

Università di Modena e Reggio Emilia

**DOTTORATO DI RICERCA IN
INGEGNERIA DELL'INNOVAZIONE
INDUSTRIALE**

Ciclo XXVI

Settore Concorsuale di Afferenza: 09/G1
Settore Scientifico Disciplinare: ING/INF-04

**New Estimation and Control Techniques for
Real World Mobile Robots**

Presentata da: **Alessio Levratti**

Direttore della scuola:
Prof. Mauro Dell'Amico

Relatore:
Dr. Cristian Secchi

Esame finale anno 2014

Copyright ©2014 by Alessio Levratti.

Reggio Emilia, Italy, March 2014.

To my wife Anna

Abstract

This thesis deals with the technology transfer of the most recent findings in the mobile robotics field into real world industrial and service applications. Specifically the presented research has been done in the framework of two projects: DiRò and REMOCAL.

The main scope of the DiRò (Distretto della Robotica Mobile) project is to create a collaboration between companies of the province of Reggio Emilia with the objective to develop new solutions in the framework of the Mobile Robotics. This thesis focuses mainly on three development objectives of this project.

First the design of a smart autonomous lawnmower, and in particular its localization algorithm, is described in this thesis. The robot uses a particular implementation of the Extended Kalman Filter (EKF) for localizing itself in the garden. Then a novel navigation strategy has been implemented, in order to make the robot doing its mowing task ensuring a complete and even coverage of the assigned working area.

Then the characterization of 3D sensors is discussed. In particular a comparison between a stereo-camera (a Bumblebee2 stereo-camera) and an RGB-D sensor (the Microsoft Kinect) is presented. Then the performance of the two sensors is tested through a localization algorithm based on an EKF.

The DiRò project discussion eventually ends with the design of a novel lead-acid batteries State of Charge (SoC) estimator also based on the EKF algorithm.

Finally the REMOCAL (Robotic Equipment for Measuring by Optics Calibrator) project is described. The project consists of the design and the realization of a mobile robot able to autonomously calibrate a car tyres convergence measurement device. In this manuscript both the mechanical design of the robot and the development of the software which controls the robot are reported.

Sommario

Questa tesi tratta del trasferimento tecnologico delle più recenti scoperte nel settore della robotica mobile in applicazioni industriali e di servizio nel mondo reale. Nello specifico, le ricerche presentate sono state svolte nel framework di due progetti: il progetto DiRò e il progetto REMOCAL.

Lo scopo del progetto DiRò (Distretto della Robotica Mobile) è di creare una collaborazione fra alcune aziende della provincia di Reggio Emilia con l'obiettivo di sviluppare nuove soluzioni nell'ambito della robotica mobile. Questa tesi tratta principalmente di tre obiettivi realizzativi di questo progetto.

In questa tesi verrà per prima cosa affrontata la progettazione di un tosaerba automatico e, in particolare, del suo algoritmo di localizzazione. Il robot usa una particolare implementazione dell' Extended Kalman Filter (EKF) per localizzarsi nel giardino. Quindi è implementata una innovativa strategia di navigazione che fa sì che il robot possa affrontare il suo compito di tosatura assicurando una copertura uniforme e completa del prato.

Dopodiché viene discussa la caratterizzazione di sensori di visione 3D. In particolare è presentato il confronto di una stereo-camera (Bumblebee2) e un sensore RGB-D (il Kinect Microsoft). Le performance dei due sensori sono testate con un algoritmo di localizzazione basato su un EKF.

La discussione sul progetto DiRò termina con la progettazione di un innovativo sistema per la stima dello Stato di Carica di batterie al piombo, anch'esso basato sull' EKF.

Infine viene descritto il progetto REMOCAL (Robotic Equipment for Measuring by Optics Calibrator). Esso consiste nella progettazione e nella realizzazione di un robot mobile capace di calibrare autonomamente un dispositivo di misura della convergenza dei pneumatici per automobili. In questo documento sono riportati sia la progettazione meccanica che la progettazione del software di controllo del robot.

Contents

1	Introduction	1
2	Mobile Robotics	5
2.1	The fundamental tasks of mobile robots	6
2.1.1	Non-Holonomic Constraints	9
2.1.2	The localization problem: what's wrong with odometry?	11
3	Localization and Sensor Fusion in Mobile Robotics	15
3.1	Introduction to Localization and Sensor Fusion	15
3.2	Kalman Filtering techniques - The Kalman Filter and the Extended Kalman Filter	17
3.2.1	The Kalman Filter	17
3.2.2	The Extended Kalman Filter	19
4	Robomower	23
4.1	Introduction	23
4.2	Hardware Setup	27
4.3	Characterization of the sensors	28
4.3.1	RFID sensor characterization	28
4.3.2	The wireless communication model	32
4.4	The Navigation strategy	33
4.5	Simulations	41
4.6	Experiments	44
4.6.1	Experiments: coverage in controlled environment	44
4.6.2	Experiments: outdoor environment	47
4.7	Discussion	50

5	3D sensors characterization	53
5.1	Introduction	53
5.2	Description of the adopted technology	55
5.2.1	3D vision sensors	55
5.2.2	Self-similar landmarks	57
5.3	Experimental characterization of the 3D vision sensors	58
5.4	EKF-based Localization using self-similar landmarks	60
5.5	Experimental results	62
5.6	Analysis of the results and conclusion	63
6	Lead-acid batteries on-line State-of-Charge estimate using an Extended Kalman Filter	69
6.1	Introduction	69
6.2	State-of-Charge Estimation Techniques	70
6.3	Proposed Method	71
6.4	System Identification	74
6.5	Experimental Results	75
6.6	Conclusions and Future Work	76
7	Project REMOCAL	79
7.1	REMO - Robotic Equipment for Measuring by Optics	79
7.2	How REMO works	80
7.3	The Calibration Procedure	86
7.3.1	The Mono-Calibration	86
7.3.2	The Stereo-Calibration and the Field-Wheel Cameras calibration	88
7.3.3	Laser calibration	90
7.3.4	Final Test	91
7.3.5	Target Bar Calibration	92
7.4	Design Requirements	92
7.5	Hardware design	93
7.5.1	Conceptual Design	93
7.5.2	Concepts Recombination and Evaluation	98
7.5.3	Hardware architecture	100
7.6	Algorithm Description	105
7.6.1	Implementation of the interface with the robot	105
7.6.2	Navigation algorithms implementation	106

7.6.3	OpenCV derived classes	109
7.7	Discussion and Future Works	110
8	Conclusion	113
A	Background on bayesian probability theory	115
A.1	The probability	115
A.2	Bayes' rule	116
	Bibliography	119

Chapter 1

Introduction

In the last years more and more mobile robotics based products appeared on the market, both for civil (i.e. autonomous lawnmowers, vacuum cleaners, etc. . .) and for industrial (i.e. AGV) applications. Mobile robotics is a very fast growing field, thanks to a very large community and to the new developing tools which enable the fast prototyping of robots. One of the most important issue in mobile robotics is the localization problem. The robot, in order to accomplish efficiently its tasks, must know its position with a certain degree of precision. This means that the robot should carry a sufficient number of sensors which, with the most appropriate algorithms, can give information about the position of the robot itself, like laser scanners, accurate GPS (Global Positioning System) devices and so on. Unfortunately, in civil application, this leads to a cost of the robot which may not be competitive on the market. The challenge is then to build robots with low cost sensors and then provide them with sensor fusion and filtering algorithms which allows the determination of the position of the robot with a sufficiently precise accuracy according to the application.

This thesis deals mainly with the “*Technology Transfer*” of the most recent discoveries in mobile robotics into real world application. Technology Transfer, also called Transfer of Technology and Technology Commercialization, is the process of transferring knowledge, technologies, “*know how*” and promising research topics from universities and research institutes.

The proposed work was done principally in the framework of two projects: the DiRò project and the REMOCAL project.

The DiRò (Distretto della Robotica Mobile) Project was born from the collaboration of companies leading both in the production of systems (i.e. Elettric80, that produces AGVs, and Emak, that produces gardening tools like lawnmowers) and in the production of components (i.e. Zivan, that produces tools for electric vehicles like batteries, Motor Power Company, that produces electric motors, Tellure Rota, that produces high performance wheels, ADEL system, that produces automatic systems), with also the col-

laboration with the laboratory INTERMEC/MECTRON, with the Department of Science and Methods of Engineering (DISMI) from the University of Modena and Reggio Emilia, with the center for innovation and Technology Transfer DEMOCENTER-SIPE, the CIRI-Meccanica Avanzata e Materiali (Interdepartmental Center for Industrial Research of the University of Bologna, which works in the field of advanced applications in mechanical engineering and materials technology.) and the Associazione Industriali of the province of Reggio Emilia. Figure 1.1 represents the DiRò project logo.



Figure 1.1: The DiRò logo

The main objective of the DiRò Project consisted of the development and in the experimental validation of methodologies and technologies for the automation industry. This was done through the pooling of specific experiences of applicants assisted by the expertise of University research groups.

The working model adopted in the project expects the systems producers (Elettric80 and Emak) to develop their technology innovation systems assisted by the producers of components (the other companies) and by the researchers of the two involved Universities. By following this cooperation model, companies are developing innovative technologies for the automation of their products and increasing their efficiency.

The work is divided into five Development Objectives (DO). Development Objectives are then split up into Specific Activities (AS) as follows:

- DO 1 - Scientific management of the activities;
- DO 2 - Movement and Energy:
 - AS 2.1 - Development of efficient and “smart” wheels;
 - AS 2.2 - Development of technologies for the charging system control of batteries.
- DO 3 - Autonomous behavior and control

- AS 3.1 - Algorithms for mobile robots localization and for context awareness in industrial environment;
- AS 3.2 - Smart navigation algorithms for small autonomous robots in civil environment.
- DO 4 - Sensors and electric actuators
 - AS 4.1 - Sensors for mobile robotics;
 - AS 4.2 - Wheels integrated actuators development;
- DO 5 - Prototypes development and technologies integration

This thesis will focus mainly on AS 3.2, AS 4.1 and on AS 2.2; the first activity had as objective the development of a localization and navigation algorithms for unstructured environments and the development of control strategies for small robots. In particular the activity leads to the realization of a prototype autonomous lawnmower, described in Chapter 4. The second activity had the objective to identify the most reliable sensory equipment both for industrial and civil environment (Chapter 5). This activity contributed both for AS 3.2 and for AS 3.1 by characterizing those sensors which could be used in those activities. Activity 2.2 had as objective the development of a robust and efficient method to control the recharge procedure of low power batteries (Chapter 6).

Then the thesis deals with the REMOCAL project. The objective of the project was the design and the realization of a mobile robot capable of autonomously calibrate the “REMO” (Robotic Equipment for Measuring by Optics) device, a useful contact-less tool for tyre repairers which can measure the toe and the camber of car tyres. This project involved CORGHI s.p.a., a company leading in the production of car maintenance tools, and in particular in tyres substitution and repairing. In this thesis the calibration procedure of the REMO device is provided, and then both hardware and software design are described.

Chapter 2

Mobile Robotics

This chapter introduces the concept of Mobile Robot. A brief description of the main typologies of mobile robots is provided. The the thesis focuses on wheeled mobile robots, and, in particular, on differential drive wheeled mobile robots. The principal tasks as well as the main issue of mobile robotics are described.

A robot (see [1]) is a machine capable of interaction with the surrounding environment through actions conditioned by a set of behavioral rules and by a set of data on the robot's state itself and collected from the environment the robot is working in.

The interaction with the surrounding environment is a crucial topic for robotics, which is evolving more and more in the study of the smart connection between perception and action.

In particular the capability of the robot to act on the surrounding environment is left to a mechanical system composed by locomotion mechanisms which allow the robot to move and by manipulators or end-effectors the robot can interact the surrounding objects with.

The perception task is accomplished by sensors which acquire information about the internal state of the machine and data on the surroundings. The elaboration of the acquired information is done by the computational unit of the robot, which can execute high abstraction level cognitive programs through analysis, motion control and learning algorithms.

Mobile robotics is a particular branch of robotics and is a fast growing field because of the vast variety of its applications in industry, home automation, security, exploration, surveillance, etc. . .

A mobile robot is an autonomous machine which is not subject to the constraint of having a fixed base position and that is capable of freely moving in a structured or unstructured environment. Even if a mobile robot's structure strongly depends on its

final purpose, the main task for mobile robots are the computation of its position in the environment, the decision of the next position to reach and the decision of the best path to follow in order to reach the goal.

2.1 The fundamental tasks of mobile robots

The basic task of mobile robots is to move from the actual position to another position (the goal) in the space while avoiding the static or dynamic obstacles it can meet on its path. Briefly the robot should be able to accomplish these tasks:

- Parking: the robot starts in an initial configuration (position and orientation in the space) and it moves to a desired final configuration (see Fig. 2.1).



Figure 2.1: Schematic representation of the “parking” task

- Path following: the robot starts in an initial configuration and it moves along a certain path which can both be computed on-line and change dynamically, according to the possible obstacles the robot can find on its path, or simply be computed “*a priori*” (see Fig. 2.2).

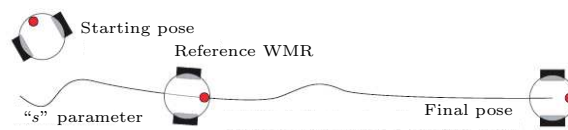


Figure 2.2: Schematic representation of the “path following” task

- Trajectory tracking: the robot starts in an initial configuration and it moves along a trajectory according to the desired speed profile (see Fig. 2.2).

In order to accomplish its tasks, the robot is equipped with a locomotion mechanism and according to it it is possible to classify the robot in different categories:

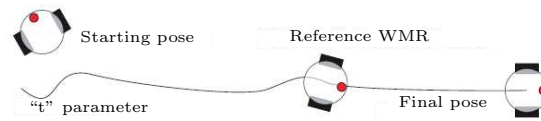


Figure 2.3: Schematic representation of the “path following” task

- Flying mobile robots (fig. 2.4): these robots, which usually have six DOF (Degrees Of Freedom), are capable of moving in all the directions exploiting propulsion systems able to make them fly. These robots are used for exploration tasks, patrolling, search & rescue, etc. . .

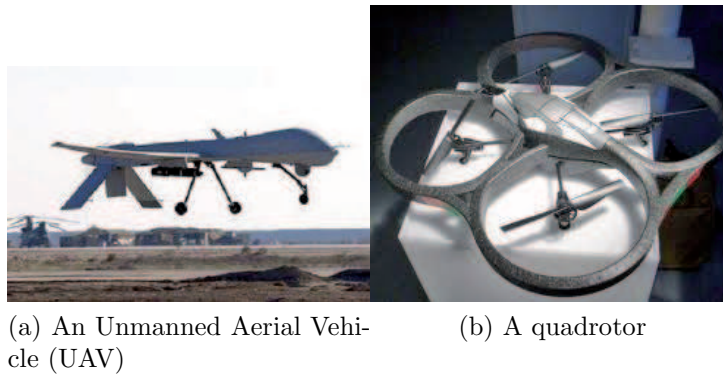


Figure 2.4: Two examples of flying robots

- Underwater mobile robots (fig. 2.5): these robots can swim underwater at depth usually not reachable by ordinary submarine vehicles.



Figure 2.5: An underwater robot

- Terrestrial mobile robots (fig. 2.6): robots that can move on more or less flat surfaces for exploration tasks, patrolling, material handling, etc. . .

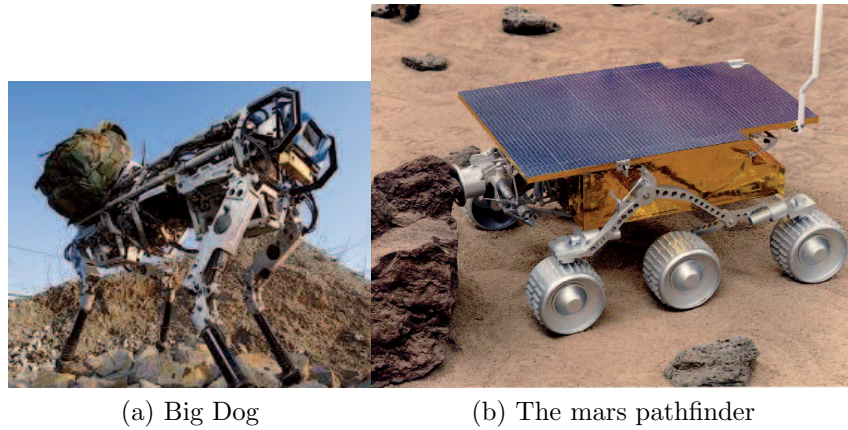


Figure 2.6: Two examples of terrestrial robots.

This last category can be further split into sub-categories according to the actuator which implements the locomotion:

- Legged robots (fig. 2.7): the robot can move by displacing its center of gravity between its support points in order to allow the movement of a weight-free limb to a desired position. The robot can have multiple set of legs.

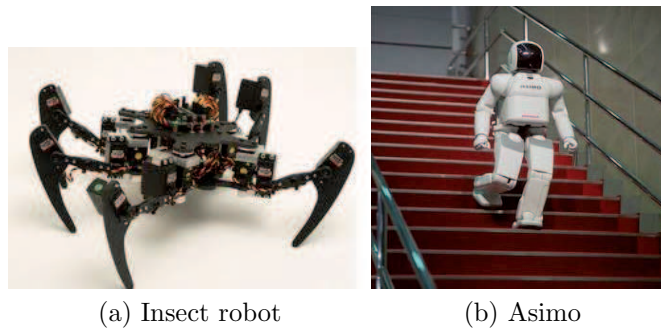


Figure 2.7: Legged mobile robots.

- Wheeled robots or Wheeled Mobile Robots (WMR): the weight of the robot is distributed on all its wheels. The robot can then move by imposing to at least one wheel a rotation (see fig. 2.8).

In this thesis only WMR with a differential drive kinematic structure will be considered. The differential drive configuration, which is the most common realization of the

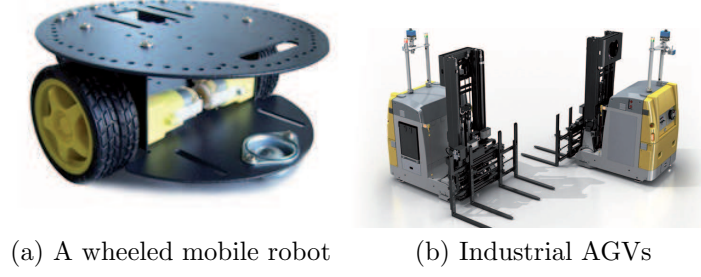


Figure 2.8: Wheeled mobile robots.

unicycle, consists in two fixed wheels, co-axial and independently actuated, and one or more castor wheels (usually smaller) whose role is to keep the robot in static equilibrium. The configuration of the robot is usually represented with the state vector $\vec{\mu} = [x, y, \vartheta]^T$, where x and y are the robot's wheelbase midpoint coordinates in the Cartesian space and ϑ is the robot's orientation (see Fig. 2.9).

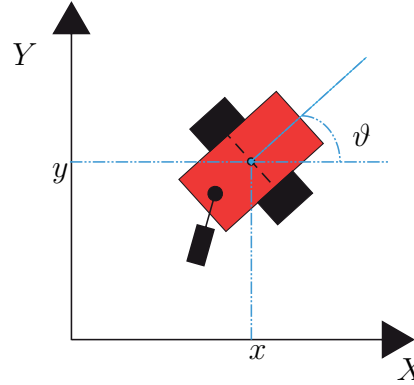


Figure 2.9: Schematic representation of a differential drive WMR

The castor wheel is passive and the two wheels are actuated independently. Let ω_R and ω_L be respectively the rotation speed of the right and of the left wheel. The robot translates if $\omega_R = \omega_L$ and it turns on itself in a positive rotation (according to the right hand rule) if $\omega_R = -\omega_L$.

2.1.1 Non-Holonomic Constraints

The use of wheels in a kinematic structure introduces the assumption that each wheel rolls without sliding (i.e. it does not slide neither longitudinally nor laterally). This assumption

means that each wheel introduces a non-holonomic constraint in the mechanical system since it does not allow a translation orthogonal to the rolling direction, as shown in Fig. 2.10. The introduction of this kind of constraints implies a limit to the instantaneous mobility of the robot, but this does not imply a reduction of the reachability of any desired configuration (e.g. parallel car parking).

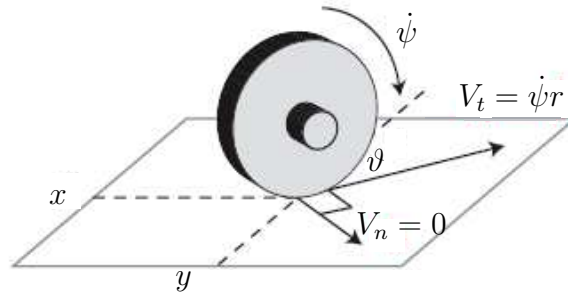


Figure 2.10: Non-holonomic constraint introduced by the wheel

Let's consider the contact point of the wheel with the ground with coordinates x, y . The velocity of this point is:

$$\begin{cases} \dot{x} = V_t \cos \vartheta + V_n \cos \left(\frac{\pi}{2} - \vartheta \right) \\ \dot{y} = V_t \sin \vartheta + V_n \sin \left(\frac{\pi}{2} - \vartheta \right) \end{cases} \quad (2.1)$$

If we consider the assumption $V_n = 0$, the system is reduced to:

$$\begin{cases} \dot{x} = V_t \cos \vartheta \\ \dot{y} = V_t \sin \vartheta \end{cases} \quad (2.2)$$

Which introduced the analytic representation of the constraint on the mobility

$$\dot{x} \sin \vartheta - \dot{y} \cos \vartheta = 0 \quad (2.3)$$

For each wheel it is possible to write eq. 2.3 for the configuration variable μ as $a(\mu) \dot{\mu} = 0$, that for N wheels can be rewritten in matrix notation with

$$A(\mu) \dot{\mu} = 0 \iff \dot{\mu} \in \ker A(\mu) \quad (2.4)$$

which is a Pfaffian constraint [2]. This means that the admissible velocities, input for the system, can be generated by a matrix $G(\mu)$ such that

$$\text{Im}(G(\mu)) = \ker(A(\mu)) \forall \mu \in \mathcal{C} \quad (2.5)$$

where $\mathcal{C}$ is the configuration space for the robot. The admissible direction of motion in the configuration space is then generated through

$$\dot{\mu} = G(\mu) \nu \quad (2.6)$$

where ν is the input vector.

Briefly, given the configuration of the robot, represented by $\mu = [x, y, \vartheta]^T$, the constraint on the instantaneous movements is

$$\dot{x} \sin \vartheta - \dot{y} \cos \vartheta = [\sin \vartheta, -\cos \vartheta, 0]^T \dot{\mu} = 0 \quad (2.7)$$

This can be represented in Pfaffian form with

$$A(\mu) \dot{\mu} = 0, \text{ s.t. } A(\mu) = [\sin \vartheta, -\cos \vartheta, 0] \quad (2.8)$$

Then, from eq. 2.5:

$$G(\mu) = \begin{bmatrix} \cos \vartheta & 0 \\ \sin \vartheta & 0 \\ 0 & 1 \end{bmatrix} \quad (2.9)$$

Let v and ω be respectively the linear speed and the angular speed of the robot. The kinematic model of the unicycle becomes:

$$\dot{\mu} = \begin{bmatrix} \cos \vartheta \\ \sin \vartheta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} \cos \vartheta & 0 \\ \sin \vartheta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \iff \begin{cases} \dot{x} = v \cos \vartheta \\ \dot{y} = v \sin \vartheta \\ \dot{\vartheta} = \omega \end{cases} \quad (2.10)$$

The velocity of the midpoint of the wheels v and the angular speed of the robot ω are biunivocally related to the wheels rotation velocity through

$$\begin{cases} v = \frac{(\omega_R + \omega_L) r}{2} \\ \omega = \frac{(\omega_R - \omega_L) r}{d} \end{cases} \quad (2.11)$$

where r is the wheels' radius and d is the distance of the wheels, nominally the dimension of the wheelbase or the axle.

2.1.2 The localization problem: what's wrong with odometry?

The evolution of robotics led mobile robots to be more and more autonomous and gave them high mobility, intended as the capability of free movements.

In order to give autonomy to mobile robots it is necessary to equip them with instruments and sensors which allow the analysis of the surrounding environment. For example,

the navigation task for a mobile robot consists of making the robot go from a starting point to a goal. The main issues for this fundamental mobile robot's mission can be summarized in three questions:

- **Where is the robot?** The robot has to know with a certain accuracy its position. This is needed in order to take correct decisions about the future movements. The “localization” of the mobile robot should answer to this question.
- **Where does the robot has to go?** In order to be able in moving successfully and to respect constraints, the robot has to know the goal position.
- **How can the robot get there?** Once the robot knows its position and the goal it has to reach, it has to decide how to do that.

The localization is the process that deals with the computation of the position of the robot in its environment. All mobile robots should know where they are located in order to be able to work autonomously. This means that the answer to the question “*Where is the robot?*” is fundamental for mobile robotics. Formally, the localization problem consists of determining the position and the orientation of the robot in an absolute reference system according to data coming from the sensors the robot is equipped with. Without knowing its current position the robot cannot decide what to do and then it cannot be autonomous.

The localization of a mobile robot can be dealt with estimation methods of the “relative” or “absolute” position:

- **Absolute estimate:** the position of the robot is computed through the elaboration of measurements provided by exteroceptive sensors (e.g. GPS, laser scanners, cameras, etc...). This estimation provides information about the position of the robot with respect to reference points (like artificial markers, walls, etc...) located in the robot's working environment.
- **Relative estimate:** the position of the robot is computed through the elaboration of measurement provided by proprioceptive sensors (e.g. encoders, Inertial Measurement Units, etc...). These measurements regard the movement of the robot itself.

In an ideal and perfect world, localization could be accomplished by using only odometric data. In fact, if odometry would be precise enough, the position of the robot could be obtained by integrating the velocities in eq. 2.10. Odometry has indeed a lot of merits:

- Very high sampling frequency.
- Has a relatively low cost and usually mobile robots' wheels motors are already equipped with sensors (like Hall's sensors or encoders), already used for the low level control, capable of providing odometric information.
- Good accuracy in the brief period.
- Can be merged with absolute measurements in order to obtain more reliable estimates.

Unfortunately, in real conditions, the non-holonomic constraints introduced by the wheels are imperfect. This means that the condition of pure rolling for the wheels falls, and the wheels can drift laterally or in the rolling direction. Drifting can be induced by many factors:

- Slippery floor.
- Sudden accelerations.
- External agents (e.g. a collision with an object).
- Non-punctiform contact with the floor.

It is also important to consider that to the listed random factors, also systematic errors should be added, like wheels misalignment, wheels diameter that differs from the nominal value (for example because of deformation of the wheel itself), etc...

By integrating the velocity data provided by odometric sensors leads inevitably to accumulate errors, which grow as time goes by. Then it becomes fundamental for the correct computation of the position of the robot the use of other more reliable source of information.

Generally, sensors provide measurements affected by some noise. In order to compute with high accuracy the position of a robot expensive sensors are needed. So it is not convenient to use only one expensive sensor to compute the position of a mobile robot. Usually information provided by different low cost sensors are merged together through a proper "sensor fusion algorithm". The next chapter will provide a deep description of the main algorithm for localization of mobile robots through sensor fusion.

Chapter 3

Localization and Sensor Fusion in Mobile Robotics

An introduction to the main algorithm for sensor fusion and mobile robots localization is provided in this chapter. In order to keep the cost of robot's sensor equipment low, is convenient to use a sensor fusion algorithm which merges the information provided by low cost (and less accurate) sensors, instead of using accurate, but expensive, sensors. Furthermore the use of certain kind of high precision sensors is not always possible. In this chapter some strategies for the use of alternative sensors for the localization through sensor fusion algorithms are illustrated.

3.1 Introduction to Localization and Sensor Fusion

In order to use sensor data to compute the position of a robot with relatively high precision, accurate and expensive sensors are required. For example, AGVs (Autonomous Guided Vehicles) usually use laser range-finder trilateration or triangulation (e.g. [3]) to compute their pose. This requires, for large environments like warehouses, hundreds of reflective markers. The position of these markers is then measured with very high accuracy and stored in the memory of the robot. Then an accurate laser range-finder, mounted on the robot, reads the distance and the bearing of at least three markers. Then, by interpolating the three circumferences with radius the range measurements and center the position of the markers, the position of the AGV can be computed. It is easy to understand that such an approach is very expensive and laborious, mostly in the positioning of the reflective markers. It would be much easier to exploit also information from many less expensive sensors, merge these data, and then compute an estimation of the position of the robot. This is what localization by sensor fusion does.

Mobile robot localization is an instance of the general localization and sensor fusion problems, which is the most basic perceptual problem in robotics [4]. Sensor fusion is the

merging of sensory data from different sources such that the resulting estimation is more accurate than the one which could be obtained individually from these sources.

Literature presents lots of algorithms for localization and sensor data fusion techniques. A large part of these algorithms is based on Bayesian probability theory. This approach leads to represent the information on the position of the robot through a Probability Density Function (PDF). Every time some new information are gained by the robot through its sensors the PDF is updated. If the robot moves for some time without collecting information from its exteroceptive sensors (i.e. the robot uses laser range finders to localize itself through the measurements of range and bearing of reflective markers, but unfortunately in the robot's neighborhood no markers can be found), the uncertainty on its position grows. This is due to the odometry. Actuators, usually, are unpredictable. Motors, for example, are subject to wear and mechanical failure. Motors also need gear-boxes in order to increase the output torque and be able to move a heavy robot, and every mechanical component which is inserted in the kinematic chain adds uncertainty in its operation due to mechanical couplings.

Sensors are also limited in what they perceive. For example the visual range and the resolution of a camera is limited by the nature of its transducers (the CCD components). A sensor, in order to be used on-board a robot, must be connected to a computer which works in a discrete world: this means that the information provided by the transducer must be discretized, and this may cause loss of information. Finally sensors too are subject to wear (for instance the motor that rotate the laser scanner and its encoder) and can break.

Another source of uncertainty is caused by model's uncertainty. The mathematical model representing the robot is approximated. These approximations derives from several factors. First the accuracy of model's parameters cannot be infinite. Also the discretization of the model introduces an approximation and then generates uncertainty.

Of course the admissible level of uncertainty changes according the application domain of the robot. A dust cleaner robot doesn't need the same accuracy of an AGV. Similarly a quadcopter used for search and rescue tasks may possibly need high accuracy in computing the position of obstacles and then avoid them, in order to pass through narrow passages.

The admissible level of uncertainty is also connected with the overall cost of a robot. A service robot may not carry expensive sensors, so its accuracy cannot be as high as the one of an AGV which has to place its load with millimeter precision in the warehouse.

Sensor fusion and localization techniques are used to overcome some of the previously listed problems and to provide better information on the robot's state.

3.2 Kalman Filtering techniques - The Kalman Filter and the Extended Kalman Filter

In this section a brief introduction on the Kalman filtering [5] techniques will be provided. In order to better understand the calculus, in Appendix A some basics on the Bayesian probability theory is provided.

3.2.1 The Kalman Filter

The Kalman Filter (KF) [6, 7] and the Extended Kalman Filter (EKF) belongs both to a subclass of Bayes Filters (see [8]), the Gaussian filters. This family of recursive state estimators shares the assumption that the belief¹ is represented through a multivariate Gaussian (aka normal) distribution. Let $\vec{x} \in \mathbb{R}^n$ be the state vector variable, $\vec{\mu} \in \mathbb{R}^n$ be the mean and $\Sigma \in \mathbb{R}^{n \times n}$ be the covariance square matrix. The density function is defined with:

$$p(\vec{x}) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu}) \right\} \quad (3.1)$$

The mean and the covariance are respectively the first and the second moments of a probability distribution, so this parameterization is called the *moments parameterization* of a Gaussian function. Consider the following system:

$$\begin{cases} \vec{x}_k = \mathbf{A}_k \vec{x}_{k-1} + \mathbf{B}_k \vec{u}_k \\ \vec{z}_k = \mathbf{C}_k \vec{x}_k \end{cases} \quad (3.2)$$

The Kalman Filter represents beliefs by the moment parameterization: at the time step k the belief is represented by the first moment of a probability distribution, the mean $\vec{\mu}_k$, and the second moment, the covariance matrix Σ_k . Posteriors are also Gaussian if:

1. The state transition probability, represented by $p(\vec{x}_k | \vec{u}_k, \vec{x}_{k-1})$ where $\vec{u}_k$ is the input vector, is a linear function and its components, nominally the state vector and the input vector, are affected with Gaussian noise. This can be summarized with

$$\vec{x}_k = \mathbf{A}_k \vec{x}_{k-1} + \mathbf{B}_k \vec{u}_k + \vec{\varepsilon}_k \quad (3.3)$$

where $\mathbf{A}_k$ is the state matrix, $\mathbf{B}_k$ is the input matrix and $\vec{\varepsilon}_k$ is an additive Gaussian noise which affects the state transition. This means that the posterior probability is given by:

¹A belief reflects the robot's internal knowledge about the state of the environment. Therefore the true state of the robot must be distinguished from its internal belief with regard to the state [4]

$$p(\vec{x}_k | \vec{u}_k, \vec{x}_{k-1}) = \det(2\pi\mathbf{R}_k)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\vec{x}_k - \mathbf{A}_k \vec{x}_{k-1} - \mathbf{B}_k \vec{u}_k)^T (\mathbf{B}_k \mathbf{R}_k \mathbf{B}_k^T)^{-1} (\vec{x}_k - \mathbf{A}_k \vec{x}_{k-1} - \mathbf{B}_k \vec{u}_k) \right\} \quad (3.4)$$

where $\mathbf{R}_k$ is the covariance matrix representing the noise $\vec{\varepsilon}_k$.

2. The measurement probability, represented with $p(\vec{z}_k | \vec{x}_k)$ where $\vec{z}_k$ is the measurement vector, is a linear function and its components, nominally the state vector and the measurement vector, are affected by Gaussian noise. This can be summarized with

$$\vec{z}_k = \mathbf{C}_k \vec{x}_k + \vec{\omega}_k \quad (3.5)$$

where $\mathbf{C}_k$ is the output matrix and $\vec{\omega}_k$ is an additive Gaussian noise which affects the measurement. This leads to defining the measurement probability density function:

$$p(\vec{z}_k | \vec{x}_k) = \det(2\pi\mathbf{Q}_k)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\vec{z}_k - \mathbf{C}_k \vec{x}_k)^T \mathbf{Q}_k^{-1} (\vec{z}_k - \mathbf{C}_k \vec{x}_k) \right\} \quad (3.6)$$

where $\mathbf{Q}_k$ is the covariance matrix representing the noise $\vec{\omega}_k$.

3. The initial belief $bel(\vec{x}_0)$ has a Gaussian distribution, nominally:

$$bel(\vec{x}_0) = p(\vec{x}_0) = \det(2\pi\mathbf{\Sigma}_0)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\vec{x}_0 - \vec{\mu}_0)^T \mathbf{\Sigma}_0^{-1} (\vec{x}_0 - \vec{\mu}_0) \right\} \quad (3.7)$$

The algorithm for the Kalman Filter is showed in Alg. 1.

Algorithm 1

Kalman Filter ($\vec{\mu}_{k-1}, \mathbf{\Sigma}_{k-1}, \vec{u}_k, \vec{z}_k$)

- 1: $\hat{\mu}_k = \mathbf{A}_k \vec{\mu}_{k-1} + \mathbf{B}_k \vec{u}_k$
 - 2: $\hat{\Sigma}_k = \mathbf{A}_k \mathbf{\Sigma}_{k-1} \mathbf{A}_k^T + \mathbf{B}_k \mathbf{R}_k \mathbf{B}_k^T$
 - 3: $\mathbf{K}_k = \hat{\Sigma}_k \mathbf{C}_k^T (\mathbf{C}_k \hat{\Sigma}_k \mathbf{C}_k^T + \mathbf{Q}_k)^{-1}$
 - 4: $\vec{\mu}_k = \hat{\mu}_k + \mathbf{K}_k (\vec{z}_k - \mathbf{C}_k \hat{\mu}_k)$
 - 5: $\mathbf{\Sigma}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \hat{\Sigma}_k$
-

The Kalman Filter is a recursive algorithm. The k^{th} step is discussed. The symbol $\vec{\mu}$ represents the state estimation returned by the Kalman Filter, which is substantially different to the state $\vec{x}$. In line 1 the time propagation is computed through the model of the system. In an ideal world where exists no uncertainty this equation would be

sufficient for the state estimation. Nevertheless, in a real world, this is not true: in fact the unpredictable effect of the noise must be considered. This is done in line 2, where the uncertainty is taken into account with the covariance Σ_k , where the process noise effect, represented with the covariance matrix $\mathbf{R}_k$, is added to the system uncertainty. By multiplying matrix $\mathbf{R}_k$ on the left with $\mathbf{B}_k$ and on the right with $\mathbf{B}_k^T$ the error of each input element is projected on the state space. These first two equations represent the prediction step of the Kalman Filter.

In line 3 the Kalman gain $\mathbf{K}_k$ is computed according to the covariance of the system which the model was propagated and the covariance of the measurement that will be introduced to correct the state prediction. This is done in line 4, where the error $(\vec{z}_k - \mathbf{C}_k \hat{x}_k)$ is computed according to what the sensors are measuring and what they are supposed to measure according to the predicted state. This error is then multiplied to the Kalman gain and added to the previously computed state prediction. Finally the system covariance Σ_k is updated.

Line 3 highlights also that as the measurement error covariance approaches to zero (i.e. the sensor which provides the measurement is very accurate) the gain $\mathbf{K}_k$ weights the residual more heavily and vice versa:

$$\begin{aligned} \lim_{\mathbf{Q}_k \rightarrow \mathbf{0}} \mathbf{K}_k &= \mathbf{C}_k^{-1} \\ \lim_{\mathbf{Q}_k \rightarrow \infty} \mathbf{K}_k &= \mathbf{0} \end{aligned} \quad (3.8)$$

On the other hand, if the prediction covariance matrix $\hat{\Sigma}_k$ approaches to zero, the $\mathbf{K}_k$ weights the residual less heavily and vice versa:

$$\begin{aligned} \lim_{\Sigma_k \rightarrow \mathbf{0}} \mathbf{K}_k &= \mathbf{0} \\ \lim_{\Sigma_k \rightarrow \infty} \mathbf{K}_k &= \infty \end{aligned} \quad (3.9)$$

The Kalman Filter algorithm is also summarized in Fig. 3.1.

3.2.2 The Extended Kalman Filter

Usually, state transitions and measurement functions are not linear. For example the model for a differential drive mobile robot proposed in Sec. 2.1.1 is non-linear. The classical Kalman Filter cannot be used for non-linear model. It is here that comes into play the Extended Kalman Filter (EKF).

The Extended Kalman Filter relaxes the assumption stated previously in Sec. 3.2.1: this time we consider non-linear transition functions and non-linear measurement functions. The considered system is:

$$\begin{cases} \vec{x}_k = f(\vec{x}_{k-1}, \vec{u}_k) + \vec{\epsilon}_k \\ \vec{z}_k = g(\vec{x}_k) + \vec{\omega}_k \end{cases} \quad (3.10)$$

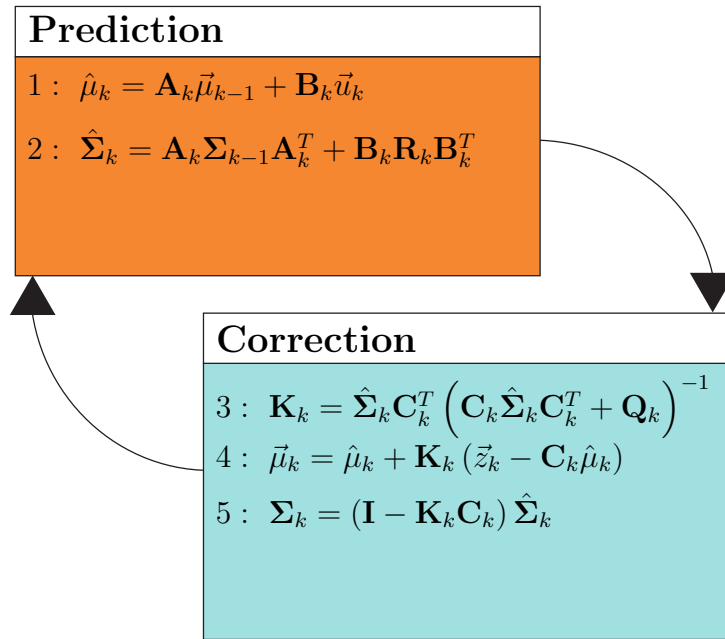


Figure 3.1: Schematic representation of the Kalman Filter prediction/correction iteration

where $f(\vec{x}_{k-1}, \vec{u}_k)$ is the state transition function and $g(\vec{x}_k)$ is the output function. $\vec{x}_k$, $\vec{u}_k$, $\vec{z}_k$, $\vec{\varepsilon}_k$ and $\vec{\omega}_k$ represent, respectively, the state vector, the input vector, the measurement vector, the input noise and the measurement noise.

The key idea behind the EKF is the linearization via a first order Taylor expansion. The linearization process approximates locally the non-linear function to a linear one which is tangent to the first at the mean of the Gaussian. The choice of the argument, nominally the linearization point, of both the non-linear function and of its linearized version is crucial: for state transition functions the most likely point is the mean of the posterior. In practice the function is approximated by its value at $\vec{x}_{k-1}$ and $\vec{u}_k$.

$$\Phi_k = \frac{\partial f(\vec{x}_{k-1}, \vec{u}_k)}{\partial x_{k-1}} \quad (3.11)$$

The matrix resulting from the linearization procedure, Φ_k , is defined as follows:

$$\Phi_k = \{\phi_{i,j}\} \text{ s.t. } \phi_{i,j} = \left. \frac{\partial f_i}{\partial x_j} \right|_{x=\vec{x}_{k-1}, u=\vec{u}_k} \quad (3.12)$$

Matrix Φ_k is commonly called Jacobian.

Similarly, a second Jacobian is obtained by linearizing the state transition function by the input:

$$\Psi_k = \frac{\partial f(\vec{x}_{k-1}, \vec{u}_k)}{\partial u_k} \quad (3.13)$$

The matrix resulting from the linearization procedure, Φ_k , is defined as follows:

$$\Psi_k = \{\psi_{i,j}\} \text{ s.t. } \psi_{i,j} = \left. \frac{\partial f_i}{\partial u_j} \right|_{x=x_{k-1}, u=u_k} \quad (3.14)$$

It is important to note that a fundamental flaw of the Extended Kalman Filter is that the distribution of the various random variables are no longer normal after undergoing their non-linear transformations. The EKF is an ad-hoc state estimator that approximates the optimality of the Bayes' rule by linearization via a first order Taylor expansion [5].

The Gaussian state transition probability density function is approximated with

$$p(\vec{x}_k | \vec{u}_k, \vec{x}_{k-1}) \approx \det(2\pi \mathbf{R}_k)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} [\vec{x}_k - f(\vec{u}_k, \vec{\mu}_{k-1}) - \Phi_k(\vec{x}_{k-1} - \vec{\mu}_k)]^T (\Psi_k \mathbf{R}_k \Psi_k^T)^{-1} [\vec{x}_k - f(\vec{u}_k, \vec{\mu}_{k-1}) - \Phi_k(\vec{x}_{k-1} - \vec{\mu}_k)] \right\} \quad (3.15)$$

As it was done previously, also the output function $g(\vec{x}_k)$ is linearized:

$$\Gamma_k = \frac{\partial g(\vec{x}_k)}{\partial x_k} \quad (3.16)$$

The matrix resulting from the linearization procedure, Γ_k , is defined as follows:

$$\Gamma_k = \{\gamma_{i,j}\} \text{ s.t. } \gamma_{i,j} = \left. \frac{\partial g_i}{\partial x_j} \right|_{x=\hat{x}_k} \quad (3.17)$$

The Gaussian output function probability density function is approximated with:

$$p(\vec{z}_k | \vec{x}_k) \approx \det(2\pi \mathbf{Q}_k)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} [\vec{z}_k - g(\hat{\mu}_k) - \Gamma_k(\vec{x}_k - \hat{\mu}_k)]^T \mathbf{Q}_k^{-1} [\vec{z}_k - g(\hat{\mu}_k) - \Gamma_k(\vec{x}_k - \hat{\mu}_k)] \right\} \quad (3.18)$$

The EKF algorithm is very similar to the (linear) KF algorithm (see 2).

Algorithm 2

Extended Kalman Filter ($\vec{x}_{k-1}, \Sigma_{k-1}, \vec{u}_k, \vec{z}_k$)

- 1: $\hat{\mu}_k = f(\vec{\mu}_{k-1}, \vec{u}_k)$
 - 2: $\hat{\Sigma}_k = \Phi_k \Sigma_{k-1} \Phi_k^T + \Psi_k \mathbf{R}_k \Psi_k^T$
 - 3: $\mathbf{K}_k = \hat{\Sigma}_k \Gamma_k^T (\Gamma_k \hat{\Sigma}_k \Gamma_k^T + \mathbf{Q}_k)^{-1}$
 - 4: $\vec{\mu}_k = \hat{x}_k + \mathbf{K}_k [(\vec{z}_k - g(\hat{\mu}_k))]$
 - 5: $\Sigma_k = (\mathbf{I} - \mathbf{K}_k \Gamma_k) \hat{\Sigma}_k$
-

The prediction step of the EKF, unlike the KF, exploits the non-linear function to compute the predicted state of the system (line 1). In line 2 the two Jacobian matrices of the non-linear $f(\vec{x}_{k-1}, \vec{u}_k)$ are computed and used to propagate the uncertainty represented with the covariance matrix. As the KF, in line 3 the Kalman gain is computed with the Jacobian of the output function. Then the correction on the estimation is performed in line 4 and on the system covariance in line 5.

Chapter 4

Robomower

In order to have a competitive price, commercial robotic lawnmowers are equipped with very few and very cheap sensors that limit their localization and navigation capabilities. Thus, mowing is achieved through a random navigation algorithm. This makes the execution time high and it poses several quality problems since it can lead to an uneven mowing. In this chapter we propose a novel navigation strategy that allows to localize the robot in the field and to plan a mowing path that guarantees an efficient covering of the garden and a lower execution time. This is done using low cost sensors as RFID tags and radio frequency devices in order to remain competitive in the market. This work, done in the framework of the DiRò project, is the result of a research carried out with EMAK s.p.a., an international producer of gardening tools, whose goal is to develop the next generation of autonomous lawnmowers.

4.1 Introduction

Nowadays mobile robots are a clear example of how robots can facilitate our daily life. Autonomous vacuum cleaners, swimming pool cleaners and lawnmowers are just a few examples of simple devices that can take care of some of our activities raising us from the task of doing them by ourselves.

In the last decade, autonomous lawnmowers are becoming increasingly popular. These robots release us from the boring task of mowing a garden and they improve home safety since many accidents per year take place while manually mowing a garden as reported in [9].

Despite of their increasing diffusion, autonomous lawnmowers available in the market are still rather inefficient. In fact, in order to sell them at a competitive price, these robots are equipped with very few sensors: a rain sensor, for sending the lawnmower at its docking station when it starts raining, a lift sensor, for stopping the blade when the robot is lifted up by a person or if it tips over, and an ultrasonic sensor or a bumper, for

obstacles detection and avoidance. In order to constraint the robot to stay in the garden, in most of the cases a boundary wire, namely a current-carrying wire fixed to the ground with pegs, is buried along the perimeter of the area to be mowed. The current generates a sort of magnetic wall that the robot senses by an inductive sensor. If the lawnmower tries to cross the wall, the robot senses it, then it turns of a random angle and then it restarts going straight again.

It is clear that with such poor sensor equipment, it is impossible for the lawnmower to localize itself and to implement an efficient navigation strategy for covering the garden. In fact, mowing is achieved by means of a random navigation strategy which is very similar to the one implemented by robotic vacuum cleaners, where magnetic walls play the role of real walls. This navigation strategy ensures a full coverage of the area to be mowed if the robot is left operating for an amount of time big enough but it has two main drawbacks. From an aesthetic point of view, it is possible that some areas of the garden are over-mowed (and unnecessarily ruined by the many passages of the wheels) and that some other areas are under-mowed. In other words, the non uniformity of the navigation strategy leads to an unevenly mowed garden. From an efficiency point of view, random navigation is unnecessarily redundant and it takes a lot of time for guaranteeing the complete coverage of the garden. This unnecessarily deteriorates the motors and leads to early maintenance.

In order to obtain an efficient and uniform mowing, it is necessary to allow the lawnmower to localize itself in the garden and to implement an efficient coverage strategy. This would guarantee a lower execution time, a better aesthetic result and a longer time before maintenance. Of course, in order to remain competitive on the market, the price of the components necessary for achieving such a navigation system should be as low as possible.

The problem of localizing a mobile robot in an outdoor environment has long history in the literature. Vision has been proposed as a possible (and cheap) solution for solving the localization problem (see e.g. [10,11]). These methods can provide a very good estimation of the robot position, but in our case the robot is moving in a very dirty environment. The rotating cutting blade of the lawnmower could throw mud, dust and cut grass on the lens of the camera, creating partial or even total occlusions and making the vision system useless.

In [12] and in [13] a Kalman filter and a Particle Filter (PF) based approaches to localize a robot operating in an outdoor environment using DGPS (Differential Global Positioning System) and INS (Inertial Navigation System) are proposed, respectively.

These methods require sensors that are too expensive for being used in autonomous lawnmowers. Furthermore, these localization methods could become inaccurate when the satellite signal is weak, such as when the robot is under trees or near walls. A DGPS based localization method is also proposed in [14]: here the position information returned by the satellite is merged with odometric information and with orientation data coming from a magnetic compass. In addition to being expensive, this method could lead to wrong localization results due to the electromagnetic interference generated by motors, batteries and other electronic devices mounted on the robot itself with the compass.

In [15] a laser scanner is exploited for localizing a robot in a dynamic outdoor environment. The main drawback of this, and of most of the mainstream localization strategies, is that, in order to obtain good performance, they exploit sensors that are too expensive to be used in a commercial lawnmower. Moreover, when using laser scanners for outdoor applications, some artificial landmarks may need to be installed in the working area which, in this case, is a garden. Such an intervention may be unacceptable for aesthetic reasons. Recently, in [16] a novel autonomous lawnmower has been proposed. This commercial robot is able to mow with very high accuracy the lawn of greens in golf courses, but its use is limited to that application: first the proposed robot has very big dimensions ($1.5\text{ m} \times 0.8\text{ m} \times 1.2\text{ m}$), then it requires the installation of at least three big infrared beacons which allow the robots to localize itself through trilateration. Like in the laser scanner case this approach is not viable for aesthetic reasons. In a garden, natural landmarks could be used as well, like trees, but the localization algorithm, based on the map of these landmarks, could become inefficient if the garden undergoes changes. If a tree gets cut or a hedge is pruned the map changes, and the localization algorithm needs a reset.

Good localization results can also be achieved using very cheap sensors. RFID (Radio Frequency Identification [17]) based localization techniques have been proposed. Due to the highly non-Gaussian noise which characterizes these sensors, usually PF based methods are successfully used to estimate the position of a mobile robot [18–20]. Accurate results, however, can only be obtained by using a large number of particles and by significantly increasing the computational burden of the algorithm. For service robot applications, like autonomous lawnmowers, the available computational power is limited for cost reason. Thus, it is not possible to implement computationally demanding algorithms such as the PF. [21, 22] propose a triangulation based localization method, where many RFID tags are densely present on the floor. In gardening applications this approach is unfeasible because it would require embedding a lot of RFID tags in the lawn.

In [23] a very efficient Kalman Filter based localization method is proposed. In particular the paper faces the localization problem by proposing two methods, a Quantized Kalman filter and a Constrained Kalman filter, and compares the two algorithms with a Particle Filter localization approach. Results are very good, but the proposed methods require that at least one RFID tag is detected by the robot.

Another kind of very inexpensive sensors are the wireless beacons [24] which can send an RF (Radio Frequency) signal that can be read by another RF sensor that can return information about the distance of the beacon. These measurements, even if very noisy, can be used to localize the position of the source of the RF signal. In [25] a statistical based method and a neural network method are proposed for estimating the distance and the position of nodes in a WSN (Wireless Sensor Network) using the RSSI (Received Signal Strength Indicator) provided by the wireless devices. In [26] an accurate PF method, based on the trilateration of the RSSI measurement given by three Bluetooth nodes, is proposed. A PF method is proposed as well in [27] with also a method to reduce the influence of multi-path and of the radio irregularities, in order to obtain a more reliable RSSI reading. These two methods, despite the reliable results in localizing the robot, are very computationally demanding. Also RSSI trilateration is not feasible for a lawnmower application: RF antennas need a power supply to work and using batteries or a cable connecting all the RF devices to provide them power could not be viable or convenient for such an application. Alternatively the robot could use only odometry (and a gyroscope to prevent drift) and the RFID system in order to obtain the motion model, but the so-built model could become inaccurate in case of displacement of the RFID tags. Also the mowing of the robot could be very inaccurate during the learning process.

The goal of this work, described in this chapter, is to design a navigation strategy that guarantees a uniform coverage of the garden and a small execution time while keeping the cost of the robotic lawnmower competitive. We show that low cost and alternative sensing techniques as RFID tags and wireless beacons can be easily installed in a commercial setup for robotic lawnmowers and we propose a novel algorithm based on Kalman filters that is able to localize the lawnmower by properly merging the information coming from RFID tags dug along the perimeter of the garden, the RSSI provided by a wireless beacon installed on the home station of the robot, the angular speed coming from three gyros and the odometric data. The output of the localization system is exploited for achieving an efficient and uniform coverage. The results are validated through simulations and experiments on the field which prove that the solution proposed in this chapter is effective for autonomous mowing systems and it improves the efficiency achievable today. This

work is the result of a research carried on with EMAK s.p.a., an international producer of gardening tools, whose goal is to develop the next generation of autonomous lawnmowers and it extends [28] and in [29] with a more extensive simulations and experiments.

The contribution of this work is to show how alternative low-cost sensors can be easily embedded in a robotic lawn mowing system and can be exploited for significantly improving the coverage performance and decreasing the mowing time. Furthermore, a decrease of the mowing time implies an increase of the life of the lawnmower since, at each cut of the garden, mechanical components are less subject to wear and the system needs less energy, preserving batteries. Finally, the low cost of alternative sensors with respect to more standard ones (e.g. laser scanner) allow to increase the performance while keeping the cost of the system competitive.

The chapter is organized as follows: in Sec. 4.2 the hardware setup of the lawn mowing system is described and in Sec. 4.3 the RFIDs and the wireless beacons are characterized. In Sec. 4.4 the localization algorithm and the navigation strategy are proposed and in Sec. 4.5 and in Sec. 4.6 these are validated through simulations and experiments. Finally in Sec. 4.7 conclusions are drawn.

4.2 Hardware Setup

In this section the main components and the setup of the robotic lawn mowing system are described.

We consider an autonomous lawnmower with a differential drive kinematic structure, the most commonly used for commercial autonomous lawnmowers. Each of the actuated wheels is provided with Hall's sensors that allow to extract an odometric information about the speed of the wheels of the robot. Furthermore, the robot is equipped with a three degrees of freedom gyroscope, providing information about the angular velocity. The obstacle detection is accomplished with infrared optical sensors, able to provide information about the presence and the distance of obstacles in the surroundings of the robot. Further in this chapter we will describe how these data can be included into an occupancy grid map, which enables the robot to implement an obstacle avoidance strategy.

In order to delimit the working area of the robot we put equally spaced pegs on the boarder of the working area itself. Inside of each peg, an RFID tag is incorporated. The RFID tags are programmed to have a unique identifying code and each tag is associated to a limited area where it can be detected from. The robot is equipped with an RFID antenna that allows to detect and identify the RFID tags. The role of these tags is to

provide the robot with a reset mechanism. When the robot detects and identifies a tag, it knows with a quite good precision where it is and it can use this information for correcting the estimate of its pose.

Nevertheless, the reset provided by the RFID may not be sufficient if the garden to be mowed is large (see Fig. 4.1). In this case, the robot could be wandering in the middle of the lawn using only odometric information and gyroscope measurements for localizing itself and it may take too much time for detecting an RFID tag for resetting its localization error. Thus, we place on the docking station of the robot (that is located in a known position, usually in a corner of the garden) an RF beacon and we equip the robot with an RF sensor that can sense the RSSI of the signal emitted. The robot can exploit this exteroceptive information for improving its localization capabilities and keeping the pose estimation error low when it is far from the RFID tags. Notice that if the garden is very big and/or if it has a very irregular shape, it may be necessary to place more beacons.

In this work only UHF (Ultra-High Frequency) RFID devices are used, since LF (Low Frequency) devices has a very limited range (maximum 10 *cm*) and HF (High Frequency) devices have a reading region of the antenna which is strongly influenced by metallic objects which are behind the antenna (the presence of a metallic objects behind the antenna reduces the reading range to an area of dimensions comparable to the distance of the metallic object itself to the antenna).

4.3 Characterization of the sensors

4.3.1 RFID sensor characterization

It has been shown in [30] that the reading region of an RFID tag in a dry indoor environment has the shape of an ellipse. Nevertheless, in our setting RFID tags are placed outdoor and it can happen that they are covered with grass, mud, dust or leaves. Of course, in case of rain, RFID can also get wet with water, but in this case the robot shouldn't work, because the lawn should not be mowed if the grass is wet, or the mowing could ruin the lawn.

The goal of this section is to characterize the reading region that can be achieved by several tag-antenna pairs in an outdoor environment and subject to all the realistic perturbations that can be found in a garden.

In this series of experiments different RFID antennas, readers, and tags were tested. The characterization was done not only to identify the combination of devices with the most stable and well shaped reading region, but also to minimize the power consumption

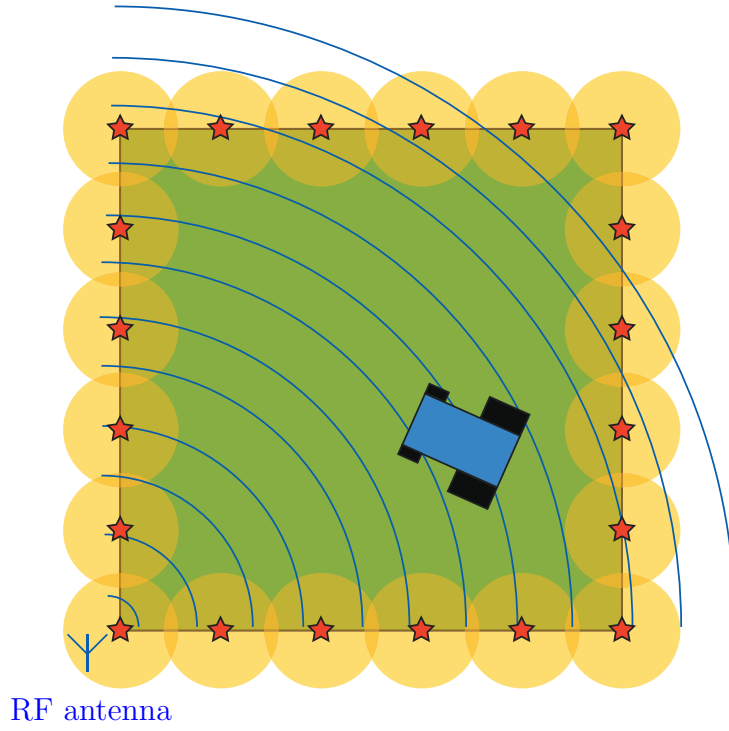


Figure 4.1: Representation of the reading area (the yellow shaded area) of the RFID tags (put in known position and represented as red stars) and of the RF signal emitted by the RF antenna

and the overall cost of the RFID sensor equipment. For clarity of exposition we use acronyms for indicating antennas, tags and readers. The antennas we tested are:

- Linearly polarized UHF RFID antenna, frequency: 869.5 MHz , dimensions: $120\text{ mm} \times 120\text{ mm} \times 10\text{ mm}$ (CAEN WANTENNAX004): **ANT1**
- Linearly polarized UHF RFID antenna, frequency: $865 \div 870\text{ MHz}$, dimensions: $80\text{ mm} \times 22\text{ mm} \times 0.5\text{ mm}$ (CAEN WANTENNAX009): **ANT2**
- Circularly polarized UHF RFID antenna, frequency: $860 \div 960\text{ MHz}$, dimensions: $245\text{ mm} \times 235\text{ mm} \times 40\text{ mm}$ (CAEN WANTENNAX005): **ANT3**
- Circularly polarized UHF RFID antenna (Antenna prototype): **ANT4**
- Linearly polarized UHF RFID antenna, Gain: $860 \div 870\text{ MHz}$, dimensions: $200\text{ mm} \times 180\text{ mm} \times 32\text{ mm}$ (ASE_FAGA201 antenna): **ANT5**

We tested both linearly and circularly polarized antennas in order to determine the one providing the best detection area and radiation pattern. Ideally, for this application, the detection should not be influenced by the orientation of the RFID tag, since the robot could approach the RFID tag from any direction and it should be able to detect the presence of the tag independently of its orientation.

During the characterization experiments we tested different tags to identify the one providing the widest and best shaped reading region, according to our application, and with the smallest dimensions, in order to insert it inside a peg that will be buried in the ground. The tested RFID tags are:

- Big dimensions UHF RFID tag ($97 \times 15 \text{ mm}$), frequency: $860 \div 960 \text{ MHz}$ (UPM RAFLATAC Wet Inlay RAFSEC short dipole): **TAG1**;
- Big dimensions UHF RFID tag ($95 \times 7.2 \text{ mm}$), frequency: $860 \div 960 \text{ MHz}$ (LAB ID Inlay UH331): **TAG2**;
- Small dimensions UHF RFID tag ($40 \times 15 \text{ mm}$), frequency: $840 \div 960 \text{ MHz}$ (LAB ID Inlay UHI4015): **TAG3**
- Square shaped omnidirectional UHF RFID tag ($40 \times 40 \text{ mm}$), frequency: $840 \div 960 \text{ MHz}$ (LAB ID Inlay UH3D40): **TAG4**

The interface with the computer of the robot is provided by the RFID reader. It is also responsible of the power provided to the antenna and of the anti-collision algorithm [31,32] for reading multiple RFID tags at the same time. The tested readers are:

- UHF RFID reader, frequency: $865.6 \div 867.6 \text{ MHz}$, dimensions: $42 \text{ mm} \times 60 \text{ mm} \times 6.3 \text{ mm}$, power consumption: 5 W (CAEN A528 Reader): **RDR1**
- Low power consuming compact UHF RFID reader, frequency: $865.6 \div 867.6 \text{ MHz}$, dimensions: $25 \text{ mm} \times 40 \text{ mm} \times 6 \text{ mm}$, power consumption: $1.6 \div 1.3 \text{ W}$ (CAEN Quark reader): **RDR2**
- UHF RFID reader, frequency: $860 \div 940 \text{ MHz}$, dimensions: $123 \text{ mm} \times 94 \text{ mm} \times 29 \text{ mm}$, power consumption: 3.6 W (ASE RFID DESKTOP READER UHF): **RDR3**

First a chalk support for the RFID reader and antenna was created with a 3D printer (see Fig. 4.2). The choice of creating the support in chalk was taken according to the

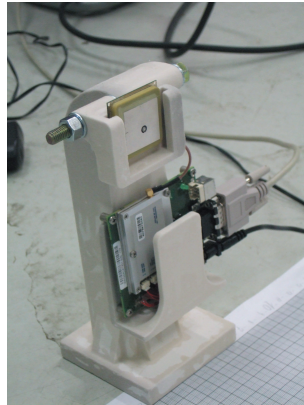


Figure 4.2: The support used for the characterization of the RFID devices

fact that a metal support could alter the electromagnetic field generated by the antenna thus altering its reading region and the result of the experiments. The support was then waterproofed with a plastic resin: this was done because the liquid absorbed by the chalk could also alter the result of the experiments.

The experiments were made by putting the RFID tag inside a peg and then by dipping it into the mud of the garden/grass/leaves. Then we measured the reading efficiency H defined as

$$H = \frac{Tags}{Acqs} \quad (4.1)$$

where $Tags$ is the number of read tags per unit time and $Acqs$ is the number of query done by the reader.

Figure 4.3 reports some of the most relevant radiation patterns obtained during the experiments. The orange shaded area represents the set of position from where the RFID tags can be read from the antenna. Circularly polarized antenna (Fig. 4.3b, Fig. 4.3c, Fig. 4.3d) can read RFID tags with any orientation, but the radiation pattern is more irregular and requires more energy in order to obtain the same dimension of the reading region than the one of the linear polarized. Linearly polarized antennas generally have a radiation pattern with a regular and symmetrical shape but the readability of an RFID tag depends on its orientation with respect to the antenna. Experiments highlighted that this issue can be bypassed by using omnidirectional tags (Fig. 4.3a). Furthermore, linear polarized antennas proved to be more reliable in outdoor environments, while their radiation patterns are less likely to change in harsh conditions.

Thus, we have chosen to use linearly polarized antennas with omnidirectional tags. The reader did not seem to influence the reliability of the tag detection and the shape of the reading region; we have chosen to use CAEN A528 Reader (RDR1). With the adopted configuration, the reading region is approximated very well with an ellipse with the major axis (≈ 100 mm) along the wire and the minor axis (≈ 40 mm).

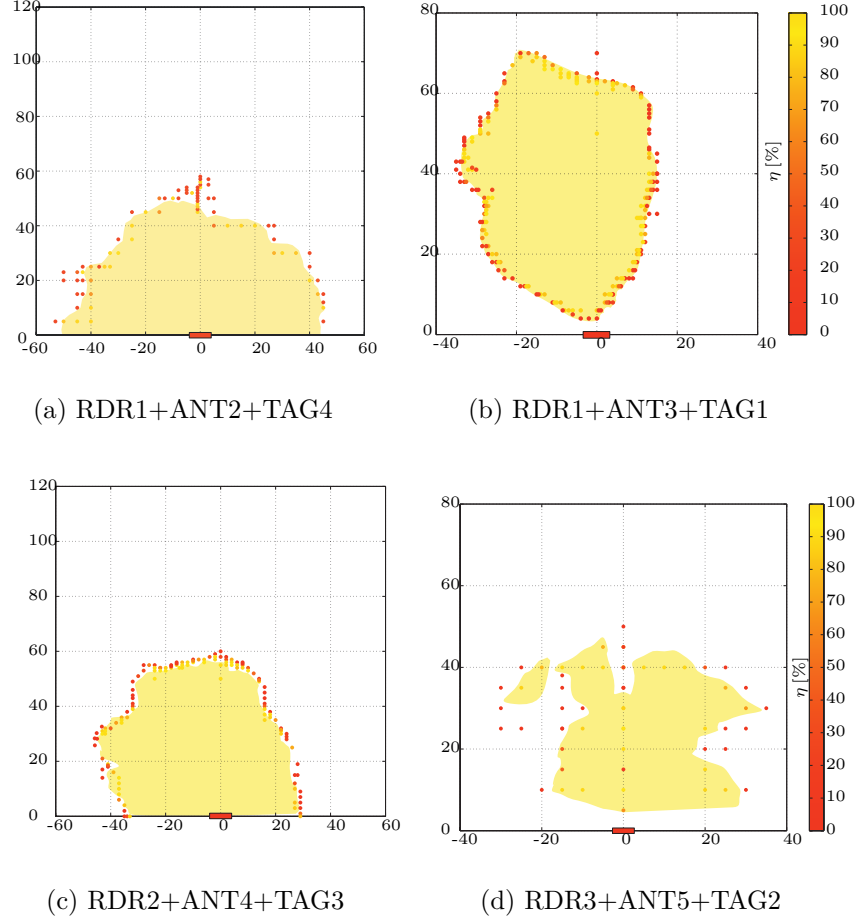


Figure 4.3: Some of the experimentally obtained radiation patterns. The orange rectangle represents the position of the antenna. Dimensions are in centimeters. The yellow shaded area represents the set of position from where there is a reading probability greater than the 98%. Coloured dots represent the RFID tags reading frequency on the border of the reading area.

4.3.2 The wireless communication model

The model of the wireless communication channel that we use in the proposed algorithm is reported in Eq. 4.2, obtained from the Friis' transmission equation [24]:

$$P_{r,i} [dBm] = P_t [dBm] + K [dB] - 10 \cdot \gamma \cdot \log_{10} \left(\frac{d_i}{d_0} \right) + \Psi_i \quad (4.2)$$

where $P_{r,i} [dBm]$ is the power received by the on-board antenna, $P_t [dBm]$ the power transmitted by the beacon and $K [dB]$ is a constant that depends on the frequency of the transmission devices and on the environment. The constant γ is the path loss coefficient. Both $K [dB]$ and γ can be obtained experimentally (see [24]). The constant d_0 is the reference distance at which we calculated the constant K and d_i is the distance between the fixed and the mobile antennas. The zero mean Gaussian noise Ψ_i is used to approximate the disturbance (like multi-path and shadowing effects) in the transmission. The variance of this measurement was obtained by characterizing the sensor in the working environment (the lawn).

During the characterization experiments we noticed that the variance of the model changes according to the relative orientation of the on-board antenna with respect to the orientation of the antenna of the RF beacon, as also remarked in [25]. In order to take into account of this behavior, we stored the experimentally obtained variance of the model with the two antennas in different relative positions and orientations in a hash-table structure and then in the algorithm, explained later in this chapter, we change the considered variance according to the orientation of the robot. In addition, to further reduce the noise associated to the RSSI measurement we apply the pre-filtering method illustrated in [33].

4.4 The Navigation strategy

In this section we will illustrate the navigation strategy we are proposing for a differential drive robotic lawnmower with the sensory equipment described in Sec. 4.2. The navigation system consists of a localization algorithm and of a simple planner that guarantees a complete coverage of the garden once the pose of the robot is known.

Several data structures were computed offline from the knowledge of the garden and of the sensors installed. First of all we discretize the working area of the robot using an occupancy grid map, initializing the cells “to be mowed” with low cost values. The dimensions of the cells are chosen to be a little smaller than the dimension of the blade mounted under the robot, in order to ensure a small overlap when covering the field. Furthermore we set to infinity the cost of those cells which are occupied by objects, that can be detected by the robot thanks to the infrared optical sensors the lawnmower is equipped with. In this chapter we do not report the model of this sensor, which can be

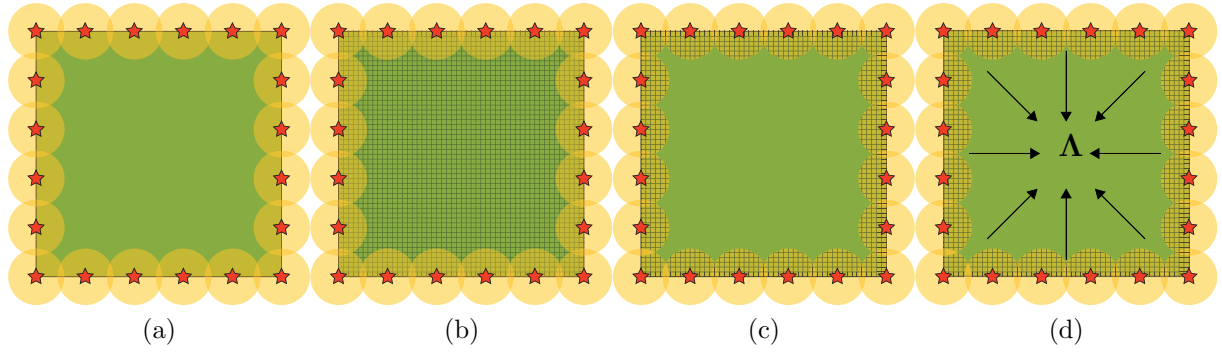


Figure 4.4: Scheme summarizing the off-line setup: Figure 4.4a represents the computation of the reading regions of the RFID tags. Figure 4.4b represents a discretized set of position of the working area of the robot, while Fig. 4.4c points out the intersection of the cells and of the reading regions. Figure 4.4d represents the storage of the cells into an hash table structure.

easily found in the literature as a range finder sensor (e.g. [4], [34]). Briefly, the term “occupancy” describes not only the presence of objects in the discretized world of the robot, but is also used for distinguish those portions of the lawn which have already been mowed by the robot. Further in this section also the navigation strategy will be explained.

Exploiting the fact that the positions and the reading areas of the RFID tags are known, according to the experiments described in Sec. 4.3.1, we build a data structure that will be useful in the localization algorithm. Let A_r be the reading area of the r^{th} RFID tag, namely the set of all “poses” from which the r^{th} RFID tag can be detected, and let $A = \cup_{r=1}^n A_r$. Let $\mathcal{C}$ be the configuration space of the robot. The set $A \cap \mathcal{C}$ is discretized with a much smaller granularity than the one used for generating the occupancy grid. We indicate with Λ the set of all the generated cells and with $\vec{\lambda}_i = (x_i, y_i, \vartheta_i)^T \in \Lambda$ the position of the i^{th} cell. Each cell represents a pose from which the robot can read at least one RFID.

We introduce an n –dimensional binary vector $\vec{\rho}$ for indicating which tags are read and which are not. If the j^{th} tag is detected, then $\rho(j) = 1$ and otherwise $\rho(j) = 0$, with $j \in [0, n]$. For each $\vec{\lambda}_i \in \Lambda$, we build ρ_i , namely the binary vector representing the RFID tags that are detected by the robot when it stands at the pose $\vec{\lambda}_i$. The poses $\vec{\lambda}_i$ and the vectors $\vec{\rho}_i$ are stored in a hash-table structure. Figure 4.4 summarizes schematically the data structure Λ ; for simplicity rotations are not displayed.

In order to reduce its computational complexity, the localization algorithm is composed of two Kalman filters: the first one is a linear Kalman Filter which provides the

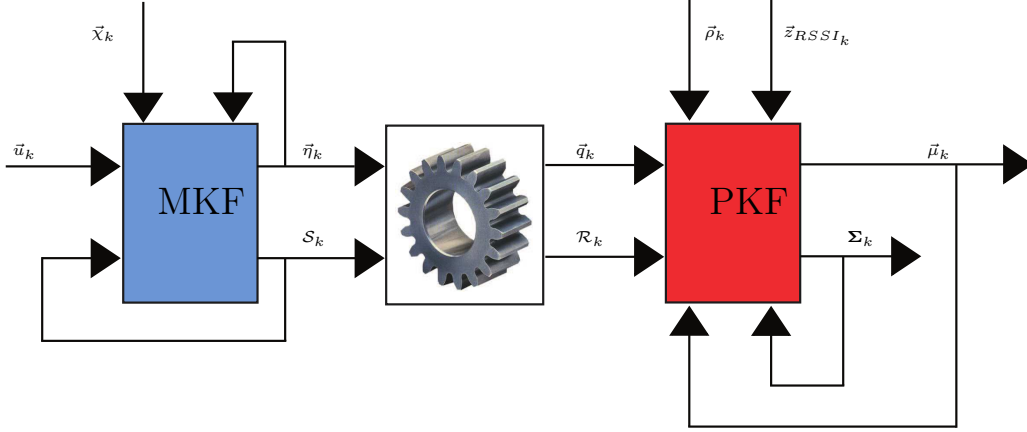


Figure 4.5: Scheme for the proposed algorithm. The gear box represents the data manipulation.

instantaneous angular velocity and the instantaneous linear velocity of the robot by merging the odometric information with the angular rate returned by the gyros. This first filter is also used to predict the angular velocity drift bias noise the gyro is characterized by, in order to prevent the drawbacks of both odometry and the gyroscope. The second one uses the estimated movements of the first filter to predict the position of the robot and merging it with the RFID readings and with the range information extracted from the RSSI measurement of the RF signal. We will refer to the first Kalman Filter as the Movement Kalman Filter (MKF) and to the second Kalman Filter as the Position Kalman Filter (PKF). The overall structure of the algorithm is represented in Fig. 4.5.

The measurement model considered for the MKF is given by:

$$\begin{cases} \begin{cases} v_k = \frac{\delta_{R,k} + \delta_{L,k}}{2} + \nu_{k,1} \\ \omega_k = \frac{\delta_{R,k} - \delta_{L,k}}{d} + \nu_{k,2} \\ \zeta_k = \zeta_{k-1} + \nu_{k,2} \end{cases} \\ \omega_{gyro,k} = \omega_k + \zeta_k + \xi_k \end{cases} \quad (4.3)$$

Here v and ω are respectively the linear speed and the angular velocity of the robot. $\delta_{R,k}$ and $\delta_{L,k}$ are the linear instantaneous velocity of the right and left wheel, while d is the length of the wheelbase. The variable ζ_k is the angular velocity drift variation which the gyros are characterized by and that is used to correct the gyroscopes readings ($\omega_{gyro,k}$) in the measurement function.

Let $\vec{\eta}_k = [v_k, \omega_k, \zeta_k]^T$ be the state vector and $\vec{u}_k = [\delta_{R,k}, \delta_{L,k}]^T$ be the input vector. This system can be represented with matrices as indicated in Eq: 4.4

$$\begin{cases} \vec{\eta}_k = \mathbf{A} \cdot \vec{\eta}_{k-1} + \mathbf{B} \cdot \vec{u}_k + \vec{\nu}_k \\ \vec{\chi}_k = \mathbf{C} \cdot \vec{\eta}_{k-1} + \vec{\xi}_k \end{cases} \quad (4.4)$$

where:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 1/2 & 1/2 \\ 1/d & -1/d \\ 0 & 0 \end{bmatrix}; \mathbf{C} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \quad (4.5)$$

where $\vec{\nu}_k$ and $\vec{\xi}_k$ are white Gaussian noises, with zero mean and covariance respectively $\mathcal{P}_k$ and σ_{gyro}^2 . The matrix $\mathcal{P}_k$ is defined as follows:

$$\mathcal{P}_k = \begin{bmatrix} K_\omega \cdot |\delta_R| & 0 & 0 \\ 0 & K_\omega \cdot |\delta_L| & 0 \\ 0 & 0 & \sigma_{DRIFT}^2 \end{bmatrix} \quad (4.6)$$

Here K_ω is an experimentally obtained constant which describes the wheel-slippage of the robot according to the movement of the wheel itself [35] and σ_{DRIFT}^2 is the variance describing the evolution of the drift bias of the measurement of the angular speed $\omega_{gyro,k}$ read by the gyro.

Let $\vec{\chi}_k = [\omega_{gyro,k}]$ be the instantaneous angular velocity read by the gyro and σ_{gyro}^2 be its variance.

Even if often the gyroscope measurements are more reliable than the odometry information, in this case the data fusion between gyros and odometry is done to prevent the effect of the drift bias noise the gyroscope itself is characterized by. In fact the used sensor (see Sec. 4.6 for more detailed information) is a low cost gyroscope, and the noise generated by the drift bias is very impacting on the angular speed readings. By using only the gyro measurements to compute the movements of the robot, could easily lead to incorrect estimation for the angular velocity of the robot.

The Kalman Filter algorithm [4, 6, 7, 36] has been defined in its classical way as described in Algorithm 3.

Algorithm 3

Movement Kalman Filter ($\vec{\eta}_{k-1}, \mathcal{S}_{k-1}, \vec{u}_k, \vec{\chi}_k$)

- 1: $\hat{\eta}_k = \mathbf{A} \cdot \vec{\eta}_{k-1} + \mathbf{B} \cdot \vec{u}_k$
 - 2: $\hat{\mathcal{S}}_k = \mathbf{A} \cdot \mathcal{S}_{k-1} \cdot \mathbf{A}^T + \mathbf{B} \cdot \mathcal{P}_k \cdot \mathbf{B}^T$
 - 3: $\mathbf{K}_k = \hat{\mathcal{S}}_k \cdot \mathbf{C}^T \cdot \left(\mathbf{C} \cdot \hat{\mathcal{S}}_k \cdot \mathbf{C}^T + \sigma_{gyro}^2 \right)^{-1}$
 - 4: $\vec{\eta}_k = \hat{\eta}_k + \mathbf{K}_k \cdot (\vec{\chi}_k - \mathbf{C} \cdot \hat{\eta}_k)$
 - 5: $\mathcal{S}_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C}) \cdot \hat{\mathcal{S}}_k$
-

The PKF estimates the pose of the robot exploiting the output of the MKF. The linear displacement ΔS_k and the angular rotation $\Delta \psi_k$ during a sampling period ΔT are obtained by:

$$\begin{aligned}\Delta S_k &= \int_t^{t+\Delta T} v_k \, d\tau = v_k \cdot \Delta T \\ \Delta \psi_k &= \int_t^{t+\Delta T} \omega_k \, d\tau = \omega_k \cdot \Delta T\end{aligned}\tag{4.7}$$

where v_k and ω_k come from the MKF. The model of the lawnmower is then given by:

$$\begin{cases} \vec{\mu}_k = f(\vec{\mu}_{k-1}, \vec{q}_k) + \vec{\kappa} \\ \vec{z}_k = g(\vec{\mu}_k) + \vec{l}\end{cases}\tag{4.8}$$

where

$$f(\vec{\mu}_{k-1}, \vec{q}_k) = \begin{cases} x_k = x_{k-1} + \Delta S_k \cdot \cos(\psi_{k-1} + \frac{\Delta \psi_k}{2}) + \kappa_1 \\ y_k = y_{k-1} + \Delta S_k \cdot \sin(\psi_{k-1} + \frac{\Delta \psi_k}{2}) + \kappa_2 \\ \psi_k = \psi_{k-1} + \Delta \psi_k + \kappa_3 \end{cases}\tag{4.9}$$

Here $\vec{\mu}_k = [x, y, \psi]_k^T$ is the state vector which represents the position of the robot and its orientation at step k ; $\vec{q}_k = [\Delta S, \Delta \psi]_k^T$ is the input vector.

The vector $\vec{\kappa}$ represents the noise associated to the model's time propagation function due to the noisy input vector $\vec{q}_k$; this noise can be approximated with a Gaussian white noise with covariance $\mathcal{R}_k$ extracted from the covariance matrix $\mathcal{S}_k$ estimated by the MKF.

The output function $g(\vec{\mu}_k)$ provides two measurements: the first is the binary vector $\vec{\rho}$ associated to the RFID detection and the second is the RSSI measurement of the RF beacon. The noise vector $\vec{l}_j$ represents the Gaussian white noise associated to the RFID and the RSSI readings. The first one is characterized by a covariance computed accordingly the estimated pose and the positions from where the RFID tag can be read, while the second is modeled with the variance σ_{RSSI}^2 obtained experimentally in Sec. 4.3.2.

The PKF algorithm is recursive and the k^{th} step is reported in Algorithm 4. In Lines 1 and 2 the algorithm predicts the pose of the robot and its covariance using the model reported in Eq. 4.9. Φ and Ψ are the Jacobians of $f(\vec{\mu}_{k-1}, \vec{q}_k)$ with respect to $\vec{\mu}_{k-1}$ and $\vec{q}_k$.

At each step the antenna searches for neighboring RFID tags and builds the N_{TAG} -dimensional binary vector $\vec{\rho}_k$ whose elements are set to 1 if the corresponding tag has been detected or to 0 otherwise and where N_{TAG} is the total number of tags (Line 3). In order to avoid false positive readings, an exponential smoothing is performed, as proposed in [23] and in [28] (Line 4) to obtain:

$$\bar{\rho}_k = \left[\sum_{l=0}^{n_w-1} \alpha^l \right]^{-1} \cdot \sum_{l=0}^{n_w-1} \alpha^l \cdot \vec{\rho}_{k-l}\tag{4.10}$$

where $\alpha \in (0, 1)$.

Algorithm 4

Position Kalman Filter $(\vec{\mu}_{k-1}, \Sigma_{k-1}, \vec{q}_k, \vec{z}_k)$

```

1:  $\tilde{\mu}_k = f(\vec{\mu}_{k-1}, \vec{q}_k)$ 
2:  $\tilde{\Sigma}_k = \Phi \cdot \Sigma_{k-1} \cdot \Phi^T + \Psi \cdot \mathcal{R}_k \cdot \Psi^T$ 
3:  $\vec{\rho}_k = setRho()$ 
4:  $\bar{\rho}_k = ExpSmooth(\vec{\rho}_k)$ 
5:  $\tilde{\rho}_k = getRho(\tilde{\mu}_k, \Lambda)$ 
6: if  $(\bar{\rho}_k \neq \vec{0}) \ \& \ (\bar{\rho}_k \neq \tilde{\rho}_k)$  then
7:   for all  $\vec{\lambda}_i \in \Lambda$  s.t.  $\vec{\rho}_i = \bar{\rho}_k$  do
8:      $Prob(\vec{\lambda}_i) = p(\vec{\lambda}_i | \tilde{\mu}_k, \tilde{\Sigma}_k)$ 
9:      $p_i = \varepsilon \cdot Prob(\vec{\lambda}_i)$ 
10:   end for
11:    $\hat{\mu}_k = \sum_{\vec{\lambda}_i \text{ s.t. } \vec{\rho}_i = \bar{\rho}_k} \vec{\lambda}_i \cdot p_i$ 
12:    $\hat{\mathcal{Q}}_k = \sum_{\vec{\lambda}_i \text{ s.t. } \vec{\rho}_i = \bar{\rho}_k} [\vec{\lambda}_i - \hat{\mu}_k] \cdot [\vec{\lambda}_i - \hat{\mu}_k]^T \cdot p_i$ 
13:    $\mathbf{K}_k = \tilde{\Sigma}_k \cdot (\tilde{\Sigma}_k + \hat{\mathcal{Q}}_k)^{-1}$ 
14:    $\bar{\mu}_k = \tilde{\mu}_k + \mathbf{K}_k \cdot [\tilde{\mu}_k - \hat{\mu}_k]$ 
15:    $\bar{\Sigma}_k = (\mathbf{I} - \mathbf{K}_k) \cdot \tilde{\Sigma}_k$ 
16: else
17:    $\bar{\mu}_k = \tilde{\mu}_k$ 
18:    $\bar{\Sigma}_k = \tilde{\Sigma}_k$ 
19: end if
20:  $\vec{z}_k = getRSSI()$ 
21: if  $\vec{z}_k \neq \emptyset$  then
22:   for  $i = 1 \rightarrow size(\vec{z}_k)$  do
23:      $\mathbf{K}_k = \bar{\Sigma}_k \cdot \Gamma^T \cdot [\Gamma \cdot \bar{\Sigma}_k \cdot \Gamma^T + \sigma_{RSSI}^2(\vec{\psi}_k)]^{-1}$ 
24:      $g(\bar{\mu}_k) = Pt[dBm] + K[dB] - 10 \cdot \gamma \cdot \log_{10} \left[ \frac{\sqrt{(\bar{x} - x_{B,i})^2 + (\bar{y} - y_{B,i})^2}}{d_0} \right]$ 
25:      $\hat{\mu}_k = \bar{\mu}_k + \mathbf{K}_k \cdot [\vec{z}_{k,i} - g(\bar{\mu}_k)]$ 
26:      $\hat{\Sigma}_k = (\mathbf{I} - \mathbf{K}_k \cdot \Gamma) \cdot \bar{\Sigma}_k$ 
27:   end for
28: else
29:    $\hat{\mu}_k = \bar{\mu}_k$ 
30:    $\hat{\Sigma}_k = \bar{\Sigma}_k$ 
31: end if
32:  $\vec{\mu}_k = \hat{\mu}_k$ 
33:  $\Sigma_k = \hat{\Sigma}_k$ 

```

The algorithm then computes the binary vector $\tilde{\rho}_k$ the robot is supposed to detect if it were in the predicted pose $\tilde{\mu}_k$ (Line 5). If $\tilde{\rho}_k \neq \bar{\rho}_k$ it means that the prediction $\tilde{\mu}_k$ is inconsistent with the RFID reading and, consequently, with the real pose of the robot. The algorithm then updates the prediction using the measurement $\bar{\rho}_k$. The probability of being in a pose corresponding to the reading $\bar{\rho}_k$ given the predicted pose $\tilde{\mu}_k$ is computed (Line 8) and normalized (Line 9). The mean and the covariance matrix of the predicted position updated using the RFID measurements is computed (Line 11 and Line 12). Finally the prediction is updated with the pose obtained using the RFID tags (Lines 13-15).

If the conditions at line 6 are not satisfied it means that either no RFID tags are detected or that the predicted position is consistent with the detected tags. Thus, the prediction obtained using odometry is used for the following part of the algorithm (Lines 17-18).

The algorithm then reads the RSSI measurement (Line 20) from the on-board antenna. If at least one beacon is “visible” by the sensor, then it is possible to further refine the available estimation by incorporating the new information. For each RSSI measurement a new Kalman gain is computed (Line 23) where $\sigma_{RSSI}^2(\bar{\psi}_k)$ is the variance of the RSSI measurement, which is chosen according to the estimated orientation of the robot through the hash table discussed in Sec. 4.3.2.

Then the Kalman gain is used to perform the update of the position estimation (Line 25) and of the covariance (Line 26). The function $g(\bar{\mu}_k)$ corresponds to Eq. 4.2 applied for an RF source located in (x_B, y_B) and the matrix $\mathbf{\Gamma}$ is its Jacobian computed with respect to $\bar{\mu}_k$.

Using the estimated pose of the robot, a path planning strategy has been designed for obtaining a smart and uniform coverage of the whole lawn. This algorithm generates the path to follow by selecting those cells of the occupancy grid in which the robot has spent less time, which have a lower cost, and then by increasing the cost of those cells in which the robot has already passed through (see Fig. 4.6). With respect to the random navigation case, by using this method the mowing redundancy is greatly decreased, as well as the energy consumption and resulting in a better-looking final shape of the lawn. This planning strategy can also be extended to irregularly shaped fields using vertical cell decomposition or other standard techniques (e.g. [37]).

In particular an AD* (Anytime Dynamic Astar [38]) algorithm for the path-planning is used. The algorithm makes the robot go from a starting point to a goal by selecting cells from the occupancy grid in order to keep low the cost of the overall path. By doing so is also possible to combine an obstacle avoidance algorithm, by changing the cost of those cells occupied by obstacles: the algorithm AD*, which can recompute the path to

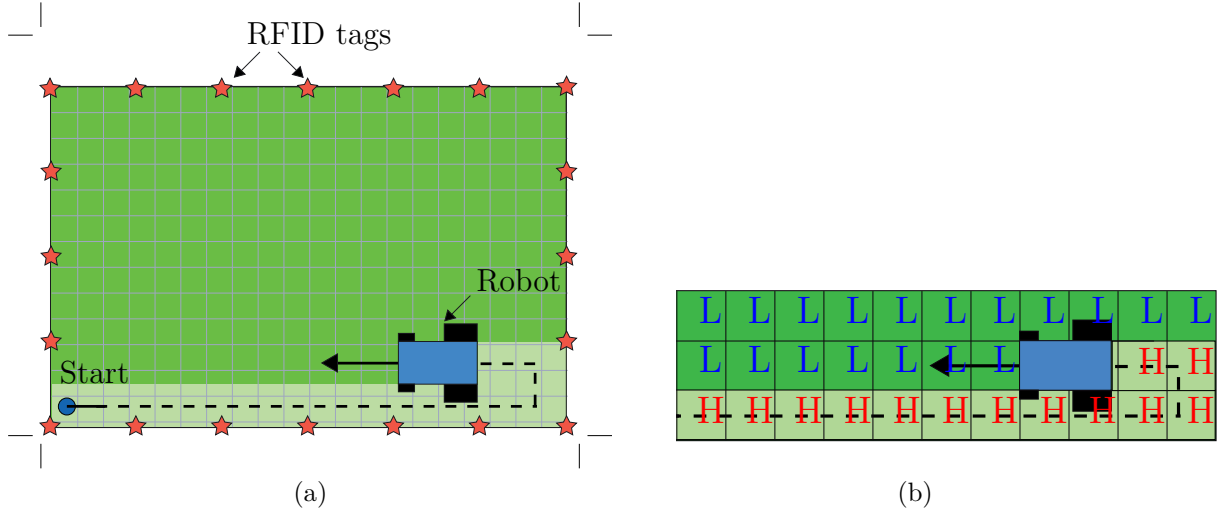


Figure 4.6: Scheme for the proposed path planning strategy

follow dynamically, can change the cells to follow in case of an obstacle which is newly mapped or a moving object is detected. In this case, if a child or a pet are detected by the robot, the robot can implement a safety procedure which can stop, slow down the movements of the wheels and of the blade or simply avoid the obstacle according to the distance of the obstacle.

By doing so the robot follows a boustrophedon path; it has been proven [39, 40] that this is the best way to guarantee a uniform and even coverage of the working area for mobile robots whose task is to reach every single area of the configuration space in the shortest time.

Let's now analyze how the split of the Kalman filter into two smaller cascaded Kalman filters can reduce the overall computational time. Consider the system defined previously in Eq. 4.8. We defined the state vector $\vec{\mu}_k \in \mathbb{R}^n, n > 0$, input vector $\vec{q}_k \in \mathbb{R}^m, m > 0$ and measurement vector $\vec{z}_k \in \mathbb{R}^l, l > 0$. As reported in [4], each update of the Extended Kalman Filter requires time $\mathcal{O}(l^{2.4} + n^2)$. Let's now consider two systems with state vectors $\vec{\mu}_{1,k} \in \mathbb{R}^{n_1}, n_1 > 0$ and $\vec{\mu}_{2,k} \in \mathbb{R}^{n_2}, n_2 > 0$, such that $n = n_1 + n_2$ and with measurement vectors $\vec{z}_{1,k} \in \mathbb{R}^{l_1}, l_1 > 0$ and $\vec{z}_{2,k} \in \mathbb{R}^{l_2}, l_2 > 0$, such that $l = l_1 + l_2$. The Kalman Filter algorithms for these two systems will have computational burden $\mathcal{O}_1(l_1 + n_1^2) + \mathcal{O}_2(l_2 + n_2^2)$. It can be easily proven that the second system has always a computational burden lower than the first by solving the following inequality:

$$(l^{2.4} + n^2) > (l_1^{2.4} + n_1^2) + (l_2^{2.4} + n_2^2) \quad (4.11)$$

4.5 Simulations

In this section we illustrate the effectiveness of the proposed localization algorithm through computer simulations. The used computer is a laptop with an Intel Core i5 processor with a clock speed of 2.70 GHz and 8 GB of RAM. We used The Mathworks MATLAB R2012a(64 bit) for the simulation.

The simulated robot moves in a rectangular shaped ($10\text{ m} \times 6\text{ m}$) working area with 32 RFID tags distributed on its border in known positions and at a distance on 1 m from each other. We used a discretization of the RFID tags reading area with cells of dimension ($50\text{ cm} \times 50\text{ cm}$) and $\pi/18\text{ rad} \approx 10^\circ$. Noises associated to the inputs and to the measurements were simulated by adding random variables generated with MATLAB.

In order to verify if the proposed method can really improve the localization and at the same time decrease the computational burden, we simulated both a monolithic version and the cascaded version of the proposed method.

We simulated the robot with seven different sensor equipment:

- Only odometry/only predictive estimation: Simulation A, fig. 4.7a
- Odometry and gyroscope measurements merged through a monolithic Kalman Filter: Simulation B, fig. 4.7b
- Odometry, gyroscope and RFID measurements merged through a monolithic Kalman Filter: Simulation C, fig. 4.7c
- Odometry, gyroscope, RFID and RSSI measurements merged through a monolithic Kalman Filter: Simulation D, fig. 4.7d
- Odometry, gyroscope and RFID measurements merged through two cascaded Kalman Filters: Simulation E, fig. 4.7e
- Odometry, gyroscope and RSSI measurements merged through two cascaded Kalman Filters: Simulation F, fig. 4.7f
- Odometry, gyroscope, RFID and RSSI measurements merged through two cascaded Kalman Filters: Simulation G, fig. 4.7g

Ten simulations per each sensor combination were performed. In Figure 4.7 a representative of the path covered by the robot per each sensor combination is shown and in Table 4.1 the results of a quantitative analysis is reported. We evaluated the following quantities:

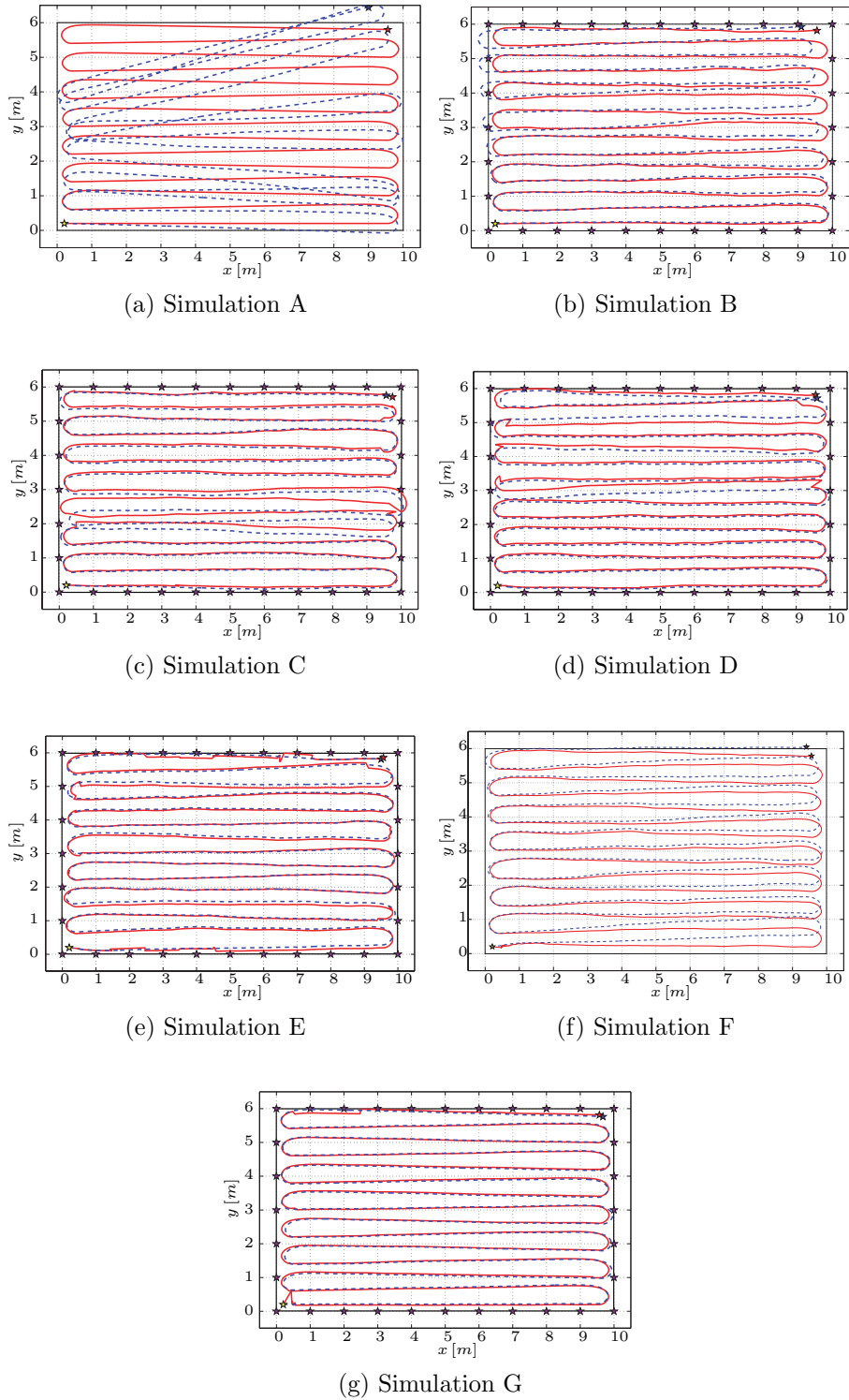


Figure 4.7: Simulation results. The black thick line, the blue dashed line and red solid line are, respectively, the border of the area, the path estimated by the robot and the ground truth. The yellow, blue, red and purple stars are, respectively, the starting position, the estimated final position, the real final position of the robot and the position of the RFID tags.

Simulation name	Average time duration [s]	Average time step [s]	Average final error [mm]	Average Root Mean Square error [mm]	Average maximum error [mm]	Average coverage [%]
A	120	\\	1776	512	4152	71.73
B	1377	0.0105	437	219	500	90.67
C	1472	0.0112	124	116	493	94.40
D	1473	0.0143	108	99	363	97.60
E	1423	0.0038	73.84	55	158	94.67
F	1424	0.0039	301	132	362	88.27
G	1464	0.0040	99	53	396	99.72

Table 4.1: Table summarizing the simulation results.

- Average time duration: the mean time the simulation lasted (in seconds, [s]) computed through all the simulations.
- Average time step: the mean time step duration of each estimation (in seconds, [s]) computed through all the simulations.
- Average final error: the mean final localization error, computed as the difference between the final estimated position and the final ground truth position through all the simulations.
- Average Root Mean Square (RMS) error: computed by using

$$RMS = \sqrt{\sum_k (\hat{x}_k - x_k)^2 + (\hat{y}_k - y_k)^2}$$

where $\hat{x}_k$ and $\hat{y}_k$ are the estimated coordinates of the robot and x_k and y_k are the coordinates of the ground truth at the k^{th} time step, and then by computing the average value through all the simulations.

- Average maximum error: mean computed through all the maximum error committed by all the simulated robots, computed as the difference between the estimated position and the real position.
- Average Coverage: the mean coverage accomplished by all the simulated robots.

The worst case occurs when the simulated robot exploits only odometric data to estimate its position (Simulation A). The covered area by the simulated robot is very

small compared to the overall assigned area. Furthermore, the robot leaves the assigned area for several centimeters.

In the simulation in which the robot exploits the three axis gyroscope (Simulation B) performance is much improved. The coverage of the assigned area is much greater than the previous case, but the robot keeps on accumulating error: this increases the RMS localization error. An approach to reset or further decrease the accumulated localization error would be considered.

Simulation C, in which we add the RFID readings, provides a significant improvement but the best case is provided by the “*full sensor fusion*”, where the simulated robot exploits all the measurements: odometry, angular speeds, RSSI measurements and RFID readings. The only drawback is that by adding more and more measurements each step of the sensor fusion algorithm takes more and more time to be computed in.

This issue is resolved by splitting the Kalman filter into two smaller filters (Simulation E, Simulation F and Simulation G). The time took by every algorithm iteration is decreased by almost an order of magnitude and the coverage and the mean localization errors are comparable to those of the monolithic version of the filter. Furthermore, in some cases, the robot performs a better job than the monolithic case (i.e. Simulation E and Simulation G compared to Simulation C and Simulation D, respectively).

4.6 Experiments

4.6.1 Experiments: coverage in controlled environment

In order to validate the proposed navigation strategy, a prototype lawnmower robot has been built (Fig. 4.8). In particular, in this section, we aim at testing the coverage the proposed algorithm can guarantee without the help of the RF antenna. The robot has three motors: two brushless DC motors for the wheels and a direct drive brushless DC motor for the blade. Each motor has its own driver.

The used gyroscope (a three degrees of freedom Silicon Sensing Pin-Point Gyro Evaluation Board) provides three angular speeds, one for each axis, and is fixed on the top of the axle of the robot. For this experiment the robot is equipped with the CAEN RFID A528 reader, the WANTENNAX004 linearly polarized antenna. In this experiment RSSI measurement was not used.

The high-level control of the robot is performed by the Real-Time controller NI CompactRIO, programmed with LabVIEW 2010 (SP1). This controller is provided with an analog input module (NI9201), an analog output module (NI9263) and a RS232 interface

module (NI9870).

A primitive obstacle detection task is performed thanks to five infrared optical sensors OJ5148|OJH-FPKG/SO/AS: three sensors are mounted on the front of the robot (one in middle, one on the left and one on the right), while the other two sensors are mounted on the right and left side of the robot, respectively.

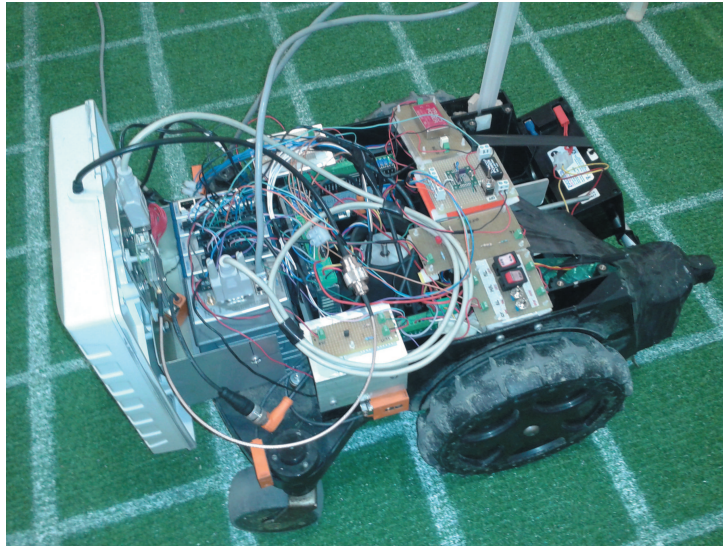


Figure 4.8: The built prototype.

Experiments took place in an indoor square arena with 2 m side and with eight RFID tags (UPM RAFLATAC Wet Inlay RAFSEC short dipole) put slightly outside (40 cm) of the border of the field. The experiments took place on a synthetic grass carpet. The ground truth was obtained through a fixed overhead camera.

The map was discretized in an occupancy grid of 10×10 cells (squares of 200 mm side). The dimension of the cell was decided taking into account the dimension of the blade of the lawnmower. Thus the robot, during its boustrophedon movement, partially covers cells in which it has already been, reducing the probability of leaving uncovered areas.

The robot starts its task from the middle of the first cell and moves along the first boustrophedon path. Once it finishes the first path, it gets back to the starting point and then starts moving following the second boustrophedon path, orthogonal to the first. Once the robot finishes the second path it gets back to the starting point.

In Fig. 4.9a the behavior of the odometry-only localization can be seen. The robot moves accordingly to the path-planner through the two boustrophedon paths, but it

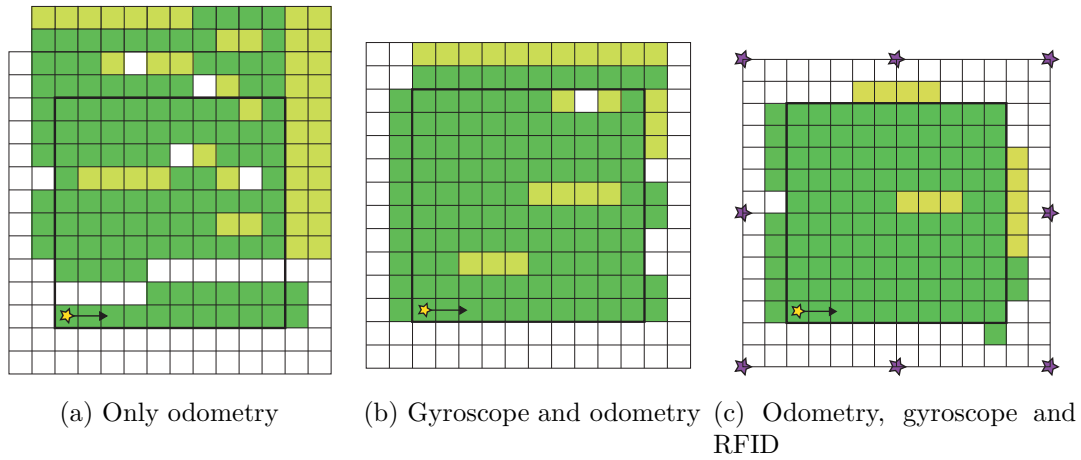


Figure 4.9: The area covered by the robot during experiments. The thick dark line marks the perimeter of the lawn. White cells represent ignored areas, dark green cells represent areas where the robot has passed through while doing the first spiral and light green cells represent those cells where the robot has passed through during the second spiral. Purple stars represents the position of the RFID labels. The yellow star and the black arrow represents respectively the initial position and orientation of the robot.

goes almost one meter away from the border of the working area. This is unacceptable, because [41] regulation requires that the robot is allowed to leave the assigned area of a maximum distance equal to its length. Furthermore the coverage of the area isn't satisfactory at all.

We also did an experiment using as localization algorithm an Extended Kalman Filter which merges only the odometric information with the gyroscope data: results are shown in Fig. 4.9b. In this case, the efficiency of the robot is greatly increased with respect to the one obtained in the dead-reckoning case. The working area is almost covered, but the robot keeps on going outside of the working area (about 40 *cm*, almost the length of the robot).

Finally we repeated the experiment with the robot localizing itself with the complete sensor fusion algorithm (figure 4.9c). The robot covers the 100% of the working area, and it leaves the assigned map of just few centimeters only when it's turning. Furthermore only the 3% of the cells were not covered during the first boustrophedon path.

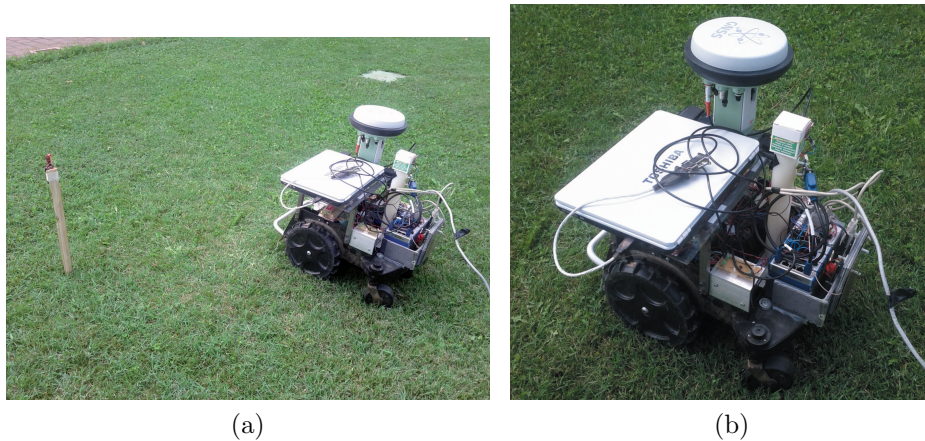


Figure 4.10: The upgraded robot with the eZ430-RF2500 RF antenna, the WANTEN-NAX009 RFID antenna and the RTK antenna

4.6.2 Experiments: outdoor environment

The algorithm was then verified experimentally in an outdoor environment. With respect to the previous experiments session, on the robot is also mounted an eZ430-RF2500 RF antenna, which provides the RSSI measurement of the signals sent by the RF beacons (also consist of eZ430-RF2500) put in known positions in the working area. The ground truth of the robot is obtained through a Leica Viva RTK (Real Time Kinematics) antenna mounted on the robot able to provide the position of the robot with 10 *mm* precision on the horizontal plane (x, y) and 20 *mm* precision for the altitude (which is not considered in the presented work). Of course the precision of the RTK antenna get worse with bad weather conditions.

Under the robot, a three edged blade is used to accomplish the mowing task. The blade diameter is 25 *cm*. The RFID antenna and the CompactRIO controller are located on the front side of the robot. The 25 *V* battery is located on the back of the robot.

The rotation speed of the mowing blade is controlled by an algorithm which modulates the speed according to the current absorption of the motor of the blade. In this way the blade will rotate faster if the grass is thicker or slower if the grass is thinner or absent (i.e. in non-lawn patches). This strategy can further reduce the energy consumption of the robot and increase the life of the battery and durability of the overall system.

This time experiments took place outdoor in a rectangle shaped arena with dimensions 10 *m* $\times$ 6 *m* side. The map was discretized in an occupancy grid of 25 $\times$ 15 cells (squares of 400 *mm* side). On the border of the working area we put omni-directional UH3D40 LabID RFID tags, each one spaced 1 *m* to the other. The RFID tags were left in the



Figure 4.11: The pegs used in the experiments

garden for a week, in order for the environment to act on them covering them with dust, mud, grass or leaves. This was done with the intention of testing the proposed method with the most realistic conditions. We placed the RF beacon with a height of 45 cm (the same height of the receiver antenna on-board the robot) in position $(-50, -50)\text{ cm}$ with the normal to the antenna oriented in the same direction of the x axis of the map. The robot performs a boustrophedon path, in order make a uniform coverage of the assigned area (the lawn to be mowed).

The RFID tags are placed in ad-hoc built pegs, Fig. 4.11. The top of the peg, where the RFID tag is encapsulated and sealed, is separated by the body (see Fig. 4.11a) of about 5 mm : by doing so we prevent the soil to act as a grounding shield for the signal emitted by the RFID.

Three experiments took place. In Experiment A we tested the robot moving and localizing itself with the gyroscope and the RFID measurements. The RFID tags help the robot to recover its true position, but in the middle of the lawn, where the RFID tags are negligible, the robot drifts because of the very slippery ground and accumulates error (see Fig. 4.12a). The robot accomplishes its mowing task without leaving relevant portion of the lawn careless.

In order to test the quality and the robustness of the RF localization, we performed an experiment (Experiment B) with the robot localizing itself exploiting only the RSSI measurement and the gyro correction as well. Even if the trajectory of the robot is corrected by the gyroscope, the robot accumulates error and leaves the assigned arena of several centimeters (see Fig. 4.12b). As said above, this is unacceptable because of the European regulation [41].

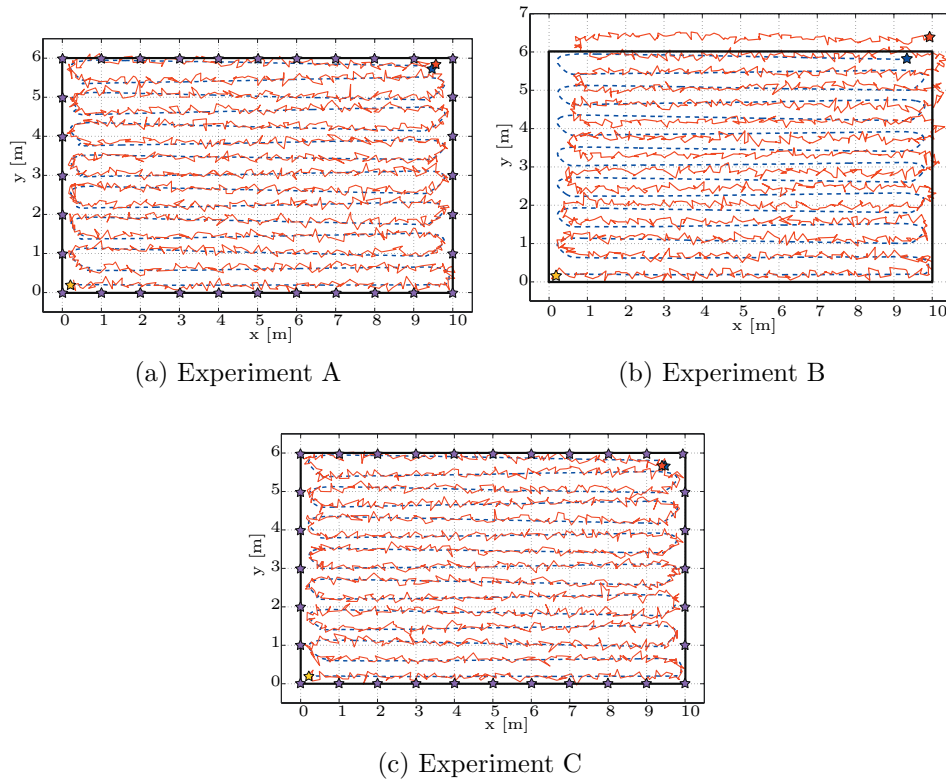


Figure 4.12: Experimental results. The black thick line, the blue dashed line and red solid line are, respectively, the border of the area, the path estimated by the robot and the ground truth obtained by the RTK. The yellow, blue, red and purple stars are, respectively, the starting position, the estimated final position, the real final position of the robot and the position of the RFID tags.

Finally the proposed method has been tested (Experiment C). Here the robot localizes itself with all the equipped sensors: the odometric data, the angular rates, the RFID readings and the RSSI measurements. Results are shown in Fig. 4.12c. The performance of the robot is improved. The accumulated localization error during the path in the middle of the garden is reduced of 28% with respect to experiment A, due to the RSSI readings, while with the RFID tags that the robot reads when it reaches the border of the lawn, the localization error is further reduced and the robot does not leave the assigned area. Like in the first experiment, even this case the robots does its mowing task accurately without ignoring any area of the lawn. Results are summarized in Table 4.2.

Experiment name	RMS error [mm]	Final error [mm]	Coverage [%]
A	126	113	99.47
B	455	678	96.80
C	109	81	99.73

Table 4.2: Table summarizing the experimental results.

4.7 Discussion

In this chapter we have shown that it is possible to use cheap and alternative sensors as RFID tags and RF beacons for improving the localization capabilities of an autonomous lawnmower. We have designed a novel localization technique that exploits RFID tags for resetting the estimate of the pose during mowing and RF beacons for improving the estimate when the robot is mowing wide areas. Both of these sensors have been intensively tested and characterized in order to have a reliable sensor model in non-nominal working condition (e.g. outdoor conditions). The estimated pose is then used for implementing a simple but efficient coverage strategy that allows to completely cover a garden in less time than commercially available robotic lawnmowers.

The mowing strategy proposed in this chapter decreases the amount of wear mechanical components are subject to and the energy required by batteries for a single cut of the garden. This can increase the life of the robotic lawnmower and decrease the necessary maintenance. Furthermore, the cost of the adopted sensor equipment is low and the solution proposed in this chapter can be put in the market at a competitive price. In particular the low cost of the RSSI antenna (the cost of the two antenna is approximately

€43) and the very low cost of the RFID tags (each RFID tag costs only €0.03) make the proposed solution viable also for very large gardens.

The effectiveness of the proposed navigation strategy is validated both in simulations and experimentally. Experiments proved that the proposed localization algorithm, based on two cascaded Kalman filters, and path planning algorithm can greatly improve the coverage of the working area made by the robotic lawnmower. The robot can accomplish its task providing an even mowing in all the garden areas avoiding unnecessary redundancy in passing in areas already visited by the robot. This can also improve the robot's durability: this strategy allows the robot to accomplish its duty in shorter time, increasing the durability of the mechanical components of the lawnmower, which are less subject to wear, and of its batteries, decreasing also the energy required in order to have a lawn completely mowed. Furthermore the proposed method can be implemented by using only low cost sensors, making the final product competitive on the market.

Furthermore experiments showed that thanks to the RFID tags put on the border of the lawn, that can confine the robot inside its working area, it is possible to avoid the current-carrying boundary wire, which requires to be dug in the lawn and also requires to be substituted if it gets accidentally cut.

Chapter 5

3D sensors characterization

While RGB-D sensors are becoming more and more popular in mobile robotics laboratories, they are usually not yet adopted for industrial applications. In fact, in this field, depth measurements are generally acquired by means of laser scanners and, when visual information is needed, by means of stereo-cameras. The aim of the work presented in this chapter is to perform an experimental validation, to compare the performance of a stereo-camera and an RGB-D sensor, in a specific application: mobile robot localization for industrial applications. Experiments are performed exploiting artificial landmarks (defined by a self-similar pattern), placed in known positions in the environment. This work is done in the framework of project DiRò.

5.1 Introduction

This work aims at providing experimental comparison of 3D vision technologies in the field of mobile robot localization in industrial applications.

Mobile robots are nowadays widely adopted in manufacturing plants. A remarkable example is represented by AGVs (Automated Guided Vehicles) used for end-of-line activities [42–44]. As for most of mobile robots applications, localization is a crucial issue: robots need to be able to measure their current position, in order to perform their tasks.

Global localization is typically solved, in industrial environment, by means of laser trilateration: each robot is equipped with a laser scanner, which measures the distance with respect to opportunely placed reflectors, that are used as landmarks. The position of each landmark is stored in a map, so that trilateration can be used by the robots to compute their global position. The main drawback of this technology is that, in order to provide reliable localization in a cluttered environment (which is typically the case in a manufacturing plant) hundreds to thousands of landmarks are required [44]. This fact makes the installation of this kind of systems very expensive and intrusive.

To avoid this issue, natural landmarks could be used, instead of artificial ones. Natural

landmarks [45] are features of the environment that are easily recognizable by means of sensors. If the position of these features is measured with respect to the map of the environment, they can be used for mobile robots localization. In [45] and references therein, laser scanners are used to identify natural landmarks: however, when moving in a manufacturing plant, robots need to cope with highly cluttered, wide environments, with the presence of several symmetries. Hence, vision sensors appear necessary to acquire sufficiently reach data.

More specifically, 3D vision sensors provide both visual and distance information: hence, they may be adopted to identify natural landmarks while measuring their relative position. On these lines, some pioneering works may be found in the literature [46], where 3D vision based localization algorithm are validated in simplified simulated frameworks.

Stereo-cameras has been adopted, since the 1950s, to acquire three-dimensional images. They are cameras with two (or more) lenses, with a separate image sensor for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images. This is obtained comparing the two images: as the distance between the two lenses is known, the so called *disparity image* may be computed, from which depth may be determined [47]. Moreover, depth information can be obtained combining vision with laser, or structured light. On these line, in the last few years, RGB-D sensors are becoming more and more popular: these sensors exploit structured light to acquire both visual (RGB) and deph (D) information about the environment [48]. However, depth measurement performance is considerably lower, if compared to a traditional laser scanner [49].

Visual localization have been widely adopted, in the last few years, for mobile robotics applications, as described, for instance, in the survey paper [50]. In [51] stereo vision is used for localization and SLAM (Simultaneous Localization and Mapping), using a stereo-camera that acquires the position of known landmarks, in indoor environment. Conversely, the work in [52] presents an outdoor visual localization application, that exploits an omni-directional camera to acquire the relative positions of some features of the environment, namely edges. Stereo-cameras are exploited in [53] for rough localization in small agricultural fields.

Due to their relatively low cost, RGB-D sensors have been extensively adopted, in the last few years, in the robotics research community. These sensors are generally used in laboratory scale experiments, for visual odometry [54], detection of humans [55], object recognition [56], obstacle avoidance [57] and localization [58].

To the best of the authors' knowledge, RGB-D sensors are not usually employed in

industrial environment. A first comparison between stereo-cameras and RGB-D sensors may be found in the literature in [59], where the performance of the two sensors is compared in a domestic environment, for elderly people monitoring.

Then, the main objective of this work is to provide an experimental evaluation of the performance of two different classes of 3D vision sensors, regarding the localization application: a stereo-camera, and an RGB-D sensor. Conversely, feature recognition is out of the scope of this work, and will be addressed subsequently. Hence, we introduce visual artificial landmarks that, given a specific pattern, are easily recognizable in the environment. Localization is then obtained with respect to the landmarks themselves, whose positions are previously recorded with respect to the map of the environment [60]. This allows us to focus our attention on sensors' performance, without being influenced by specific issues, related to object recognition algorithm.

The contribution of the work may then be summarized as follows:

1. We implement a localization algorithm that exploits Extended Kalman Filter for data fusion of odometry with 3D visual information. Specifically, 3D vision sensors are used to compute the relative positions of self-similar landmarks, that were previously exploited in the literature for navigation purposes.
2. We experimentally compare the accuracy of the relative position measurements provided by a stereo-camera and by an RGB-D sensor.
3. We experimentally evaluate the performance of the visual localization system, comparing the results obtained with the stereo-camera and with the RGB-D sensor.

The outline of the chapter is as follows. Section 5.2 describes the adopted 3D vision sensors, and provides the main characteristics of the landmarks used for localization. In Section 5.3 we experimentally compute the covariance matrices associated to the measurements provided by the sensors. Section 5.4 describes the the localization algorithm based on Extended Kalman Filter. Experiments are then described in Section 5.5. Finally, Section 5.6 gives some concluding remarks.

5.2 Description of the adopted technology

5.2.1 3D vision sensors

As stated in the Introduction, we are interested in comparing the performances of two different technologies to acquire 3D vision data: stereo-camera and RGB-D sensor. In

this section we briefly describe the sensors employed in our experimental setup.

Microsoft Kinect RGB-D sensor

Microsoft Kinect is a low cost 3D vision sensor developed for video-game purpose. The Kinect sensor [61] is a horizontal bar connected to a small base with a motorized pivot (Fig. 5.1). The device features an RGB camera, coupled with a depth sensor, that is an infrared projector (infrared light emitter) combined with a monochrome CMOS sensor (infrared detector), which captures video data in three dimensions under varying ambient light conditions. The acquired image has a resolution of 640×480 pixel. Visible and depth information can be acquired at a maximum rate of $30Hz$.

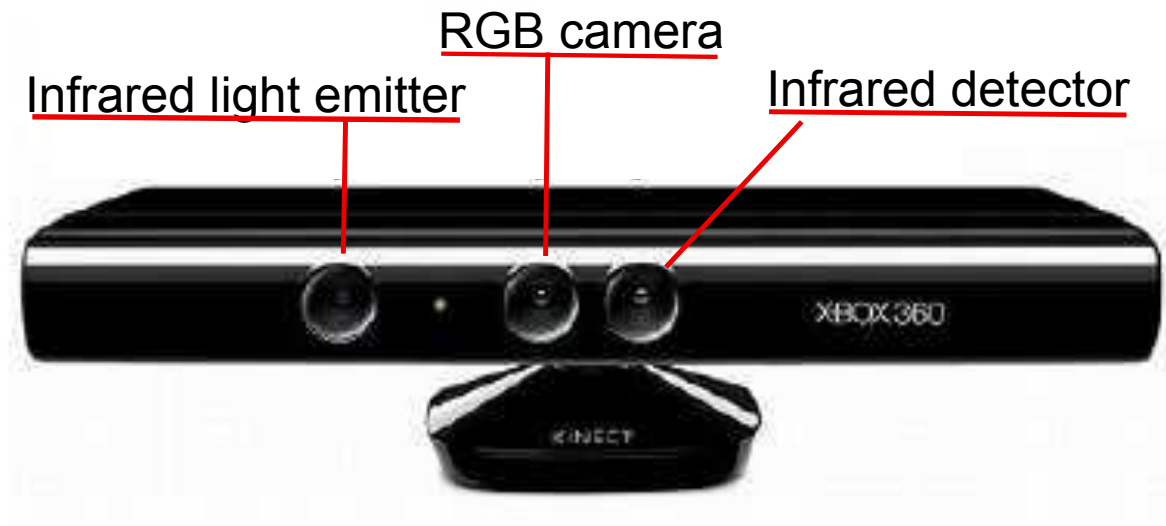


Figure 5.1: Microsoft Kinect RGB-D sensor

As Kinect is intended for video-game purpose, it is a very low cost sensor: its price ranges €120.00 to €160.00.

PointGrey Bumblebee2 Stereo-camera

PointGrey Bumblebee2 [62] is a FireWire (IEEE-1394) stereo-camera, which is equipped with two Sony $1/3''$ progressive scan color CCD sensors. The image may be acquired at a resolution of 1024×768 pixel with a rate of $20Hz$, or with a resolution of 640×480 pixel with a rate of $48Hz$. The camera is pre-calibrated against distortion and misalignment



Figure 5.2: PointGrey Bumblebee2 stereo-camera

(RMS calibration error is guaranteed to be within 0.1 pixel). As Bumblebee2 stereo-camera is intended for professional use, it is considerably more expensive than Kinect: in fact, its price is around €2300.00.

5.2.2 Self-similar landmarks

Exploiting information acquired by means of the 3D vision sensors, the robot is supposed to autonomously localize itself with respect to a map. More specifically, artificial landmarks are placed in the environment, and their positions are stored, with respect to the global map.

Hence, the robot is required to identify the landmarks, to measure their relative position, and subsequently to localize itself.

As stated in the Introduction, the main purpose of replacing laser scanners with vision sensors is the possibility to exploit the so called natural landmark: avoiding the need for artificial landmarks would significantly reduce the installation costs for mobile robotic systems in industrial applications. However, unique identification of natural landmarks in dynamic environments is a hard problem. Conversely, for localization purposes, exact identification and measurement of the relative position of the landmarks is critical. Hence, as the purpose of this work is to compare the performance of two different kind of 3D

vision sensors, we consider a simplified localization framework: specifically, we introduce artificial landmarks that are reliably identifiable.

In particular, we exploit self-similar landmarks, introduced in [63] for navigation purposes. An example of self-similar landmark is represented in Fig. 5.3. More specifi-

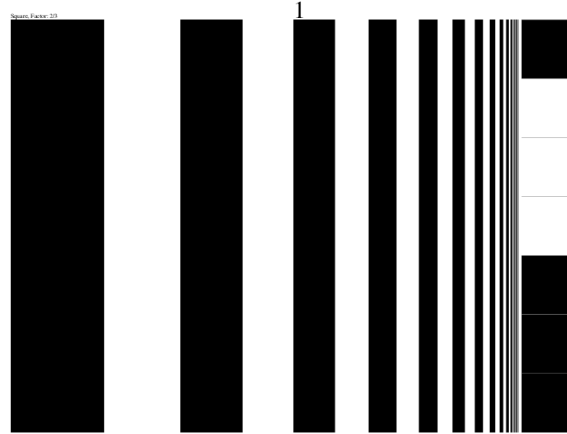


Figure 5.3: Landmarks used for localization: self-similar pattern on the left-hand side, binary unique identifier on the right-hand side

cally, these landmarks are defined by means of self-similar intensity patterns. A function $f : \mathbb{R}^+ \mapsto \mathbb{R}$ is p -similar for the scale factor p , where $0 < p < 1$, if $f(x) = f(px)$, $\forall x > 0$. Hence, a p -similar function is identical to itself scaled by p in the x direction. Consequently, a p -similar pattern can be detected in the environment, by comparing the observed intensity pattern to a version of itself scaled by p . This characteristic makes the identification of this kind of landmarks very reliable [63].

As shown in Fig. 5.3, the left-hand side of the landmark represents a self-similar pattern, while the right-hand side of the landmark encodes the unique identifier of each landmark as a binary code. The position of each landmark (with respect to the map) is then stored, as well as its unique identifier. As will be explained in the next sections, these data will be exploited for localization purpose, acquiring the relative positions of the landmarks with the 3D vision sensors.

5.3 Experimental characterization of the 3D vision sensors

In order to evaluate the performance of a localization algorithm based on 3D vision, it is necessary to analyze the quality of the measurements provided by the sensors.

The quality of a measurement can be quantified by means of the associated covariance matrix [4]. Generally speaking, in fact, a sensor provides inaccurate data: the covariance matrix is a positive semidefinite and symmetric matrix, that measures the squared expected deviation from the mean value of the acquired data.

The covariance matrix may be estimated performing repeated experiments, and analyzing the acquired data set. In this particular application, we are interested in measuring the relative position of an object, with respect to the current position of the sensor. As we are considering wheeled mobile robots moving in indoor environment, we are interested in computing two-dimensional relative positions, on a plane identified by the ground floor.

More specifically, let (x_s, y_s) be the sensor's reference frame, where the y_s axis is parallel to the sensor's optical axis. Then, experiments have been performed to evaluate the accuracy of the sensors in measuring the relative position of an object, along the x_s and y_s axes.

In particular, we measured the relative position of a landmark (described in Section 5.2.2) with respect to both the adopted sensors. We experimentally verified that, with the RGB-D sensor's resolution (640×480 pixel) a landmark is correctly identifiable if its distance is between $1m$ and $3m$. Conversely, exploiting the full resolution of the stereo-camera (1024×768 pixel) landmarks are identifiable up to $4m$.

As the aim of the experiments is to compare the performance of two different technologies, it is necessary to use the same conditions in both cases: specifically, we adopted the same resolution for both sensors, namely 640×480 pixel. Thus, the experiments were performed as follows:

- The measurement errors along the x_s and y_s axes have been measured independently:
 - the y_s coordinate of the landmark has been varied from $1m$ to $3m$, while keeping $x_s = 0m$.
 - the x_s coordinate of the landmark has been varied from $-1.25m$ to $+1.25m$, while keeping $y_s = 2.5m$.
- We acquired 25 images for each axis and for each sensor.

The covariance matrices $\mathcal{Q} \in \mathbb{R}^{2 \times 2}$ have been approximated as diagonal matrices. Namely:

$$\mathcal{Q} = \begin{bmatrix} \sigma_{x_s}^2 & 0 \\ 0 & \sigma_{y_s}^2 \end{bmatrix} \quad (5.1)$$

where $\sigma_{x_s}^2$ and $\sigma_{y_s}^2$ represent the variance of the measurements along the x_s and y_s axis, respectively. In fact, as described in [4], the off-diagonal elements are negligible if compared to the diagonal elements.

We obtained the following experimental results (all results are in m^2):

- The covariance matrix of the measurements acquired with the Bumblebee2 stereo-camera is the following:

$$\mathcal{Q}_b = \begin{bmatrix} 1.8 \times 10^{-3} & 0 \\ 0 & 3.9 \times 10^{-4} \end{bmatrix} \quad (5.2)$$

- The covariance matrix of the measurements acquired with the Kinect RGB-D sensor is the following:

$$\mathcal{Q}_k = \begin{bmatrix} 2.7 \times 10^{-3} & 0 \\ 0 & 6.3 \times 10^{-3} \end{bmatrix} \quad (5.3)$$

It is worth noting that these results are obtained using the same resolution for both sensors, that is 640×480 pixel. Hence, it is possible to conclude that the stereo-camera provides more reliable relative position measurements: in fact, the values of the covariance matrix are significantly smaller, with respect to the ones computed for the RGB-D sensor.

5.4 EKF-based Localization using self-similar landmarks

We consider the problem of localizing a mobile robot operating in an indoor environment of known dimension, with the landmarks described in Section 5.2.2 scattered in it in known positions. The proposed localization method is based on the popular Extended Kalman Filter (EKF) Localization algorithm (see [4, 5]).

The algorithm is recursive and the k -th step is discussed. Here we consider $\vec{\mu}_k \in \mathbb{R}^3$ as the state vector representing the position and orientation of the robot, namely $\vec{\mu}_k = [x, y, \vartheta]^T$.

Moreover, $\vec{u}_k \in \mathbb{R}^2$ represents the input vector, namely $\vec{u}_k = [\delta_R, \delta_L]$, with δ_R and δ_L the instantaneous displacement of the right and left wheels respectively.

The vector $\vec{z}_{k,i} \in \mathbb{R}^2$ represents the measurement vector related to the i -th landmark, which has been detected by the on-board sensor, namely $\vec{z}_{k,i} = [r_i, \alpha_i]^T$, where r_i and α_i are the range and bearing measurements, respectively.

Let $\vec{m}_i = [x_m, y_m]^T$ be the position of the i -th landmark according to the reference frame of the map, we define $\vec{m}$ as the data structure containing the position information

Algorithm 5 Extended Kalman Filter Localization

```

1: Algorithm_EKF_Localization( $\vec{\mu}_{k-1}, \mathcal{S}_{k-1}, \vec{u}_k, \vec{z}_k, \vec{m}$ )
2:  $\hat{\mu}_k = g(\vec{\mu}_{k-1}, \vec{u}_k)$ 
3:  $\mathcal{G}_{k-1} = \left[ \frac{\partial g_i}{\partial \mu_j} \right]; \mathcal{V}_k = \left[ \frac{\partial g_i}{\partial u_j} \right]$ 
4:  $\hat{\mathcal{S}}_k = \mathcal{G}_{k-1} \cdot \mathcal{S}_{k-1} \cdot \mathcal{G}_{k-1}^T + \mathcal{V}_k \cdot \mathcal{R}_k \cdot \mathcal{V}_k^T$ 
5: for all  $\vec{z}_{k,i}$  do
6:    $\mathcal{H}_k = \left[ \frac{\partial h_i}{\partial \mu_j} \right]; \mathcal{P}_k = \left[ \frac{\partial h_i}{\partial p_j} \right]$ 
7:    $\mathcal{K}_k = \hat{\mathcal{S}}_k \cdot \mathcal{H}_k^T \cdot \left( \mathcal{H}_k \cdot \hat{\mathcal{S}}_k \cdot \mathcal{H}_k^T + \mathcal{P}_k \cdot \mathcal{Q}_{x,y} \cdot \mathcal{P}_k^T \right)^{-1}$ 
8:    $\hat{\mu}_k = \hat{\mu}_k + \mathcal{K}_k \cdot [\vec{z}_{k,i} - h(\hat{\mu}_k, \vec{m}_i)]$ 
9:    $\hat{\mathcal{S}}_k = (\mathcal{I} - \mathcal{K}_k \cdot \mathcal{H}_k) \cdot \hat{\mathcal{S}}_k$ 
10: end for
11:  $\mu_k = \hat{\mu}_k; \mathcal{S}_k = \hat{\mathcal{S}}_k$ 
12: return  $\mu_k, \mathcal{S}_k$ 

```

of the whole landmark set. The matrix $\mathcal{S}_k$ is the covariance matrix associated to the estimation uncertainty.

Line 2 of Algorithm 5 represents the computation of the time propagation of the system: we consider the motion model of the differential drive wheeled mobile robot (refer to [64]):

$$g(\mu_{k-1}, u_k) = \begin{cases} x_k = x_{k-1} + \frac{\delta_R + \delta_L}{2} \cos\left(\vartheta_{k-1} + \frac{\delta_R - \delta_L}{2d}\right) \\ y_k = y_{k-1} + \frac{\delta_R + \delta_L}{2} \sin\left(\vartheta_{k-1} + \frac{\delta_R - \delta_L}{2d}\right) \\ \vartheta_k = \vartheta_{k-1} + \frac{\delta_R - \delta_L}{d} \end{cases} \quad (5.4)$$

where d is the distance between the two actuated wheels.

In Line 3 of Algorithm 5, the two jacobians of the time propagation function are computed, with respect to the state vector $\vec{\mu}_k$ and the input vector $\vec{u}_k$. Then, Line 4 computes the prediction of the covariance matrix, and $\mathcal{R}_k \in \mathbb{R}^{2 \times 2}$ is the covariance associated to the input noise.

Lines 5 to 10 compute the update steps of the Extended Kalman Filter for each landmark which has been detected by the sensor. In particular, Line 6 computes the jacobian of the output function $h(\hat{\mu}_k, \vec{m}_i)$, defined as follows:

$$h(\mu_k, m_{k,i}) = \begin{cases} \rho_{k,i} = \sqrt{\Delta x_{k,i}^2 + \Delta y_{k,i}^2} \\ \alpha_{k,i} = \text{atan2}(\Delta y_{k,i}, \Delta x_{k,i}) - \vartheta_k \end{cases} \quad (5.5)$$

$$\begin{aligned} \Delta x_{k,i} &= x_k - x_{m,i} \\ \Delta y_{k,i} &= y_k - y_{m,i} \end{aligned} \quad (5.6)$$

where μ_k represents the state vector, and $\vec{p}_k = [\Delta x, \Delta y]^T$.

Subsequently, Line 7 computes the so-called Kalman gain, $\mathcal{K}_k$, exploiting the covariance matrix associated to the measurement uncertainty, that is $\mathcal{Q}_{x,y} \in \mathbb{R}^{2 \times 2}$. In particular, the term $\mathcal{P}_k \cdot \mathcal{Q}_{x,y} \cdot \mathcal{P}_k^T$ and the computation of the jacobian $\mathcal{P}_k$ are necessary because, as described in Section 5.3, we characterized the sensors with respect to the (x_s, y_s) coordinates, and not according to the range and bearing considered in the output function. Then, in Lines 8 and 9, the estimation and covariance update are computed. Finally, in Line 11 the state and the covariance estimations are returned.

5.5 Experimental results

The localization algorithm described in Section 5.4 has been implemented on an Adept MobileRobots Pioneer 3-DX robot, controlled by means of the software framework ROS (Robot Operating System) [65, 66].

A variable number of landmarks (from 1 to 5) have been placed in the environment, in known positions. The experimental setup is represented in Fig. 5.4.



Figure 5.4: Experimental setup

In order to precisely evaluate the performance of the localization system, experiments have been implemented in static conditions. Namely, the objective of the algorithm is to compute the position of the robot under the following conditions:

- The robot's initial position is
$$\begin{cases} x_0 = 0; \\ y_0 = 0; \\ \theta_0 = \frac{\pi}{2} \end{cases} \quad (5.7)$$

- The robot does not move, but an error is added to the odometry measurements. Specifically:
 - at the first iteration, the odometry measurement provides the following data:
 $\delta_R = \delta_L = 0.50m$.
 - at the subsequent iteration, the odometry measurement provides the following data: $\delta_R = \delta_L = 0.0m$.

Hence, the EKF-based localization algorithm is required to compensate the error provided by the odometry, acquiring the landmarks' relative position with the 3D vision sensors.

This experimental test have been performed with the Kinect RGB-D sensor, and with the Bumblebee2 stereo-camera, in three different experimental setup, characterized by the presence of 1, 3 and 5 landmarks, respectively. Statistical results obtained during the experiments are collected in Figs. 5.5, 5.6 and 5.7. These plots represent the result of the EKF-based localization algorithm (using data acquired with both sensors): blue solid line represent the real value, the red dotted line represents the measurement computed with the Kinect, black dashed line represents the measurement computed with the Bumblebee2. Standard deviation is shown as well. As expected, the uncertainty of the measurements decreases, as the number of iteration grows. Moreover, adding more landmarks, it is possible to obtain a higher accuracy and a faster convergence. The fact that, as shown in Section 5.3, the Bumblebee2 camera provides more accurate measurements than the Kinect generates higher precision in the localization. More specifically, as shown in Figs. 5.5, 5.6 and 5.7, the Bumblebee2 camera leads to a more accurate localization, in term of position (Figs. 5.5 and 5.6) and orientation (Fig. 5.7).

5.6 Analysis of the results and conclusion

In this work we performed several experimental tests to compare the performance of 3D vision sensors for mobile robot localization. In particular, we adopted two different sensors: a PointGrey Bumblebee2 stereo-camera, and a Microsoft Kinect RGB-D sensor.

A localization algorithm based on Extended Kalman Filter has been implemented, to combine odometry with the measurement of the relative positions of artificial landmarks, defined by a self-similar pattern.

As expected, the higher quality of the stereo-camera produces better localization performance, in terms of precision, with respect to the RGB-D sensor. However, it is necessary to perform further experiments, in order to better evaluate the robustness of both

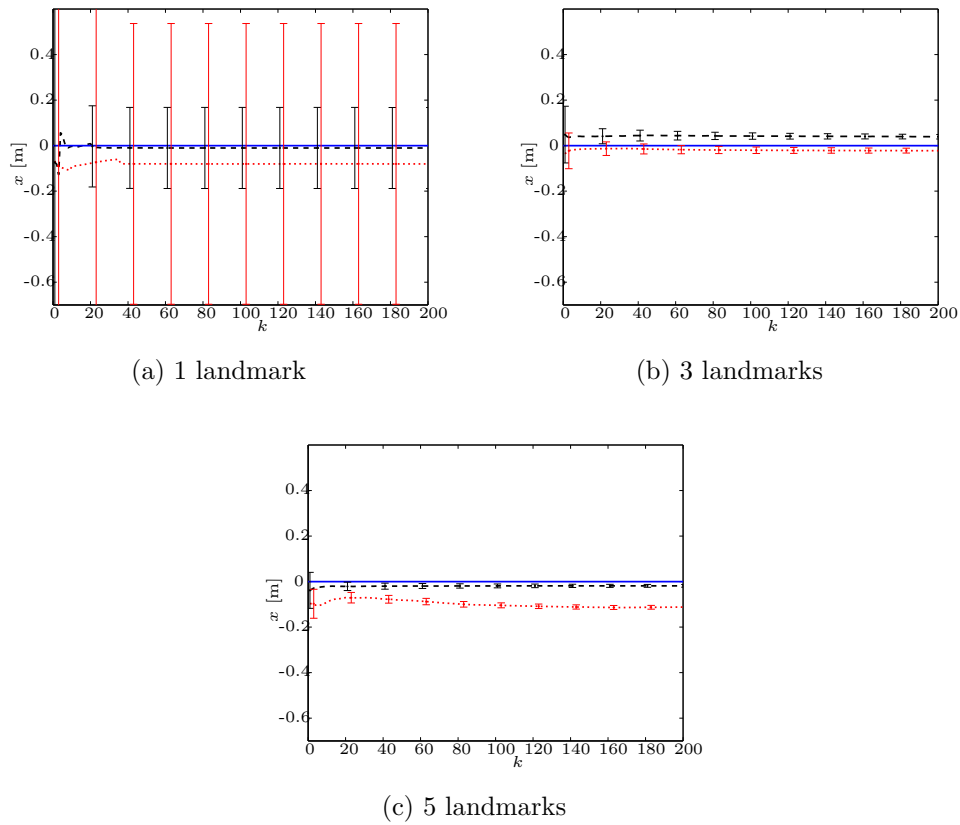


Figure 5.5: Localization experiment in static condition: measurement of the x coordinate. Blue solid line represents the ground truth, red dotted line represents the measurement computed with the Kinect, black dashed line represents the measurement computed with the Bumblebee2

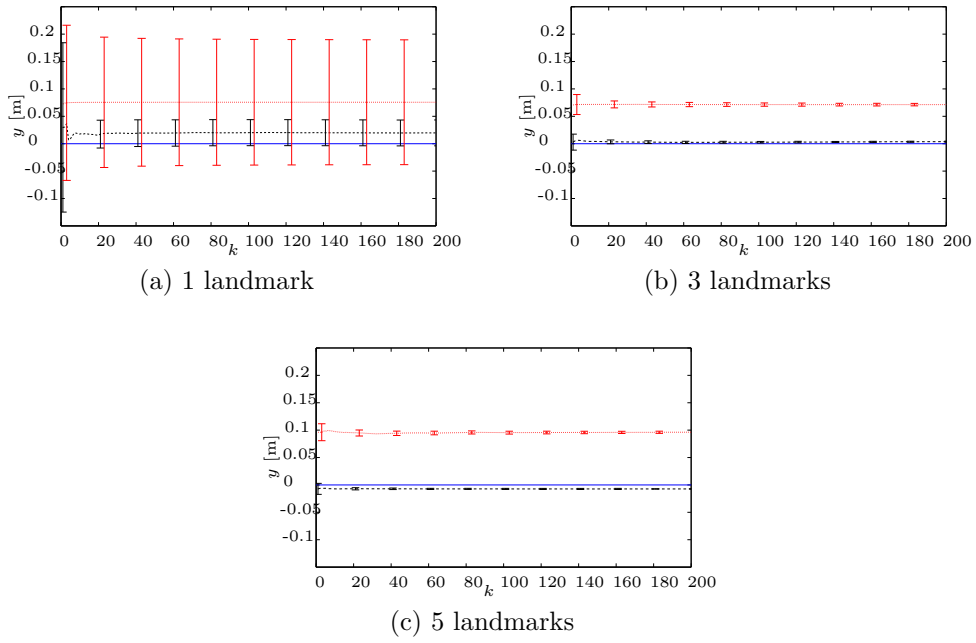


Figure 5.6: Localization experiment in static condition: measurement of the y coordinate. Blue solid line represents the ground truth, red dotted line represents the measurement computed with the Kinect, black dashed line represents the measurement computed with the Bumblebee2

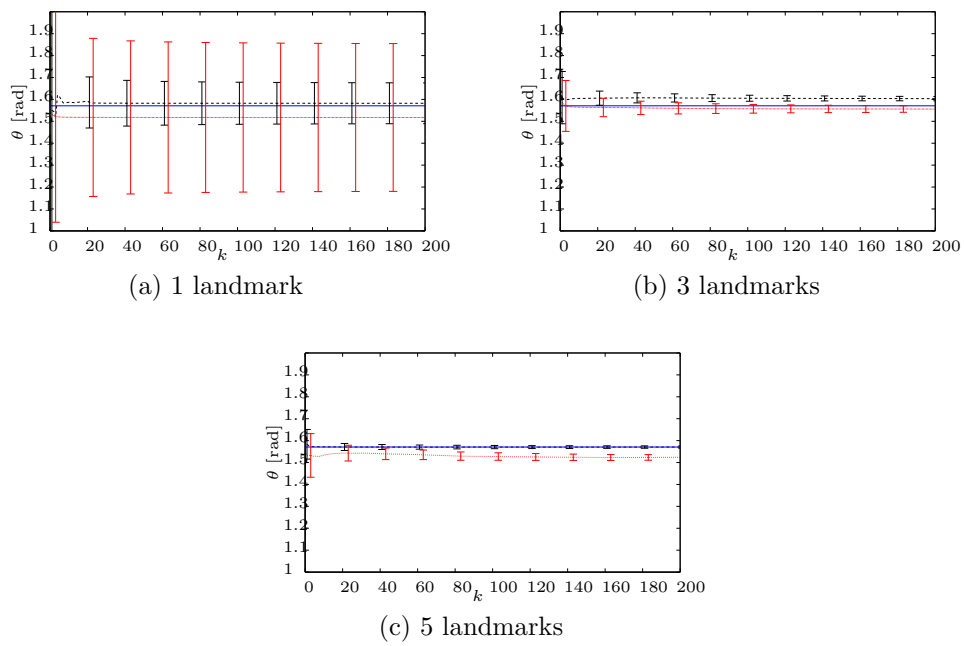


Figure 5.7: Localization experiment in static condition: measurement of the orientation, θ . Blue solid line represents the ground truth, red dotted line represents the measurement computed with the Kinect, black dashed line represents the measurement computed with the Bumblebee2

sensors in realistic industrial applications, that is in the presence of worst environmental conditions.

Moreover, future work will also aim at better characterizing the quality of the measurements of the sensors. In particular, we will investigate the presence of possible systematic errors, that might be filtered to enhance the obtained results. Furthermore, we will provide better estimates of the covariance matrices, possibly characterizing them with respect to range and bearing measurements (instead of measuring the variance along (x_s, y_s) axes).

Chapter 6

Lead-acid batteries on-line State-of-Charge estimate using an Extended Kalman Filter

This chapter describes a method for the estimation of the State-of-Charge of lead-acid batteries during the discharge phase that can be accomplished on-line, that is the State-of-Charge information is made available and is kept up-to-date in real-time while the device is operating. The algorithm is based on an Extended Kalman Filter that yields a measurement-corrected estimate of the battery State-of-Charge on the basis of established battery mathematical models. Model parameters are determined prior to running the Kalman Filter by a battery charger whose control software has been purposely modified. Eventually, experimental results based on the discharge of a 50Ah lead-acid battery are presented. This work, done in the framework of the project DiRò, is the result of a research carried out with Adel System, a company leading in the production of electric power sources (e.g. DC/DC converters, AC/DC converters, UPS systems, battery chargers, ...).

6.1 Introduction

Nowadays battery-powered devices are getting more and more widespread as increasingly efficient electronic circuits and systems are being developed for almost every field of application, from leisure personal connectivity to industrial and robot applications. Such devices have a noticeable advantage over their grid-powered counterparts in that they can obviously operate in the absence of mains availability thus ensuring a higher degree of operation freedom. However, since they are battery-operated, it is of utmost importance that at any instant their remaining operating time and therefore the available battery charge be accurately known. Established techniques for State-of-Charge (SoC) evaluation include current integration over time or SoC correlation with open-circuit battery voltage,

but the figures thus obtained are often unreliable mainly because actually SoC is dependent on environmental conditions and battery usage history: these figures are not easily accounted for by such techniques. In that respect, model-based predictive-corrective SoC evaluation approaches might yield more consistent results. The purpose of this work is to describe a method for the on-line evaluation of the state-of-charge of lead-acid batteries during discharge that does not rely on complex mathematics and thus can be implemented on a small-to-medium computational capabilities microcontroller.

6.2 State-of-Charge Estimation Techniques

Irrespective of its chemistry or technology any battery is characterized by its capacity, which is the total charge (usually expressed in Ampere-hours, Ah) that can be removed from a fully charged battery thus discharging it completely [67]. The State-of-Charge (SoC) of a battery can be intuitively defined at a given time as the quota of the total charge of a battery that is still usable at such time, as expressed in (6.1):

$$SoC = \frac{Q_{total} - Q_{removed}}{Q_{total}} \quad (6.1)$$

where Q_{total} and $Q_{removed}$ are the battery capacity and the charge utilized by the load so far respectively. However, for a given battery, the maximum value of $Q_{removed}$ is not an univocally defined quantity but it varies according to battery operating conditions, such as discharge current and temperature [68]. For example, for the lead-acid systems an increase in the discharge current magnitude causes a decrease in the battery effective capacity; such phenomenon is mathematically described by the Peukert Law [69]. Therefore, SoC is not an univocally defined quantity either; moreover it cannot be measured directly but it has to be estimated according to measurable parameters such as battery voltage, current, and temperature. One of the most common techniques for assessing the SoC of a battery is Ampere-hour counting [70], that is based on the measurement of the added or removed charge by integrating the battery charge or discharge current over a given time interval, as in (6.2), where the discharge current is measured:

$$Q_{removed} = \int_t^{t+\Delta t} i_{discharge}(t) dt \quad (6.2)$$

Thus an accurate measurement of the amount of Ampere-hours added during charge or removed during discharge can be accomplished provided the integrator electronics drift is kept under strict control and its offset is periodically corrected. Then (6.1) is applied to

compute SoC. In order to have more accurate SoC figures, the Peukert Law can be used to correct the value of Q_{total} based on the discharge current magnitude. Another commonly used technique for SoC evaluation is the measurement of the open-circuit voltage of a battery which has been kept disconnected from any load for some time [71]. The assumption is that the self-discharge and the other secondary chemical reactions inside the battery are neglected so that the open-circuit voltage of a battery is approximately equal to the sum of the individual cells equilibrium voltages, which in turn are dependent on the concentrations of the reacting species inside the cell electrolyte. Such concentrations are connected with the SoC of the battery. The SoC-vs-open circuit voltage relation must be known, either from the battery technical documentation or identified by means of curve fitting of data collected during battery charge and discharge cycles as in [72]; moreover the battery must have been rested to make sure that surface charge has been removed, but the time required for such condition to be met is not univocally defined. The method is clearly unpractical for any SoC estimation to be accomplished online. Battery models have been devised with diverse degrees of accuracy (and complexity) in order to provide a set of mathematical equations predicting SoC as a function of measurable battery parameters (voltage, current and temperature). In some models, the effect of a chemical process is modeled as a contribution term in an output equation (e.g. the overpotential voltage that is the effect of chemical processes is one of the contributions that superimpose to give the battery voltage under load). This is the case of the Shepherd model [73] described more in detail in the next paragraph. In other models, chemical processes taking place during charge or discharge inside the battery cells are modeled by their electrical analogous counterpart (e.g. the electrolyte diffusion modeled by a capacitor inside an electrical network). A review of established battery models can be found in [74]. Irrespective of the accuracy of the models, the main drawback of a merely predictive approach is that influence quantities (such as battery previous usage history) cannot be known a priori and are not easily taken into account into the model, so that large SoC estimation errors are likely to occur.

6.3 Proposed Method

In order to make the SoC estimation robust with respect to unpredictable influence quantities, a predictive-corrective approach has been adopted based on the Kalman filter (KF) [36], which is capable of estimating the state of a physical system on the basis of a time-evolution model of the state itself; such estimation is then corrected with mea-

surement data carried out on the actual system. Then, the so-corrected state estimate is used by the KF to predict the state of the physical system at the subsequent observation time. Since at any time the current state estimation is a function of the past evolution of the actual physical system and it is then corrected on the basis of measurements taken from the system, such predictive-corrective approach might in principle give state estimations that are more robust with respect to unpredictable influence quantities that are not taken into account in the system model as compared to merely predictive techniques. To keep the computational complexity as low as possible the battery was modeled as a physical system characterized by SoC as the only state variable. Equation (6.3) displays the time-evolution model used [67]:

$$SoC_{k+1} = SoC_k - \frac{i_k T_s}{C_{nom}} \quad (6.3)$$

where T_s is the sampling time interval, $i_k > 0$ is the battery discharge current measured during T_s and supposed to be constant therein, C_{nom} is the nominal battery capacity and the subscript k identifies the k^{th} value in a time-sequence. During discharge, (6.3) predicts a monotonically decreasing SoC time evolution, the difference between two subsequent state values amounting to the fraction of the fully-charged battery Ah rating that has been drawn by the load during the latest T_s time interval. In order to correct such state estimate, a model that connects the SoC with a battery measurable output parameter must be introduced. For such purpose the Shepherd model with the overvoltage contribution in the form suggested by Pfister [75], as displayed in eq. 6.4, is used.

$$V_{bat,k} = U_0 - R_{int}i_k - \frac{K_p}{SoC_k} \quad (6.4)$$

In (6.4), $V_{bat,k}$ is the battery voltage under load, U_0 is the battery equilibrium potential, R_{int} is the internal resistance of the battery, $i_k > 0$ is the battery discharge current and K_p accounts for the polarization overvoltage, i.e. the superposition of the deviations of each cell potential from the equilibrium value as a consequence of a non-zero net current density inside the battery. Such current density increases as the battery discharges because of the decreasing amount of active material available to take part to the battery discharge reaction until a limiting current density value is reached, resulting in the overvoltage term increase. This accounts for the apparent lower voltage under load in a discharged battery as compared to a charged one. A detailed explanation of the overvoltage phenomenon can be found in [76]. The KF estimates SoC as per equation 6.3. To correct such state estimate, a battery voltage measurement is carried out, and is compared against the

expected battery voltage as predicted by the Shepherd equation 6.4 evaluated at the SoC value estimated by the time-evolution model in eq. 6.3. Each step, by computing the filter gain value that minimizes the system state uncertainty, it is possible to get a more accurate estimation of the current value of such state variable. Equation 6.5 summarizes the system:

$$\begin{cases} SoC_{k+1} = A \cdot SoC_k + B \cdot i_k + q_k \\ V_{bat,k} = g_{Shepherd}(SoC_k, i_k) + r_k \end{cases} \quad (6.5)$$

where $A = 1$, $B = -T_s/C_{nom}$; q_k and r_k are the process and measurement noise contributions respectively, and are supposed to be uncorrelated, white and having normal probability distribution of variance Q and R respectively. The disturbances that affect the time evolution of the system state variable are modeled as a contribution q_k superimposed to the system input i_k . Due to the nonlinear relation between SoC and V_{bat} in eq. 6.4, an Extended Kalman Filter (EKF) has been used, which basically operates by linearizing eq. 6.4 at the current SoC value. The EKF that has been developed is summarized in Algorithm 6.

Algorithm 6

State of Charge Estimation ($SoC_{k-1}^+, P_{k-1}^+, i_{k-1}, V_{bat,k}$)

- 1: $SoC_k^- = A \cdot SoC_{k-1}^+ + B \cdot i_{k-1}$
 - 2: $P_k^- = A \cdot P_{k-1}^+ \cdot A^T + B \cdot Q \cdot B^T$
 - 3: $H_k = \left. \frac{\partial g_{Shepherd}(SoC_k, i_k)}{\partial SoC_k} \right|_{SoC_k=SoC_k^-}$
 - 4: $K_k = P_k^- \cdot H_k^T \cdot (H_k \cdot P_k^- \cdot H_k^T + R)^{-1}$
 - 5: $SoC_k^+ = SoC_k^- + K_k \cdot [V_{bat,k} - g_{Shepherd}(SoC_k^-, i_k)]$
 - 6: $P_k^+ = (I - K_k \cdot H_k) \cdot P_k^-$
-

Provided the model and measurement noise processes have been characterized (i.e. respectively Q and R are known), the filter is initialized with the best known initial estimates of the state variable (SoC) and its error variance P_k . After that, the filter operation consists of an iteration of a prediction algorithm (steps 1-3 in algorithm 6), followed by a correction algorithm (steps 4-6), the corrected values thus obtained being fed to the prediction algorithm the following iteration. In step 1 and 2, current SoC and its error variance are estimated based on the previously available corrected values. In step 3 the measurement function is linearized at the estimated current SoC value. Then, at the beginning of the correction algorithm, the Kalman gain is computed in step 4. The SoC value estimated in step 1 is then corrected in step 5 basically as a weighted average of

the estimated SoC value and the difference between the battery voltage measured during the latest T_s and the battery voltage value estimated by the Shepherd model in eq. 6.4 at the estimated SoC value. Eventually the SoC error variance P_k is corrected as well, at step 6. Such implementation is simple in being characterized by a single state variable and measurement equation thus requiring no matrix mathematics since all quantities are scalar.

6.4 System Identification

The operation of the Kalman filter discussed so far requires that not only the initial SoC estimate and the values of variance values Q and R be known, but also that the values of the battery parameters U_0 , R_{int} and K_p of the measurement equation 6.4 be determined. The SoC measurement system is to be implemented in a commercially available battery charger (ADEL System P/N: CBI2420A), which is a 24V multi-chemistry battery charger which has a MODBUS interface onboard that allows data logging and device control being accomplished. The system identification procedure is initiated immediately after the end of a charge phase, when initial SoC and U_0 values are determined by the battery charger. Prior to doing that, the battery charger removes surface charge that would lead to an U_0 overestimation. SoC value will be then updated by the filter as it iterates throughout battery discharge. R_{int} is evaluated by the life-test subsystem of the CBI2420A at the onset of the discharge phase to make sure that the chemical contribution to such parameter value is minimum. After U_0 and R_{int} have been determined, the over-voltage polarization term K_p is evaluated after the onset of discharge when SoC=1 (i.e. 100%) by monitoring the battery voltage under load V_{bat} until its initial steep decrease has terminated, then solving eq. 6.4 for K_p as displayed by eq. 6.6:

$$K_p = U_0 - V_{bat} - R_{int} i_{discharge} \quad (6.6)$$

The values assigned to Q and R determine the degree of confidence that the filter will assign to the predictive and the corrective contributions to the overall estimation of SoC respectively. The choice of the measurement variance R has been based on the evaluation of the battery voltage readout of the battery charger used which displays a 145mV resolution. The variance R of the battery voltage measurement has been assessed by statistical analysis of a set of 9500 samples of a known voltage (24118mV as measured by Agilent P/N: U3606A Multimeter/ DC Power Supply) applied to the battery terminals of the battery charger. The magnitude of such voltage has been chosen to be as close as

possible to one corresponding to a digital code transition of the battery voltage digitizer inside the CBI2420A device. Thus acquired data have been downloaded to a laptop through the MODBUS interface. Statistical analysis of such data set yielded a mean voltage value of $24119mV$, thus very close to the measured value and a variance $\sigma_V^2 = 4707mV^2$. Therefore $R = 4707mV^2$ was assumed. The disturbances that affect the time evolution of SoC are assumed to be modeled as a noise current contribution superimposed to the battery discharge current i_k . According to such assumption, Q shall be equal to the variance of the discharge current readout of the battery charger. For such purpose, with the aid of an electronic load (ARC S.r.l., P/N: DL2840) a constant discharge current has been imposed through the discharge current measurement circuit of the battery charger. A set of 30000 discharge current samples has been acquired by the battery charger and then downloaded to the laptop through the MODBUS interface. The magnitude of such constant current has been set to be as close as possible to one corresponding to a digital code transition of the discharge current digitizer. Statistical analysis of the collected data set yielded a variance value $\sigma_I^2 = 341mA^2$. Therefore $Q = 341mA^2$ was assumed.

6.5 Experimental Results

The developed SoC estimation algorithm is currently being developed on two separate platforms. The system identification procedure has been implemented on the commercially available 24V multi-chemistry battery charger which is managed by an 8-bit microcontroller (Freescale, P/N: MC9S08DZ32). Its firmware has been purposely modified in order to compute U_0 , R_{int} and K_p . On the contrary, the Kalman filter algorithm has been implemented on a Windows®-based laptop using MATLAB® (version R2006b, The MathWorks) environment. The experimental data have been collected from two 12V 50Ah open lead acid batteries wired in series (SAF 6050340) that have been fully charged using a IU0U0 charging curve with a maximum current of $C_n/10 = 5A$, C_n being the battery nominal capacity, or 50Ah. Then the battery has been discharged at a constant current down to 1.5V/cell (to prevent the battery from getting into deep discharge) using an electronic load. By means of the MODBUS interface of the battery charger, the battery voltage, the discharge current and the initial SoC information have been acquired and downloaded to the laptop where they have been then processed by the Kalman filter. A complete charge cycle was performed then, at the onset of discharge, system identification occurred as described, giving the following figures: $U_0 = 26684mV$, $R_{int} = 45m\Omega$ and $K_p = 1750mV$. The battery was then discharged at a constant current of 8.8A. With

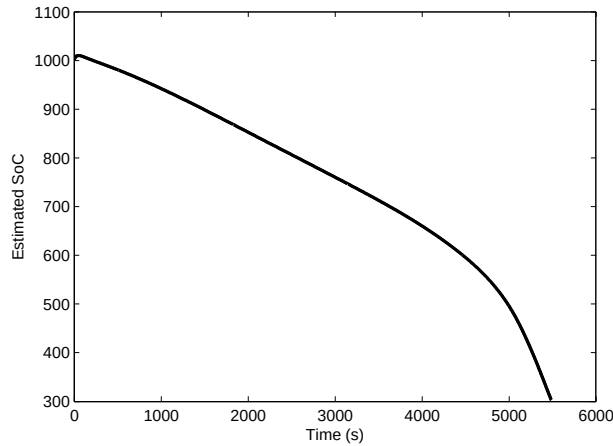


Figure 6.1: State-of-Charge time-course as estimated by the developed Extended Kalman Filter.

such figures both the SoC time-course worked out by the Kalman filter and the SoC estimation error variance time-course which describes the uncertainty associated to the SoC estimation have been evaluated. Figure 6.1 displays the SoC time-course as worked out by the developed Extended Kalman Filter during the battery discharge phase. SoC is expressed with a 10x scale factor so that 100% SoC is represented as 1000 in the graph.

Neglecting the initial transient, the estimated SoC time-course apparently decreases at an approximately constant rate over time until a steeper SoC decrease occurs when battery has almost reached the 1.5V/cell end-of-discharge voltage value despite the discharge current is kept constant throughout the discharge phase. Such behavior might be due to the current density increase inside the battery that occurs when the positive and negative plates are getting covered by lead sulphate and consequently their active area is becoming smaller, this in turn increasing the rate of the battery discharge. Figure 6.2 displays the time-course of the variance connected with the SoC estimation error. Actually, the quantity is descriptive of the uncertainty associated to the difference between the actual battery State-of-Charge and the one estimated by the Kalman filter [77]. Under a statistical perspective, the lower the SoC error variance, the closer the estimated SoC value to the actual state value.

6.6 Conclusions and Future Work

A system for the online estimation of the State-of-Charge of lead-acid batteries during discharge has been developed and has been tested in operation. The algorithm proposed

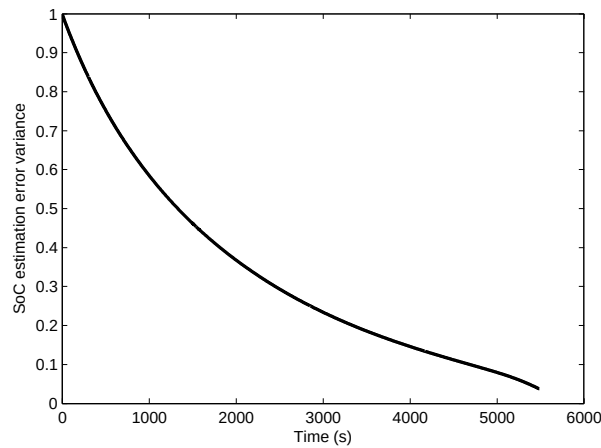


Figure 6.2: SoC error variance time-course as estimated by the developed Extended Kalman Filter.

is based on an Extended Kalman Filter that estimates the value of a single state variable of the system relying on established mathematical models. First, system identification is performed at the end of the charge phase and at the onset of the subsequent discharge phase. The identification process has been successfully implemented on an 8-bit microcontroller. Then, the filter thus initialized is operated iteratively. As it has been shown, no complex matrix mathematics is required, so we deem that this algorithm can be successfully implemented on a small 8-bit microcontroller as well. Future work includes extensive testing of the algorithm also on AGM and gel lead-acid batteries, implementation of the Kalman filter on the microcontroller and eventually the upgrade of the algorithm for the SoC evaluation during the lead-acid battery charging phase also.

Chapter 7

Project REMOCAL

In this chapter the realization of an autonomous robot for the calibration of tyres toe and camber measurement device will be discussed. In particular both the design of the robot mechanics and the algorithms will be presented. The work was done in collaboration with CORGHI s.p.a, a company leading in the production of car maintenance tools, and in particular in tyres substitution and repairing.

7.1 REMO - Robotic Equipment for Measuring by Optics

“REMO” (Robotic Equipment for Measuring by Optics, see Fig. 7.1b) is a measurement device able to provide a very precise measurement about the toe and the camber of car tyres. It is equipped with three cameras and three laser sensors. Two cameras are in the front of the device: the composition of data they provide can build a stereoscopic image. The third camera is on the side of the device, and it is used to provide the relative position and orientation of the REMO with respect to a chessboard-marker positioned at the end of the bridge the car is on top of (see Fig. 7.1a). Two lasers are on the front side of the device, between the two frontal cameras: they emit a cross-shaped beam on the tyre. One more laser is mounted under the cross-shaped laser emitters and is a distance meter. The two frontal cameras can then acquire an image of the projected cross on the tyre rim, and, by using an appropriate computer vision algorithm, the REMO can compute the orientation of the tyre with respect to the device itself. By measuring the orientation of all the tyres of a car an algorithm can then compute the toe and the camber of the wheels of the vehicle without using clamps. The next section will provide a further description of how the REMO system works.

The three cameras and the laser sensors need an accurate calibration, that by now is done manually. The goal of the project is to realize a mobile robot able to move

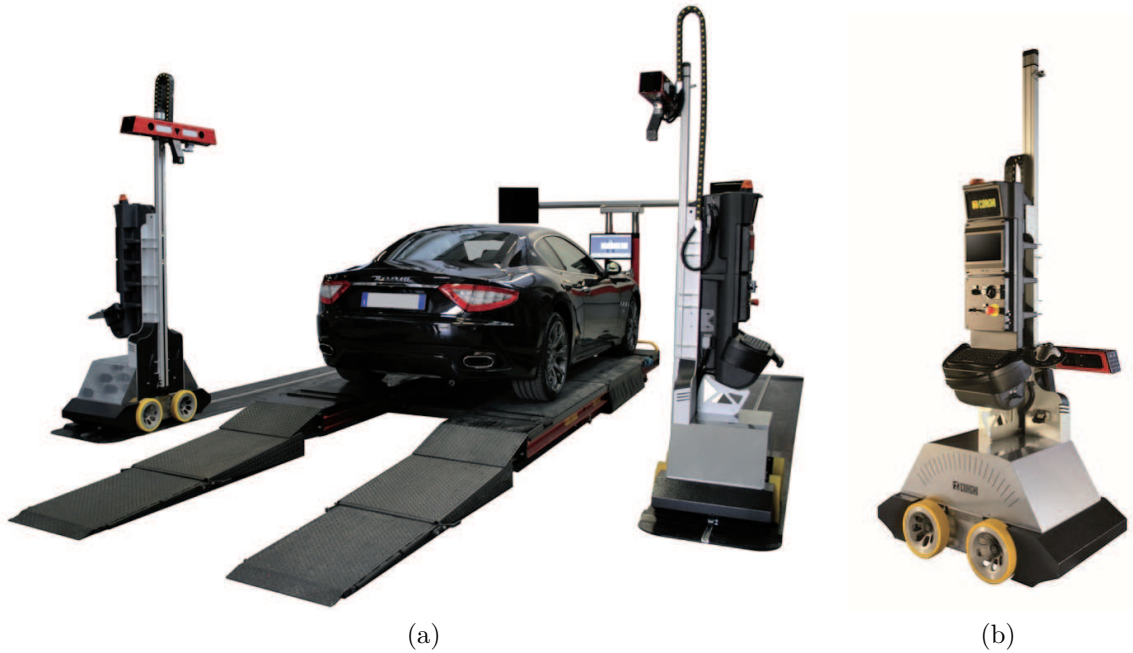


Figure 7.1: A Maserati on a bridge with two REMO devices on the right and left sides of the car (on the left) and a particular of the REMO device (on the right)

autonomously in the environment where the calibration takes place and to rotate the measurement device in order to take pictures from the cameras of the calibration chess-board in many orientations. By doing so it is possible to build the data structures needed for the calibration of the overall device.

The chapter is organized as follows: in Section 7.2 a description of the measurement procedure for the car's wheels alignment is provided. In Section 7.3 the calibration procedure is described. Section 7.4 describes the design requirements for the robot, while Sec. 7.5 and Sec. 7.6 describe, respectively, hardware architecture and software architecture. Section 7.7 conclusions and future works are reported.

7.2 How REMO works

REMO is composed by various sensors and components (see Fig. 7.2):

- Two Near Infra-Red (NIR) cameras on the front side. These are called Wheel Cameras.
- One NIR camera on a lateral side. This is called Field Camera.

- Two laser device on the front side (in between the two frontal cameras) able to project a structured light.
- One laser range finder on the front side (under the other two lasers).
- Three NIR LED (Light Emitting-Diode) boards able to enlighten the scene captured by the cameras. Two boards are on the front side, while the other board is on the side the third camera is.

A detailed description of the cameras parameters can be found in Table 7.1.

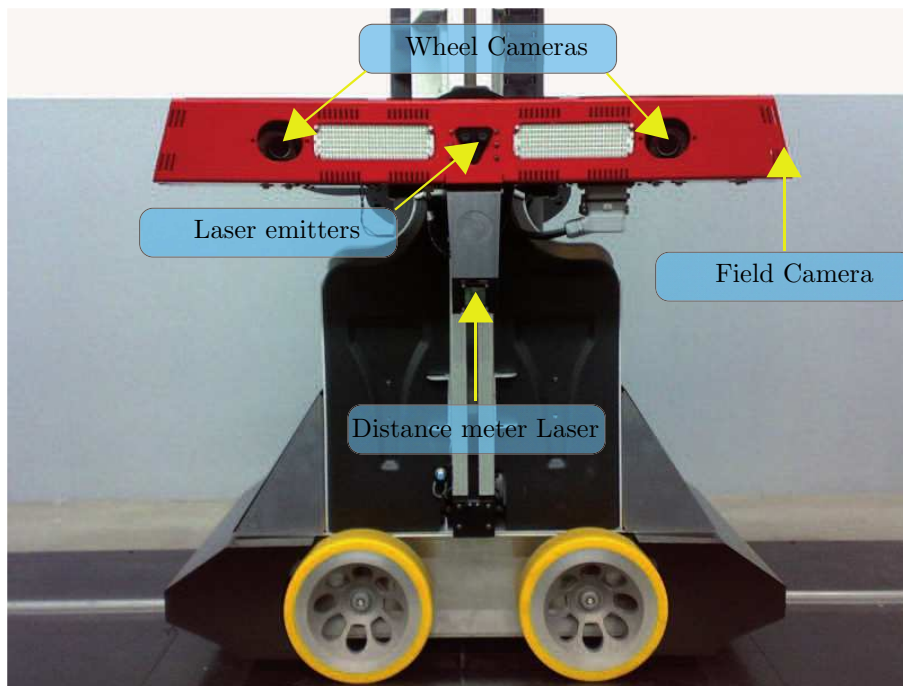


Figure 7.2: A frontal caption of the REMO.

In order to work properly, as can be seen in Fig. 7.1a, there must be two REMO devices, one on the left side and one on the right side of the vehicle. These two devices are mounted on two mobile robots able to move along a track fixed to the ground and parallel to the bridge the car is on. The two REMO devices are symmetrical and the field cameras points towards the front side of the vehicle, where the reference chessboards are. Figure 7.3 shows a schematic representation of the two symmetrical devices. It is important to note the different positions of the Camera 1 (C1) on the two devices, since all the measurements are reported with respect to the position of C1 camera.

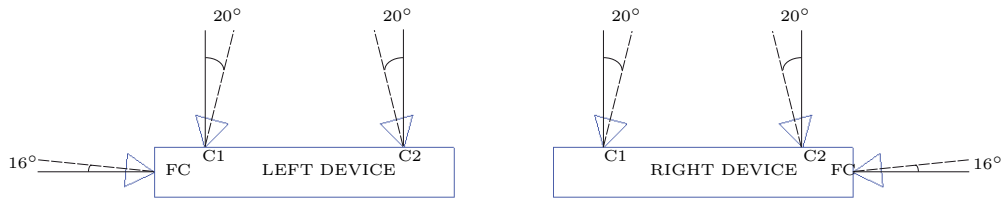


Figure 7.3: A schematic representation of the the two symmetrical REMO devices

Figure 7.4 represents the vision volumes of the REMO device. The picture represents the volume within which the camera has stereoscopic vision capabilities. This volume is obtained by intersecting the two vision pyramids of the wheel cameras. The “best focus” plane (highlighted in green) is far 900 *mm* from the device. The lateral pyramid represents the vision volume of the field camera.

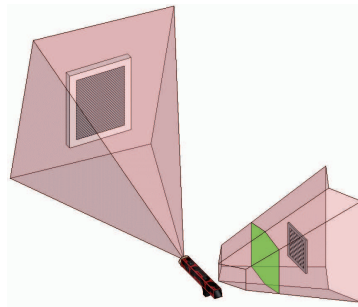


Figure 7.4: The vision volume of the REMO device.

Once the vehicle to calibrate is on the bridge in the garage, the measurement procedure can start. The tyre repairer moves the two robot carrying the REMO device in order to

Parameter	Wheel cameras	Field Camera
Effective Focal Length (EFL)	5	6
Object distance	700	4000
Image distance	5.04	6.01
Sensor aspect ratio	1.333	1.333
Paraxial magnification	−0.007194	−0.001502
Field Of View (FOV)	$69.9^\circ \times 55.33^\circ$ (82.29°)	$60.72^\circ \times 47.44^\circ$ (72.43°)
Object rectangle	978.56×733.92 (1223.2)	4686.29×3514.72 (5857.87)
Image rectangle	7.04×5.28 (8.8)	7.04×5.28 (8.8)

Table 7.1: Table summarizing the cameras parameters.

frame the right (or left) rear wheel of the vehicle while the laser range finder reads the distance of the wheel. The two laser devices on the front of the REMO project two perpendicular lines on the tyre. At the same time the two frontal LED boards emit a flash illuminating the wheel. The two cameras, then, acquire the image of the wheel. Since the cameras work in the NIR field, like the lasers and the LED boards, the acquired image represent an “X” and a circle (see Fig. 7.8) as other features of the surrounding scene, in visible light, are filtered out of the image. Then the lasers are shut down and acquisition procedure is repeated, acquiring a picture representing the footprints of the rim of the tyre. By subtracting the second picture to the first one, four intersections of the laser beams can be found. Let’s call these intersections as “Laser segments”. Through these four laser segments is possible to build a toroidal solenoid as can be evinced in Fig. 7.5

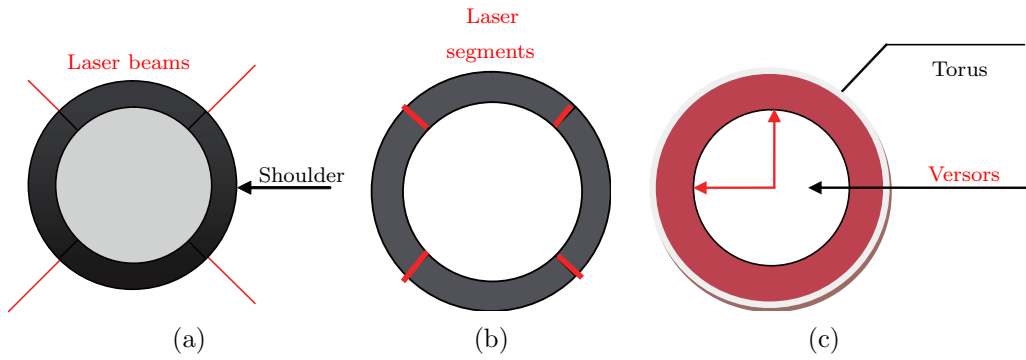


Figure 7.5: Three pictures summarizing the acquisition procedure of REMO

Once the toroidal solenoid is obtained, then it is possible to compute its center and the two versors (see Fig. 7.5c) which represent the toe and the camber of the tyre.

By framing the field target (i.e. the reference chessboards), which are two spatially fixed reference targets for the REMO (see Fig. 7.7), with the Field Cameras it is possible to obtain the geometric attitude of the vehicle and its characteristic angles. In particular all the measurements are finally reported according to the left reference target (see Fig. 7.6). By repeating the measurement for all the wheels of the vehicle, by building all the toroids and by computing their centers and versors it is possible to compute the symmetry axis of the vehicle and then all the derived measurements. All the obtained coordinates are computed with respect to the C1 camera.

In order to clarify the systems of coordinates, Table 7.2 reports the frames according to each component of the REMO device.

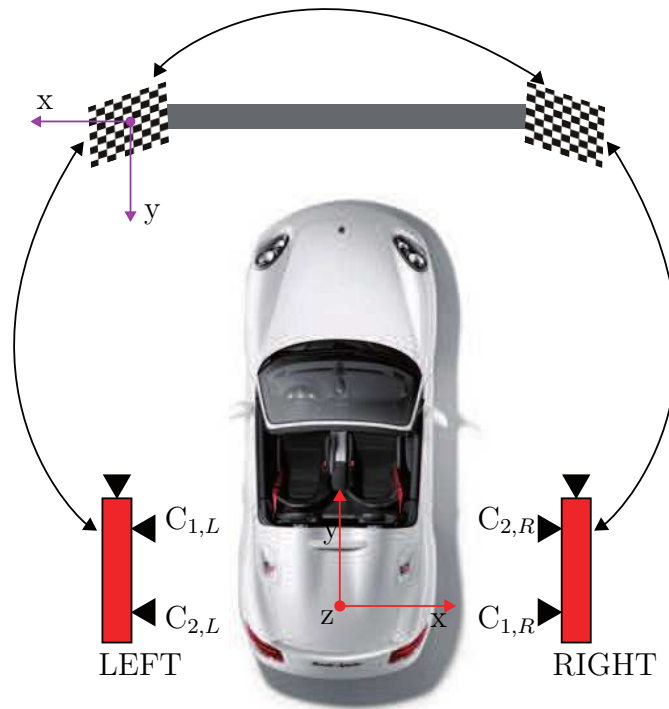


Figure 7.6: The layout of the garage. The red coordinates system represents the center of the rear wheelbase, while the purple coordinates system represents the global frame.

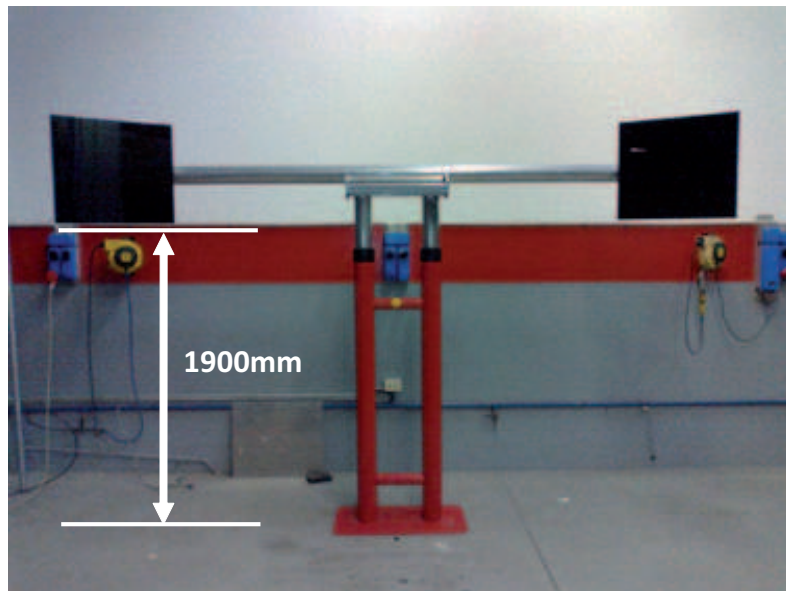


Figure 7.7: The reference targets bar

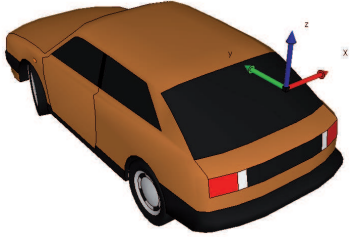
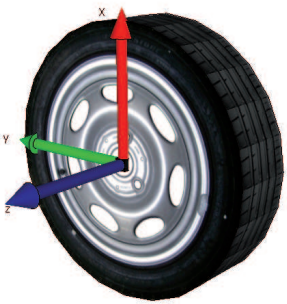
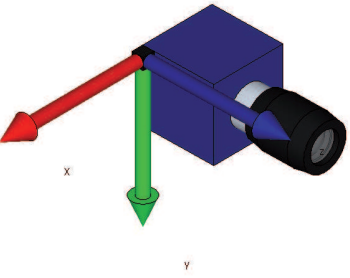
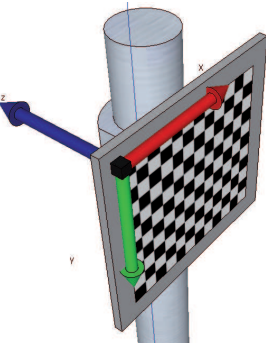
Vehicle: • x axis: pointing from the side of the vehicle, directed from the driver (left) to the passenger (right). • y axis: coincides with the traveling direction of the vehicle. • z axis: pointing upwards.	
Wheel: • x axis: pointing upwards. • y axis: coincides with the traveling direction of the vehicle. • z axis: coincides with the rotation axis of the wheel.	
Camera: • x axis: right direction (according to the direction of the z axis). • y axis: pointing downwards. • z axis: coincides with the optical axis of the camera itself.	
Target: • x axis: right direction (according to the direction of the z axis). • y axis: pointing downwards. • z axis: pointing towards inside the target itself.	

Table 7.2: Table summarizing the orientation of the coordinate systems for the REMO measurement system.

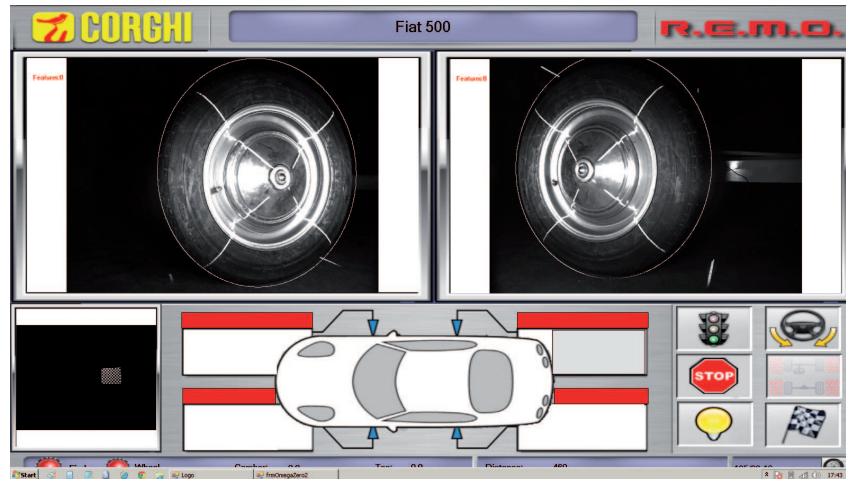


Figure 7.8: The mask generated by the laser projections

Once the distance of the two robots is obtained through the Field Cameras measurements, through the dimensions of the wheels (which can be easily obtained by reading a code on the tyre itself) and the coordinates of each wheels' center acquired previously, the two robots are placed in relation to each other obtaining the necessary measurements to the diagnosis of the vehicle wheel alignment.

The front and rear wheelbases of the vehicle are computed by interpolating the pairs of centers of the front and rear wheels, respectively. The misalignment of every single wheel can be obtained by comparing the axis of the wheels with the corresponding wheelbase. By doing so it is possible to compute the toe and the camber angles (see Fig. 7.9) of the wheels of the car. Finally the tyre repairer can change manually the orientation of the wheels by acting on the regulation screws.

7.3 The Calibration Procedure

In order for the REMO device to work properly and return the correct measurements for the attitude of the vehicle, for each sensor of REMO a very accurate calibration procedure must take place.

In this section the manual calibration procedure will be described.

7.3.1 The Mono-Calibration

First every single camera must be calibrated. We will refer to this procedure as the “mono-calibration”. The camera to be calibrated is mounted on a fixed structure in

order to frame a target which is moving on a rail in longitudinal direction (with respect to the camera), on the direction of the z axis of the camera (see Table 7.2). The target can also tilt on its x and y axis (for the direction of the axis, please refer again to Table 7.2). The mono-calibration tools can be seen in Figure 7.10.

The target is a chessboard with squares 20 mm side and has dimension 24×30 squares.

A software then moves back and forth the target and also tilts it in order for the camera to take shots of the chessboard with different positions and orientations. Once the software has taken the proper number of pictures of the chessboard a calibration software run in order to obtain the intrinsic and extrinsic parameters of the camera.

The intrinsic parameters are stored into the *intrinsic matrix* or also known as *calibration matrix*:

$$\mathbf{K} = \begin{bmatrix} f_x & f_\theta & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (7.1)$$

where f_x and f_y are the focal length in pixels, respectively on the x and y axis, f_θ is the skewness of the pixel. c_x and c_y , also represented in pixels, are the coordinates of the principal point of the sensor. Then it is possible to obtain the coefficients of the distortion polynomial. In this case a second order distortion polynomial is computed. More information about the mono-calibration procedure can be found in [79].

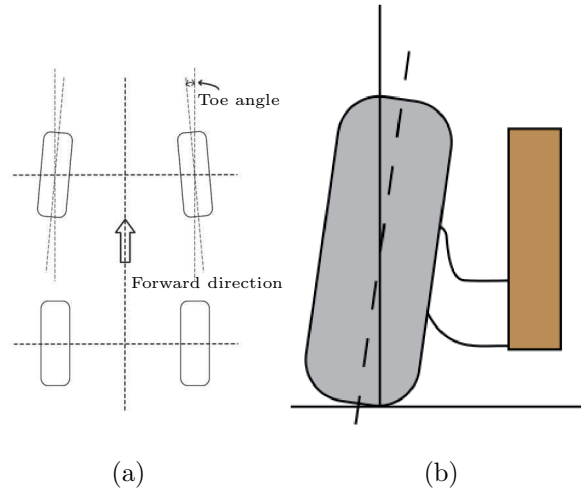


Figure 7.9: The toe angle (left picture) and the camber angle (right picture). In particular the toe angle represented in the first picture is positive and the camber angle in the second picture is negative. Font [78].

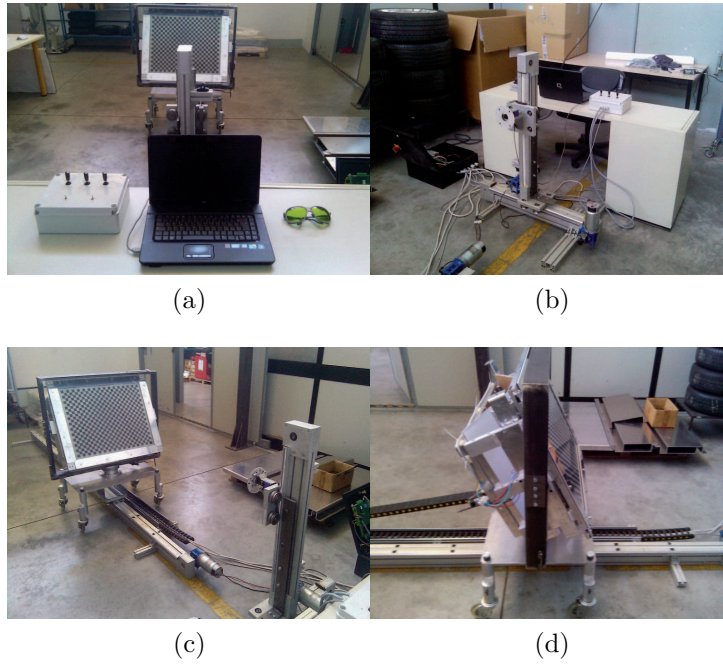


Figure 7.10: The tools for the mono calibration of each camera of the REMO device

7.3.2 The Stereo-Calibration and the Field–Wheel Cameras calibration

Then the stereo calibration is performed. The calibration procedure is now performed manually with a cart (see Fig. 7.11). The REMO device is mounted on a two degrees of freedom joint (see Fig. 7.11b) that allows the pitch and roll rotation of the device, which are respectively the rotation for the x axis and for the y axis. The third rotation (Yaw movement along the z axis) is provided by moving the whole cart. Figure 7.12 describes the rotation axis of the REMO and how its reference system is.

The objective of this calibration step is to obtain the rototranslations which computes the transformations between the cameras. The rototraslations are represented with homogeneous matrices (see Eq. 7.2).

$$\mathbf{H} = \left[\begin{array}{c|c} \mathbf{R}(\phi, \theta, \psi) & \begin{matrix} x \\ y \\ z \end{matrix} \\ \hline 0 & 1 \end{array} \right] \quad (7.2)$$

The calibration consists of framing a “*calibration target*” of known dimensions with two chessboards on it. The two frontal Wheel Cameras frame one chessboard while the Field

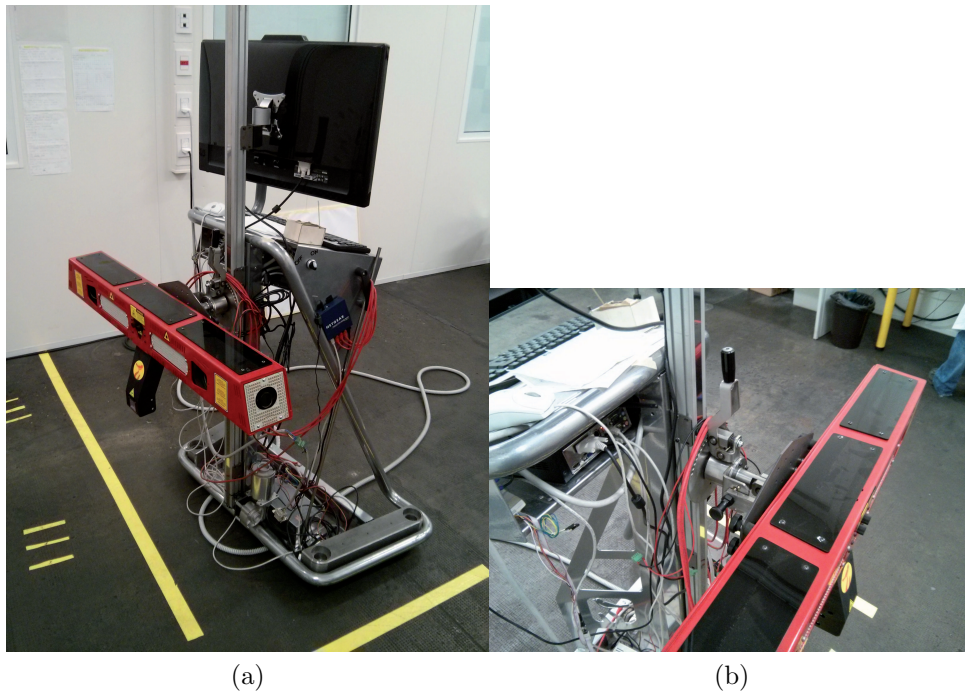


Figure 7.11: A picture of the REMO carrying cart used for the manual calibration of the measurement device.

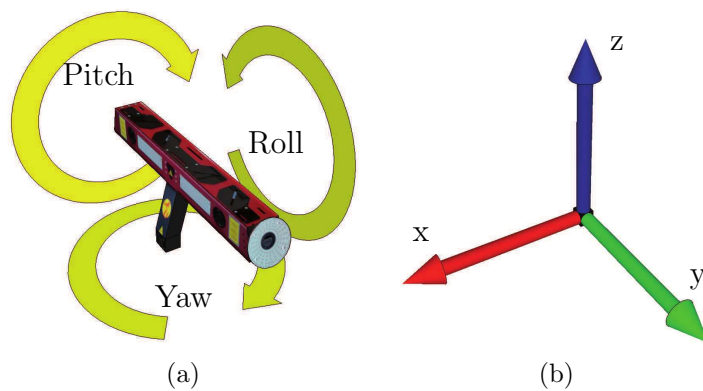


Figure 7.12: The Roll, Pitch and Yaw rotation of the REMO and its reference system

Camera frames the other chessboard (see Fig. fig: fieldWheelCal). The calibration of the two frontal Wheel Cameras is called “*Wheel Calibration*” and generates the $\mathbf{G}$ homogeneous matrix, which represents the rototraslation transformation between the C1 camera and the C2 camera. The calibration of the Field Camera is called “*Field–Wheel calibration*”, because it computes the homogeneous matrix representing the rototranslation between the Field Camera and the Wheel Camera C2, and generates the $\mathbf{G1}$ homogeneous matrix.

The target chessboards used for this calibration has squares 35 *mm* with dimension 10×11 squares.

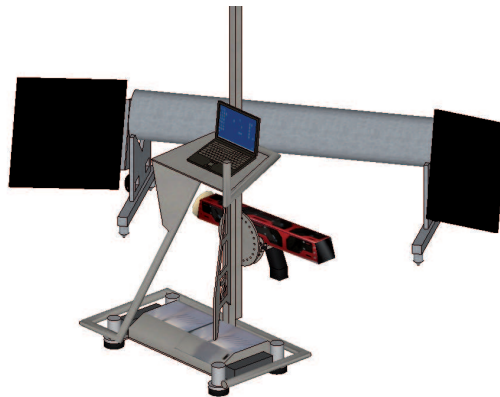


Figure 7.13: A schematic representation of the field/wheel calibration. The black squares represent the position of the calibration chessboards.

All the transformation are then referred to the C1 camera (see Fig. 7.6 for information about the systems of coordinates). For more information about the Stereo-Calibration see again [79].

7.3.3 Laser calibration

As for the cameras, also lasers need an accurate calibration procedure. First the laser range finder is calibrated. This time the procedure consists of a scaling of the range finder. The uncalibrated laser is used to measure the distance of an object posed in different known positions (see Fig. 7.14a), then the scaling which optimally interpolates all the measurements is performed.

Then the line-projector laser devices must be calibrated. The calibration consists of rotating the laser device itself in order for the projected lines to be orthogonal one to the other (see Fig. 7.14b). The lines are not visible to the human eyes, so the pictures of

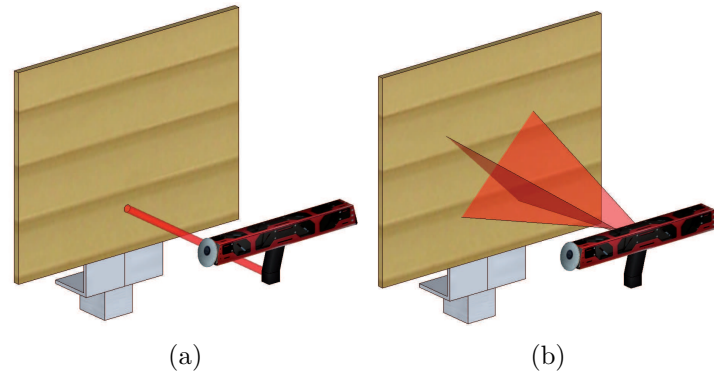


Figure 7.14: A schematic representations for the laser calibrations.

the lines are shot and then calibrated using the frontal C1 Wheel Camera of the REMO device.

7.3.4 Final Test

Once all the components of the REMO system (cameras, lasers line emitters and laser range finder) are calibrated, a final test is performed, in order to check if the calibration procedure was correct. The two REMO devices are mounted on the robots in a test arena. Then these devices are used to measure the toe and the camber angles of a so-called “*zeroed car-chassis*”, which is showed in Fig. 7.15. This dummy car was built with very high precision and mechanical tolerances and resulting in a chassis with toe and camber angle equal to zero.



Figure 7.15: The “*zeroed car-chassis*”

The convergence measurement is performed on the dummy car and the reading error of the REMO devices are then computed. Through a special software some “New Addenda of Precision” (NAP) are computed and used to correct the operation of the REMO device.

7.3.5 Target Bar Calibration

Like all the other components of the REMO device, also the targets bar (see Fig. 7.7) is a fundamental component of the REMO system and needs an accurate calibration. The calibration occurs after the target bar has been fixed to the ground of the tyre repair garage.

The target bar is framed with a pre-calibrated generic camera and then multiple shots are taken. Then the rototraslation transformation is computed through the optimal homogeneous matrix which best interpolates the obtained measurements.

7.4 Design Requirements

The stereo calibration of the Wheel Cameras needs angled shots of the targets. Here is the calibration procedure:

1. Tilt the REMO device until it reaches $\beta = 25^\circ$ and $\gamma = 45^\circ$
2. Move the REMO device along the x, y, z axis in order to frame the target inside the stereopsis area and the field target inside the vision pyramid of the field camera.

Table 7.3 summarizes the excursions needed for the stereo-calibration of the wheel cameras of the REMO device.

Parameter	Excursion
x	$-100 \text{ mm}, +500 \text{ mm}$
y	$\pm 1000 \text{ mm}$
z	$-500 \text{ mm}, +1000 \text{ mm}$
α	0°
β	$\pm 25^\circ$
γ	$\pm 45^\circ$

Table 7.3: Table summarizing the stereo-calibration required excursions

The procedure for the field-wheel calibration is:

1. The device is moved subsequently along the x, y, z axis with respect to the initial position and the maximum translation are computed, in order for the robot every position to frame correctly inside the stereopsis area the wheel target and inside the vision pyramid the field target. Let be $\mathcal{C}$ the set of all this positions.

2. The device is moved in every extreme position of the set $\mathcal{C}$ and then the device is tilted in order to find the maximum roll, pitch and yaw angles.

Table 7.4 summarizes the excursion needed for the field-wheel calibration.

Parameter	Excursion
x	$\pm 400 \text{ mm}$
y	$\pm 160 \text{ mm}$
z	$\pm 280 \text{ mm}$
α	$\pm 15^\circ$
β	$\pm 25^\circ$
γ	$\pm 15^\circ$

Table 7.4: Table summarizing the required excursions for the field-wheel cameras calibration

7.5 Hardware design

This section describes step by step the design process of the REMOCAL robot. First the conceptual design of the robot and the generated concepts are reported, then the hardware architecture is described with also the choice of the motors. Finally the realization of the robot is reported, with a full description of the prototype.

7.5.1 Conceptual Design

According to the requirements listed in Sec. 7.4, four concepts for possible robots have been proposed. For every generated concept pros and cons have been evaluated according to the requirements, the cost of the overall system, the controllability of the robot.

The most critical aspects to considerate during the design procedure were the kinematic configuration and the lifting device for moving the REMO on the z axis.

REMOCAL Concept 1

The concept robot, which can be seen in Fig. 7.16, has the following features:

- Differential drive kinematic structure: two motors can actuate two frontal wheels while two rear “caster” wheels (for solving the stability issue)

- PRO: differential drive robots are very easy to control and has a relatively low drift. The odometry provided in the short time period by the wheels is enough reliable. It is also easy to implement trajectories following.
 - CON: the differential drive configuration requires two independently actuated motors.
- REMO device z axis displacement: the REMO device is connected to a support able to shift vertically thanks to a torque screw nut. The lifting device is actuated through a motor and the displacement is computed by measuring the rotation of the crankshaft through an encoder the motor is equipped with.
 - PRO: the lifting device needs a low power motor to work with, thanks to the small reduction ratio provided by the torque screw nut. It also ensure very high repeatability.
 - CON: very slow movements.

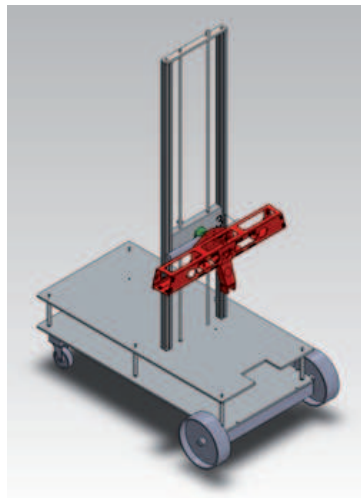


Figure 7.16: Concept 1

REMOCAL Concept 2

The second generated concept idea of the robot, illustrated in Fig. 7.17, has the following features:

- Tricycle kinematic structure: one motor actuates a steerable wheel. Other four wheels (two caster wheels and two fixed wheels) are used for the stability issue.

- PRO: Low drift. It is easy to follow trajectories.
- CON: the required space for maneuvers is higher, with respect to other kinematic configurations.
- REMO device z axis displacement: the REMO device is connected to a linear guide actuated by a motor.
 - PRO: the linear guide is a commercial component. It ensures high repeatability.
 - CON: this kinematic configuration requires two motors: one for actuate the wheel and one for steering it. Otherwise a motorwheel is needed. High cost.

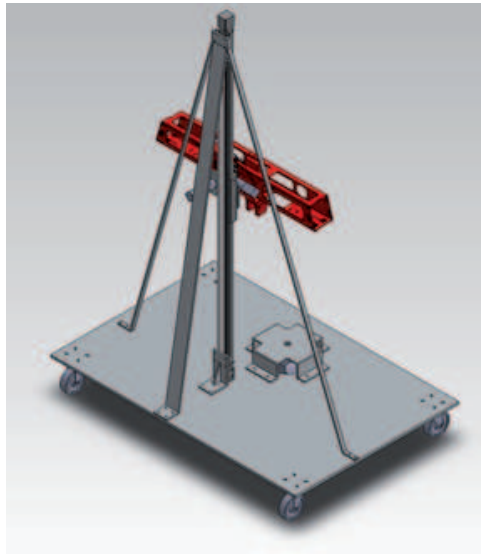


Figure 7.17: Concept 2

REMOCAL Concept 3

This concept is a slight modification of the Concept 2 described previously. The robot is provided with a further motor able to adjust the position of the REMO device, in order to increase the positioning precision.

The kinematic structure and the lifting device are the same used for Concept 2. Figure 7.18 describes the additional degrees of freedom.

- PRO: the positioning precision is increased. The robot requires less maneuvers for making the REMO device reach the desired configuration.
- CON: two linear guides (and two motors) are required. The cost and the size of the robot are increased.

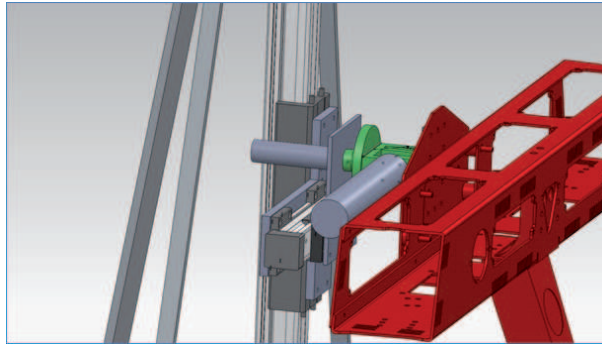


Figure 7.18: The additional degrees of freedom

REMOCAL Concept 4

The last concept is a recombination of the first one. Figure 7.19 represents the conceptual prototype.

- Kinematic structure with “Mecanum wheels” (also known as “Swedish wheels” or “omni-directional wheels”).
 - PRO: movements possible in every direction. It is easy for the robot to follow very complex trajectories.
 - CON: very high drift. The efficiency of the wheels is very low. A motor is required for each wheel (four).
- REMO device z axis displacement: the REMO device is connected to a support able to shift vertically thanks to a torque screw nut. The lifting device is actuated through a motor and the displacement is computed by measuring the rotation of the crankshaft through an encoder the motor is equipped with.
 - PRO: the lifting device needs a low power motor to work with, thanks to the small reduction ratio provided by the torque screw nut. It also ensure very high repeatability.
 - CON: very slow movements.

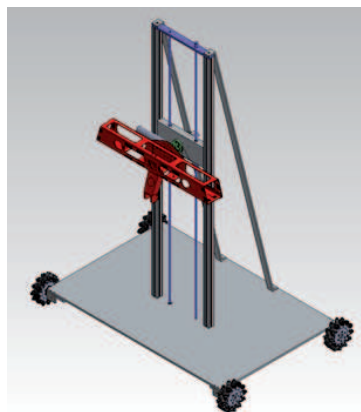


Figure 7.19: Concept 3

7.5.2 Concepts Recombination and Evaluation

After a deep evaluation process of the proposed conceptual ideas, a concept recombination occurred. The final result was a robot with a differential drive kinematic configuration and a linear motor as lifting device.

Two additional degrees of freedom for the REMO device are given through a wrist carrying the REMO and connected to the linear guide's cart, in order to allow the roll and pitch rotations for the device. This configuration was chosen also because the robot's chassis is the same of the already commercialized robot.

Figure 7.20 represents a sketch of the final concept.

The robot is actuated through five motors:

- two motors actuating the wheels.
- one motor actuating the linear guide.
- one motor for the roll movement of the REMO device.
- one motor for the pitch movement of the REMO device.

The chosen motors are from MAXON. It was decided to have motors from the same producer in order to have a system with the same communication protocol and with the same libraries to implement controls with.

Table 7.5 summarizes the components of the motors, while Table 7.6 summarizes the mechanical characteristics of the motors. On the side of this table the name of the driver is also reported. For space reason and for clarity the encoders are not reported.

The robot can be powered by two 24 V_{DC} different battery packs, which also work as a counterbalance for the robot itself. The first one is a 120 Ah battery and weights 30 kg , while the second one is a 41 Ah battery and weights 14.5 kg . A USB-webcam is also mounted on the frontal side of the robot. It is used to compute the position of the robot with respect to markers put in the working environment. This function is explained in detail later in this thesis.

The chosen prototype allows freedom of movement on the horizontal plane, thus requirements on the distance from the calibration targets are fully satisfied, as well as the orientation about the z axis (the yaw rotation) requirement. The configuration of the 2 DOF joint (the wrist holding the REMO device), allows rotation of $\pm 45^\circ$ for the roll and $\pm 90^\circ$ for the pitch (see tables 7.3 and 7.4).

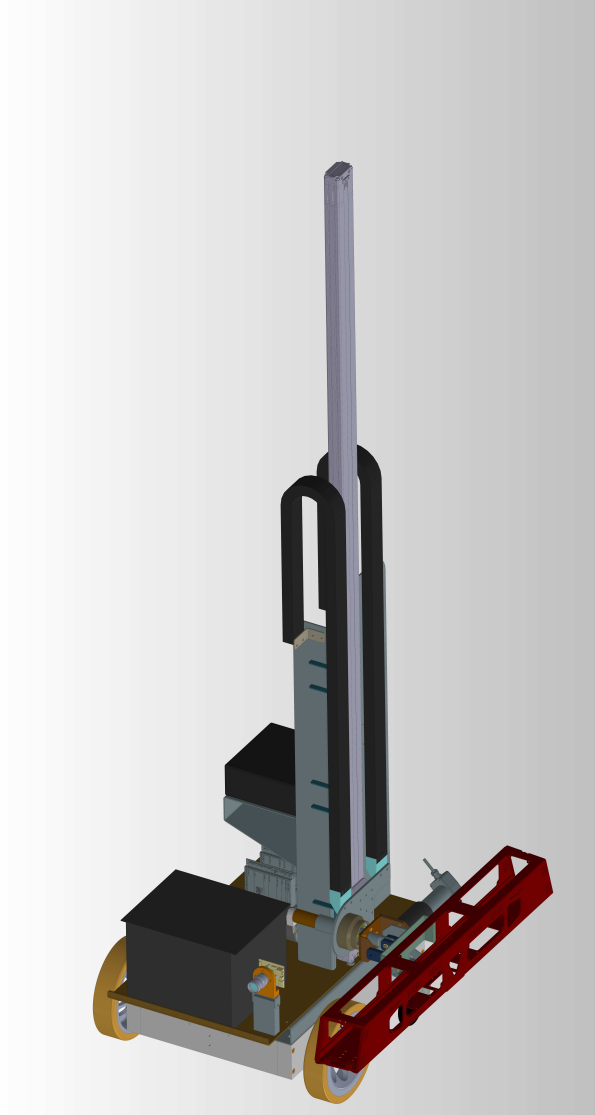


Figure 7.20: A sketch representing the chosen conceptual prototype

Figure 7.21 represents the overall sketch of the REMOCAL robot with some indicative dimensions. Figure 7.22 represents some particular sketches of the designed robot: the 2 DOF (Degrees Of Freedom) joint and the wheelbase.

7.5.3 Hardware architecture

Each motor system (motor + gear reduction + brake + encoder) is connected to a driver. Each driver, which has a unique ID code, is connected to a CAN (Controller Area Network) bus. The so built network allows to communicate to every motor by indicating the ID code of the motor. Every driver can also be connected to a laptop computer through a USB cable. This allows to interact with the network of the motors by connecting a single computer to a driver (the gateway). Figure 7.23 outlines the structure of the drivers and motors links.

On the robot two kinds of drivers have been mounted: the EPOS2 70/10 controllers, which require $11 - 70 V_{DC}$ voltage supply, and the EPOS2 50/10 controller (only for the motor responsible of the roll movement), which requires $11 - 50 V_{DC}$ voltage supply. Drivers (which are equipped with a DC/DC converter voltage stabilizer) are directly connected to the battery packs.

In order to work properly, the drivers connected to those motors equipped with brakes

Motor function	Component
Linear	Planetary Gearhead GP 81 A $\varnothing 81$ mm, 20 – 120 Nm
	RE 65 $\varnothing 65$ mm, Graphite Brushes, 250 W
	Encoder HEDL 9140, 500 CPT, 3 Channels, With Line Driver RS 422
	Brake AB, 24 V_{DC} , 2.5 Nm
Pitch	Planetary Gearhead GP 62 A $\varnothing 62$ mm, 8 – 50 Nm
	EC 45 $\varnothing 45$ mm, brushless, 250 W, CE approved, with Hall sensors
	Encoder HEDL 9140, 500 CPT, 3 Channels, With Line Driver RS 422
	Brake AB 28, 24 V_{DC} , 0.4 Nm
Roll	Planetary Gearhead GP 42 A $\varnothing 42$ mm, 3 – 15 Nm, Ceramic Version
	RE 35 $\varnothing 35$ mm, Graphite Brushes, 90 W
	Encoder HEDL 5540, 500 CPT, 3 Channels, With Line Driver RS 422
	Brake AB 28, 24 V_{DC} , 0.4 Nm
Wheels	Planetary Gearhead GP 62 A $\varnothing 62$ mm, 8 – 50 Nm
	EC 45 $\varnothing 45$ mm, brushless, 250 W, CE approved, with Hall sensors
	Encoder HEDL 9140, 500 CPT, 3 Channels, With Line Driver RS 422

Table 7.5: Summary table for the motors' components

Roll	Motor		Controller: EPOS2 50/5
	Name	RE 35 $\varnothing 35$ mm, Graphite Brushes, 90 W	
	Voltage	24 V	
	Nominal speed	7000 RPM	
	Nominal torque	0.101 Nm	
	Gear reducer		
	Name	Planetary Gearhead GP 42 A $\varnothing 42$ mm, 3 – 15 Nm	
	Reduction ratio	637/2	
	Shaft diameter	10 mm	
	Maximum torque	15 Nm	
	Brake		
	Name	Brake AB 28, 24 V _{DC} , 0.4 Nm	
Voltage	24 V _{DC}		
Torque	0.4 Nm		
Pitch	Motor		Controller: EPOS2 70/10
	Name	EC 45 $\varnothing 45$ mm, brushless, 250 W, with Hall sensors	
	Voltage	24 V	
	Nominal speed	4300 RPM	
	Nominal torque	0.331 Nm	
	Gear reducer		
	Name	Planetary Gearhead GP 62 A $\varnothing 62$ mm, 8 – 50 Nm	
	Reduction ratio	41553/176	
	Shaft diameter	8 mm	
	Maximum torque	50 Nm	
	Brake		
	Name	Brake AB 28, 24 V _{DC} , 0.4 Nm	
Voltage	24 V _{DC}		
Torque	0.4 Nm		
Linear motor	Motor		Controller: EPOS2 70/10
	Name	RE 35 $\varnothing 35$ mm, Graphite Brushes, 90 W	
	Voltage	24 V	
	Nominal speed	3710 RPM	
	Nominal torque	0.485 Nm	
	Gear reducer		
	Name	Planetary Gearhead GP 81 A $\varnothing 81$ mm, 20 – 120 Nm	
	Reduction ratio	250047/4913	
	Shaft diameter	14 mm	
	Maximum torque	120 Nm	
	Brake		
	Name	Brake AB 44, 24 V _{DC} , 2.5 Nm	
Voltage	24 V _{DC}		
Torque	2 Nm		
Wheels	Motor		Controller: EPOS2 70/10
	Name	EC 45 $\varnothing 45$ mm, brushless, 250 W	
	Voltage	24 V	
	Nominal speed	4300 RPM	
	Nominal torque	0.331 Nm	
	Gear reducer		
	Name	Planetary Gearhead GP 62 A $\varnothing 62$ mm, 8 – 50 Nm	
	Reduction ratio	204687/2057	
Shaft diameter	8 mm		
Maximum torque	50 Nm		

Table 7.6: Table summarizing the mechanical data of the motors. On the left column the movements are reported. The central column reports the components name and their characteristics. On the right column the driver's name are reported.

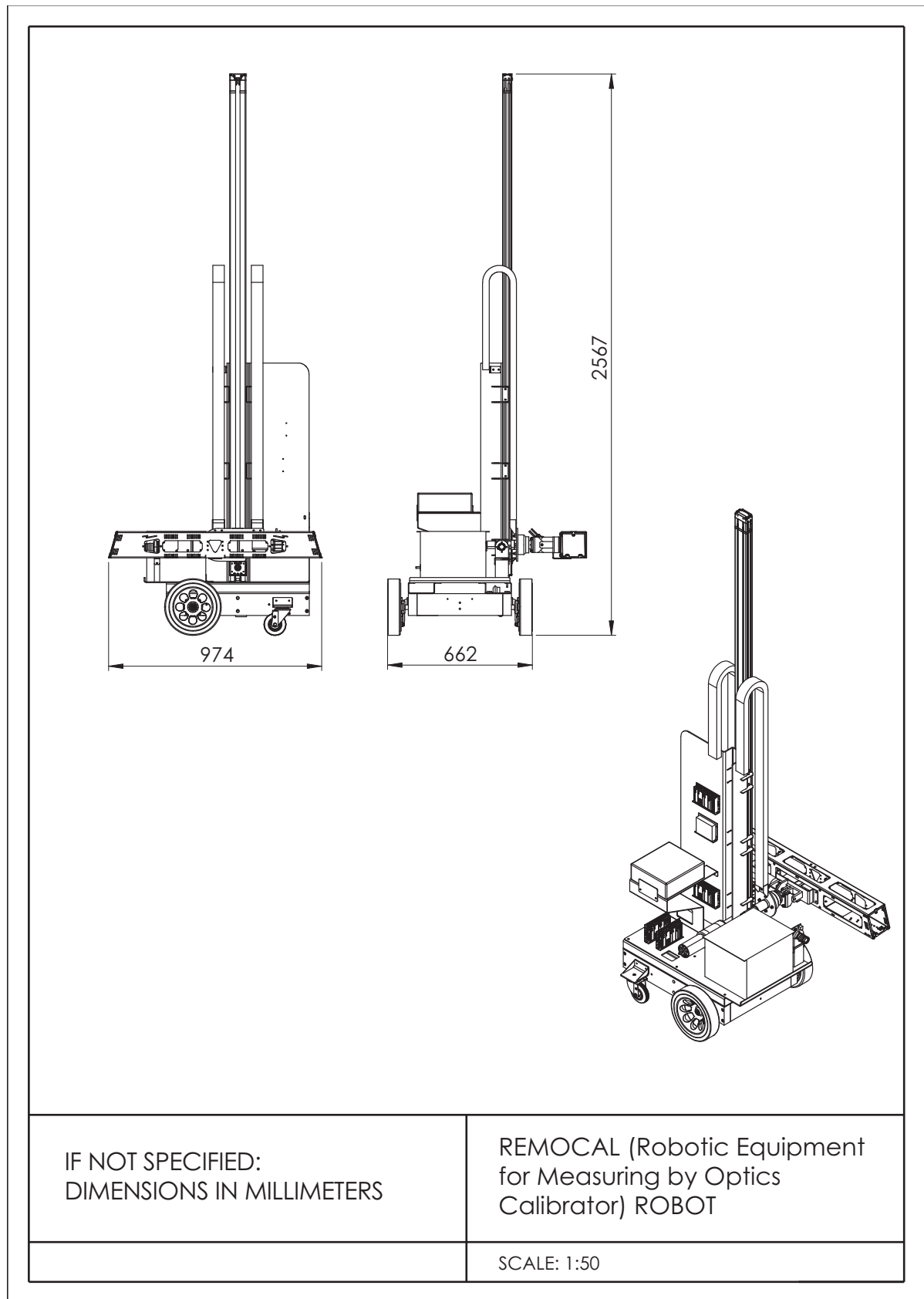


Figure 7.21: An overall sketch of the designed robot.

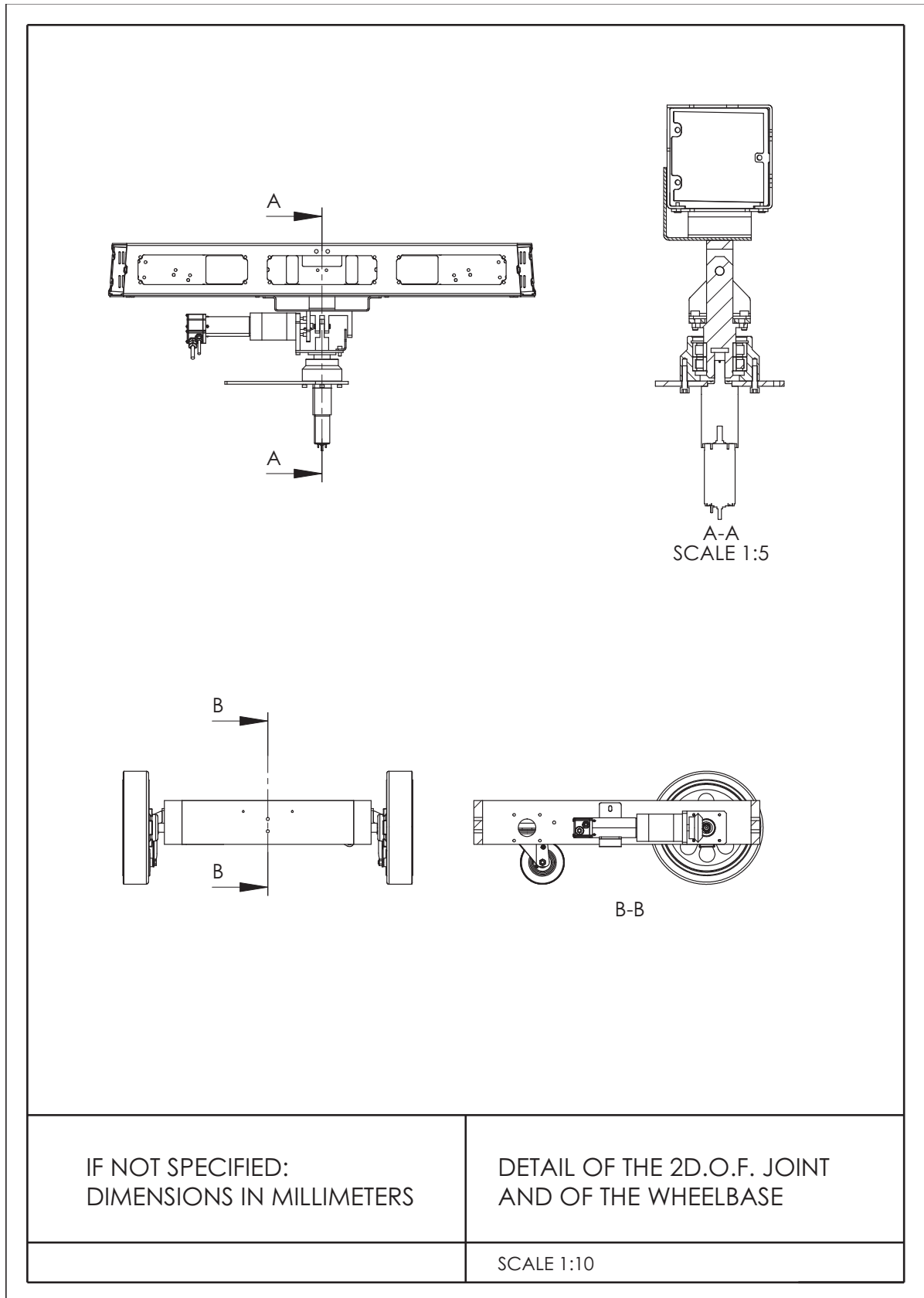


Figure 7.22: A particular view of the 2 DOF joint and of the wheelbase.

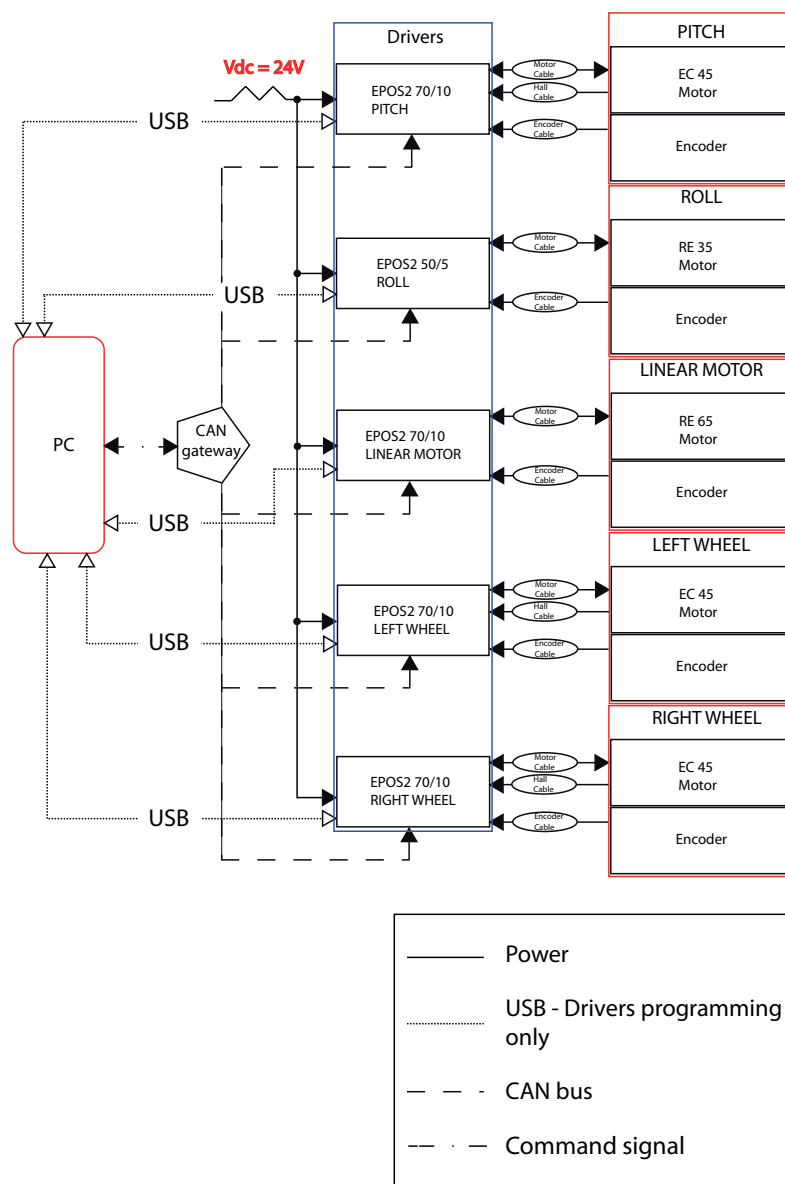


Figure 7.23: A scheme illustrating the connections between drivers and motors for the REMOCAL robot. The scheme also describes the signal exchange between the motors, the drivers and PC. The red box groups the motors, while the blue box groups the drivers.

need a further voltage supply ($24 V_{DC}$ stabilized) which is provided by a DC/DC converter connected to the battery pack.

Each driver implements three classical control loops for DC motors: a current (inner) loop, a velocity loop and a position (outer) loop. Drivers implement for each loop PID (Proportional Integrative Derivative) controllers, which can be auto-tuned. Drivers also implement a low level safety control for the position and the speed of the motor. In order to avoid collision between the limbs of the robot a safety procedure has been implemented both at low level (on the drivers) and at higher level (the software controlling the robot).

On the front of the robot a USB-webcam is mounted. This camera is used to provide visual odometry information, i.e. the position of the robot according some markers positioned in the working environment of the robot.

7.6 Algorithm Description

The REMOCAL robot's controller is realized through C++ programming language. The programming environment is Microsoft Visual Studio 2010, under Microsoft Windows 7 operating system.

The developed software is built with an “object-oriented” strategy, in order to facilitate the reuse of the code and the implementation of other code trunks. Three different typologies of classes have been developed: the first one provides an interface with the hardware of the robot, while the second one is more “algorithmic” and provides tools for the computation of the position of the robot into its working area and for navigating from a point to another. The third typology derives from the well known OpenCV libraries [80–82]. It is used for markers recognition and for visual odometry. These class are respectively described in the following three subsections.

7.6.1 Implementation of the interface with the robot

Figure 7.24 represents a class diagram of the classes which create an interface with the robot. Here is a brief description:

- Maxon: a class provided with the dll files for the implementation of the communication with the motor through the CAN.
- MaxonMotor: a class representing motors mounted on the REMOCAL robot. This class allows to make motors move and read their states (e.g. like position, velocity, etc...).

- Remocal: a class representing the whole robot system. It contains as many instances of the object MaxonMotor as the number of motors on the robot (nominally five). This class also contains the robot position on the map (intended as (x, y, ϑ)) and the altitude and orientation of the REMO device. Through this object is possible to move the robot. In few words, this class generates an object representing a virtual version of the REMOCAL robot.

7.6.2 Navigation algorithms implementation

Then a higher control level is created through functions which allow the robot to compute and keep track of the position and orientation of each its reference system, and to compute the velocity commands needed for the robot to reach the desired configuration (i.e. the position of the robot into the global reference frame). This is done with two more classes: “*Trajectory_planner*” and “*tf*” (the name was inspired by the “*tf*” package of ROS [66]).

The class “*Trajectory_planner*” provide a very simple way for mobile robots parking tasks (see task explanation in 2) based on differential flatness. The velocity command for the robot to move towards a goal, are generated according to the distance (both linear and angular) of the robot to the goal configuration (see [34]). For the kinematic model, refer to eq. 2.10. In particular commands, given the coordinates of the goal configuration $(\tilde{x}, \tilde{y}, \tilde{\vartheta})$, are generated through these formulas

$$\begin{cases} \rho = \sqrt{\Delta x^2 + \Delta y^2} \\ \alpha = -\vartheta + \text{atan2}(\Delta y, \Delta x) \\ \beta = \Delta \vartheta - \alpha \end{cases} \quad (7.3)$$

where

$$\begin{cases} \Delta x = \tilde{x} - x \\ \Delta y = \tilde{y} - y \\ \Delta \vartheta = \tilde{\vartheta} - \vartheta \end{cases} \quad (7.4)$$

The control law then are the following:

$$\begin{cases} v = k_\rho \rho \\ \omega = k_\alpha \alpha + k_\beta \beta \end{cases} \quad (7.5)$$

where $k_\rho, k_\alpha, k_\beta$ are gains. The controlled system is locally exponentially stable if $k_\rho > 0$, $k_\beta < 0$, $k_\alpha - k_\rho > 0$. Obviously v and ω are respectively the linear velocity and the angular velocity of the robot. By changing the gains $k_\rho, k_\alpha, k_\beta$ it is possible to make the path towards the goal smoother or sharper.

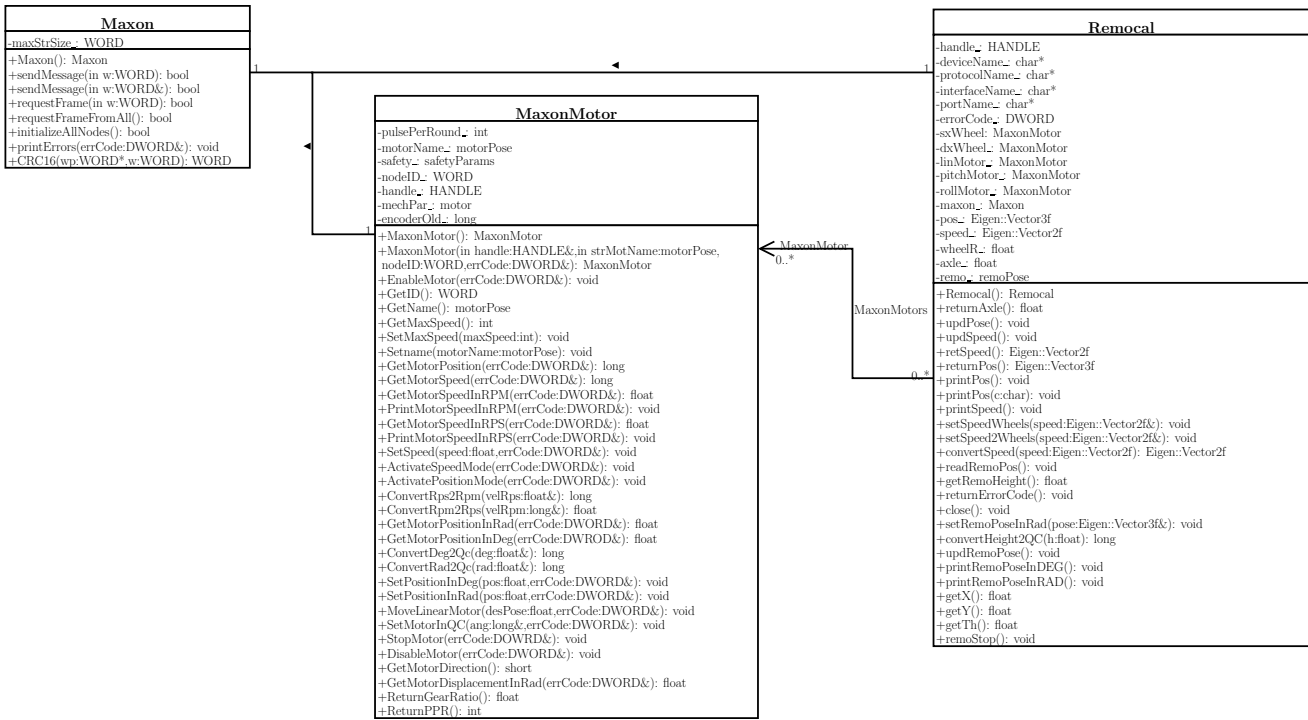


Figure 7.24: UML class diagram for the REMOCAL robot

7.6. Algorithm Description

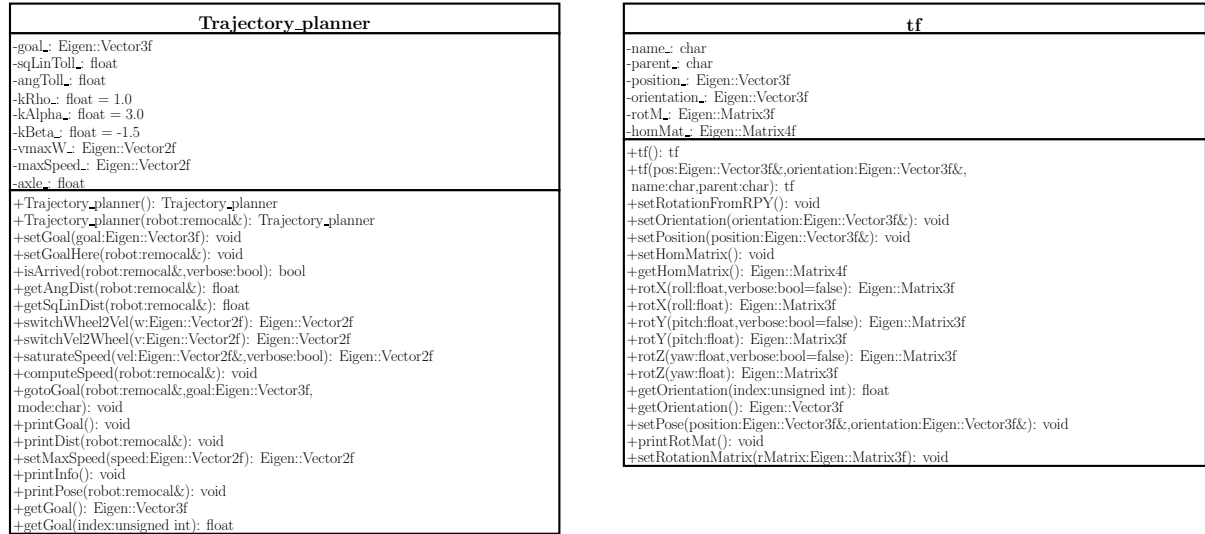


Figure 7.25: UML class diagram for the trajectory planner and for the transformation computation

The “*tf*” class provides a way to represent and connect the various reference frames present in the robot. Figure 7.26 represents the reference frames present in the REMOCAL robot. The transformation between frames are computed through homogeneous matrices.

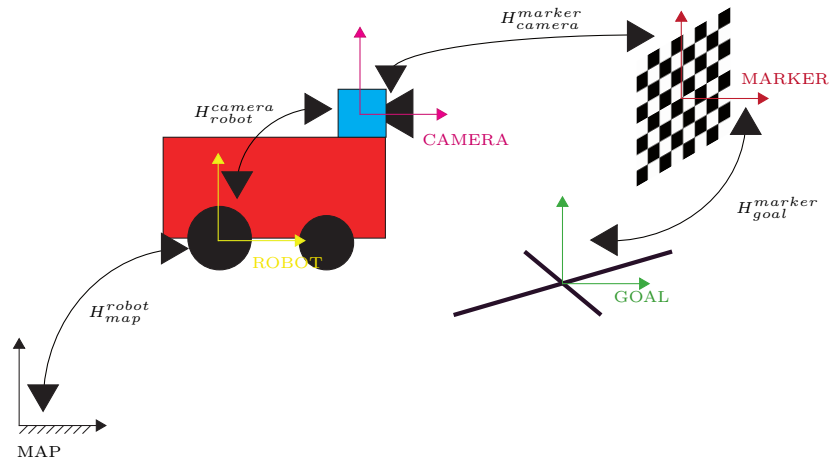


Figure 7.26: Scheme summarizing the frames tree structure

The homogeneous matrices represent the transformation needed to travel from a frame to another. The matrix is defined as follows:

$$\mathbf{H}_{\text{parent}}^{\text{actual}} = \left[\begin{array}{c|c} \mathbf{R}(R, P, Y) & \begin{matrix} x \\ y \\ z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (7.6)$$

This representation allows to compute transformation between frames with relative ease. For example, referencing Fig. 7.26, the position of the goal in the global “map” frame can be extracted by the resulting matrix

$$\mathbf{H} = \mathbf{H}_{\text{map}}^{\text{robot}} \cdot \mathbf{H}_{\text{robot}}^{\text{camera}} \cdot \mathbf{H}_{\text{camera}}^{\text{marker}} \cdot \mathbf{H}_{\text{goal}}^{\text{marker}} \quad (7.7)$$

which also has the structure explained in eq. 7.6.

7.6.3 OpenCV derived classes

As anticipated earlier, the visual odometry is provided by a USB–webcam. The camera, which has been calibrated off–line through the same calibration algorithm cited in 7.3.1, provides information about the position of the robot. In particular some particular markers have been used. Each marker (i.e. the one represented in fig. 7.27) has an internal code given by 5 words of 5 bits each. The codification employed is a slight modification of the hamming code. In total, each word has only 2 bits of information out of the 5 bits employed. The other 3 are employed for error detection. As a consequence, we can have up to 1024 different IDs. The main difference with the hamming code is that the first bit (parity of bits 3 and 5) is inverted. So, ID 0 (which in hamming code is 00000) becomes 10000 in our code. The idea is to prevent a completely black rectangle from being a valid marker ID, with the goal of reducing the likelihood of false positives with objects of the environment (see [83]). The functions used for this task are the same proposed in [83].

The robot, as soon as it is activated, starts searching for a marker with a certain code. The goal position, who is represented according to the coordinates of the marker, is wrote on an external text file. Once the robot frames the marker it starts moving towards the goal by extracting the desired position with eq. 7.7 . When the robot has reached the goal it starts moving the REMO device, by actuating the linear motor and the 2 DOF wrist joint carrying the REMO itself. The REMO can shot some pictures of the chessboard framed in that position. Finally the robot starts searching for another marker or it returns to the home position.

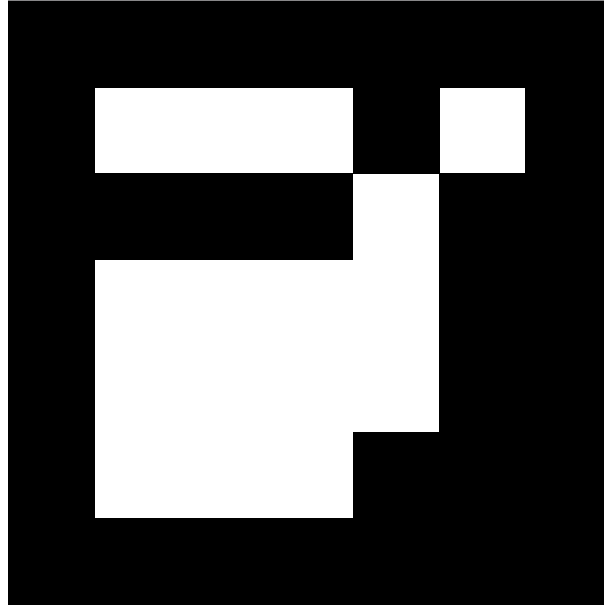


Figure 7.27: The marker used for determining the position of the robot.

7.7 Discussion and Future Works

In this chapter the hardware and software design for a mobile robot capable of autonomously calibrate a tyre convergence measurement device have been proposed. The realized robot can substitute the human intervention in the stereo calibration, in the mono calibration and in the field-wheel calibration of the REMO device.

In order to build a more efficient as possible robot, many conceptual solutions were analyzed. Then after a deep evaluation one concept was developed and, finally, realized. The robot has five motors: two motors for the wheels, one motor for the linear motor, and two motors for the 2 DOF wrist (one for the roll and one for the pitch).

The control software was designed as modular as possible, in order to make the software reusable and easily expandable with new features.

Future works expect to make robot even more autonomous by autonomously mapping the markers and by deciding the best positions for the calibration procedure of the REMO. Also a commercial implementation of the designed robot are under development. In particular a fully autonomous version of the REMO robot which can navigate in the workshop without being bound to a rail: the robot moves between cars parked into the workshop and autonomously takes pictures of the parked cars' wheels. Then it will disassemble damaged or old wheels, mount new wheels and calibrate their alignment

autonomously. This will be done foreseeing a fully autonomous workshop, where driverless cars (like [84] and [85]), according to scheduled maintenance, enter the workshop and the robot repairer autonomously takes care of the car.

Chapter 8

Conclusion

This thesis dealt with the technology transfer of new mobile robotics related finding into real world application. In particular the presented work was done in the framework of two projects involving several companies.

First the DiRò project was presented: the thesis focused mainly in three activities. The first activity (Chapter 4) consisted in the design and development of a smart lawnmower. The robot exploited a particular variant of the Extended Kalman Filter (EKF) to localize itself in the working area. In order to make the final product competitive on the market, only low cost sensors were used: specifically the robot used a three axis gyroscope, an RFID reader and an RF antenna to localize itself. Furthermore a smart coverage strategy was also proposed, in order for the robot to accomplish its mowing task without useless redundancy and without leaving area of the garden with higher grass.

The second activity, reported in chapter 5 consisted into the characterization of 3D sensors: in particular the performance of a stereocamera and an RGB-D sensor, the Microsoft Kinect, were compared. Furthermore the sensors were tested through the implementation of an EKF-based localization algorithm.

The final part of the DiRò project (chapter 6) consisted into the development of a Lead-acid batteries State of Charge estimator based on the Kalman Filter. The model parameters were experimentally obtained. The SoC estimator was then tested through experiments.

Then the thesis ends in chapter 7 describing the REMOCAL project. The project consisted in the design and realization of an autonomous mobile robot capable of calibrating a car tyres toe and camber measurement device, the REMO. Both the hardware and software design are described.

Appendix A

Background on bayesian probability theory

In this Appendix, a brief introduction to Bayes' probability theory is provided.

A.1 The probability

Formally, the probability of the output of a casual discrete event is defined as:

$$p(A) = \frac{\text{Number of possible outputs which favors event } A}{\text{Total number of outputs}} \quad (\text{A.1})$$

The probability an event could be A or B is the sum of the single probabilities:

$$p(A \cup B) = p(A) + p(B) \quad (\text{A.2})$$

If the probability of two distinct events is independent (event A does not influence event B) the probability both the events occur is the product of the single probabilities:

$$p(A \cap B) = p(A)p(B) \quad (\text{A.3})$$

Finally, the conditioned probability of an event A with respect an event B is the probability that A occurs by knowing the event B is verified:

$$p(A | B) = \frac{p(A \cap B)}{p(B)} \quad (\text{A.4})$$

If A and B are independent the equation results:

$$p(A | B) = \frac{p(A \cap B)}{p(B)} = \frac{p(A)p(B)}{p(B)} = p(A) \quad (\text{A.5})$$

A.2 Bayes' rule

The Bayes rule relates a conditional probability $p(A | B)$ to its inverse $p(B | A)$:

$$p(A | B) = \frac{p(B | A) p(A)}{p(B)} = \begin{cases} \frac{p(B | A) p(A)}{\sum_{\neg A} p(B | \neg A) p(\neg A)} & \text{Discrete} \\ \frac{p(B | A) p(A)}{\int p(B | \neg A) p(\neg A) d(\neg A)} & \text{Continuous} \end{cases} \quad (\text{A.6})$$

where the notation $\neg A$ indicates the complementary event of A . It is important to note that the denominator in eq. A.6 does not depends on $p(B)$. In fact, often, $p(B)^{-1}$ is referred as the Bayes' normalizer and denoted with η , thus:

$$p(A | B) = \eta p(B | A) p(A) \quad (\text{A.7})$$

More information on Bayes' probability theory applied to mobile robotics can be found on [4].

Acknowledgements

I would like to thank Dr. Cristian Secchi (my PhD advisor) and Prof. Cesare Fantuzzi for giving me the opportunity to undertake the PhD studies. During these years I received a lot of advice and help from them.

Many thanks go to all my colleagues of the ARSControl laboratory: Lorenzo, Francesco, Valerio, Federica, Alan, Lorenzo, Luca, Elena, Matteo, Federico, Giuseppe.

My greatest thanks go to my wife Anna, who supported me during these three years.

Finally a special thank go to my mother: thank you for everything.

Bibliography

- [1] BRUNO SICILIANO AND OUSSAMA KHATIB. *Springer Handbook of Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2008.
- [2] HOWIE M CHOSSET. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.
- [3] TOSHIFUMI TSUKIYAMA. **Mobile robot localization from landmark bearings**. In *XIX IMEKO World Congress Fundamental and Applied Metrology*, 2009.
- [4] SEBASTIAN THRUN, WOLFRAM BURGARD, AND DIETER FOX. *Probabilistic Robotics*. The MIT Press, 2005.
- [5] GREG WELCH AND GARY BISHOP. **An Introduction to the Kalman Filter**. University of North Carolina at Chaper Hill, 1995.
- [6] RUDOLPH EMIL KALMAN ET AL. **A new approach to linear filtering and prediction problems**. *Journal of basic Engineering*, **82**(1):35–45, 1960.
- [7] RUDOLPH E KALMAN AND RICHARD S BUCY. **New results in linear filtering and prediction theory**. *Journal of Basic Engineering*, **83**(3):95–108, 1961.
- [8] JULIEN DIARD, PIERRE BESSIERE, AND EMMANUEL MAZER. **A survey of probabilistic models using the bayesian programming methodology as a unifying framework**. 2003.
- [9] SHEILA M LOVE, DENNIS P GROGAN, AND JOHN A OGDEN. **Lawn-mower injuries in children**. *Journal of Orthopaedic Trauma*, **2**(2):94–101, 1988.
- [10] M. AGRAWAL AND K. KONOLIGE. **Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS**. **3**:1063–1068, 2006.
- [11] S Y CHEN AND SENIOR MEMBER. **Kalman Filter for Robot Vision : A Survey**. *Industrial Electronics, IEEE Transactions on*, **59**(11):4409–4420, 2012.
- [12] HEESUNG CHAE, CHRISTIAND, SUNGLOK CHOI, WONPIL YU, AND JAEIL CHO. **Autonomous navigation of mobile robot based on DGPS/INS sensor fusion by EKF in semi-outdoor structured environment**. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1222 –1227, oct. 2010.
- [13] WONKYO SEO, SEOYOUNG HWANG, JAEHYUN PARK, AND JANG-MYUNG LEE. **Precise outdoor localization with a GPSINS integration system**. *Robotica*, **31**(03):371–379, July 2012.

- [14] G. NEWSTADT, K. GREEN, D. ANDERSON, M. LANG, Y. MORTON, AND J. MCCOLLUM. **Miami Redblade III: A GPS-aided Autonomous Lawnmower**. 7(2):115–124, December 2008.
- [15] H. ZHAO, M. CHIBA, R. SHIBASAKI, S. XIAOWEI, C. JINSHI, AND Z. HONGBIN. **SLAM in a dynamic large outdoor environment using a laser scanner**. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1455 –1462, may 2008.
- [16] A.D. SMITH, H.J. CHANG, AND E.J. BLANCHARD. **An outdoor high-accuracy local positioning system for an autonomous robotic golf greens mower**. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2633 –2639, may 2012.
- [17] KLAUS FINKENZELLER AND RACHEL WADDINGTON. *RFID Handbook*. 2003.
- [18] DOMINIK JOHO, CHRISTIAN PLAGEMANN, AND WOLFRAM BURGARD. **Modeling RFID signal strength and tag detection for localization and mapping**. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3160 –3165, may 2009.
- [19] D. HAHNEL, W. BURGARD, D. FOX, K. FISHKIN, AND M. PHILIPPOSE. **Mapping and localization with RFID technology**. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, pages 1015 – 1020 Vol.1, april-1 may 2004.
- [20] SEBASTIAN SCHNEEGANS, PHILIPP VORST, AND ANDREAS ZELL. **Using RFID Snapshots for Mobile Robot Self-Localization**. In *Proceedings of the 3rd European Conference on Mobile Robots (ECMR 2007)*, pages 241–246, Freiburg, Germany, September 19-21 2007.
- [21] BYOUNG-SUK CHOI, JOON-WOO LEE, AND JU-JANG LEE. **An improved localization system with RFID technology for a mobile robot**. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 3409 –3413, nov. 2008.
- [22] BYOUNG-SUK CHOI, JOON-WOO LEE, JU-JANG LEE, AND KYOUNG-TAIK PARK. **A Hierarchical Algorithm for Indoor Mobile Robot Localization Using RFID Sensor Fusion**. *Industrial Electronics, IEEE Transactions on*, 58(6):2226 –2235, june 2011.
- [23] MAURO BOCCADORO, FRANCESCO MARTINELLI, AND STEFANO PAGNOTTELLI. **Constrained and quantized Kalman filtering for an RFID robot localization problem**. *Autonomous Robots*, 29(3-4):235–251, June 2010.
- [24] ANDREA GOLDSMITH. *Wireless communications*. 2005.
- [25] A. AWAD, T. FRUNZKE, AND F. DRESSLER. **Adaptive Distance Estimation and Localization in WSN using RSSI Measures**. In *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, pages 471 –478, aug. 2007.

- [26] A.N. RAGHAVAN, H. ANANTHAPADMANABAN, M.S. SIVAMURUGAN, AND B. RAVINDRAN. **Accurate mobile robot localization in indoor environments using bluetooth.** In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4391–4396, may 2010.
- [27] J. GRAEFENSTEIN AND M.E. BOUZOURAA. **Robust method for outdoor localization of a mobile robot using received signal strength in low power wireless networks.** In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 33–38, may 2008.
- [28] ALESSIO LEVRATTI, MATTEO BONAIUTI, CRISTIAN SECCHI, AND CESARE FANTUZZI. **An Inertial / RFID Based Localization Method for Autonomous Lawnmowers.** In *10th International IFAC Symposium on Robot Control, 2012, SYROCO 2012. Proceedings*, September 2012.
- [29] ALESSIO LEVRATTI, CRISTIAN SECCHI, AND CESARE FANTUZZI. **A low cost localization algorithm for an autonomous lawnmower.** In *9th International Workshop on Robot Motion and Control, RoMoCo 2013*, July 2013.
- [30] R. ALIBERTI, E DI GIAMPAOLO, AND G. MARROCCO. **A model to estimate the RFID read-region in real environments.** In *Wireless Technology, 2008*, pages 1711–1714, Amsterdam, The Netherlands, 2008. IEEE.
- [31] JAE-RYONG CHA AND JAE-HYUN KIM. **Novel anti-collision algorithms for fast object identification in RFID system.** In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, **2**, pages 63–67. IEEE, 2005.
- [32] LEIAN LIU, ZHENHUA XIE, JINGTIAN XI, AND SHENGLI LAI. **An improved anti-collision algorithm in RFID system.** In *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*, pages 5–pp. IEEE, 2005.
- [33] E. MENEGATTI, A. ZANELLA, S. ZILLI, F. ZORZI, AND E. PAGELLO. **Range-only SLAM with a mobile robot and a Wireless Sensor Networks.** In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 8–14, may 2009.
- [34] ROLAND SIEGWART, ILLAH R NOURBAKSH, AND DAVIDE SCARAMUZZA. *Introduction to Autonomous Mobile Robots, second edition*. The MIT Press, 2011.
- [35] A. MARTINELLI, N. TOMATIS, A. TAPUS, AND R. SIEGWART. **Simultaneous localization and odometry calibration for mobile robot.** In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, **2**, pages 1499–1504 vol.2, oct. 2003.
- [36] GREG WELCH AND GARY BISHOP. **An Introduction to the Kalman Filter.** In *Practice*, pages 1–16, 2006.
- [37] STEVEN M LAVALLE. *Planning Algorithms*. 2006.
- [38] MAXIM LIKHACHEV, DAVID I FERGUSON, GEOFFREY J GORDON, ANTHONY STENTZ, AND SEBASTIAN THRUN. **Anytime Dynamic A*: An Anytime, Replanning Algorithm.** In *ICAPS*, pages 262–271, 2005.

- [39] C. DAS, A. BECKER, AND T. BRETLE. **Probably approximately correct coverage for robots with uncertainty.** In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1160–1166, sept. 2011.
- [40] BING-MIN SHIU AND CHUN-LIANG LIN. **Design of an autonomous lawn mower with optimal route planning.** In *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, pages 1–6, april 2008.
- [41] EUROPEAN REGULATION. **EN 60335-2-107.** *European Regulation*, 2010.
- [42] P. R. WURMAN, R. D’ANDREA, AND M. MOUNTZ. **Coordinating hundreds of co-operative, autonomous vehicles in warehouses.** In *IAAI’07 Proceedings of the 19th national conference on Innovative applications of artificial intelligence*, **2**, 2007.
- [43] R. OLMÍ, C. SECCHI, AND C. FANTUZZI. **Coordination of multiple AGVs in an industrial application.** In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1916–1921, may 2008.
- [44] D. RONZONI, R. OLMÍ, C. SECCHI, AND C. FANTUZZI. **AGV global localization using indistinguishable artificial landmarks.** In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 287–292, may 2011.
- [45] S. F. HERNANDEZ–ALAMILLA AND E. F. MORALES. **Global Localization of Mobile Robots for Indoor Environments Using Natural Landmarks.** In *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, pages 1–6, dec. 2006.
- [46] G. SPAMPINATO, J. LIDHOLM, L. ASPLUND, AND F. EKSTRAND. **Stereo vision based navigation for automated vehicles in industry.** In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, sept. 2009.
- [47] O. FAUGERAS. *Three-Dimensional Computer Vision*. The MIT Press, 1993.
- [48] K. LAI, L. BO, X. REN, AND D. FOX. **A large-scale hierarchical multi-view RGB-D object dataset.** In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824, may 2011.
- [49] C. S. PARK S. W. KIM, D. KIM, AND S. R. OH. **Comparison of plane extraction performance using laser scanner and Kinect.** In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 153–155, nov. 2011.
- [50] G. N. DESOUZA AND A. C. KAK. **Vision for mobile robot navigation: a survey.** *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **24**(2):237–267, feb 2002.
- [51] S. TAKEZAWA, T. ISHIMOTO, AND G. DISSANAYAKE. **Optimal Control for Simultaneous Localisation and Mapping Problems in Indoor Environments with Stereo Vision.** In *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pages 4749–4754, nov. 2006.
- [52] S. NUSKE, J. ROBERTS, AND G. WYETH. **Visual localisation in outdoor industrial building environments.** In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 544–550, may 2008.

-
- [53] L. PIYATHILAKA AND R. MUNASINGHE. **Vision-only outdoor localization of two-wheel tractor for autonomous operation in agricultural fields.** In *Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on*, pages 358–363, aug. 2011.
- [54] M. FIALA AND A. UFKES. **Visual Odometry Using 3-Dimensional Video Input.** In *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pages 86–93, may 2011.
- [55] L. XIA, C. C. CHEN, AND J. K. AGGARWAL. **Human detection using depth information by Kinect.** In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15–22, june 2011.
- [56] K. LAI, L. BO, X. REN, AND D. FOX. **Sparse distance learning for object recognition combining RGB and depth information.** In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4007–4013, may 2011.
- [57] P. BENAVIDEZ AND M. JAMSHIDI. **Mobile robot navigation and target tracking system.** In *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pages 299–304, june 2011.
- [58] N. GANGANATH AND H. LEUNG. **Mobile robot localization using odometry and kinect sensor.** In *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*, pages 91–94, jan. 2012.
- [59] E. E. STONE AND M. SKUBIC. **Passive in-home measurement of stride-to-stride gait variability comparing vision and Kinect sensing.** In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 6491–6494, 2011.
- [60] X. ZHANG AND N. NAVAB. **Tracking and pose estimation for computer assisted localization in industrial environments.** In *Applications of Computer Vision, 2000, Fifth IEEE Workshop on.*, pages 214–221, 2000.
- [61] **Microsoft Kinect.** <http://www.xbox.com/en-US/Kinect>.
- [62] **PointGrey Bumblebee2.** <http://www.ptgrey.com/products/bumblebee2/>.
- [63] A. J. BRIGGS, D. SCHARSTEIN, D. BRAZIUNAS, C. DIMA, AND P. WALL. **Mobile robot navigation using self-similar landmarks.** In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, **2**, pages 1428–1434 vol.2, 2000.
- [64] G. ORIOLO, A. DE LUCA, AND M. VENDITTELLI. **WMR control via dynamic feedback linearization: design, implementation, and experimental validation.** *Control Systems Technology, IEEE Transactions on*, **10**(6):835 – 852, nov 2002.
- [65] M. QUIGLEY, K. CONLEY, B. P. GERKEY, J. FAUST, T. FOOTE, J. LEIBS, R. WHEELER, AND A. Y. NG. **ROS: an open-source Robot Operating System.** In *ICRA Workshop on Open Source Software*, 2009.
- [66] **ROS: Robot Operating System.** <http://www.ros.org/>.
-

- [67] GREGORY L PLETT. **Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 2. Modeling and identification.** *Journal of power sources*, **134**(2):262–276, 2004.
- [68] SHUO PANG, JAY FARRELL, JIE DU, AND MATTHEW BARTH. **Battery state-of-charge estimation.** In *American Control Conference, 2001. Proceedings of the 2001*, **2**, pages 1644–1649. IEEE, 2001.
- [69] R. KAUSHIK AND I.G. MAWSTON. **Discharge characterisation of lead/acid batteries.** *Journal of Power Sources*, **28**(1-2):161–169, 1989.
- [70] SABINE PILLER, MARION PERRIN, AND ANDREAS JOSSEN. **Methods for state-of-charge determination and their applications.** *Journal of power sources*, **96**(1):113–120, 2001.
- [71] JAMES H AYLOR, ALFRED THIEME, AND BW JOHNSTON. **A battery state-of-charge indicator for electric wheelchairs.** *Industrial Electronics, IEEE Transactions on*, **39**(5):398–409, 1992.
- [72] JIUN-FU CHEN AND CHIEH-CHIH WANG. **Simultaneous localization and mapping using a short-range passive RFID reader with sparse tags in large environments.** In *Advanced Robotics and its Social Impacts (ARSO), 2010 IEEE Workshop on*, pages 136–141, oct. 2010.
- [73] CLARENCE M SHEPHERD. **Design of primary and secondary cells II. An equation describing battery discharge.** *Journal of the electrochemical society*, **112**(7):657–664, 1965.
- [74] ALA AL-HAJ HUSSEIN AND ISSA BATARSEH. **An overview of generic battery models.** In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–6. IEEE, 2011.
- [75] RUSSELL JAMES PFISTER. **Electrochemical state models of lead-acid batteries.** Technical report, Massachusetts Inst. of Tech., Cambridge (USA), 1979.
- [76] D BERNDT. **Electrochemical energy storage.** *Battery Technology Handbook*, pages 1–100, 2003.
- [77] GREGORY L PLETT. **Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 1. Background.** *Journal of Power sources*, **134**(2):252–261, 2004.
- [78] <http://www.wikipedia.org/>.
- [79] YI MA, STEFANO SOATTO, JANA KOSECKA, AND S. SHANKAR SASTRY. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.
- [80] GARY BRADSKI AND ADRIAN KAEHLER. *Learning OpenCV: Computer vision with the OpenCV library*. O’reilly, 2008.
- [81] GR BRADSKI. **The OpenCV Library (2000).** *Dr Dobb’s Journal of Software Tools*.
- [82] **Open Source Computer Vision Library - OpenCV.** <http://opencv.org/>.

- [83] DANIEL LÉLIS BAGGIO, SHERVIN EMAMI, DAVID MILLÁN ESCRIVÁ, KHVEDCHENIA IEV-GEN, NAUREEN MAHMOOD, JASON SARAGIH, AND ROY SHILKROT. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Pub., 2012.
- [84] SEBASTIAN THRUN. **Google’s driverless car**, 2012.
- [85] M. BERTOZZI, A. BROGGI, E. CARDARELLI, R.I. FEDRIGA, L. MAZZEI, AND P.P. PORTA. **VIAC Expedition Toward Autonomous Mobility [From the Field]**. *Robotics Automation Magazine, IEEE*, **18**(3):120–124, 2011.