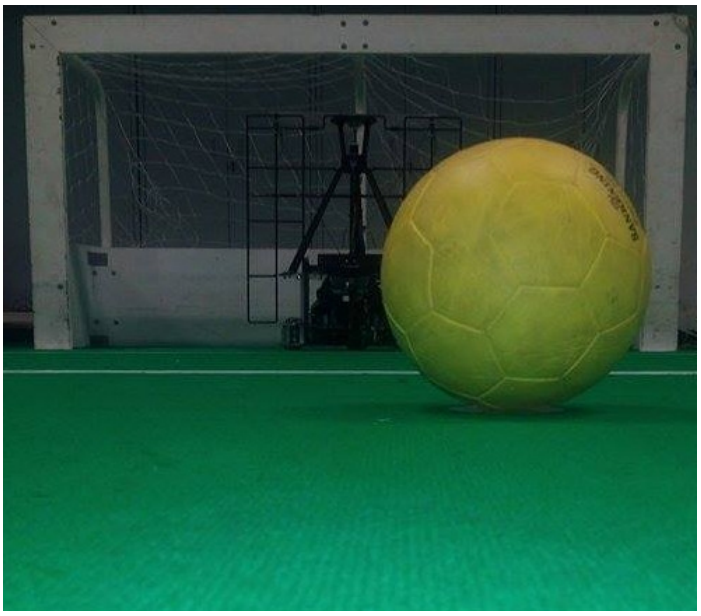
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2015

Eduardo Rodolfo Teixeira Ferreira

Deteção de baliza para Guarda-Redes e Atacante usando um Laser Range Finder para a equipa CAMBADA

Goal detection using a Laser Range Finder for Goalkeeper and Striker from CAMBADA team



Eduardo Rodolfo Teixeira Ferreira

Deteção de baliza para Guarda-Redes e Atacante usando um Laser Range Finder para a equipa CAMBADA

Goal detection using Laser Range Finder for Goalkeeper and Striker from CAMBADA team

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sobre a orientação científica do Prof. Doutor António Neves e Prof. Doutor Manuel Bernardo Cunha, Professores do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri

Presidente

Professor Doutor José Luis Costa Pinto de Azevedo

Professor Auxiliar, Universidade de Aveiro

Vogais

Doutor João Manuel Leite da Silva

Coordenador da Área de Robótica, Fábrica Centro Ciência Viva de Aveiro

Professor Doutor António José Ribeiro Neves

Professor Auxiliar, Universidade de Aveiro (orientador)

Agradecimentos

A toda equipa CAMBADA pelo tempo despendido, ambiente proporcionado e pela oportunidade de trabalhar com a mesma, deixo um obrigado. Ao professor António Neves um especial obrigado, pela paciência mesmo nos piores momentos, pela confiança depositada em qualquer situação e por ser o orientador que qualquer aluno pode desejar. Pelo apoio e sempre presentes para qualquer dúvida agradeço igualmente ao João, Ricardo e Filipe e aos professores Nuno Lau, Bernardo Cunha e José Luís Azevedo.

Agradeço à minha família por esta longa caminhada, por sempre terem acreditado que era capaz e por todo o apoio dado. Agradeço igualmente à minha segunda família (Sr. Zé, D. Sofia, Zé, Carlos e Alzira) por me terem acolhido e sempre me tratarem bem.

Devido às diversas actividades e aos anos passados na cidade de Aveiro, as amizades feitas são incontáveis, quero agradecer a todos pelos momentos passados.

Por último, à Sofia, por ser o meu pêndulo, por me obrigar a estudar, e por ser quem é, um obrigado muito especial.

Resumo

Para que um robô seja autónomo e móvel, necessita estar equipado com vários sensores que o ajudem a ter uma perceção do mundo que o rodeia, de forma a obter a sua própria localização e a deteção de objectos.

CAMBADA é a equipa de futebol robótico do grupo de investigação IRIS, da unidade de investigação IEETA, da Universidade de Aveiro que participa na Liga de Robôs Médios da RoboCup.

Em competição, para ganhar, o principal objetivo de uma equipa durante um jogo é não sofrer golos e marcar o maior número possível, desta forma, esta tese foca-se em dotar um agente de uma melhor capacidade de localização em situações defensivas e ofensivas de jogo.

Foi introduzido um *laser range finder* aos robôs da equipa CAMBADA, tornando-os aptos a detetar a sua própria baliza e a do adversário, e a detetar oponentes em situações específicas do jogo.

Com a nova informação adquirida e adaptando os *behaviors* `Goalie` e `Penalty`, agora o guarda-redes da equipa CAMBADA está apto a detetar e rastrear a sua própria baliza e o avançado da equipa CAMBADA tem uma melhor performance em situações de penalty.

O trabalho desenvolvido foi incorporado no software de competição dos robôs, o que permite nesta tese apresentar resultados experimentais de testes efectuados nos robôs em laboratório.

Abstract

For a robot be autonomous and mobile, it requires being attached with a set of sensors that helps it to have a better perception of the surrounding world, to manage to localize itself and the surrounding objects.

CAMBADA is the robotic soccer team of the IRIS research group, from IEETA, University of Aveiro, that competes in the Middle-Size League of RoboCup.

In competition, in order to win, the main objective of the game it's to score more goals than the conceded, so not conceding goals, and score as much as possible it's desirable, thus, this thesis focus on adapt an agent with a better localization capacity in defensive and offensive moments.

It was introduced a *laser range finder* to the CAMBADA robots, making them capable of detecting their own and the opponent goal, and to detect the opponents in specific game situations.

With the new information and adapting the `Goalie` and `Penalty` behaviors, the CAMBADA goalkeeper is now able to detect and track its own goal and the CAMBADA striker has a better performance in a penalty situation.

The developed work was incorporated within the competition software of the robots, which allows the presentation, in this thesis, of the experimental results obtained with physical robots on the laboratory field.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The robotics field is nowadays a massive research area. An area that studies robots, and whether having or not the notion, they are introduced in our daily lives. Thus, there are many types of robots, used for distinct environments and distinct applications, such as home environments, for example an autonomous vacuum cleaner or an automatic cooking machine, and industrial environments where different kind of robots are introduced in production lines to maximize them. Due to the fact of being involved in our daily lives, as the robotics field is moving towards artificial intelligence and the concept that a robot can compete or rival with the human being, there are the three laws of robotics, principles made by Isaac Asimov in "I, Robot" [1], which are:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given by human beings except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First and Second Laws.

Although the fact they're not real laws, but principles, and besides the fact they are from a science fiction book, every researcher in the artificial intelligence field wants them to be obeyed by its creation.

1.1 Robot

Robots are electromechanical or biomechanical devices, with distincts applications, different shapes, but all share the same purpose, being capable of, autonomously or pre-programed, fulfill a specific given task. It is an automatic device provided with several sensors and can be divided in different categories, because different tasks, requires different qualities, form and functions from the robot. One of those categories divisions can be[2]:

Mobile Robots

Robots that are able to move, provided with wheels, mechanical legs, or other hardware that allows it to move.

Stationary Robots

Robots designed in general to the industry production lines, performing repetitive tasks, that can become boring and tiresome to the human being. Usually only need maintenance and re-programming to perform other tasks.

Autonomous Robots

Robots capable of "thinking" by themselves, deciding which tasks to perform based on external conditions. They can be mobile or stationary robots.

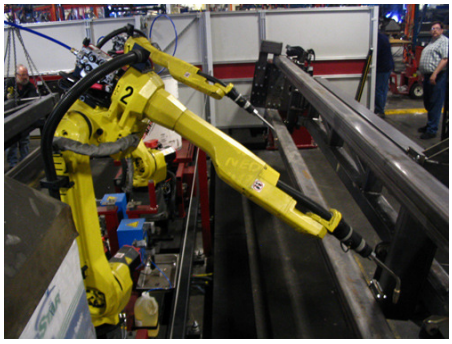
Remote-control Robots

Robots controled remotely by human beings that decide which task to perform. They can be mobile or stationary as well.

In Figure 1.1 two different robots are shown. In Figure 1.1(a) an industrial manufacturing system introduced in a production line it's simultaneously autonomous and stationary and in Figure 1.1(b) the AscTec Firefly, an hexadrone that's obviously mobile, and it's remotely controlled. However, with the introduction of vision and by reprogramming it, it may be autonomous.

1.2 RoboCup

One of the best ways to accelarate development is competing, so there are many robotic competitions where professional teams and researchers from all over the world solve engineering problems in a competitive way. RoboCup [3], started in 1997, is one of



(a) A stationary autonomous robot introduced in a production line[4].

(b) AscTec Firefly hexadrone, a remote-controlled mobile robot.

Figure 1.1: Example of two very different robots built for different purposes.

those competitions which occurs every year, and gathers groups of researchers to solve challenging problems and an opportunity to share knowledge and implementations, thus making technology improve faster. The RoboCup federation has created a goal in order to motivate researchers, that in the middle of the 21th century, a team of autonomously humanoid robots be able to defeat the current human team winner of the football World Cup.

Football as a cooperative team game, offers several individual problems, such as identification, points of interest localization, dribbling and shooting, and offers colective problems, such as passing the ball, positioning and strategic plays. These were the major motivations for the beggining of the competition, thus, RoboCup covers diverse forms of robotic soccer, being the most relevant ones:

Middle Size League (MSL)

In this league, robots have no standard format, but limited dimension and weight. Each team has 5 autonomous mobile robots, and as human intervention is forbidden on the robots, they are composed by a set of sensors and as they need to cooperate, they communicate with each other via WiFi.

Small Size League

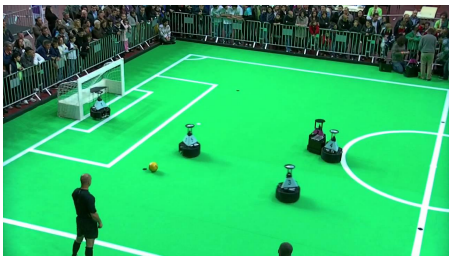
With a set of smaller autonomous robots, now the processing is made in a central processing unit, based on the information given by a video camera set above the field, the central unit sends decisions taken via WiFi.

Humanoid League

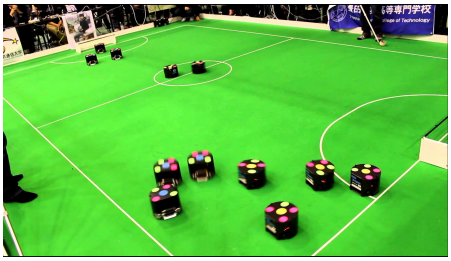
The most challenging and complex competition, as the humanoid resemblance makes the basic aspects of the game a challenging task to fulfill.

Simulation League

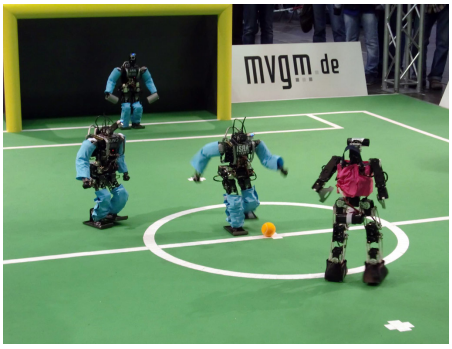
In this competition there are no real robots, being each one a single process, that runs through a simulated environment.



(a) Medium-size League soccer.



(b) Small-size League soccer competition.



(c) Humanoid soccer competition.

(d) 3D Simulation soccer competition.

Figure 1.2: Different robotic soccer competitions in RoboCup.

As the competition grew, some other challenges besides football were introduced, such as:

RoboCup Rescue

Competition where teams face a disaster scenario and must find fictional victims and map the environment.

RoboCup Junior

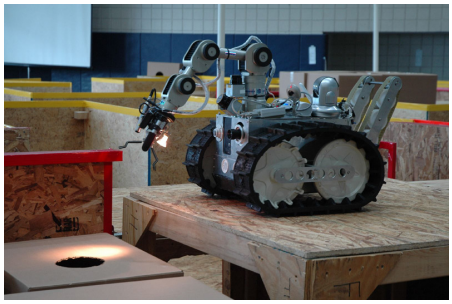
In order to increase the interest among high school students to the robotics and artificial intelligence areas, some more simple challenges are given to this young students for their first contact in the area.

RoboCup Home

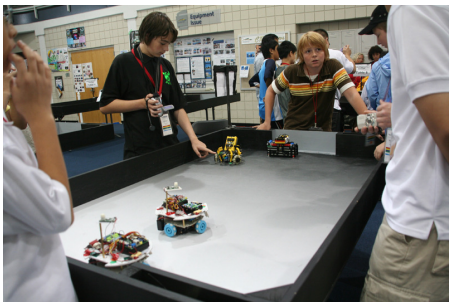
Focus on the development of autonomous mobile robots with main objective to do domestic tasks. This competition focus on the human-robot interact, with a set of tasks in order to facilitate the daily life.

RoboCup Work

This competition aims to develop robots for industrial solutions, making this robots cooperate with humans to fulfill a specific task.

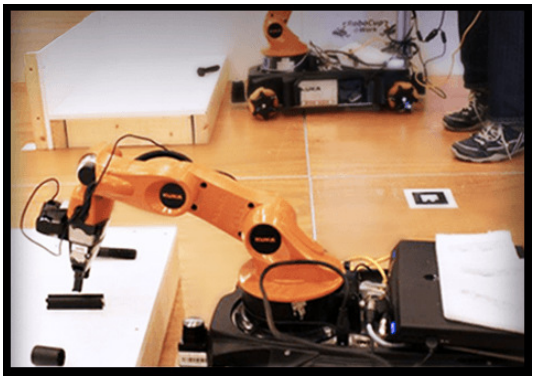


(a) Robocup@Rescue competition.



(b) RoboCup@Junior competition.

(c) RoboCup@Home competition.



(d) RoboCup@Work.

Figure 1.3: Other RoboCup competitions.

1.3 CAMBADA team

Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture (CAMBADA) is the MSL soccer team created in October 2003 at IEETA/DET, University of Aveiro. As said by A. Neves et al [5], the project coordinated by the Transverse Activity on Intelligent Robotics group of the Institute of Electronic and Telematic Engineering of Aveiro (IEETA), involves several people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers and the

software development in areas such as image analysis and processing, sensor and information fusion, reasoning and control, cooperative sensing approach based on a Real-Time Database, comunications among robots and the development of an efficient basestation.

Since the creation of the CAMBADA team, there has been a remarkable progress, both in the hardware and software of the agent, and proof of that notable work are the numerous podium places conquered in RoboCup world championships. In 2008 the team achieved its best ranking, winning the MSL and has achieved notable results in the techical challenge of the RoboCup MSL and in the RoboCup Scientific Challenge.

1.4 Motivation and Thesis Structure

1.4.1 Motivation

In the scenario of an autonomous mobile robot, the envolving environment recognition has an high significance and impact to the robots beliefs, and a vision based on landmarks method localization, though being a reliable method, it is conditioned to a good capture and external environment light conditions that may affect it. Thus, other sensors information can be included, such as a laser range finder, for well known environments, with reference points that can help measure relative distance to certain objects and the robot localizing itself.

In the robot soccer game scenario, such objects as the goal, teammates, opponents and the ball, are well defined objects, and some can be seen as landmarks, therefore used to help the robot localizing itself on the field. In a soccer game the main objective is scoring more goals than the opponent, and a good positioning for the goalkeeper to defend its own goal, in order to avoid the opponent team scoring the least goals possible is of extreme importance, as well as for the striker a better known position of the opponent goalkeeper and goal to have an high accuracy scoring goals. In this thesis it was implemented and introduced in the CAMBADA team, a laser range finder to detect the own goal by the goalkeeper, and to detect the opponent goalkeeper and boundaries of the opponent goal, to maximize the scoring in a penalty kick set piece.

1.4.2 Thesis Structure

In chapter 2, an overview on what an autonomous mobile robot is, a brief example of the sensors that may be a part of it and different methods of localization based on

probabilistic maps is made. Following in chapter 3, the sensors available in the CAMBADA robot are described and a description and study of the laser range finder used for this thesis. In chapter 4, a brief overview of the CAMBADA software is summarized, and an explanation of the implemented method for the goalkeeper localize its own goal, in order to achieve a better positioning when defendig it, and results and comparisons with the previous implemented method achieved with vision process. At chapter 5, we have an ovierview of the `Penalty` role and behaviors of the software CAMBADA, and the improvements made with the introducing of a laser range finder. Finally at chapter 6, it is presented the conclusions of the developed work and obtained results, and future work as well.

Chapter 2

Autonomous Mobile Robots

As said by Ciancarini *et al.*[6], since the 1980s, software agents and multi-agent systems have grown into what is now one of the most active areas of research and development activity in computing in general. This increase of interest belongs to the concept of an agent as an autonomous system, capable of interacting with another agents to meet its own, or a common goal.

Robotics have achieved its greatest success to date in the world of industrial manufacturing. Robot arms, or manipulators, comprise a 2 billion dollar industry [7]. Nowadays, the electronic industry is composed by machines with high precision and speed that allows a mechanization of processes for a production exponentiation and creation of extremely small products, which led us to the possibility of creating mobile phones, laptops and other electronic objects increasingly smaller and lighter. This perk isn't exclusive to the use in electronic industry, but as well in other fields, which makes possible a large series production.

Yet, for all of their successes, these commercial robots suffered from a fundamental disadvantage: lack of mobility. A fixed manipulator has a limited range of motion that depends on where it is bolted down. As stated before, this substancial evolution in electronic field, was due to the industry robotization, and not least, thanks to the existence in the human DNA of a constant search for a bigger knowledge and ambition for always want to increasingly evolve. After the mobility problem solved, doing it autonomously it's the next massive research field.

Agents can be divided in several different categories, being one of those, the one which we will place greater emphasis in this thesis, the autonomous mobile agent. Those agents must be capable of mobility, i.e., equipped with mechanisms for its own locomotion, and

autonomous, or in other words, capable of taking decisions based on its actual state and the envolving environment without the direct human intervention. Thus, they must be reactive and work autonomously and continuously in a dynamic environment, making decisions based on its actual state and the environment it believes that exist, and do actions that can affect the environment, to achieve its goals.

Different goals, require different robot structures, and in Figure 2.1 a variety of autonomous mobile robots are presented.

Nomad from Figure 2.1(a), was a robot vehicle designed at Carnegie Mellon University and NASA Ames to prepare near-term planetary missions. The field test was designed to demonstrate and validate robust locomotion, navigation, visualization and communication in a long-distance, long-duration travel traverse in Chile's Atacama desert to simulate NASA missions on Mars, Moon and Antarctica [8].

In Figure 2.1(b) an Autonomous Underwater Vehicle (AUV) from Massachusetts Institute of Technology capable to map and monitor pockets of the ocean to track the health of fishiries, and survey marine habitats and species [9].

AtlasCAR from Figure 2.1(c) is a prototype for research on Advanced Driver's Assistance Systems from ATLAS project held in University of Aveiro, with the mission to develop and enable the proliferation of advanced sensing and active systems designed for implementation in automobiles and affine platforms [10].

AMIGO from Figure 2.1(d) is a service robot from Tech United. AMIGO stands for Autonomous Mate for IntelliGent Operations and is focussed on the human-robot interaction and cooperation, navigating and mapping in a dynamic environment, computer vision and object recognition under normal light conditions, object manipulation, adpative behaviors, behavior integration, ambient intelligence, standardization and system integration [11].

Much of the challenge is then: how can robots move autonomously through a world who can be always changing?

One of the most important tasks of an autonomous system of any kind is to acquire knowledge about its environment, thus, for the robot to be autonomous, it is necessary that it has some notion of its own structure and the environment it's in, and a continuous actualization of those aspects. In order to achieve it, besides the computer processor unit (CPU), its provided with a wide variety of sensors, and with the data collected from

(a) An autonomous mobile robot from NASA called Nomad[12].

(b) An autonomous underwater vehicle developed in MIT[13].

(c) AtlasCAR, a driverless car from ATLAS project of Aveiro University[10].

(d) AMIGO, a service robot project from Tech United of Eindhoven University[14].

Figure 2.1: Different types of autonomous mobile robots.

them, through the processing of those information, individually or fusing it expecting to obtain better results, he must has good *perception*, i.e., it must obtain the most precise model of the environment where it's located.

2.1 Sensors

Actually exists a large variety of sensors which can be used in the robotic fields, some of them are very simple and used to measure simple values and others more sophisticated that can acquire information from the robot's environment or directly measure its global position. Which some of those types are[15]:

- `Contact Sensors`:

  Those are simple touch sensors that allows to detect the physical contact with objects, and pressure sensors that can be used to determine different levels of

applied strength, but they can have low precision.

- `Distance Sensors`:

  There is a wide range of options in this section, such as, ultrasound sensors, infrared and laser sensors, which provides a big offer at precision and range levels. An example of a distance sensor can be seen at Figure 2.2(b).

- `Position Sensors`:

  Sensors capable of providing an absolute position in an environment, or its relative position to other point. Examples of this type, are the triangulation method, which use a variety of sensors to calculate the absolute position of the robot, and the GPS, which use satellites, but is impracticable on our work due to its inability to locate in indoor environments.

- `Image Sensors`:

  Sensors that detect and convert variable attenuations of light waves to small bursts of current forming an image. Some examples of image sensors are digital cameras, night vision equipment, among others. An example of an image sensor can be seen in Figure 2.2(a).

- `Rotation Sensors`:

  Sensors equipped with a potentiometer or a gyroscope, that in different ways, allows to calculate the angles of rotation from the robot.

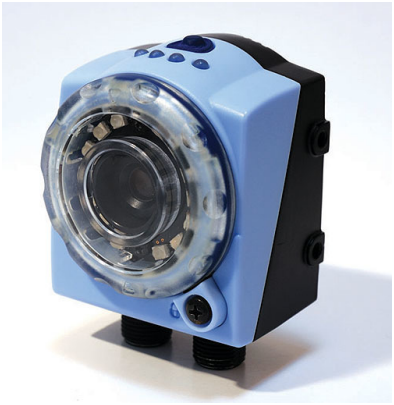
- `Environment Sensors`:

  Here we have a large variety of sensors, such as light, sound, temperature, humidity and pressure sensors, among others, that allows the robot to evaluate the environment where it's located.

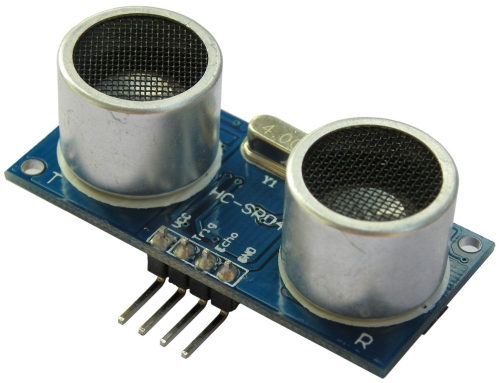
Besides the presented ones, there are other types not mentioned here and they can also be classified by the different types of data acquired and the way they acquire these information. Therefore, they can be classified as intern or extern and as passive or active:

- `Intern`:

  They collect intern robot parameter information such as the charge level of intern bateries, wheels velocity and orientation.



(a) Image sensor[16].



(b) Distance sensor[17].

Figure 2.2: An example of two different types of sensors.

- `Extern`:

  They collect data from the surrounding world, such as the cited above, contact, distance, vision and laser sensors.

- `Passive`:

  Sensors that measure ambient environment energy.

- `Active`:

  Sensors that emit energy into the environment, measuring the echo/response from it.

Individually or with data fusion between at least two sensors, allow us to obtain information that will be processed by the robot in order to calculate and estimate the environment it believes is in.

2.1.1 Active Distance Sensors

As the main goal of this thesis is the goal detection, we will give more emphasis to active distance sensors, because the majority of them directly measures the distance from the sensor to nearby objects, which can be used by the robot to self-location or mapping its environment. There are sensors that use the propagation of sound or electromagnetic waves and their echoes to measure the distance that an obstacle that lies within range where the scan is performed. Thus, the time-of-flight is the principal way in which lasers and ultrasounds are based, and consist in a pulse transmission and in its echo reception. Nevertheless, the quality of a measure relies on several factors, like the sensor precision

measuring of the time interval between the moment a pulse is transmitted and the time its echo arrives, the signal speed on the environment, and the interaction between signal and the object, which can be an absorbent surface or can lead to specular reflections.

As sensors are a fundamental part in the process of *perception*, sensor noise has to be taken in consideration because they reduce the useful information content of sensor readings.

2.2 Localization

After the robot has gathered information from the environment it's in, the next big step is to estimate its own localization, or in other words, it must be able to calculate its relative positions to other objects of interest, and with that information it must be able of self-localization, and it is through localizing on a map that the robot hopes to do so.

In the context of this thesis, it was used a map-based approach for the localization, where the robot explicitly attempts to localize itself by collecting sensor data, then updating some belief about its position with respect to a map of the environment. This kind of approach leads to some advantages as the existance of the map itself represents a mean for communication between human and robot, as the human can simply give a new map whenever the robot changes to a new environment. The risk in here is that the map is a given internal representation of the world, and not the real world itself, and a little divergence from reality can lead to an undesirable behavior from it.

The methods desirably should solve the *kidnapped robot problem*, the situation when an autonomous robot self-location algorithm carries him to an arbitrary location. This problem is a big issue in the robotics localization solutions, and only a few algorithms are able to recover from that failures.

2.2.1 Probabilistic Map-Based Localization

Probabilistic techniques differ from geometric approaches, which doesn't provide any indication of the relative changes between various possible robot positions, as they explicitly identify probabilities with the possible robot positions.

As we are in a static environment, we use a map-based approach, therefore, we use probabilistic techniques. In the following sections we present three methods of proba-

bilistic localization. The first one, *Markov Localization*[18], uses an explicitly specified probability distribution across all posible robot positions. The second method, *Kalman Filter Localization*[19], uses a Gaussian probability density representation of a robot position and scan matching for localization. The third and last one, *Monte-Carlo localization*[20], estimates the position and orientation as it moves and senses given landmarks in the environment.

As the robot moves it can keep track of its motion using odometry, and due to its uncertainty, after a while, the robot will be as well very uncertain about its true position, so it must localize itself in relation to its environment map. The sensors that acquire the information from the environment, and this information has some degree of accuracy, they have some noise associated. Those observations can, therefore, be described as a probability function. So when we use this methods, we are facing a probabilistic problem. So first we try to predict the new position based on the odometry, and then we combine both odometry prediction and the results from the measurements, if they're available, to estimate a belief for the new position.

In the case of the Markov localization, the robot's belief state is represented as separate probability assignements for every possible robot pose in the map, so we must update the probability of every cell. Kalman filter represents the robot's belief state using a single, well-defined Gaussian probability density function, and thus retains a μ and σ parameterization of the robot's belief about position with respect to the map, and, updating this parameters, it's all that's required. Monte Carlo localization uses fast sampling techniques to represent the robot's belief of its position. As the robot moves or senses, importance re-sampling is applied to estimate the posterior distribution.

Markov Localization

Markov localization tessellate the required map into a finite number of possible robot poses in that map, usually into a fixed decomposition grid. Depending on the accuracy wanted, we can divide into smaller or bigger cells, thus the number of possible poses can range from several to millions of positions.

We assume that the environment is static and the robots movement is the only thing that affects the sensors readings, therefore, Markov localization tracks the robot's belief state using an arbitrary probabilty density fucntion, as it transforms a probabilistic belief at time *t-1* into a belief at time *t*. A powerful update machine is required to compute

when new information is incorporated with the previous belief to calculate a new belief state, and as it being a probabilistic belief at each moment, it mantains the probability for every possible pose on the state space.

Figure 2.3 illustrates a simple 1D example presented by S. Thrun *et al.* of the Markov localization algorithm. At Figure 2.3(a), in an initial state, the robot initiates a flat probability density function for its possible location. At Figure 2.3(b) as it encounters a first door, the probabilty function becomes multimodal, as it knows is near a door, but the given map has more than one similar cell with a high probability of success. At Figure 2.3(c), the robot has moved from the door, and still not founded another landmark, thus, its probability function stills multimodal in the most likely positions the robot beliefs its in. In Figure 2.3(d) after found a second door and matchmaking the new information with previous beliefs, the probability function becomes now unimodal and sharply defined. In Figure 2.3(e), the robot has moved again, but has now an unimodal probability function, with high value in its position belief. With this example we can verify that with Markov localization approach, the robot has the ability to localize itself from a initial lost state, and with a higher number of landmarks, the robot can iterate quickly to an unimodal probabilistic function, with high belief value.

Kalman Filter Localization

Robots usually include a large number of sensors, each providing some noise, and optimal localization should take into account the information provided from them all. Kalman filter localization is a powerful technique to achieve this sensor information fusion.

Unlike Markov localization, the Kalman filter approach is based on the assumption that the robot uncertainty, *belief*, can be represented by an unimodal Gaussian distribution, and thus can be represented by its mean and covariance. But first we make the simplifying assumption that the system does not change between the first and second measurement, so it can estimate the mean μ and the covariance σ. Then, the Kalman filter estimates a position by using a form of feedback control: the filter estimates the process state at some time, called the *predictor* equations, which are responsible for projecting forward (in time) the current state and error covariance estimates, and obtain the *a priori* estimates for the next time step, and then obtains feedback in form of (noisy) measurements, called *corrector* equations, responsible for incorporate a new

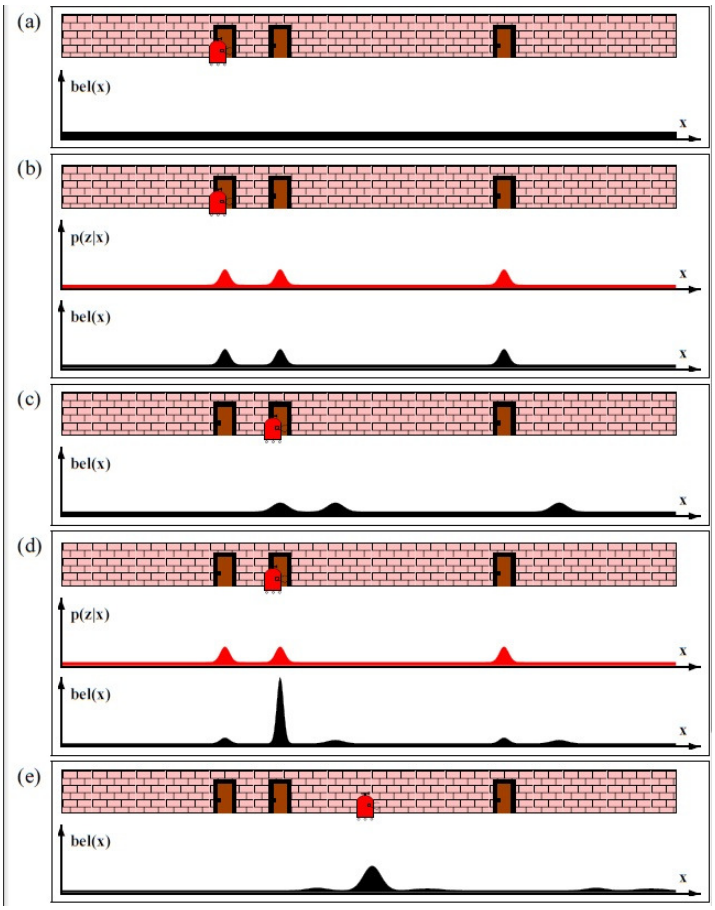


Figure 2.3: Illustration of Markov localization algorithm. Each picture depicts the position of the robot in the hallway and its current belief $bel(x)$. (b) and (d) additionally depict the observation model, which describes the probability of observing a door at the different locations in the hallway[21].

measurement into the *a priori* estimate to obtain a improved *a posteriori* estimate.

To a better understanding, Figure 2.4 shows a well defined environment, where it's given a map of that environment to the robot, represented by a collection of features, called *correspondence variables*, that are necessarily well identified by the robot, which are a condition because the filter work on the assumption of Gaussian measurements. When those conditions are not met, the identity of the landmarks has to be determined during localization, using for example, using the strategie *maximum likelihood*, where one first determines the most likely value of the correspondence variable and then applies the filter, thereby causing the Kalman filter localization can be extended for the global localization problem.

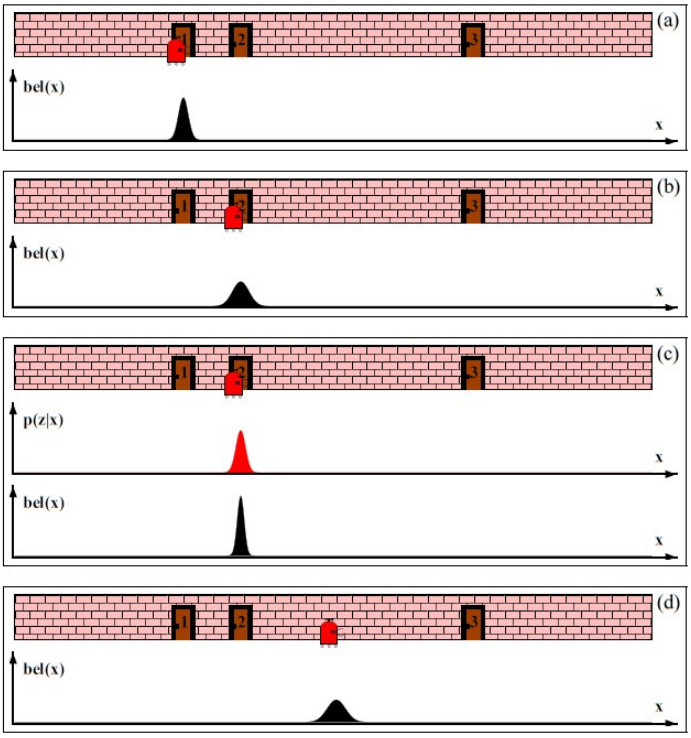


Figure 2.4: Illustration of the application of the Kalman Filter algorithm to mobile robot localization. All densities are represented by unimodal Gaussians[21].

In the 1D example in the Figure 2.4, we make the first assumption that the correspondence variables determined by the doors, have an unique identifier, and a second assumption, needed for the algorithm as stated before, that the initial position is well known, as can be seen in Figure 2.4(a), where the robot is near the door 1, and the position is represented by the Gaussian distribution shown.

The robots starts now moving (Figure 2.4(b)), and its *belief* is now convolving with the Gaussian prediction, resulting in an increase of the Gaussian distribution width, as the uncertainty increases.

In Figure 2.4(c), the robot has now a correspondence from the measurements, resulting in an update of the estimated localization, and its belief. The variance of the resulting Gaussian distribution belief is now smaller, thus by integrating the two independent estimates (*prediction* and *corrector*), it is expected to make a more certain of robot's localization than each of the estimates separetly.

In Figure 2.4(d), similarly to Figure 2.4(b), as the robot moves in the hallway, the robot's uncertainty in its position increases again, since the method continues to incor-

porate motion uncertainty into the robot's *belief*.

Monte Carlo Localization

The Monte Carlo localization approach [20][22] approximates a probability distribution by a set of discrete particles. The initial belief is obtained by randomly generating some pose particles from the prior distribution and assigning to each a importance factor.

The accuracy of the aproximation is determined by the size of the particle set, thus increasing the number of particles, and increasing the accuracy of the aproximation, and its left to the user the trade off of between the accuracy of the computing and the computacional resources necessary to run the algorithm.

Moreover, adding random particles into the particle set is a common feature. By adding those new samples to the probability distribution computed from previous measurements, the Monte Carlo approach is able to quickly recover from a kidnapping situation, as without them, the algorithm would keep focusing on the particles from the old outdated pose, and most likely take a very long time to converge to the correct new pose. This new random particles, usually do not affect the global estimation when it's focused on the correct location, but can be crucial for the case the robot suddenly moves to near one of them.

There are several approaches to compute the robot pose from a sample set [23]. The most common ones are:

- `Overall Averaging`:

  In the perfect scenario, where there are no kidnapping situations and a known initial position, the distribution can be assumed to always be unimodal, and we can compute the average of all samples. Nevertheless, a single outlier can strongly influence the result.

- `Best Particle`:

  In a sample set with weights for each particle, there is a simplist approach, selecting the particle with the best weighting. This approach can lead to problems in globally position, by giving the best weight to a wrong particle.

- `Binning`:

  Dividing the state space into *bins*, and by density extraction, for each bin, the

number of samples are counted. Is computed the average from the samples contained in the bin with more samples. Altough is a robust improvement from the first approach mentioned, the method has its flaws, as the user needs to carefully choose the size and number of bins, because of the inherent choice pros and cons. In addition there are likely to exist instabilities at the edges of the bins.

- `K-means Clustering`:

  Given a sample set of particles and maximum number k of possible clusters, the algorithm iteratively converges to the centers of k samples clusters. This seems to be the best approach to achieve the best results, but there's always need a carefully chosen from the parameters, as the algorithm can become computationally too expensive.

The Figure 2.5 represents an 1D example of the Monte Carlo localization. In Figure 2.5(a) the initial *belief* it's a set of random pose particles uniformly generated for the entire pose space.

In Figure 2.5(b), the robot has now moved and its sensors found a door. The algorithm analyses each particle and assigns it an importance based on the estimated particle, the current observation and the model. The resulting particle set shown, stills identical to the one in Figure 2.5(a), but with different heights for each particle representing the given importance.

This importance is then used for the re-sampling process, that based on the set of the Figure 2.5(b) generates new particles more concentrated around the most likely positions. The importance factor for those new generated particles will be equal as each is one of the possible state, as can be shown at the Figure 2.5(c) after a little motion.

As the robot gets to another point of interest and the sensors find another door, and the measurement assigns new importance heights as illustrated in Figure 2.5(d). At this point, most of cumulative probability mass becomes centered in a point of the map, which is the most likely location, and hopefuly the right one. At Figure 2.5(e) a motion updates the particles again and generates a new re-sampling phase, again with equal importance factor, but more concentrated than from previous Figure 2.5(c).

We can conclude that at each motion step without measurements, the new re-sampling particles tend to disperse, but at each motion step merged with new measurements tend to concentrate the particles around the correct location. Then, using

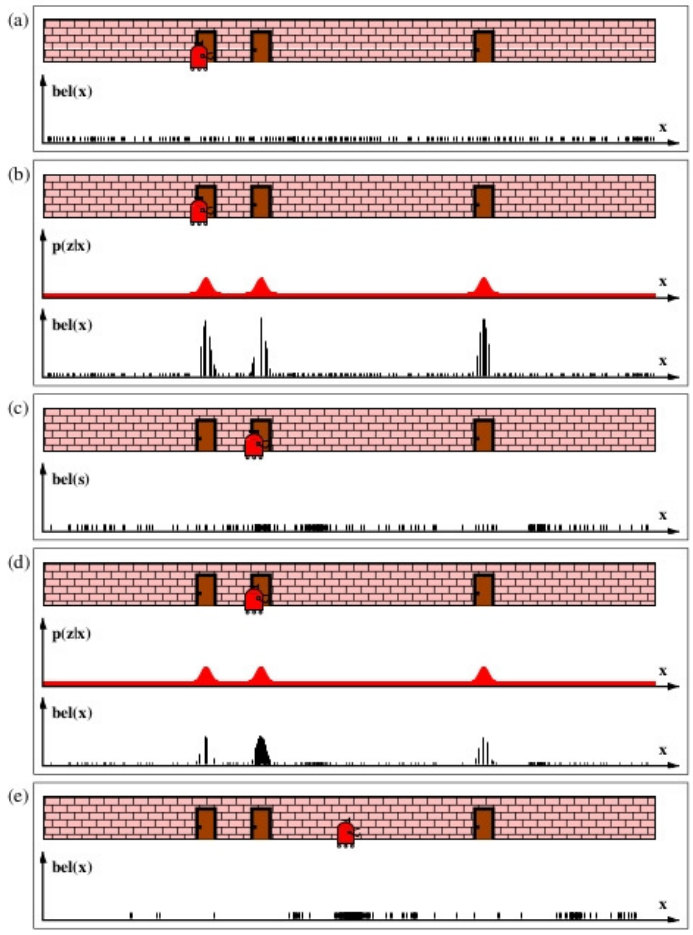


Figure 2.5: Monte Carlo Localization, a particle filter applied to mobile robot localization[21].

one of the above referenced approaches it's computed the *belief* position.

Comparisons

The fundamental difference in the representation of *belief* leads to the following advantages and disadvantages between the three methods:

- `Markov localization`:

  It allows the localization to start at any unknown position, and as well, recover from ambiguous situations like the kidnapped problem, because the robot can track multiple, completely disparate possible positions. However, the need to update all positions within the given state at any time, requires a discrete representation of the given map, such as a grid or topological graph, which can leads to enormous

quantity of memory and computational power, that limits the precision and map size.

- `Kalman Filter localization`:

  Is inherently both precise and efficient, and can be used in continous world representations, although, the need of an initially known position and as the uncertainty of the robot can become too large, and thus not unimodal, the robot can fail to capture the multitude of possible robot positions and irrevocably enter in a kidnapped situation.

- `Monte Carlo localization`:

  Has the advantage not beeing bound to assumptions, like the Kalman Filter which assumes a Gaussian distribution, not needing an initially known position, can recover from kidnapping situations, thus less susceptible to measurements errors, and has an higher accuracy as we choose a large set of particles. On the other hand, this higher accuracy imposes a trade off between higher accuracy and computational weight.

2.2.2 CAMBADA Localization

Created on the scope of robotic soccer, the algorithm implemented in CAMBADA team was first developed by M. Lauer et al.[24] for the Tribots (Neuroinformatics Group, from University of Osnabrck, Germany) robot soccer team competing in the middle-size league. It is based on guided update steps modeling the localization problem as an error minimization task [24], in a well structured environment (in this case, a robotic soccer field), using the field lines. The straight lines of bounding boxes of the field and areas and the curved lines for the center circle and corners arcs, define the given map where the robot need to localize itself and are landmarks than can be detected by visual sensors.

The algorithm was created to work on environments where there are no uniquely identifiable landmarks and relies on a Look Up Table (LUT) built over the map which, for each position, keeps the minimum distance to the closer landmark.

The algorithm starts by assuming a given location as true and estimates the error of that location related to measured distances by comparing with the LUT distances to the existing landmarks, i.e, as a location is given and assumed as true, it will compute the distances from that pose to the landmarks represented by the observation model

acquired indirectly from the visual sensor, in this case, the white points assumed to be field lines (landmarks) and compare them to the ones known for each position in then defined map landmarks, which in this case, points of the complete field lines.

The objective is to compute the best approach to the correct location, i.e, minimize the error, of the match between the acquired landmarks and known ones by gradually correcting the estimation. The location that will be tested in the next step of the process is given by applying a displacement to the current location based on the gradient of the current error, using an algorithm named RPROP [25].

As the study case is a robotic soccer field, where the landmarks are the white lines, which are known *a priori* as they are defined in the competition rules, the LUT of the map it's then based on a representation of the distance for each position on the field to the closer line of the field (landmark) as shown at Figure 2.6.

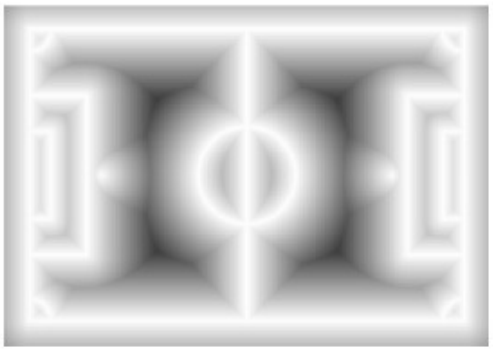
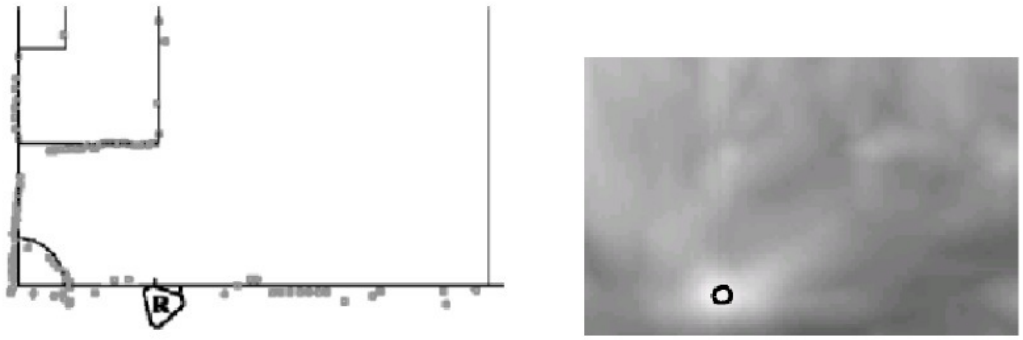


Figure 2.6: Illustration of a LUT map used by the algorithm. Darker areas indicate positions where the distance to the nearest line is large, while brighter areas indicate positions with smaller distances[26].

At a given instant, as stated earlier, the robot gets a visual sensor information, and sees a set of points, which it assumes as white and that they're over the field lines, and consider them as landmarks, which it'll use in the minimization error function. In Figure 2.7(a), the grey points are the set of line points, and the error function estimates a position by estimating the error and error gradient on the current position comparing to the information given from the LUT and iteratively try new positions in the direction of the least error gradient until an acceptable position with a low error is found. The Figure 2.7(b) plots the error function on a grey scale, being brighter areas the zones with smaller errors and the dark ones the zones with larger errors. Finally the most probable

position estimated (the one with lowest error) is given by the black circle.



(a) Solid black lines are field lines and gray circles are landmark points used for the localization estimate. "R" is the robot belief for is position.

(b) Brighter areas indicate positions with smaller errors and darker areas the ones with larger errors.

Figure 2.7: On (a) the line points for a localization estimate and on (b) the corresponding error function plot[24].

In this chapter, it was highlighted the autonomous mobile robot, the fact that to achieve the expected goals, for each specific situation, it must be provided with several types of sensors, and the importance of visual and distance sensors to interact with the surrounding world, allowing the robot to interact with it. Some localization methods were introduced, that using the information acquired from those sensors, enable the robot to localize itself. One of the contributions to this thesis was the inclusion of a new distance sensor, summarized in the next chapter, and adapting this algorithm to localize the robot on the soccer field based on these new information acquired.

Chapter 3

Adding a Laser Range Finder (LRF) to the CAMBADA Robots

As stated before, the sensors are an important and crucial part of the robots hardware, used for motion, localization and performing almost every task. In this thesis, it was introduced a new sensor to the existing ones on CAMBADA robots, the *laser range finder* (*lrf*), studied and implemented, and it is used for object detection. In this chapter we have a brief description of the main sensors from the CAMBADA robot, the laser operation and the specific laser used in the thesis.

3.1 The CAMBADA hardware

The CAMBADA platform was built with ease of transportation, robustness and modularity in mind. A geometric solution with an asymmetrical hexagon shape was chosen. In this section, a brief overview of the CAMBADA robot is made, and a wider explanation can be found in [27].

Figure 3.1 presents the built in-house CAMBADA robot. At the top of the robot, we have the vision system implemented, that gives an omnidirectional view, which consists on a catadioptric set with an IDS Gigabit Ethernet camera pointing upwards to an hyperbolic mirror. Before using it, a calibration is required, namely: the camera's colormetric parameters, the inverse distance map computation, the definition of the region of interest in the image that has to be processed, and the color ranges for object segmentation [28]. The detection of objects is currently a mixture of color-based and shaped-based approaches. For the first one, the calibration is done in the HSV (Hue,

Saturation and Value) color space.

Figure 3.1: Current robot used in CAMBADA team. At the top it's the vision system, at the center the main processor unit. At the bottom, on the center it's the grabbers and kicker, and at the sides, are the wheels belonging to the locomotion hardware.

A little lower, the main processing unit can be seen and consists in a typical laptop, running a lighweight Linux distribution. It is responsible from receiving and processing the information coming from sensorial data from the hardware, executing high-level coordination and decision algorithms and producing signals for the low-level layer. This unit is also resposible for the robots inter-communication handling.

On the bottom of the robot, in the center circle, is the ball handling mechanism, also known as "grabber", and the ball kicking mechanism, also known as "kicker". The grabber is based on a double active handler. It relies on two DC motors that control the ball's surface velocity on two points. The kicker, responsible for making lob-shots and passes, is a electromagnetic kicker whose main element is an electromechanical solenoid.

Yet at the bottom of the robot, in the two small circles on the sides, is presented the motion hardware, omni-directional wheels based on an aluminum 3-piece sandwich structure. The motion function is accomplished by using three different nodes (micro-controllers), one per motor/encoder pair, that are responsible for driving the motors and

reading encoder data at the same time.

The low-level layer sensing and actuating system consists in a series of microcontrollers inter-connected with a CAN (Controlled Area Network) bus. Each microcontroller is responsible for certain tasks, as seen in the motion case, and the communication between the Main Processor Unit and the Low-Level Bus is done via a USB bi-directional gateway.

3.2 The Laser Sensor

As the *lrf* is the main sensor used in this thesis, a basic description of the technology will be given.

Laser stands for **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation. An excited atom returns to a lower energy state by either spontaneously radiating a photon or being stimulated to do so by electromagnetic radition, and when a photon is emitted by stimulation, it goes in same direction and phase as the stimulating radition.

This electromagnetic radition has to be of the right frequency, the one that corresponds to the difference in the energy between the excited state and the lower state it goes to by emitting a photon, as an emitted photon can cause an avalanche effect by triggering other atoms to emit photons, and as all atoms seek to stay in a energy state as low as possible, few atoms are in an excited state under normal circunstances, as if that would happen, the radition would instead be absorbed and photons would spontaneously be emitted, in which case, they would be sent out in random directions and random phase.

So the idea in a laser is to excite atoms to a much higher percentage than the normal, so the stimulated emission of photons will be the dominating effect, and the wave/energy transmitted forms the laser beam[29].

With applications in so many areas, using it to measure distances is one of them, and a *lrf* is a device that uses a laser beam to measure the distance to an object giving an overview of the surrounding area with milimeter precision. In Figure 3.2 is shown the *lrf* URG-04LX-UG01 used in this thesis, and a more detailed description will be made later in this chapter.

Figure 3.2: Laser Range Finder URG-04LX-UG01 used in the work.

3.2.1 Operation

The dominating techniques for laser based range measurements are time of flight techniques and phase-shift techniques.

Time of Flight Technique (TOF)

TOF makes use of the propagation speed of sound or an electromagnetic wave, and a short pulse is sent out in the direction of an object and the time until it's reflected and return that is measured as can be seen in Figure 3.3. In general, the travel distance is given by:

$$d = \frac{1}{2} \times c \times t \tag{3.1}$$

where

d = distance from the laser to object

c = speed of wave propagation (in the laser case is the speed of light)

t = time of flight (round trip)

To implement such system, high precision resources for measuring time are needed, since the speed of an electromagnetic signal is $3.10^8 m/s$, thus evident that measuring time of flight for such lower distances is more technologically challenging and only recently become affordable and robust for use on mobile robots.

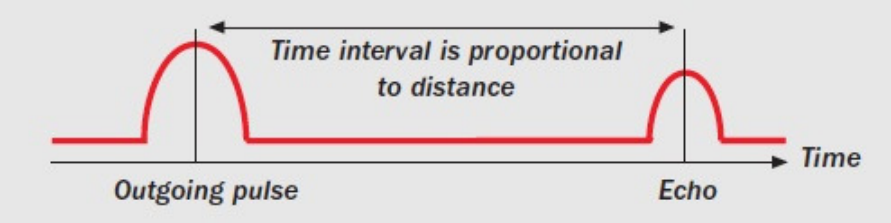
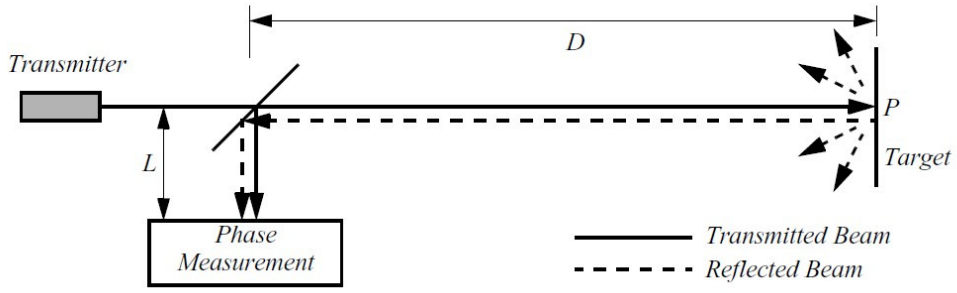


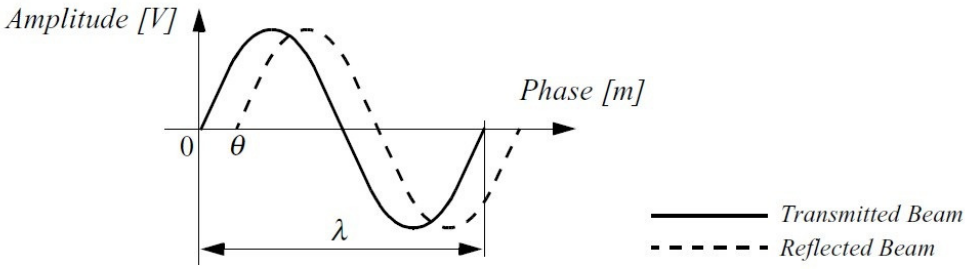
Figure 3.3: Example of the functioning of Time of Flight[7].

Phase-Shift Technique

In phase-shift systems a continous modulated wave is transmitted, and will hit a point P in the environment as can be shown in figure 3.4(a). The idea is to compare the phase shift between the returned signal with a reference signal generated by the same source. Using the Doppler shift, the velocity of the target, if wanted, can also be measured.



(a) Schematic of laser rangefinding by phase-shift measurement.



(b) Range estimation by measuring the phase shift between transmitted and and received signal.

Figure 3.4: The fuctioning of Shift-Phase System[7].

Figure 3.4(b) shows how this technique can be used to measure distances. The

wavelength of the signal obeys to the equation $c = f \times \lambda$ where c is the speed of light, f the frequency of the signal and λ is the wavelength.

To measure the distance to the object and the total distance D' covered by the emitted light is given by

$$D' = L + 2D = L + \frac{\theta}{2\pi} \times \lambda \tag{3.2}$$

where D and L are the distances defined in figure 3.4(a). The wanted distance D, between laser and target, is therefore given by

$$D = \frac{\lambda}{4\pi} \times \theta \tag{3.3}$$

where θ is the electronically measured phase difference referenced before, and λ the known modulating wavelength.

The *lrf* makes a scan that maps all objects in its proximity, and with laser sensors, it is possible to make quick measurements with reduced line of sight. When we make a scan, after each read, the sensor moves a fraction from its angular resolution and makes a new read, repeting, constinously, the process until it reaches the end of its range detection. As we can see in Figure 3.5, the *lrf* has a non-radiated area called dead zone in which it doens't take any measures.

This kind of lasers usualy are characterized by its resolution, frequency and angular resolution.

Resolution

The resolution or precision, is the maximum error that the laser has for each position in a certain direction.

Frequency

Indicates the number of times in each second that the laser does the scan for its scanning range.

Angular resolution

Indicates the number of samples that the laser can obtain in its scanning range.

The sensor has a minimum and maximum range, being the minimum in the milimeters order and the maximum in the meters or tens of meters order.

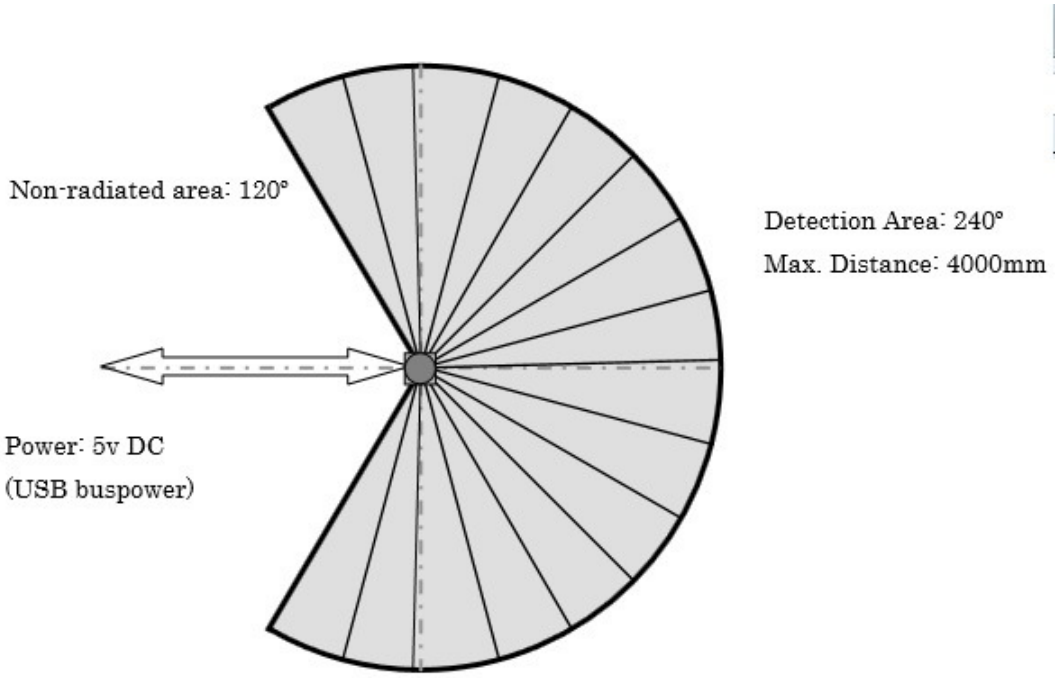


Figure 3.5: Detectable area from laser.

As the used *lrf* is based on Shift-Phase technique, the measurement quality and success depends on various factors, such as:

- interaction with the target (e.g., surface absorption, specular reflections);
- variation of propagation speed;
- the speed of the mobile robot and target (in the case of a dynamic target).

Inside the laser exists a spinning disk which changes the laser measurement direction, allowing all angular space to be measured. Data collected is then processed by a circuit in the laser, packaged and sent through communication ports connected to a larger processing unit like a laptop.

3.3 URG-04LX-UG01

There are many laser manufacturers that are different in their range, performance, energy consumption and price. For this thesis we used the *lrf* sensor URG-04LX-UG01 from Figure 3.2 that has a detectable area as shown in Figure 3.5 where the laser is the

center circle in the figure and the detectable area the incomplete grey circle involving the laser.

The light source consists in an infrared laser of wavelenght $785nm$ with laser class 1 safety. Its scan area is 240° semicircle with maximum radius range of $4000mm$. Angular resolution is approximately 0.36° and sensor outputs the distance measured at every different point (683 steps). Laser beam diameter is less than $20mm$ at $2000mm$ with maximum divergence $40mm$ at $4000mm$. The principle of distance measurement is based on calculation of the phase difference, due to which it is possible to obtain stable measurement with minimum influence from object's color and reflectance.

The sensor is equipped with RS232C and/or USB for interfacing with an external device, and communication can be done via any one of these interfacing channels. Drivers or packages needed to communicate can be found at company website in [30].

Figure 3.6 shows some parameters that can be useful and important when reading the measurement data from sensor. The scanner rotates in an anti-clockwise direction when viewed from top. Detection Range (E) it's the maximum angle the sensor scans for measurement and Angular Resolution is defined as the 360° divided by the Slit Division (F). Table 3.1 shows the measurement parameters of the sensor URG-04LX-UG01 model.

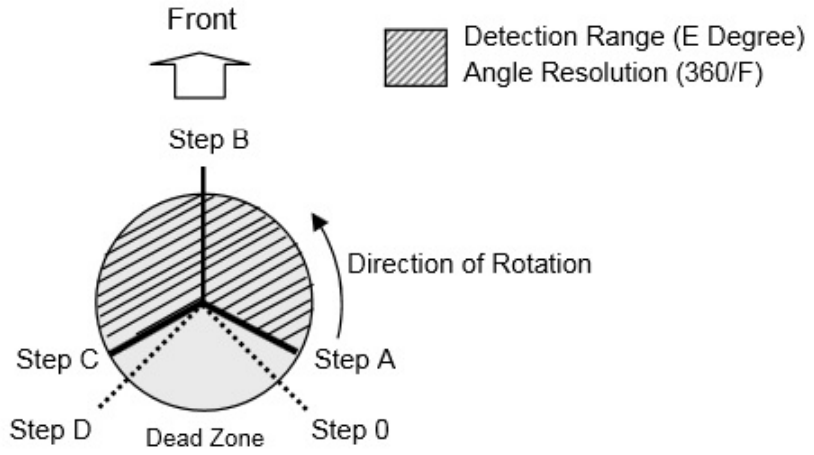


Figure 3.6: Measurement parameters for Hokuyo laser models.

In Figure 3.7(a) it is possible to see an image obtained from an interface tool created by J. Silva[26]. After the scan, the algorithm knows the angle for each step and distance measured and it's possible to create a 2D map of the surrounding area observed by the *lrf* corresponding to the one in Figure 3.7(b). Comparing both images from Figure 3.7,

Step 0	First Measurement Step	0
Step A	Initial Measurement Step of Detection Range	44
Step B	Sensor Front Step	384
Step C	End point of Detection Range	725
Step D	Last Measurement Point	768
E	Detection Range	239.77°
F	Slit Division	1024

Table 3.1: Measurement Parameters of Sensor Model URG-04LX-UG01

it is possible to see an open door at the left, an open hall in front-left and the relief of the machine in the front-right.

3.3.1 Data validation

In order to validate the specifications given by the manufacturer in its website [30], several measures have been acquired in a well known environment.

In Table 3.2, three measures for each distance, for several distances were made, in order to compare the theorical and practical accuracy of the measurements. In the laser manufacturer specifications, it's said that for measured objects in a distance interval from $20mm$ to $1000mm$ it has an accuracy of $\pm 30mm$ and for distances between $1000mm$ and $4000mm$ it has an accuracy of $\pm 3\%$, and with $1mm$ resolution. Analyzing the obtained results for the first interval, $20mm$ to $1000mm$, the maximum error was nearly $25mm$, which is near the maximum theorical error, but lower so we can say that the *lrf* has reliable measurement in that range. For the interval $1000mm$ to $4000mm$, although it was expectable an increase in the error as the distance to the object increases, maximum errors are all below the maximum given by the manufacturer and the increase in the error with the increasing distance to the object isn't so significant as expected, as the measures are quite reliable to real distances.

In Table 3.3, for each number of scans made, the time interval taken to perform the task was acquired, and according to the manufacturer each scan requires $100ms$. Analyzing the results, we observed that for any number of scans required, all of them took around an extra of $126ms$ to perform the task, which we can conclude that another operation is performed by the *lrf*, and the scans take approximately $100ms$ each as

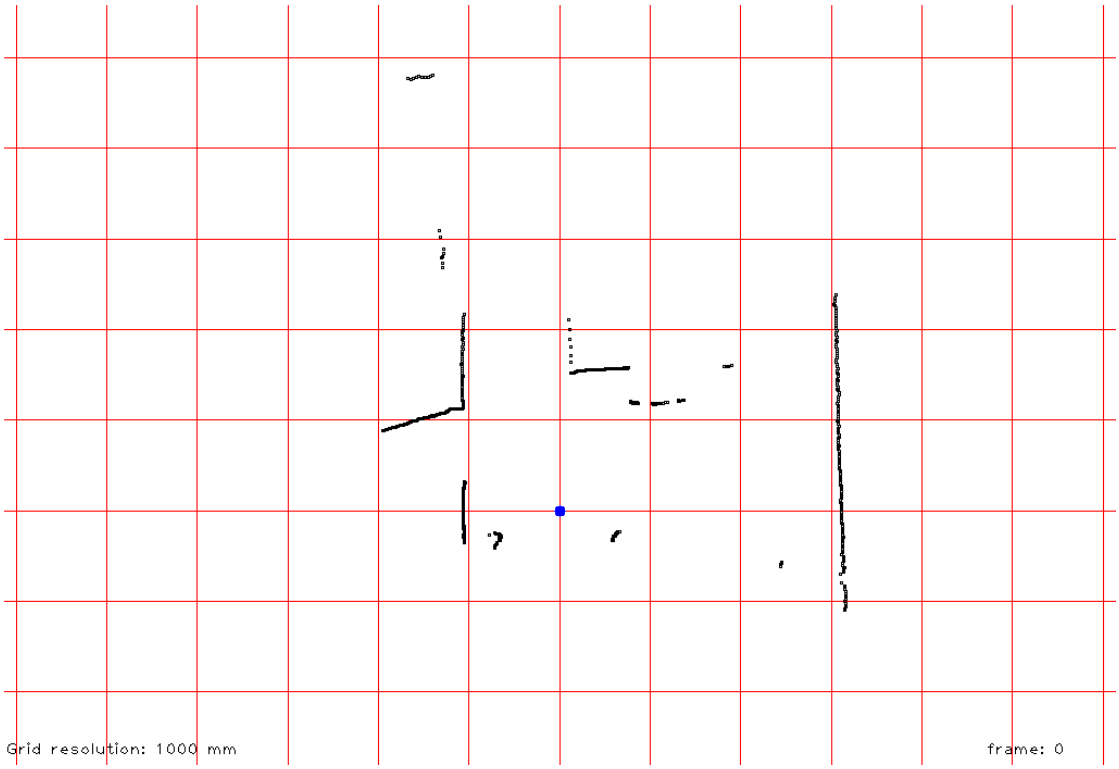
indicated by the manufacturer.

Real Measure	Measure 1	Measure 2	Measure 3	Mean	Error	Max. error
50	49	44	50	47,67	2,33	30
100	100	97	100	99	1	30
150	136	135	130	133,67	16,33	30
200	189	192	193	191,33	8,67	30
250	249	246	239	244,67	5,33	30
300	317	306	311	311,33	11,33	30
350	363	363	367	364,33	14,33	30
400	423	424	418	421,67	21,67	30
450	461	456	461	459,33	9,33	30
500	505	511	511	509	9	30
550	563	555	564	560,67	10,67	30
600	627	624	623	624,67	24,67	30
650	670	668	663	667	17	30
700	715	718	719	717,33	17,33	30
750	761	759	760	760	10	30
800	810	811	811	810,67	10,67	30
850	858	862	859	859,67	9,67	30
900	919	915	916	916,67	16,67	30
950	956	958	960	958	8	30
1000	1009	1010	1011	1010	10	30
1100	1108	1107	1102	1105,67	5,67	33
1200	1218	1205	1209	1210,67	10,67	36
1300	1316	1317	1318	1317	17	39
1400	1421	1422	1422	1421,67	21,67	42
1500	1513	1510	1517	1513,33	13,33	45
1600	1619	1619	1623	1620,33	20,33	48
1700	1720	1723	1724	1722,33	22,33	51
1800	1824	1815	1825	1821,33	21,33	54
1900	1927	1931	1927	1928,33	28,33	57
2000	2017	2019	2016	2017,33	17,33	60
2200	2213	2206	2217	2212	12	66
2400	2401	2401	2404	2402	2	72
2600	2589	2591	2593	2591	9	78
2800	2785	2780	2777	2780,67	19,33	84
3000	2997	2999	2999	2998,33	1,67	90
3200	3208	3206	3210	3208	8	96
3400	3408	3412	3414	3411,33	11,33	102
3600	3638	3630	3826	3631,33	31,33	108
3800	3819	3830	3826	3825	25	114
4000	4002	4005	4006	4004,33	4,33	120
4400	4435	4432	4434	4433,67	33,67	132
4700	4732	4737	4735	4734,67	34,67	141
5000	5013	5021	5009	5014,33	14,33	150

Table 3.2: Practical distances acquired by the *lrf* and the error between practical and real distances. All values are in *milimeters*.

Number of scans	Measure 1	Measure 2	Measure 3	Mean	Teorical Time
1	0,225255	0,225436	0,224539	0,225	0,100
2	0,325939	0,325833	0,325709	0,326	0,200
3	0,426126	0,426181	0,426124	0,426	0,300
4	0,526576	0,526616	0,526340	0,527	0,400
5	0,626736	0,627041	0,627023	0,627	0,500
6	0,727242	0,727256	0,727369	0,727	0,600
7	0,828339	0,827903	0,827718	0,828	0,700
8	0,927571	0,927857	0,927627	0,928	0,800
9	1,02835	1,02833	1,02834	1,028	0,900
10	1,12873	1,12885	1,12807	1,129	1,000

Table 3.3: Time taken to realize a desired number of scans. All values are in *seconds*.



(a) Capture from URG-04LX-UG01 from a surrounding area.

(b) Real image corresponding to the obtained capture from the *lrf*.

Figure 3.7: Representation of a capture with *lrf* URG-04LX-UG01 and corresponding real world environment.

Chapter 4

Goal Detection Using the LRF

Moving through the environment is one of the most challenging skills required of a mobile robot, as success in navigation implies success at the four building blocks of navigation[7]: *perception*, the robot must interpret its sensors to extract meaningful data; *localization*, the robot must determine its position in the environment; *cognition*, the robot must decide how to act to achieve its goals; and *motion control*, the robot must modulate its motor outputs to achieve the desired trajectory. Of these four components, localization has received the greatest research attention in the past decade and, as result, significant advances have been made on this front.

Localization algorithms try to estimate the best fitting position and orientation of a robot in the field, but they work in an absolute base. The CAMBADA team uses two different coordinate systems for different purposes. Absolute coordinates are used for localizing themselves and other objects of interest in respect to the field, where origin resides in the fields center, the y-axis points to the direction of the opponents goal and x-axis points to the right side of y-axis. On the other hand, the relative coordinate system has its origin on the center of the robot, its y-axis grows to the front of the robot and x-axis to the right side of the robot as can be seen in Figure 4.1.

As the goalkeeper main objective is to prevent the team from suffering the least possible goals, in defensive situations it must has a correct positioning regarding its goal and the opponent striker. In order to do that it must has the best perception of his location in the field, and as important, where its own goal is. In the actual positioning from the CAMBADA's goalkeeper in defensive moments, it assumes that the center of the goal is the middle of the goal line, which in to CAMBADA absolute coordinate system and for the standard field is $(0, -9)$.

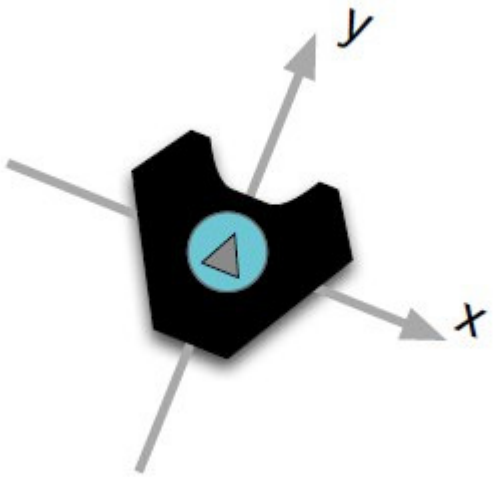


Figure 4.1: The relative coordinate system used by CAMBADA[27].

In a game, it's necessary to avoid bad positioning in defensive moments, as the goal isn't attached to the field, and can be displaced from what the robot beliefs are, which can cause a malposition of the goalkeeper. In Figure 4.2, two possible situations of the stated problem are visible. In the Figure 4.2(a), in a penalty situation, where the ball is located on the penalty marker, the goal center is shifted to the left, and consequently, as the goalkeeper isn't aware of that, it is located on the position of the field it believes it's the goal center. In Figure 4.2(b) the goal continues with the same shift, and the ball is now located far than the left post. In this situation, the goalkeeper closes the angle by defending leaning against the left post. As seen in the image, by not being aware of the misplacement of the goal, there is a gap between the robot and the left post that shouldn't exist.

Using the *lrf* sensor presented and described in the previous chapter, an alternative localization method based on the relative position of the goalkeeper to the goal was implemented, incorporated on the robots *beliefs* and tested, expecting to prevent those failures.

(a) Ball positioned in the penalty marker and the goalkeeper defendig what it's goal center beliefs are.

(b) Ball positioned to the left of the goalie and bad positioning from the goalkeeper, which should be near the left post.

Figure 4.2: Two goalie defensive positions with the goal slightly shifted to the left.

4.1 The CAMBADA Software Architecture

The general architecture of the CAMBADA robots is described in [31]. The robots follow a biomorphic paradigm, each being centered on a main processing unit, the *brain*, which is responsible for the higher-level coordination, i.e. the coordination layer. This main processing unit handles external communication with the other robots and has high bandwidth sensors, typically vision, directly attached to it. Finally, this unit receives low bandwidth sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system, the *nervous system*.

4.1.1 The Realtime Data Base

The robot main process unit is responsible for a set of processes, while reading information coming from other sensors in the platform, and being part of a multi-agent system, communicates with teammates and coach via Wi-Fi IEEE 802.11a and 802.11b standards, and this communication protocol supports the **R**eal**t**ime **D**ata **B**ase (RtDB) [32], which is a middleware that provides a seamless access to the complete team state using a distributed database, partially replicated to all team members, as it's depicted on Figure 4.3.

The RtDB besides being used by the robots to share information, like absolute positons and current conditions, it is also intensively used as an inter-process communication mean, due to is capability of defining local memory items that are not broadcasted to other robots, but are still accessible by other processes running ot the same agent en-

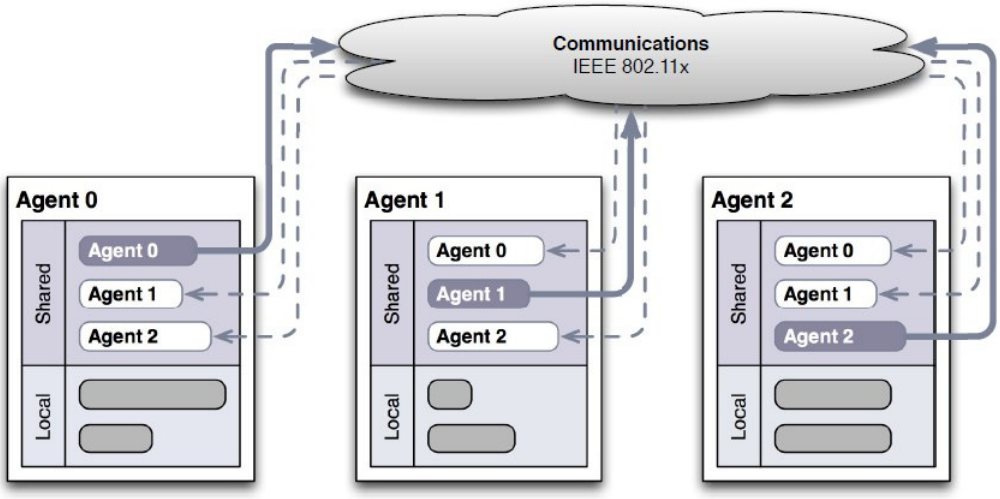


Figure 4.3: A RtDB setup example with three agents[27].

vironment. This local information includes sensorial data from the vision system and the hardware platform and also commands that are sent to the low-level control layer through a gateway interface. The following processes: *monitor*, *vision*, *hwcomm*, *agent* and *comm* are the processes running on each robot and basestation, the *basestation* which it's an application used to vizualise and control the robots state remotely, and the software *coach*, responsible for controlling the team basic strategy. A brief description of each can be found at [27].

4.1.2 WorldState (WS) for robotic soccer

A representation of the world state is a need in almost every scenario with autonomous robots and robotic soccer is no exception. When building the representation of the surroundings of a robot in the robotic soccer scenario, basic entities will exist and must be considered in the deliberation process in order to robots be able to play. In a general overview of the WS entities, there are some mandatory elements in the world representation, such as the ball position and velocity, the position of teammates and opponents, the boundaries of the field and its internal state.

The WS in CAMBADA architecture is a module that contains a data structure called **Robot** that holds all necessary data to characterize the robot in game, such as [26]:

- **ID**, the number that identifies itself.

- **Role**, the role that the robot currently it's in.
- **Behavior**, the current action whithin the role the robot decided to take with the current conditions.
- **Colors**, both team and goal (since we use a color name to distinguish the field side).
- **Position**, which is the 2D ground position of the robot represented on the field axis and the angle it has, measured from the opponent goal.
- **Velocity**, a representation of the robot linear velocity on the ground plane and its angular velocity, measured around itself.
- **Internal info**, a set of flags and values used for decision making and management of the robot internal state, such as flags that mark the robot running, having a role remotely defined or being coached. As for values, usually some coordination values and points are present in the robot structure, to allow communication of decisions that need cooperation.
- **Ball**, a data structure with some basic attributes for the ball, which include its position on the ground both relative to the robot and in the field absolute coordinates, its velocity and a boolean flag of visibility, and another for either it is or is not ball engaged on the robot.
- **Obstacles**, a sublist of obstacles visually detected by the robot. Each obstacle is represented by its position on the ground and an ID, which is the teammate ID if it was identified as a teammate or another value outside the team IDs.

The WS structure is updated each cycle of the agent with the new information that becomes available.

4.1.3 Integration process

On the running software architecture, the first step on each cycle is the integration of all available information and its management. The integration process executes several tasks with the information gathered from the several sources. The integration itself is divided into several tasks executed in a specific order, as for most of them, this order is important, as later tasks depend on the data produced by earlier tasks.

Figure 4.4 represents the tasks performed by the integrator. The initial step is to read the information available in the fields of the RtDB, as vision information, robot platform information and shared information.

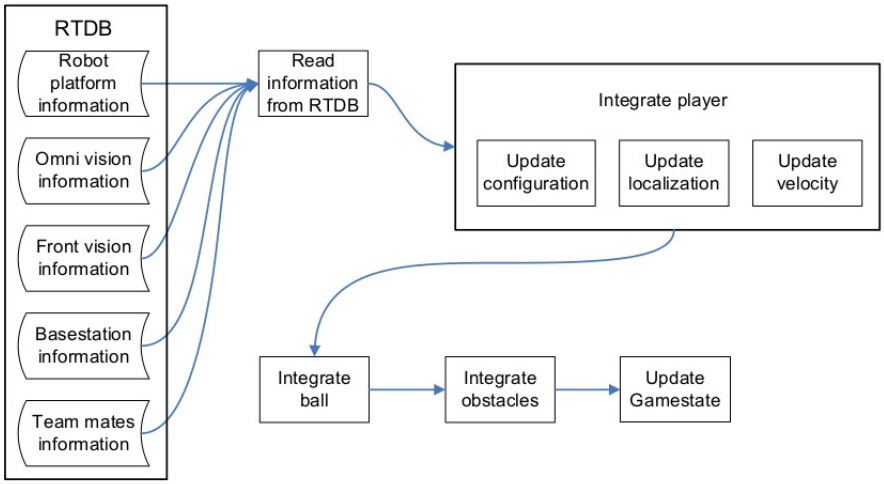


Figure 4.4: Integration task sequence diagram[26].

Following, an update of the robot configuration, localization and velocity is made. The acquired data from the vision is now used in the robot localization method to estimate the new position on the field. Using this new position and previous positions it estimates its velocity.

After knowing its own position and velocity, the robot is able to estimate the positions of the field objects as the ball and obstacles (teammates position are integrated in the updating of its configuration).

After all integrations, a game state update based on the new information and the referee signal is executed.

4.1.4 Behaviors

In the 1st International RoboCup MSL Workshop, held in Kassel in November 2008, the Brainstormers Tribots presented their Behavior-Based approach [33][34]. Their implementation besides combining two different architectures [35][36], included an interface extension to the behaviors, with two additional conditions, the **Invocation Condition** (IC) which specifies the conditions that will start the behavior, and the **Commitment Condition** (CC) which are the conditions to keep in the behavior. In the CAMBADA

team architecture, the Behavior interface is composed by the calculate method and two other methods to verify the IC and CC. Thus, a behavior can only be selected if the IC are true, and only remains in such behavior if it's suitable for the situation which is verified by the CC.

Thence, beyond their specif behavior methods, all derived behaviors need to implement the following methods from the Behavior base class [27]:

- `virtual void calculate(DriveVector *dv)`:
  The main block of the Behaviors, where it calculates a DriveVector with the desired velocities and the kicker and grabber states.

- `virtual bool checkIC()`:
  The derived Behaviors have to implement this method to define the Invocation Condition inside.

- `virtual bool checkCC()`:
  The derived Behaviors have to implement this method to define the Commitment Conditions inside.

- `virtual void gainControl()`:
  The derived Behaviors can implement this method to be notified when they gain control. The callback can be used, for instance, to initialize variables.

- `virtual void loseControl()`:
  The derived Behaviors can implement this method to be notified when they lose control to other Behavior. The callback can be used, for instance, to clean up variables.

A list of the behaviors that are implemented in the CAMBADA team is presented below.

Ball Handling Behaviors

These behaviors allows the robot to decide, under certain conditions, how to handle with the ball. The implemented behaviors are: `BBallBodyProtect`, `BKickToTheirGoal`, `BReceiveBall`, `BReleaseBall` and `BRelieveBall`.

General Behaviors

Are general behaviors that usually are triggered overwriting other Behaviors. The implemented behaviors are: `BAvoidTheirGoalArea`, `BStop` and `BStopRobotGS`.

Goalie Behaviors

As the name sugests, these are the implemented behaviors to the goalkeeper. The implemented behaviors are: `BGoalieDefend`, `BGoalieDefendHigh`, `BGoalieGoToGoal`, `BGoalieKickAway`, `BGoaliePenalty`.

Midfielder Behaviors

Implemented behaviors specificaly to robots assigned with the midfielder role. The implemented behaviors are: `BGoToStrategicPosition`, `BMidLineBlock` and `BMidTransitionCover`.

Penalty Behavior

Implemented behaviors exclusively to the penalty set piece. The implemented behaviors are: `BPenaltyGoToVisibleBall`, `BPenaltyKick`, `BPenaltyMoveFront`, `BPenaltyPosition`.

Replacer Behavior

Implemented behaviors specificaly to robots assigned with replacer role. The implemented behaviors are: `BReplacerAlign`, `BReplacerBallPassedStop`, `BReplacerPass` and `BReplacerPos`.

Striker Behavior

Implemented behaviors specificaly to robots assigned with striker role. The implemented behaviors are: `BStrikerContourOpponent`, `BStrikerGoToBall`, `BStrikerOurFieldDribble`, `BStrikerOurFieldPass`, `BStrikerPass` and `BStrikerRelieveBall`.

Without Ball Behaviors

These are the implemented behaviors when the robot doens't has the ball, whether the opponent team has the ball or not, and whether they are in gameplay or not. The implemented behaviors are: `BActiveInterception`, `BAvoidOutField`, `BBarrier`, `BCornerTouch`, `BGoToBall`, `BGoToVisibleBall`, `BParking`, `BReceiverPosition`, `BSearchBall`, `BSearchBallBarrier`, `BTaxi` and `BTour`.

There are other implemented behaviors such as Kick Calib, Test and Utils.

To decide which behavior to take, and as the robot in MSL it is in a dynamic environment, it was developed an hybrid agent. A reactive component able to react to fast changes in the environment (ball and opponent team) and a deliberative part, the **arbitrator**, the module responsible for selecting a certain behavior in a cycle, whithin the candidates arranged in a priority queue, which it's the most fitting one, based on their IC and CC conditions.

4.2 Localization with LRF

To include the sensorial information acquired by the *lrf* a new process named **LRF** was created, which is the process responsible to acquire the raw data from the *lrf*, the angular measurement, that will be stored at the local RtDB item for the agent to use.

4.2.1 LRF Integration Process

In the Figure 4.5 a brief description of the *lrf* data integration is shown. First, it's readed the lrf data field available in the RtDB and it's integrated in the agent process for further localization update.

For the specific case of the **LRF** data acquired, the available data stored in the item, will be integrated as shown in the schematic in Figure 4.6.

As shown in the Figure 4.5, in the agent process side, before reading the stored data in the RtDB, a time validation is made, i.e., if the present data was stored at 100 cycles or more, it is discarded. After reading the data, it will convert it from angular measurement points to 2D axis vectors, as can be seen, and saved in an array of *lrf* vector points. The *lrf* lines validation step is made by counting the number of vectors, because in the previous step, only angular points measured within the *lrf* range are converted to 2D axis, thus to this validation suceeds, an high number of points needs to be filled in that vector array. Inserting a single vector, it's the used method to make the time validation, by making it fail at data validation. The following step it's the robot localization according to the *lrf* data. Then it's introduced the two new fields in the robot WS: the **robot lrf field coordinates** which are the absolute field coordinates calculated that will be used for future localizations based on *lrf*, and the **goal center**

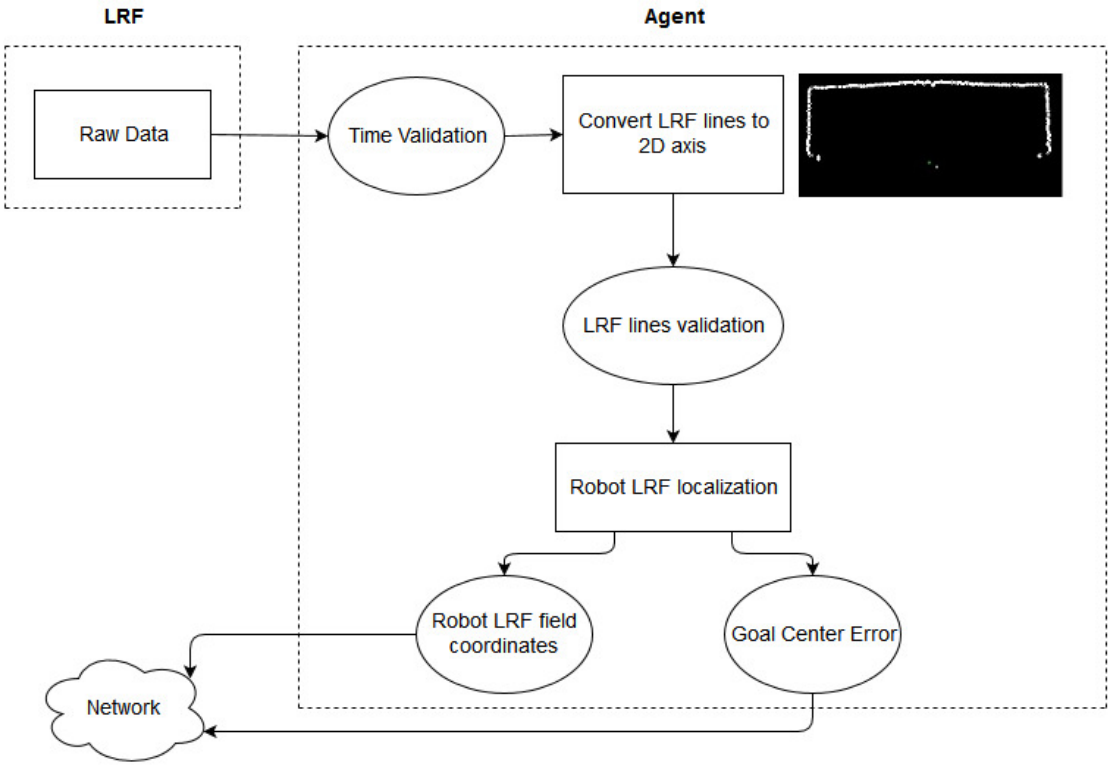


Figure 4.5: Diagram of LRF information fusion task performed by the soccer agent on CAMBADA.

error which it is the error between the goal real position according to the data acquired and the standard position $(0, -9)[m]$. This error is obtained with the difference between the robot *lrf* absolute field coordinates and the robot vision absolute field coordinates. If the robot *lrf* field coordinates are $(0, 0)$ we use the robot vision field coordinates for the first localization.

4.2.2 Mapping

As CAMBADA team uses the Tribots algorithm for localization, a map with landmarks as the one in figure 2.6 is required, and to create such a map, previous dimensions knowledge of the goal it's required. The goal can be approximated to 2D lines, 2 side lines and the goal bottom line. The map will be a representation of the measured distances from each cell of the field map to those lines, i.e., the minimum distance for each possible position in the field to every line points.

As the map will be a distance matrix, Figure 4.7 is a image representation in a gray scale, i.e., knowing the distance in each cell, will be assigned a corresponding value

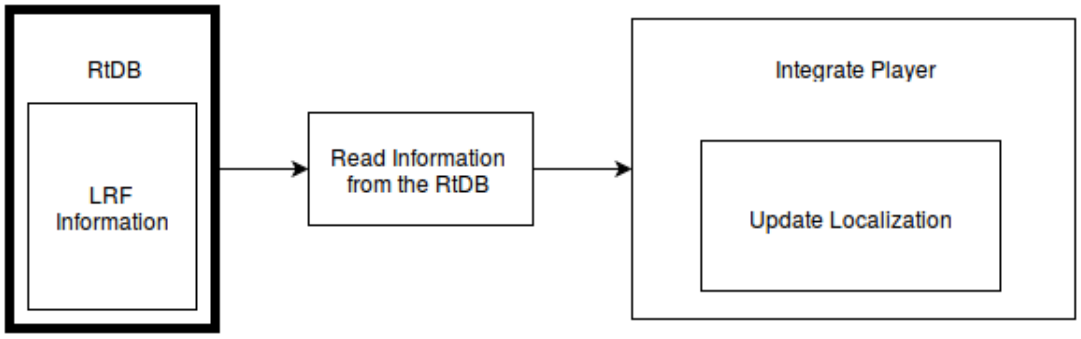
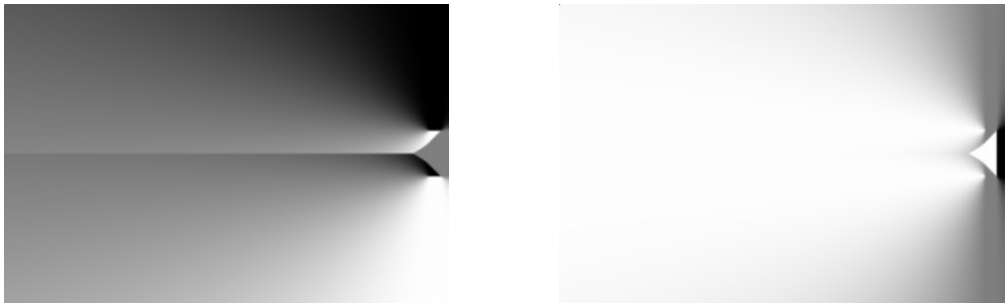


Figure 4.6: LRF integration task sequence diagram.

between 0 and 255, being the white lines in Figure 4.7 the representation of the goal lines, lighter cells represent lower distances and the more the distance for the lines grow, the more the cell becomes darker.

Figure 4.7: Illustration of the goal LUT map used by the algorithm. Brighter areas indicate positions with smaller distances, while darker areas indicate postions where the distance to the nearest goal line is large.

Another kind of map is required. It is the distance derivative in order to positions x and y. To calculate the partial derivatives matrix from the calculated distances we apply a Sobel filter [37]. The Sobel filter calculates the gradient for each cell, giving the higher variation direction from high to low distances and the amount of variation in this direction. Thus, we know for each cell, if it has a soft or blunt distance variation. In Figures 4.8(a) and 4.8(b) we can see the graphic representations of the goal LUT gradient in order to x and y, respectively.



(a) Representation in order to x.

(b) Representation in order to y.

Figure 4.8: Graphic representation of the gradient.

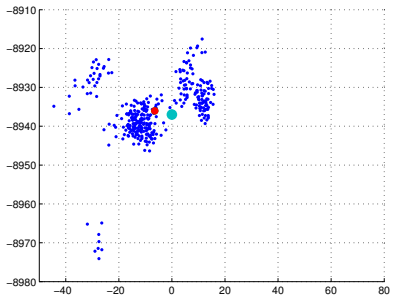
4.2.3 Results

In this chapter, to solve the current unkwown goal position to the goalkeeper, the *lrf* as a solution was implemented and tested. For the first test, the goalkeeper and the goal were centered in the field bottom line, so its center in CAMBADA team field it's the absolute position $(0, -8937)[mm]$. Removing the motion from the goalkeeper, for some time a set of vision localization and *lrf* localization values were measured and compared.

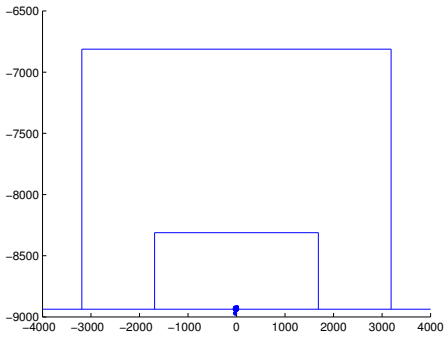
In Figure 4.9, the set of localization points measured with the *lrf* and vision were ploted. Figure 4.9(a) are the localization points acquired by the laser and Figure 4.9(c) are the localization points acquired from the vision process, that shows a single dot, due to the fact all measured localizations point matched. Those two plots show that the *lrf* as a localization method, despite the points are more spread than in the vision points plot, the average value point $(-6, -8936)[mm]$ when compared to the vison average value point $(77, -8930)[mm]$, shows that the laser has a better localization in the field. Those measured points are shown in Figures 4.9(b) and 4.9(d) in a lower resolution plot and with the goal area and penalty area represented.

Figure 4.10 shows the error between the localization points measured by the *lrf* and vision processes at the same cycle. The cloud points centered on the average error value $(84, 6)[mm]$ can be seen as systematic error.

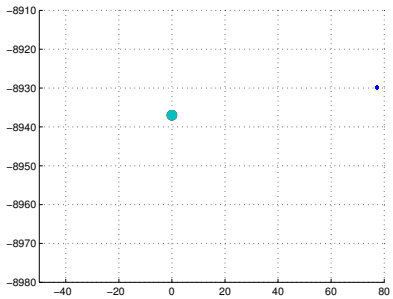
The goalkeeper was then moved to the absolute position $(-1000, -8312)[mm]$ in the field. A position in front of the left post on the X-axis and on the top of the goal area in the Y-axis, to make the study of the error between the vision localization and the *lrf* localization. Figures 4.11(a) and 4.11(c) shows the localization points distribution for each method and figures 4.11(b) and 4.11(d) an expansion of these points with the representation of the goal area lines and penalty area lines.



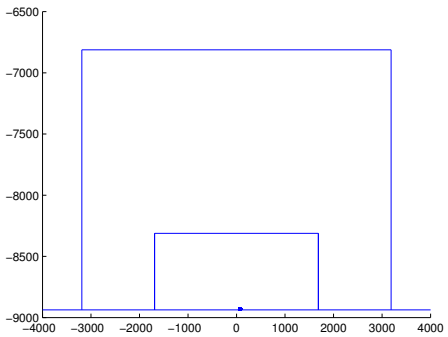
(a) LRF localization points.



(b) LRF localization points within the areas.



(c) Vision localization points.



(d) Vision localization points within the areas.

Figure 4.9: Localization points acquired by the LRF and vision processes with the goalkeeper in $(0, -8937)[mm]$ and the goal centered in the absolute position $(0, -8937)[mm]$. Blue dots are the localization points acquired, red dot is the average of those points and cyan dot is the real position of the goalkeeper.

The plot of the localization points shows a similar distribution as in Figure 4.9 where the goalkeeper is centered in the goal, with a *lrf* average value point $(-1057, -8328)[mm]$ and the vision average value point $(-1097, -8389)[mm]$. Figures 4.11(b) and 4.11(d) shows that for a lower resolution, the difference between the two localization methods is very small, even though when comparing that mean point with the real localization, the laser presents a better result.

The distribution of the error points shown at Figure 4.12 shows that the error stills systematic, with average value $(-40, -61)[mm]$.

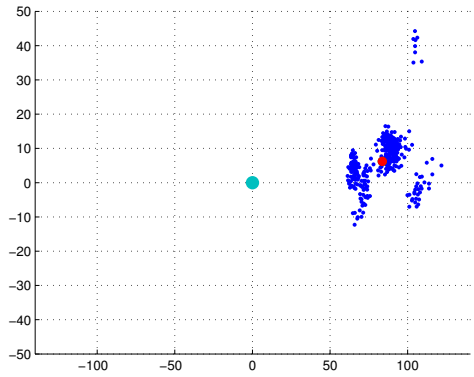
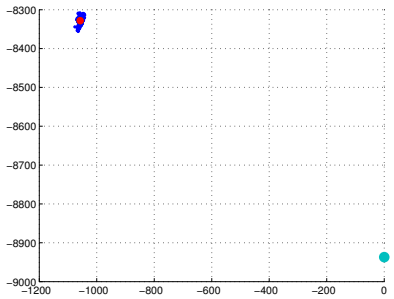


Figure 4.10: Error points for each cycle between LRF localization method and Vision localization method. Green dot is the $(0, 0)$ point which is the desired convergent point.

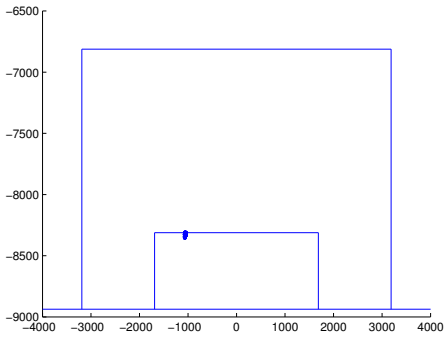
Now, keeping the goalkeeper in the absolute position $(-1000, -8312)[mm]$, the goal was moved along the bottom line, therefore in the X-axis, being now its center the absolute position $(-310, -8937)$. Figure 4.13 presents the localization points, Figures 4.13(a) and 4.13(b) shows the localization points according to the *lrf*, with an average value $(-729, -8365)[mm]$ and Figures 4.13(c) and 4.13(d) shows the localization points (which still a single point, as they all match), according to the vision capture, with an average value $(-1097, -8388)[mm]$. As expected, the vision localization doesn't follow the displacement of the goal, giving a localization point similar to the real absolute position of the goalkeeper in the field, while the *lrf* localization point tracks this displacement, and has localization point similar to the expected one, $(-690, -8312)[mm]$.

In Figure 4.14 the error points between the laser and vision localizations points are shown, with an average value of $(-367, 23)[mm]$. The average value has a difference of $(57, 23)[mm]$ from the expected one.

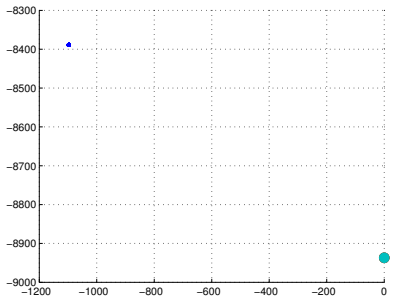
Again without moving the goalkeeper from previous position, now the goal its moved in the X-axis and Y-axis from its center, being the goal center absolute position $(-220, -8827)$. Figure 4.15(a) and 4.15(b) shows the new localization from the laser, with an average value $(-815, -8462)[mm]$. If the *lrf* keeps up tracking this displacement, an absolute position $(-780, -8422)[mm]$ in the field is expected. From Figures 4.15(c) and 4.15(d) it's clear that the vision localization doesn't keep up with the displacement and still gives a similar position, $(-1097, -8387)[mm]$ from the previous examples where the goalkeeper



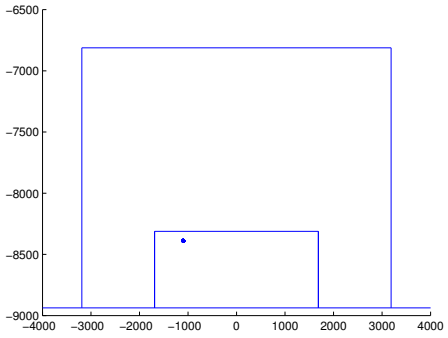
(a) LRF localization points.



(b) LRF localization points within the areas.



(c) Vision localization points.



(d) Vision localization points within the areas.

Figure 4.11: Localization points acquired by the LRF and vision processes with the goalkeeper in the absolute position $(-1000, -8312)[mm]$ and the goal center in the absolute position $(0, -8937)[mm]$. Blue dots are the localization points acquired, red dot is the average of those points and cyan dot is the center of the goal.

was in the same position.

Figure 4.16 shows the difference values between the localization points acquired from the *lrf* and the vision, with average value of $(-281, 74)[mm]$ and an error of $(-61, -36)[mm]$ between the mean value and the expected difference, as the green dot represents the expected point, with a value of $(-220, 110)$.

4.2.4 Conclusions

Evaluating the difference values from the expected position and the *lrf* localization position of $(84, 6)[mm], (-40, -61)[mm], (-57, -23)[mm]$ and $(-61, -36)[mm]$ for each

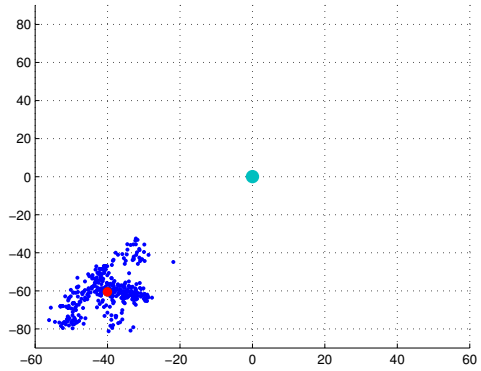


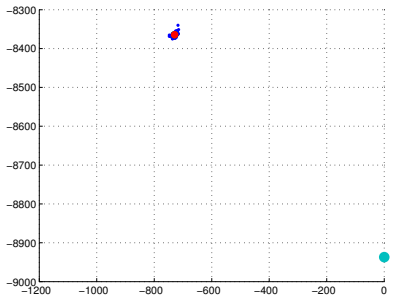
Figure 4.12: Error points for each cycle between LRF localization method and Vision localization method with the robot in the absolute position $(-1000, -8312)[mm]$. Green dot is the $(0, 0)$ point which is the desired and expected convergent point.

measure, it was ensured that the *lrf* tracks the goal displacement that may occur.

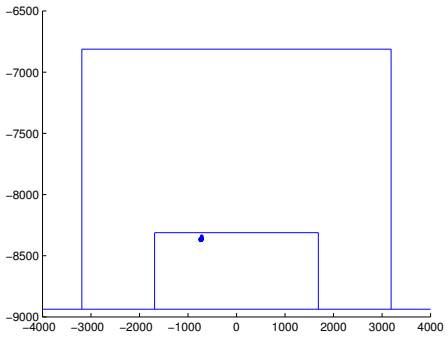
Moreover, by analyzing the error between the *lrf* obtained values and the vision, it can be stated that the *lrf*, as localization method, can be implemented in the goalkeeper, as long as the goal stays in the laser range and fixed to a known position. That way, fixes and enhancements to the real position of the goalkeeper on the field can be achieved with a deeper study between the mean error differences from vision and *lrf* methods for each localization.

To achieve the desired expectations, of a good positioning from the goalkeeper when defending the goal, the *lrf* solution was implemented. Thus, to integrate the acquired data from the laser, a program to access the device was implemented, the RtDB was adapted to connect the data from the *lrf* process side to the agent side. A localization algorithm based on the laser acquired data was then implemented. This localization method is exclusively used by the goalkeeper in situations where it's near the goal.

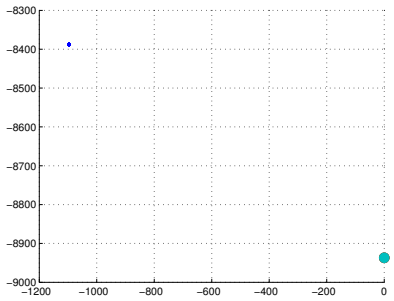
To integrate this new defending position given by the localization method implemented, the new information was introduced by modifying the Robot data structure, with the new field. By changing all behaviors assigned to the role Goalie, adding the gap between the real goal center and the one previously assumed, it was reached with success the expected results. This new fields are only updated when the goalkeeper is near the goal, more specifically when its absolute position in the Y-axis its lower than



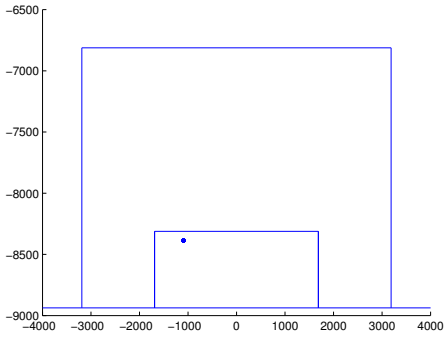
(a) LRF localization points.



(b) LRF localization points within the areas.



(c) Vision localization points.



(d) Vision localization points within the areas.

Figure 4.13: Localization points acquired by the LRF and vision processes with the goalkeeper in the absolute position $(-1000, -8312)[mm]$ and the goal center in the absolute position $(-310, -8937)[mm]$. Blue dots are the localization points acquired, red dot is the average of those points and cyan dot is the center of the goal.

$-5.5[m]$ and its X-axis absolute position is between $2.5/-2.5[m]$. When the goalkeeper is outside that boundaries it ignores the *lrf* information assuming that the goal its well positioned.

In Figure 4.17 two different situations where the goal was shifted from the bottom line center, identicals to the two from Figure 4.2, but with the implemented solution. Figure 4.17(a) shows a simple situation where the goal was displaced from the bottom line center and the goalkeeper is well positioned to what its beliefs are for the goal center. In figure 4.17(b) was introduced a ball, which it's the object the goalkeeper must protect its goal from. Now the goalkeeper is well positioned and defends near the post leaving

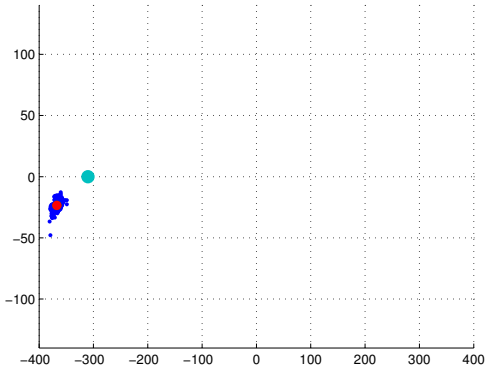
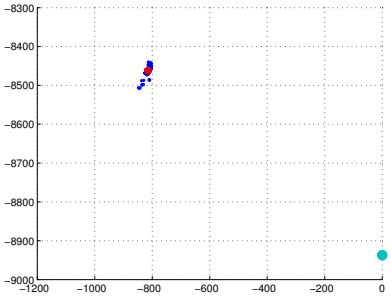
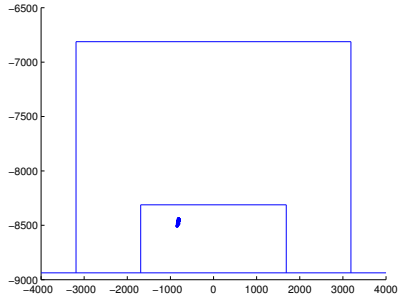


Figure 4.14: Error points for each cycle between LRF localization method and Vision localization method with the robot in the absolute position $(-1000, -8312)[mm]$ and goal center in the absolute position $(-310, -8937)[mm]$. Green dot is the $(-310, 0)$ point which is the desired and expected convergent point.

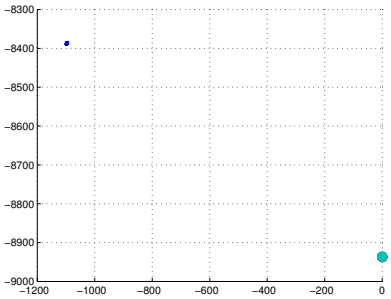
no gap, thus proving that the goalkeeper can now get a proper positioning to defend its goal in such adversive situations.



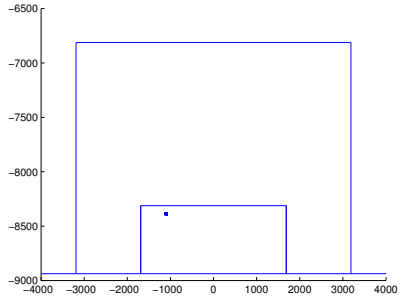
(a) LRF localization points.



(b) LRF localization points within the areas.



(c) Vision localization points.



(d) Vision localization points within the areas.

Figure 4.15: Localization points acquired by the LRF and vision processes with the goalkeeper in the absolute position $(-1000, -8312)[mm]$ and the goal center in the absolute position $(-220, -8827)[mm]$. Blue dots are the localization points acquired, red dot is the average of those points and cyan dot is the center of the goal.

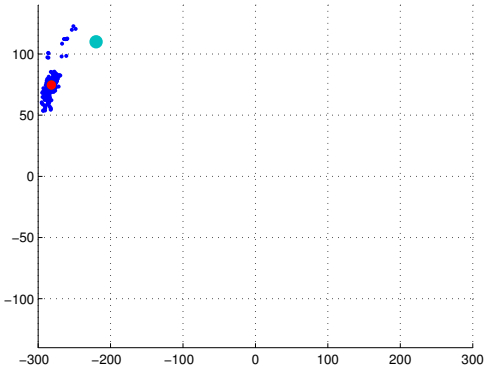
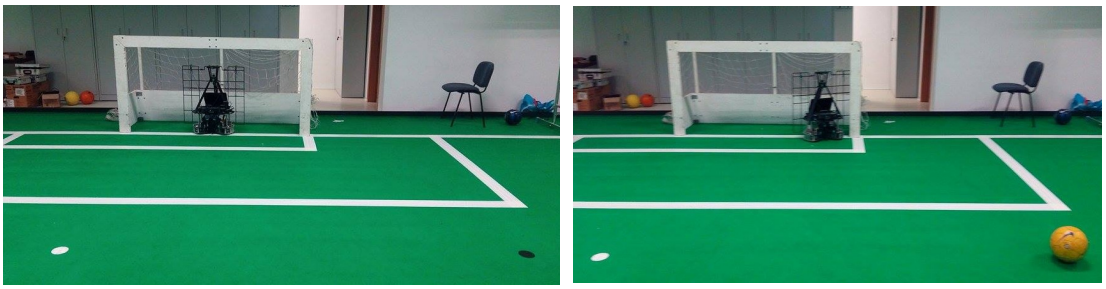


Figure 4.16: Error points for each cycle between LRF localization method and Vision localization method with the robot in the absolute position $(-1000, -8312)[mm]$ and goal center in the absolute position $(-220, -8827)[mm]$. Green dot is the $(-310, 0)$ point which is the desired and expected convergent point.



(a) Goal displaced to the left and the goalkeeper defendig what it's goal center beliefs are.

(b) Ball positioned to the left of the goalie and a correct positioning from the goalkeeper.

Figure 4.17: Two goalie defensive positions with the goal slightly shifted to the left.

Chapter 5

Penalty Kick Improvement with LRF

The penalty kick assumes a big importance and decisive part of a soccer game, as it's a simple set piece that faces two opposite players in a relatively small and well defined strategy space. Currently, as indicated in the RoboCup rules [38], the penalty kicks are not awarded during the two periods of play in a match, and specify that only in a game which ends in a draw after the two periods of play, and the game must have a winner, the execution of penalty kicks will take place to decide the winner. The fact that the penalty kick isn't just a set piece during gameplay where the offensive team can score, but a game winner decision, sets a even more importance to the success (goal scoring).

Currently, at CAMBADA team, in this specific set piece, the side which the striker chooses to shoot only depends on the position of the opponent goalkeeper, as it assumes that the goal is well centered in the field bottom line, and shoots to near one of the posts, positions $(800, 9000)[mm]$ near the right post or $(-800, 9000)[mm]$ near the left post, depending on the presented situation.

As the goal isn't attached to the ground, such assumption can't be assumed to always be correct. As an example of a situation where such belief would be bad, imagine that the goal is shifted to the left side, and the opponent goalkeeper follows that shift and it is well positioned and centered in the goal. The striker concludes that the goalkeeper is dislocated to the left, and shoots to the right side. If there was a major shift, the choosen point to shoot can be outside the boundaries, which will lead to a failing situation.

Introducing the *lrf* it's expected the detection of the posts and the goalkeeper, leading

to a well free space definition, thus hoping to increase the success ratio.

A brief explanation on how a player is assigned with penalty kick behavior will be exposed in the following subchapter.

5.1 The Penalty Kick Role

As the improvement made was in the penalty kick, a wider Penalty Kick role and behaviors explanation will be made.

Roles have an arbitrator and a callback method implemented by each derived Role, such as the Penalty Kick Role, which makes decisions and/or update internal variables of its own worldstate information. Behaviors and roles are created once in the beggining and are selected based on their conditions. In the constructor of the derived Role, a set of behaviors is added to its arbitrator queue. In Figure 5.1 the set of behaviors included in the arbitror queue can be seen.

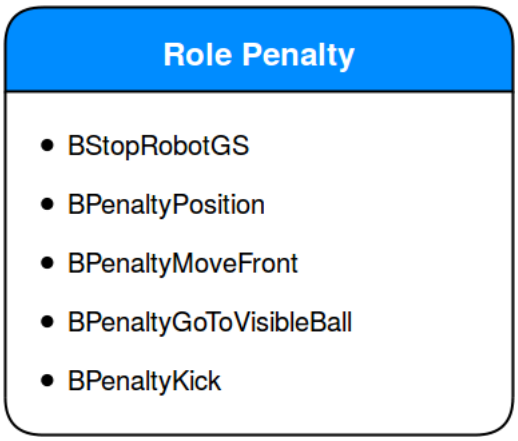


Figure 5.1: Behaviors added to arbitrary queue for role Penalty.

The first behavior is the *BStopRobotGS*, that is a global behavior, included in all roles arbitrator queues, with an high priority and will be triggered whenever the game state changes to Stop.

Next on the list we have the *BPenaltyPosition*, which is triggered when the game state changes to Penalty on our behalf, and the robot will move to the standby position waiting further instructions from the referee. As the standby position where it waits the referee signal to start the set piece is the field center, when it's given authorization to begin the penalty set piece, the robot vision may not see the ball, and in such cases, the

behavior *BPenaltyMoveFront* will trigger and make the robot moves slowly forward in the direction of the penalty marker. When the start signal is given, if the robot sees the ball, or when its vision detects the ball while moving front, the *BPenaltyGoToVisibleBall* will trigger and the robot will move to the ball in order to take the penalty.

Lastly, once the ball is grabbed, the behavior *BPenaltyKick* triggers, the robot calculates the variables it needs, and makes the kick. Afterwards, the robot becomes immobilized waiting further instructions. Figure 5.2 shows the flow chart of how a penalty is taken.

5.2 Adding the LRF solution

The BPenaltyKick behavior decides which side to shoot, by using the vision process, detecting the opponent goalkeeper as an obstacle, and setting the point to shoot according to the deviation of the goalkeeper from the point $(0, 9000)$, but sometimes the goal isn't positioned as expected, thus the *lrf* sensor approach as solution to define the boundaries of the goal and the position of the opponent goalkeeper was implemented. Thus, the moment the robot grabbs the ball, and the *BPenaltyKick* behavior is triggered, two extra possible solutions were implemented.

Figure 5.3 shows two of four possible situations when the striker grabs the ball. In Figure 5.3(a) as the striker grabs the ball, a scan from the surrounding capture the points from the goal and the opponent goalkeeper. In the second situation seen at Figure 5.3(b), similar to the situation from Figure 5.3(a), the *lrf* captures the goal points, but the opponent goalkeeper has an absorvent surface and it's undetectable. A third possible situation unshown, it's a mix between the two previous, where the goalkeeper is detectable but still absorves some of the radiation. A fourth and unlikely, but still possible situation, is when the striker grabs the ball and has no opponent goalkeeper.

Two possible solutions were implemented, one *Hybrid Solution* which takes advantage of the opponent goalkeeper detection from the vision capture, and *LRF Solution* where all points of interest are calculated based on the laser acquired information. In both methods, the *lrf* data requires a time validation.

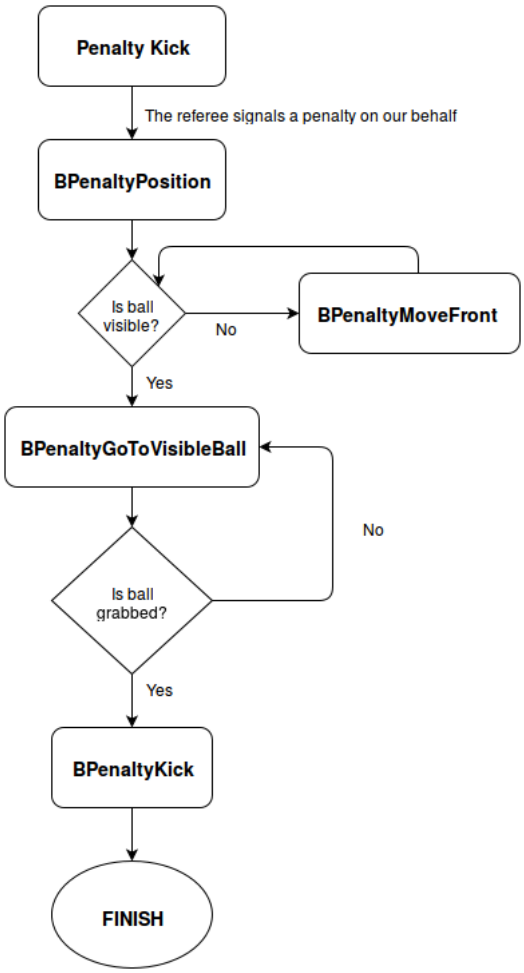


Figure 5.2: Flow chart of the Penalty Role.

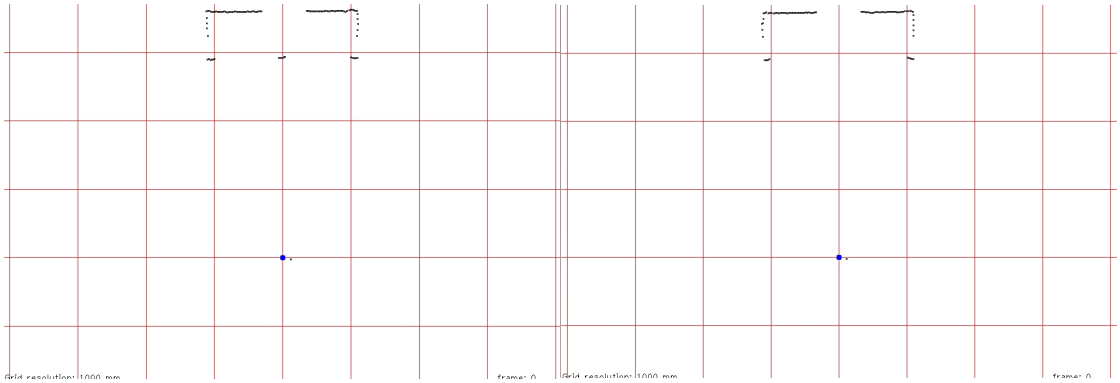
5.2.1 Hybrid Solution

Due to the specific conditions of a penalty kick set piece, where only two players participate, if the vision captures an obstacle in a specific area near the goal, it has an extremely high probability to be the opponent goalkeeper.

This solution was implemented to ignore the cases where the opponent goalkeeper absorves, completely or partially, the laser radiated light.

Figure 5.4 shows how the hybrid solution delimits the boundaries of the opponent goalkeeper and the inside boundaries of the posts.

In the first phase, the opponent goalkeeper center point is acquired from the information defined with the vision information. As the RoboCup rules state that the maximum width of the players are $50cm$, preventing the worst case scenarios, the right and left boundaries are defined by adding or subtracting $25cm$, respectively, to the X-axis vision



(a) Capture acquired from the *lrf* with the opponent goalkeeper visible.

(b) Capture acquired from the *lrf* with the opponent goalkeeper invisible.

Figure 5.3: Two possible captures in a penalty situation.

acquired point.

The last step it's to define the boundaries from the right and left post, which are found, on the premise that the goal isn't displaced more than $10cm$ in the Y-axis, by analyzing and finding the first points that lie close to the distance from the laser to the bottom line, when converging from the initial and final step to the middle.

With the four parameters well defined, the next step is to compare the space left between the boundaries of the goalkeeper and the post, and choose the point to shoot near the post where the gap between the goalkeeper and the post is bigger.

5.2.2 LRF Solution

In this method, not only the goal boundaries are defined by the *lrf*, but the opponent goalkeeper boundaries as well. In Figure 5.5, a diagram on how both boundaries are defined is shown.

The first step it's to find the boundaries of the goal. Similarly to the hybrid solution, analyzing the scan from the begin until finding points at the expected distance, and from the end of the scanning, backwards, until find points at the expected position, thus defining the inside boundaries of the posts.

Forward, by analyzing the data from the *lrf* corresponding to the 0° orientation, the presence of the opponent goalkeeper is acquired with the absence of points at the distance to the goal bottom, thus filling all three possible situations (goalkeeper visible, invisible or a mix between both). For the two possible situations, different approachs were made.

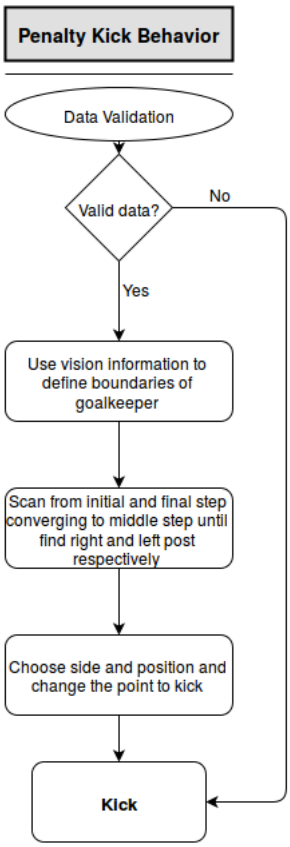


Figure 5.4: Flow chart of the Hybrid method implemented.

Opponent goalkeeper is present ahead

For such situations, as seen before in Figure 5.3, the opponent goalkeeper was already found. Now, it's a simple scan for both sides of the data point analyzed before. Thus, reading the Y-axis data point for each side until finding the expectable first point at the goal bottom distance.

Opponent goalkeeper is not ahead

For such cases, the question is: is it the goalkeeper there or not? To reach such conclusion, similarly to the previous scenario, the starting point of the analysis it's the same. But now, the corresponding distance is now the one to the goal bottom, thus, a scanning until a non goal back distance point is reached it's made. If that point has higher distance to the respective post than a few centimeters, it's assumed its the first data point from the opponent goalkeeper, and its boundaries are defined, assuming again that the opponent goalkeeper has maximum width of $50cm$. If not, an analog scan for the other side of the central point is made and

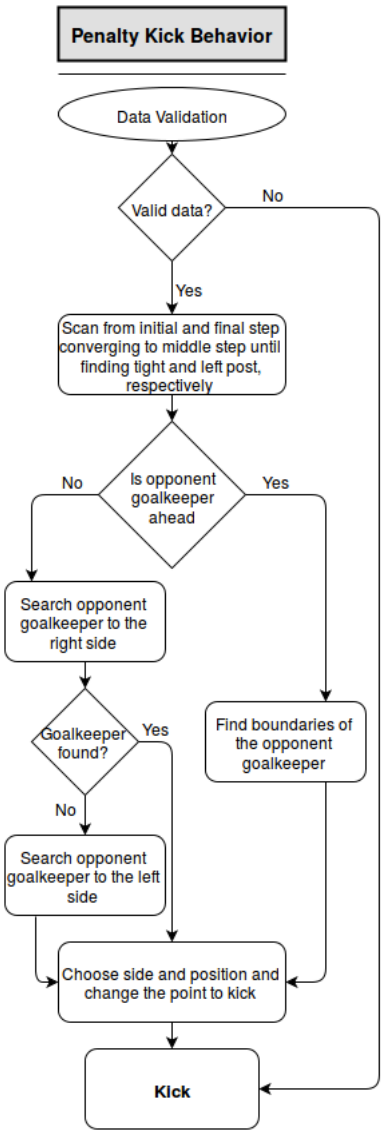


Figure 5.5: Flow chart of the pure LRF method implemented.

analog reasoning is made. If the goalkeeper isn't found as well, it's assumed that the opponent goalkeeper isn't between the limits of the goal.

After defining all four parameters, the same simple and obvious solution implemented for the hybrid method to decide which side to shoot it's made. For the cases where the boundaries of the goalkeeper remain untouchable, i.e., the goalkeeper isn't found inside the goal, it was chosen a random side, as the goalkeeper isn't in the goal. To the other case, we choose the larger free space between the inside boundaries of the posts and their respective boundary of the goalkeeper.

The only assumption made on the implementation of both methods it's that the goal doesn't has a displacement higher than $10cm$. The misplacement of the ball outside of the penalty marker is prevented as well the orientation of the laser as the robot grabs the ball.

5.3 Results

Penalty kick situations currently have an high importance, as they define a game winner. To ensure that the striker is well prepared, and has an high ratio of success in penalty kicks, the *lrf* as an improvement was implemented. For different situations, a set of 50 penalties were taken.

5.3.1 Goal well positioned

Figure 5.6 are the results for the old implemented method, based only on vision, the hybrid solution and the *lrf* solution, when the goal it's well centered in the bottom line. All solutions have high ratio of success and are reliable, provided the conditions remain.

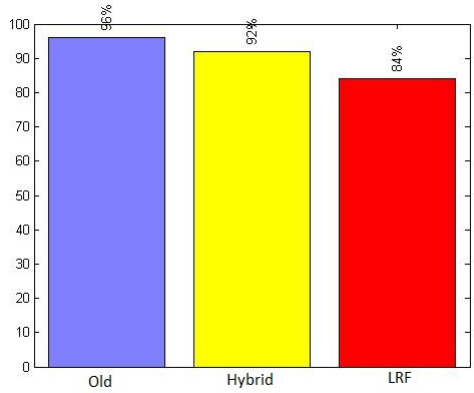
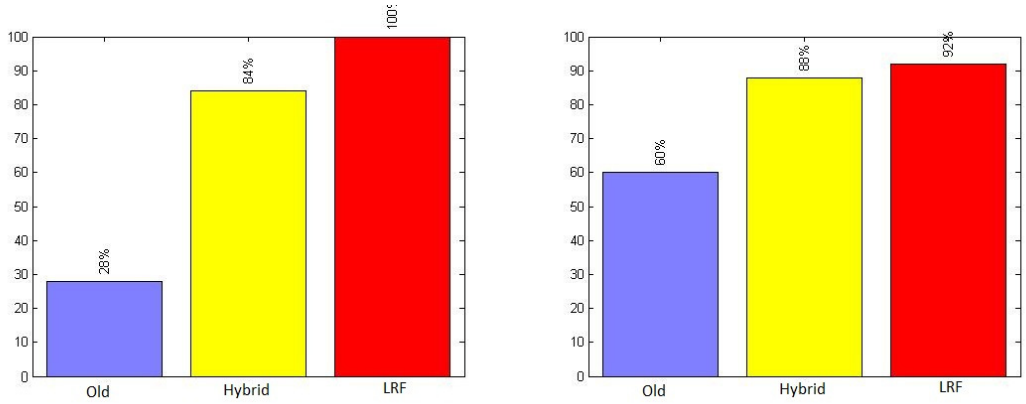


Figure 5.6: Success ratio for the penalty kicks taken with the goal well centered in the bottom line for each method.

5.3.2 Goal displaced 10cm

Figure 5.7 shows the results with the goal displaced to the left side of the striker 10cm. In Figure 5.7(a) the opponent goalkeeper isn't tracking the misplacement of the goal, while on Figure 5.7(b) it does. Analyzing the results, for the implemented methods,

as in previous situation, have an high ratio, nearly the 100%, while the previous existing method has an expectable decay, since the choosing point for shooting stills in the limits of the goal for both sides, but relies on luck in the situation where the opponent goalkeeper it's in the bottom line center and can be a little to the right side, getting the striker shooting for the open free space. In the second case, the goalkeeper tracks the displacement of the goal, and the striker will always choose to shoot near the post, that may or may not be goal.



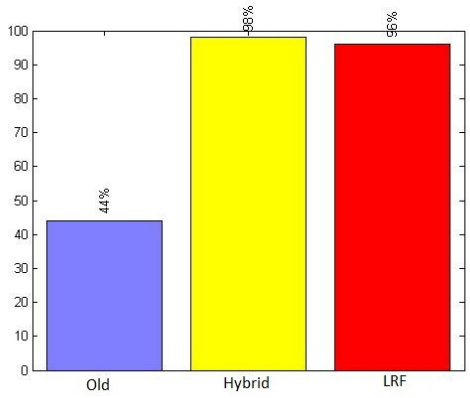
(a) Success ratio for the penalty kicks taken when the opponent goalkeeper doesn't track the goal displacement.

(b) Success ratio for the penalty kicks taken when the opponent goalkeeper tracks the goal displacement.

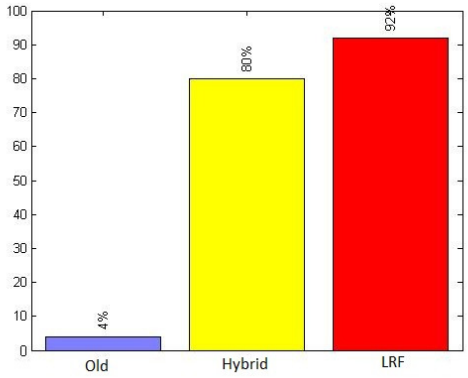
Figure 5.7: Success ratio for the penalty kicks taken with the goal displaced 10cm to the left from the bottom line center for each method.

5.3.3 Goal displaced 25cm

From the previous analysis, as it's expectable, with the big goal displacement from the bottom line center, and as can be seen in Figure 5.8, which shows the results for the three methods, the older method based only on the positioning of the opponent goalkeeper, will only have a decent ratio in the situation where the opponent goalkeeper doesn't track the displacement of the goal as can be seen in Figure 5.8(a), as it can fortunatelly choose the side which the goal has been displaced. The scenario where the goalkeeper tracks and follows the displacement of the goal and it's well positioned, the striker will have a low success ratio as can be seen in Figure 5.8(b). In the cases of the implemented methods, although not perfect, still have high ratios for both scenarios.



(a) Success ratio for the penalty kicks taken when the opponent goalkeeper doesn't track the goal displacement.



(b) Success ratio for the penalty kicks taken when the opponent goalkeeper tracks the goal displacement.

Figure 5.8: Success ratio for the penalty kicks taken with the goal displaced 25cm to the right from the bottom line center for each method.

Chapter 6

Conclusion and Future Work

The main objective of this thesis was to efficiently detect, and effectively define the boundaries of our own goal for a more efficiently positioning in defensive moments of the game, and the opponent goal and goalkeeper for the Penalty Kick set piece, to maximize the the ratio of scored goals.

6.1 Conclusion

When set side by side, the before and after, the localization in the field according to the goal when using information from the *lrf*, if the goal is stationary, the laser proved to be a reliable source for localization. Thus, when moving the goal, as its boundaries absolute positions will be different from past beliefs of the robot, by localizing itself only through the *lrf* information and comparing the difference between the vision localization and *lrf* localization, we can now find the new boundaries of the goal and have a correct positioning when defending the goal during a gameplay situation.

For the penalty improvement, the previous behavior, was fully functional and has a high ratio, if the goal is well centered in the bottom line. As the goal moves from that center point, if the opponent goalkeeper keeps up with that deviation, the ratio falls critically. When the opponent goalkeeper does not follow that deviation, randomness in the opponents goal deviation for one side or other from the bottom line center point, will define how successfull the robot will be scoring goals. By introducing the laser improvement, both hybrid and pure laser information methods, have an high scoring ratio, although lower when the goal is centered in the bottom line when compared with the previous behavior, but a major improvement when the goal is moved, whether the

opponent goalkeeper follows that deviation from the goal to the bottom line center point or not.

6.2 Future Work

We can state that positive results and improvements in the agent behaviors were achieved, but assert that those can not be improved is going against the will of the human being to allways improve.

So for the goalkeeper goal detection, we can think ahead, and prepare the algorithm for when the back of the goal will be substituted by actual nets, getting similar to the real ones used in soccer. For the penalty, as the information it's only acquired as the robot grabs the ball, the opponent goalkeeper can move from the initial point to another. So, by making new measures only near the points where the goalkeeper was detected in that last measure, and finding if the goalkeeper has some motion.

Other improvements can be made in other behaviors, such as in free play, when passing the ball to a teammate. By fusing the vision information, for knowing if the obstacle it's a teammate, acquiring information from the *lrf* and obtaining the reference positioning of the teammate can improve the passing point.

Moreover, outside the soccer environment, introducing the laser in different applications, as an example, with the technology advance, smaller lasers can be constructed, WiFi communication between the laser and processing unit can be accomplished, introducing the laser in a blind stick to detect objects that may arm the user and warn him, it's one of the many applications.

Bibliography

[1] I. Asimov *I, Robot*, December 1950

[2] *http://www.electronicsteacher.com/robotics/type-of-robots.php* last seen at October 30 of 2015

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda and E. Osawa *Robocup: The robot world cup initiative* in The First International Conference on Autonomous Agent

[4] *http://www.embedded.com/design/real-world-applications/4419781/Giving-robotic-systems-spatial-sensing-with-visual-intelligence* last seen at October 30 of 2015

[5] A. Neves, J.L. Azevedo, N. Lau, B. Cunha, J. Silva, F. Santos, G. Corrente, D.A. Martins, N. Figueiredo, A. Pereira, L. Almeida, L.S. Lopes and P. Pedreiras *CAMBADA soccer team: from robot architecture to multiagent coordination*, chapter 2, pages 19-45, I-Tech Education and Publishing, Vienna, Austria, January 2010

[6] P. Ciancarini and M.J. Wooldridge *Agent-Oriented Software Engineering*, June, 2000

[7] R. Siegwart and I.R. Nourbakhsh *Introduction to Autonomous Mobile Robots*, 2004

[8] N.A. Cabrol, G. Chong-Diaz, J.M. Dohm, M. Pereira Arredondo, G. Dunfield, V.C. Gulick, A. Jensen-Iglesia, R. Keaten, C. Herrera Lamelli, R. Landheim, P.C. Lee, L. Pederson, T. Roush, K. Schwehr, C.R. Stoker, A. Zent *Atacama I: Science Results of the 1997 Nomad Rover Field Test in the Atacama Desert, Chile* in 29th Annual Lunar and Planetary Science Conference, March 16-20, 1998, Houston, TX, abstract no. 1013

[9] *http://news.mit.edu/2015/cognitive-underwater-robots-0507* last seen at November 12 of 2015

[10] *http://atlas.web.ua.pt/index.html* last seen at November 12 of 2015

[11] *http://www.techunited.nl/en/amigo* last seen at November 12 of 2015

[12] *http://www.frc.ri.cmu.edu/projects/bigsignal/2000/background/combined_files_for_shriver/images/* last seen at November 09 of 2015

[13] *http://buildbot.com.br/blog/engenheiros-do-mit-aprimoram-robos-submarinos-autonomos/* last seen at November 09 of 2015

[14] *http://www.robocup2013.org/press/* last seen at November 12 of 2015

[15] *http://osorio.wait4.org/palestras/farlei-robos-parte2.pdf* last seen at October 30 of 2015

[16] *http://www.designworldonline.com/smart-vision-sensors-from-datasensor/* last seen at November 09 of 2015

[17] *http://stm32f4-discovery.com/2014/08/library-30-measure-distance-hc-sr04-stm32f4xx/* last seen at November 09 of 2015

[18] D. Fox, W. Burgard and S. Thrun *Active Markov localization for mobile robots.* In *Robotics and Autonomous Systems*, Volume 25, Issues 3-4, 30 November 1998, Pages 195-207

[19] K.O. Arras, J.A. Castellanos, R. Siegwart *Feature-Based Multi-Hypothesis Localization and Tracking for Mobile Robots Using Geometric Constraints.* In *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, May 11-15, 2002

[20] S. Thrun, D. Fox, W. Burgard, F. Dellaert *Robust Monte Carlo localization for mobile robots.* In *Artificial Intelligence*, Volume 128, Issues 1-2, May 2001, Pages 99-141

[21] S. Thrun, W. Burgard and D. Fox *Probabilistic Robotics*, The MIT Press, 2005

[22] D. Fox, W. Burgard, F. Dellaert and S. Thrun *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots*, Carnegie Mellon University Pittsburgh, PA and University of Bonn, Germany

[23] T. Laue, T. Rfer *Pose Extraction from Sample Sets in Robot Self-Location - A Comparison and a Novel Approach*, Safe and Secure Cognitive Systems, Bremen, Germany

[24] M. Lauer, S. Lange and M. Riedmiller *Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization*, in RoboCup 2005: Robot Soccer World Cup IX, LNCS. Springer, 2005, pp 142-153

[25] M. Riedmiller and H. Braun *A direct adaptive method for faster backpropagation learning: the RPROP algorithm*, in IEEE International Conference on Neural Networks, volume 1, pages 586-591, 1993

[26] J. Silva *Perception and software architecture for mobile robotics*, Electronic, Telecomunications and Informatic Department, University of Aveiro

[27] R. Dias *Agent Architecture of the CAMBADA Robotics Soccer Team*, Electronic, Telecomunications and Informatic Department, University of Aveiro

[28] A. Neves, A. Pinho, D. Martins and B. Cunha *An efficient omnidirectional vision system for soccer robots: from calibration to object detection.* in Mechatronics, 21(2):339-410, March 2011

[29] P. Jensfelt *Approaches to Mobile Robot Localization in Indoor Environments*, 2001

[30] *http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/urg-04lx-ug01/* last seen at October 16 of 2015

[31] J.L. Azevedo, B. Cunha and L. Almeida *Hierarchical distributed architectures for autonomous mobile robots: a case study.* in Proc. of the 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2007, pages 973-980, 2007.

[32] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L.S. Lopes. *Coordinating distributed autonomous agents with a real-time database: The CAMBADA project.* in Proc. of the 19th International Symposium on Computer and Information Sciences, ISCIS 2004, volume 3280 of Lecture Notes in Computer Science, pages878886. Springer, 2004.

[33] S. Lange, C. Muller, and S. Welker *Behavior-based approach* November 2008

[34] M. Lauer, S. Lange, and M. Riedmiller *Cognitive concepts in autonomous soccer playing robots* Cognitive Systems Research, 2010

[35] R.A. Brooks *A robust layered control system for a mobile robot* Robotics andAutomation, IEEE Journal of, 2(1):1423, March 1986

[36] A.S. Rao and M.P. Georgeff *Modeling rational agents within a BDI-architecture* In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pages 473484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[37] *https://en.wikipedia.org/wiki/Sobel_operator* last seen at October 23 of 2015

[38] *http://wiki.robocup.org/images/3/3f/Msl-rules_2015.pdf* last seen at October 24 of 2015