_1 in the world 2; green: mapped environment; red: real environment; blue: real pose; dashed red: estimated pose.

from another one, it transforms this information in its own reference frame by means of simple geometric considerations. Numerical simulations have shown



(a) Single-robot SLAM results



(b) Multi-robot SLAM results

Fig. 1.35. Results obtained by the robot r_2 in the world 2; green: mapped environment; red: real environment; blue: real pose; dashed red: estimated pose.

the effectiveness of the proposed algorithm, with particular reference to the environment mapping results.

Table 1.6. Averaged indexes over 100 simulations - World 2

	robot r_1	robot r_2
μ_j	$3.9975 \cdot 10^{-8}$	$1.6435e \cdot 10^{-8}$
ν_j	$4.5004 \cdot 10^{-3}$	$3.0543 \cdot 10^{-3}$
τ_j	0.4089 s	0.4754 s

1.8 Sensors switching policies

When a problem of localization has to be solved, a Kalman Filter is often used to reconstruct the robot position; it comes natural to think that the more sensors you use, the better estimate you get. However, it must be taken into account that sensors employ robot's batteries, and an intensive use reduces the robot autonomy. Moreover, it has been shown in [57] that sometimes, when tracking mobile targets, the estimation can be affected by a bias; then it is not always convenient to exceed with the number or use of sensors.

The most important is that there are situations where multiple sonar sensors cannot operate simultaneously, for example when they use the same frequency band [58].

When the filter is not run onboard 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 filter. Thus, using an appropriate sensors switching policy, only a part of all available sensors will be used, and the energy consumption will be reduced. However, this may result in a lower accuracy of the robot state estimation.

Due to the above considerations, the problem is related to designing a measurements switching strategy which allows to find 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 section, assuming to use ultrasonic sensors, a new sensors switching policy is presented. Such policy is based on the sonar sensor physical characteristics and, on the contrary of the switching rule presented in [31], it has a very low computational cost.

1.8.1 Sonar sensor model

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. 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.

As remarked in [47] and [48], the measurements provided by these sensors are very influenced by the angle of incidence between the "sensor beam", whose typical shape is shown in Figure 1.9, and the surface intercepted. When the sensor is perpendicular to a flat surface, the measurement provided is the true range, within tolerance, to the surface. However, the measurement error can be much larger when the beam strikes a surface at a different incidence angle, γ. In this work the main purpose is to reduce such disadvantage choosing the sensors to use on the basis of their related incidence angles.

1.8.1.1 Incidence Angle

Consider an ultrasonic sensor, S, which provides the distance, y, from a surface U, as depicted in Figure 1.9.

Let $l(x^1, x^2)$ be the tangent line to the boundary of the surface in the intersection point, A, between the incidence surface and the sensor axis. Defining $\vec{r}$ as the unit vector of the sensor axis and $\vec{b}$ as the unit vector of $l(x^1, x^2)$, it is possible to define the incidence angle γ as

$$\gamma = \arccos(\vec{r} \cdot \vec{b}) \tag{1.54}$$

The more the incidence angle is near to $\frac{\pi}{2} rad$, the better the measurement provided by the ultrasonic sensor is. Using (1.54) the following incidence parameter can be defined:

$$p = \cos\gamma = \vec{r} \cdot \vec{b} \tag{1.55}$$

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

1.8.1.2 Measurement Model

To take in consideration the influence of the incidence angle into the measurement model related to the ultrasonic sensor, we model the measurement provided by such sensor as

$$y_M = y_R + \xi(p)$$

where y_R is the real distance between the sensor and the incidence surface, y_M is the measurement provided by the ultrasonic sensor and $\xi(p)$ is a Gaussian noise of zero mean and covariance matrix $V(p)$ related to the parameter p. 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 measurement, the incidence function $\xi(p)$ has to be chosen such that:

1. its standard deviation is monotonic increasing in p;
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 $\gamma = \frac{\pi}{2} rad$.

1.8.2 Robot localization and new sensors switching policy

The NEKF localization algorithm, introduced in Section 1.6.4 is used to estimate the robot's state. The goal now is to develop an effective strategy to select the most reliable measurements.

Assuming to have j sensors available, choosing at any time instant to activate q out of them $(q \ll j)$ is an old problem. The interest about such problem was motivated, more than two decades ago, by the limited capacity of the transmission medium and the low computational power of the processors. The above limitations imposed to select at any instant only a few sensors.

Although nowadays such difficulties are vanished, due to the huge developments in the transmission and computational devices, a new and important problem, especially in mobile robot applications, is related to save the sensors' batteries lifetime. To this end it could be very meaningful to device a policy that uses a small part of the available sensors to limit power consumption. Obviously such policy has an impact on the quality of the estimate since there is a trade off between the number of sensors used and the performance of the resulting estimation algorithm. Thus the best policy is the one that, for a number $q \ll j$ of sensors, gives the best estimate, in some statistical sense.

To formalize the problem, suppose that at each instant only q out of j sensors are activated. A q-valued output equation $h(\cdot)$ and a $q \times n$ output matrix C_k related to such sensors can be evaluated by Eq. (1.5) and (1.24). Changes in the activation sequence clearly return different estimates of x_k for the EKF. Our aim is to obtain the best estimate, in some statistical sense, using only $q \ll j$ sensors.

The proposed switching policy is based on the sonar sensor model presented in Section 1.4.1. The main idea is:

> *Among the j sensors, choose the subset of q ones related to the best incidence angles.*

Given the set of available sensors $\{S_i, i = 1, \ldots, j\}$, each of them is related to an incidence angle γ_i and then to an incidence parameter p_i. The q sensors related to the lower values of p_i are chosen.

If no assumption on the environment is made and the environment is totally unknown, the incidence angles are not known *a-priori*. To use the proposed sensors switching rule an estimate of the incidence angles, $\{\hat{\gamma}_i, i = 1, \ldots, j\}$, or, which is completely equivalent, of the associated incidence parameters $\{\hat{p}_i, i = 1, \ldots, j\}$, has to be found.

To obtain such estimate the information provided by the NBA can be used.

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 $r_i(g) = [g\ ,\ a_i g]^T\ ,\ g \in \mathbb{R}$, thus, the unit vector, $\vec{r}$, related to the sensor's axis is $\vec{r}_i = r_i(g)/||r_i(g)||$.

Using the current estimate $\hat{x}_{k|k}$, an estimation of a_i can be found as $\hat{a}_i = \tan(\hat{\theta}_{k|k} + \alpha_i)$. Thus an estimation of $r_i(g)$ is $\hat{r}_i(g) = [g\ ,\ \hat{a}_i g]^T$ and finally the estimated unit vector is

$$\hat{\vec{r}}_i = \hat{r}_i(g)/||\hat{r}_i(g)|| \tag{1.56}$$

The Neighbours 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 boundary of the portion of the environment intercepted by such sensor's axis. Therefore, for each used sensor S_i, an approximation of the unit vector $\vec{b}_i$ of the line $l_i(x^1, x^2)$ can be found as

$$\hat{\vec{b}}_i = \hat{b}_i(g)/||\hat{b}_i(g)|| \tag{1.57}$$

where $\hat{b}_i(g)$ is the approximation of the parametrized vector on the line $l_i(x^1, x^2)$, that is $\hat{b}_i(g) = [g\ ,\ \hat{c}_i g]^T$

At this point, using the equations (1.56) and (1.57) along with the Eq. (1.55), it is possible to obtain an approximation of the incidence parameter related to each sensor S_i as

$$\hat{p}_i = \hat{\vec{r}}_i \cdot \hat{\vec{b}}_i \tag{1.58}$$

Thanks to the above equation, at each time step k the following steps can be performed to use the presented switching logic:

Sensors Switching Algorithm

Given $\mathcal{I}_k$ and $(\hat{\bar{s}}_k, \hat{\bar{c}}_k) = NBA(\mathcal{I}_k, \{i = 1, \ldots, j\}, k)$, do

1. evaluate $\hat{\vec{r}}_i, i = 1, \ldots, j$, using (1.56)
2. evaluate $\hat{\vec{b}}_i, i = 1, \ldots, j$, using (1.57)
3. evaluate $\hat{p}_i = \hat{\vec{r}}_i \cdot \hat{\vec{b}}_i, i = 1, \ldots, j$, using (1.58)
4. $\mathcal{I}_{k+1} = \mathcal{MIN}(\{\hat{p}_i, i = 1, \ldots, j\}, q)$

where, $\mathcal{I}_k$ is the set of indexes used at previous step by the switching rule 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.

Other possible sensors switching logics can be found in [59] for the state estimation of linear systems and in [31] for the state estimation of nonlinear systems using the Extended Kalman Filter. Such switching logics are based on choosing the subset of sensors whose measurements have maximum effect on the current estimation. These policies have a combinatorial computational cost on the number, j, of available sensors.

On the contrary, the proposed policy has a polynomial computational cost on the number, j, of sensors. This cost is mainly related to the use of the NBA in the switching algorithm. Moreover, using the proposed policy along with the NEKF algorithm, the computational cost of the filter is not increased since the parameters $(\hat{\bar{s}}_k, \hat{\bar{c}}_k)$ obtained by the filtering algorithm can be used also by the switching policy. In such situation, the additional cost due to the switching policy, in the worst case, is only related to sorting the elements $\hat{p}_i, i = 1, \dots, j$, and it is $O(j \log j)$. Clearly such cost is less than the NEKF cost.

In principle q could depend on k, but here we will assume it constant for simplicity. Since there is no guarantee on the optimality of the proposed choice to find the optimal switching sequence (finding it would be a problem of combinatorial complexity), we look at this criterion as a heuristic method to improve the battery life without affecting the obtained estimation.

1.8.3 Experimental results

To evaluate the performance of the proposed algorithm for the mobile robot localization problem, some experimental tests have been made.

First of all, to estimate the noise covariance function, a series of experiments has been performed, 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, and the corresponding standard deviation has been estimated. The obtained results are shown in Figure 1.36.

Using the obtained data, the standard deviation of the incidence function $\xi(\cdot)$ can be estimated through an 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 p and thus the experimental data confirm the correctness of the model used for the ultrasonic sensors. The obtained estimate of the standard deviation function of $\xi(\cdot)$ can be used to evaluate the covariance matrix $V(\overline{p}_k)$ used in the NEKF algorithm.

At this point, to evaluate the performance of the proposed sensors switching rule two series of experiments have been performed, each one refereeing to a different trajectory followed by the robot Khepera III. Each trajectory is internal to a “simple” rectangular world of $1.5 \times 1.0m$. The first trajectory is a circle of radius $0.20m$ centered in $(0.60, 0.50)m$. The trajectory starts from $(0.60, 0.30)m$ and continues in the counterclockwise. The second one is a square of $0.3m$ with $(0.60, 0.30)m$ as starting point. For each of them, 20 experiments have been performed. A sample time $T = 1$ second has been taken. Both trajectories are completed in $k_f = 100$ seconds. For each trajectory a trapezoidal profile has been imposed to the wheels angular velocities.

In each experiment, three different kinds of filter 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 sensor configurations. The second

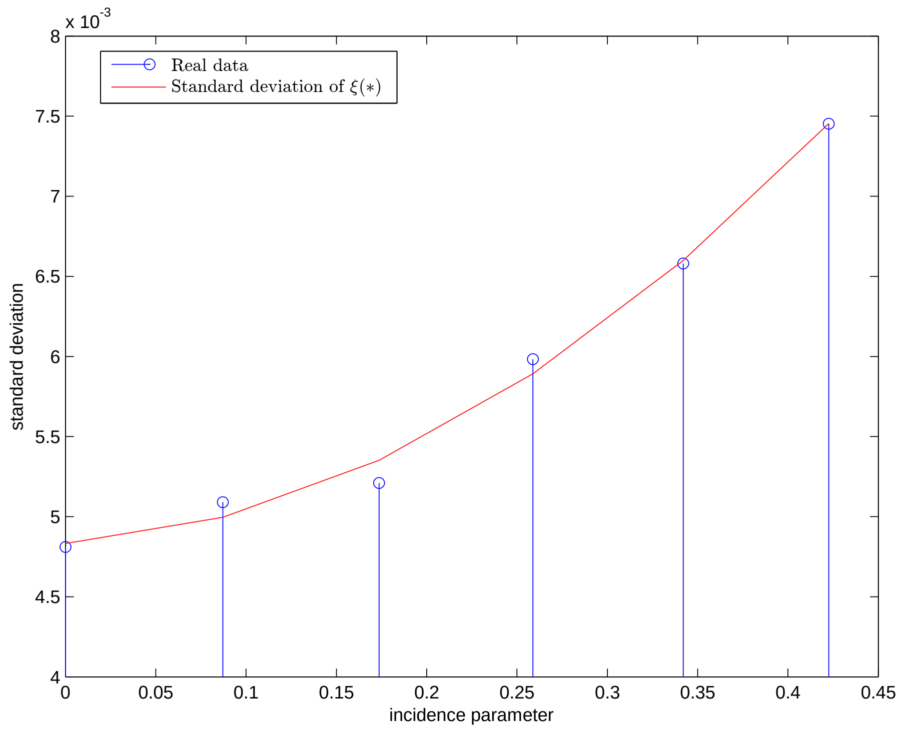


Fig. 1.36. standard deviation function estimation

one uses two sensors ($q = 2$) out of the five available, switching between sensors using the above discussed switching policy (I_k is chosen at each step using the proposed sensors switching algorithm in order to include the measurements with the best incidence angles). The third one uses all five sensors together ($I_k = \{1, \dots, 5\}, k \in \{0, \dots, k_f\}$), and it is used as a reference. The following values for the covariance parameters and the initial conditions were common to all experiments:

- $W = \text{diag}\{0.00029, 0.00019, 0.09274\}$, which corresponds to a standard deviation of 0.017 m on x^1, 0.014 m on x^2 and of 17° on θ.
- $V(0) = \text{diag}\{0.081^2, 0.123^2, 0.061^2, 0.096^2, 0.050^2\}$.
- $\hat{x}_{0|0} = [0.59, 0.31, 0]'$; the true initial state of the robot is $x_0 = [0.6, 0.3, 0]'$ for each trajectory.
- $P_{0|0} = \text{diag}\{0.01, 0.01, 0.003\}$, which corresponds to a standard deviation of $0.1m$ on the robot position and 1° on the orientation.

In all the performed experiments, a value of $R = 0.2m$ has been adopted as proximity radius in the NbA algorithm. With regard to the initialization of the set $\mathcal{M}$, of acquired environment points, and of the parameters

Table 1.7. index $\varepsilon[\times 100]$

Traj.	**Circle**	**Square**
Switching	8%	13%
z_1	10%	14%
z_2	21%	15%
z_3	13%	14%
z_4	17%	17%
z_5	16%	13%
z_6	10%	12%
z_7	15%	17%
z_8	16%	15%
z_9	21%	17%
z_{10}	12%	13%
All	18%	15%

$\{(\hat{s}_i, \hat{c}_i), i = 1, \ldots, 5\}$, we start our filters 15 steps after the beginning of each experiment. During these initial steps measurements are acquired to form the initial condition for $\mathcal{M}$ and $\{(\hat{s}_i, \hat{c}_i), i = 1, \ldots, 5\}$. To evaluate the performance of the filters the following index has been introduced

$$\varepsilon[\%] = \frac{1}{k_f + 1} \sum_{k=0}^{k_f} \frac{||x_k - \hat{x}^*_{k|k}||}{||x_k||} \times 100$$

where $\hat{x}^*_{k|k}$ is the estimated state with one of the tested filters and x_k is the real value of the robot pose. Such index gives information about the state estimation error. In Tables I the obtained results are reported; the values are averaged over the 20 experiments. In this table, "Switching" refers to the proposed switching policy with $q = 2$; "All" refers to the case where all sensors are used together; "z_i" indicates a generic pair of sensors among the $\binom{5}{2}$ available.

As it can be seen by the obtained results, the proposed switching rule provides very good estimation results for both the trajectories yielding to a very low value of the estimation error ε. Please note that due to the high measurement noise, using all the sensor is not the best possible choice. Obviously if the used sensors are not too noisy, the minimum value of the estimation error would be obtained using all the sensors simultaneously.

To further test the proposed switching rule, a set of 1000 numerical simulations, based on the same standard deviation function of $\xi(\cdot)$ and parameters of 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 performs better than each other possible fixed pair of sensors.

1.8.4 Remarks

In this section, a sensors selection policy has been described, in order to improve the performance of the EKF used to localize a mobile robot. The localization algorithm is based only on local data and does not require any assumption on robot's working environment. Moreover a new sensor model has been proposed and validated by experimental data. Real experiments using the robot Khepera III have been performed and the obtained results have shown that the proposed switching rule leads to high estimation performance, regardless of the path followed by the robot.

As future research directions loss of information on the data transmission channel may be considered and the proposed switching rule will be adapted to other extensions of the Kalman filter, such as the Unscented Kalman Filter. Moreover studies on switching rule related to a time varying number of used sensors ($q = q(k)$) may be taken into account.

Part II

Sensors systems

2

Displacements monitoring systems using new low-cost technologies

2.1 Introduction

This chapter is dedicated to presenting a new system for structural health monitoring. In particular, we refer to the dynamic tests on shaking tables, an important class of experiments aimed to monitor the behaviour of certain structures under the action of a seismic force. In addition to a shaking table, which is provided with a number of degrees of freedom from a minimum of one to a maximum of six in order to simulate the seismic action, such experiments involve the use of appropriate measurement systems able to detect the structural response at certain points.

The structure behavior is typically detected by various kinds of sensors, such as accelerometers, gyroscopes and potentiometers. A limitation of this approach lies in the fact that such devices must necessarily be placed in contact with the structure to be examined , a procedure that may be quite ineffective in the case of destructive tests. Then it looks convenient to use alternative sensing systems to the above mentioned devices.

In this context, computer vision can be a valuable tool to support the development of alternative approaches to the detection of displacements corresponding to structural points . In [60] an example of vision-based system is presented, which uses high-resolution cameras to detect the movements of specific markers placed on the structure to be analyzed. Special IR emitters are used to illuminate the scene, while cameras are equipped with IR filters, so that only light reflected by the markers is visible. Thanks to the high resolution used cameras, the system is able to identify marker movements with high accuracy. However, the primary assumption made in this work is that markers movements take place on a plane; this is a restrictive hypothesis that in general would be not verified, for example during tests on shaking tables with more than one degree of freedom or tests on irregular shaped buildings. Although the same paper presents some solutions to reduce the error due to the failure to satisfy the primary hypothesis , the most probably suitable solution would be to use a stereovision algorithm, effective for the identification of objects in 3D space using 2D images.

So, the goal of the present work is to build a system able to replace classical sensors usually adopted in structural monitoring, using a really promising technology in the field of 3D-space monitoring which in recent years has attracted more and more interest in the research world; it makes use of the so called depth images. A depth image is a grayscale image that contains scene depth information for each pixel, so it is a very useful tool to describe many different kinds of surfaces. Such an image can be calculated by disparity estimation between two stereo-images. Depth images can be used in many fields such as gaming, robotics, medical field, etc., thanks also to the invention of ToF (Time of Flight) cameras [61] and low-cost RGB-D sensors, such as Kinect and Xtion [62, 63]. The TOF cameras calculate the distance from an object by evaluating the phase delay between an IR wave emitted by the camera and the one reflected by the object.

On the other side, RGB-D sensors are capable to measure both color and depth information; in fact, in addition to capturing color images by using a RGB camera, such sensors include a light source used to project an infrared light pattern onto an observed object and another device to capture the pattern reflected by the object; an internal processor is used to compare the emitted pattern shape and the received one, in order to reconstruct the depth values. In this way, a depth image can be created without the need of processing two color images, reducing the required memory. However, more details about the depth image creation will be given in the following sections. We intend to adapt such an interesting technology to an important issue in structural monitoring, such as shaking table tests, in order to improve the monitoring system security. In particular, we focus on a displacements monitoring system.

First of all, in Section 2.2 a definition of the problem to be addressed will be given and the requirements needed to implement a system based on depth image for structural monitoring will be described; then, in Section 2.3 , the system will be described in detail, with explanations about the theoretical models used in the various steps of analysis; Section 2.4 will present some experimental results, and finally, in Section 2.5, conclusion will be drawn and future works will be discussed.

2.2 Problem statement

The goal of the present paper is to try to create a system based on depth image for structural monitoring during shaking table tests . This system will make use of the above mentioned RGB-D sensors, in particular of the Microsoft Kinect. In addition to capturing color images by using a RGB camera, such sensors are capable to measure the pixels depth without the need of processing two color images.

A monitoring system is generally be composed of various sensors , depending on the size of the structure to be monitored. Therefore, in addition to data acquisition devices, an appropriate interface software is required to process all the acquired images.

Moreover, also the choice of the markers to be placed on the structure has an important role in the system operations effectiveness; in fact, a small markers size allows to select a high number of measurement points, but too small markers may be not identified with acceptable accuracy; on the other side, markers of too large size would be simpler to locate, but they would not allow to choose many measuring points, unless to place them at short distances from each other, which would complicate their distinction by depth data processing algorithm.

2.3 The new depth images based monitoring system

This section is related to the detailed system description. The first part will describe the theoretical used models and how depth images are created using such models. Then the marker identification algortihm will be presented.

2.3.1 Pin-hole model

The pin-hole model is the most used mathematical model in images representation. A representation of such a model is given in Figure 2.1, where:

- Image plane is the plane where all the object captured by the camera are projected;
- Optical centre is the point where all the external light ray pass by;
- Principal plane, a parallel plane to the Image plane, which contains the optical centre;
- Principal point is the point placed on the line normal to the plane and passing by the optical centre;
- Focal length, that is the distance between Optical centre and Image plane

For each pixel in an image, the Eq. (2.1) is used to obtain the corresponding 3D information in the camera reference frame.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \\ 1 \end{bmatrix} \tag{2.1}$$

where

- $[u \ v \ 1]$ is the pixel representation of a 3D point; to obtain information about point position in the environment reference frame, Eq. (2.2) is used;
- $[\frac{x'}{z'} \ \frac{y'}{z'} \ 1]$ is the homogeneous coordinates representation of a 3D point in the camera reference system;

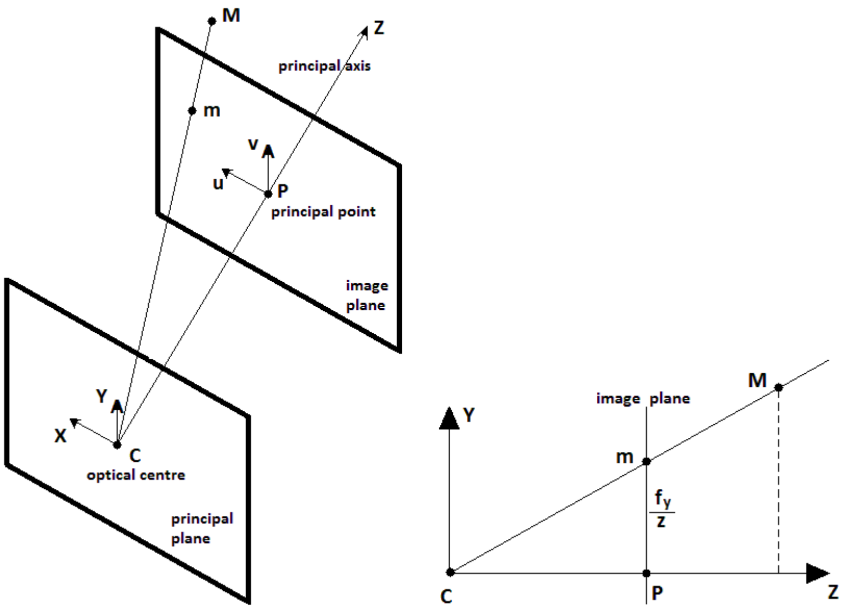


Fig. 2.1. Pin hole model

- f_x, f_y are related to the camera focal length;
- c_x, c_y are the principal point coordinate;

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \underbrace{R}_{R \in \mathbb{R}^{3\times3}} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \underbrace{t}_{t \in \mathbb{R}^{3\times1}} \tag{2.2}$$

The matrix R and the vector t are called "Extrinsic parameters", while f_x, f_y, c_x, c_y are also called "Intrinsic parameters"; all these parameters can be assigned by default or they can be computed by ad-hoc calibration procedures. The above presented model is valid for all values of z except from 0, that means the optical centre cannot projected on any image point. Moreover, the introduced relationship between image points and the corresponding 3D points is linear and does not take into account any distortion due to lens non-linearity. Unfortunately, it is not always possible to use such a simple model, but it is necessary to take into account lens distorsions. The simplest lens distorsion model [64], shown in Eq. (2.3) uses low-order polynomials

$$\begin{aligned} x'' &= x' \cdot [1 + k_1(x'^2 + y'^2)^2 + k_2(x'^2 + y'^2)^4] \\ y'' &= y' \cdot [1 + k_1(x'^2 + y'^2)^2 + k_2(x'^2 + y'^2)^4] \end{aligned} \tag{2.3}$$

where k_1 and k_2 are named "radial distortion coefficients" x' and y' are calculated using the Eq. (2.2), while x'' and y'' can replace x' and y' in the Eq. (2.1).

A more complex model [65] is presented in the Eq. (2.4)

$$\begin{aligned} x'' &= x' \cdot [1 + k_1(x'^2 + y'^2)^2 + k_2(x'^2 + y'^2)^4] + 2 \cdot p_1 \cdot x' \cdot y' + p_2 \cdot (3 \cdot x'^2 + y'^2) \\ y'' &= y' \cdot [1 + k_1(x'^2 + y'^2)^2 + k_2(x'^2 + y'^2)^4] + 2 \cdot p_2 \cdot x' \cdot y' + p_2 \cdot (x'^2 + 3 \cdot y'^2) \end{aligned} \tag{2.4}$$

where p_1 and p_2 are named "tangential distortion coefficients".

2.3.2 Depth images building

The depth images are built by the sensors using stereo-vision principles, based on the assumption that the same scene is observed by two different points of view. There are various techniques to recognize a point in an image; some operate in the pixel domain, by analyzing the correlation between blocks of two different images, while other ones operate in the features domain, looking for elements in the image as segments, angles, etc.. When using a depth sensor, a source of near-infrared light is used to project a pattern of fixed spots onto an object, while another device captures the image reflected by the illuminated object. The same spots are detected in the two different images; each spot is projected on two different points in the corresponding image planes; such a difference is called "disparity" and its value is used to calculate the z-coordinate of the observed point, using the Eq. (2.5); in this equation, x_1 and x_2 are, respectively, the x coordinates of the points p_1 and p_2 represented in Figure 2.2.

$$z = \frac{T \cdot f}{x_1 - x_2} \tag{2.5}$$

2.3.3 Markers identification

The image to be analyzed in our application is always composed by a large smooth region on which there are some small-size areas to identify. Depth images are grayscale images whose pixels intensity gives information about the depth of each observed point. The markers identification is based on some thresholding operation to apply to the image in order to make the marker regions clearer; the result is a black and white image, on which connected components are searched. The center of each connected component is considered as the corresponding marker position (in pixel).

Here the importance of properly place the markers on the physical structure, already introduced in Section 2.2, is much clearer. If it is true that a large number of markers allows to have more measurement points, it is true as well that markers size should not be too small because the sensor accuracy is however not extremely high; moreover, markers should not be placed one at a short distance from the other ones, in order to avoid that two different markers are identified with the same connected component.

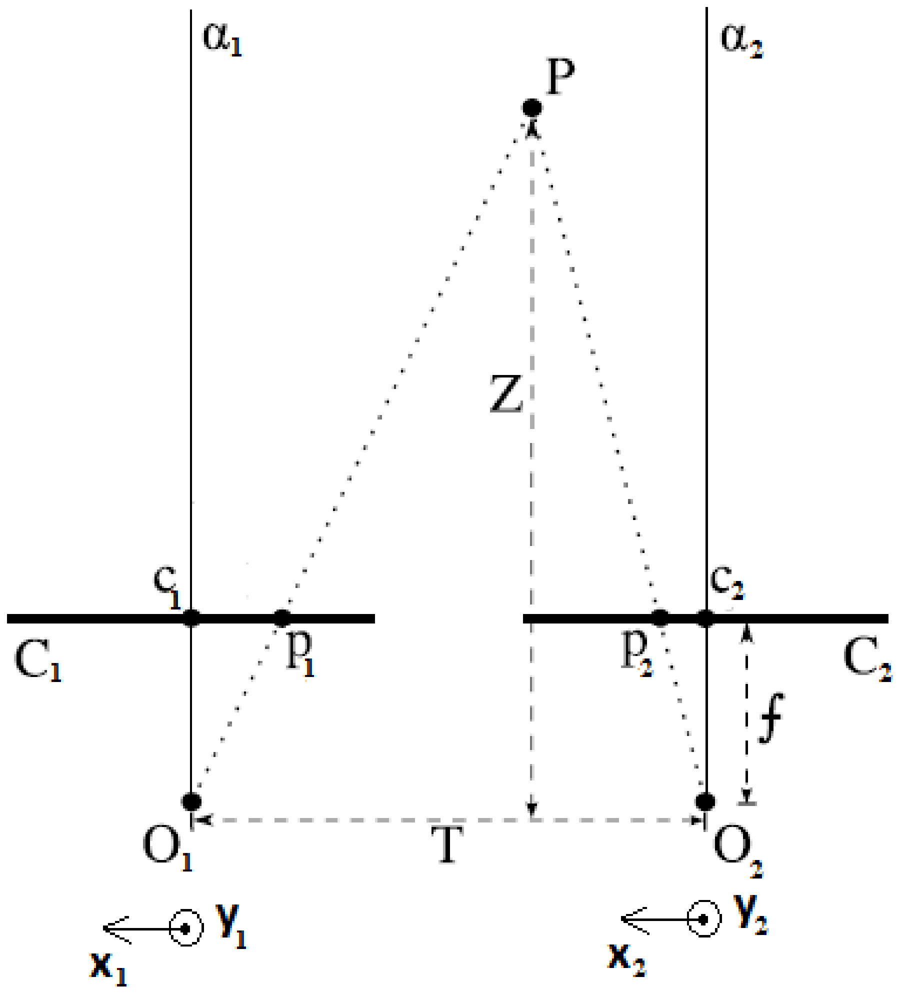


Fig. 2.2. Image disparity calculation

However, to improve the results accuracy, we have choosen to use more than one sensors to identify the same markers, in order to calculate an average of the results of each single sensor. In particular, given a set of N_s sensors, each marker position can be identified as described in the Eq. (2.6)

$$m_{ij} = \frac{1}{N_s}\sum_{k=1}^{N_s} m_{ijk} \tag{2.6}$$

2.4 Simulations and results

In this section, performed experiments and obtained results are presented. Some datasets have been built in order to test our system in different situations; each of them is composed by a series of 640×480 depth images and it is related to a different shaking table test simulation. The same reference frame is assumed for camera and world, so the extrinsic parameters are $R = I_{3\times 3}$ and $t = 0$. For simplicity, it is assumed no distortion, so all distortion coefficients are 0. We consider a wall with dimensions 1.5m $\times$ 2m (see Figure 2.3), where we put a grid of 5×7 markers. 4 kinds of different shaking table tests

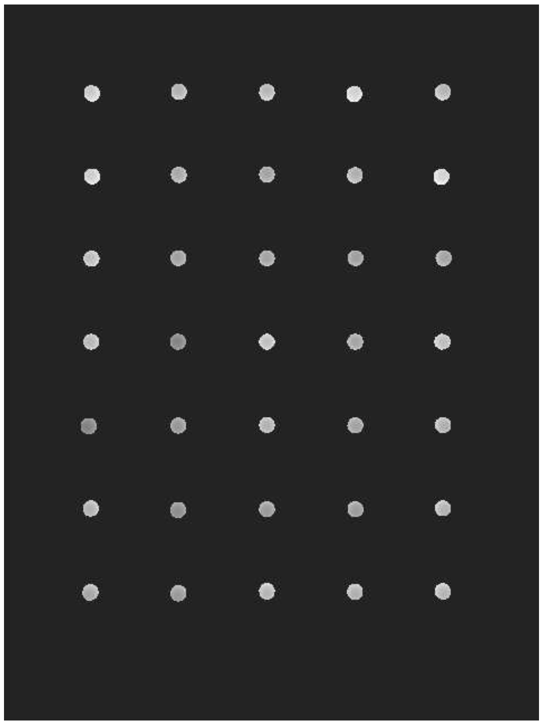


Fig. 2.3. An example of depth map

have been considered for such a structure:

1. Markers displacements along x-axis with frequency $f = 2.5hz$ (Test "wall1-exp1")
2. Markers displacements along z-axis with frequency $f = 4.5hz$ (Test "wall1-exp2")
3. Markers displacements along x-axis snd z-axis, respectively with frequencies $f_x = 2.5hz$ $f_z = 4.5hz$ (Test "wall1-exp3")

4. Markers displacements along x-axis snd z-axis: sum of three modes on x-axis respectively with frequencies $f_{x1} = 2.5hz$, $f_{x2} = 3.8hz$, $f_{x3} = 5hz$ and modal participation factor $p_{x1} = 1$, $p_{x2} = 0.9$, $p_{x3} = 0.8$ and sum of three modes on z-axis respectively with frequencies $f_{z1} = 4.5hz$, $f_{z2} = 6.2hz$, $f_{z3} = 7.6hz$ and modal participation factor $p_{z1} = 1$ $p_{z2} = 0.9$ $p_{z3} = 0.8$ (Test "wall1-exp4")

Results are collected in Tables 2.1, 2.2 and 2.3; each value is expressed in millimeters. In order to not increase the system costs too much, we limited the number of used sensors to 10; a lower number would reduce the costs, but as it can be observed in the above mentioned tables, the accuracy would be also affected. The accuracy differences depending on the sensors number are less clear in the first three tests, where the markers movement was characterized by simple modes, while the last test "wall1-exp4" underlines much more the convenience in using a larger number of sensors.

Table 2.1. x-coordinate error [mm]

Dataset	3 sensors		5 sensors		10 sensors	
	Mean	Variance	Mean	Variance	Mean	Variance
wall1-exp1	0.854	0.689	0.969	0.875	0.871	0.604
wall1-exp2	1.002	1.152	0.862	0.790	0.813	0.629
wall1-exp3	1.022	1.040	0.901	0.731	0.872	0.604
wall1-exp4	1.047	1.061	0.923	0.749	0.878	0.591

Table 2.2. y-coordinate error [mm]

Dataset	3 sensors		5 sensors		10 sensors	
	Mean	Variance	Mean	Variance	Mean	Variance
wall1-exp1	1.147	1.392	1.291	1.752	1.171	1.255
wall1-exp2	1.410	2.150	1.239	1.532	1.189	1.242
wall1-exp3	1.352	2.000	1.203	1.460	1.170	1.234
wall1-exp4	1.375	2.076	1.221	1.518	1.169	1.231

2.5 Concluding Remarks

We presented a new system for structural health monitoring. In particular, such a system can be used for displacement monitoring during tests on shaking tables, an important class of experiments aimed to monitor the behavior

Table 2.3. z-coordinate error [mm]

Dataset	3 sensors		5 sensors		10 sensors	
	Mean	Variance	Mean	Variance	Mean	Variance
wall1-exp1	4.649	12.274	3.545	7.184	2.661	4.024
wall1-exp2	4.451	11.405	3.563	7.219	2.659	4.024
wall1-exp3	4.472	11.472	3.584	7.292	2.661	4.026
wall1-exp4	4.476	11.465	3.559	7.228	2.657	4.030

of certain structures under the action of a seismic force; such experiments involves the use of appropriate measurement systems able to detect the structural response at certain points. Our system is based on a really promising technology in the field of 3D-space monitoring, which uses the so called depth images; using such a technology, it is possible to obtain information about 3D space without needing to process two color images. Another important advantage is that such a system is able to replace classical sensors like strain gauges, accelerometers, etc., usually adopted in displacements monitoring. In this way, the sensing system will be protected from possible damages due to the contact with the under-test structure, especially during destructive tests. Results show that our system is able to estimate the position of some selected points on a structure with a good accuracy. We reckon that such results can be further improved by using some more sophisticated mathematical tools, for example like Kalman Filter; however, in this case, it would be necessary to know the dynamic model of the under test structure.

3

Visual sensing using low cost technologies: a compression procedure

3.1 Introduction

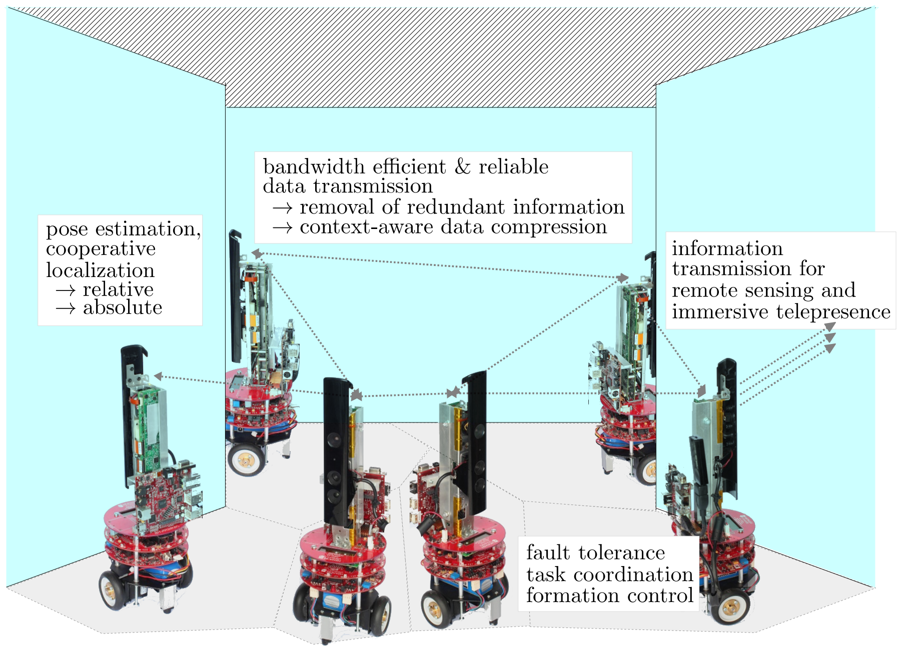


Fig. 3.1. An indoor ping and exploration scenario showing the RGB-D sensor equipped Monash University's experimental mobile robots "eyeBugs". A typical application would be mapping of indoors after a disaster such as the nuclear reactor accident in Fukushima. There are numerous challenges that need to be addressed as summarized on the diagram. In this paper we propose a solution to minimize the bandwidth utilization.

Recently developed RGB-D sensors that are capable of capturing depth information as well as the conventional RGB color data open up possibilities for new applications such as mapping, exploration, or remote sensing especially in environments difficult and/or dangerous to explore like, for example, the nuclear reactors in Chernobyl and Fukushima after the accidents in 1986 and 2011. These applications inevitably require transmission and management of large amounts of information, thus it is useful to adopt strategies for efficiently representing, at the same time, texture and depth information of captured scenes. This is possible using RGB-D sensors, which capture two sets of data: One for texture (RGB image) and another one for depth data (depth image). Separating color data and depth data allows to avoid sending two color images for representing the 3^{rd} dimension. An RGB image is characterized by three grayscale levels, while a depth image is represented by only one grayscale level. Thus, using a depth image instead of a second color image, it is possible to reduce the total amount of data but continuing to represent texture and depth of a scene.

A depth image contains, for each captured pixel, sensor-to-object distance information, so it is very useful to represent accurately many different kinds of surfaces. Such an image can be calculated using the disparity information between two stereo-images. Some early examples of depth-based systems were presented in [66] for videoconferencing, while the ATTEST project [67] shows the guidelines of a general 3DTV system.

Depth images can be obtained by using ToF (Time of Flight) cameras [61] or the above mentioned low-cost RGB-D sensors, such as Kinect and Xtion [62, 63].

The ToF cameras calculate the distance from an object by evaluating the phase delay between two infrared (IR) waves, one emitted by the camera and the second one reflected by the object.

On the other hand, RGB-D sensors are capable of measuring both color and depth information. In addition to capturing color images by using an RGB camera, such sensors include a light source which projects a near-infrared light pattern onto an observed object and an IR camera to capture the reflected pattern. Then, it compares the geometry of the emitted pattern and the received one to reconstruct the depth values. In this way, a depth image can be created without the need for processing two color images.

Depth image information is extremely useful in a wide range of applications, expecially in robotics, gaming or medical diagnostics. In the following paragraphs we briefly provide an overview of a number of typical examples.

In [68] a solution for SLAM problem (Simultaneous Localization and Mapping) is presented. RGB-D sensors are used as an effective alternative to other distance sensors, in order to improve the localization and mapping results. In [69], a SLAM algorithm based on depth images is proposed, where some heuristics are used to overcome the range limit of RGB-D sensors up to 4 meters. In [70], an algorithm for relative position estimation between two different sensors is proposed; it uses depth images of the same scene, captured by different sensors, to estimate the relative sensors position. Depth images and RGB-D sensors are also used for many applications in the medical field. In [71] an algorithm for tracking object is proposed, in order to replace some less efficient systems used in operating rooms, while in [72] and [73] some Kinect-based applications for physical rehabilitation are presented, mainly based on recognizing human gestures.

RGB-D sensors can be used to form networks for various purposes. Some examples can be found in home monitoring applications, in order to help people with limited mobility [74], and natural disasters [75].

RGB-D sensors, when they are used in networks of visual sensors, need to transmit data to remote servers or other cooperating devices (see Figure 3.1). It is clear that the bandwidth usage is a critical issue that points out the necessity of data compression, in order to improve the bit rate or the memory requirements. At the same time, loss of details in the reconstructed image quality can be a critical issue as well. Image compression has reached an advanced level for what concerns RGB images with the well known JPEG2000 [76, 77], PNG [77] and H.264/AVC [77, 78], but the problem of compressing depth images is still an open issue, which attracts a significant number of researchers [79, 80, 81, 82, 83].

A standard image compression algorithm can be applied to depth images treating them as grayscale images. When a sequence of depth images has to be compressed, such a strategy is often combined with a motion compensation algorithm, in order to reduce the redundancy between consecutive images. Some works are focused on two-dimensional compensation methods (2D-BM) [84], in which an image is partitioned into blocks of M $\times$ N pixels and a block matching procedure between two different images is performed, adopting mean square error (MSE) or mean absolute error (MAE) to evaluate the accuracy of the resulting motion vector.

However, such methods may result in inaccurate image reconstruction, as depth images contain information on the distance between the sensor and the observed scene. In [85] a 3D block matching algorithm is proposed, which outperforms 2D-BM only at high bit rates; [86] solves this problem by using an adaptive algorithm which is able to select the best block matching strategy. In each case, the described algorithms are related to situations with a fixed camera. In general, depth values related to an object change over the time, so using a classical compression scheme may be not an effective choice.

The idea presented in this paper is to design a fast and effective compression method for depth images without making any assumptions about camera positions and object shapes.

We first present the problem statement and describe our solution in Section 3.2. Then, in Section 3.3 we describe the algorithm performance evaluation. Finally, in Section 3.4, we present our concluding remarks and discuss future works.

3.2 Proposed compression strategy

In this section, first, the addressed problem is described, then the proposed solution is presented.

3.2.1 Problem statement

The problem is to design an algorithm that minimizes the amount of depth data to be compressed and transmitted, while allowing a good reconstruction of the depth image at the receiver side. That can be performed by designing some functions that are able to interpolate data, in order to avoid transmitting information about each single pixel. It would be convenient to divide an image into blocks before compressing it; in this way, the larger the block size, the simpler will be the necessary model to approximate it. An image can be divided into blocks with or without considering the shape of objects.

Image segmentation is identification of sharp edges and smooth areas in an image, in order to characterize each object in a scene. For example, in [87], the authors propose a graph-based image segmentation, where each segment is a connected component of a graph in which vertices correspond to pixels and edges are weighted by a measure of the dissimilarity between two neighboring vertices, as it might be the absolute intensity difference or the Euclidean distance between the two points corresponding to the two vertices.

The result of applying a segmentation algorithm to an image is a set of partitions, each one representing a different smooth region in the observed scene. Each partition of such a set can be used to design a function able to give an accurate reconstruction of the represented region in the image. Starting from the depth pixel values, a set of 3D points can be easily computed for each partition by exploiting the pinhole model and the intrinsic camera parameters [64]; then, a model which accurately fits such points can be calculated; at the end, to compress and transmit all models to the receiver, a significantly lower amount of information is required with respect to transmitting the whole uncompressed depth image, but the receiver will be able in any case to reconstruct the original image with high fidelity.

However, depending on the scene, a segmentation approach simply based on identifying different objects in an image, like the one presented in [87], may not be really suitable to a procedure of depth image compression based on a division in blocks. The reason is that objects usually have irregular shapes, so, to compress and transmit information about a partition, it would be necessary to characterize also the perimeter of each object, which is not so obvious and, moreover, it might take much more time than the necessary one to have a significant result in depth approximation. An alternative way is to insert each segment in a bounding box; unfortunately, for many times objects in a scene are placed so that the above mentioned approach leads to have many blocks which include pixel related to different objects; this leads to an unavoidable overlapping between different fitting models. A typical example is shown in

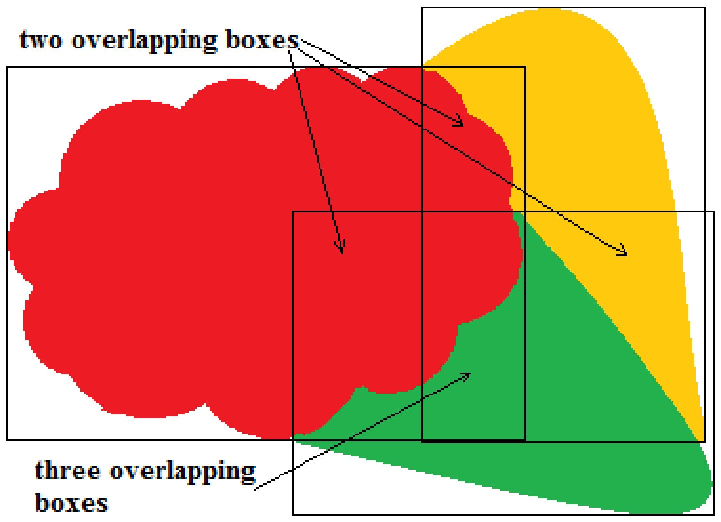


Fig. 3.2. Overlapping boxes

Figure 3.2; three different objects are bounded by a box, but there are image pixels shared between two or three boxes. In [82], the authors try to overcome this problem by performing an additional partitioning of each segment in blocks of regular size, in order to improve the accuracy of the fitting strategy; however, this solution may significantly affect the computation time.

To avoid block overlaps, the solution adopted in the present paper is to perform a quadtree segmentation of the image to be compressed, proposed in [88]. This procedure is suitable for square images; unfortunately, the images acquired by the Kinect Sensor are not square; in this case, one of the patterns presented in Figure 3.3 can be used in order to consider all the original image pixels. In this way, a 480×640 image is divided into one square of 480×480 and three squares of 160×160; then, the segmentation procedure is repeated for each square.

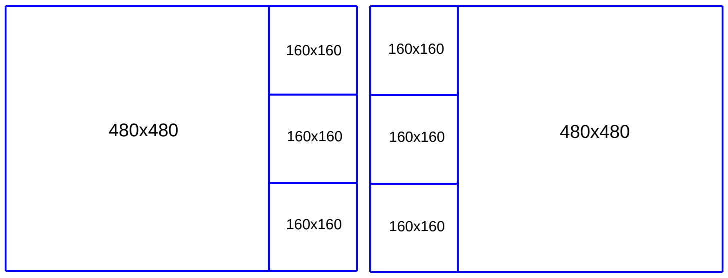
The quadtree segmentetion procedure, adapted to the images captured by the Kinect, is presented in Algorithm 2, while Figure 3.4 shows some examples of its application. The threshold d_{max}, used to decide whether or not divide a depth block, is calculated by the algorithm; the input parameter is d_{perc} is a percentage value between 0 and 1; in this way, the algorithm can be easily adapted to images represented by different number of bits (see line 2.10 in the Algorithm 2).

3.2.2 Surface-Based Blocks Approximation Algorithm (SBBA)

Some works about image compression using fitting models make the hypothesis that most objects surfaces can be approximated by a plane [82, 83]; this

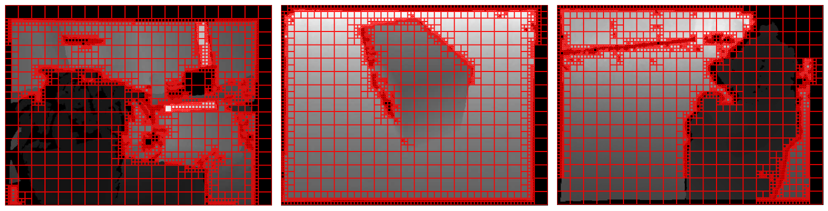
Algorithm 2 The quadtree segmentation algorithm

```
 1: procedure SegmentationProcedure($I$,$N$,$d_{perc}$,$B_{sm}$,$B_{sM}$)
 2:     Parameters - I:depth image, N: number of bits to represent I, $B_{sm}$: minimum block side size, $B_{sM}$: maximum block side size, $d_{perc}$: constant value between 0 and 1.

 3:     $L = \emptyset$, $B = \emptyset$
 4:     for $i \leftarrow 1, 4$ do
 5:         $l_k = \{j_{sk}, j_{ek}, i_{sk}, i_{ek}\}$   ▷ Label identified by 4 values delimiting a pixels block
 6:         add $l_k$ to $L$
 7:         $b_k$=blockFromLabel(I,$l_k$)   ▷ $b_k$ is a subset of depth pixels
 8:         add $b_k$ to $B$
 9:     end for   ▷ apply Pattern 1 or Pattern 2 to the depth image I
10:     $d_{max} = d_{perc} \cdot (2^N - 1)$;
11:     repeat
12:         for all depth blocks $b_k \in B$ do
13:             if toBeDivided($b_k$,$d_{max}$,$B_{sm}$,$B_{sM}$)==true then
14:                 $l_{k1} = \{j_{sk}, \frac{j_{sk}+j_{ek}}{2}, i_{sk}, \frac{i_{sk}+i_{ek}}{2}\}$
15:                 $l_{k2} = \{j_{sk}, \frac{j_{sk}+j_{ek}}{2}, \frac{i_{sk}+i_{ek}}{2} + 1, i_{ek}\}$
16:                 $l_{k3} = \{\frac{j_{sk}+j_{ek}}{2} + 1, j_{ek}, i_{sk}, \frac{i_{sk}+i_{ek}}{2}\}$
17:                 $l_{k4} = \{\frac{j_{sk}+j_{ek}}{2} + 1, j_{ek}, \frac{i_{sk}+i_{ek}}{2} + 1, i_{ek}\}$
18:                 delete $l_k$ from $L$
19:                 add $l_{k1}$, $l_{k2}$, $l_{k3}$ and $l_{k4}$ to $L$
20:                 delete $b_k$ from $B$
21:                 add $b_{k1}$, $b_{k2}$, $b_{k3}$ and $b_{k4}$ to $B$   ▷ Replace the block $b_k$ with the 4 equal-sized blocks in which $b_k$ is divided
22:             end if
23:         end for
24:     until $\exists b_k \in B$ : toBeDivided($b_k$,$d_{max}$,$B_{sm}$,$B_{sM}$)==true
25:     return $L, B$
26: end procedure

27: procedure toBeDivided($db$,$d_{max}$,$B_{sm}$,$B_{sM}$)   ▷ $db$ is a depth block
28:     calculate $Mdv = max(db)$ , $mdv = min(db)$;
29:     return $[(Mdv - mdv) > d_{max}$ & $size(db) > B_{sm})]$ || $size(db) > B_{sM}$
30: end procedure

31: procedure blockFromLabel(I,l)
32:     $b = \{I(j,i) : l(1) \leq j \leq l(2), l(3) \leq i \leq l(4)\}$
33:     return b
34: end procedure
```



(a) Pattern 1 (b) Pattern 2

Fig. 3.3. Patterns for quadtree image segmentation



(a) Dataset "Plant" (b) Dataset "Cabin" (c) Dataset "Teddy3"

Fig. 3.4. Some examples of quadtree image segmentation

assumption allows to save a considerable amount of bandwidth, but it is too restrictive because a generic object has a 3D shape. In this context, we hypothesize that using a higher-order polynomial model improves significantly the results; we will show it in Section 3.3.

The SBBA algorithm (see 3) makes no assumption about the environment structure, in order to design a general compression procedure for different kinds of scenes.

The value 3 as maximum degree for the model variables x and y has been chosen after some evaluating experiments. Indeed, a greater degree increases significantly the amount of data to be compressed and it does not match always to a greater accuracy of the reconstructed image. Once the fitting model is calculated, model data have to be transmitted. There are two possibilities:

- to compress and transmit model coefficients $\{a_{ij} \mid i \in [1\ 3]\ ,\ j \in [1\ 3]\}$: when choosing this first option, it is necessary to send other additional data in order to characterize the boundaries of the block approximated by the considered model; for example, given the set of 3D-points corresponding to a generic depth-block k, the label defined at the line 2.5 of the Algorithm 2 can be modified as follows:

$$l_k = \{j_{sk}\ , j_{ek}\ , i_{sk}\ , i_{ek}\ , y_{mk}\ , y_{Mk}\ , x_{mk}\ , x_{Mk}\ \} \tag{3.1}$$

Algorithm 3 The SBBA (Surfaces Based Blocks Approximation) algorithm

```
1: Parameters - I:depth image, N: number of bits to represent I, $B_{sm}$: minimum
   block side size, $B_{sM}$: maximum block side size, $d_{perc}$: constant value between 0
   and 1.

2: procedure SBBA($I$,$N$,$d_{perc}$,$B_{sm}$,$B_{sM}$)
3:     $\{L, B\}$=SegmentationProcedure($I$,$N$,$d_{perc}$,$B_{sm}$,$B_{sM}$)
4:     for all $b_k \in B$ do
5:         $points_{b_k} = RandomSample(b_k, n_p)$     ▷ Calculate a set of 3D-points
   starting from a set of random values in $B$;
6:         Calculate a 2-variables polynomial model to approximate each depth
   value as $\hat{z} = f(x, y) = \sum_{j=0}^{3} \sum_{i=0}^{3-j} a_{ji} \cdot x^i \cdot y^j$
7:     end for
8: end procedure
```

where $[y_{mk}, y_{Mk}]$ and $[x_{mk}, x_{Mk}]$ are, respectively, the delimiters of the y-coordinates and of the x-coordinates related to the space points obtained from the block k. In this way, at the decoder side, it is easy to define a proper set of x and y values to be used with the Eq.6 to reconstruct the depth image; for example, the simplest choice is to define a set of equally spaced x values between x_{mk} and x_{Mk} and a set of equally spaced y values between y_{mk} and y_{Mk};

- to choose some points and transmit the points coordinates, in order to reconstruct the surface at the decoder side: This second solution would considerably simplify the choice of the set of x and y values at the decoder side but, since the fitting model is not linear, the obtained model would not be the same one obtained at the encoder side, so the image accuracy would be reduced.

We adopt the first solution in this work; then, to transmit the model coefficients, we use the Huffman coding; in addition to being an optimal coding procedure, it can be easily adapted to the trasmission of the model coefficients [76]. When transmitting a set of numbers, Huffman coding associates a string of bits to each numerical value. The value with the highest number of occurrences in the set to be coded will be associated to the shortest string of bits; on the contrary, the value with the lowest number of occurrences will be associated to the longest string of bits.

In this way, after taking a training set of depth images in a scene, the fitting blocks are calculated and the related coefficients are used to build a lookup table containing a certain number of pairs $\langle value, stringofbits \rangle$, to be adopted to compress other depth images of the same scene. If it was necessary to code a value without any corresponding string of bits, it would replaced by the nearest one among the values in the previously built table.

3.2.3 Fast Hole Filling Algorithm

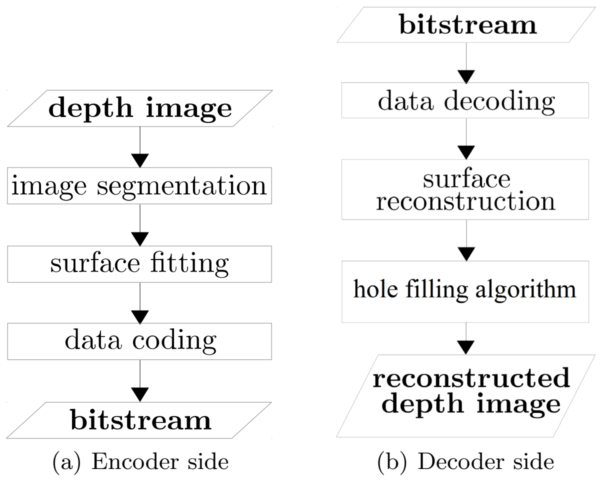


Fig. 3.5. Operational overview of the compression procedure.

Different approximations during the segmentation/partitioning/compression/ decompression processes may lead to some loss of information. To face this issue, a hole filling algorithm has been designed to be used at the decoder side, in order to further increase the PSNR.

A complete representation of our compression procedure is given in Figure 3.5.

In the literature there exist different kinds of hole-filling algorithms. As shown in [81], the simplest algorithm consists in choosing simply the nearest pixel, in order to shade the object shape near the hole to be filled. This algorithm is not suitable for scenes with a large number of different objects or objects with irregular shapes.

Other algorithms presented in [89] and [90] are based on interpolations of reconstructed blocks; these algorithms allow surely to obtain more accurate results, but the required computational cost is very high.

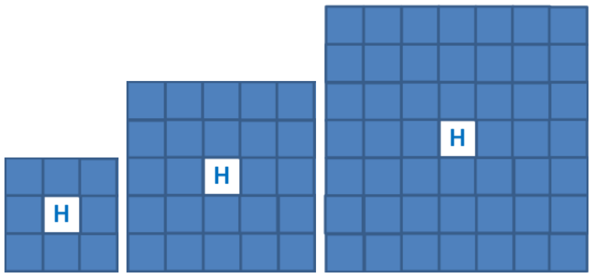
In the present paper, a simple algorithm has been designed, based simply on averaging a certain number of pixels surrounding each hole. In particular, given a depth image I and an integer w_s, a pixel $I(j,i)$ with 0-value in I can be replaced as indicated in the Eq.3.2.

$$I(j,i) = \frac{1}{n_z}\sum_{k=1}^{n_z} p_{jik} \quad , \quad p_{ijk} \in P_{ij} \tag{3.2}$$

where the set P_{ji} containts n_z depth pixel values and it is defined as

$$P_{ji} = \{I(j_h, i_h) : I(j_h, i_h) \neq 0, j - w_s \leq j_h \leq j + w_s, i - w_s \leq i_h \leq i + w_s\} \tag{3.3}$$

As specified in the Eq. (3.3), only non-zero pixels are considered, because a zero-pixel means lack of information, thus it cannot be considered a reliable contribution to the hole-filling procedure.



(a) w_s=1 (b) w_s=2 (c) w_s=3

Fig. 3.6. Window size in the hole-filling algorithm

3.3 Performance Evaluation

In this section, we present the performance evaluation of the method. Datasets provided by the Computer Vision Group in Technical University of Munich [91] have been used to carry out the experiments. Different scenes with different kinds of objects have been considered; each of them is composed by a series of 480 × 640 color and depth images, acquired using a RGB-D sensor. For each dataset, the camera parameters are provided. As an accuracy evaluation criterion, the PSNR is used, i.e. each reconstructed image is compared with the original one. The quadtree segmentation procedure parameters are $d_{max} = 0.3$, $B_{sm} = 8$ and $B_{sM} = 32$.

3.3.1 Bandwidth Usage Analysis

We show now that an image blocks approximation based on a generic polynomial surface leads to a more efficient usage of the available bandwidth with respect to using a planar surface. First of all, the results about bandwidth saving are presented, while the subsequent part is related to the performances analysis of the hole-filling algorithm.

Obviously, the total amount of transmitted bits for each single block is greater when using surface-approximation, as the model degree is greater in that case; in particular, a plane can be characterized by 4 coefficients, while a third-order polynomial surface is represented by 10 coefficients. However, a greater bandwidth consumption for transmitting a single block may still correspond to a gain in the reconstructed whole image accuracy and/or to an improvement in total bandwidth required to transmit the whole image; in addition to Table 3.1, which shows that there is a significant amount of saved bandwidth, Table 3.2 presents the ratio between the total number of bits used to transmit image information and the number of reconstructed pixels; the number of consumed bits per single pixel is always lower when using surface-based blocks approximation; in particular, there is a bit-per-pixel reduction up to 21% (dataset "Dishes").

Table 3.1. Percentage of saved bandwidth

Dataset	Value
Plant	15.63 %
Dishes	26.64 %
Coke	21.65 %
Cabin	4.68 %
Office	1.50 %
Teddy3	11.08 %

Table 3.2. Bits-per-pixel

Dataset	P:Planar fitting	S:Surface fitting
Plant	1.58	1.34
Dishes	1.51	1.19
Coke	1.09	0.88
Cabin	0.62	0.61
Office	0.46	0.44
Teddy3	0.73	0.67

Moreover, the computational cost required by the use of surface-based approximation instead of plane-based approximation is not increased significantly, as shown in Table 3.3; as it can be observed, the difference is always about some milliseconds, compared to a good gain in bandwidth usage; moreover, the computation time variance is very low.

Concerning the reconstructed image accuracy, Table 3.4 presents a comparison of PSNR values obtained for each dataset, using planar approximation and surface-based approximation. As the Table 3.4 indicates, there is a slight accuracy loss, which is never greater than 10%. Actually this is a limit of our algorithm, which needs to be investigated more in depth, even though these values depend also on the quality of the original images. Some images have

Table 3.3. Fitting time comparison

Dataset	Planar approximation		Surface approximation	
	Mean	Variance	Mean	Variance
Plant	4.4×10^{-2}	2.80×10^{-5}	4.6×10^{-2}	1.06×10^{-5}
Dishes	4.0×10^{-2}	1.42×10^{-5}	4.8×10^{-2}	6.19×10^{-6}
Coke	4.1×10^{-2}	2.42×10^{-5}	4.4×10^{-2}	9.18×10^{-6}
Cabin	4.1×10^{-2}	8.74×10^{-5}	4.2×10^{-2}	1.13×10^{-5}
Office	4.0×10^{-2}	8.40×10^{-6}	4.6×10^{-2}	1.20×10^{-4}
Teddy3	3.9×10^{-2}	4.80×10^{-6}	4.6×10^{-2}	5.61×10^{-6}

many pixels equals to 0, that means lack of information; indeed, the worst result has been obtained when using the dataset "Teddy3", which is the one with the biggest number of holes.

Table 3.4. PSNR comparison

Dataset	Planar approximation		Surface approximation	
	PSNR	No of holes	PSNR	No of holes
Plant	16.37	159 K	15.63	157 K
Dishes	14.35	194 K	12.93	202 K
Coke	15.46	207 K	14.29	210 K
Cabin	16.80	72 K	15.38	79 K
Office	19.76	73 K	19.61	67 K
Teddy3	17.55	136 K	16.21	143 K

The last part of the present subsection shows the effectiveness of the adopted hole-filling algorithm. Figure 3.7 shows that it is always possible to further increase the PSNR value by choosing $w_s > 1$, that means to select the size of the region around a hole which is supposed to be related to the same object in the scene. Obviously, a value of w_s much greater than 1 leads probably to include pixel related to other objects. Lastly, Figure 3.8 to 3.13 show a visual comparison between the original images and the reconstructed ones.

3.3.2 Compression ratio analysis

Section 3.3.1 has presented the ability of the SBBA algorithm to efficiently use the available bandwidth, even though this leads to a slight decrease in image accuracy.

In this second part of the results presentation it is shown that our algorithm is capable to achieve very high compression ratios, compared to other

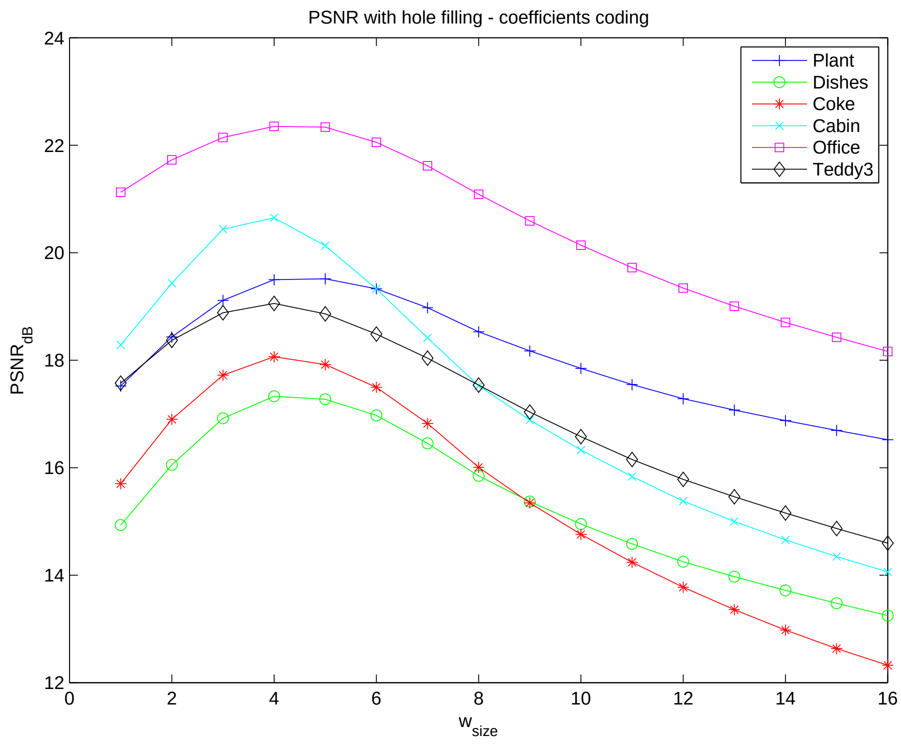
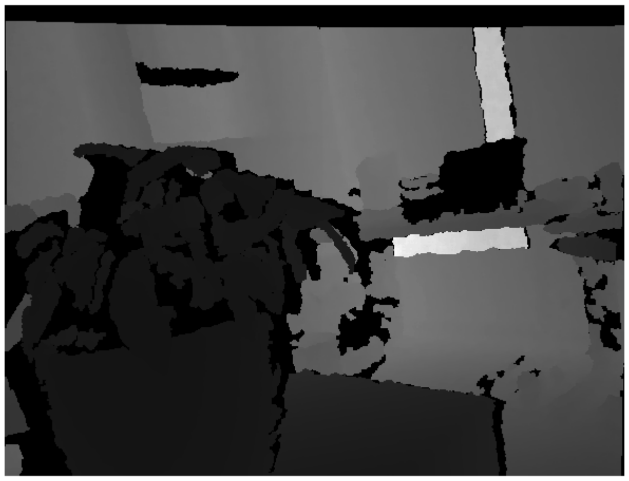


Fig. 3.7. PSNR vs. window size (w_s)

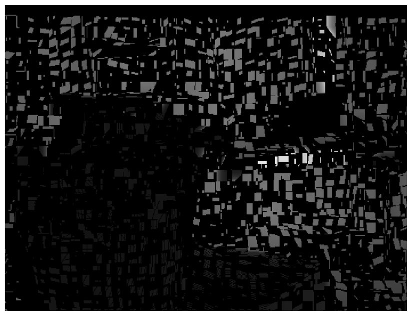
standard compression algorithms. Each original image is a 480 × 640 depth image and each pixel is represented by a 16-bits integer, so the uncompressed image size is 600KB. The compression ratio is defined as

$$cr = 1 - \frac{bits_c}{bits_o} \tag{3.4}$$

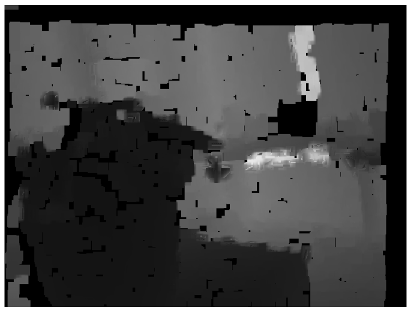
where $bits_c$ and $bits_o$ are, respectively, the number of bits required to transmit the compressed image and the number of bits required to transmit the original image. Results are summarized in Table 3.5: as it can be observed, polynomial-models based depth compression leads to obtain a compression ratio of an order of magnitude bigger than standard compression algorithms; more in detail, a surface-based blocks approximation leads to the best results for each different dataset.



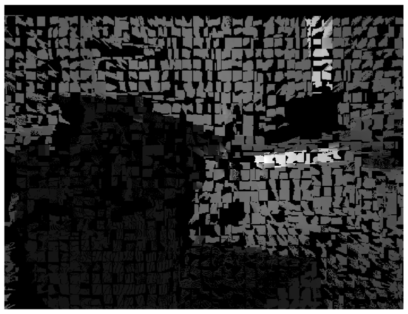
(a) Original image



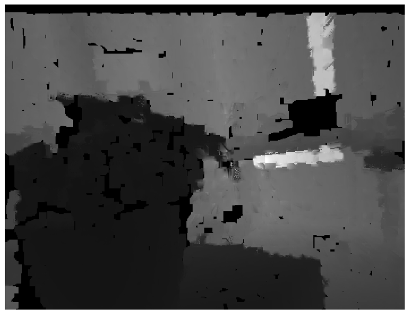
(b) Reconstructed image without hole-filling, planar blocks approximation



(c) Reconstructed image with hole-filling, planar blocks approximation

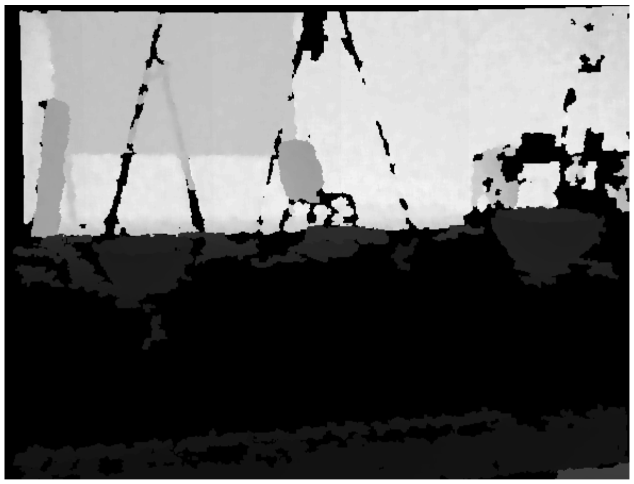


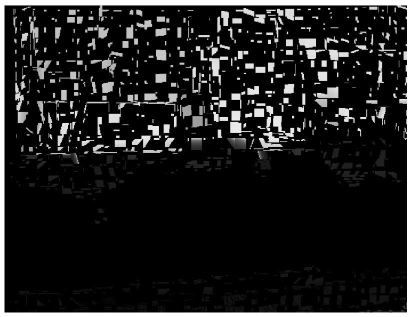
(d) Reconstructed image without hole-filling, surface blocks approximation



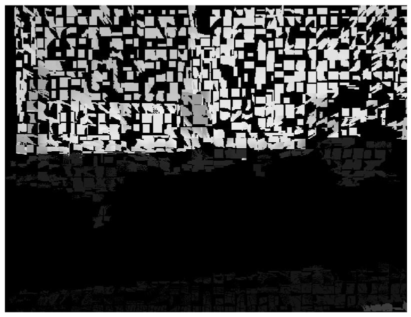
(e) Reconstructed image with hole-filling, surface blocks approximation

Fig. 3.8. Image reconstruction results : Dataset "Plant"


(a) Original image


(b) Reconstructed image without hole-filling, planar blocks approximation

(c) Reconstructed image with hole-filling, planar blocks approximation


(d) Reconstructed image without hole-filling, surface blocks approximation

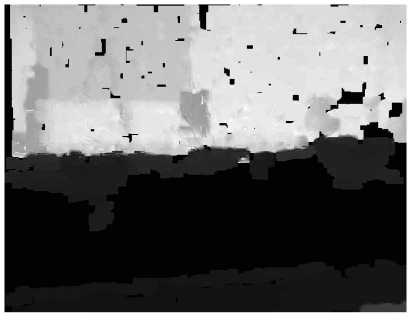
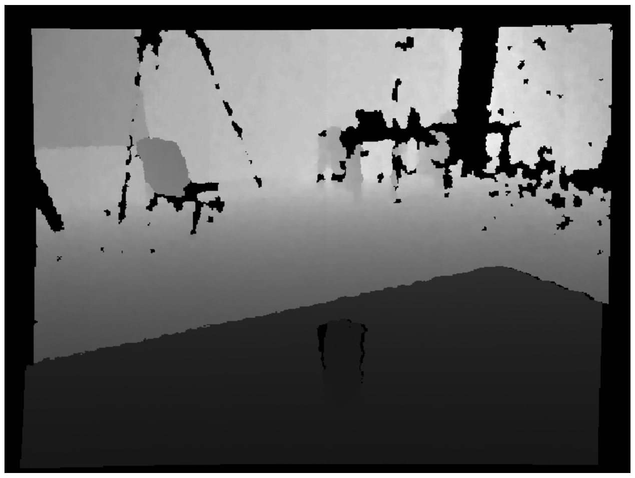
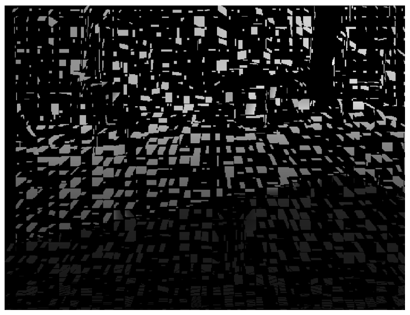
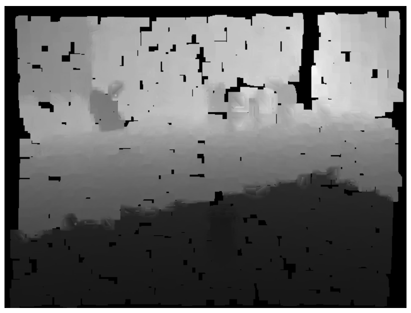
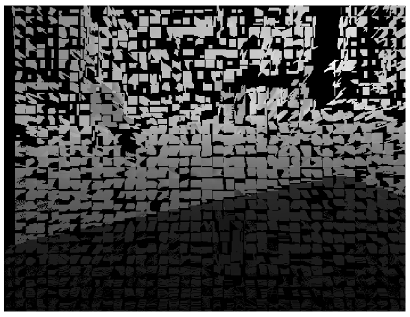

(e) Reconstructed image with hole-filling, surface blocks approximation

Fig. 3.9. Image reconstruction results : Dataset "Dishes"


(a) Original image


(b) Reconstructed image without hole-filling, planar blocks approximation


(c) Reconstructed image with hole-filling, planar blocks approximation


(d) Reconstructed image without hole-filling, surface blocks approximation

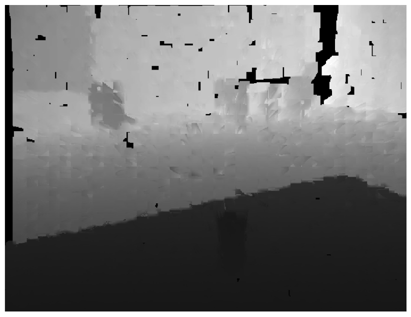
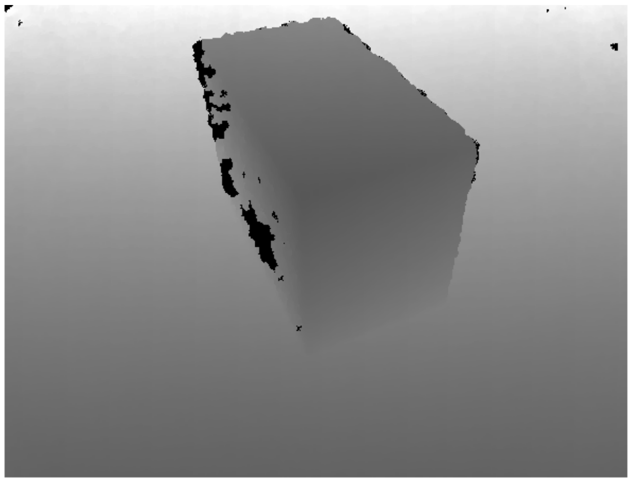
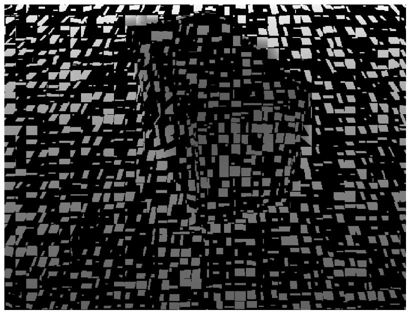
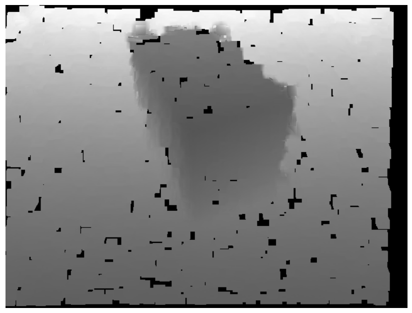
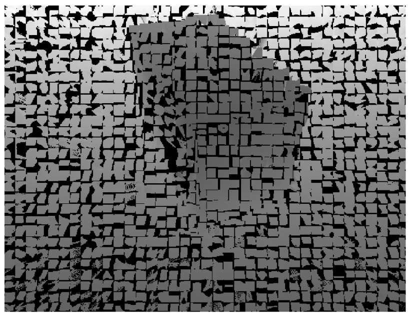

(e) Reconstructed image with hole-filling, surface blocks approximation

Fig. 3.10. Image reconstruction results : Dataset "Coke"


(a) Original image


(b) Reconstructed image without hole-filling, planar blocks approximation


(c) Reconstructed image with hole-filling, planar blocks approximation


(d) Reconstructed image without hole-filling, surface blocks approximation

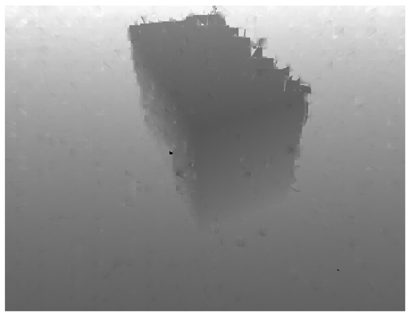
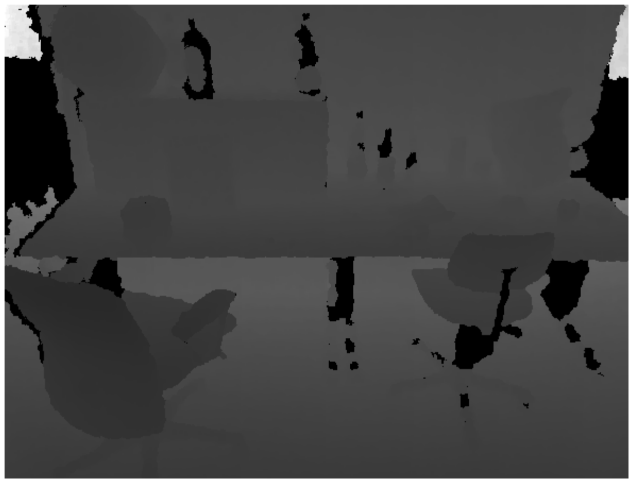
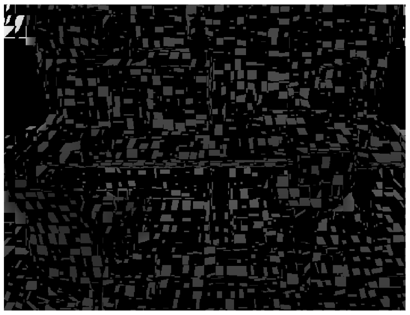

(e) Reconstructed image with hole-filling, surface blocks approximation

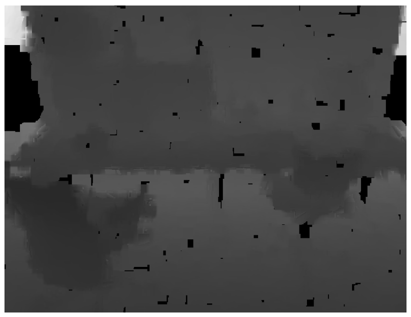
Fig. 3.11. Image reconstruction results : Dataset "Cabin"



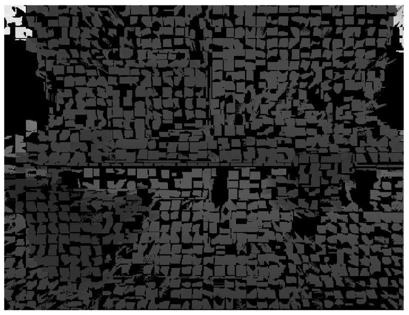
(a) Original image



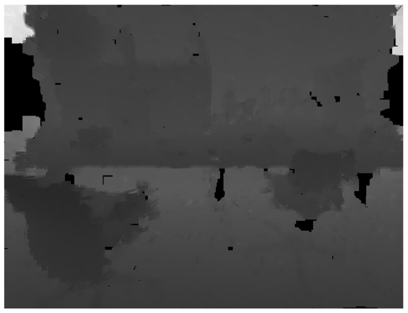
(b) Reconstructed image without hole-filling, planar blocks approximation



(c) Reconstructed image with hole-filling, planar blocks approximation

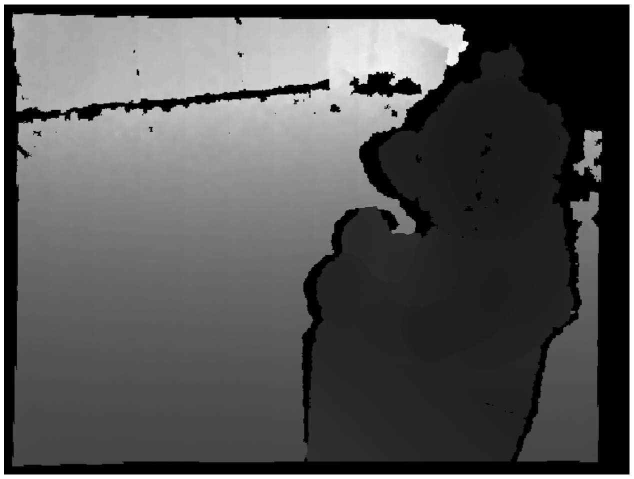


(d) Reconstructed image without hole-filling, surface blocks approximation

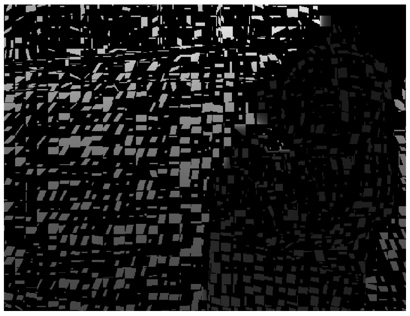


(e) Reconstructed image with hole-filling, surface blocks approximation

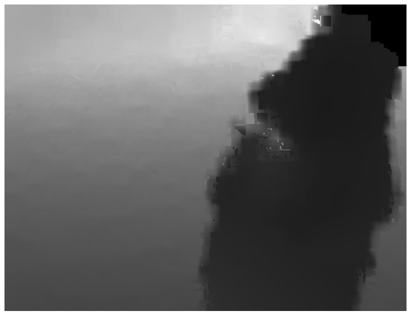
Fig. 3.12. Image reconstruction results : Dataset "Office"



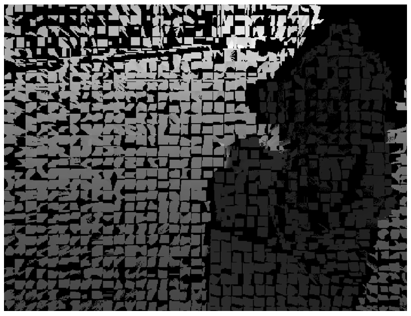
(a) Original image



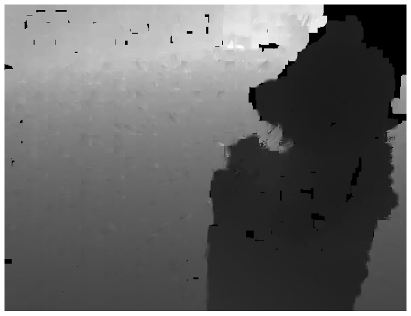
(b) Reconstructed image without hole-filling, planar blocks approximation



(c) Reconstructed image with hole-filling, planar blocks approximation



(d) Reconstructed image without hole-filling, surface blocks approximation



(e) Reconstructed image with hole-filling, surface blocks approximation

Fig. 3.13. Image reconstruction results : Dataset “Teddy3”

Table 3.5. Compression ratio comparison - PBBA: Plane Based Blocks Approximation, SBBA: Surface Based Blocks Approximation

DATASET	JPG	PNG	PBBA	SBBA
Plant	4.38	3.77	20.93	24.51
Dishes	6.06	4.79	28.62	39.01
Coke	7.30	5.40	44.94	57.37
Cabin	5.93	5.05	33.89	35.55
Office	6.15	4.92	45.38	46.07
Teddy3	5.73	4.67	39.49	44.41

3.4 Concluding Remarks

We have addressed the problem of efficiently compressing and transmitting depth image. The SBBA algorithm has been designed to improve the use of the available bandwidth.

The choice to approximate the scene objects surface by polynomial functions allows to make no assumptions about environment structure, making the algorithm usable with a large variety of scenes.

Moreover, at the decoder-side, a fast, but very effective hole-filling algorithm has been designed to further increase the PSNR of the depth image reconstructed using the decoded data.

Several experiments have been performed on different datasets, in order to test the algorithm for various kinds of scene. Results show that the proposed algorithm is able to overcome the performances of other compression algorithms, such as the ones making the hypothesis that objects have planar surfaces [82, 83].

Different kinds of results have been obtained; first of all, it is shown that our surface-based algorithm is capable to reduce, sometimes significantly, the necessary bandwidth to transmit a depth image and the number of used bits-per-pixel; moreover, it is testified that our algorithm is able to achieve high compression ratios, overcoming the performances of other standard compression algorithms.

The main limit of our algorithm is that the PSNR values are not very high yet; then, as future work, we are trying to improve the algorithm effectiveness looking for other suitable mathematical models to be used to approximate depth blocks; the goal is to improve the reconstructed image accuracy while keeping the algorithm ability to efficiently use the available bandwidth.

Moreover, having an effective depth-data compression procedure open up interesting possibilities related to 3D objects reconstruction. A team of mobile robots, each one equipped with an RGB-D sensor, can be used to acquire different views of an object or a scene, and the compression algorithm would be useful for reducing the time required to exchange data necessary to reconstruct the considered object.

Thesis Conclusions

The present work has been focused on developing low-cost systems and methodologies for improving building security, an important task which involves all the building life, from the design to the daily service.

The building security systems may answer to two different kinds of need: "prevention" and "care". Prevention is related to all studies and activities aimed to improve building security under construction; we could mention structural engineering, which is focused on developing mathematical models for characterizing structural behaviour. On the other side, care includes all the activities performed on the structure under regular service, e.g. design of dynamic dampers and all what is used as support to the main structure.

Computer science and electronic engineering can be useful in both cases. Computer and other sensors systems can be used as useful supports to acquire and analyze data for preliminary diagnosis and in-situ investigations. Moreover, the high level achieved by technology nowadays allows to develop also low-cost systems, while keeping a good accuracy, thus letting the user a larger variety of tools to choose, depending on the needs.

For a complete discussion, both prevention and care have been covered. The common point has been looking for cheap solutions, using different kinds of sensors currently available on the market.

The Chapter 1 has been focused on SLAM problem; the idea has been to develop solutions to a common problem in context of building security, which is environment exploration expecially in dangerous situations, like after a natural disaster. Different aspects have been covered:

- In Section 1.6, only the localization problem has been studied; it has been shown that different version of the Extended Kalman Filter working in parallel, each one based on different kinds of measurements, can be suitably combined in order to emphasize the qualities and overcome the defects of each sensor. Using the proposed MKF, two Kalman gains have to be computed and preliminary theoretical studies on the gains optimal values have been performed. The algorithm has been contrasted with other Extended Kalman filters, based on only on board sensors (NEKF), only out of board sensors (OEKF), both on board and out of board sensors (EKF). The obtained results are very encouraging.

- In Section 1.7, some SLAM algorithm in a totally unknown environment have been described. A first solution, proposed in Section 1.7.1, has provided very good localization results, but it has a very high computational cost to be used for real applications. Such limitation has been overcome in Section 1.7.2, where some ad-hoc heuristics have been introduced; the algorithm computation time has been significantly reduced without considerably affecting the localization results. Finally, in Section 1.7.4, a SLAM strategy involving a team of mobile robots has been proposed; each robot is able to build an its own environment map, but the additional possibility to exchange data among robots allows each single team member to reconstruct also environment parts not explored yet.
- In Section 1.8, a new strategy for a selective sensors activation strategy has been described, showing that the use of all the acquired measurements is not always the best choice; thus, this strategy allows to obtain good localization results while optimizing the energy required to power the robots sensors, usually provided by batteries with limited capacity.

The Chapter 2 is dedicated to developing a new displacements monitoring system, using a new low-cost sensor which is attracting more and more interest in research. Such a system can be used for preliminary analysis, in particular for displacements monitoring during tests on shaking-table, an important class of experiments performed to study the behaviour of a structure under the action of a seismic force. The idea has been to develop a system based on the RGB-D sensor, a new kind of visual sensor able to acquire also 3D information about the observed scene (depth map). It has to be specified that the current used sensors accuracy is not so high to compare it with high resolution cameras, but the presented technology is really promising and the presented monitoring system may become a valid alternative to classical displacement sensors.

Finally, in Chapter 3, a new method for compressing data acquired by an RGB-D sensor has been proposed. The proposed algorithm has been designed on purpose for depth map compression, showing its ability to overcome the performance of other standard image compression algorithms.

Future Works

The present thesis has been focused more on developing methodologies and algorithms for scientific problems like SLAM, accurate sensing, data compression. For what concerns SLAM problem, it may be further investigated looking for other mapping models to test.Then, the next work may be the implementation of a complete control scheme for mobile robots, including localization and adaptive path planning. Also, it would be interesting to adapt the implemented algorithms to measurements acquired by other sensors, for example like the Microsoft Kinect, here used for other purposes.

About the SLVision system, it has been specifically designed and can be considered as alternative to classical sensors, in order to avoid the direct contact with the under-test structure. In this way, the sensing system is protected from possible damages during destructive test. However, other kinds of monitoring systems may be replaced by a low-cost alternative, like laser-scanner, widely used in cultural heritage monitoring.

The last part of the thesis, concerning the compression of the depth images, describes one of the first algorithm for this purpose, and may be extended looking for more accurate interpolation models to be adapted to the approximation of surfaces.

References

[1] S.Protasoni. *Figini e Pollini.* Ed. by Milano: Mondadori Electa. 2010.

[2] F.Mondada, E.Franzi, A.Guilgnard, and J.Nicoud. "Khepera: The Optimal Tool for Development in Mobile Robotics". In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on.* Vol. 3. May 1995, pp. V14–. DOI: 10.1109/ROBOT.1995.525757.

[3] D.A.Christian. *The Mt. Erebus Explorer Control System.* Ed. by Carnegie Mellon University. The Robotics Institute, Carnegie Mellon University,5000 Forbes Avenue Pittsburgh, 1993.

[4] J.E.Bares and D.S.Wettergreen. "Dante II: Technical Description, Results, and Lesson Learned". In: 18 (July 1999), pp. 621–649.

[5] E.D.Dickmanns, R.Behringer, D.Dickmanns, T.Hildebrandt, M.Maurer, F.Thomanek, and J.Schiehlen. "The seeing passenger car 'VaMoRs-P'". In: *Intelligent Vehicles '94 Symposium, Proceedings of the.* Oct. 1994, pp. 68–73. DOI: 10.1109/IVS.1994.639472.

[6] Randall C. Smith and Peter Cheeseman. "On the Representation and Estimation of Spatial Uncertainty". In: *The International Journal of Robotics Research* 5.4 (1986), pp. 56–68. DOI: 10.1177/027836498600500404. eprint: http://ijr.sagepub.com/content/5/4/56.full.pdf+html. URL: http://ijr.sagepub.com/content/5/4/56.abstract.

[7] Y.Misono, Y.Goto, Y.Tarutoko, K.Kobayashi, and K.Watanabe. "Development of laser rangefinder-based SLAM algorithm for mobile robot navigation". In: *SICE, 2007 Annual Conference.* Sept. 2007, pp. 392–396. DOI: 10.1109/SICE.2007.4421015.

[8] D.Joubert, W.Brink, and B.Herbst. "Pose uncertainty in occupancy grids through Monte Carlo integration". In: *Advanced Robotics (ICAR), 2013 16th International Conference on.* Nov. 2013, pp. 1–6. DOI: 10.1109/ICAR.2013.6766589.

[9] M.Montemerlo, S.Thrun, D.Koller, and B.Wegbreit. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: *In Proceedings of the AAAI National Conference on Artificial Intelligence.* AAAI, 2002, pp. 593–598.

[10] L.Maohai, H.Bingrong, C.Zesu, and L.Ronghua. "Novel Mobile Robot Simultaneous Localization and Mapping Using Rao-Blackwellised Particle Filter". In: *International Journal of Advanced Robotic Systems* 3.3 (2006), pp. 231–238.

[11] M.Llofriu, F.Andrade, F.Benavides, A.Weitzenfeld, and G.Tejera. "An embedded particle filter SLAM implementation using an affordable platform". In: *Advanced Robotics (ICAR), 2013 16th International Conference on.* Nov. 2013, pp. 1–6. DOI: 10.1109/ICAR.2013.6766537.

[12] AK.Pandey, K.M.Krishna, and H.Hexmoor. "Feature Chain Based Occupancy Grid SLAM for Robots Equipped with Sonar Sensors". In: *Integration of Knowledge Intensive Multi-Agent Systems, 2007. KI-*

MAS 2007. International Conference on. Apr. 2007, pp. 283–288. DOI: 10.1109/KIMAS.2007.369823.

[13] J.D.Tardós, J.Neira, P.M.Newman, and J.J.Leonard. "Robust mapping and localization in indoor environments using sonar data". In: *Int. J. Robotics Research* 21 (2002), pp. 311–330.

[14] R.O.Duda and P.E.Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1 (Jan. 1972), pp. 11–15. ISSN: 0001-0782. DOI: 10.1145/361237.361242. URL: http://doi.acm.org/10.1145/361237.361242.

[15] JS.Lee, K.Chanki, and C.Wan Kyun. "Robust RBPF-SLAM using sonar sensors in non-static environments". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* May 2010, pp. 250–256. DOI: 10.1109/ROBOT.2010.5509635.

[16] S.Fu, H.Liu, L.F.Gao, and Y.X.Gai. "SLAM for mobile robots using laser range finder and monocular vision". In: *Mechatronics and Machine Vision in Practice, 2007. M2VIP 2007. 14th International Conference on.* Dec. 2007, pp. 91–96. DOI: 10.1109/MMVIP.2007.4430722.

[17] E. Ivanjko and I. Petrović. "Extended Kalman filter based mobile robot pose tracking using occupancy grid maps". In: *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, MELECON 2004.* Vol. 1. 2004, pp. 311–314.

[18] J.A. Castellanos, R.Martinez-Cantin, J.D. Tardós, and J. Neira. "Robocentric map joining: Improving the consistency of EKF-SLAM". In: *Robotics and Autonomous Systems* 55.1 (2007). Simultaneous Localisation and Map Building, pp. 21 –29. ISSN: 0921-8890. DOI: http://dx.doi.org/10.1016/j.robot.2006.06.005. URL: http://www.sciencedirect.com/science/article/pii/S0921889006001448.

[19] J.Bar-Shalom, X.R.Li, and T.Kirubarajan. *Estimation with Application to Tracking and Navigation.* John Wiley and Sons, 2001.

[20] W.Yuanxin, H.Dewen, W.Meiping, and H.Xiaoping. "Unscented Kalman filtering for additive noise case: augmented vs. non-augmented". In: *American Control Conference, 2005. Proceedings of the 2005.* June 2005, 4051–4055 vol. 6. DOI: 10.1109/ACC.2005.1470611.

[21] S.J.Julier and J.K.Uhlmann. "Unscented filtering and nonlinear estimation". In: *Proceedings of the IEEE* 92.3 (Mar. 2004), pp. 401–422. ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.823141.

[22] R.Vullings, B.De Vries, and J.W.M.Bergmans. "An Adaptive Kalman Filter for ECG Signal Enhancement". In: *Biomedical Engineering, IEEE Transactions on* 58.4 (Apr. 2011), pp. 1094–1103. ISSN: 0018-9294. DOI: 10.1109/TBME.2010.2099229.

[23] B.Hongwei, J.Zhihua, and T.Weifeng. "IAE-adaptive Kalman filter for INS/GPS integrated navigation system". In: *Systems Engineering and Electronics, Journal of* 17.3 (Sept. 2006), pp. 502–508. DOI: 10.1016/S1004-4132(06)60086-8.

[24] B.Sridhar, P.Smith, R.E.Suorsa, and B.Hussien. “Multirate and event driven Kalman filters for helicopter passive ranging”. In: *Control Applications, 1992., First IEEE Conference on*. Sept. 1992, 800–805 vol.2. DOI: 10.1109/CCA.1992.269745.

[25] P.Seong-Taek Park and L.Jang Gyu. “Improved Kalman filter design for three-dimensional radar tracking”. In: *Aerospace and Electronic Systems, IEEE Transactions on* 37.2 (Apr. 2001), pp. 727–739. ISSN: 0018-9251. DOI: 10.1109/7.937485.

[26] Z.Zhi-Qiang, M.Xiao-Li, and W.Jian-Kang. “Quaternion-Based Kalman Filter With Vector Selection for Accurate Orientation Tracking”. In: *Instrumentation and Measurement, IEEE Transactions on* 61.10 (Oct. 2012), pp. 2817–2824. ISSN: 0018-9456. DOI: 10.1109/TIM.2012.2196397.

[27] S.Y.Chen. “Kalman Filter for Robot Vision: A Survey”. In: *Industrial Electronics, IEEE Transactions on* 59.11 (Nov. 2012), pp. 4409–4420. ISSN: 0278-0046. DOI: 10.1109/TIE.2011.2162714.

[28] P.Muraca, P.Pugliese, and G.Rocca. “An Extended Kalman Filter for the state estimation of a mobile robot from intermittent measurements”. In: *Control and Automation, 2008 16th Mediterranean Conference on*. June 2008, pp. 1850–1855. DOI: 10.1109/MED.2008.4602251.

[29] L.D’Alfonso, W.Lucia, P.Muraca, and P.Pugliese. “Filters for mobile robots: EKF, UKF and sensor switching - experimental results”. In: *Control and Automation (ICCA), 2011 9th IEEE International Conference on*. Dec. 2011, pp. 925–930. DOI: 10.1109/ICCA.2011.6137952.

[30] G.Cotugno, L.D’Alfonso, P.Muraca, and P.Pugliese. “A new Extended Kalman Filter based on actual local information for mobile robots”. In: *9th European Workshop on Advanced Control and Diagnosis, ACD*. Nov. 2011.

[31] V.Crugliano, L.D’Alfonso, P.Muraca, and P.Pugliese. “Experimental results of a sensor switching policy for mobile robots”. In: *Control Automation (MED), 2010 18th Mediterranean Conference on*. June 2010, pp. 581–585. DOI: 10.1109/MED.2010.5547732.

[32] C.Hyeonwoo and K.Sang-Woo. “Mobile Robot Localization Using Biased Chirp-Spread-Spectrum Ranging”. In: *Industrial Electronics, IEEE Transactions on* 57.8 (Aug. 2010), pp. 2826–2835. ISSN: 0278-0046. DOI: 10.1109/TIE.2009.2036633.

[33] H.Soonshin, L.HyungSoo, and L.JangMyun. “An Efficient Localization Scheme for a Differential-Driving Mobile Robot Based on RFID System”. In: *Industrial Electronics, IEEE Transactions on* 54.6 (Dec. 2007), pp. 3362–3369. ISSN: 0278-0046. DOI: 10.1109/TIE.2007.906134.

[34] L.Hung-Hsing, T.Ching-Chih, and H.Jui-Cheng. “Ultrasonic Localization and Pose Tracking of an Autonomous Mobile Robot via Fuzzy Adaptive Extended Information Filtering”. In: *Instrumentation and Measurement, IEEE Transactions on* 57.9 (Sept. 2008), pp. 2024–2034. ISSN: 0018-9456. DOI: 10.1109/TIM.2008.919020.

[35] A.T.Nelson and E.A.Wan. "A two-observation Kalman framework for maximum-likelihood modeling of noisy time series". In: *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on.* Vol. 3. May 1998, 2489–2494 vol.3. DOI: 10.1109/IJCNN.1998.687253.

[36] P.Fan, L.F.Sui, B.Wang, and W.Wang. "Particle Filter-Weight Estimation and Dual Particle Filter". In: *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on.* May 2009, pp. 1–4. DOI: 10.1109/IWISA.2009.5073000.

[37] *KTEAM Corporation [Online].* http://www.k-team.com.

[38] C.Hsu, H.Chen, and C.Lai. "An Improved Ultrasonic-Based Localization Using Reflection Method". In: *Informatics in Control, Automation and Robotics, 2009. CAR '09. International Asia Conference on.* Feb. 2009, pp. 437–440. DOI: 10.1109/CAR.2009.93.

[39] L.Ji-Hong, L.Mun-Jik, P.Sang-Hyun, and K.Jong-Geol. "Range sonar array based SLAM for P-SURO AUV in a partially known environment". In: *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on.* Nov. 2012, pp. 353–354. DOI: 10.1109/URAI.2012.6463014.

[40] Teddy N. Yap Jr. and Christian R. Shelton. "SLAM in Large Indoor Environments with Low-Cost, Noisy, and Sparse Sonars". In: *Proceedings of the IEEE International Conference on Robotics and Automation.* 2009, pp. 1395–1401.

[41] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I". In: *Robotics Automation Magazine, IEEE* 13.2 (June 2006), pp. 99–110. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[42] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping (SLAM): part II". In: *Robotics Automation Magazine, IEEE* 13.3 (Sept. 2006), pp. 108–117. ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1678144.

[43] J. Neira and J.D. Tardós. "Data association in stochastic mapping using the joint compatibility test". In: *Robotics and Automation, IEEE Transactions on* 17.6 (Dec. 2001), pp. 890–897. ISSN: 1042-296X. DOI: 10.1109/70.976019.

[44] H.P.Moravec. "Sensor fusion in certainty grids for mobile robots". In: *AI magazine* 9.2 (1988), pp. 61–74.

[45] D. Wolf and G.S. Sukhatme. "Online simultaneous localization and mapping in dynamic environments". In: *Robotics and Automation (ICRA), 2004 IEEE International Conference on.* Vol. 2. Apr. 2004, pp. 1301–1307. DOI: 10.1109/ROBOT.2004.1308004.

[46] H.C.Roh, C.H.Sung, M.T.Kang, and M.J.Chung. "Fast SLAM using polar scan matching and particle weight based occupancy grid map for mobile robot". In: *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on.* Nov. 2011, pp. 756–757. DOI: 10.1109/URAI.2011.6146004.

[47] Michael Drumheller. “Mobile robot localization using sonar”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (1987), pp. 325–332.

[48] Q.Yang, K.Yuan, J.Li, and H.Wang. “A histogram sensor fusion method for mobile robots”. In: *Information Acquisition, 2004. Proceedings. International Conference on.* June 2004, pp. 339–343. DOI: 10.1109/ICIA.2004.1373384.

[49] L.Pedraza, D.Rodriguez-Losada, F.Matia, G.Dissanayake, and J.V.Miro. “Extending the Limits of Feature-Based SLAM With B-Splines”. In: *Robotics, IEEE Transactions on* 25.2 (Apr. 2009), pp. 353–366. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2013496.

[50] E.Parzen. *Stochastic Processes.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1999. ISBN: 9780898714418. URL: http://books.google.it/books?id=mwW-iODttSQC.

[51] D.Fox, J.Ko, K.Konolige, B.Limketkai, D.Schulz, and B.Stewart. “Distributed Multirobot Exploration and Mapping”. In: *Proceedings of the IEEE* 94.7 (July 2006), pp. 1325–1339. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.876927.

[52] Sebastian Thrun. “A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots”. In: *The International Journal of Robotics Research* 20.5 (2001), pp. 335–363. DOI: 10.1177/02783640122067435. eprint: http://ijr.sagepub.com/content/20/5/335.full.pdf+html. URL: http://ijr.sagepub.com/content/20/5/335.abstract.

[53] A.Howard, G.Sukhatme, and M.J.Mataric. “Multirobot Simultaneous Localization and Mapping Using Manifold Representations”. In: *Proceedings of the IEEE* 94.7 (July 2006), pp. 1360–1369. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.876922.

[54] X.S.Zhou and S.I.Roumeliotis. “Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case”. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* Oct. 2006, pp. 1785–1792. DOI: 10.1109/IROS.2006.282219.

[55] T.Tong, H.Yalou, Y.Jing, and S.Fengchi. “Multi-robot cooperative map building in unknown environment considering estimation uncertainty”. In: *Control and Decision Conference, 2008. CCDC 2008. Chinese.* July 2008, pp. 2896–2901. DOI: 10.1109/CCDC.2008.4597854.

[56] Luca Carlone, Miguel Kaouk Ng, Jingjing Du, Basilio Bona, and Marina Indri. “Simultaneous Localization and Mapping Using Rao-Blackwellized Particle Filters in Multi Robot Systems”. English. In: *Journal of Intelligent & Robotic Systems* 63.2 (2011), pp. 283–307. ISSN: 0921-0296. DOI: 10.1007/s10846-010-9457-0. URL: http://dx.doi.org/10.1007/s10846-010-9457-0.

[57] W.D. Blair and T.Bar-Shalom. “Tracking maneuvering targets with multiple sensors: does more data always mean better estimates?” In:

Aerospace and Electronic Systems, IEEE Transactions on 32.1 (Jan. 1996), pp. 450–456. ISSN: 0018-9251. DOI: `10.1109/7.481286`.

[58] V. Gupta, T.H. Chung, B.Hassibi, and R. M. Murray. "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage". In: *Automatica* 42.2 (2006), pp. 251 –260. ISSN: 0005-1098. DOI: `http://dx.doi.org/10.1016/j.automatica.2005.09.016`. URL: `http://www.sciencedirect.com/science/article/pii/S0005109805003663`.

[59] L.Carotenuto, P.Muraca, and G.Raiconi. "On the optimal design of the output transformation for discrete-time linear systems". English. In: *Journal of Optimization Theory and Applications* 68.1 (1991), pp. 1–18. ISSN: 0022-3239. DOI: `10.1007/BF00939932`. URL: `http://dx.doi.org/10.1007/BF00939932`.

[60] Francesco Lunghi, Alberto Pavese, Simone Peloso, Igor Lanese, and Davide Silvestri. "Computer Vision System for Monitoring in Dynamic Structural Testing". English. In: *Role of Seismic Testing Facilities in Performance-Based Earthquake Engineering.* Ed. by Michael N. Fardis and Zoran T. Rakicevic. Vol. 22. Geotechnical, Geological, and Earthquake Engineering. Springer Netherlands, 2012, pp. 159–176. ISBN: 978-94-007-1976-7. DOI: `10.1007/978-94-007-1977-4_9`. URL: `http://dx.doi.org/10.1007/978-94-007-1977-4_9`.

[61] M.Hansard, S.Lee, O.Choi, and R.Horaud. *Time of Flight Cameras - Principles, Methods and Applications.* Springer, 2012.

[62] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. "Depth Mapping Using Projected Patterns". In: *Application US20 080 240 502 A1.* 2008, pp. 1–7.

[63] H. Haggag, M. Hossny, D. Filippidis, and D.Creighton. "Measuring depth accuracy in RGBD cameras". In: *IEEE International Conference on Signal Processing and Communication Systems (ICSPCS) 2013.* 2013, pp. 1–7.

[64] Richard Szeliski. *Computer Vision - Algorithms and Applications.* Ed. by Springer. 2011.

[65] American Society of Photogrammetry. *Manual of Photogrammetry.* Fourth edition. 1980.

[66] J. R. Ohm, K.Grüneberg, E.Hendriks, and E. Izquierdo. "A Realtime Hardware System for Stereoscopic Videoconferencing with Viewpoint Adaptation". In: *Signal Processing: Image Communication.* 1998, pp. 147–171.

[67] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Van de Gool, E. Ofek, and I. Sexton. "An Evolutionary and Optimised Approach on 3D-TV". In: *In Proceedings of International Broadcast Conference.* 2002, pp. 357–365.

[68] K. Kamarudin, S. MMamduh, A. Y. M. Shakaff, S. M. Saad, A. Zakaria, A. H. Abdullah, and L. M. Kamarudin. "Method to Convert Kinect's 3D Depth Data to a 2D Map for Indoor SLAM". In: *IEEE 9th International*

Colloquium on Signal Processing and its Applications. 2013, pp. 247–251.

[69] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake. “A robust RGB-D SLAM algorithm”. In: *International Conference on Intelligent Robots and Systems*. 2012, pp. 1714–1719.

[70] X. Wang, Y. A. Şekercioğlu, and T. Drummond. “A Real-Time Distributed Relative Pose Estimation Algorithm for RGB-D Camera Equipped Visual Sensor Networks”. In: *7th ACM/IEEE International Conference on Distributed Smart Cameras*. 2013, pp. 247–251.

[71] A. Tuntakurn, S. S. Thongvigitmanee, V. Sa-Ing, S. Hasegawa, and S. S. Makhanov. “Natural Interactive 3D Medical Image Viewer Based on Finger and Arm Gestures”. In: *2013 Biomedical Engineering International Conference (BMEiCON-2013)*. 2013, pp. 1–5.

[72] N. Kitsunezaki, E. Adachi, T. Masuda, and J. Mizusawa. “KINECT Applications for The Physical Rehabilitation”. In: *IEEE*. 2013, pp. 294–299.

[73] H. Funaya, T. Shibata, Y. Wada, and T. Yamanaka. “Accuracy Assessment of Kinect Body Tracker in Instant Posturography for Balance Disorders”. In: *2013 7th International Symposium on Medical Information and Communication Technology (ISMICT)*. 2013, pp. 213–217.

[74] A. B. H. Mohamed, T. Val, L. Andrieux, and A. Kachouri. “Using a Kinect WSN for home monitoring: proncipple, network and application”. In: *2012 International Conference on Wireless Communications in Unusual and Confined Areas (ICWCUCA)*. 2012, pp. 1–5.

[75] D. G. Costa, L. A. Guedes, F. Vasques, and P. Portugal. “Adaptive Monitoring Relevance in Camera Networks for Critical Surveillance Applications”. In: *International Journal of Distributed Sensor Networks* (2013), pp. 1–14.

[76] D. Taubman and M. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2002.

[77] K. Sayood. *Introduction to Data Compression*. Ed. by MK Morgan Kauffmann - Elsevier. 2006.

[78] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. “Overview of the H.264/AVC Video Coding Standard”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (2003), pp. 560–576.

[79] M. Sarkis and K. Diepold. “Depth map compression via compressed sensing”. In: *16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 737–740.

[80] E. Ekmekcioglu, S. T. Worrall, and A. M. Kondoz. “A Temporal Subsampling Approach for Multiview Depth Map Compression”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 19.8 (2009), pp. 1209–1213.

[81] J. Y. Lee, H. C. Wey, and D. S. Park. “A Fast and Efficient Multi-View Depth Image Coding Method Based on Temporal and Inter-View

Correlations of Texture Images". In: *IEEE Transaction on Circuits and Systems for Video Technology* 21 (2011), pp. 1859–1868.

[82] S. Milani, P. Zanuttigh, P. Zamarin, and S. Forchhammer. "Efficient depth map compression exploiting segmented color data". In: *International Conference on Multimedia and Expo (ICME)*. 2011, pp. 1–6.

[83] L. F. R. Lucas, N. M. M. Rodrigues, C. L. Pagliari, E. A. B. da Silva, and S. M. M. de Faria. "Efficient depth map coding using linear residue approximation and a flexible prediction framework". In: *19th International Conference on ImageProcessing (ICIP)*. 2012, pp. 1305–1308.

[84] A. Puri, H. M. Hang, and D. L. Schilling. "An efficient block matching algorithm for motion-compensated coding". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP*. 1987, pp. 1063–1066.

[85] B. Kamolrat, W. A. C. Fernando, and A. Kondoz. "3D Motion Estimation for Depth Image Coding in 3D Video Coding". In: *IEEE Transactions on Consumer Electronics* 55.2 (2009), pp. 824–830.

[86] B. Kamolrat, W. A. C. Fernando, and M. Mrak. "Adaptive motion-estimation-mode selection for depth video coding". In: *2010 IEEE International Conference on Digital Object Identifier*. 2010, pp. 702–705.

[87] P. F. Felzenszwalb and D. P. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: 2 (2004), pp. 167–181.

[88] W. I. Grosky and R. Jain. "Optimal Quadtrees for Image Segments". In: *IEEE Transactions on Pattern Analysis and machine intelligence* 1 (1983), pp. 77–83.

[89] W. R. Mark, L. McMillan, and G. Bishop. "Post-Rendering 3D Warping". In: *Symposium on Interactive 3D Graphics, Providence, RI, April 27-30*. 1997, pp. 7–16.

[90] H. M. Wang, C. H. Huang, and J. F. Yang. "Block-Based Depth Maps Interpolation for Efficient Multiview Content Generation". In: *IEEE Transaction on Circuits and Systems for Video Technology* 21 (2011), pp. 1847–1858.

[91] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 573–580.