Barcode-based Localization of Low Capability Mobile Robots in Structured Environments

Duarte da Cruz Baptista Dias

Dissertation submitted for obtaining the degree of Master in Electrical and Computer Engineering

Juri Committee

President: Doutor Carlos Filipe Gomes Bispo
Advisor: Doutor Rodrigo Martins de Matos Ventura
Co-Advisor: Doutor Porfírio Silva
Vowel: Doctor Anders Lyhne Christensen

June 2012
I would like to thank my supervisors, Professor Rodrigo Ventura and Doctor Porfírio Silva, for their guidance and valuable counsel throughout this work.

I would like as well to give my thanks to Professors João Pedro Gomes and Pedro Lima for their assistance and access to the facilities used in this work.

I would also like to express special thanks to my lab partners, namely: João Carvalho, Pedro Vieira and Miguel Vaz, for their support, stimulating discussions and delicious cookies.

I would like to leave a kind word to all the students under Professor Rodrigo Ventura supervision for their energy and group cohesion which kept me motivated to move forward.

I would like to stress the importance of my friends whose presence and support kept me fresh and flexible, making me able to perform this work the most aptly possible.

Finally I would like to thank my brother for his discussions and valuable suggestions, and to my mother for her priceless support without which the execution of this work would not be possible.
Abstract

Research on multi-robot systems often demands the use of a large population of small, cheap, and low capability robots. Many applications require localization and navigation capabilities, since they involve the movement of individuals or the transport of objects through specific goal points. However, localization methods are computationally expensive, which makes it difficult for the considered robots to support those methods.

This thesis addresses the problem of absolute localization of such low capability robots using onboard sensors and local computation. An onboard B&W camera is used to detect and decode artificial visual landmarks deployed along the environment. These landmarks comprise a dual-layer barcode encoding the landmark pose with respect to a global coordinate frame, dispensing the need for an onboard landmark map. Only part of the camera image is used, saving computational resources. A theoretical framework which provides bounds to the possible barcode observation areas, is developed, allowing the choice of the barcode and camera parameters along with the best barcode poses in the environment which satisfy the barcode visualization requirements. The localization estimations are obtained using an Extended Kalman Filter (EKF) fusing odometry readings with absolute pose estimates obtained from the camera.

The method is implemented on an e-puck robot with 8KB of RAM and a 16MIPS processor. The barcode detector and the developed theoretical framework performances are evaluated through static barcode visualization experiments of several camera poses relative to the barcode. The localization method performance is evaluated comparing the location estimates with both ground truth and odometry.

Keywords: Absolute Localization, Low Capability Robots, Barcode Detection, Homography, Extended Kalman Filter
Resumo

O dimensionamento de Sistemas multi-robôs requer normalmente a utilização de um elevado número de robôs pequenos, baratos e de fraca capacidade. Muitas aplicações exigem capacidades de localização e de navegação, pois envolvem o movimento de indivíduos ou o transporte de objectos para pontos específicos. No entanto, os métodos de localização são computacionalmente caros, o que dificulta a sua implementação nesses robôs.

Esta tese aborda o problema de localização absoluta para robôs de fraca capacidade, utilizando os sensores a bordo e capacidades computacionais próprias. Uma câmara monocromática é utilizada para detectar e descodificar marcadores artificiais no ambiente. Estes consistem em códigos de barra de duas camadas que codificam, no seu código interno, a pose do marcador respectivamente a um referencial global, dispensando a necessidade de um mapa de marcadores. Só parte da imagem da câmera é utilizada, poupando recursos computacionais. Um conjunto de ferramentas que fornecem limites à área de detecção do código de barras, são desenvolvidas, permitindo a escolha dos parâmetros do código de barras e da câmera bem como as melhores posições dos códigos de barras no ambiente que satisfaçam os requisitos de visualização. As estimativas de posição são obtidas utilizando um filtro de Kalman estendido (EKF) juntando leituras de odometria com estimação de pose absoluta extraídas da informação da câmera.

O método é implementado num robô e-puck com 8KB de RAM e um processador de 16MIPS. O desempenho do detector de códigos de barras e as ferramentas teóricas são avaliados através de experiências de visualização de códigos de barras em várias posições da câmera relativas ao código de barras. O método de localização é avaliado comparando as estimativas de localização com os resultados de odometria e de ground truth.

Palavras-Chave: Localização Absoluta, Robôs de Fracas Capacidades, Detecção de Códigos de Barra, Homografia, Filtro de Kalman Estendido
Contents

Abstract v

Resumo vii

Contents xi

List of Figures xiv

List of Abbreviations xv

1 Introduction 1

1.1 Approach ................................................................. 2

1.2 Literature Review .................................................... 3

1.2.1 Localization Systems ............................................ 3

1.2.2 Landmarks ............................................................ 4

1.2.3 Related Work on the E-puck Localization .................... 6

1.2.4 Barcode Detection .................................................. 6

1.3 Structure ................................................................. 8

2 Barcode Detector 9

2.1 Geometric Formulation ............................................... 9
# List of Figures

1.1 The E-puck features ................................................................. 2

1.2 The approach ................................................................. 3

1.3 Localization system architectures ........................................ 5

2.1 Barcode description ................................................................. 10

2.2 The pinhole camera model ....................................................... 11

2.3 Detection problem ................................................................. 12

2.4 Top view of the detection problem ........................................ 13

2.5 barcode observation boundaries ............................................. 16

2.6 Observation distances ............................................................ 17

2.7 Barcode encoding methodologies ............................................. 19

2.8 Double layer barcodes ............................................................ 20

2.9 Influence of the barcode height on the detection range ................. 20

2.10 Barcode detection process ...................................................... 23

2.11 Signal binarization process ..................................................... 24

2.12 Guard set extraction ............................................................ 26

2.13 Homography transformation .................................................... 28

2.14 Embedded code extraction grid ............................................. 30
LIST OF FIGURES

3.1 Predict and update steps .................................................. 35
3.2 Motion model ................................................................. 37
3.3 Camera to robot transformation ........................................... 38
3.4 Observation model ......................................................... 40
3.5 Localization system time diagram ...................................... 42
4.1 Barcode detector experimental setup .................................... 46
4.2 Frontal barcode detection rate ............................................ 47
4.3 Barcode detection rate with orientation ................................. 48
4.4 Camera relative pose extraction ......................................... 49
4.5 Localization scenario ...................................................... 50
4.6 Ground Truth Distribution ................................................. 52
4.7 Path estimation results: single lap ....................................... 53
4.8 Path estimation results: several laps ................................... 53
4.9 Estimation error quantification ........................................... 54
4.10 Recovery from a kidnapping situation ................................. 56
List of Abbreviations

B&W  Black and White
CRC  Cyclic Redundancy Check
EKF  Extended Kalman Filter
FOV  Field of View
KF   Kalman Filter
MCL  Monte Carlo localization
MIPS Millions of Instructions per Second
NRZ  Non-Return to Zero
RAM  Random Access Memory
ROI  Region of Interest
Chapter 1

Introduction

There are several multi-robot systems which make use of numerous relatively simple robots for potential group-level benefits including scalability, flexibility, and robustness to individual failures. A swarm robotic system is a good example of such a system [1, 2]. Since the number of robots in these systems tends to be large, the cost necessary for each robot should be sufficiently small to allow the system’s feasibility. Consequently, each robot contains low sensing, energy and computational resources, making it a low capability robot.

The implementation of these multi-robot systems is usually focused on the emergent collective behavior of the system, rather than the behavior of each individual robot. In particular, Swarm Robotics directs its attention towards modeling the effect of the robots’s local interactions on the global system’s behavior, often inspired by system-level functioning of social insects [2]. This fact causes multi-robot systems to cast aside localization and navigation problems necessary for robots to be capable of performing precise movements in the environment. In fact, only recently has this topic been addressed to multi-robots systems [3]. However, most practical, task oriented, applications require such capabilities, since they involve the movement of robots or the transport of objects through specific goal points.

Localization methods are usually computationally expensive in terms of space and time requirements [4]. Since the considered robots have low computational capabilities, they cannot support the methods required for the implementation of such methods. This thesis intends to approach this problem and to provide the possibility for multi-robot systems composed of large amounts of robots to be applied in new practical applications, by proposing an onboard localization system for low capably robots.

The robot targeted by this work is the E-puck educational robot from the EPFL university. The extremely low computational and memory capabilities, specified in Figure 1.1 and the large userbase in the referred multirobot systems, make this robot ideal to test the proposed localization system.
CHAPTER 1. INTRODUCTION

More information about the E-puck features and architecture can be found in [5].

![The E-puck’s sensor and computational capabilities. Adapted from [5]](image)

<table>
<thead>
<tr>
<th>CPU</th>
<th>64 MHz (16 MIPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>8KB</td>
</tr>
<tr>
<td>Flash</td>
<td>164KB</td>
</tr>
<tr>
<td>Camera</td>
<td>480x640 pixels</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>1772KB/s to 11520KB/s</td>
</tr>
<tr>
<td>IR</td>
<td>~4cm Maximum Range</td>
</tr>
</tbody>
</table>

Figure 1.1: The E-puck’s sensor and computational capabilities. Adapted from [5]

1.1 Approach

The developed system assumes structured environments based on corridors. A camera sensor is inserted in front of a differential wheel robot, at a certain height and pointing in a direction parallel to the ground. Barcodes are deployed on the corridor walls, perpendicular to the ground, with their vertical center aligned with the camera height, being used as artificial landmarks. Their unique embedded code allows for automatic landmark discrimination and landmark environment position storage, dispensing the use of an onboard landmark map, thus avoiding the memory and computational efforts that would be needed to process it. Also, their vertical redundancy allows for barcode detection using only a few lines of the camera sensor, eliminating most of the computational effort needed for image processing. Finally, their fixed stripe number and length properties allows the extraction of the robot position relative to the landmark. Figure 1.2 describes the presented layout.

The localization system is implemented by combining a dead reckoning based localization system with an absolute localization system based on barcode detection. The dead reckoning system directly uses the number of steps issued to the robot wheels to predict robot movement. The absolute localization system is implemented by a barcode detector, which performs landmark localization in the environment, using the barcode’s embedded code, and robot position relative to the landmark, provided the above described barcode morphology information, every time a barcode is visualized by the camera. Most of the existent methods perform the detection using the whole camera image, demanding computational capabilities which are too high for the considered robot types. Also, the methods which perform barcode detection with single scan lines do not account for perspective deformities and thus, do not allow for the extraction of the robot position relative to the barcode. This demands the development, in this work, of a light barcode detector able to account for those deformities.
1.2. LITERATURE REVIEW

1.2.1 Localization Systems

Localization (or position detection) systems can be classified according to several points of view: 1) type of environment, which can be indoors or outdoors; 2) type of data provided, which can be absolute position in the environment or relative position to some point in the environment [6]; 3) type of architecture used, which can be centralized, decentralized, or distributed. Centralized or monolithical architectures take place in just one single robot. Decentralized systems are composed of several separate parts. Distributed or cooperative systems use several independent robots, which cooperate with each other to estimate their positions. The three types of architectures are illustrated in Figure 1.3 to allow a better understanding of their differences. Nowadays, probabilistic methods (Kalman filtering (KF), topological and grid-based Markov localization and Monte Carlo localization (MCL)) are used to perform accurate localization taking into account the sensor noise [6]. Sensor fusion methods merge information gathered from two or more sensors increasing the localization accuracy and adding robustness to single sensor failures [7].

Relative localization systems provide data associated to incremental steps, that refer to a start reference point, and estimate the position with a dead reckoning process, which consists in the data integration over the steps. For example, odometry uses wheel encoders to count the angular displacement of each wheel and obtain, from that, the distance traveled by the robot and its heading direction. Another example is the inertial navigation techniques, that use gyroscopes and accelerometers to measure the
rate of rotation and acceleration of the robot\cite{5}. These systems are the most widely used because of their simplicity, high data output rate and low cost. However, slippage problems and the inaccuracies of the used cinematic models can lead to errors that grow with each new step\cite{6}. Absolute localization systems provide data that refer to the robot position in the environment. The quality of each position estimation is only dependent on the noise of the sensor that was used to extract the data and the inaccuracies of the models used to relate the data to the environment, which means that the system performance errors are independent from the current incremental step. Therefore, many approaches use dead reckoning systems together with other absolute localization systems, normally at a lower updating rate, in order to correct the cumulative errors.

Localization systems that are implemented in a monolithical architecture are more robust in the sense that their behavior and accuracy can be better predicted since they only depend on a single robot. This allows them to be easier to tune for the desired standards. Low computational capabilities or limited sensing capabilities, given the structure of the environment, often motivates decentralized architectures. For example in\cite{9}, the mobile device is sensorless, to allow its simplicity, a fixed camera collects information of the device’s position in the environment, sending it to a control station that computes the device’s position and gives navigation commands to it, accordingly. However, this type of architecture introduces communication overhead between the independent components, causing delay in the system that can be critical for the navigation decisions. The above architectures relate to single robots, and, if applied to multi-robot systems, they cannot use the position estimates of the several robots to improve the overall results. To take advantage of this fact, one can use distributed systems, as in\cite{10}, where each robot communicates position information with its neighbors and iteratively computes its own position. However, the inherent communication overhead can cause convergence and scalability problems and the system complexity makes it hard to predict and to adapt to the desired accuracies.

1.2.2 Landmarks

Landmarks are used in absolute localization systems to allow the association of the robots with a specific environment position. Landmarks can consist of elements completely independent from the robot that is trying to localize itself, or, in the case of distributed systems, can also include the other independent robots of the system. Landmarks can be classified according to: 1) their role in the localization process, which can be active or passive; 2) the way they are inserted in the environment, which can be by natural existence in the environment or by artificial insertion\cite{8}.

Active landmarks actively send out location information and, without performing an active search in the environment, the robot senses the signals sent out by the landmark to determine its position. In
1.2. LITERATURE REVIEW

Figure 1.3: Example of three localization systems using different architectures: a) Monolithic [6]; b) Decentralized [9]; c) Distributed.

[6] A mobile robot uses a circular photovoltaic cell array to capture the signal of a spinning beacon landmark, and computes the distance and orientation to it by measuring the time the beacon needs to pass through the array and the correspondent incidence zone in the array. Passive landmarks do not actively transmit signals and the robot has to actively look for these landmarks to acquire position measurements. For example, in [11] a mobile robot, equipped with a camera looking at the ceiling, detects the corridor lamps in the image and uses that information to maintain its position in the middle of the corridor. Active landmarks remove computational problems inherent to landmark detection but due to their active nature, they can be expensive to construct and maintain [8].

Artificial landmarks are specifically designed and placed in the environment for localization purpose only, which means that, with these landmarks, the robot that is performing localization benefits from advantage, such as localization accuracy and computational simplicity. In the decentralized system presented in [9], two colored emitters are inserted in the mobile device. The emitters are colored for better detection in the image taken by the fixed external camera, and there are two of them to allow for geometric disambiguation of the device’s location. Natural landmarks are already part of the target environment. In [11], the corridor lamps are good examples of these type of landmarks. Since one cannot design these landmarks, the computational complexity for recognizing and processing
CHAPTER 1. INTRODUCTION

them is higher and the reliability lower. In the former example, the corridor lamps are all equal which means the robot needs to perform additional computations to estimate its position along the corridor. However, artificial landmarks require that the environment has to be engineered \cite{8}.

1.2.3 Related Work on the E-puck Localization

Some localization systems are already developed for the e-puck robot. In \cite{12}, a distributed localization system is theorized, in which the e-pucks use their microphones and speaking devices to compute the relative distance and bearing from each other. The problem with this implementation, adding also the already discussed problems of distributed systems, is that one of the e-pucks must be able to know its absolute position in the environment. Also in \cite{12} a decentralized system is implemented, using an external mobile component that performs monocular SLAM with a webcam to compute its environment position and, along with a second webcam, uses stereo vision to detect and compute the environment position of each e-puck. SLAM methods together with stereo vision require the use of components with stronger sensing and computational capabilities which makes them hard to be deployed under low-cost conditions. Also, the fact that the system is decentralized means that there is a possibility of communication overhead that can cause a delay on the system that can be critical for navigation decisions, as stated above.

In \cite{13}, another decentralized system is proposed, in which an external component receives the sensor data from the e-puck camera and performs landmark detection and localization processing with particle filters. This system uses colored landmarks and images which allows for landmark discrimination, but demands more computational and memory resources to process and store colored images. In \cite{14}, it is proposed a monolithical system, also based on particle filters for localization, that uses gray images to extract bearing information from black landmarks in the scenario. This system is lighter in terms of landmark detection but it allows only for relative positioning of the robot to the landmarks, requiring extra model matching techniques \cite{8} to obtain the absolute position of the robot.

The last two systems require an onboard map for the landmark positions in the environment, which makes them limited in terms of landmark number scalability and, consequently, in terms of the environment dimensions. For this case, this factor is of extreme importance given the high memory restrictions inflicted in the problem.

1.2.4 Barcode Detection

Several algorithms that can perform barcode detection already exist for simple portable 2D cameras. The detection process can be divided into three parts: 1) barcode localization in the image, which
1.2. LITERATURE REVIEW

is based on the barcode morphology (parallel black and white bars), 2) obtaining barcode geometric parameters such as the lengths of projected stripes in the camera, and geometric distortions induced by the perspective transformation inherent of the observation point, 3) decoding the embedded code, using the computed geometric parameters to obtain the barcode stripes, and using the barcode encoding to decode them.

For barcode localization, a large variety of methods exists, but in general, all of them have the objective of extracting the regions in the image, where it is most likely the barcode to exist, called regions of interest (ROI). The most common techniques use gradient methods [15, 16, 17, 18] and wavelet transforms [19] to find regions with high unidirectional derivative density caused by the barcode stripes. Other methods also use binarization methods to find dense black and white regions [20]. All of these methods result in small disconnected regions. For ROI extraction the most common methods apply region connecting methods, based on gradient direction similarities [16, 17] or binary region connection [15, 20], consisting in finding the edges of the ROI while connecting the inner regions. Morphological processes can be used to perform binary region connection [18, 19], automatically connecting the regions. However, those processes add greater computational complexity. Black and white bar classification is also performed in this stage, using the results of the applied binarization methods.

For extracting geometric parameters from the obtained ROIs, two types of methods are most common. The first method consists on computing the widths of black and white stripes, by taking a sample scan line from the ROI [16, 19, 20]. This method relies on the fact that the barcode stripes have their width multiple to the width of the smallest stripe. If the barcode orientation was not taken with the ROI extraction, it can be also computed in this stage by comparing the stripe positions in several horizontal scan lines [21]. The second method consists in performing a perspective transformation estimation [15, 17], using the ROI edges, which intrinsically contains the information about the stripe lengths and the geometric distortions of the projected stripes in the images, which can be considered while decoding.

The decoding process is performed also on sampled scan lines. The applied decoding methodology varies in respect to the considered type of barcode encoding. It is also important to stress that, for both stripe width extraction and decoding, the scan lines are taken in the direction according to the barcode’s computed orientation. For some applications [19, 20], where the barcode is considered with their stripes disposed vertically on the image, horizontal scan lines are considered.

The methods which use the whole image to scan for barcodes, require high memory and computational resources, since they need to store and to process all the image. Also, the methods that perform barcode detection with single scan lines do not account for geometric distortions and thus, do not allow for the extraction of the robot pose relative to the barcode.
1.3 Structure

This dissertation is organized as follows: In Chapter 2 the barcode detection problem is described, studying how the barcode and camera parameters affect the observation area of a barcode. A simple barcode design method is presented, derived from the conducted study, generating barcodes with a desired observation area. Also, the methods for barcode decoding are discussed. In Chapter 3 the localization system is presented, along with the motion and observation models used and the methodology used to implement and combine them. In Chapter 4 the implemented barcode detector and the localization system are evaluated, by comparing the results of those methods with ground truth. Finally Chapter 5 concludes the dissertation and discusses the future work.
Chapter 2

Barcode Detector

In this chapter the barcode detection problem is addressed by developing a barcode detection algorithm and studying the theoretical limitations associated to that detection. To obtain the theoretical limitations, a mathematical framework is developed by geometrically formulating the detection problem in function of the camera and barcode parameters. The barcode detection algorithm is developed by addressing problems such as the barcode localization in the image, the extraction of the embedded code, and the extraction of the relative pose of the camera to the barcode. The design process applied for the barcodes used in this work is also presented.

2.1 Geometric Formulation

2.1.1 Barcode Model

In this work, the considered barcode is presented in Figure 2.1. The direction orthogonal to the stripes is the horizontal direction. A barcode stripe is a fixed size uniform color rectangular region. Each barcode is composed of a fixed amount of stripes, $N_{\text{stripes}}$, of width and height respectively, $X_{\text{dim}}$ and $h$. The notation $X_{\text{dim}}$ is used in accordance to the barcode terminology, which specifies the width of the stripe as the X-dimension. Each stripe can be either black, referred as bar or as binary ‘1’, or white, referred as space or as a binary ‘0’.

Three major elements are present in the barcode: the embedded code, the guard sets and the quiet zones. Figure 2.1 associates these elements to their correspondent locations in the barcode. The embedded code is the inner stripe sequence of the barcode, which is unique for each barcode and is capable of storing simple information within it. Let the number of stripes in the embedded code be
denoted $N_{\text{inf}}$. The guard sets are sequences of bar-space pairs, present on each barcode edge, going in the direction of the corresponding edge to the barcode center, with a function of helping barcode detectors to obtain the barcode’s edges and $X_{\text{dim}}$. Several pairs are used to increase the uniqueness probability of the guard sequence and also to increase accuracy of $X_{\text{dim}}$ estimation in the presence of sensor noise. Let the number of pairs be denoted $N_{\text{guard}}$. The quiet zones are set of spaces placed around the barcode to prevent the detectors from confusing the barcode’s signal from the rest of the irrelevant signal.

The total number of barcode stripes is the sum of the stripes of the embedded code with the number of stripes of both the guard sets, which means $N_{\text{stripes}} = N_{\text{inf}} + 4N_{\text{guard}}$. The barcode width, $w$, is the combined width of all the barcode stripes, which means $w = N_{\text{stripes}}X_{\text{dim}}$. The barcode height is the same as the single stripe height, $h$. In this work, a 3D frame is used for the barcode to define its position and orientation in the environment. The center of the frame corresponds to the vertical and horizontal center of the barcode. The z direction is the same as the one in which the stripe height is defined, facing up. The x direction is orthogonal to the barcode surface and its faced to the side from where the barcode can be seen. The y direction defined as the cross product between x and z, in order to obtain an orthogonal, right-hand frame. Figure 2.1 also represents the position of the described barcode frame.

![Barcode elements, frame and geometric parameters.](image)

**Figure 2.1:** Barcode elements, frame and geometric parameters.

### 2.1.2 Camera Model

A single camera is used to visualize the barcodes. There are several methods to model the camera but, for the sake of simplicity, this work uses the simplest model - the pinhole model - presented in
2.1. GEOMETRIC FORMULATION

Figure 2.2 and can be found in [22]. In this model the camera is defined by two major elements: the camera plane which consists in the image’s pixel matrix, and the focal point, $O$. The focal point represents the camera’s position in 3D space. The number of horizontal and vertical pixels in the camera plane are nominated respectively, $Q_x$ and $Q_y$. The distance between the camera plane and the focal point is the focal length, $f$. The axis that passes through the focal point and is orthogonal to the camera plane is the optical axis. The intersection with the optical axis and the camera plane is called principal point, $o$. It is usual that $f$ and the coordinates of $o$ are expressed in pixels. A point in 3D space, $P$, is represented in the camera plane by its intersection with the line formed by the focal point and the respective 3D point and its nominated $p$. A 2D frame is defined for the camera plane to identify each pixel coordinate in the image. A 3D frame is defined for the focal point, using the optical axis to specify the camera’s direction, to specify the camera’s position and orientation. In the figure, the axes are switched from their usual positions to gain benefits in the problem formulation, which will become clear further on. The horizontal and vertical observation boundaries of the camera are often represented by observation angles called fields of view (FOV). Since the principal point is normally near the camera plane center, one can approximately define the fields of view as follows:

\[
FOV_x = 2 \arctan \left( \frac{Q_x}{2f} \right) , \quad FOV_y = 2 \arctan \left( \frac{Q_y}{2f} \right) ,
\]

where $FOV_x$ and $FOV_y$ are respectively, the horizontal and FOVs.

\[\text{Figure 2.2: The pinhole camera model.}\]
2.1.3 Detection Problem

Consider a situation in which a camera is pointed to a barcode, as described in Section 1.1 and represented in Figure 2.3. The barcode and the camera heights are named respectively, \( z_{\text{barcode}} \) and \( z_{\text{camera}} \) and initially are considered different from each other since currently this fact does not affect the detection problem. An horizontal line is acquired from the camera plane to scan the barcode. Any horizontal line can be acquired since the projection on the ground for every line, represented in Figure 2.4 is the same. However, the horizontal line which contains the principal point, named here as principal line, is used because it is parallel to the ground and thus always captures the barcode, independently from the camera’s distance to it.

![Detection Problem Diagram](image)

In the ground projection, the \( N_{\text{stripes}} + 1 \) barcode stripe transitions are considered, named from \( b_0 \) to \( b_{N_{\text{stripes}}} \). The projection of \( b_k \) in the camera plane and the respective projection line are named respectively, \( \Delta_k \) and \( t_k \). The 2D position and orientation of the camera, defined by the focal point’s projection in the ground, are respectively, \( (x_{cb}, y_{cb}) \) and \( \theta_{cb} \), and together compose the camera pose with respect to the barcode frame. The camera plane is represented in Figure 2.4 by the red line segment, along with the horizontal coordinate of the principal point, \( o_x \). The optical axis is denominated \( r_o \) and the orthogonal axis is named \( r_p \). The intersection of \( r_o \) with the \( y \) axis of the barcode is named \( B_2 \). The intersection of \( r_p \) with the \( y \) axis of the barcode is named \( B_1 \). The lines \( r_o \) and \( r_p \) can be expressed as:

\[
\begin{align*}
    r_o : & \quad y = B_2 + \tan(\theta)x , \\
    r_p : & \quad y = B_1 - (\tan(\theta))^{-1}x .
\end{align*}
\]  

(2.2)

The angle \( \theta \) defines the slope of the optical axis in the barcode frame. The angle represents the inclination of the camera to the barcode. It is positive when the camera is in the first quadrant of
2.1. GEOMETRIC FORMULATION

Figure 2.4: The Ground projection of the detection problem, and the key parameters.

the barcode frame, null when the camera is pointing frontally to the barcode, and negative when the camera is in the fourth quadrant of the barcode frame. Only these two quadrants are interesting for the detection problem, since the barcode is not turned to the other two quadrants. This makes $\theta$ vary between $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Each stripe transition, $b_k$, can be defined as:

$$b_k = \frac{N_{\text{stripes}} - 2k}{2} X_{\text{dim}}, \quad (2.3)$$

where $k$ is the respective stripe number and $X_{\text{dim}}$ is expressed in meters. Observing the green triangle in Figure 2.4 one can easily deduced that:

$$\frac{o_x - \Delta_k}{f} = \frac{\cos(\theta) b_k - B_2}{\sin(\theta) B_1 - b_k} = \frac{m B_2 - b_k}{B_1 - b_k}, \quad (2.4)$$
where \( \Delta_k, o_x \) and \( f \) are expressed in pixels and \( b_k, B_1 \) and \( B_2 \) in meters. Here \( m \) is the slope of \( r_p \), with a value of \((\tan(\theta))^{-1}\). The length of each projected stripe in the camera can thus be defined as the distance between the projections of two consecutive transitions:

\[
L_k = |\Delta_{k-1} - \Delta_k| = f|m| \left| \frac{B_2 - b_{k-1}}{b_{k-1} - B_1} - \frac{B_2 - b_k}{b_k - B_1} \right| = fX_{\text{dim}}|m| \left| \frac{B_1 - B_2}{(B_1 - b_{k-1})(B_1 - b_k)} \right|. \tag{2.5}
\]

For the possible camera inclinations, the absolute signs can be dropped since the nominator and the denominator are always positive. From (2.5) it can be observed that the projection of any stripe of the barcode in the camera has its length multiple to \( fX_{\text{dim}} \). This factor can be seen has a trade-off between the camera resolution and the size of the barcode stripes. The visualization problem is symmetric with respect to the x-axis of the barcode, which means that it is enough to study the problem for the first quadrant.

## 2.2 Barcode Design

### 2.2.1 Distance Constrains

Since the objective is to observe the barcode, it is crucial to understand the minimum and maximum observation distances and boundaries from where the camera can observe the barcode. The minimum observation boundary which allows barcode detection, for each \( \theta \), is described by the geometrical locus from where at least one barcode edge coincides with the limits of the camera’s \( \text{FOV}_x \), defined by two lines presented in Figure 2.5 with the following expressions:

\[
r_1 : y = b_0 + \tan \left( \theta - \frac{\text{FOV}_x}{2} \right) x, \quad r_2 : y = b_{\text{stripes}} + \tan \left( \theta + \frac{\text{FOV}_x}{2} \right) x. \tag{2.6}
\]

This boundary is implied by the camera’s \( \text{FOV}_x \), the barcode width, \( w \), which is directly related to the stripe width, \( X_{\text{dim}} \), and the number of barcode stripes, \( N_{\text{stripes}} \), and also the inclination angle to the barcode, \( \theta \). The minimum observation distance is here defined as the smallest distance, from the camera to the barcode surface in the y-axis, going in the direction defined by the current inclination, \( \theta \), from where the camera can visualize the barcode, named \( d_{\text{min}} \). It can be computed by obtaining the length of the line segment defined from the intersection point of the y-axis of the barcode with \( r_o \) and the intersection point of \( r_1 \) and \( r_2 \):

\[
d_{\text{min}}(\theta) = \frac{X_{\text{dim}}(4N_{\text{guard}} + N_{\text{inf}})\sec(\theta)}{\tan \left( \theta + \frac{\text{FOV}_x}{2} \right) - \tan \left( \theta - \frac{\text{FOV}_x}{2} \right)}, \tag{2.7}
\]
The maximum observation boundary, for each $\theta$, is described by the geometrical locus from where the camera is observing the barcode with at least one projected stripe with the minimum length that still allows barcode detection, $L_{\text{min}}$. For every $\theta$ in the first quadrant, the smallest projected stripe length is the $N_{\text{stripes}}$th, which means that the boundary can be defined, replacing $k$ with $N_{\text{stripes}}$ in (2.5) and using (2.2) to replace $B_1$ and $B_2$ with the variables $x$, $y$ and $\theta$, as:

$$L_{\text{min}} = \frac{1}{\sin^2(\theta)} \left( y + \frac{x}{\tan(\theta)} + \frac{f X_{\text{dim}}^x}{2} \right) \left( y + \frac{x}{\tan(\theta)} + \frac{N_{\text{stripes}} - 2}{2} \right).$$  \hspace{1cm} (2.8)$$

The latter expression describes an hyperbola expression in the $(x, y)$ space, parametrized by $N_{\text{stripes}}$, the camera resolution, represented by the camera’s focal length, $f$, the stripe width, $X_{\text{dim}}$, the number of stripes, $N_{\text{stripes}}$, and also the inclination angle to the barcode, $\theta$. Only one side of the hyperbola is considered. The hyperbola expression is as follows:

$$r_h : x^2 A_{xx} + y^2 A_{yy} + 2xy A_{xy} + 2xB_x + 2yB_y + C = 0,$$  \hspace{1cm} (2.9)$$

where

$$A_{xx} = 1, \quad A_{yy} = \tan^2(\theta), \quad A_{xy} = \tan(\theta),$$

$$B_x = \frac{X_{\text{dim}}}{2} \left( (N_{\text{stripes}} - 1) \tan(\theta) - \frac{f}{L_{\text{min}} \cos^2(\theta)} \right), \quad B_y = \frac{X_{\text{dim}}}{2} \left( (N_{\text{stripes}} - 1) \tan(\theta) \right),$$

$$C = -\frac{X_{\text{dim}}^2 (N_{\text{stripes}} - 2) N_{\text{stripes}} \tan^2(\theta)}{4}.$$  

The Nyquist theorem for digital signal recovery \[23\] states that the receiver cannot recover signals with frequencies higher than one half of the sampling rate. Since the camera samples the signal with a period of 1 pixel, the lowest period allowed for the signal is of 2 pixels, which means $L_{\text{min}}$ is 1 pixel. The maximum observation distance from a point, $p_{\text{max}}$, in $r_h$, is the distance from that point to the barcode surface in the $y$-axis, in the direction defined by the current inclination, $\theta$, and it can be computed in two steps. The first step consists in obtaining $p_{\text{max}}$’s $x$ coordinate by computing the intersection between $r_h$ and a line, named $r$, defined by $p_{\text{max}}$ and the intersection between $r_1$ and $r_2$. This is done by replacing $y$ in the hyperbola expression, with the $r$’s equation:

$$y = \alpha + \beta x, \quad x^2 A_{xx} + y^2 A_{yy} + 2xy A_{xy} + 2xB_x + 2yB_y + C = 0,$$

$$x^2(A_{xx} + A_{yy} \beta^2 + 2A_{xy} \beta) + 2x(B_x + B_y \beta + A_{yy} \beta \alpha + A_{xy} \alpha) + C + A_{yy} \alpha^2 + 2B_y \alpha = 0,$$  \hspace{1cm} (2.10)$$
CHAPTER 2. BARCODE DETECTOR

Figure 2.5: Minimum (green lines) and maximum (red hyperbola) barcode observation boundaries with their respective critical distances.

where $\alpha$ and $\beta$ are $r$'s parameters. The $x$ coordinate of $p_{\text{max}}$ can be computed, choosing the biggest positive solution of (2.10). The second step consists in computing the distance from $p_{\text{max}}$ and the $y$-axis, in the direction defined by $\theta$:

$$d_{\text{max}}(\theta) = \frac{x_{p_{\text{max}}}(\theta)}{\cos(\theta)}.$$  (2.11)

There are three relevant maximum observation distances, defined when $r$ matches $r_1$, $r_o$ and $r_2$, named respectively, $d_{\text{max}1}$, $d_{\text{max}2}$ and $d_{\text{max}3}$. The distance $d_{\text{max}1}$ is the smallest distance and any camera in a position under this distance and inside the minimum observation boundary always obtain barcode observations compliant with the sampling theorem, which means in ideal circumstances it can detect the barcode. The distance $d_{\text{max}3}$ is the biggest distance which means that any camera in a position above this distance cannot observe the barcode. The distance $d_{\text{max}2}$ is defined in the same axis as $d_{\text{min}}$ allowing an intuitive comparison between the maximum and minimum distances. Figure 2.6
2.2. BARCODE DESIGN

shows the typical behavior of the minimum and maximum observation distances in function of the camera inclination.

Figure 2.6: Relevant distances with camera inclination. Within the green area, all the conditions are gathered for barcode detection inside the camera’s $FOV_x$.

### 2.2.2 Absolute Pose Encoding

As stated in Section 1.1, the barcode is deployed on the environment at a certain 2D position, $(x_w^b, y_w^b)$, and orientation, $\theta_w^b$, jointly referred as the barcode pose. This pose is stored inside the barcode embedded code, composed of $N_{inf}$ stripes. The $x_w^b$, $y_w^b$ and $\theta_w^b$ coordinates are encoded respectively, with $N_x$, $N_y$ and $N_\theta$ stripes. An error detection sequence of $N_{error}$ stripes is added to prevent wrong detections. This means that $N_{inf} = N_x + N_y + N_\theta + N_{error}$. The coordinates $x_w^b$ and $y_w^b$ can vary according to the environment boundaries in each dimension, respectively $[x_{lmin}, x_{lmax}]$ and $[y_{lmin}, y_{lmax}]$. The orientation $\theta_w^b$ can vary between the $[0, 2\pi]$ interval. The pose is encoded in the barcode in three steps. First each coordinate is converted into an approximate binary sequence, $bin_x$, $bin_y$ and $bin_\theta$, using the following process:

$$bin_c = \text{binary} \left( \left\lfloor 2^{n_c} \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \right\rfloor \right), \quad (2.12)$$

where $c$ is the respective coordinate, $c_{\text{min}}$ and $c_{\text{max}}$ the coordinate’s variation boundaries, and $n_c$ the number of bits used for the sequence. The operator binary transforms decimal numbers to binary numbers. The greater $n_c$, the more precise is each coordinate conversion and thus the greater the
resolution of the respective environment dimension. Next, the binary sequences are attached together in the $bin_x-bin_y-bin_\theta$ order into one single binary sequence. In the end of the sequence it is also attached an error correction sequence, $bin_{error}$, obtained by applying the Cyclic Redundancy Check (CRC) algorithm on the merged sequence. The CRC algorithm is not explained here but the reader may refer to [24] for more information.

Finally the resulting binary sequence is encoded to the barcode, using a binary signal encoding method. Three methods were considered: Unipolar Non-Return to Zero (NRZ), Bi-Phase, and Manchester. In this work, the unipolar NRZ will be denoted as just NRZ, to simplify the notation. Each method considers a base encoding period, $T$, during which a bit is encoded. The NRZ method puts the signal to 1 (bar) or 0 (space) in the current period, depending if it is encoding a '1' or '0'. The Bi-Phase method alternates the signal from 1 (bar) to 0 (space) or vice-versa, in the period transitions, and if it is encoding a '1' it also alternates the signal in the middle of the period. The Manchester method always alternates the signal in the middle of the period, from from 1 (bar) to 0 (space) or vice-versa depending if it is encoding, respectively, a '0' (negative transition) or a '1' (positive transition). For more information about these or more encoding methods the reader can refer to [25].

Figure 2.7 shows the encoding results from applying the three methods to a binary sequence. If a bar encodes a '1' and a space a '0', that means the Bi-Phase and Manchester methods would need twice the number of barcode stripes than the NRZ method, which means that with a fix number of barcode stripes, the NRZ method can encode twice the bits than the other two methods, allowing barcode pose storage with bigger precisions. On the other hand, the Bi-Phase and Manchester methods guarantee a transition between every two stripes, which allows signal synchronization when detecting the barcode. The NRZ encoding is the preferred here since the other two are spatially much less efficient: this would double the barcode size for the same amount of encoded bits and the same stripe width.

### 2.2.3 Double Layered Barcodes

In order to encode a more precise pose and to increase the CRC sequence, a second layer is introduced in the barcode, making it a double layer barcode. The length of the binary sequence that can be encoded is now doubled. The higher bits of the sequence are encoded in the upper layer and the lower bits in the lower layer. Each layer is detected separately with a scan line for the respective layer, as illustrated in Figure 2.8.

Although this approach enables a greater precision in the barcode pose insertion and a better error detection, it raises several challenges. Firstly, since there are now two scan lines, at least one cannot correspond to the principal line, which means there is a certain distance from where that scan line will start to fail to capture the barcode, which will preclude the barcode detection. Secondly, since...
2.2. BARCODE DESIGN

Figure 2.7: Barcodes for different encoding methods with the same base encoding period.

there are two layers to scan, it must be guaranteed that each scan line is scanning its respective layer. These two problems require the scan lines to be as close as possible to the principal line, at a height of \( z_{\text{camera}} \), to maximize the distance from where the scan lines start to fail to capture their respective barcode layers. Also, \( z_{\text{barcode}} \) must be the same as \( z_{\text{camera}} \) to maintain the scan lines in their respective layers along all the detection area. The distance of each scan line to the principal line should not be too small to account for uncertainties on \( z_{\text{camera}} \) and \( z_{\text{barcode}} \) and small differences between \( z_{\text{camera}} \) in different robots. For simplicity each scan line is chosen with the same distance from the principal line.

The scan line inclination introduces a new maximum observation distance, \( d'_{\text{max}} \), illustrated in Figure 2.9. Analyzing Figure 2.4, it can easily be shown that every parameter from \( y_b \) axis can be projected to the plane considered in Figure 2.9 by multiplying them by a factor of \( \sin(\theta) \). \( b'_0 \), \( b'_{N_{\text{stripes}}} \) and \( B'_2 \) are respectively, the projections of \( b_0 \), \( b_{N_{\text{stripes}}} \) and \( B_2 \) on that plane. The new maximum observation distance is defined as the distance from where the projection of the scan lines on the most distant barcode horizontal limit touch the correspondent layer’s vertical limit. From Figure 2.9 this distance to the barcode center can be expressed as follows:

\[
    d^*_{\text{max}}(\theta) = d'_{\text{max}} - 0.5\, w \sin(\theta) = h \frac{f}{Q_y} - 0.5\, w \sin(\theta). \tag{2.13}
\]
Figure 2.8: A 3D perspective of a double layer barcode scan, with the chosen horizontal scan lines for each layer (in red) and the respective projections in the barcode plane (also in red).

Figure 2.9: Side view of the detection problem, from a plane orthogonal to the ground, parallel to the camera’s optical axis and passing through the barcodes center.

### 2.2.4 Design Method

In this work the barcodes are designed to comply with barcode pose precision, error detection and guard sequences sizes, and minimum distance visualization specifications. For the first specification the number of stripes encoding each pose coordinate, $N_x$, $N_y$, and $N_\theta$, are computed using the respective specified precisions respectively, $pr_x$, $pr_y$, and $pr_\theta$. The value for a general coordinate, $c$, between $c_{\text{min}}$ and $c_{\text{max}}$, can be encoded into a binary sequence with Eq. (2.12). According to the specified precision for that coordinate, $pr_c$, at least $c + pr_c$ has to be encoded in a new sequence. This means that the respective integers, obtained in Eq. (2.12), for $c$ and $c + pr_c$ have to be different. Therefore, one can write:

\[
\begin{align*}
2^{n_c} \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} - 2^{n_c} \frac{c + pr_c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} & \geq 1 \Leftrightarrow 2^{n_c} \frac{pr_c}{|c_{\text{max}} - c_{\text{min}}|} \geq 1. 
\end{align*}
\]  

(2.14)
2.2. BARCODE DESIGN

Bearing in mind that each bit of the binary sequence is encoded with \( T \) barcode stripes, according to the chosen encoding process (for NRZ, \( T \) is 1), the number of stripes required to encode the general coordinate, \( N_c \), can be obtained from the latter expression, as:

\[
N_c = \text{ceil} \left( \frac{T \log_2 \left( \frac{|c_{max} - c_{min}|}{pr_c} \right)}{2} \right) .
\]  
(2.15)

The operator \textit{ceil} is a round up process, necessary to convert a non integer result to the number of bits to be used without compromising the specified precision. The second and third specifications consist respectively, in directly specify \( N_{error} \) and \( N_{guard} \). The number of stripes can now be computed:

\[
N_{stripes} = 4N_{guard} + N_{inf} = 4N_{guard} + N_x + N_y + N_\theta + N_{error} .
\]  
(2.16)

For double layer barcodes, the sequence elements \( N_x, N_y, N_\theta \) and \( N_{error} \) are expected to be even, to allow an easy division between lower and upper layers. So after their computation, every odd elements are summed with one to become even. The final specification is satisfied by using a barcode stripe length, \( X_{dim} \), that allows the specified minimum observation distance of \( d_{min} \), considering a specified camera’s \( \text{FOV}_x \), for any camera inclination \( \theta \). It can be observed in Figure 2.6 that the null inclination is the most restrictive in terms of limiting the minimum distance, so \( X_{dim} \) is computed for that case, using (2.7), and replacing \( \text{FOV}_x \) with \( f \) and \( Q_x \) according to (2.1):

\[
X_{dim} = d_{min}(0) \frac{Q_x}{f(4N_{guard} + N_{inf})} .
\]  
(2.17)

For double layer barcodes, the height of the barcode is computed to guarantee that the maximum observation distance caused by the scan line inclination is greater than a certain value, \( D \). From (2.13), the barcode height can be expressed as:

\[
h = \frac{Q_y D}{f} + 0.5 w \sin(\theta) \frac{Q_y}{f} ,
\]  
(2.18)

where \( w \) is the barcode width, expressed by \( X_{dim} N_{stripes} \). Since, from (2.13), \( d_{max}^* \) is greater for \( \theta = \frac{\pi}{2} \), \( h \) is computed using the latter expression for that \( \theta \) value.

Also, the barcode can be designed to be centered with the specified camera height, \( z_{camera} \), to ensure that the scan lines hit the barcode. For double layer barcodes this is mandatory. The upper edge of the barcode is placed at the height of \( z_{camera} + 0.5h \) and the lower edge at the height of \( z_{camera} - 0.5h \). For double layer barcodes the upper layer’s lower edge and the lower layer’s upper edge are both placed at the height of \( z_{camera} \), and the upper layer’s upper edge and the lower layer’s lower edge at
a height of $z_{\text{camera}} + 0.5h$ and $z_{\text{camera}} - 0.5h$, respectively. The user is referred to Figure 2.9 for a better understanding of the problem’s geometric characteristics considered to derive the barcode height positioning.

### 2.3 Detection Algorithm

#### 2.3.1 Overview

This section describes the process developed to detect barcodes from single scan line signals. Once the signal is captured, it is converted into grayscale. To allow stripe classification, each pixel of the signal is classified into black and white using a binarization method. To perform barcode localization in the signal, barcode hypotheses are generated from the binarized signal. Sets of contiguous pixels with the same color are packed into regions. Guard sets are identified by groups of contiguous black-white region pairs with low color intensity and relative region length divergences. Each barcode hypothesis is associated with a ROI, which consist in a group of regions confined by guard sets pairs. For each hypothesis, the geometric parameters (which predict the evolution of the stripe lengths, $L_k$, along the barcode) are computed by estimating the perspective transformation that translates the real stripe transition positions, $b_k$, to their respective projected positions, $\Delta_k$. Each obtained transformation is evaluated to check if it corresponds to a valid barcode projection in the camera. The hypotheses with invalid perspective transformations are not considered.

The embedded code sequence is then extracted from the hypotheses, by classifying each stripe of the sequence according to the black and white region pattern in the embedded code’s signal area of the respective hypothesis. The CRC method is applied to the extracted sequences to detect possible invalid sequences. The hypotheses with invalid sequences are cast away. The remaining hypotheses are considered valid barcodes, and the relative pose of the camera to the barcode is computed by decomposing the perspective transformation that was earlier computed for each hypothesis. Figure 2.10 shows the information flow along the described process.

For double layer barcodes, the embedded codes from the upper and lower layers are decoded independently and treated as if they had no error correction bits. The detected sequences from the upper and lower layers which have their edge positions in the respective scan lines not differing more than a few pixels, are merged together to form a whole sequence. Error detection is then applied to the merged sequences to identify possible detection errors.
2.3. DETECTION ALGORITHM

The signal is obtained by acquiring an image in a rectangular zone of the camera with \( Q_x \) pixels width and \( Q_y \) pixels height, containing principal line in the center, as illustrated in Figure 2.9. Figure 2.11 a) shows the results of an acquisition example. The scan lines are chosen according to the barcode model used. For single layer barcodes the principal line, in the center of the image, is chosen. For double layer barcodes the upper line is used to scan the upper layer and the lower line is used to scan the lower layer.

Each scan line consists in an array of pixels, \( u_k \), with \( k \) from 1 to \( Q_x \). Each pixel is converted into grayscale accordingly to their color representation system. The resulting pixels are classified as black or white depending on whether they are respectively, below or above a constant threshold, \( th \). This threshold is computed by averaging the pixels extremes of the signal:

\[
th = \frac{u_{\text{max}} + u_{\text{min}}}{2},
\]  

(2.19)

where \( u_{\text{max}} \) and \( u_{\text{min}} \) are respectively, the maximum and minimum intensities of the signal. This threshold has the advantage of automatically adapting to the current light intensity conditions of the
environment or the different reflection proprieties present on the current used barcodes. Figure 2.11 b) illustrates the signal binarization process with an example.

Figure 2.11: Signal binarization process. a) Example of an image acquisition. b) Threshold computation for one of the scan lines of the image. Pixels with intensities above an below the threshold are classified respectively, as white or black. Pixel with intensities sufficiently close to the threshold are to be treated as noise. In red it is shown the only ROI obtained for this scan line.

2.3.3 Hypotheses Formulation

The next step consists in generating plausible barcode hypothesis from the binarized signal, and is composed of three phases: 1) region analysis, 2) hypothesis generation, 3) hypothesis evaluation. In the first phase, contiguous pixels with the same color are packed into regions, \( r_{gk} \). A barcode stripe, bar or space, will appear in the signal as one region, black or white respectively. Each region has a beginning pixel, \( u_i \), a color intensity, \( c_k \), obtained by averaging the color intensity (on the original signal) of all the pixels of that region, and a length, \( l_k \), consisting in the amount of pixels inside the region. In order to eliminate regions that fall under the situation illustrated in Figure 2.11 b) inside
the orange box, only white and black regions with $c_k$ respectively, above and bellow the binarization threshold by at least $dc_{min}$ percent, are considered. From the considered regions, all the guard sets, $gd_j$, are formed by grouping sets of $2N_{guard} - 1$ adjacent black and white regions, ignoring the guards most inner spaces since they can be mixed with the embedded code. Each guard has a beginning region, $rg_i$, and an ending region, $rg_f$. The differences between the length of the $gd_j$’s regions are measured by a relative standard deviation of the region lengths, computed as follows:

$$dl_j = \sqrt{\frac{\sum_k (l - l_k)^2}{l}}, \quad k \epsilon [i, f],$$

(2.20)

where $l$ is the bar length average of the $gd_j$’s regions. To eliminate guard possibilities formed by regions representing parts of the scenario other than the barcode, which have no region length similarity relations between them, only guards with $dl_j$ below a certain value, $dl_{max}$, are considered. From one can observe that the projected stripes have increasing length divergences with the camera’s inclination $\theta$. Thus $dl_{max}$ imposes a maximum inclination for barcode detection. However, since the guard stripes are sequential, the changes in the lengths of their regions will not be too significant and if $dl_{max}$ is kept substantially high only an inclination near $\frac{\pi}{2}$ would cause the detections to fail, and most of the detection area for these inclinations is already non existent. Figure 2.12 presents the extracted regions and guards for the signal acquired in Figure 2.11.

For the second phase of this step, an hypothesis, $hp_n$, is formed with pairs of the extracted guards. In a guard pair, the guard on the left, $gd_l$, is always considered as the left guard, and the guard on the right, $gd_r$, is always considered as the right guard. The guards are paired up accounting for the minimum number of stripes that the used encoding method requires to have between the two guards. For NRZ encoding, this distance is unitary, since it does not demand any transition inside the encoding sequence. The guards can be exhaustively combined and evaluated. Although this solution is guaranteed to extract all the observed barcodes, it generates large sets of hypothesis to evaluate, which would consume a lot of time. Instead, a region connection method, inspired by [20], is used to collect ROIs. A barcode contained in a ROI has its guards in its edges, making it unnecessary to perform guard combinations. ROIs are extracted from the previously extracted and filtered regions. Figure 2.11 shows the ROI extracted from the acquired signal, using this method.

A brief description of the latter method is now provided. First, the smallest black region, called seed, supposed to be the smallest stripe projection of a barcode, is found. The maximum size of the ROI, $w_{ROI}$, which contains the seed can be computed from the seed’s length, $l_k$, as follows:

$$w_{ROI} = (N_{stripes} + 2l_{cone})l_k + slack,$$

(2.21)
Table 2.6: Results of the barcode detector.

<table>
<thead>
<tr>
<th>k</th>
<th>u_i</th>
<th>class</th>
<th>l_k</th>
<th>c_k</th>
<th></th>
<th>r_gk</th>
<th>r_gj</th>
<th>d_l_{max}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>68</td>
<td>21</td>
<td>5</td>
<td>0.212</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>69</td>
<td>0</td>
<td>22</td>
<td>5</td>
<td>0.251</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>70</td>
<td>1</td>
<td>23</td>
<td>4</td>
<td>0.243</td>
<td>3</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>74</td>
<td>0</td>
<td>24</td>
<td>6</td>
<td>0.263</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>91</td>
<td>1</td>
<td>25</td>
<td>3</td>
<td>0.247</td>
<td>5</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>92</td>
<td>0</td>
<td>26</td>
<td>5</td>
<td>0.278</td>
<td>6</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>106</td>
<td>1</td>
<td>27</td>
<td>3</td>
<td>0.247</td>
<td>7</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>108</td>
<td>0</td>
<td>28</td>
<td>6</td>
<td>0.278</td>
<td>8</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>1</td>
<td>29</td>
<td>4</td>
<td>0.247</td>
<td>9</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>126</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>0.282</td>
<td>10</td>
<td>19</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>156</td>
<td>1</td>
<td>31</td>
<td>4</td>
<td>0.239</td>
<td>11</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>159</td>
<td>0</td>
<td>32</td>
<td>6</td>
<td>0.251</td>
<td>12</td>
<td>23</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>160</td>
<td>1</td>
<td>33</td>
<td>4</td>
<td>0.102</td>
<td>13</td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>179</td>
<td>0</td>
<td>34</td>
<td>7</td>
<td>0.498</td>
<td>14</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>230</td>
<td>1</td>
<td>35</td>
<td>4</td>
<td>0.004</td>
<td>15</td>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td>16</td>
<td>233</td>
<td>0</td>
<td>36</td>
<td>14</td>
<td>0.498</td>
<td>16</td>
<td>31</td>
<td>34</td>
</tr>
<tr>
<td>17</td>
<td>238</td>
<td>1</td>
<td>37</td>
<td>4</td>
<td>0.004</td>
<td>17</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>18</td>
<td>240</td>
<td>0</td>
<td>38</td>
<td>8</td>
<td>0.498</td>
<td>18</td>
<td>35</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>244</td>
<td>1</td>
<td>39</td>
<td>4</td>
<td>0.004</td>
<td>19</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>247</td>
<td>0</td>
<td>40</td>
<td>58</td>
<td>0.498</td>
<td>20</td>
<td>39</td>
<td>42</td>
</tr>
</tbody>
</table>

\[ dc_{\text{min}} = 0.3 \quad \text{and} \quad dl_{\text{max}} = 1.0 \]

Figure 2.12: Guard set extraction. a) Region, \( r_gk \), formation. b) Guard, \( gd_j \), formation. Black regions are classified as '1's (possible bars), and white regions as '0's (possible spaces). The highlighted regions and guards (in orange) are discarded since they fail respectively, in intensity and length similarities, defined by the thresholds \( dc_{\text{min}} \) and \( dl_{\text{max}} \).

where \( q_{\text{zone}} \) is the number of stripes contained in the quiet zones in each side of the barcode and \( \text{slack} \) is a factor that accounts for sensor noise and for projected stripe length divergences in the presence of perspective. The latter parameter requires the barcodes to be a certain distance apart from each other, otherwise, in a frontal case, more than one barcode can be caught inside the same ROI. After \( w_{\text{ROI}} \) is computed, the ROI can be built. The method begins connecting regions, starting from the seed, switching alternately from left to right and right to left, until no more black regions are inside a possible range of \( w_{\text{ROI}} \) pixels. Next the formed ROI is extracted from the signal, and the same process is repeated, without considering the signal regions of the previous obtained ROIs, until no more seeds are available. For more information about region connecting methods the reader is referred to [20].

The third phase of this step consists in associating a perspective transformation, \( H_n \), to each hypothesis, \( hp_n \). This \( H_n \) translates the real points in the barcode surface, \( b \), to the respective projected points in the camera plane, \( \Delta \), as illustrated in Figure 2.13. The transformation can be obtained from the previous geometric formulation of the problem. However, to compute the transformation parameters, \( B_1, B_2 \) and \( m \), it would be necessary to solve a nonlinear mean square system. Also, this transformation
becomes unstable around frontal detections, since $B_1$ goes to infinity for those cases, as it can be concluded in Figure 2.13. Instead the 2D homography concept is used, which provides a linear mean square system with a unique solution which can be computed quickly. In Figure 2.13 $\alpha_1$ and $\beta_1$ are unit vectors, characterizing respectively, the directions of the camera plane and the barcode surface, and $\alpha_0$ and $\beta_0$ are translation vectors of the origins of the respective planes, relative to the camera’s focal point, centered in the camera frame. One can define $b$ and $\Delta$ in the homogeneous space as follows:

$$bh = \begin{bmatrix} b \\ 1 \end{bmatrix}, \quad \Delta h = \begin{bmatrix} \Delta \\ 1 \end{bmatrix}. \quad (2.22)$$

From the camera frame one can conclude the following:

$$A = \begin{bmatrix} \alpha_1 & \alpha_0 \end{bmatrix}, \quad B = \begin{bmatrix} \beta_1 & \beta_0 \end{bmatrix}, \quad \Delta h = \lambda A^{-1} B b^h. \quad (2.23)$$

The 2D homography is defined by a 2x2 Matrix, named $H$, which translates $b^h$ to $\Delta^h$ as follows:

$$\Delta^h = \lambda H b^h, \quad H = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}. \quad (2.24)$$

The transformation parameters are now $\lambda, h_{11}, h_{12}, h_{21},$ and $h_{22}$. The homogeneous expression in (2.24) can be manipulated as follows:

$$\frac{1}{\lambda h_{22}} \begin{bmatrix} \Delta \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{h_{11}}{h_{22}} & \frac{h_{12}}{h_{22}} \\ \frac{h_{21}}{h_{22}} & 1 \end{bmatrix} \begin{bmatrix} b \\ 1 \end{bmatrix}. \quad (2.25)$$

From this expression one can conclude the following:

$$\Delta (h_{21}^* b + 1) = bh_{11}^* + h_{12}^* \quad (2.26)$$

where $h_{11}^*, h_{12}^*$, and $h_{21}^*$ are respectively, the parameters $h_{11}, h_{12}$ and $h_{21}$ divided by $\lambda h_{22}$. The latter expression can be written as follows:

$$\Delta = \begin{bmatrix} b & 1 & -b \Delta \end{bmatrix} \begin{bmatrix} h_{11}^* \\ h_{12}^* \\ h_{21}^* \end{bmatrix}. \quad (2.27)$$
Figure 2.13: Interpretation of the 2D homography concept in the barcode detection problem.

The parameters $h_{11}^*$, $h_{12}^*$ and $h_{21}^*$ can be computed for each hypothesis, using the transitions of the guards which are associated with it. $\Delta_0$ to $\Delta_{2N_{\text{guard}} - 1}$ and $\Delta_{(N_{\text{stripes}} - (2N_{\text{guard}} - 1))}$, obtained from the pixels in the edges of the regions representing the respective stripe of guards $gd_l$ and $gd_r$, are respectively associated with, $b_0$ to $b_{2N_{\text{guard}} - 1}$ and $b_{(N_{\text{stripes}} - (2N_{\text{guard}} - 1))}$ to $b_{N_{\text{stripes}}}$, obtained from (2.3). The resulting linear system of $2(N_{\text{guard}} - 1)$ equations of the type (2.27) can now be solved for $h_{11}^*$, $h_{12}^*$ and $h_{21}^*$ using mean squares. The reader should note that $\lambda h_{22}$ is not important for computing the transformation, acting just as a scaling factor, which is automatically computed when the transformation is applied to each point.

The performance of each computed transformation is then measured by computing the estimation error. The inverse transformation is applied to the projected transitions used in the estimation, and
2.3. DETECTION ALGORITHM

the results are compared with the associated real transitions of the barcode. The squared error is then used as the estimation error:

$$e = \sum_k \left( b'_k - b_k \right)^2$$

where $b'_k$ results from applying the inverse of the estimated transformation to the corresponding point $\Delta_k$. A constant validation threshold is used to chose only the hypotheses with consistent perspective transformation.

2.3.4 Embedded Code Extraction

After plausible hypotheses are extracted, their embedded code sequences are extracted directly from the binarized signal. This sequence is contained inside the signal region between the guards $gd_l$ and $gd_r$ of the respective hypothesis. Since the guards most inner spaces were ignored when obtaining the guard sets, they are also extracted here, along with the embedded code. Thus, the resulting sequence will always have at least one ‘0’ in the beginning and the ending of the extracted sequences, corresponding to the unconsidered guard spaces. These two ‘0’s are to be excluded from the sequence since they do not belong to the embedded code.

The previously computed perspective estimation for each hypothesis is used in the extracting process of the embedded code. The stripes of the embedded code plus the most inner stripes of both barcode guards go from the $2N_{guard} - 1$’th to the $N_{stripes} - (2N_{guard} - 1)$’th stripe. The middle points of these stripes in the barcode surface, $s_k$, are computed from averaging the respective stripe transitions, as follows:

$$s_k = \frac{b_{k-1} + b_k}{2},$$

where $b_{k-1}$ and $b_k$ are the transitions of stripe $k$, defined in (2.3). The predicted projections of these middle points, $M_k$ are computed using (2.26) with the $h_{11}$, $h_{12}$, and $h_{21}$ parameters estimated for the current hypothesis, replacing $b$ with $s_k$ and $\Delta$ with $M_k$. The predicted projections are rounded up in order for them to correspond to a pixel number, and then the corresponding pixel in the binarized signal is checked to extract the value (‘1’ or ‘0’) each stripe encodes. If the pixel is classified as black, it means that the corresponding stripe is black (bar), which means it encodes a ‘1’. If the pixel is classified as white, it means that the corresponding stripe is white (space), which means it encodes a ‘0’. The extracted stripe values are inserted into a sequence with the same order than of the respective
CHAPTER 2. BARCODE DETECTOR

stripes. Figure 2.14 illustrates the predicted projection of the middle points and the respective pixel value, for the hypothesis considered in the ROI presented in Figure 2.11.

Figure 2.14: Predicted projected middle points and transitions of the barcode during the detection process.

Since the perspective estimation is computed with only the guard transitions, its transformation performance deteriorates from the barcode edges to the barcode center. To prevent this problem, the middle points start to be evaluated from the edges to the center of the barcode, and every black to white or white to black transition, that is found along the way, is used to update the perspective estimation. Recursive least squares [26] is used to avoid the recomputation of the whole system, reducing the computational complexity of this extra step. This method should improve substantially the results of the embedded code extraction process for most of the barcodes. However, for barcodes that have transitions only in the center (which is possible with NRZ encoding), this method will not bring much improvement, since the perspective estimation is only updated in the center, when most of the stripe classifications are already done.

After the sequence is formed, the guard inner stripes are excluded and the resulting sequence is decoded using the inverse encoding transformation defined by the used encoding method, in order to obtain the binary sequence of the barcode data. For NRZ encoding the inverse transformation is the same, which means, an encoded ‘0’ means a ‘0’ in the binary sequence and an encoded ‘1’ means a ‘1’ in the binary sequence. Finally, the resulting sequence is scanned for errors using the CRC method. Only
hypothesis with valid embedded codes are considered.

### 2.3.5 Pose Extraction

After the embedded coded sequence validation, the camera pose in the barcode frame, defined by the 2D position and orientation of the camera, respectively \((x^b, y^b)\) and \(\theta^b\), is obtained for each remaining hypothesis. This last step involves the estimated perspective transformation from each hypothesis. The transformation uniquely defines each perspective, since the system allows only one solution provided associations between \(b\) and \(\Delta\). Also, each perspective is unique for each pose of the camera with respect to the barcode frame, since if at least one of the transformation parameters is changed, part of the associations between \(b\) and \(\Delta\) would be modified, which means there are no multiple poses with the same perspective. These two facts show that the perspective transformation completely defines the camera pose in the barcode frame, and consequently, provided its parameters, that pose can be obtained.

In this work, the camera pose is obtained from a 2D homography decomposition. Considering the camera frame, \(\alpha_0, \alpha_1\) in expression (2.23) and consequently \(A\) in (2.24), are defined, observing Figure 2.13 as:

\[
\begin{align*}
\alpha_0 &= \begin{bmatrix} f \\ ox \end{bmatrix}, \\
\alpha_1 &= \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \\
A &= \begin{bmatrix} f & 0 \\ ox & -1 \end{bmatrix}.
\end{align*}
\] (2.30)

Relating \(A^{-1}B\) in (2.23) with \(H\) in (2.24) one can obtain the following:

\[
\begin{bmatrix} \beta_{1x} & \beta_{0x} \\ \beta_{1y} & \beta_{0y} \end{bmatrix} = h_{22} \begin{bmatrix} 0 & f \\ -1 & ox \end{bmatrix} \begin{bmatrix} h_{11}^* & h_{12}^* \\ h_{21}^* & 1 \end{bmatrix},
\] (2.31)

where \(h_{11}^*, h_{12}^*, \) and \(h_{21}^*\) are respectively, the parameters \(h_{11}, h_{12}, \) and \(h_{21}\) divided by \(\lambda h_{22}\), as before. Given the camera parameters \(f\) and \(ox\) which can be obtained, for every camera, with a calibration process, and provided the perspective transformation parameters estimated for each hypothesis, \(h_{11}^*, h_{12}^*, \) and \(h_{21}^*\), one can obtain the vectors \(\beta_0\) and \(\beta_1\) apart from the scaling factor \(h_{22}\):

\[
\begin{align*}
\beta_1 &= h_{22} \begin{bmatrix} fh_{21}^* \\ -h_{11}^* + oxh_{21}^* \end{bmatrix}, \\
\beta_0 &= h_{22} \begin{bmatrix} f \\ h_{12}^* + ox \end{bmatrix}.
\end{align*}
\] (2.32)
The value of $h_{22}$ is the one that brings $\beta_1$’s norm to unity, which means:

$$h_{22} = \frac{1}{\sqrt{f^2 + (h_{12} + a_z)^2}}. \quad (2.33)$$

From Figure 2.13 one can see that $\beta_0$ is directly the position of the barcode center with respect to the camera frame, and $\beta_1$ gives the orientation of the barcode surface, which defines the barcode orientation angle in the camera frame, which is of opposite value of the camera orientation in the barcode frame, $\theta_c^b$. The expression for that angle can be obtained from the angle between the vector orthogonal to $\beta_1$ with the unit vector defining the camera optical axis, and is as follows:

$$-\theta_c^b = \pi + \text{sign} \left( \frac{\beta_{1x}}{\beta_{1y}} \right) \arccos \left( \frac{1}{\sqrt{1 + \left( \frac{\beta_{1x}}{\beta_{1y}} \right)^2}} \right). \quad (2.34)$$

From this expression, $\theta_c^b$ can be computed. The camera position in the barcode frame, $(x_c^b, y_c^b)$, can be obtained by applying the inverse of the barcode axis transformation matrix to the origin of the camera frame, as follows:

$$\begin{bmatrix} x_c^b \\ y_c^b \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-\theta_c^b) & -\sin(-\theta_c^b) & \beta_{0x} \\ \sin(-\theta_c^b) & \cos(-\theta_c^b) & \beta_{0y} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (2.35)$$
Chapter 3

Localization System

In this chapter the localization algorithm is derived based on the Extended Kalman Filter (EKF) algorithm. The motion and observation models of the EKF are developed using respectively, a simple linearized kinematic model for differential wheel robots, and the measurements obtained from the barcode detector (barcode absolute pose in the environment and camera relative pose to the barcode). In the end of the chapter, a brief description of the implementation of the localization algorithm in the low capability robot is provided.

3.1 Localization Algorithm

3.1.1 Bayesian Filter

The robot position in the environment is associated with a state, \( x_k \). In each \( x_k \) the robot can acquire measurements related to the robot movement or the observation of environment landmarks. A movement related measurement, \( u_k \), captures the movement carried out from \( x_k \) to \( x_{k+1} \). This measurement is subjected to noise, \( w_k \). The state \( x_{k+1} \) is predicted from \( x_k \), \( u_k \) and \( w_k \) with a function, \( f_k(x_k, u_k, w_k) \), referred as the motion model. An environment related measurement, \( z_k \), measures some quantity computed from the landmark observations acquired in the current state, \( x_k \). The measurement is subjected to noise, \( v_k \) and can be predicted from the current state, \( x_k \), and \( v_k \) with a function, \( h_k(x_k, v_k) \), referred as observation or sensor model. The measurement noises, \( w_k \) and \( v_k \), are associated with inaccuracies of the respective models and environment noise. Thus, they are modeled by probability distributions which capture their observed statistical behavior. Consequently, \( u_k \) and \( z_k \) are represented as random variables \( U_k \) and \( Z_k \). The sets containing the measurements
taken until state $x_k$ are named $u^k$ and $z^k$, and the sets containing the corresponding random variables are named $U^k$ and $Z^k$.

Since every measurement is subjected to noise, one can only perform estimations on $x_k$, consisting on probability distributions, $X_k$, which can express the estimation uncertainties. Thus, the state’s estimate can be expressed as $P(X_k = x|U^k = u^k, Z^k = z^k)$. If $u_k$ is acquired, then $x_k$ changes to $x_{k+1}$. The estimate for this new state becomes $P(X_{k+1} = x|U^k = u^k, Z^k = z^k, U_k = u_k)$. Using the Bayesian law combined with the total probability law, this new estimate can be written as:

$$P(X_{k+1} = x|U^k = u^k, Z^k = z^k, U_k = u_k) = \int P(X_{k+1} = x|X_k = t, U^k = u^k, Z^k = z^k, U_k = u_k)P(X_k = t|U^k = u^k, Z^k = z^k, U_k = u_k)dt. \hspace{1cm} (3.1)$$

Provided a motion model, $x_{k+1}$ is completely described by $x_k$ and $u_k$, which means that in the first element of the integral in (3.1), one can drop the $X_{k+1}$’s dependencies on $U^k$ and $Z^k$. Also, in the last element of the integral, one can drop the $X_k$’s dependency on $U_k$, since $x_k$ occurs before $u_k$ is acquired. Thus the latter expression can be simplified in:

$$P(X_{k+1} = x|U^k = u^k, Z^k = z^k, U_k = u_k) = \int P(X_{k+1} = x|X_k = t, U_k = u_k)P(X_k = t|U^k = u^k, Z^k = z^k)dt, \hspace{1cm} (3.2)$$

where the left element of the integral is completely defined by the motion model, and the right element is the current state’s estimate. This means, given $X_k$, it is possible to compute $X_{k+1}$ provided $u_k$, using (3.2). This process is referred in the literature as a predict step, since it is predicting the next robot state given a movement measurement. If $z_k$ is acquired, the new estimate for the state becomes $P(X_k = x|U^k = u^k, Z^k = z^k, Z_k = z_k)$. Using Bayes’ law the following can be written:

$$P(X_k = x|U^k = u^k, Z^k = z^k, Z_k = z_k) = \frac{P(Z_k = z_k|X_k = x, U^k = u^k, Z^k = z^k)P(X_k = x|U^k = u^k, Z^k = z^k)}{P(Z_k = z_k|U^k = u^k, Z^k = z^k)}. \hspace{1cm} (3.3)$$

Since the denominator of the latter expression does not depend on $x$, it can be seen as a normalization constant, $\frac{1}{\eta}$, which brings the integral of the above distribution to unity. Also, provided an observation model, $x_k$ completely describes each $z_k$, which means that in the left side of the numerator, one can
3.1. LOCALIZATION ALGORITHM

drop the $X_k$’s dependencies on $Z_k$ and $U^k$. Thus the expression can be simplified in:

$$P(X_k = x|U^k = u^k, Z^k = z^k, Z_k = z_k) = \eta P(Z_k = z_k|X_k = x) P(X_k = x|U^k = u^k, Z^k = z^k), \quad (3.4)$$

where the left element of the right side of the expression is completely defined by the observation model, and the right element is the previous state estimate without the novelty provided by $z_k$. This means, given $X_k$, it is possible to obtain a refined estimation, $X^*_k$ provided $z_k$, using (3.4). This process is referred in the literature as an update step since it consists of a state estimate update. Examples of the predict and update steps are illustrated in Figure 3.1. These two steps describe a Bayesian filter on which this work is based to perform robot localization.

Figure 3.1: Predict and update steps when performing robot position estimation across the environment.


3.1.2 Motion Model

The robot used in this work is a differential wheel robot, which means its previous orientation is needed besides \( u_k \), to define its new position. Thus, the robot orientation in the environment, \( \theta_k \), must be estimated together with its position in the environment, \((x_k, y_k)\). Therefore each robot state consists here in a 2D pose in the environment, \((x_k, y_k, \theta_k)\). Figure 3.2 illustrates a possible movement between two states. The distances traveled by the right and left wheels between those states, respectively \( \delta_r \) and \( \delta_l \), compose \( u_k \). The resulting displacement in the orientation is obtained as follows:

\[
\delta_\theta = \theta_{k+1} - \theta_k = \frac{\delta_r - \delta_l}{L},
\]

where \( L \) is the distance between the robot wheels. In this work, \( u_k \) is collected at a rate sufficiently fast compared to the maximum speed of the robot. Under this condition \( \delta_\theta \) is small for each \( u_k \), which allows to approximate the curved trajectory illustrated in Figure 3.2, into a linear one. The \( x \) and \( y \) displacements, respectively \( \delta_x \) and \( \delta_y \), can be obtained as follows:

\[
\delta_x = \delta_s \cos(\theta_k), \quad \delta_y = \delta_s \sin(\theta_k).
\]

Relating \( \delta_s \) with \( \delta_r \) and \( \delta_l \), and using (3.5), the robot pose in \( s_{k+1} \) can be obtained as follows:

\[
\begin{bmatrix}
  x_{k+1} \\
  y_{k+1} \\
  \theta_{k+1}
\end{bmatrix} =
\begin{bmatrix}
  x_k \\
  y_k \\
  \theta_k
\end{bmatrix} +
\begin{bmatrix}
  \cos(\theta_k) & \cos(\theta_k) \\
  \sin(\theta_k) & \sin(\theta_k) \\
  \frac{1}{L} & -\frac{1}{L}
\end{bmatrix}
\begin{bmatrix}
  \delta_r \\
  \delta_l
\end{bmatrix} +
\begin{bmatrix}
  \cos(\theta_k) & \cos(\theta_k) \\
  \sin(\theta_k) & \sin(\theta_k) \\
  \frac{1}{L} & -\frac{1}{L}
\end{bmatrix}
\begin{bmatrix}
  w_r \\
  w_l
\end{bmatrix}.
\]

The latter expression defines the motion model. Here, the movement noise, \( w_k \), is included in the measurements \( \delta_r \) and \( \delta_l \), modeled by a zero mean Gaussian distribution, as follows:

\[
w_k = \begin{bmatrix} w_r \\ w_l \end{bmatrix} \sim N \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, q \begin{bmatrix} \delta_r^2 & 0 \\ 0 & \delta_l^2 \end{bmatrix} \right),
\]

where \( w_r \) and \( w_l \) are the noise variables for the respective wheel, and \( q \) is a proportionality factor, with a value between 0 and 1. The noise spreads across the motion model, modifying (3.7) as follows:

\[
\begin{bmatrix}
  x_{k+1} \\
  y_{k+1} \\
  \theta_{k+1}
\end{bmatrix} =
\begin{bmatrix}
  x_k \\
  y_k \\
  \theta_k
\end{bmatrix} +
\begin{bmatrix}
  \cos(\theta_k) & \cos(\theta_k) \\
  \sin(\theta_k) & \sin(\theta_k) \\
  \frac{1}{L} & -\frac{1}{L}
\end{bmatrix}
\begin{bmatrix}
  \delta_r \\
  \delta_l
\end{bmatrix} +
\begin{bmatrix}
  \cos(\theta_k) & \cos(\theta_k) \\
  \sin(\theta_k) & \sin(\theta_k) \\
  \frac{1}{L} & -\frac{1}{L}
\end{bmatrix}
\begin{bmatrix}
  w_r \\
  w_l
\end{bmatrix}.
\]
3.1. LOCALIZATION ALGORITHM

3.1.3 Observation Model

Barcodes are the considered landmarks to be used with the localization system. Therefore, the barcode detector developed in Chapter 2 is applied to provide the landmark measurements. Each barcode measurement is composed of an embedded code, containing the landmark’s absolute pose in the environment, \((x^b_w, y^b_w, \theta^b_w)\), and the pose of the observing robot’s camera in the barcode frame, \((x^c_b, y^c_b, \theta^c_b)\).

The binary sequence of the embedded code is composed by the three independent sequences for each element of \((x^b_w, y^b_w, \theta^b_w)\), respectively \(\text{bin}_x\), \(\text{bin}_y\) and \(\text{bin}_\theta\). To obtain each element, the inverse of the process defined in (2.12) is applied to the respective sequence:

\[
\begin{align*}
  c &= c_{\text{min}} + \frac{\text{decimal}(\text{bin}_c)}{2^{n_c}}(c_{\text{max}} - c_{\text{min}}),
\end{align*}
\]  

(3.10)

where \(c\) is the respective element, \(c_{\text{min}}\) and \(c_{\text{max}}\) the element’s variation boundaries, and \(n_c\) the number of bits used for the sequence. The operator \(\text{decimal}\) transforms binary numbers to decimal numbers.

As illustrated in Figure 3.3, the robot pose does not coincide exactly with the camera pose. However, their relation is supposed to be known, for example, by specifying the camera pose in the robot frame, \((x^c_r, y^c_r, \theta^c_r)\). A rotation in 2D space of an angle \(\theta\), is defined as follows:

\[
M(\theta) = \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) & 0 \\
  \sin(\theta) & \cos(\theta) & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]  

(3.11)
CHAPTER 3. LOCALIZATION SYSTEM

The robot pose in the camera frame, \((x^r_c, y^r_c, \theta^r_c)\), can be obtained as follows:

\[
\begin{bmatrix}
x^c_r \\
y^c_r \\
\theta^c_r 
\end{bmatrix} = -M(\theta^c)^{-1}
\begin{bmatrix}
x^r_r \\
y^r_r \\
\theta^r_r 
\end{bmatrix}.
\] (3.12)

This pose can be initially computed since the position of the camera in the robot is supposed to be fixed and known. The robot pose in the barcode frame, \((x^r_b, y^r_b, \theta^r_b)\), is obtained by converting \((x^r_c, y^r_c, \theta^r_c)\) to the barcode frame, using \((x^b_c, y^b_c, \theta^b_c)\), as follows:

\[
\begin{bmatrix}
x^r_b \\
y^r_b \\
\theta^r_b 
\end{bmatrix} = M(\theta^b)
\begin{bmatrix}
x^c_r \\
y^c_r \\
\theta^c_r 
\end{bmatrix} +
\begin{bmatrix}
x^b_r \\
y^b_r \\
\theta^b_r 
\end{bmatrix}.
\] (3.13)

From Figure 3.4 it can be concluded that, for each barcode measurement, it is possible to obtain the robot pose in the environment, \((x_k, y_k, \theta_k)\), using \((x^b_w, y^b_w, \theta^b_w)\), computed from (3.10), and \((x^r_b, y^r_b, \theta^r_b)\), computed from (3.13), as follows:

\[
\begin{bmatrix}
x_k \\
y_k \\
\theta_k 
\end{bmatrix} = M(\theta^w)
\begin{bmatrix}
x^r_b \\
y^r_b \\
\theta^r_b 
\end{bmatrix} +
\begin{bmatrix}
x^b_w \\
y^b_w \\
\theta^b_w 
\end{bmatrix}.
\] (3.14)

In this work, \((x_k, y_k, \theta_k)\) is considered as the environment observation, \(z_k\), which means the latter expression defines the observation model. The observation noise, \(v_k\), is included in \((x^r_c, y^r_c, \theta^r_c)\), repre-
senting the estimation errors in the homography matrix derived from errors in the barcode transition projections in the camera. This noise is modeled by a zero mean Gaussian distribution, as follows:

\[
v_k = \begin{bmatrix} v_x \\ v_y \\ v_\theta \end{bmatrix} \sim N\left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, R(x^c_b, y^c_b, \theta^c_b) \right),
\]

(3.15)

where \( v_x \), \( v_y \) and \( v_\theta \) are the noise variables for each pose element, and \( R \) the 3x3 covariance matrix which translates their uncertainties. In this work an empirical model is proposed for this matrix, which is derived in the results section. Since \((x^c_b, y^c_b, \theta^c_b)\) is used to compute \((x^r_b, y^r_b, \theta^r_b)\), \( v \) needs to be considered in (3.13). Adding the noise and using (3.13) in (3.14), the observation model can be written as follows:

\[
z_k = \begin{bmatrix} x^b_w \\ y^b_w \\ \theta^b_w \end{bmatrix} + M(\theta^b_w) + M(\theta^b_w + v_\theta) + M(\theta^b_w + v_\theta) + M(\theta^b_w + v_\theta).
\]

(3.16)

The latter expression can be written in a simplified way, using \((x_k, y_k, \theta_k)\) defined in (3.14), as follows:

\[
z_k = \begin{bmatrix} x_k(v_\theta) \\ y_k(v_\theta) \\ \theta_k(v_\theta) \end{bmatrix} + M(\theta^b_w) + M(\theta^b_w + v_\theta).
\]

(3.17)

### 3.1.4 Extended Kalman Filter

In this work, the Extended Kalman Filter (EKF) method is used to implement the Bayesian filter for the estimation of the robot pose in the environment. The noise, \( w_k \) and \( v_k \), and estimate distributions are assumed Gaussian. This method linearizes the motion and observation models, respectively \( f_k(x_k, u_k, w_k) \) and \( h_k(x_k, v_k) \), around the noise and estimate averages, resulting in the respective linear expressions:

\[
x_{k+1} = f_k(\hat{x}_k, w_k, 0) + A_k(x_k - \hat{x}_k) + W_k w_k,
\]

\[
x_k \sim \mathcal{N}(\hat{x}_k, P_k), \quad x_{k+1} \sim \mathcal{N}(\hat{x}_{k+1}, P_{k+1}), \quad w_k \sim \mathcal{N}(0, Q_k),
\]

(3.18)

\[
z_k = h_k(\hat{x}_k, 0) + H_k(x_k - \hat{x}_k) + V_k v_k,
\]

\[
x_k \sim \mathcal{N}(\hat{x}_k, P_k), \quad v_k \sim \mathcal{N}(0, R_k),
\]

(3.19)
CHAPTER 3. LOCALIZATION SYSTEM

Figure 3.4: Example of a barcode observation scenario, under the robot’s k’s state.

\[ A_k = \frac{\partial f_k}{\partial x_k} (\hat{x}_k, 0), \quad W_k = \frac{\partial f_k}{\partial u_k} (\hat{x}_k, 0), \quad H_k = \frac{\partial h_k}{\partial x_k} (\hat{x}_k, 0), \quad V_k = \frac{\partial h_k}{\partial v_k} (\hat{x}_k, 0), \quad (3.20) \]

where \( \hat{x} \) and P are the distribution parameters of the estimates, and \( Q_k \) and \( R_k \) represent the noise in the respective model, for each measurement. According to the Kalman Filter (KF) method, presented in [27], the estimates obtained from (3.2) and (3.4), based on linear models and Gaussian distributions, have Gaussian distributions. In [27], the expressions of the distribution parameters for those estimates are derived. In [28], those expressions are adapted to the models defined in (3.18) and (3.19). The adapted expressions for the estimates obtained in the predict step are as follows:

\[ x_{k+1} \sim P(X_{k+1} = x|U^k = u^k, Z^k = z^k, U_k = u_k) = N(\hat{x}_{k+1}, P_{k+1}), \quad (3.21) \]

\[ \hat{x}_{k+1} = f_k(\hat{x}_k, u_k, 0), \quad P_{k+1} = A_k P_k A_k^T + W_k Q_k W_k^T. \quad (3.22) \]

The adapted expressions for the estimates obtained in the update step are as follows:

\[ x_k^* \sim P(X_k = x|U^k = u^k, Z^k = z^k, Z_k = z_k) = N(\hat{x}_k^*, P_k^*), \quad (3.23) \]

\[ \hat{x}_k^* = \hat{x}_k - K_k (z_k - h_k(\hat{x}_k, 0)) \quad P_k^* = (I - K_k H_k) P_k, \quad (3.24) \]
3.2. SYSTEM IMPLEMENTATION

\[ K_k = P_k H_k^T \left( H_k P_k H_k^T + V_k R_k V_k^T \right)^{-1}. \] (3.25)

The parameter \( K_k \) is referred in the literature as the Kalman Gain. In this work, \( Q_k \) and \( R_k \) are expressed by the covariance matrices defined respectively, in (3.8) and (3.15). The estimate, \( x_k \), is represented by the robot pose in the environment, \((x_k, y_k, \theta_k)\). Therefore, the estimate covariance, \( P_k \), is a 3x3 matrix. The motion model is defined in (3.9), with \( u_k \) represented by \((\delta_r, \delta_l)\). The parameters \( A_k \) and \( W_k \) are obtained by applying to the model the respective expressions in (3.20). The results are as follows:

\[
A_k = \begin{bmatrix}
1 & 0 & (\delta_r + \delta_l) \frac{\sin(\theta_{k-1})}{2} \\
0 & 1 & (\delta_r + \delta_l) \frac{\cos(\theta_{k-1})}{2} \\
0 & 0 & 1
\end{bmatrix}, \quad W_k = \begin{bmatrix}
\frac{\cos(\theta_{k-1})}{2} & \frac{\cos(\theta_{k-1})}{2} & 0 \\
\frac{\sin(\theta_{k-1})}{2} & \frac{\sin(\theta_{k-1})}{2} & 0 \\
1 & -
\end{bmatrix}. \] (3.26)

The observation model is defined in (3.17). The parameters \( H_k \) and \( V_k \) are obtained by applying to the model the respective expressions in (3.20). The results are as follows:

\[
H_k = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad V_k = \begin{bmatrix}
1 & 0 & - (x_c^e \sin(\theta_c^e) + y_c^e \cos(\theta_c^e)) \\
0 & 1 & - (y_c^e \sin(\theta_c^e) - x_c^e \cos(\theta_c^e)) \\
0 & 0 & 1
\end{bmatrix}. \] (3.27)

3.2 System Implementation

3.2.1 Extended Kalman Filter Implementation

The robot pose in the environment, \((x_k, y_k, \theta_k)\), is estimated onboard the robot. An initial estimate, defined by its distribution parameters \( \hat{x}_0 \) and \( P_0 \), is provided, which describes the robot’s initial pose. The movement measurements, defined by the distance traveled by each wheel, \( \delta_r \) and \( \delta_l \), are measured using directly the stepper motor pulses. The predict step is implemented by using the gathered \((\delta_r, \delta_l)\) information and the current pose estimate in (3.22), with \( f_k \) defined in (3.9), and \( A_k \) and \( W_k \) defined in (3.26). The images captured by the onboard camera are analyzed for barcode measurements, using a barcode sensor, which applies the method described in chapter 2. The update step is implemented by using \((x^b_w, y^b_w, \theta^b_w)\) and \((x^c_b, y^c_b, \theta^c_b)\), extracted from each barcode measurement, in (3.24), with \( h_k \) defined in (3.17), and \( H_k \) and \( V_k \) defined (3.27).
3.2.2 System Architecture

The localization system is divided into three main parts: barcode detection for every captured image, movement measurement acquisition and robot pose estimation. Since capturing the image and processing it to scan for barcodes is the most demanding task, barcode detection is considered the bottleneck, and thus it is always executing. The acquired barcode measurements are assigned with a time stamp corresponding to the half of the respective image capture time, $t_{cap}$. Movement measurements are periodically acquired while the execution of the barcode detection or the pose estimation. The measurements contain information about the movement carried out by each wheel between storage periods, $v_o$, assigned with a time stamp referring to the time when the information was stored, $t_o$. After scanning the current image for barcodes, a new image capture starts immediately along with the robot pose estimation. Figure 3.5 shows the time diagram for the three phases.

![Time diagram for the major processing parts of the localization system.](image)

The pose estimations, ($\hat{x}, P$), are performed using EKF predict and update estimation steps, as defined in the previous subsection, with the stored movement information and barcode measurements obtained while processing the last image. First, the predict step is applied over all the odometry information stored between the beginning of the image capture and $t_{cap}$. Then, the update step is applied over the barcode measurements, $z$, gathered from the image. Finally the predict step is again applied over all the odometry information stored between $t_{cap}$ and the ending of the image capture. If no barcode measurements are available, only the predict steps are performed. In order for the robot to estimate
its pose while capturing and processing the images, the predict step is immediately applied to the
current robot pose estimate, \((\hat{x}, P)_{\text{current}}\), provided the odometry \((v_o, t_o)\). The estimated pose from
the beginning of the capture, \((\hat{x}, P)_{\text{start\_capture}}\), is stored. When the image is processed the pose is
estimated using the latter process applied on \((\hat{x}, P)_{\text{start\_capture}}\). The main process developed to run
onboard the E-puck can be described by the following algorithm:

\begin{algorithm}
\caption{The E-puck behavior}
\begin{algorithmic}[1]
\Procedure{EpuckMainLoop}{ }
\Loop
\State cleanup odometry list
\State \((\hat{x}, P)_{\text{start\_capture}} \leftarrow (\hat{x}, P)_{\text{current}}\)
\State StartCapture
\While{capturing}
\State \((t_o, v_o) \leftarrow \text{GatherIncomingOdometry}\)
\State add \((t_o, v_o)\) to odometry list
\State \((\hat{x}, P)_{\text{current}} \leftarrow \text{Predict\_Pose}((\hat{x}, P)_{\text{current}}, (t_o, v_o))\)
\EndWhile
\State \((t_{\text{cap}}, z) \leftarrow \text{FinishCapture};\)
\ForAll{\((t_o, v_o) \in \text{odometry list}\)}
\If{\(t_o < t_{\text{cap}}\)}
\State \((\hat{x}, P)_{\text{start\_capture}} \leftarrow \text{Predict\_Pose}((\hat{x}, P)_{\text{start\_capture}}, (t_o, v_o))\)
\EndIf
\EndFor
\State \((\hat{x}, P)_{\text{start\_capture}} \leftarrow \text{Update\_Pose}((\hat{x}, P)_{\text{start\_capture}}, z)\)
\ForAll{\((t_o, v_o) \in \text{odometry list}\)}
\If{\(t_o > t_{\text{cap}}\)}
\State \((\hat{x}, P)_{\text{start\_capture}} \leftarrow \text{Predict\_Pose}((\hat{x}, P)_{\text{start\_capture}}, (t_o, v_o))\)
\EndIf
\EndFor
\State \((\hat{x}, P)_{\text{current}} \leftarrow (\hat{x}, P)_{\text{start\_capture}}\)
\EndLoop
\EndProcedure
\end{algorithmic}
\end{algorithm}
Chapter 4

Performance Evaluation

This chapter provides the evaluation for the developed barcode detector and localization algorithms. The barcode detection rates and ranges are evaluated throughout several distance and orientation configurations in order to analyze the performance of the barcode detector and to validate the developed mathematical framework. A statistical evaluation for the extracted relative pose of the camera to the barcode, over all experiment configurations, is conducted in order to provide a noise model for the observation measurements. The accuracy and convergence proprieties of the developed localization system are evaluated by comparing its estimates against the groundtruth and odometry estimates on a triangular scenario. A kidnapping situation is also considered to evaluate the algorithm convergence speed.

4.1 Barcode Detector

To evaluate the accuracy of the barcode design framework and the performance of the barcode detection algorithm, a barcode dataset was created, composed of single and double layer barcodes. In each layer there are 24 stripes: 16 stripes are used for the embedded code, and 2 bar-space pairs for each guard set. For single layer barcodes, the embedded code reserves 12 bits (4 bits for each coordinate) to store \((x^b_w, y^b_w, \theta^b_w)\) and the last 4 bits to store the CRC error detection bits. For double layer barcodes, the embedded code reserves 24 bits (8 bits for each coordinate) to store \((x^b_w, y^b_w, \theta^b_w)\) and the last 8 bits to store the CRC error detection bits. Each barcode is associated with several images captured from the robot’s camera, visualizing the barcodes from fixed positions. The barcodes are deployed on a vertical white wall. The distance to the barcode of each fixed camera position, \(d\), is varied between 10cm and 80cm, for single layer barcodes, and between 10cm and 60cm, for double layer barcodes, in
5cm steps. The inclination to the barcode of each fixed camera position, $\theta$, is varied, for both barcode types, between 0$^0$ and 60$^0$ in 15$^0$ steps. The camera has a focal length of 170 vertical pixels and 680 horizontal pixels, and the captured images are composed of 480 pixels width and 8 pixels height. Figure 4.1 shows two barcode visualization position examples: one with and one without inclination.

Two cases are here considered. The first evaluates the barcode detection performance for $\theta = 0^0$. Single and double layer barcode detection performance are both evaluated, using 11 barcodes of each type: 5 pathological cases, corresponding to the best and worst possible detection conditions for the barcode detector, and other 6 random barcodes. Figure 4.2 presents the obtained detection rates in function of $d$, for each barcode type. One can observe that those rates become zero for all the single layer barcodes at approximately 70cm, and for double layer barcodes at approximately 50cm. Also the detection rate drops sharply in the latter case. The detection ranges differ because they are being limited by different maximum observation distances. For single layer barcodes, the maximum observation distance is defined according to the boundary set by the hyperbola expression in (2.9), which is shown in Figure 4.3 by the black dot corresponding to $\theta = 0^0$. For double layer barcodes, the maximum observation distance is derived by the scan line inclination, and is computable from (2.13). In this case ($\theta = 0^0$), and for the barcode and camera parameters used, this distance is about 53cm.

For single layer barcodes, which use 4 CRC error detection bits, 27 performed detections in 2409 detection experiments, passed the CRC test with incorrect embedded codes, corresponding to a possibility of error occurrence of about 1 percent. For double layer barcodes, which use 8 CRC error detection bits, only 14 performed detections in 2070 detection experiments, passed the CRC test with incorrect embedded codes, corresponding to a possibility of error occurrence of about 0.5 percent, which is half than the first case. Since any barcode detected by the developed algorithm is considered in
the localization system for pose update, any detection error might be fatal for the convergence of the localization algorithm, since it can correspond to a barcode in any arbitrary environment coordinate. This fact stresses the importance of double layer barcodes for the proper functioning of the localization system, since they provide larger CRC sequences.

The second case represents the barcode detections for the several $\theta$ values. Single and double layer barcode detection performance are both evaluated, using one random barcode of each type. Figure 4.3 presents the obtained detection rates in function of $d$, for each $\theta$, along with the theoretical maximum observation distances defined according to the boundary set by the hyperbola expression in (2.9), for each $\theta$. One can observe that, while the single layer barcode detection ranges follow their theoretical maximum observation distance, the double layer barcode detection ranges only follow their theoretical distances, if those are bellow 50cm. This can be explained, as above, by the existence of a second maximum observation distance, derived from the scan line inclinations, for the double layer barcodes, which varies around 50cm. A $L_{\min}$ of 1.5 pixels, instead of the early discussed 1 pixel, was used since
it better explains the results.

Figure 4.3: Barcode detection rate with distance against theoretical maximum observation distance (black dots), for several inclination cases, using a random barcode. a) Single layer barcode. b) Double layer barcode.

Figure 4.4 shows the statistical behavior of the camera pose estimates in the barcode frame, \((x_{c}^{b}, y_{c}^{b}, \theta_{c}^{b})\), obtained for the respective detections presented in Figure 4.3, for only double layer barcodes. Each fixed camera position is associated to a camera pose estimate in the barcode frame, by averaging all the poses extracted from correct barcode detections, and a covariance matrix, obtained from the same extracted poses. From that data, it is possible to build an empirical model for the 3x3 covariance matrix, \(R\), representing the noise when extracting \((x_{c}^{b}, y_{c}^{b}, \theta_{c}^{b})\), defined in (3.15). Since only the double layer barcodes are considered to be used together with the localization system, only the experiments related to that barcode type are considered to build the noise model. The covariance matrix can be expressed as follows:

\[
R(x_{c}^{b}, y_{c}^{b}, \theta_{c}^{b}) = \begin{bmatrix}
    r_{xx} & r_{xy} & r_{x\theta} \\
    r_{xy} & r_{yy} & r_{y\theta} \\
    r_{x\theta} & r_{y\theta} & r_{\theta\theta}
\end{bmatrix}, \tag{4.1}
\]

where \(r_{ab}\) is the element of the covariance matrix which relates pose element \(a\) with pose element \(b\). The coordinates \((x_{c}^{b}, y_{c}^{b})\) are assumed independent from \(\theta_{c}^{b}\), which means that \(r_{x\theta}\) and \(r_{y\theta}\) are set to zero. The upper 2x2 matrix relating the \(x\) and \(y\) pose elements, is decomposed into the rotation and eigenvalue matrices, defining respectively, the uncertainty ellipsoid axis orientation and length, as...
4.1. BARCODE DETECTOR

follows:

$$
\begin{bmatrix}
  r_{xx} & r_{xy} \\
  r_{yx} & r_{yy}
\end{bmatrix} =
\begin{bmatrix}
  \cos(\alpha) & -\sin(\alpha) \\
  \sin(\alpha) & \cos(\alpha)
\end{bmatrix}
\begin{bmatrix}
  \lambda_1 & 0 \\
  0 & \lambda_2
\end{bmatrix}
\begin{bmatrix}
  \cos(\alpha) & -\sin(\alpha) \\
  \sin(\alpha) & \cos(\alpha)
\end{bmatrix}^T,
$$

(4.2)

where $\lambda_1$ and $\lambda_2$ are the eigenvalues defining the length of respectively, the major and the minor axes of the uncertainty ellipsoid, and $\alpha$ is the rotation angle of the major axis. An analysis of $\lambda_1$ and $\lambda_2$ shows exponential behaviors with the distance to the barcode, $d$, similar for all inclinations to the barcode, $\theta$. An analysis of the rotation angle shows that there is little influence of $d$ and a linear relation with $\theta$. The empirical regressions, using mean squares, for both eigenvalues and the rotation angle, for the 2x2 matrix representing that uncertainty, are as follows:

$$
\lambda_1(d) = e^{25.70d-15.46} \quad \lambda_2(d) = e^{16.15d-17.27},
$$

(4.3)

$$
\alpha(\theta) = 1.91\theta - 1.43\text{sign}(\theta).
$$

(4.4)

In the latter regressions, $\theta$ can be replaced with $\theta^c_b + \pi$, and $d$ with $\sqrt{(x^c_b)^2 + (y^c_b)^2}$, in order to obtain a covariance matrix dependent on $(x^c_b, y^c_b, \theta^c_b)$.

Figure 4.4: Statistic behavior for the extraction of the camera’s pose in barcode frame, for a random double layer barcode.
4.2 Localization System

The localization system was evaluated using the scenario defined by Figure 4.5. The robot performs several laps around the scenario, estimates its pose in real time and transmits the results to an external computer. The algorithm is saving odometry measurements with a period of 100ms and the barcode sensor takes about 400ms to process each image for barcodes (300ms for image capture and 100ms for image processing). Three double layer barcodes are deployed in the scenario, with their positions shown in the figure. A marker is placed on top of the robot, solidary with the robot frame, and is tracked by an external camera, using the ARToolKit Toolbox\footnote{URL: http://www.hitl.washington.edu/artoolkit/ retrieved 8 March 2012} in order to provide a ground truth for the robot pose in the environment. The ARToolKit Toolbox gives the marker 3D pose in the tracking camera frame, defined as \((X, Y, Z, \Psi, \Theta, \Phi)\), where the first three elements define the center of the marker. This pose needs to be converted to a 2D pose in the scenario frame, \((x_s, y_s, \theta_s)\).

![Figure 4.5: Scenario used to analyze the performance of the developed localization system.](image)

The 2D position in the scenario of the marker, \((x_s, y_s)\), is directly obtained from a rigid transformation applied to its 3D position \((X, Y, Z)\) in the tracking camera. This transformation relates points from an arbitrary plane to the plane \(z = 0\), which is the scenario, and is defined as follows:

\[
\begin{bmatrix}
x_s \\
y_s \\
0
\end{bmatrix} = T(X_{3D}) \begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
\]

(4.5)

where \(T\) is the rigid transformation, consisting in a pre-calibrated 3D rotation and translation. To calibrate this transformation, a set of poses in the scenario frame, \((x_s, y_s, \theta_s)\), and ARToolKit frame, \((X, Y, Z, \Psi, \Theta, \Phi)\), are matched. The Procustres algorithm is applied to the positions of those matches,
4.2. LOCALIZATION SYSTEM

\((x_s, y_s)\) and \((X, Y, Z)\), estimating the optimal rotation and translation.

The orientation in the scenario is obtained by projecting, to the scenario plane, the 3D axis defining the marker’s 3D orientation in the tracking camera frame. The x-axis of the marker is used as orientation axis. This means that, when using the marker with the robot, the x-axis of the marker must be aligned with the robot’s orientation axis, that is also its x-axis. One of the points that characterizes the axis is the center of the marker, \((x_s, y_s, \theta_s)\). The other end of the axis, \((x_{s1}, y_{s1}, \theta_{s1})\), can be computed by two steps. First, the end of the x-axis in the marker frame, \((1, 0, 0)\), is converted to the ARToolKit frame using \((X, Y, Z, \Psi, \Theta, \Phi)\) as follows:

\[
\begin{bmatrix}
X_{\text{axis}} \\
Y_{\text{axis}} \\
Z_{\text{axis}}
\end{bmatrix} = R(\Psi, \Theta, \Phi) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + T(X, Y, Z) \tag{4.6}
\]

where \(R\) and \(T\) are, respectively, the rotation matrix and the translation vector transforming points from the marker to the tracking camera frame. Second, \((X_{\text{axis}}, Y_{\text{axis}}, Z_{\text{axis}})\) is converted to \((x_{s1}, y_{s1})\) by using the pre-calibrated transformation presented in (4.5). The orientation angle in the scenario, \(\theta_s\), is then obtained:

\[
\theta_s = \arctan \left( \frac{<(x_{s1}, y_{s1}) - (x_s, y_s), (0, 1)>}{<(x_{s1}, y_{s1}) - (x_s, y_s), (0, 1)>} \right) \tag{4.7}
\]

The ground truth accuracy is defined using the matches used to calibrate \(T(X_{3D})\), present in (4.5). Each 3D pose is converted to a scenario pose \((x'_s, y'_s, \theta'_s)\) by applying the latter method using the recently calibrated \(T(X_{3D})\). A transformation error can now be defined:

\[
(e_x, e_y, e_{\theta}) = (x'_s, y'_s, \theta'_s) - (x_s, y_s, \theta_s) \tag{4.8}
\]

where \((x_s, y_s, \theta_s)\) is the 2D scenario pose that matched to the 3D pose which originated \((x'_s, y'_s, \theta'_s)\).

To model the error distribution of the ground truth, the point grid in Figure 4.6a was used, considering 60 points in total. From the experiments, this error has shown to have almost zero mean, \(\mu\) (less than \(10^{-5}\) meters and 0.25 degrees). The obtained covariance matrix, \(V\), is used to characterize the degree of accuracy. The random variable representing the error defined in (4.8) is named \(E_{GT}\) and is assumed to follow a multivariate normal distribution with zero mean and \(V\) as covariance matrix, constant for any position in the scenario. This fact can be eventually observed if one increases the number of points that model the distribution. To examine the possibility of the assumption, first the error is normalized into a unitary uncorrelated multivariate normal distribution, \(Z\), according to the Mahalanobis decorrelation.
transformation, described as follows:

\[ Z = \Sigma^{-1}(E_{GT} - \mu) \quad \Sigma \Sigma^T = V \quad Z \sim N\left(\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, I_{3x3}\right) \] (4.9)

Figure 4.6: Ground truth stochastic behavior. a) Grid of points scattered across the scenario, used for the experiments (4 orientations for each point). b) Histogram of \( L \) values computed for the errors obtained with the point grid, using a discretization of 34 bins, against a \( \chi^2_3 \) distribution.

The elements of \( Z \), \((Z_x, Z_y, Z_\theta)\), are all uncorrelated and thus, from \( Z \) distribution, have standard normal distributions, \( N(0,1) \). In [29] it is stated that the sum of the squares of \( k \) standard normal distributed random variables is distributed according to the chi-squared distribution with \( k \) degrees of freedom, \( \chi^2_k \). Therefore, \( Z^T Z \) should be distributed according to \( \chi^2_3 \). Using (4.9) one can obtain the following:

\[ Z^T Z = (X - \mu)^T (\Sigma^T)^{-1} \Sigma^{-1} (X - \mu) = (X - \mu)^T V^{-1} (X - \mu) = L \] (4.10)

where \( L \) is a scalar random variable following the \( \chi^2_3 \) distribution. Figure 4.6(b) shows the histogram of \( L \) values computed for the 60 grid points used to model the ground truth. From the figure one can observe that the behavior of the histogram is similar to the \( \chi^2_3 \) distribution, apart from two outliers which are caused by larger orientation errors on two of the points. A test for goodness fit, described in [29], is applied to all the \( L \) values, divided in the first 9 bins of Figure 4.6(b) plus 1 bin which represents the rest of the \( \chi^2_3 \) distribution, and where the outliers are included. Using a p-value of 5%, the null hypothesis stating that the \( L \) values follow a \( \chi^2_3 \), cannot be rejected. Thus the experimental data supports the assumption that the distribution of the \( L \) value, obtained from the normalization of \( E_{GT} \) using the Mahalanobis decorrelation transformation, is \( \chi^2_3 \).
4.2. LOCALIZATION SYSTEM

Figure 4.7 shows the results of the algorithm against the ground truth for just one lap of an experiment, in order to illustrate the accuracy of the algorithm. Figure 4.8 shows the results of the algorithm in comparison with odometry alone, for the whole path of an experiment, in which the robot is supposed to follow a triangular route. It is observed that while the algorithm estimations maintain the robot localized in a triangular route, the odometry based estimator progressively diverges from the route.

Figure 4.7: Result path estimation with the developed localization system (green/lighter) against the ground truth path (blue/darker). A single random lap was selected from the whole path.

Figure 4.8: Result path estimation with just odometry (red/darker) and with the developed localization system (green/lighter), for an experiment in which a triangular route is performed.
Figure 4.9 shows the Euclidean distance and the orientation errors between the results of the algorithm and the ground truth, throughout the whole path of an experiment. The marked points correspond to instants where the algorithm used barcode information for pose update. The figure shows a maximum errors about 5.5cm in position (Figure 4.9 a)) and 25° in orientation, and average errors of about 1.6cm in position and 4° in orientation (Figure 4.9 b)).

Figure 4.9: Estimation error quantification of the algorithm along the robot’s path. a) Position error. b) Orientation error. c) Statistic behavior comparison with the ground truth using a p-value of 1%.
It is possible to compare the obtained errors with the ground truth accuracy by computing the $L$ values, defined in (4.10), for each one of them, using the statistical parameters of the ground truth ($\mu$ and $V$). Figure 4.9(c) shows the computed $L$ values for the errors obtained for the entire trajectory. From the ground truth calibration, the $L$ value computed for the errors of the ground truth is assumed to follow a $\chi^2_3$ distribution. The error from the experiments is considered within the ground truth accuracy, if its respective $L$ value could have been generated from that distribution, with a significance of at least 1%. Therefore, a p-value of 1% over the $\chi^2_3$ is used to establish this relation for each isolated error point. The p-value is presented in Figure 4.9(c) with the black line. All points above the line can be generated by the ground truth with less than 1% probability, and thus the errors can be considered to be caused by the algorithm inaccuracies. For points below the line, nothing can be concluded about algorithm inaccuracies. From the results, one can observe that about 73% of the points are below the p-value limit, which means that for the majority of the robot path, the observed localization error is within the ground truth accuracy, and thus the algorithm can be considered at least as accurate as the ground truth.

To evaluate the recovery capability of the localization system in case of a pose estimation divergence from the real pose, an experiment was conducted where the robot starts in a position different from the initial estimation and at some point, after the first position estimations using barcodes, a kidnap situation is simulated, by deliberately moving the robot to a different pose in the scenario. Figure 4.10 depicts the distance and orientation errors between the algorithm’s results and the ground truth for that experiment. From the figure one can observe that in the beginning of the experiment and when the kidnapping occurs the localization error is high, and keeps rising with the robot movement (this happens because the orientation of the robot is also changed in the kidnapping). However, provided very few barcode observations (in this case two to four) the error drops instantly to normal values, which means that the robot algorithm has again converged to the right estimate. From there, the estimations resume their normal behavior.

From the conducted experiments one can conclude that by using barcodes for pose estimation, the robot pose estimate stays on the correct path, in contrast with using only the odometry measurements, which leads the robot pose estimate to a progressive degeneration from the correct path. Also, in case the pose estimate diverges from the real robot pose, the algorithm exhibits a fast recovery provided only a few barcode measurements.
Figure 4.10: Estimation error quantification for a kidnapping situation. a) Position error. b) Orientation error. c) Statistic behavior comparison with the ground truth using a p-value of 1%.
Chapter 5

Conclusion

This work presents the development of a localization system for robots with extreme low computational capabilities, specifically the E-puck robot, with 8KB RAM and a 16MIPS processor. For the used barcode and camera configurations, the barcode detector is capable to detect barcodes with distances from 5cm to 50cm and angles from 0 to 60 degrees. The greater the inclination of the robot to the barcode, the smaller is the maximum barcode detection range of the robot. From the evaluation over the localization system one can observe that, for experiment configurations, the localization method converges with a maximum error of about 5cm, which is almost half of the robot size. It is also showed that 73% of the obtained error is within the limits of the ground truth precision.

The use of barcodes provides several advantages: the barcode vertical redundancy minimizes the size of the image acquired by the onboard camera, used for landmark detection; the guard sets delimiting the barcode edges computationally simplify the barcode localization process in the image; the barcode unique embedded code allows the encoding of the barcode absolute pose in the environment, dispensing onboard landmark maps. The homography concept was used to minimize the computational effort present in the extraction of the camera relative pose to the barcode. This pose, along with the barcode absolute pose in the environment, allows to measure the robot absolute pose in the environment, for each barcode detection. This measure allows the implementation of the EKF method with a very simple observation model, simplifying the computational complexity of the robot pose estimation.

Despite all the simplifications performed in the developed methods, the available 8KB of memory space is being used almost entirely, half of it occupied by the used 8 B&W lines of the camera. Also, due to the low processing capabilities of the E-puck robot (16MIPS), each image takes around 300ms to be captured, which introduces delays between the landmark visualization and the actual localization update, which cause offsets in the estimations.
CHAPTER 5. CONCLUSION

With this developed localization system, multi-robot systems composed of large amounts of robots have the possibility of embracing new practical applications, where the need for self-localization is imperative.

5.1 Future Work

The implemented barcode detector uses a binarization process based on a threshold to identify black and white regions from the acquired camera images. This threshold is computed for each image, which means it is not affected by a change of the overall light intensity in the scenario. However, since the threshold is the same for all the image, it can’t be adapted to situations where the intensity of the wall changes across the image. An example of this phenomena can occur when the camera acquires an image of a scenario corner. On that image, the walls in each side of the corner have different pixel intensities. If the barcode is located in the wall with the lower intensity, there is a chance that all the barcode is classified as black, thus precluding the detection. To resolve this issue one can use higher order thresholds, based on polynomial expressions, which are capable of adapting to the differences on the image pixel intensities. The higher the order of the polynomial, the better it adapts to each situation, but, on the other hand, the greater the computational complexity required to obtain it.

The barcodes chosen for the detections are the double layer barcodes, since they maximize the amount of information that can be inserted into the embedded code. However, they make the detections very sensitive to height variations in the camera onboard each robot and in the vertical center of each deployed barcode. To tackle this problem, colored barcodes and images can be used. Since the three colors (red, green, and blue) are used to encode each colored barcode stripe, one could insert three times the information that would be possible in a black and white barcode, dispensing the need for a double barcode layer. Therefore, these barcodes can be decoded using just one colored line of the camera, requiring less space for each image. Also, the sensitivity to height variations becomes much lower. However, a colored image acquisition takes more time, and the information extraction for each stripe requires a greater computational effort.

To deploy the barcodes in the scenario, the positions which are supposed to have a better chance of being visualized by the robots, are selected. Therefore, each set of positions is dependent on the possible robot routes inside the scenario. If those routes are changed, the whole set needs to be re-obtained. These sets are not obtained automatically, and result from an analysis of the specified scenario and robot routes, which consumes time. The solution would be to synthesize an analysis process, applied on the barcode, scenario and robot route specifications, which automatically computes the barcode positions in the scenario.


