UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

João Guilherme Oliveira Carvalho de Melo

Onboard Perception and Localization for Resource-Constrained Dynamic Environments: A RoboCup Small Size League Case Study

Recife
2023

João Guilherme Oliveira Carvalho de Melo

Onboard Perception and Localization for Resource-Constrained Dynamic Environments: A RoboCup Small Size League Case Study

A M.Sc. Dissertation presented to the Center of Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Concentration Area: Computer Engineering

Advisor: Edna Natividade da Silva Barros

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

M528o Melo, João Guilherme Oliveira Carvalho de
Onboard perception and localization for resource-constrained dynamic environments: a RoboCup small size League Case Study / João Guilherme Oliveira Carvalho de Melo – 2023.
87 f.: il., fig., tab.

Orientadora: Edna Natividade da Silva Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Engenharia da computação. 2. Robôs móveis autônomos. 3. Auto-localização. 4. Monte Carlo Localization. 5. RoboCup. I. Barros, Edna Natividade da Silva (orientadora). II. Título

621.39 CDD (23. ed.) UFPE - CCEN 2023 – 127

João Guilherme Oliveira Carvalho de Melo

“Onboard Perception and Localization for Resource-Constrained Dynamic Environments: A RoboCup Small Size League Case Study”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Engenharia da Computação.

Aprovado em: 7 de agosto de 2023.

ABSTRACT

Self-localization consists of estimating a robot's position and orientation (pose) regarding its operating environment and is a fundamental skill in autonomous mobile robot navigation. Monte Carlo Localization (MCL) is a particle filter-based algorithm that addresses the localization problem by maintaining a set of particles that represent multiple hypothesis of the current robot's state. At each step, the particles' are moved according to the robot's motion and their likelihoods, also called importance weights, are estimated based on the similarities between measurements acquired by the robot and their expected values, given the particle state. Then, a resampling algorithm is applied to the distribution, generating a new set based on the current weights.

MCL finds successful utilization in RoboCup robot soccer leagues for solving the self-localization problem in humanoid and standard platform competitions. At 2022, this problem was also introduced in the RoboCup Small Size League (SSL) within the Vision Blackout Technical Challenge, which restricts teams to use onboard sensing and processing only for executing basic soccer tasks, instead of the typical SSL approach that uses an external camera for sensing the environment, but no solutions were proposed for self-localization so far. Therefore, this work presents an integrated pipeline for solving the SSL self-localization problem while also detecting the environment's dynamic objects, using onboard monocular vision and inertial odometry data.

We enhance the MCL using insights from implementations of other RoboCup leagues, improving the algorithm's robustness regarding imprecise measurements and motion estimations. Also, we increase the algorithm's processing speed by adapting the number of particles in the set according to the confidence of the current distribution, also called Adaptive MCL (AMCL). For that, we propose a novel approach for measuring the quality of the current distribution, based on applying the observation model to the resulting particle of the algorithm. The approach was able to drastically increase the system's computation speed, while also maintaining the capability to track the robot's pose, and the confidence measure may be useful for making decisions and performing movements based on the current localization confidence.

Keywords: autonomous mobile robots; self-localization; Monte Carlo Localization; RoboCup.

RESUMO

Auto-localização é uma habilidade fundamental no campo de robôs móveis autônomos e consiste em estimar a posição e orientação de um robô em relação ao seu ambiente de operação. Localização de Monte Carlo (em inglês *Monte Carlo Localization* - MCL) é um algoritmo baseado em filtros de partículas, abordando o problema de localização através de um conjunto de partículas que representam múltiplas hipóteses do estado atual do robô. Em cada iteração, as partículas são movimentadas de acordo com os deslocamentos realizados pelo robô e suas verossimilhanças são estimadas com base nas similaridades entre medidas adquiridas pelo robô e seus valores esperados, dados os estados das partículas. Em seguida, um novo conjunto de partículas é gerado com base nos pesos atuais através de algoritmos de reamostragem e o processo é reiniciado. MCL é utilizado com sucesso em diversas ligas de futebol de robôs da *RoboCup* para resolver o problema de localização, especialmente em competições de robôs humanóides e de plataformas padronizadas. Em 2022, este problema foi introduzido na categoria *Small Size League* (SSL) através do desafio técnico chamado *Vision Blackout*, que restringe os times a utilizarem apenas técnicas de sensoriamento e processamento embarcados para executar tarefas do futebol de robôs. Assim, este trabalho apresenta uma solução integrada para resolver o problema de auto-localização no contexto de SSL enquanto, conjuntamente, detecta objetos dinâmicos do ambiente, utilizando informações adquiridas por uma câmera monocular e odometria inercial embarcados. Nós aprimoramos o algoritmo de MCL utilizando idéias de implementações propostas por outras pesquisas realizadas em outras ligas da *RoboCup*, garantindo mais robustez à imprecisões em medidas e estimativas de odometria. Ademais, nós aceleramos a velocidade de processamento do algoritmo adaptando o número de partículas utilizadas de acordo com a confiança atual da distribuição, método também chamado de MCL adaptativo. Para isto, propomos uma nova abordagem para medir a qualidade da distribuição atual, baseada em aplicar o modelo de observação ao estado resultante do algoritmo de localização. A abordagem foi capaz de aumentar drasticamente a velocidade de processamento do sistema, sem perder sua capacidade de rastrear a localização do robô, e a nova métrica de confiança também pode ser aproveitada para tomar decisões e realizar movimentos que favoreçam a convergência do algoritmo de localização.

Palavras-chaves: robôs móveis autônomos; auto-localização; *Monte Carlo Localization*; *RoboCup*.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Robotics systems can be described as physical devices that perceive and manipulate the environment through computer-controlled mechanisms (THRUN; BURGARD; FOX, 2005), designed to assist or supplant humans on predetermined tasks for increasing productivity, improving safety, or enhancing efficiency (ALBUQUERQUE et al., 2017). The increasing integration of robots in daily life is progressively transforming the way humans interact with the world, finding successful utilization in several domains, including products manufacturing in industrial assembly lines (KUKA, 2016), inspecting hazardous environments in the oil and gas industry (YU et al., 2019), assisting surgeons with Robotically Assisted Surgical Sytems (RASS) (KLODMANN et al., 2021), automating household tasks with home service robots (ZACHIOTIS et al., 2018), transporting charges with autonomous aerial and ground vehicles (SRINIVAS; RAMACHANDIRAN; RAJENDRAN, 2022), and many others.

Among these systems, Autonomous Mobile Robots (AMR) perform highly relevant roles in society and industry (KOLAR; BENAVIDEZ; JAMSHIDI, 2020), and must be able to navigate through dynamic, unpredictable environments autonomously. Despite their employment in a wide variety of applications, each following different requirements and constraints, AMRs are built on top of common capabilities: perception, localization, mapping, cognition, path planning, and motion control (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

To push the state-of-the-art in robotics and artificial intelligence (AI), the Robot World Cup Initiative (RoboCup) was created in 1997 (KITANO et al., 1997). It proposes the use of soccer games as a research platform, establishing the goal of developing a team of fully autonomous humanoid robots capable of winning an official soccer match against the human World Cup champion team until 2050. This challenge requires integrated studies in multiple areas, such as real-time sensor fusion, reactive behavior, strategy acquisition, learning, real-time planning, multi-agent cooperation, and coordination, context recognition, computer vision, decision-making, motor and robot control, and many others. Currently, to address these problems, RoboCup presents several different robot soccer leagues, each of them focusing on different aspects of this goal.

1.1 MOTIVATION

Among the RoboCup soccer leagues, the Small Size League (SSL) focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic adversarial environment. In this League, games take place between two teams of 6 (division B) or 11 (division A) omnidirectional mobile robots (OMR), which must respect the height and diameter constraints of 150 and 180 mm, respectively, playing with an orange golf ball on a green carpeted field. These objects are tracked by a standardized vision system, the SSL-Vision (ZICKLER et al., 2010), which detects soccer elements and estimates their positions regarding the field coordinates.

The typical approach adopted by SSL teams is to use off-field computers for executing most of the computation, receiving objects' positions, field geometry information, and referee commands, leaving the off-board computation to process vision filtering, decision-making, path planning and position control, sending velocity commands to the robots through radio frequency wireless communication using minimal bandwidth. Even though this architecture has been successfully employed throughout the years of SSL, it presents limitations regarding data reliability and latency:

- One common issue during matches is ball occlusion, which occurs when a robot's projection on the camera image overlaps the ball, causing the camera not to see it.
- SSL-Vision detection is based on color segmentation, which is prone to false positives, occasionally causing wrong position estimations.
- Perception updates are limited to the cameras' capture frame rates, and commands only reach the robots after being computed and communicated, limiting the latency between commands.

As the League improves towards more complex and dynamic strategies, with an increasing number of robots, larger soccer fields, and faster velocities, these limitations become more relevant, demanding faster and more reliable solutions for sensing, perceiving and tracking elements of the environment.

Researches in the SSL have shown that faster robot position updates can be achieved by merging SSL-Vision information with onboard tracking using odometry methods based on embedded inertial sensors, reducing the overall system's latency (ABBENSETH; OMMER, 2015). Sensors fusion is also employed on state-of-the-art autonomous navigation systems, not only reducing latency, but also enhancing the perception's reliability and accuracy (KOLAR; BENAVIDEZ; JAMSHIDI, 2020).

Following these approaches and for encouraging teams to propose solutions to overcome the SSL architectural limitations, the League has a technical challenge for developing autonomous capabilities for robots. The Vision Blackout challenge incentives teams to explore local sensing and processing rather than using the typical SSL approach of an off-board computer and a global set of cameras sensing the environment (SMALL SIZE LEAGUE TECHNICAL COMMITTEE, 2022a). This challenge focuses on demonstrating standard SSL soccer skills (e.g. locomotion, ball manipulation and collaboration between robots), without the SSL-Vision software running, and is split into multiple stages designed to have different levels of difficulty and evaluate each of these desired capabilities. In the 2022 edition, stages and their objectives consisted of:

- **Stage I - Grabbing a stationary ball somewhere on the field:** designed to demonstrate basic sensing and locomotion capabilities.
- **Stage II - Score with the ball on an empty goal:** designed to demonstrate similar skills to stage 1, plus the ability to localize and aim at a known target on the field.

- **Stage III - Move the robot to specific coordinates:** demonstrates strong localization abilities without the use of the global vision system.
- **Stage IV - Score an indirect goal (two robots needed):** designed to demonstrate coordination skill of multiple robots without external vision. The robots must collaborate to pass and score the goal.

For this challenge robots have no height restrictions, enabling teams to add computing modules and onboard cameras to the robots. Still, the SSL environment demands high throughput and the size limitations coupled with using a battery as a power source require solutions to have low power consumption. Also, precise kicks and passes on high distances are executed during matches, requiring accurate position estimations. In summary, these conditions require proposed solutions for the Vision Blackout challenge to take into account hard trade-offs between power consumption, size, processing speed and accuracy.

Recent work have shown that grabbing the ball, scoring a goal, and making passes can be performed autonomously by detecting SSL objects (balls, goals, and robots), estimating their relative positions, and using this information for making decisions and acting (MELO et al., 2022). These skills compound the basis for solving stages I, II, and IV of the SSL challenge and, even though they could be solved without global localization knowledge, the authors report that this information (global localization) might be the key to surpassing the major limitations of their proposed system, pointed as: planning more efficient paths, discarding out-of-field information, avoiding penalties, and making more efficient field explorations.

Besides improving performances on stages I, II and IV, global position information is a must for solving the challenge's stage III (move the robot to specific coordinates) and, despite other RoboCup leagues presenting methods for it (RÖFER; JÜNGEL, 2004; RÖFER; LAUE; THOMAS, 2006; RÖFER et al., 2019; SEEKIRCHER; LAUE; RÖFER, 2011; MUZIO et al., 2016; BURCHARDT; LAUE; RÖFER, 2011), computing self-localization is still an open-problem in the SSL.

1.2 RESEARCH PROBLEM

Self-localization consists of estimating a robot's position and orientation (pose) regarding its operating environment and is a fundamental skill in autonomous mobile robot navigation (GUTMANN et al., 1998; GUTMANN; FOX, 2002). Studies in this research field are commonly split in two problems: local localization, or position tracking, and global localization. The first focuses on tracking the robot's moves for estimating its trajectory and can be used for computing a global position if the initial pose is known as prior. In contrast, the global localization problem aims at finding the robot's pose by making observations in a previously mapped environment, under the condition of unreliable or no initial pose information. Solutions for these problems rely on uncertain measurements and predictions, based on the environment and the robot's actions. Therefore, several approaches were developed for considering uncertainties in robot localization, using mainly probabilistic functions for representing measurements, movements and poses (THRUN; BURGARD; FOX, 2005), and GUTMANN; FOX (2002) categorizes them into three classes: Kalman filters, grid-based Markov localization, and Monte Carlo methods.

Kalman Filters (KF) rely on the premise that the robot's pose, motion and measurements can be represented as Gaussian distributions, integrating uncertainty into their computations. KF-based algorithms also assume this information to follow linear behaviours, which does not apply for many robotics systems. Therefore, Extended Kalman Filters (EKF) and Unscented Kalman Filters (UKF) are proposed as solutions for representing non-linear systems (PANIGRAHI; BISOY, 2021). Even though these methods are widely employed for solving position tracking problems, their inherent characteristic of estimating a single hypothesis of the robot's pose makes them unable to solve the global localization problem or to recover from total localization failures (GUTMANN; FOX, 2002).

An approach for addressing the global localization problem is to use Grid-based Markov Localization (ML), which maintains a probability density over a discrete space of all possible locations of a robot in its environment (FOX; BURGARD; THRUN, 1999). It splits the mapped environment into grid cells and estimates their beliefs based on subsequent observations. Smaller cells result in more accurate position estimates, however, since measurement updates must be executed for each cell, computational cost increases. Main advantages of this method are its global search space, enabling it to solve global localization problems, also dealing robot kidnappings, and flexibility for different motion and sensor models. However, depending on the dimension, resolution and size of the grid, the method might not be feasible for real-time applications (GUTMANN; FOX, 2002).

Other alternatives for overcoming KF-based limitations regarding global positioning involve using Multi-Hypothesis Localization (MHL), which represents uncertainty as a mixture of Gaussian's, enabling the estimation of any distribution probability, but resulting in a high computational effort (KOSE; AKIN, 2007). Another approach is to fuse advantages from KF-based and ML methods (ML-EKF) (GUTMANN, 2002), using ML for providing robustness and localization recovery to the algorithm and EKF for improving accuracy (GUTMANN; FOX, 2002).

The third class of localization algorithms, namely Monte Carlo Localization (MCL), are based on particle filters (DELLAERT et al., 1999). The particles represent multiple hypothesis of the current robot's state, their beliefs are estimated based on subsequent observations of the environment and their states are updated according to the robot's motion. MCL progressively draws samples from this particles' set, according to their beliefs, converging to a distribution that represents the robot's real state. Since the number of particles is finite, the representation is approximate, but its non-parametric property enables the algorithm to deal with a much broader space of distributions than KF-based solutions, while also requiring less computational effort than ML-based methods due to the reduced state space representation.

Particle filter-based algorithms were proven to suit a wide variety of applications (ELFRING; TORTA; MOLENGRAFT, 2021), with MCL outperforming other probabilistic methods in the global localization problem, achieving a better trade-off between accuracy, robustness, computation time, and memory consumption (DELLAERT et al., 1999; GUTMANN; FOX, 2002).

The MCL general algorithm describes how information from measurements of the environment can be fused with estimations of the robot's movements for regressing the robot's pose over time. Therefore, for each application, observation and motion models need to be specified and adjusted, since their accuracy directly impacts the designing process of the MCL. Also, the algorithm's speed and reliability are highly influenced by the number of particles utilized for representing the distribution. For highlighting the impacts of these intrinsic characteristics of the algorithm, ELFRING; TORTA; MOLENGRAFT (2021) states five main challenges that must be taken in consideration when developing a particle filter:

- Particles Degeneracy: along the iterations, some of the particles' weights might decrease to negligible values, wasting computational efforts on samples that do not represent the robot's state. This issue is commonly addressed by using resampling schemes (when to perform resampling), and algorithms (how to resample).
- Sample Impoverishment: resampling usually leads to multiple instances of the same particle. If the system's motion noise is low, the distribution tends to collapse into a single point in the state space, reducing the algorithm's robustness to measurement deviations and most likely causing it to diverge.
- Particles Divergence: particle filters are prone to diverging, i. e., converging to a distribution that does not represent the actual state of the system. Inaccurate measurements, incorrect modeling assumptions or even hardware failures are some of the reasons that cause this effect. Therefore, monitoring particles divergence is a necessary step in any particle filter, specially for real-time systems.
- Selecting the Importance Density: in mobile robots localization, the importance density is defined by a motion model, i. e., a function that reproduces the robots movements in the particles, typically added to Gaussian noise. Odometry methods are commonly applied at this step (HE et al., 2023).
- Real-time Execution: the number of particles must be sufficient to represent the desired distribution. However, increasing the size of the samples set directly affects computation costs of the algorithm, which is a hard constraint in many real-time applications.

Existent self-localization solutions from RoboCup soccer leagues propose adaptations on the original MCL algorithm for dealing the particles degeneracy, impoverishment, divergence and importance density issues (RÖFER; JÜNGEL, 2004; MUZIO et al., 2016; RÖFER et al., 2019), using low number of particles for guaranteeing real-time execution. However, compared to other leagues, SSL has the most dynamic matches with the most constrained onboard hardware, requiring solutions to reduce the computational efforts of MCL. This can be achieved by adapting the number of particles on-the-fly (ELFRING; TORTA; MOLENGRAFT, 2021), also called Adaptive Monte Carlo Localization (AMCL), and several approaches are proposed in the literature for varying the number of particles by mainly measuring the quality of the current distribution (STRAKA; ŜIMANDL, 2009).

In recent work, HE et al. (2023) presented a new Localization Confidence Estimation (LCE) method, which evaluates the current credibility of the AMCL by the matching degree between multiple laser scan points in the frame of estimated robot pose and the map. The authors also propose a model for controlling the number and quality of particles in the filter, based on the LCE result. However, the LCE algorithm is based on multiple laser measurements, which provide accurate and reliable distances. Also, the confidence value is estimated based on observations from a single iteration, which is prone to cause divergences on applications that use more unreliable measurements.

1.3 OBJECTIVES

This work proposes a novel approach for estimating the confidence of the Monte Carlo Localization over multiple iterations using reduced observations. We use this information for adapting the number of particles in a linear form, also adapting the variance of the distribution according to the algorithm's confidence, allowing the system to recover from total localization failures.

For implementing these methods, an MCL algorithm for the self-localization problem from the SSL Vision Blackout challenge was developed. The solutions were designed for integrating the architecture presented in (MELO et al., 2022), aiming at mitigating its major drawbacks. Also, for allowing offline tests and evaluations, and contributing to the SSL community, a dataset from an SSL robot navigating on the field, containing onboard-recorded and ground truth information, was generated.

In summary, with the ultimate goal of building a fully autonomous SSL robot, the present work aims at:

- Introducing a solution for estimating an SSL robot's self-localization using onboard sensing and processing only.
- Proposing a confidence metric for evaluating the localization quality during execution.
- Proposing an approach for increasing the processing speed of MCL-based self-localization algorithms.

From these specific objectives, this work's contributions can be summarized in:

- An integrated pipeline for solving the SSL self-localization problem while also detecting the environment's dynamic objects, using onboard monocular vision and inertial odometry data.
- Novel methods for estimating an AMCL distribution quality and adapting the number of particles.
- Public onboard-recorded datasets, containing multiple scenarios and ground truth data for accuracy evaluation.

- Performance evaluations of different methodologies running on an embedded system.

The following chapters detail the theoretical background, related work, our proposed methods for performing self-localization, the implemented solution and experimental setup, results from experiments, and the conclusions taken from this work. Chapter 2 presents the theoretical background. Chapter 3 depicts other researches that were used as basis for developing our methods. Chapter 4 explains our proposed self-localization infrastructure and algorithms. Chapter 5 details how we implemented a complete system for evaluating our methods. Chapter 6 presents quantitative and qualitative analysis of the solutions proposed in this work. Finally, Chapter 7 discusses the results, contributions, limitations, and future work from this research.

2 THEORETICAL BACKGROUND

This chapter presents the background knowledge that supports the proposed methods in this thesis. Firstly, we explain how the kinematics model of a four-wheeled omnidirectional robot can be derived and employed for computing its odometry based on the wheels' rotations. Then, we discuss the utilization of camera intrinsic and extrinsic parameters for converting image pixels to relative positions. Finally, the general Monte Carlo Localization (MCL) algorithm is presented, detailing how the previous subjects can be employed in its implementation and highlighting some of the algorithm's limitations.

2.1 INERTIAL ODOMETRY

Probabilistic robot localization algorithms typically rely on techniques for tracking the robot's movements along iterations (DELLAERT et al., 1999). This process is called odometry, and can be achieved by integrating the robot's velocities over time, resulting in local movements. In the case of a wheeled mobile robot, local velocities can be determined from the wheels' speeds by using the robot's kinematics model, which describes how each actuator contributes to the robot's movement (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). For example, the odometry of a four-wheeled omnidirectional robot can be estimated in four steps:

1. Measure the wheels' velocities (using optical encoders, for example).
2. Convert wheels velocities $(\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dot{\phi}_4)$ to the robot's velocities $(\dot{x}_r, \dot{y}_r, \dot{\theta}_r)$ using the kinematics model.
3. Integrate velocities over time, using the sampling period t_{sample}, resulting in local displacements $(\Delta x, \Delta y, \Delta\theta)$.
4. Update the robot's pose based on the estimated movement $pose_{new} = pose_{old} + (\Delta x, \Delta y, \Delta\theta)$, on which the $+$ operation also involves rotating the movement to the robot's coordinates.

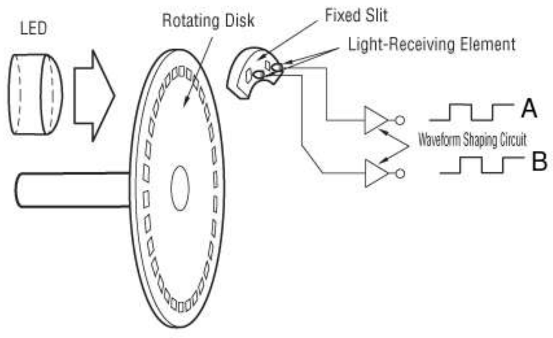
The following subsections explain in detail how each of these steps can be computed on a four-wheeled omnidirectional robot, which is the platform utilized for evaluating the proposed methods in this thesis. Note that even though the presented examples focus on a single mobile robot model, these concepts can be reproduced on other types of robots by adapting the equations (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.1.1 Measuring Velocities

Encoders are typically used to control the position or speed of wheels and other motor-driven joints in mobile robots (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). They can be classified into two types: absolute and incremental. The first disposes information regarding the absolute position of the encoder within one revolution. The second reports changes in position, not being able to keep track of absolute positions. These sensors can be implemented within different technologies: mechanical, optical, or magnetic. In this Subsection, we present an in-depth explanation of how measurements from an optical incremental encoder can be utilized for computing a wheel's speed.

A rotary encoder has two phase-shifted output signals, A and B, which are generated based on the motor shaft's rotation. For that, a rotating disk with a fine optical grid is coupled to the motor shaft, and a light source is positioned pointing towards an optical receiver, as shown in 1. The output signals are squared waveforms created from the *light* and *dark* states of the light receiver. One typical approach in mobile robotics uses quadrature encoders, on which A and B channels are shifted in 90^o.

Figure 1 – Optical quadrature encoder.



Source: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Given the encoder's resolution, which is measured by cycles per revolution (CPR), the wheel's angular velocity $\dot{\phi}$ can be computed by counting the number of pulses at a fixed sampling period t_{sample}. Due to encoders being proprioceptive sensors, i.e. they measure values in reference to their own system, they can lead to systematic errors. When applied to the problem of robot localization, for example, these errors accumulate over time, requiring significant corrections.

Another approach for making velocity measurements in mobile robots is to use Inertial Measurement Units (IMU). For example, gyroscopes, which are heading sensors that preserve their orientation in relation to a fixed reference frame, can be used for measuring the robot's angular speed $\dot{\theta}$ and, consequently, its angular movement $\Delta\theta$, by integrating speeds over time.

2.1.2 Kinematics Modeling

Forward kinematic models of motion describe how the robot as a whole moves as a function of its geometry and individual wheel behavior (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). These analytical models can be derived by expressing the kinematic constraints of individual wheels and combining them to express the whole robot's kinematic constraints.

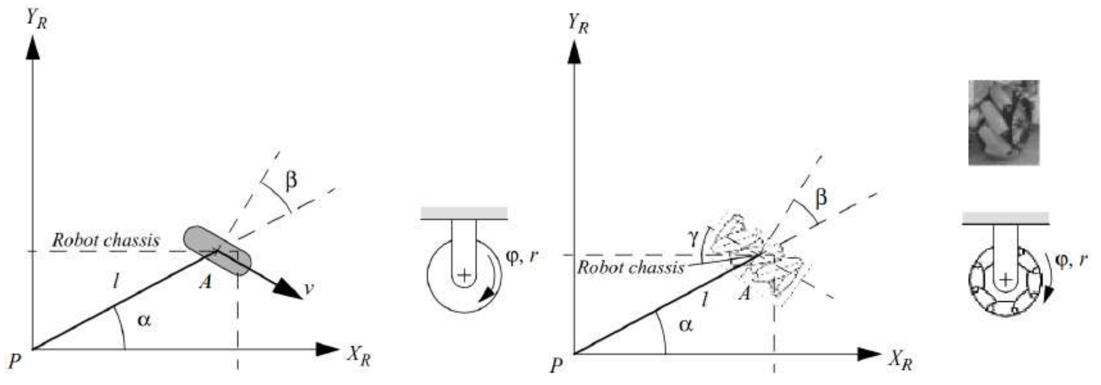
The robot employed throughout this thesis is built on top of four omnidirectional wheels, also called Swedish wheels, which consist of a fixed standard wheel with rollers attached to the wheel perimeter, allowing it to perform movements in any direction. The motion constraint is derived identically to the rolling constraint for a fixed standard wheel, except that the formula is modified by adding γ such that the effective direction along which the rolling constraint holds is along this zero component rather than along the wheel plane (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011):

$$\begin{pmatrix} sin(\alpha+\beta+\gamma) & -cos(\alpha+\beta+\gamma) & -lcos(\beta+\gamma) \end{pmatrix} R(\theta)\dot{\xi}_I - r\dot{\phi}cos(\gamma) = 0 \tag{2.1}$$

where α, β and γ correspond to the angles illustrated in Figure 2, r and $\dot{\phi}$ are the radius and angular speed of the wheel, $R(\theta)$ is a rotation matrix around the robot's orientation and $\dot{\xi}_I$ express the robot's velocities regarding the global reference frame $(\dot{x}, \dot{y}, \dot{\theta})$.

Figure 2 – Parameters of a fixed standard wheel (a) and a Swedish (omnidirectional) wheel (b).

(a) A fixed standard wheel and its parameters. (b) A Swedish wheel and its parameters.



Source: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

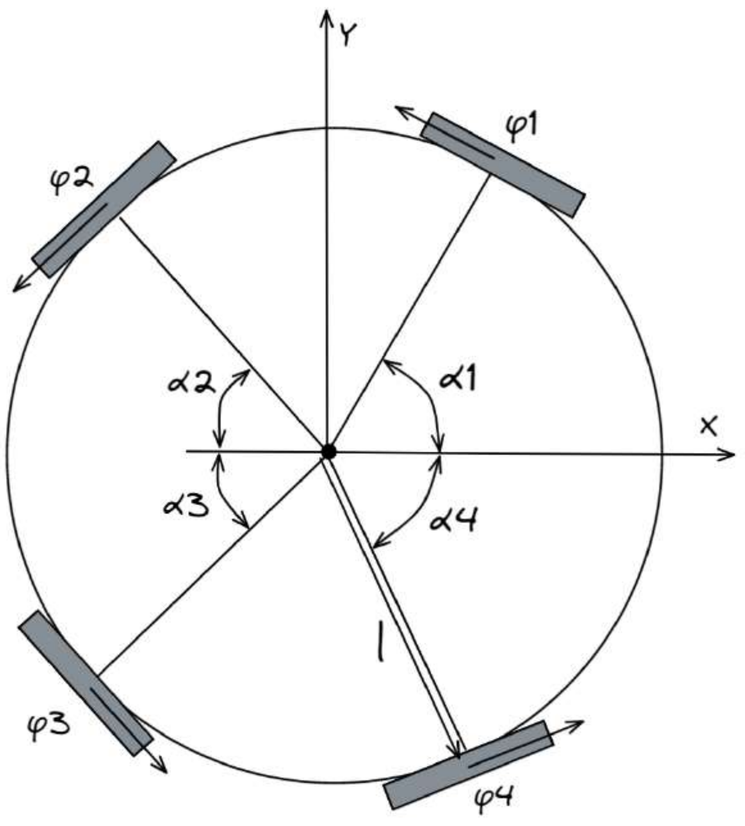
Since omnidirectional wheels impose no kinematic restrictions on the robot chassis, the kinematics constraints of a four-wheeled omnidirectional robot chassis can be computed by combining its wheels rolling constraints into a single equation:

$$J_1 R(\theta)\dot{\xi}_I - J_2\dot{\phi} = 0 \tag{2.2}$$

on which J_1 denotes a matrix with projections for all wheels to their motions along their individual wheel planes, and J_2 is a diagonal matrix whose entries are the radius of each wheel. Figure 3 illustrates the geometry of the omnidirectional robot employed throughout this work, from which Equation 2.3 can be derived.

$$\begin{pmatrix} 1/r & 0 & 0 & 0 \\ 0 & 1/r & 0 & 0 \\ 0 & 0 & 1/r & 0 \\ 0 & 0 & 0 & 1/r \end{pmatrix} \begin{pmatrix} -sin(\alpha_1) & cos(\alpha_1) & l \\ -sin(\alpha_2) & -cos(\alpha_2) & l \\ sin(\alpha_3) & -cos(\alpha_3) & l \\ sin(\alpha_4) & cos(\alpha_4) & l \end{pmatrix} \begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{pmatrix} \tag{2.3}$$

Figure 3 – Geometry representation of a four-wheeled omnidirectional robot used for deriving kinematics model.



Source: Author

Equation 2.3 enables to compute the necessary wheels' angular speeds that result in $(\dot{x}, \dot{y}, \dot{\theta})$ robot velocities regarding the global reference frame, which is called inverse kinematics. However, the opposite can also be estimated, i.e., measurements from wheels' rotations can be used for estimating the current robot velocities. This relation is called forward kinematics and can be derived by Equation 2.4.

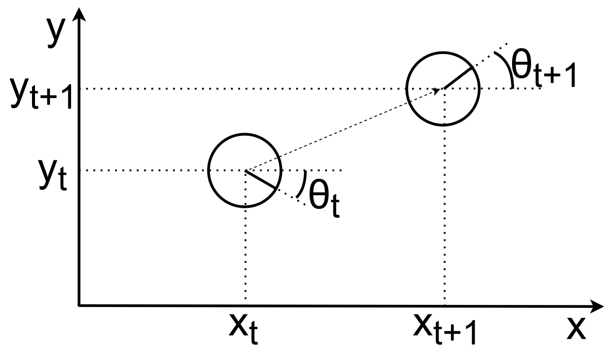
$$\dot{\xi_I} = R(\theta)^{-1} J_1^{-1} J_2 \dot{\phi} \tag{2.4}$$

Note that, for a four-wheeled omnidirectional robot, J_1 is a non-invertible matrix. Therefore, pseudo-inverse techniques can be used for estimating an approximation of J_f^{-1} (BEN-ISRAEL, 2003).

2.1.3 Integrating Velocities and Updating Poses

The forward kinematics model estimates the robot's velocities by measuring its wheel velocities. If the sampling period t_{sample} utilized during measurements is known, the robot's movement $(\Delta x, \Delta y, \Delta\theta)$ can also be computed by integrating these values.

Figure 4 – A robot moving.



Source: Author

Let us assume a four-wheeled robot knows its localization and is located at (x_t, y_t, θ_t) in a coordinates system at time t_i with measured wheels' angular speeds $(\dot{\phi}_t^1, \dot{\phi}_t^2, \dot{\phi}_t^3, \dot{\phi}_t^4)$, which we shall denote as $\dot{\phi}_t$, as illustrated in Figure 4. From Equation 2.4, the robot's current velocities $(\dot{x}, \dot{y}, \dot{\theta})$ can be derived, denoted by $\dot{\xi_I}$.

After a time $\Delta t = t_{sample}$, the robot has moved to $(x_{t+1}, y_{t+1}, \theta_{t+1})$, and its movement can be estimated by integrating $\dot{\xi_I}$ over the elapsed time. This can be calculated by assuming that t_{sample} is small enough so that $\dot{\phi}$ is constant during the given period. However, this also results that θ, and consequently $R(\theta)^{-1}$, changes with time. The rotation matrix is composed of sines and cosines regarding θ. It can be proved for both these functions that, for a small Δt, the following approximation is valid:

$$\int_0^{\Delta t} f(\theta_t + wt)dt \approx f(\theta_t + w\Delta t/2) \int_0^{\Delta t} dt = f(\theta_t + w\Delta t/2) \cdot \Delta t \tag{2.5}$$

where w denotes the robot's rotation speed $\dot{\theta}$ and $\Delta t = t_{sample}$. Thus, the resulting function from Equation 2.5 can also be expressed as $f\left(\theta_t + \frac{\dot{\theta}_t t_{sample}}{2}\right) = f\left(\frac{\theta_t + \theta_{t+1}}{2}\right)$. Finally, the robot's movement regarding the global reference can be derived from:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix} = R\left(\theta_t + \frac{\dot{\theta}_t t_{sample}}{2}\right)^{-1} J_1^{-1} J_2 \dot{\phi} \cdot t_{sample} \tag{2.6}$$

The relation from Equation 2.4 computes global velocities. However, it can be modified for expressing velocities regarding the robot's local coordinates, an shown in Equation 2.7, which can be applied to Equation 2.6 for expressing the robot's global movement, resulting in Equation 2.8.

$$\dot{\xi_R} = R(\theta)\dot{\xi_I} = J_1^{-1} J_2 \dot{\phi} \tag{2.7}$$

$$\begin{pmatrix}\Delta x \\ \Delta y \\ \Delta \theta\end{pmatrix} = R\left(\theta_t + \frac{\dot{\theta}_t t_{sample}}{2}\right)^{-1}\begin{pmatrix}\Delta x^R \\ \Delta y^R \\ \Delta \theta^R\end{pmatrix} = R\left(\theta_t + \frac{\dot{\theta}_t t_{sample}}{2}\right)^{-1}\dot{\xi}_R \cdot t_{sample} \tag{2.8}$$

This interpretation is useful for tracking the robot's movements without knowing its initial pose. In this case, we assume its initial pose to be at the origin, i.e., $(x_t, y_t, \theta_t) = (0, 0, 0)$ for $t = 0$, and update the following poses regarding this reference.

Finally, the robot's poses are updated in two steps: first, Equation 2.7 computes the robot local velocities and, secondly, the current pose is added to the local movement rotated to the global axis:

$$\begin{pmatrix}x_{t+1} \\ y_{t+1} \\ \theta_{t+1}\end{pmatrix} = \begin{pmatrix}x_t \\ y_t \\ \theta_t\end{pmatrix} + R\left(\theta_t + \frac{\dot{\theta}_t t_{sample}}{2}\right)^{-1}\dot{\xi}_R \cdot t_{sample} \tag{2.9}$$

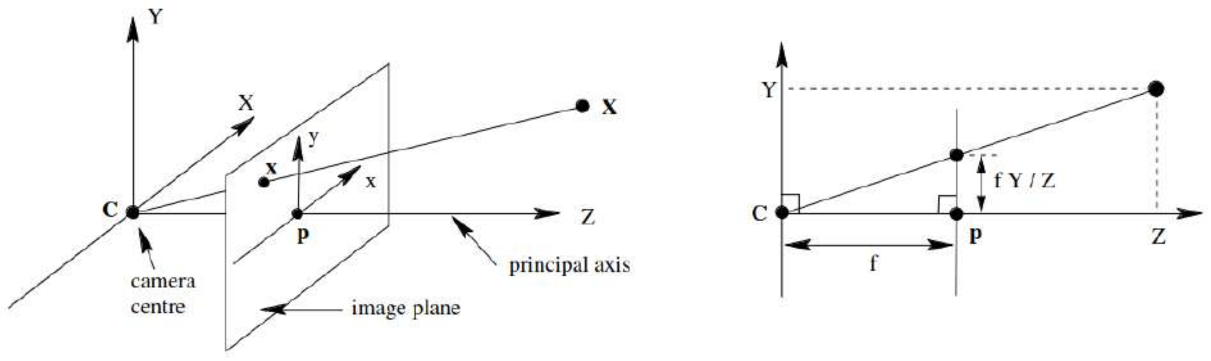
2.2 CAMERA TRANSFORMATIONS

Vision-based sensors are widely employed in mobile robotics for perceiving the environment due to their capability of providing an enormous amount of information (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011), being commonly used for computing measurements and modeling observations in self-localization problems (THRUN; BURGARD; FOX, 2005). However, a fundamental problem with visual images is that the image formation process projects 3D world points into a 2D image plane, thereby losing depth information (SZELISKI, 2010).

Retrieving relative positions from the camera image is an important step in vision-based autonomous navigation since it allows building maps, acquiring localization information, and planning actions regarding relative points and objects. 3D world coordinates can be retrieved from 2D images if additional information is provided, by using range sensors, multiple images, or previous knowledge from the environment characteristics, for example. This process is also referred to as Inverse Perspective Transformation (IPT), and can achieved by inverting the projection relations employed during image formation.

Besides the additional information, IPT requires one knowing the camera's intrinsic and extrinsic parameters and the equations that model the image projections. This Section describes these parameters and how they affect the image formation process, and explains the pinhole camera model, a commonly employed equation for projecting 3D points to the image plane. Additionally, we present a method for computing the relative positions of ground points using previously calibrated camera parameters. The algebraic formulations from the following subsections are derived mainly from (HARTLEY; ZISSERMAN, 2003).

Figure 5 – Pinhole camera geometry.



Source: (HARTLEY; ZISSERMAN, 2003)

2.2.1 Image Formation

A camera is a mapping between the 3D world (object space) and a 2D image, which can be expressed using the tools of projective geometry (HARTLEY; ZISSERMAN, 2003). In this manner, the pinhole camera model is the most specialized and simplest model for computing image projections. Figure 5 illustrates how a point in the space is projected to the image plane following this model.

Let $\boldsymbol{X} = (X, Y, Z)^\mathsf{T}$ be a point in space, C the camera center and p the principal point. In the pinhole camera model, the image point $\boldsymbol{x} = (x, y)^\mathsf{T}$ is mapped by the projection of the XC line on the image plane, or focal plane, defined by $Z = f$, where f is the camera's focal length. Note that the image plane is placed in front of the camera centre.

For now, we assume that the origin of coordinates in the image plane is at the principal point. Thus, from the similar triangles in Figure 5, the point $(X, Y, Z)^\mathsf{T}$ gets mapped to $(X/f, Y/f, f)^\mathsf{T}$. Since all points on the image are mapped to $Z = f$, this information can be ignored and the transformation

$$(X, Y, Z)^\mathsf{T} \to (fX/Z, fY/Z)^\mathsf{T} \tag{2.10}$$

describes the central projection mapping from world to image coordinates. This is a mapping from Euclidean $\mathbb{R}^3$ to $\mathbb{R}^2$. Thus, we must find a linear transformation that performs this operation.

Assuming the world and image points are represented by homogeneous vectors, the central projection can be expressed as a linear mapping between their coordinates. Therefore, Equation 2.10 can be written in terms of matrix multiplication as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \to \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.11}$$

This expression was derived by assuming that the principal point and the origin of coordinates in the image plane are coincident, which may not be true. Thus, Equation 2.11 can be rewritten as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \to \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.12}$$

The pinhole camera model assumes that the image coordinates are Euclidean coordinates having equal scales in both axial directions. Other models consider the possibility of having non-square pixels, adding a skew parameter s that represents a skewing of the pixel elements so that the x- and y-axes are not perpendicular, and different proportions for the x- and y-axes, expressed by $\alpha_x = fm_x$ and $\alpha_y = fm_y$. Similarly, the principal point can be expressed as $\bar{\boldsymbol{x}}_0 = (x_0, y_0)$, with coordinates $x_0 = m_x p_x$ and $y_0 = m_y p_y$. These considerations lead to a new equation:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \to \begin{pmatrix} \alpha_x X + sY + Zx_0 \\ \alpha_y Y + Zy_0 \\ Z \end{pmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ & \alpha_y & y_0 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.13}$$

These internal characteristics compound the so-called intrinsic parameters of a camera, which are used for expressing how points in the camera's coordinate frame are projected onto the image. As seen in Equation 2.13, this operation is performed mainly by a matrix multiplication, which is called the camera calibration matrix, expressed K as

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

then 2.13 has the concise form:

$$\boldsymbol{x} = K\,[I|\mathbf{0}]\,\boldsymbol{X}_{cam} \tag{2.15}$$

where $\boldsymbol{x}$ contains the pixel coordinates, $[I|0]$ represents a matrix divided up into a 3×3 block (the identity matrix) plus a column zero vector, and $\boldsymbol{X}_{cam}$ is the world point coordinates regarding the camera axis. In general, points in space will be expressed in terms of a different Euclidean coordinate frame, known as the world coordinate frame. Therefore, the relative rotation and translation between the world and camera coordinate frames can be added to Equation 2.15.

Assume $\tilde{\boldsymbol{X}}$ is a 3-dimensional vector representing the coordinates of a point in the world coordinate frame, and $\tilde{\boldsymbol{X}}_{cam}$ represents the same point in the camera coordinate frame. We may write $\tilde{\boldsymbol{X}}_{cam} = R(\tilde{\boldsymbol{X}} - \tilde{\boldsymbol{C}})$, where $\tilde{\boldsymbol{C}}$ represents the coordinates of the camera regarding the world and R is a 3 $\times$ 3 rotation matrix representing the orientation of the camera coordinate frame. This equation may be written in homogeneous coordinates as

$$\boldsymbol{X}_{cam} = \begin{bmatrix} R & -R\tilde{\boldsymbol{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{\boldsymbol{C}} \\ 0 & 1 \end{bmatrix} \boldsymbol{X}. \tag{2.16}$$

Putting this together with Equation 2.15 leads to

$$\boldsymbol{x} = KR[I| - \tilde{\boldsymbol{C}}]\boldsymbol{X} \tag{2.17}$$

where $\boldsymbol{X}$ now is expressed in terms of the world coordinate frame. The parameters of R and $\tilde{\boldsymbol{C}}$, which relate the camera orientation and position to a world coordinate system, are called the external or extrinsic parameters. It is often convenient not to make the camera center explicit, and instead to represent the world to image transformation as $\tilde{\boldsymbol{X}}_{cam} = R\tilde{\boldsymbol{X}} + \boldsymbol{t}$. In this case we can express the complete transformation for projecting world points into the camera image as

$$\boldsymbol{x} = K[R|\boldsymbol{t}]\boldsymbol{X}. \tag{2.18}$$

where $\boldsymbol{t} = -R\tilde{\boldsymbol{C}}$.

Lastly, as suggested by Equation 2.10, the pixel's (u, v) coordinates on the image are mapped by $\boldsymbol{p} = (u, v, 1)^T = \boldsymbol{x}/Z$. Also, the skew parameter will be zero for most normal cameras. Therefore, the Z coordinate is commonly expressed as a scale factor s, resulting in our final camera model equation employed throughout this thesis:

$$s\boldsymbol{p} = K[R|\boldsymbol{t}]\boldsymbol{X}. \tag{2.19}$$

2.2.2 Inverse Perspective Transformation

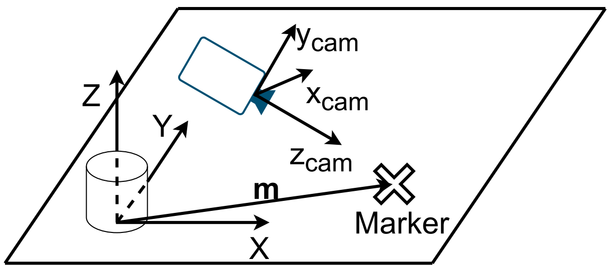
The image formation process projects 3D world points on a 2D plane. In contrast, Inverse Perspective Transformation (IPT) uses the camera model and parameters for estimating a pixel's position in the world coordinates frame. Isolating $\boldsymbol{X}$ from Equation 2.19 we derive the following:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = sR^{-1}K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} - R^{-1}\boldsymbol{t}. \tag{2.20}$$

Note that, even if K, R, and t are known, the $(X, Y, Z)^T$ position of a pixel $(u, v)^T$ can not be retrieved, because the s factor will still be unknown. Therefore, additional information is needed. One common approach is to use range sensors for measuring the distance to the given point on space, enabling to solve the equation.

Another approach for solving Equation 2.20 is to use previous knowledge of the environment characteristics (MELO; BARROS, 2023). Figure 6 illustrates a camera with a fixed translation t and rotation R regarding the coordinates system given by (X, Y, Z). The referred environment has a ground marker positioned on the floor, i.e. $Z = 0$.

Figure 6 – Camera with fixed rotation and translation regarding a coordinates system.



Source: Author.

If the camera intrinsic and extrinsic parameters are know, the marker's center position $\boldsymbol{m}$ can be retrieved from the camera image by solving Equation 2.20 for $Z = 0$ and the (u_{marker}, v_{marker}) that corresponds to the marker's center. This approach will be employed throughout this thesis for estimating relative positions of ground points on a robot soccer field regarding the robot's coordinates system.

2.3 MONTE CARLO LOCALIZATION

Monte Carlo Localization (MCL) was the chosen method for solving the self-localization problem in this research due to its ability to represent generalized distributions with a reduced state space (THRUN, 2002). It overcomes limitations from Kalman Filter-based techniques regarding global localization by using samples for representing posteriors, which maintain multiple hypotheses of the robot's state, making no assumptions of linearity or Gaussianity in the model (DOUCET; GODSILL; ANDRIEU, 2000). The algorithm also outperforms grid-based localization methods regarding memory consumption and computation speed due to its samples converging to most probable states, reducing the search space (THRUN; BURGARD; FOX, 2005).

MCL estimates probability density functions (PDF) over the robot pose using particle filters: approximate techniques for calculating posteriors in partially observable controllable Markov chains with discrete time (THRUN, 2002). The basic idea is to represent the belief $bel(x_t)$ by a set of M samples $\chi_t = \{x_t^1, x_t^2, ..., x_t^M\}$, called particles, and recursively approximate their posteriors based on updates from subsequent measurements $z_{1:t} = \{z_1, z_2, ..., z_t\}$ and predictions from control inputs $u_{1:t} = \{u_1, u_2, ..., u_t\}$ (THRUN; BURGARD; FOX, 2005). This process is solved by a Bayes filter, which works in two essential steps:

1. Calculates a belief $\bar{bel}(x_t)$ over the state x_{t-1} and control input u_t.
2. Estimate the probability that measurement z_t has been observed at state x_t.

The following subsections present more in-depth explanations of particle filters, including the basis of their mathematical derivation and how they are employed in the robot localization problem, the so-called MCL. Also, the main limitations and enhancements of the MCL algorithm are presented. The theoretical formulations from this section are derived mainly from (THRUN; BURGARD; FOX, 2005) and (DELLAERT et al., 1999).

2.3.1 The Particle Filter Algorithm

The particle filter algorithm is a nonparametric implementation of the Bayes filter, on which the key idea is to represent the posterior $bel(x_t)$ by a finite set of state samples drawn from a distribution that approximates the PDF of the system's state. The samples, also called particles, are denoted as

$$\chi_t = \{x_t^1, x_t^2, ..., x_t^M\} \tag{2.21}$$

where M denotes the number of particles and each particle x_t^M (with $1 \leq m \leq M$) represents a hypothesis of the system's state at time t. The algorithm follows the idea that regions of the state space that are more populated by samples are more probable to represent the real state. Therefore, the likelihood for a hypothesis to be drawn shall be proportional to its posterior given by the Bayes filter:

$$x_t^m \sim p(x_t | z_{1:t}, u_{1:t}). \tag{2.22}$$

The algorithm constructs the belief $bel(x_t)$ recursively from its past values, which also means that the particle set χ_t is built from its last iteration χ_{t-1}. The general particle filter workflow is shown in Algorithm 1, consisting of three main steps, which we shall denote as Prediction, Observation, and Resampling:

1. Prediction: for each particle, a hypothetical state x_t^m is generated from x_{t-1}^m based on u_t, the control input from time t, which involves sampling from a distribution that predicts state changes from control inputs $p(x_t|x_{t-1}, u_t)$. Thus, the algorithm requires a function that estimates how states are affected by the inputs.
2. Observation: then, the probability that the current measurement z_t was taken from the given state x_t^m is computed by $w_t^m = p(z_t|x_t^m)$, called importance factor. The w_t^m is usually interpreted as the weight of the m-th particle in the set. Note that, for this step, an observation model that relates measurements to states is required.
3. Resampling: finally, M particles from the temporary set $\bar{\chi}_t$ are drawn with replacement, each with probabilities proportional to their importance weights w_t^m, i.e., particles with

higher probabilities (given by the importance factor) are drawn more frequently than less probable ones. This process is also called importance resampling, and the resulting distribution χ_t represents an approximation of the posterior $bel(x_t)$.

Algorithm 1 Particle Filter Algorithm

```
Require: $\chi_{t-1}$, $u_t$, $z_t$
  $\bar{\chi}_t = \chi_t = \emptyset$
  for $m = 1$ to $M$ do
      sample $x_t^m \sim p(x_t|u_t, x_{t-1}^m)$
      $w_t^m = p(z_t|x_t^m)$
      $\bar{\chi}_t = \bar{\chi}_t + \langle x_t^m, w_t^m \rangle$
  end for
  for $m = 1$ to $M$ do
      draw $i$ with probability $\propto w_t^i$
      add $x_t^i$ to $X_t$
  end for
  return $\chi_t$
```

2.3.2 Mathematical Formulation

The Algorithm 1 is derived by addressing a state estimation problem with a Bayesian filter, which is based on the Bayes theorem of conditional probabilities, stated by

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{2.23}$$

from which also follows the theorem of total probability:

$$p(x) = \int p(x|y)p(y)dy \tag{2.24}$$

We shall think that each particle represents a sequence of states $x_{0:t}^m = x_0^m, x_1^m, ..., x_t^m$ and the particle filter must calculate the posterior over the whole sequence $bel(x_{0:t}) = p(x_{0:t}|u_{1:t}, z_{1:t})$ based on the inputs and measurements of the system along the iterations. We start by applying the Bayes rule to the target posterior:

$$p(x_t|z_{1:t}, u_{1:t}) = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \tag{2.25}$$

on which the denominator only depends on measurements, being the same for all states, thus we denote it as $p(z_t|z_{1:t-1}, u_{1:t}) = 1/\eta$. In addition, the Bayes filter algorithm assumes that the state holds all the necessary information for predicting measurements, i.e., the current measurement does not depend on past measurements or inputs, only on the current state. This assumption results in:

$$p(z_t|x_t, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \tag{2.26}$$

which combined with Equation 2.25, follows to:

$$p(x_t|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t}). \tag{2.27}$$

Note that, in this equation, $p(x_t|z_{1:t}, u_{1:t})$ represents our target distribution, and $p(x_t|z_{1:t-1}, u_{1:t})$ a proposal distribution, which can be expanded by 2.24, resulting in:

$$p(x_t|z_{1:t-1}, u_{1:t}) = \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})p(x_{t-1}|z_{1:t-1}, u_{1:t})dx_{t-1}. \tag{2.28}$$

Following the assumption that the current state is only affected by the previous one, and does not depend on past measurements and inputs, which yields $p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$, and that future control inputs do not affect past states, allowing us to omit u_t from $p(x_{t-1}|z_{1:t-1}, u_{1:t})$, we get:

$$p(x_t|z_{1:t-1}, u_{1:t}) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|z_{1:t-1}, u_{1:t-1})dx_{t-1}. \tag{2.29}$$

Note that $p(x_{t-1}|z_{1:t-1}, u_{1:t-1})$ represents the belief at the previous step $bel(x_{0:t-1})$ and that the prediction phase in the particle filter executes the exact same operation from 2.29. Then, in the observation step, the algorithm computes 2.27 based on the resulting proposal distribution $p(x_t|z_{1:t-1}, u_{1:t})$ and the observation model $p(z_t|x_t)$, resulting in a new representation of the target distribution $p(x_t|z_{1:t}, u_{1:t})$.

2.3.3 Monte Carlo Localization

Monte Carlo Localization (MCL) employs the particle filter algorithm for approximating a robot's state, defined by its pose (position and orientation), regarding a map of the environment. The belief $bel(x_t)$ is represented by a set of M particles $\chi_t = \{x_t^1, x_t^2, ..., x_t^m\}$. The general MCL algorithm is presented in 2, which is obtained by substituting the appropriate probabilistic motion and perceptual models into the particle filter. The prediction step uses particles from the current distribution as starting points and a model of the robot's motion for sampling. Then, the observation model is applied to the predicted states, estimating their importance weights'. Particles are initialized with an uniform importance factor $1/M$.

The motion model assumes the function of the predictive step from the particle filter algorithm, which describes how particle's state evolve based on current control inputs u_k. In MCL, the state is defined by the robot's pose, thus, a prediction correspond to the robot's movement. THRUN; BURGARD; FOX (2005) describes two types of motion models for mobile robots operating in the plane. The first predicts the robot's movements based on received velocity commands, while the second assumes that odometry measurements can be retrieved while the robot is moving, usually obtained by integrating wheel encoders information. Both models suffer form drift and slippage, but odometry ones are suggested as more accurate options, since they are calculated based on measured velocity values.

Algorithm 2 Monte Carlo Localization

```
Require: $\chi_{t-1}$, $u_t$, $z_t$
  $\bar{\chi}_t = \chi_t = \emptyset$
  for $m = 1$ to $M$ do
      $x_t^m$ = sample_from_motion_model$(u_t, x_{t-1}^m)$
      $w_t^m$ = measurement_model$(z_t|x_t^m)$
      $\bar{\chi}_t = \bar{\chi}_t + \langle x_t^m, w_t^m \rangle$
  end for
  for $m = 1$ to $M$ do
      draw $i$ with probability $\propto w_t^i$
      add $x_t^i$ to $X_t$
  end for
  return $\chi_t$
```

For sampling from the odometry motion model, rotation and translation are assumed as independent movements and random noises are added to the measured values before predicting the particle's new state. Algorithm 3 illustrates how this procedure can be performed, on which $u_t = (\bar{\Delta x}, \bar{\Delta y}, \bar{\Delta \theta})$ represents the estimated movements from the odometry, and $x_{t-1} = (x, y, \theta)$ accounts for the previous particle state. δ_{rot1} and δ_{trans} are derived from the robot's translation, and δ_{rot2} from rotation. The **sample**(b) function generates a random value from a zero-centered distribution (e.g. Gaussian, or triangular) with variance b, therefore these movements are added to random noises with variances proportional to the distance traveled and rotation performed, according to the α_n factors, which are adjusted from the robot's behaviour. The particle state is updated by adding the estimated movements to the current state with orientation correction.

Algorithm 3 Sample From Motion Model

```
Require: $x_{t-1}$, $u_t$
  $\delta_{rot1} = atan2(\bar{\Delta y}, \bar{\Delta x}) - \bar{\Delta \theta}$
  $\delta_{trans} = \sqrt{\bar{\Delta x}^2 + \bar{\Delta y}^2}$
  $\delta_{rot2} = \bar{\Delta \theta} - \delta_{rot1}$

  $\hat{\delta}_{rot1} = \delta_{rot1} - \textbf{sample}(\alpha_1 \delta_{rot1}) + \alpha_2 \delta_{trans}$
  $\hat{\delta}_{trans} = \delta_{trans} - \textbf{sample}(\alpha_3 \delta_{trans} + \alpha_4(\delta_{rot1} + \delta_{rot2}))$
  $\hat{\delta}_{rot2} = \delta_{rot2} - \textbf{sample}(\alpha_1 \delta_{rot2} + \alpha_2 \delta_{trans})$

  $x' = x + \hat{\delta}_{trans} cos(\theta + \hat{\delta}_{rot1})$
  $y' = y + \hat{\delta}_{trans} sin(\theta + \hat{\delta}_{rot1})$
  $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

  return $x_t = (x', y', \theta')$
```

Next in the MCL algorithm, the measurement model is employed for updating the particle's importance weight. For visual sensors, this process is typically modeled by using projective geometry to estimate relative range r and bearing ϕ of mapped landmarks from the robot's local coordinate frame. The particle's likelihood $w_t^m = p(z_t|x_t^m)$ is computed as a similarity between the measured and expected values, typically assuming that their errors follow Gaussian distributions, i.e., the more similar the values are, the more likely that state represents the true robot's state.

With the updated state x_t^m and likelihood w_t^m, the weighed sample is added to the proposal distribution $\bar{\chi}_t$. After repeating this process for the M particles, importance resampling is performed, which consists of drawing M samples from $\bar{\chi}_t$, each with probability proportional to the given weight, and adding them to the target distribution χ_t, which shall converge to the robot's true posterior over time.

2.3.4 MCL Properties and Augmentation

The main advantage of MCL compared to other localization methods is its suitability for almost any kind of distribution, mostly due to its non-parametric nature (THRUN; BURGARD; FOX, 2005). The algorithm's accuracy and processing speed are directly affected by number of particles, therefore, these characteristics can be traded off during execution by adapting the number of samples.

The native MCL algorithm also suffers from divergence problems, i.e., if it converges to the wrong state, the filter is unable to recover. This problem arises specially when a small number of particles (e.g. $M = 50$) is employed, which is a common condition in hardware constrained applications. Thus, THRUN; BURGARD; FOX (2005) suggests an approach to estimate the localization accuracy, measured by the mean of importance weights:

$$w_{avg} = \frac{1}{M} \sum_{m=1}^{M} w_t^m \tag{2.30}$$

This measure shall be employed for representing the confidence, or quality, of the distribution. However, a reliable approach is to smooth w_{avg} over multiple steps, which leads to the idea of maintaining short-term w_{fast} and long-term w_{slow} averages of this value, which can be interpreted as quality measurements for the distribution over time. Those are given by:

$$w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow}) \tag{2.31}$$

$$w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast}) \tag{2.32}$$

where α_{slow} and α_{fast} are decay factors with $0 \leq \alpha_{slow} \ll \alpha_{fast} < 1$. The divergence between these factor can be utilized for adding random samples in the distribution during resampling, allowing the MCL to recover total localization failures. This approach, also called Augmented MCL, is presented in Algorithm 4, on which a random sample is added with probability $max(0, 1 - \frac{w_{fast}}{w_{slow}})$, otherwise a sample is drawn from the proposal distribution.

Algorithm 4 Augmented Monte Carlo Localization

```
Require: $\chi_{t-1}$, $u_t$, $z_t$
  $\bar{\chi}_t = \chi_t = \emptyset$
  for $m = 1$ to $M$ do
      $x_t^m =$ sample_from_motion_model$(u_t, x_{t-1}^m)$
      $w_t^m =$ measurement_model$(z_t | x_t^m)$
      $\bar{\chi}_t = \bar{\chi}_t + \langle x_t^m, w_t^m \rangle$
      $w_{avg} = w_{avg} + \frac{1}{M} w_t^m$
  end for
  $w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$
  $w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$
  for $m = 1$ to $M$ do
      if with probability $max(0, 1 - \frac{w_{fast}}{w_{slow}})$ then
          add random pose to $\chi_t$
      else
          draw $i$ with probability $\propto w_t^i$
          add $x_t^i$ to $X_t$
      end if
  end for
  return $\chi_t$
```

3 RELATED WORK

This work aims at solving the robot self-localization problem in the SSL environment using only onboard sensing and processing. Therefore, this chapter discusses existent solutions for this problem applied at other RoboCup soccer leagues, presenting insights on how to implement and enhance the fundamental steps of the MCL algorithm. Ideas from some of these approaches were fused for modeling observations, motion, and particles resampling in this thesis.

Approaches for the self-localization problem from RoboCup researches do not present solutions for adapting the MCL set sizes. The number of particles directly affects the distribution quality and processing speed of the particle filter. Thus, methods for adapting the sample set size are discussed, highlighting the main challenges for developing an adaptive particle filter. Ideas from these work were used for implementing the proposed methods in this thesis.

3.1 SELF-LOCALIZATION IN ROBOCUP SOCCER LEAGUES

As pointed by ELFRING; TORTA; MOLENGRAFT (2021), when designing a particle filter-based solution for a specific problem, the environment and robot's constraints and requirements must be considered. Therefore, to implement an MCL algorithm for estimating a robot's pose on a soccer field, specific motion and observation models have to be defined according to the behaviours of the utilized sensors, actuators and methods utilized for making measurements and estimating the robot's movements.

Robot soccer fields are composed mainly of a green floor, white lines, and goals. However, RoboCup leagues employ varied field, goals and lines sizes, according to their robots capabilities and the objectives of the League. Also, some leagues include additional references that can be used for helping robot's localization, such as beacons, colored goal posts, or boundary walls. The following subsections describe how these static objects and landmarks are typically used for modeling observations in self-localization solutions. Besides, since each League employs a different robot, with distinct movement capabilities, their specific motion models and techniques for enhancing resampling schemes are presented as well.

3.1.1 Sony Four-Legged Robot League

The Sony Four-Legged Robot League (SFRL) was the first RoboCup competition to employ standardized fully autonomous robots (VELOSO et al., 1998), being extinguished in 2009 and lately referred to as Standard Platform League (SPL). Games were played by teams of up to five AIBO robots (abbreviated from **A**rtificial **I**ntelligence Ro**BO**t), with highly constrained sensing and processing capabilities (KITANO et al., 1998). Therefore, solutions for onboard perception, localization, decision-making and acting must be robust and efficient.

Throughout the years, the League's field size increased from approximately 3.5 x 2 to 7.5

x 5 meters and, besides the colored goals and white landmarks, the field also had walls on the borders and colored beacons. RÖFER; JÜNGEL (2004) proposes an approach for acquiring localization information by detecting points of these elements inside grid lines. The grids are composed of vertical and horizontal lines of the image, where each line is scanned pixel by pixel performing color segmentation. After pixels are classified, a finite state machine decides whether objects or landmarks exist in that line, also determining which they correspond to and their edge points. The detected edge points are converted to relative angles in the robot's coordinates system, using the onboard camera's extrinsic parameters.

The field points' relative positions are fused with odometry information in an MCL algorithm for estimating the robot's self-localization. For that, motion and observation models are defined. The first, expresses the effects of actions on the robot's pose. The second describes the probability for taking certain measurements at certain locations. RÖFER; JÜNGEL (2004) summarizes the complete algorithm pipeline in a four steps loop:

1. Particles are moved according to the motion model of the previous action of the robot.
2. Probabilities q_i are determined for all particles on the basis of the observation model for the current sensor readings.
3. Resampling is performed, moving more particles to the locations of samples with a high probability.
4. The average of the probability distribution is determined, representing the best estimation of the current robot pose.

Motion Model. For motion modeling, as their odometry computes rough estimations of the robot's movements, the authors suggest adding a random error Δ_{error} that depends on the distance traveled and the rotation performed since the last self-localization. The new particles' poses are computed by: $pose_{new} = pose_{old} + \Delta_{odometry} + \Delta_{error}$, where $+$ operations involve coordinates rotation to the robot's axis.

Observation Model. The authors lately propose an improved observation model, on which separate probabilities for beacons and goals, horizontal field lines, vertical field lines, field walls, and goal edges are represented (RÖFER; LAUE; THOMAS, 2006). Only the probabilities for edge points that were actually detected are taken into account for computing the overall similarities.

For beacons and goals, a normalized difference between the measured $\omega_{measured}$ and expected $\omega_{expected}$ angles is computed by $d = \frac{|\omega_{measured} - \omega_{expected}|}{\pi}$, and applied to a sigmoid function to determine the similarity s:

$$s(\omega_{measured}, \omega_{expected}) = \begin{cases} e^{-50d^2} & \text{if } d < 1 \\ e^{-50(2-d)^2} & \text{otherwise} \end{cases} \tag{3.1}$$

The probability $q_{landmarks}$ is computed by the product of these similarities:

$$q_{landmarks} = \prod_{\omega_{measured}} s(\omega_{measured}, \omega_{expected}) \tag{3.2}$$

For field lines, borders and goals, similarities are determined from the measured angle $\omega_{measured}$ and the expected angle $\omega_{expected}$ for a certain pose by applying a sigmoid function to the difference of both angles weighted by a constant σ:

$$s(\omega_{measured}, \omega_{expected}, \sigma) = e^{-\sigma(\omega_{measured} - \omega_{expected})^2} \tag{3.3}$$

If α and β refer to vertical and horizontal angles, respectively, and $|v|$ the robot's speed absolute value, the overall similarity for lines, borders and goals edge points are calculated from:

$$q_{edgeType} = s(\alpha_{measured}, \alpha_{expected}, 10 - 9\frac{|v|}{200}) \cdot s(\beta_{measured}, \beta_{expected}, 100) \tag{3.4}$$

The proposed method reduces the weight of observations the faster the robot walks. Also, for reducing the computing costs, only three points of each edge type, if detected, are randomly selected for estimating the samples' similarities. Another enhancement proposed by the authors is to limit the change of the probability of each sample for each edge type, guaranteeing a more stable distribution. Therefore, edge similarities are updated as follows:

$$q_{new} = \begin{cases} q_{old} + \Delta_{up} & \text{if } q > q_{old} + \Delta_{up} \\ q_{old} - \Delta_{down} & \text{if } q < q_{old} - \Delta_{down} \\ q & \text{otherwise.} \end{cases} \tag{3.5}$$

For landmarks, $(\Delta_{up}, \Delta_{down})$ is $(0.1, 0.05)$, for edge points, it is $(0.01, 0.005)$.

Finally, the overall probability for a sample is computed by:

$$q = q_{landmarks} \cdot q_{longitudinalLines} \cdot q_{latitudalLines} \cdot q_{fieldWalls} \cdot q_{goals} \tag{3.6}$$

Resampling. In this step, samples are copied from the old distribution to the new one, according to their probabilities q_i, thus, more probable samples are copied more often and improbable samples are removed. After choosing the samples, the authors suggest locally moving them, where the less probable a sample is, the more it gets moved (RÖFER; JÜNGEL, 2004), according to the following equation:

$$pose_{new} = pose_{old} + \begin{pmatrix} \Delta_{trans}(1-q) \times rnd \\ \Delta_{trans}(1-q) \times rnd \\ \Delta_{rot}(1-q) \times rnd \end{pmatrix} \tag{3.7}$$

where rnd is a number between -1 and 1 generated by a uniform distribution, and values for Δ_{trans} and Δ_{rot} are typically 20 cm and 30º.

Estimating the Robot's Pose. The authors propose discretizing the (x, y, θ) state space into 10 × 10 × 10 cells, and searching for the 2 × 2 × 2 region that contains the maximum number of samples. Among the selected cluster, the (x_{robot}, y_{robot}) coordinates of the robot are determined by the samples' average, while the angle θ_{robot} is calculated as the orientation of the sum of all direction vectors:

$$\theta_{robot} = atan2\left(\sum_i \sin(\theta_i), \sum_i \cos(\theta_i)\right) \tag{3.8}$$

Reported quantitative results show that the approach was able to localize the robot with approximately 10 cm errors (less than 5% of the field's length) using 100 samples for the particle filter. Besides, the proposed method's reliability was demonstrated throughout RoboCup editions, also showing that beacons are unnecessary for playing soccer.

Several concepts introduced in this League were migrated for solving self-localization problems at other RoboCup soccer competitions (LAUE; RÖFER, 2006). Also, the highly constrained hardware and field similarities to SSL (with walls on the borders, similar goals and dimensions) suggest that solutions from the SFRL should fit the SSL environment. However, the fast-paced SSL matches demand high processing speeds, which can be improved by reducing the number of particles of the distribution.

3.1.2 Standard Platform League

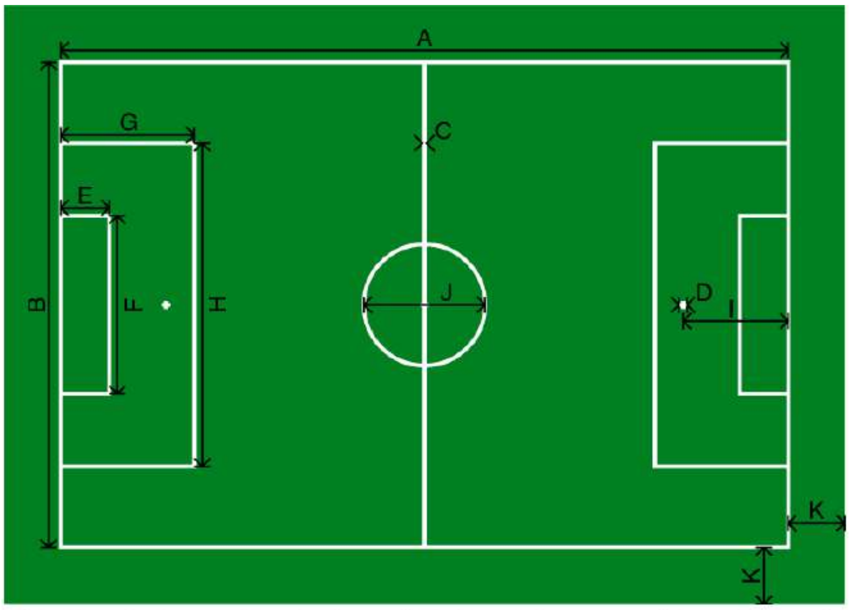
Four-legged robots were adopted as a standard due to their walking control being easier than for biped robots (VELOSO et al., 1998), leaving teams to focus on addressing other research problems. This approach was succeeded by the introduction of Two-Legged Competition, mainly referred to as Standard Platform League (SPL), with the Nao robot as a standard platform (GOUAILLIER et al., 2008; ROBOTICS, 2018).

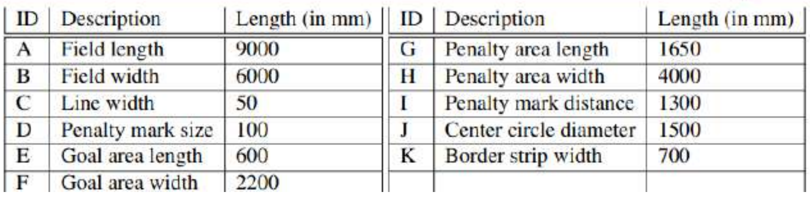
The SPL currently operates in two divisions: Champions Cup (CC) and Challenge Shield (CS), which compete in 7 vs. 7 and 5 vs 5 matches, respectively. The field consists of 8 mm artificial turf mounted on a flat wooden base with a total area of length 10.4 m and width 7.4 m and specifications shown in Figure 7. The goals are 1500 mm long, composed of goalposts and a crossbar made from 3 white cylinders with a diameter of 100 mm. In comparison to the SFRL, there are no colored references for recognizing field sides, and there are no walls on the borders, which difficulties detecting them. In contrast, the field lines and goal posts are thicker, making them easier to detect.

Robust and precise self-localization are important requirements in the SPL and straightforward textbook implementations from the particle filter algorithm are utilized for addressing this challenge (THRUN; BURGARD; FOX, 2005). Even though the solutions achieve robust, precise, and efficient self-localization, the results are directly affected by the quality and quantity of the incoming perceptions (RÖFER et al., 2019).

Recent work in the League propose methods for detecting balls, robots and field borders using Deep Learning-based solutions (RÖFER et al., 2022). In contrast, the field line detection relies on a grid of horizontal and vertical scan lines, which pixel colors are segmented in green, white and a generic color for the rest, similarly to the approaches employed in the SFRL, except that solutions for lighting-independent detection are also proposed (RÖFER et al., 2022). Previously to the proposed Deep Learning-based techniques, all visual knowledge

Figure 7 – Schematic diagram of the soccer field (not to scale) and corresponding dimensions in mm from the Standard Platform League (SPL).





ID	Description	Length (in mm)	ID	Description	Length (in mm)
A	Field length	9000	G	Penalty area length	1650
B	Field width	6000	H	Penalty area width	4000
C	Line width	50	I	Penalty mark distance	1300
D	Penalty mark size	100	J	Center circle diameter	1500
E	Goal area length	600	K	Border strip width	700
F	Goal area width	2200			

Source: (ROBOCUP, 2023)

relevant for position estimation were based on the field line detection (RÖFER et al., 2019).

The image processing system recognizes points on field lines and points on edges between the field and the goals. For estimating particles' similarities in the MCL algorithm, each point is projected to the field, given the current pose of the sample and the current pose of the camera relative to the robot (LAUE; RÖFER, 2007), enabling to compute the (x_{error}, y_{error}) between estimated and expected relative coordinates, where the first roughly corresponds to the distance error, and the second corresponds to the bearing error. Both error distances are transferred back into pixel distance errors by dividing them by the forward distance to the measured point $x_{relative}$ plus the height of the camera above the ground h. The weight w_t^m is determined by modeling the errors of the (x, y) coordinates as Gaussian normal distributions based on the computed error values and a standard deviation σ:

$$w_t^m = w_t^m \cdot \mathcal{N}\left(\left|\frac{x_{error}}{h + x_{relative}}\right|, \sigma\right) \cdot \mathcal{N}\left(\left|\frac{y_{error}}{h + x_{relative}}\right|, \sigma\right) \tag{3.9}$$

Such as in (RÖFER; LAUE; THOMAS, 2006), a method for avoiding strong oscillations in similarities is utilized, by filtering the particle's probability over n frames (typically 60) (RÖFER et al., 2019):

$$w_{particle}^{new} = \frac{w_{particle}^{old} \times (n-1) + w_{current}}{n} \tag{3.10}$$

LAUE; RÖFER (2007) also proposes a motion model based on the robot's odometry that adds noises to the movements, computed as follows:

$$\begin{pmatrix} x_t^m \\ y_t^m \end{pmatrix} = \begin{pmatrix} x_{t-1}^m \\ y_{t-1}^m \end{pmatrix} + R_{t-1}^m \begin{pmatrix} \Delta x_t + sample(max(\lambda_+ \Delta x_t, \lambda_- \Delta y_t, \lambda_n \bar{w}_{t-1}^m)) \\ \Delta y_t + sample(max(\lambda_+ \Delta y_t, \lambda_- \Delta x_t, \lambda_n \bar{w}_{t-1}^m)) \end{pmatrix} \tag{3.11}$$

$$\theta_t^m = \theta_{t-1}^m + \Delta\theta_t + sample(max(\lambda_\theta \Delta\theta_t, \lambda_d |(\Delta x_t, \Delta y_t)|, \lambda_r \bar{w}_{t-1}^m)) \tag{3.12}$$

where R_{t-1}^m is the rotation matrix corresponding to θ_{t-1}^m, $sample(x)$ is a function that returns a random value in the interval $[-x, x]$, and all λ are factors that scale the noise ratio depending on the robot's motion and the current weighting of the sample. Instead of limiting the noise ranges between fixed values, as in (RÖFER; JÜNGEL, 2004), the authors suggest scaling the range based on how the weighting of an individual sample relates to the average of all M samples:

$$\bar{w}_{t-1}^m = max\left(\frac{\sum_i w_t^i}{M w_t^m} - 1, 0\right)^2 \tag{3.13}$$

Using this approach, samples with a less than average weighting are moved even if the robot is not in motion at all, allowing the samples to move toward the real position of the robot. The authors report employing 100 particles in the implementation. However, as we shall present in Chapter 6, this set size results in high computation times and the algorithm's speed can be improved by adapting the number of particles based on the current confidence.

3.1.3 Simulation 3D League

RoboCup Soccer 3D Simulation League consists of a simulation environment of a soccer match with two teams, each one composed by up to 11 simulated Nao robots, the official robot used in the SPL since 2008. RoboCup Soccer 3D competition allows the possibility for enhancements in the design and implementation of multi-agent high-level behaviors at the same time it provides a solid low level platform for a realistic physical simulation of the game (MUZIO et al., 2016).

Researches in this League propose a method to extract landmarks by matching observed lines and actual lines and selecting the pair with highest likelihood, eliminating ambiguities (MUZIO et al., 2016). The work also presents an observation model for landmarks (goal posts and corner flags) that uses relative angles and distances for computing similarities:

$$p(Z_t^{landmark} | x_t^{[m]}) = \prod_j exp\left(-\frac{(d_j - \hat{d}_j^{[m]})^2}{2\sigma_d^2}\right)\left(-\frac{(\psi_j - \hat{\psi}_j^{[m]})^2}{2\sigma_\psi^2}\right) \tag{3.14}$$

In this model, $p(Z_t^{landmarks} | x_t^{[m]})$ represents the probability of a measurement $Z_t^{landmarks}$ being made from a state x, which is computed from the measured d_j and expected $\hat{d}_j^{[m]}$ relative distances, and respective relative angles ψ_j and $\hat{\psi}_j^{[m]}$. σ_d and σ_ψ are noise parameters generated by the simulation server. In the SSL, relative angles and distances can be used for modeling observations more reliably than relative x, y coordinates and we take insights from Equation 3.14 for implementing our proposed methods.

3.2 ADAPTIVE MONTE CARLO LOCALIZATION

The number of particles utilized affects the MCL algorithm's speed and quality. If the number is high, it is more likely to converge to the correct state in a few iterations, but its computational complexity increases. Otherwise, i.e., if a small sample set is used, the algorithm runs faster but is more likely to diverge (FOX, 2003). One effective approach for improving the efficiency of particle filters is to adapt the number of samples over time.

As presented by FOX (2003), in the beginning of a global localization task, the robot is highly uncertain of its state and a large number of samples is needed to accurately represent its belief. On the other extreme, once the robot knows where it is, only a small number of samples suffices to accurately track its position. The typical approach of employing a fixed number of samples requires enough particles for solving both global localization and position tracking problems, resulting in a distribution with a high number of samples collapsed in similar states.

Adaptive particle filters focus on changing the sample size on-the-fly, saving computational resources, while also maintaining a number of particles capable of representing the current distribution of the particle filter. For that, the algorithm must be able to evaluate the quality of the current distribution and estimate the desired number of samples. There are several proposed methods for adapting particle filters sizes (STRAKA; ŜIMANDL, 2009), and the following subsections focus on presenting two methods: the first estimates the ideal number of particles for the distribution based on the Kullback-Leibler Divergence (KLD) between the target distribution and the current distribution of the particle filter; the second injects or removes particles from the filter based on comparisons between the current robot measurements and expected measurements by the resulting position from the AMCL algorithm.

3.2.1 KLD-Sampling Particle Filter

FOX (2003) introduces a likelihood-based adaptation technique, i.e., it adapts the number of particles based on the likelihood of observations. The intuition behind this approach is as follows: if the measured and expected values from observations have high similarity, importance weights are large and the sample set remains small (typical case during position tracking), otherwise, if measurements do not match expected values, as is the case when the robot's global position is uncertain or it lost track of its position, the sample weights are small and the sample set becomes large.

The key idea to formulate the approach is as stated as:

"At each iteration of the particle filter, determine the number of samples such that, with probability $1-\delta$, the error between the true posterior and the sample-based approximation is less than ε."

It computes the error between the target PDF $\hat{p}$ and its sample based approximation p using the Kullback-Leibler distance (KL-distance) (COVER; THOMAS, 2006), a function that measures the difference between probability distributions (or densities), which in case is given by:

$$D_{KL}(\hat{p},p) = \sum_{\mathbf{x}_i} \hat{p}(\mathbf{x}_i|\mathbf{z}^i) log \frac{\hat{p}(\mathbf{x}_i|\mathbf{z}^i)}{p(\mathbf{x}_i|\mathbf{z}^i)}. \tag{3.15}$$

Assuming that the target PDF $\hat{p}$ can be represented by a piece-wise discrete PDF with k different bins, $\hat{p} = (\hat{p1}, \hat{p2}, ..., \hat{p}_k)$ represents a vector containing the true probability of each bin. The sample based approximation p approximates these probabilities using n particles. The authors present an analytical proof that, with probability $1-\delta$, the KL-distance between the target and the true distributions is less than ε, if we choose the number of samples n as:

$$n = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta}, \tag{3.16}$$

where χ^2_{k-1} is a χ^2 (chi-squared) distribution with $k-1$ degrees of freedom. For determining n according to 3.16, the Wilson-Hilferty transformation (JOHNSON; KOTZ, 1970) is applied to compute the quantiles of the χ^2 distribution, resulting in:

$$n = \frac{k-1}{2\varepsilon}\left(1 - \frac{2}{9(k-1)}\sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right)^3, \tag{3.17}$$

where $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal distribution, which are available in standard statistical tables.

Algorithm 5 presents the author's proposed implementation for inserting the KLD-sampling technique into the particle filter algorithm. The number of supported bins k for the predictive distribution is updated after each sample generated. It also checks whether the minimum number of samples has been generated (typically set to 10). The sampling process is guaranteed to terminate, since for a given bin size Δ, the maximum number k of bins is limited, which also limits the maximum number n_χ of desired samples.

In the authors' experiments, KLD-sampling yields better approximations using only 6% of the samples required by the fixed approach. The method was evaluated in the context of indoor mobile robot localization using data collected by sonar and laser range-finders along with odometry measurements from a commercial Pioneer robot. The best results were achieved employing a value of 0.99 for $(1-\delta)$, and a fixed bin size Δ of 50cm x 50cm x 10º. The maximum number of samples was limited to $100,000$, and error bounds ε were varied between 0.4 and 0.015. The results show that, starting with $40,000$ samples, the algorithm reduces the number of particles to 184 on average, still achieving good tracking results (average error of 52.8cm), while the fixed approach requires 750 samples to achieve comparable accuracy.

Algorithm 5 KLD-Sampling Particle Filter

```
Require: $S_{t-1} = \{(x_{t-1}^i, w_{t-1}^i) | i = 1, ..., n\}$ representing the belief $Bel(x_{t-1})$, control input
  $u_{t-1}$, observation $z_{t-1}$, bounds $\varepsilon$ and $\delta$, bin size $\Delta$, minimum number of samples $n_{\chi_{min}}$

  Initialize $S_t := \emptyset$, $n = 0$, $n_\chi = 0$, $k = 0$, $\alpha = 0$

  while ($n < n_\chi$ and $n < n_{\chi_{min}}$) do
      // Resampling: Draw state from previous belief
      Sample an index $j$ from the discrete distribution given by the weights in $S_{t-1}$
      // Sampling: predict next state
      Sample $x_t^n$ from $p(x_t | x_{t-1}, u_{t-1})$ with given $x_{t-1}^j$ and $u_{t-1}$
      $w_t^n \leftarrow p(z_t | x_t^n)$
      $\alpha \leftarrow \alpha + w_t^n$
      $S_t \leftarrow S_t \cup \{(x_t^n, w_t^n)\}$
      if ($x_t^n$ falls into empty bin $b$) then
          $k \leftarrow k + 1$
          // Mark bin as non-empty
          $b\_has\_sample \leftarrow True$
          if $n > n_{\chi_{min}}$ then
              // Update the number of desired samples
              $n_\chi \leftarrow \frac{k-1}{2\varepsilon}\left(1 - \frac{2}{9(k-1)}\sqrt{\frac{2}{9(k-1)}} z_{1-\delta}\right)^3$
          end if
          // Update number of generated samples
          $n \leftarrow n + 1$
      end if
  end while
  // Normalize importance weights at the end
  for $i = 1, ..., n$ do
      $w_t^i \leftarrow w_t^i / \alpha$
  end for
  return $S_t$
```

This work played an important role in the study of adaptive Monte Carlo Localization, as it introduced the application of adaptive sample set sizes to the mobile robot self-localization problem, presenting an analytical method for estimating the ideal number of particles and adaptations of the original MCL algorithm for inserting this step.

3.2.2 Enhanced AMCL for Dynamic Featureless Environments

HE et al. (2023) presents a novel method for adapting the number of samples in an AMCL algorithm. The method consists of three parts:

- Patrol Iterative Closest Point (ICP): matches real and virtual laser scans generated from the map.
- Localization Confidence Estimation (LCE): estimates the localization confidence of the

AMCL and Patrol ICP.

- Adaptive Particle Injector (API): calculates the number of particles that need to be inserted into the AMCL.

The ICP is employed for minimizing the localization error by matching the scans of the current position with a set of virtual measurements at the estimated localization of the robot using a pre-built map of the environment. For obtaining the virtual scans, the algorithm draws rays from the currently estimated robot pose covering a 270-degree range with a 0.375-degree step, detecting up to 10 meters distant points.

According to the environment map and the currently estimated robot pose, the authors propose evaluating the matching degree between measured and expected laser scan values for estimating LCE. The detailed steps are described in Algorithm 6, showing that the ratio between the number of matched and total points computes LCE.

Algorithm 6 Localization Confidence Estimation Algorithm

```
Require: pre-built map $M$, laser scan reading $F_t$, search radius $r$, and $s_t$ representing estimated localization at time $t$.
Ensure: $sizeof(F_t) > 0$ and $M! = empty()$
  Construct KD-tree to represent obstacle points of grid map $M$. $LCE \leftarrow 0$, $N_t \leftarrow sizeof(F_t)$, $k \leftarrow 0$
  Get $\bar{F}_t$ by mapping the scan points $F_t$ at pose $s_t$ onto the grid map.
  for $i < N_t$ do
      if $\bar{F}_t$ has adjacent points within radius $r$ then
          $k \leftarrow k + 1$
      end if
  end for
  $LCE \leftarrow \frac{k}{N_t}$
  return $LCE$
```

Thirdly, the API module computes the number of particles to be inserted or removed based on the LCE scores of the pose calculated by Patrol ICP and the average weight of AMCL particles, denoted as $s1$ and $s2$, respectively, an adjustable coefficient parameter α_1, and the angle between the ICP matching result and the AMCL estimated poses, referred to as $head_{(p_{ICP}, p_{AMCL})}$. The number of inserted particles at a given iteration is calculated by:

$$N = \begin{cases} \alpha_1 \left(\frac{1}{1{,}001 - s_1} - \frac{1}{1{,}001 - s_2} \right) & \text{if } s_1 > s_2 \text{ and } s_1 \geq 0.6 \\ 1 & \text{if } s_2 > s_1 \text{ and } s_1 \geq 0.6 \\ 0 & \text{if } s_1 < 0.6 \text{ or } head_{(p_{ICP}, p_{AMCL})} > 90^o \end{cases} \tag{3.18}$$

The authors' results show that the proposed method was able to overcome other adaptive particle filters, being able to reduce localization drift in dynamic featureless environments. The varied sample set sizes also decreases the occurrence of robot kidnapping. The LCE calculation allows to fully exploit the information from measurements, increasing the localization accuracy, speeding up the convergence of AMCL, and also reducing the number of particles during position tracking, demonstrating the method's reliability in computing localization confidence. However, N_t measurements are employed for computing the LCE, requiring several observations at each update for achieving a reliable confidence measure. Such number of measurements might not be feasible for a resource-constrained vision-based observation model, since each measurement increases the system's overall computation time.

3.3 RELATED WORK CONSIDERATIONS

In this Chapter, we depicted past work from RoboCup soccer leagues that present techniques for improving the MCL algorithm's robustness regarding imprecision and deviations in measurements and movement estimations. However, these methods were designed to suit humanoid and standard platform (NAO and AIBO) robot soccer environments, where matches tend to be less dynamic and slower-paced compared to the SSL. Therefore, besides adapting their solutions for the SSL robots and field characteristics, we also searched for methods to increase the localization processing speed.

We reviewed two methods to reduce the computational complexity of the MCL algorithm by adapting the number of particles of the distribution during its execution. These techniques, also called Adaptive Monte Carlo Localization (AMCL), are based on the same idea: when the robot's localization is highly uncertain, a large number of samples is needed to find the correct the robot state; in contrast, as the algorithm converges, more hypotheses are eliminated from the search space, and less samples are needed to represent the robot's belief. Thus, we take insights from these work to accelerate the localization algorithm.

Table 1 summarizes the main contributions, characteristics, and limitations of the work presented in this Chapter, highlighting their impacts to the methods implemented in this thesis. We depict the robots utilized by the authors in their experiments, the environment on which the solutions were evaluated, the key insights we took from their work for improving our localization algorithm, and their drawbacks, i.e., the major limitations that restrains these solutions to be directly employed at the RoboCup SSL environment. To review how these methods compare to our work, Chapter 7 presents the key insights and drawbacks from this research in Table 7.

Table 1 – Review of main contributions and drawbacks from the related work.

Title (Author, Year)	**Platform**	**Environment**	**Key Insights**	**Drawbacks**
An Enhanced Adaptive Monte Carlo Localization for Service Robots in Dynamic and Featureless Environments (HE et al., 2023)	Service robot (model not specified) equipped with SICK TIM561 lidar	Featureless indoor corridor with dynamic objects	- Likelihood-based adaptive set sizes - Estimates the localization confidence based on the robot's measurements	- Performs multiple measurements in each iteration - Acquires measurements with a range sensor
B-Human Team Report and Code Release 2019 (RÖFER et al., 2019)	NAO Robot	RoboCup Standard Platform League	- Avoids strong oscillations in similarities by filtering the particle's probability over multiple frames	- The similarity filtering slows down the convergence
Monte Carlo Localization for Robocup 3D Soccer Simulation League (MUZIO et al., 2016)	NAO Simulation	RoboCup Simulation 3D League	- Models observations by distances and bearing angles - Projects angles from the particles to field lines	- The simulated environment provides more reliable measurements and perceptions - Measurement standard deviations are assumed to be invariant regarding the measurement distance
Particle Filter-based State Estimation in a Competitive and Uncertain Environment (LAUE; RÖFER, 2007)	Kid-Size Humanoid Robot	RoboCup Humanoid League	- Adds noises to the robot's motion model - Scales measurement errors by their measured distances	- Odometry is prone to large errors - Computes similarities based on relative x, y coordinates
Particle-Filter-Based Self-Localization Using Landmarks and Directed Lines (RÖFER; LAUE; THOMAS, 2006)	AIBO Robot	RoboCup Sony Four-Legged Robot League	- Models observations for field boundaries and goals	- Calculates the correspondences between measured and expected points using lookup tables
Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League (RÖFER; JÜNGEL, 2004)	AIBO Robot	RoboCup Sony Four-Legged Robot League	- Adds random noises to the motion model - Moves resampled particles according to their probabilities - Estimates the robot pose as the samples' average from the largest particles' cluster	- Odometry is prone to large errors - Uses fixed limits for deviations in resample - Clustering may cause strong shifts in the resulting particle state
Adapting the Sample Size in Particle Filters Through KLD-Sampling (FOX, 2003)	Pioneer II	Indoor corridors	- Calculates the number of desired samples analytically and updates during resample - Suggests adjusting the spacing between particles by the uncertainty	- Does not provide metrics for evaluating the localization confidence - Allows strong oscillations in the number of particles between iterations

Source: Author

4 PROPOSED SELF-LOCALIZATION METHODOLOGY

This chapter presents the proposed methodology for addressing the mobile robot self-localization problem in dynamic environments under hardware constrained platforms. The Monte Carlo Localization (MCL) algorithm is employed due to its ability to deal with generalized distributions, solving both global localization and position tracking problems. However, for improving its efficiency, the number of particles is adapted during execution, as presented in section 3.2, reducing the computational complexity during position tracking and enlarging the search space for global localization. The set size varies based on the current quality of the distribution, for which we propose a novel measure, computed by applying the measurement model to the resulting MCL state, given by the weighted sum of the samples states, and averaging it over multiple iterations, instead of calculating it based on several measurements from a single iteration, as suggested by HE et al. (2023).

4.1 PROPOSED SELF-LOCALIZATION PIPELINE

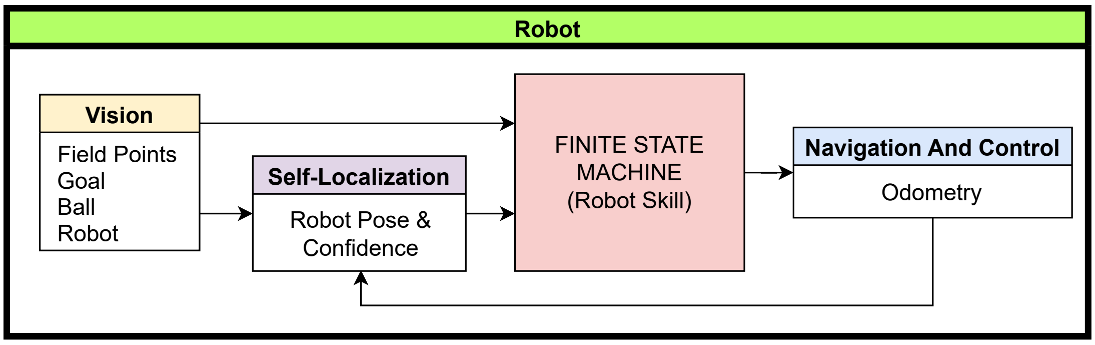
Within the researches for building an autonomous SSL robot and previously to the work presented in this thesis, we proposed an architecture for executing basic SSL soccer skills autonomously without global localization knowledge (MELO et al., 2022). However, results from experiments highlighted that this information might be the key for solving the main limitations of the system. Therefore, we present an improved pipeline that integrates self-localization to the autonomous robot functionalities, which is illustrated in Figure 8.

The architecture employs the robot's odometry, computed from the wheels' encoders and gyroscope measurements (MELO et al., 2022), for implementing the MCL motion model, which is used for propagating particles according to the robot's movement. The measurements from the environment are acquired by detecting the SSL field elements, namely the goal, field lines, and field boundaries, using onboard vision. These information are fed into the self-localization module, which regresses the robot's pose over time and computes a measure of confidence on this estimation. The outputs from self-localization and vision processing are employed by the finite state machine (FSM) which, based on desired skill to be executed, or task to be solved, makes decisions and sets the robot's desired action. Lastly, navigation and control execute the commanded action.

4.1.1 Embedded Vision

The architecture presented by MELO et al.(2022) employs a vision processing pipeline that detects SSL goals, robots, and balls and estimates their relative positions based on their projections on the ground (MELO; BARROS, 2023). In the SSL, three main classes of field elements can be used as references for localization: goals, field markings (lines), and boundaries.

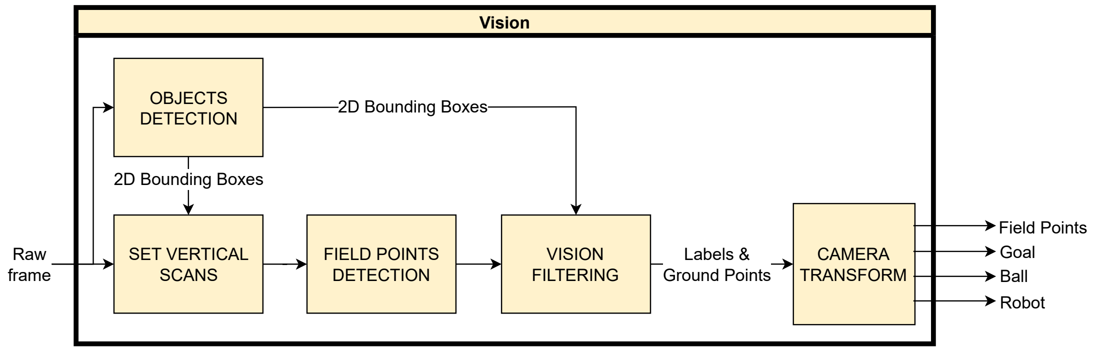
Figure 8 – Proposed system architecture.



Source: Author

Thus, we integrated field lines and boundaries detection into the vision processing pipeline, resulting in the module illustrated in Figure 9, and employed the camera transformations presented in section 2.2 for computing their relative positions.

Figure 9 – Onboard vision processing pipeline.



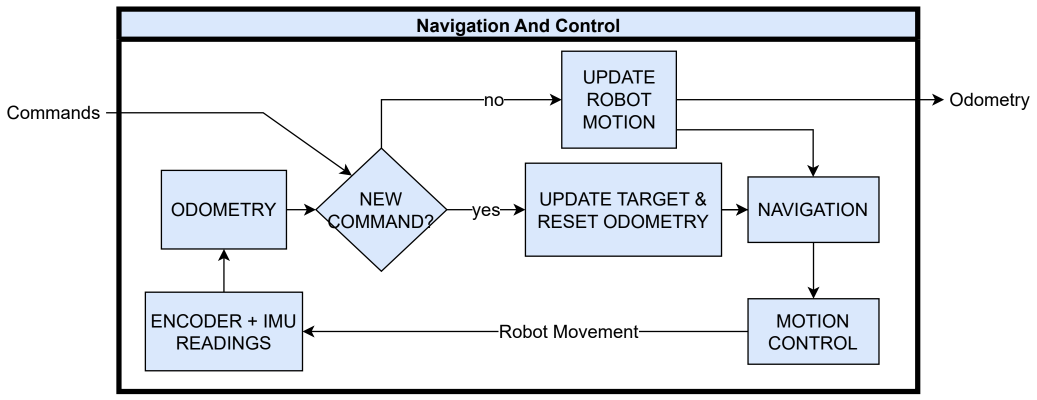
Source: Author

Vision processing starts by detecting the objects 2D boxes on the current frame. Then, grid lines on the image are defined, avoiding lines that intersect detected objects. Thirdly, the grids are scanned for detecting points from the field lines and boundaries (RÖFER; JÜNGEL, 2004). In vision filtering, the bounding boxes and the field pixels are filtered for excluding false-positives, and objects get represented by their ground-points, i.e., a pixel that approximates the object's bottom-center point. Finally, all ground-points have their relative (x, y) positions estimated using the camera transformations presented in section 2.2, by assuming they lay on the ground.

4.1.2 Navigation and Control

This module is depicted in Figure 10, executing mainly motion control and odometry estimation. It performs a loop that checks whether a new command was received from the FSM; if yes, it updates the robot's target destination and movement type that should be executed, and resets the odometry displacement. If not, the robot's pose is updated according to the odometry estimation, which is sent back to the self-localization module. Then, navigation sets the movement that should be performed for reaching the desired target, which is turned into velocity commands, controlled by the motion control. The robot's movements are estimated from the wheels' encoders and IMU readings, computing the odometry, and closing the module's loop (MELO et al., 2022).

Figure 10 – Low-level control and trajectory estimation.

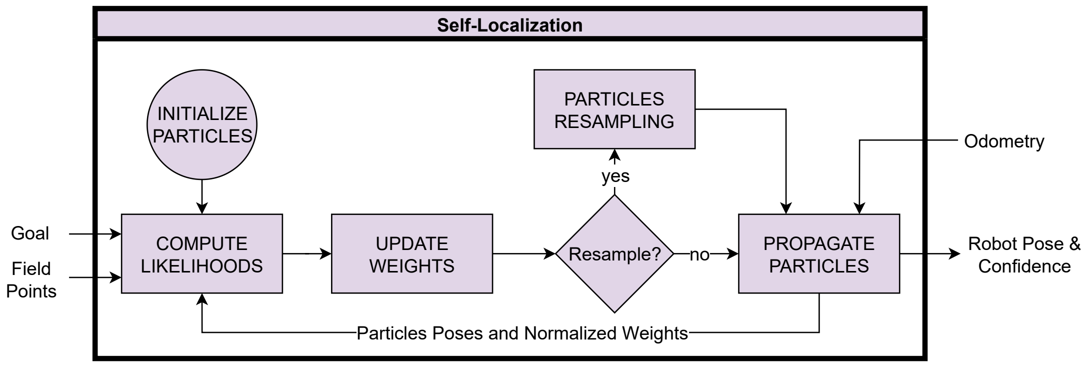


Source: Author

4.1.3 Self-Localization

The proposed self-localization technique is based on the MCL algorithm. We initialize the samples and compute their likelihoods based on the similarities between the robot measurements from the environment and their expected values from each particle state. After normalizing the particles' weights, resampling is performed if needed and the particles are propagated based on the motion model using the odometry estimation. The resulting distribution is used for approximating the robot's current pose and the algorithm's confidence.

Figure 11 – Schematic for MCL-based self-localization.



Source: Author

4.2 MCL ENHANCEMENTS BASED ON ROBOCUP RESEARCHES

Section 3.1 presents how most RoboCup soccer leagues employ the MCL algorithm for solving self-localization problems. Even though the solutions are based on straightforward textbook implementations, the teams propose minor enhancements for the general algorithm, increasing its capability to deal with erroneous measurements, inaccurate motion models, and insufficient resamplings. This section details how we incorporated some of the enhancements proposed by these researches in the original MCL algorithm.

4.2.1 Motion Model

The motion model from Algorithm 3, proposed by THRUN; BURGARD; FOX (2005), decomposes the robot's translation in polar coordinates, i.e. distance and direction, and presents a method for adding noises to the movement proportional to the distance traveled, the rotation performed, and the amount of direction change. However, researches from RoboCup leagues show that simpler models can be utilized, by adding deviations proportional to the (x, y) movements to the odometry's translation. Therefore, our motion model follows the approaches from RÖFER; JÜNGEL(2004) and RÖFER et al.(2019), and is described in Equation 4.1.

Firstly, θ_{t-1} refers to the orientation from the sample state x_{t-1}. Meanwhile, $\bar{\Delta x}$, $\bar{\Delta y}$, and $\bar{\Delta\theta}$ denote the measured motion from the odometry input u_t regarding the local reference frame. Next, the estimated movements are added to Gaussian noise, with zero mean and standard deviations proportional to the absolute value of the movement performed by adjustable factors $\gamma = (\gamma_x, \gamma_y, \gamma_\theta)$. The state x_t is updated by rotating the movement to the global reference frame and adding it to the previous x_{t-1}.

$$x_t = x_{t-1} + \begin{pmatrix} cos(\theta_{t-1}) & -sin(\theta_{t-1}) & 0 \\ sin(\theta_{t-1}) & cos(\theta_{t-1}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{\Delta x} + \mathcal{N}(0, \gamma_x|\bar{\Delta x}|) \\ \bar{\Delta y} + \mathcal{N}(0, \gamma_y|\bar{\Delta y}|) \\ \bar{\Delta\theta} + \mathcal{N}(0, \gamma_\theta|\bar{\Delta\theta}|) \end{pmatrix} \tag{4.1}$$

4.2.2 Measurement Model

In this work, measurements are acquired by distances of landmarks and lines on the soccer field relative to the robot's local reference frame. Since distance measurements are less accurate for further points, we adopt a similar approach to LAUE; RÖFER (2007), which divides the errors by the forward distance plus the camera height. Similarly, we assume the standard deviation σ_d to be proportional to the measured distance, which yields:

$$p(z_t^j|x_t^m) = exp\left[-\frac{(d_j - \hat{d}_j^m)^2}{2\sigma_d^2}\right] = exp\left[-\alpha_d\left(1 - \frac{\hat{d}_j^m}{d_j}\right)^2\right] \tag{4.2}$$

where $p(z_t^j|x_t^m)$ expresses the likelihood of the m-th particle regarding the difference between the j-th distance measurement d_j and its expected value $\hat{d}_j^m$, given the sample state x_t^m. In practice, this method reduces the weight of further measurements to the distribution. The α_d factor is an adjustable parameter for weighting the impact of observations to the distribution (RÖFER; JÜNGEL, 2004), which updates particles importance weights by:

$$w_t = w_{t-1} \prod_j p(z_t^j|x_t^m) \tag{4.3}$$

4.2.3 Resampling

Researches comparing different resampling algorithms present conflicting and inconclusive results, show that there is no best solution (ELFRING; TORTA; MOLENGRAFT, 2021). Therefore, for this step, a systematic resampling was selected, mostly due to its reduced computational complexity, however other popular schemes, such as multinomial resampling, residual resampling, or stratified resampling, could be employed as well (LI; BOLIC; DJURIC, 2015).

The samples are selected based on the systematic resampling algorithm and locally moved based on their importance weights (RÖFER; JÜNGEL, 2004). Less probable samples are moved more, following the rule from Equation 3.7. The complete resampling process is shown in Algorithm 7, on which W and X represent the weights and states of the proposal distribution, N is the total number of particles, and rnd expresses a function that samples a number between -1 and 1 according to a uniform distribution. The δ_{trans} and δ_{rot} are parameters for adjusting the pose deviation added to resampling.

Besides from choosing the resampling algorithm, it is also important to define when resampling should be performed. A common approach is to resample at each time step, however, this causes an increase in computational costs and may also lead to particles impoverishment, i.e., samples getting concentrated in a small region (ELFRING; TORTA; MOLENGRAFT, 2021). Thus, three conditions were defined for deciding if resampling is needed:

1. Samples weights' sum is low (before normalization): occurs when all particles have low likelihoods, which indicates degenerancy, i.e., the distribution may have converged to a wrong region of the state space.

2. Confidence from total odometry motion since last resampling is low: odometry accumulates errors over time and, without resampling, states are updated only by the motion model. Thus, the confidence in this distribution drops the more the robot moves, requiring resampling.

3. Confidence from a specific sample is high (after normalization): if one particle is assigned with a high importance weight after normalization (which means other are low), that sample is more likely to represent the true robot's state, and resampling will reproduce more particles in that region.

Algorithm 7 Systematic Resampling with Motion Deviation

```
Require: $W = \{w_t^n, n = 1, 2, ..., N\}$, $X = \{x_t^n, n = 1, 2, ..., N\}$, $N$

  Initialize: $\chi_t = \emptyset$, $n = 0$, $m = 0$
  $Q =$ cumulative_sum$(W)$
  $u_0 = (1/N) \times |rnd|$

  while $n < N$ do
      $u = u_0 + n/N$
      while $Q[m] < u$ do
          $m = m + 1$
      end while
      $x = X[m] + (1 - W[m]) \begin{pmatrix} \Delta_{trans} \times rnd \\ \Delta_{trans} \times rnd \\ \Delta_{rot} \times rnd \end{pmatrix}$
      $w = W[m]$
      $\chi_t \leftarrow \chi_t \cup \langle x, w \rangle$
      $n = n + 1$
  end while

  $\chi_t \leftarrow$ normalize_weights$(\chi_t)$
  return $\chi_t$
```

4.2.4 Integrating Enhancements into MCL

The previous subsections show how the motion model, measurement model, and resampling were enhanced for improving the MCL general algorithm. Algorithm 8 shows how these methods were integrated into the MCL update step, on which X and W represent the particles' states and weights, M is the number of samples, u and z are the current motion from odometry and measurements from the environment, and Δd maintains the cumulative distance traveled and rotation performed since the last resampling.

It starts by initializing the weights' sum $W_{sum} = 0$ and updating the displacement Δd. Then, each sample has its state x^m propagated according to the motion model from Equation 4.1, its likelihood $p(z|x^m)$ computed using the measurement model 4.2, and importance weight updated by 4.3, which gets added to the weights' sum W_{sum}. Consecutively, the weights W are normalized by $w^m = w^m/W_{sum}$, and, if one of the conditions for resampling is satisfied, resampling is performed with Algorithm 3.7 and Δd is reset to 0.

The presented enhancements improve the MCL robustness regarding deviations in sensor measurements and odometry, however it still maintains a fixed number of samples, which leaves room for improving the algorithm's computational costs. In addition, no confidence, or quality, evaluation metrics are presented in this method, which is a useful information not only for adapting the number of particles, but also for making decisions or planning actions (SEEKIRCHER; LAUE; RÖFER, 2011).

Algorithm 8 Enhanced Monte Carlo Localization Update

```
Require: $X = \{x^m | m = 1, 2, ..., M\}$, $W = \{w^m | m = 1, 2, ..., M\}$, $u$, $z$, $\Delta d$
  Initialize: $W_{sum} = 0$
  $\Delta d = \Delta d + |u|$
  for $m = 1$ to $M$ do
      $X[m] =$ sample_from_motion_model$(u, x^m)$
      $W[m] = w^m \times$ measurement_model$(z, X[m])$
      $W_{sum} = W_{sum} + W[m]$
  end for
  $W \leftarrow$ normalize_weights$(W)$
  if needs_resampling$(W_{sum},$ max$(W), \Delta d)$ then
      $\langle X, W \rangle \leftarrow$ systematic_resampling$(X, W, M)$
      Reset $\Delta d = 0$
  end if
  return $X, W$
```

4.3 ADAPTING THE NUMBER OF PARTICLES

Section 3.2 reviews two methods of Adaptive Monte Carlo Localization (AMCL). The first derives from a statistical analysis based on the number of bins needed for representing the current distribution (FOX, 2003). The second adapts the number of particles based on the Localization Confidence Estimation (LCE), a metric for evaluating the confidence of the current distribution, computed by matching real and virtual laser scans (HE et al., 2023). Differently from these two methods, in the Augmented MCL algorithm, reviewed in 2.3.4, THRUN; BURGARD; FOX (2005) suggests that the quality of the distribution should be measured over multiple steps and inserts samples randomly based on this metric, but keeps a fixed set size.

In summary, these methods propose different approaches for addressing the same concept: when confidence is low, the algorithm needs to perform global localization, thus the distribution should be spread through a wider state space. However, the higher the confidence gets, MCL tends to act as a position tracking, concentrating its particles in a narrower space, thus needing less samples for representing the true posterior accurately.

In the next section, we propose a novel method for measuring the quality, or confidence, of the MCL, which derives from measuring the likelihood of the resulting state from the weighted average of the distribution and smoothing it over multiple steps. We also present how this measure can be used for updating the number of particles and how they are inserted in the distribution.

4.3.1 Confidence Estimation

The MCL algorithm estimates a distribution for approximating a PDF that represents the robot's belief $bel(x_t)$. Therefore, a straightforward approach for guessing the robot's true state is to compute the weighted average of the particles' states, with weights given by their normalized likelihoods. Some authors also propose splitting the distribution in clusters before averaging (RÖFER; JÜNGEL, 2004).

No matter the methods, the particles represent multiple hypothesis of the robot's state, but the robot localization ends up being represented as a single state. Thus, as proposed by HE et al. (2023), we wish to evaluate the quality of the MCL not by the distribution, but by the resulting state. Also, as suggested by THRUN; BURGARD; FOX (2005), we smooth this estimation through multiple time steps by updating it with a weighted average.

The key idea is to apply the measurement model, also called the observation model, to the resulting state from the MCL, commonly given by the average state $x_t^{avg} = \sum_m^M w_t^m x_t^m$, acquiring its current likelihood $w_t^{avg} = p(z_t|x_t^{avg})$. From now on, we shall refer to the pair $\langle x_t^{avg}, w_t^{avg} \rangle$ by average particle. Then, we update the quality measurement q_t by the rule:

$$q_t = \alpha_q q_t + (1 - \alpha_q) w_t^{avg}, \tag{4.4}$$

which we call **S**mooth **W**eighting of **A**verage **P**article **O**bservations (SWAPO).

4.3.2 Set Size Adaptation

We perform a linear mapping for computing the desired set size M_{des} based on the current quality of the distribution q through a $map(x)$ function that corresponds to:

$$y = map(x, x_{min}, x_{max}, y_{min}, y_{max}) = \begin{cases} y_{max} & \text{if } x \geq x_{max} \\ y_{min} & \text{if } x \leq x_{min} \\ y_{min} + \frac{(x - x_{min})(y_{max} - y_{min})}{x_{max} - x_{min}} & \text{otherwise} \end{cases} \tag{4.5}$$

We omit the minimum and maximum parameters for simplicity, resulting in:

$$M_{des} = map(1 - q) = \begin{cases} M_{max} & \text{if } 1 - q \geq q_{max} \\ M_{min} & \text{if } 1 - q \leq q_{min} \\ M_{min} + \frac{((1-q) - q_{min})(M_{max} - M_{min})}{q_{max} - q_{min}} & \text{otherwise} \end{cases} \tag{4.6}$$

which increases M_{des} the higher the uncertainty $1 - q$ is. Also, a low confidence suggests that global localization is needed and samples should be spread through a broader state space. Therefore, we also adjust the deviations Δ_{trans} and Δ_{rot}, presented in Algorithm 7, in the resampling step according to the current confidence q:

$$\Delta_{des} = map(1 - q) = \begin{cases} \Delta_{max} & \text{if } 1 - q \geq q_{max} \\ \Delta_{min} & \text{if } 1 - q \leq q_{min} \\ \Delta_{min} + \frac{((1-q) - q_{min})(\Delta_{max} - \Delta_{min})}{q_{max} - q_{min}} & \text{otherwise} \end{cases} \tag{4.7}$$

Equation 4.6 determines the number of desired samples, but the real size of the distribution is only updated if resampling is performed. Thus, a new condition for resampling was added, by checking if the current M and desired M_{des} number of particles differ for more than a threshold: $\Delta M = |M - M_{des}| > \Delta M_{max}$.

4.3.3 Adaptive MCL through SWAPO

We insert the confidence estimation step after weights normalization, since SWAPO is computed from the Average Particle. Also, the desired number of particles should be updated before resampling, since it derives a condition for performing resample. Therefore, Algorithm 9 presents how these features are integrated into the previously enhanced MCL algorithm. Note that the q value from the previous step is required now, and that different values for Δ_{max} and Δ_{min} should be chosen for rotation and translation.

Algorithm 9 Adaptive Monte Carlo Localization through SWAPO

```
Require: $X = \{x^m | m = 1, 2, ..., M\}$, $W = \{w^m | m = 1, 2, ..., M\}$, $u$, $z$, $\Delta d$, $q$
 1:
 2: //Initialize weights' sum and integrate odometry displacement:
 3: $W_{sum} = 0$
 4: $\Delta d = \Delta d + |u|$
 5:
 6: //Propagate particles and compute importance weights:
 7: for $m = 1$ to $M$ do
 8:     $X[m] \leftarrow$ sample_from_motion_model$(u, x^m)$                ▷ Equation 4.1
 9:     $W[m] \leftarrow w^m \times$ measurement_model$(z, X[m])$           ▷ Equation 4.2
10:     $W_{sum} = W_{sum} + W[m]$
11: end for
12: //Update weights:
13: $W \leftarrow$ normalize_weights$(W)$
14:
15: //Update confidence and number of particles from SWAPO:
16: $x^{avg} = \sum_m^M W[m]X[m]$
17: $w^{avg} \leftarrow$ measurement_model$(z, x^{avg})$                    ▷ Equation 4.2
18: $q = \alpha_q q + (1 - \alpha_q) w^{avg}$                               ▷ Equation 4.4
19: $M_{des} \leftarrow map(1 - q, q_{min}, q_{max}, M_{min}, M_{max})$     ▷ Equation 4.6
20: $\Delta M = |M - M_{des}|$
21:
22: //Resample if needed:
23: if needs_resampling$(W_{sum}, \max(W), \Delta d, \Delta M)$ then
24:     $\Delta_{trans} \leftarrow map(1 - q, q_{min}, q_{max}, \Delta^{trans}_{min}, \Delta^{trans}_{max})$   ▷ Equation 4.7
25:     $\Delta_{rot} \leftarrow map(1 - q, q_{min}, q_{max}, \Delta^{rot}_{min}, \Delta^{rot}_{max})$         ▷ Equation 4.7
26:     $\langle X, W \rangle \leftarrow$ systematic_resampling$(X, W, M_{desired}, \Delta_{trans}, \Delta_{rot})$   ▷ Algorithm 7
27:     Reset $\Delta d = 0$
28: end if
29: return $X, W$
```

In Algorithm 9, X and W account for the samples states and weights, u and Δd are the current and accumulated motion estimated from the odometry, z are the current measurements, and q is the current localization confidence. The proposed method starts by initializing the weights' sum with 0 (line 3) and adding the current odometry to the accumulated displacement (line 4). Then, the particles are moved according to the motion model 4.1 based on the current odometry u (line 8), their similarities are computed from the measurement model 4.2 (line 9), and added to the weights' sum (line 10). This process is repeated for all samples and, at the end, their weights are updated by normalized values (line 13).

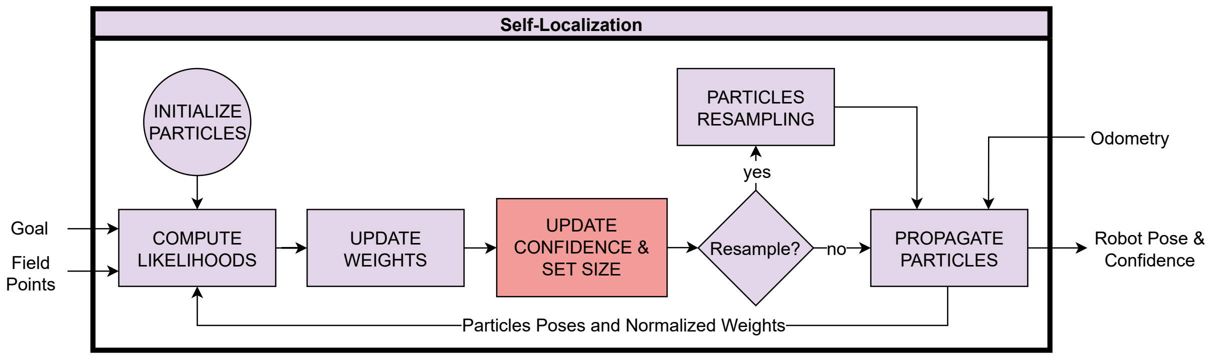
The so-called Average Particle is computed by the weighted average from the MCL samples (line 16) and its similarity is estimated by the measurement model (line 17). The localization confidence q is updated using 4.4 (line 18) and the number of desired particles is computed by a map function 4.6 (line 19). Finally, the absolute difference between the current and desired number of particles is calculated (line 20) for checking resampling conditions.

Besides the resampling conditions presented in section 4.2.3, we also check whether the ΔM difference is larger than a threshold (line 23). If any of the conditions is satisfied, Δ_{trans} and Δ_{rot} are updated according to the current localization confidence, i.e., the higher the confidence is, the more these values decrease, generating tighter poses in the resampling 4.7 (lines 24 and 25). Then, systematic resampling is performed using Algorithm 7 (line 26), generating a new set for representing the distribution, adapting the number of particles and their scattering in the states' space. After resampling, the accumulated odometry displacement is reset (line 27) and the algorithm returns the new set of samples and weights (line 29).

With this approach, the algorithm is able to reduce its computational complexity during position tracking, due to the reduced number of samples. Meanwhile, the adaptive deviations during resampling allows to enlarge the search space in case of low confidence, recovering from localization losses once they are detected by the quality measure.

Based on Algorithm 9 we add a step to the schematic presented in Figure 11, which includes updating the localization confidence and the desired number of particles for representing the distribution. The resulting schematic is shown in Figure 12, highlighting the new block in red. Note that, the number of particles that should be drawn from the distribution is estimated before the resampling step occurs.

Figure 12 – Schematic for MCL-based self-localization with confidence estimation and adaptive set sizes.



Source: Author

5 METHODOLOGY IMPLEMENTATION

The proposed methods were designed to solve the mobile robot self-localization problem under hardware-constrained scenarios in dynamic environments, which require fast and efficient solutions. Thus, the techniques must be evaluated not only regarding their accuracy and robustness towards the localization problem, but also by their computational costs running in an embedded device.

Following the motivations that led to this research, the techniques were implemented and evaluated within the RoboCup Small Size League (SSL) environment, addressing the self-localization problem for a four-wheeled omnidirectional robot inside an SSL soccer field with reduced dimensions. The hardware and software architectures employed for evaluation were based on previous work that intend to develop a fully autonomous SSL robot (MELO et al., 2022). We utilize their proposed solutions for odometry estimation and objects detection (MELO; BARROS, 2023) for compounding the self-localization algorithm, while also integrating this new capability to their functional pipeline.

Our self-localization algorithm relies on previous knowledge of the map, i.e., the robot's operational environment, which in our application corresponds to the SSL soccer field. We shall present the main characteristics of this environment, such as dimensions, features, and landmarks, based on the most recent rules from the RoboCup SSL competition (SMALL SIZE LEAGUE TECHNICAL COMMITTEE, 2022b).

In the SSL, three main classes of field elements can be used as references for localization: goals, field markings (lines), and boundaries. The first are already detected by the previously mentioned objects detection approach (MELO; BARROS, 2023). Therefore, techniques for detecting the remaining elements also needed to be developed for acquiring measurements from the environment.

Before deploying the self-localization algorithm to the target device, it is important to run offline tests with onboard-recorded data for adjusting parameters and observing behaviours. Thus, we generated a dataset with multiple paths and scenarios, containing all the necessary data for testing and evaluating self-localization and tracking algorithms under various conditions.

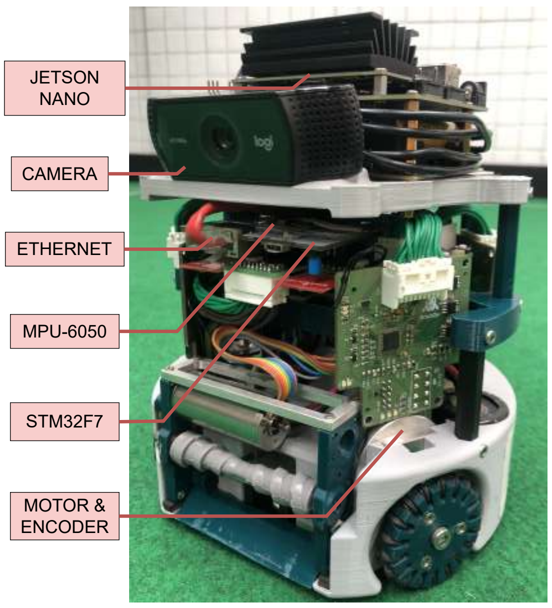
The following section presents the characteristics of the SSL environment, on which the self-localization approach was evaluated, followed by a brief review from the complete robot architecture presented by MELO et al. (2022), highlighting their hardware specifications, since it shall be employed throughout this thesis. Then, an approach for detecting the soccer field lines and boundaries is explained. Afterwards, the process of data recording is described. Finally, we explain how the proposed self-localization algorithm was implemented based on the available data and the proposed techniques.

5.1 ROBOT ARCHITECTURE AND SPECIFICATIONS

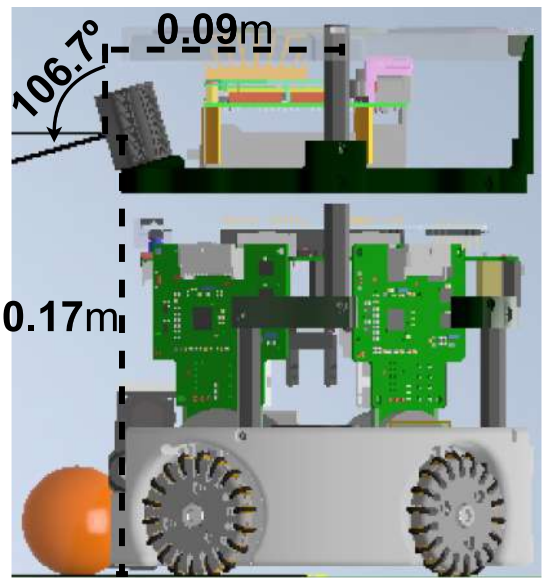
A complete architecture for executing basic soccer skills autonomously in the SSL was introduced in recent work (MELO et al., 2022). The experiments evaluated the proposed system within three tasks, namely grabbing a ball, scoring on an empty goal, and passing the ball to another robot, achieving high success rates.

Figure 13 – SSL Robot with onboard vision module.

(a) Robot hardware devices.



(b) Camera position regarding the robot's axis.



Source: (MELO et al., 2022) and author.

The system's main hardware specifications are summarized in Table 2 and illustrated in Figure 13. A 4GB NVIDIA Jetson Nano, containing a CPU and a GPU, is employed for processing embedded vision and decision-making, achieving an average processing speed of 30 frames per second with 10.8 Watts power consumption. Onboard images are captured by a Logitech C922 monocular camera with a 640x480 resolution. An ARM Cortex-M7 microcontroller unit (MCU), namely STM32F767ZI, processes low-level control and odometry, which movements are estimated from 2-channel 1024 steps incremental encoders and an MPU-60X0 family Inertial Measurement Unit (IMU). The CPU and MCU communicate through User Datagram Protocol (UDP) socket packets for guaranteeing low latency (CAVALCANTI; JOAQUIM; BARROS, 2022).

Their functional pipeline can be summarized in four modules, where the first two are executed by the Jetson Nano's CPU and GPU, and the remaining by the MCU:

1. Objects Detection and Position Estimation: balls, robots, and goals have their 2D bounding boxes detected by a CNN-based technique, and their relative positions are estimated utilizing the camera's intrinsic and extrinsic parameters regarding the robot axis (MELO; BARROS, 2023).

Table 2 – Hardware Specifications.

Hardware	**Specifications**
CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	128-core Maxwell
MCU	STM32F767ZI
Camera	Logitech C922
IMU	MPU-6050
Encoders	MILE 1024 CPT
Motors	Maxon EC-45 flat - 50W
Battery	LiPo 2200mah 4S 35C

Source: (MELO et al., 2022)

2. Decision-Making: Finite State Machines (FSM) are designed for solving the desired tasks, based mainly on the objects' relative positions, setting a target destination, a movement type, and action commands, such as kicking the ball.

3. Embedded Navigation: defines three types of movements, designed for performing rotations around the robot's axis, rotating around a target point (usually a detected object), and performing linear movements towards a target destination.

4. Trajectory Estimation: wheels' encoders and gyroscope readings are employed for estimating the robot's odometry. The angular movement is calculated by periodically integrating angular speed from the gyroscope, while translational movements are computed from the wheels' speeds applied to the robot's kinematics model.

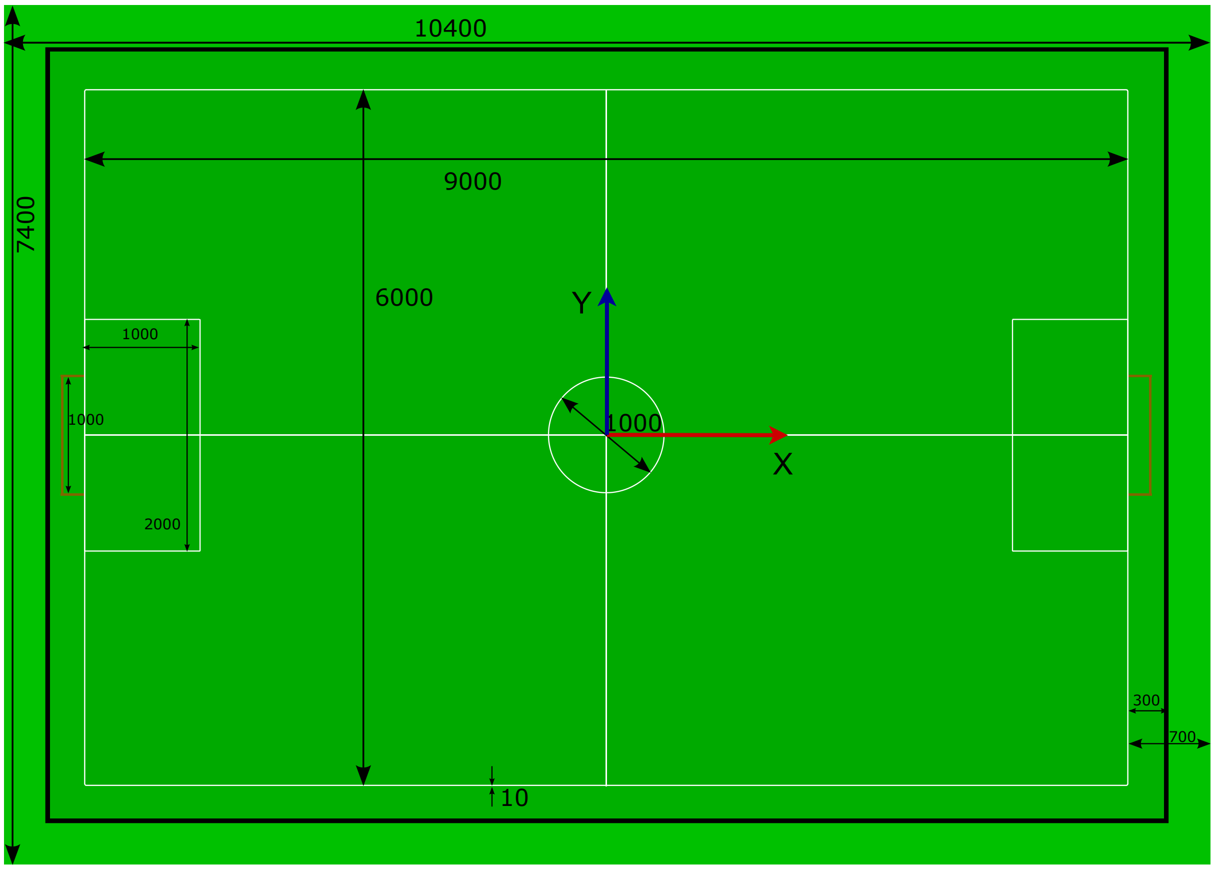
Note that the soccer skills were implemented without global localization knowledge. However, more detailed analysis from experiments highlighted the importance of this information for surpassing the main limitations of the proposed system, pointed as: planning more efficient paths, discarding out-of-field information, avoiding entering in the defender area, and making more efficient field explorations.

The proposed methods from this thesis were implemented employing the hardware presented by MELO et al. (2022), which have shown to suit the problem of developing an autonomous robot for implementing SSL basic skills. As presented in section 4.1, we integrate the self-localization algorithm as a new module to the system and evaluate its performance regarding processing speed and localization accuracy.

5.2 THE SSL ENVIRONMENT

The Small Size League is divided into two divisions with separate tournaments, namely division A and division B, on which games are played between two teams of 11 and 6 robots, respectively. Both divisions employ fields and robots with the same general characteristics,

Figure 14 – SSL field dimensions (in mm) and markings for division B.



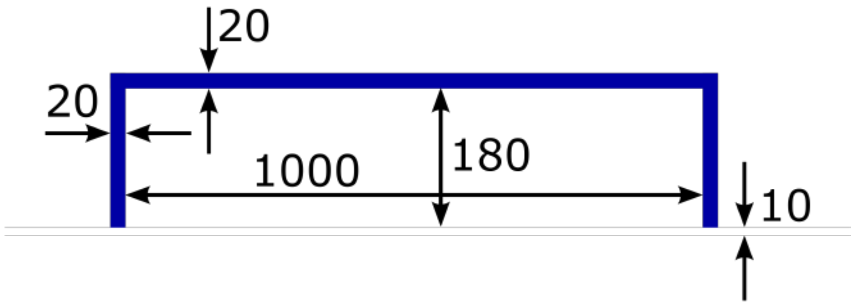
Source: (SMALL SIZE LEAGUE TECHNICAL COMMITTEE, 2022b)

except that division A adopts higher field dimensions, due to the increased number of robots. Thus, for simplicity, we present the field characteristics based on division B, which we shall refer to as SSL-B from now on.

Figure 14 illustrates the SSL-B environment dimensions, borders, markings, and goals. The field is fit into a 10.4 meters times 7.4 meters green carpeted surface with a playing area of 9 meters times 6 meters. The robots' area continues from the playing area for 0.3 meters, referred to as field margins, delimited by the so-called field boundaries, or borders, which are barred by 0.1 meters tall black walls. The field markings are made of 0.01 meters wide and white (paint, spray, white carpet or strong tape) lines. Lastly, the goals consist of two 0.16 meters high vertical side white walls joined at the back by a 0.16 meters high vertical rear white wall, as shown in 15.

For computing the robot's localization, we adopt the same coordinates system from the SSL-Vision software (ZICKLER et al., 2010), which defines the positive x-axis pointed to the right, the positive y-axis to the top, and the positive z-axis upward the ground plane, with the origin at the field's center, which yields that points of the ground correspond to the $Z = 0$ plane.

Figure 15 – Top-view of SSL goal dimensions (in mm) for division B. All walls are white.

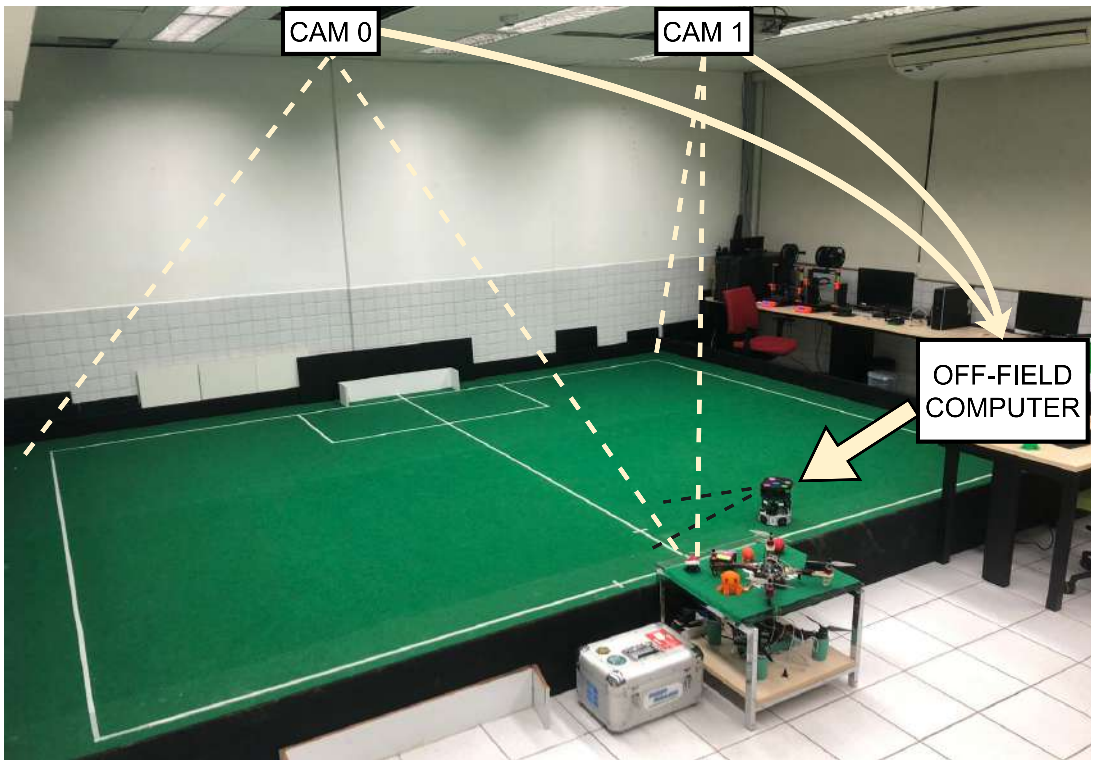


Source: (SMALL SIZE LEAGUE TECHNICAL COMMITTEE, 2022b)

5.3 DATA RECORDING

Due to the unavailability of a complete SSL-B field, the experiments were conducted within a 6m x 4.5m area, configured as positive-half SSL field, which dimensions are detailed in Figure 17. We generated a dataset containing onboard-recorded images and odometry data, acquired with the hardware presented in section 5.1, and ground-truth positions, acquired by SSL-Vision (ZICKLER et al., 2010). Figure 16 illustrates the environment employed throughout the recordings.

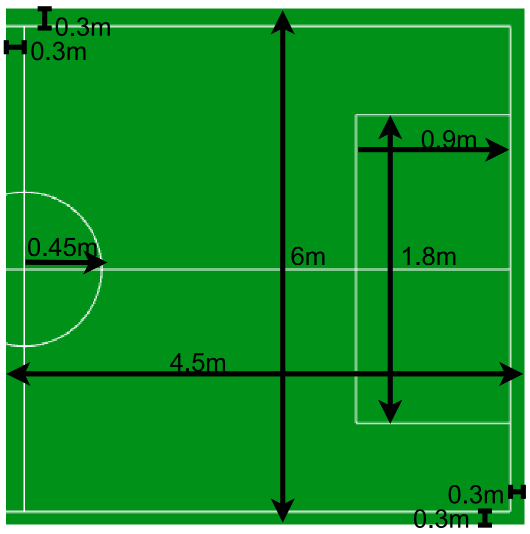
Figure 16 – Recordings environment with a 6m x 4.5m SSL field.



Source: Author

Figure 18 illustrates examples of images from the dataset, which corresponding ground-truth positions and odometry data are saved in CSV files. We split the dataset into three types of paths: squared (sqr), random (rnd), and in-game situations (igs). For each path type, we set

Figure 17 – Reduced positive-half field dimensions in millimeters.



Source: Author

the robot's maximum speed to 1 (labeled as 01), 1.5 (02), and 2 m/s (03), resulting in nine scenarios with different average speeds, duration, and odometry accuracies. Table 3 depicts the dataset characteristics for each scenario, containing their durations, average translational and rotational velocities of the robot over the trajectory, average frame capture rates and the odometry's Root Mean Square Error (RMSE), which is used for composing the MCL algorithm.

Squared Paths. The robot performs three laps of 5m x 2.5m rectangular paths in this scenario, mainly with forward and clockwise rotational moves. As a result, the robot rarely sees or detects the goal during the trajectories, and reaches higher velocities due to distances between points being further.

Random Paths. The robot navigates through 25 random poses on the field, combining linear and angular movements. At the start, the robot looks towards the goal, allowing for corrections of mirrored pose estimations. In contrast, inertial odometry tracking accumulates more errors, due to the longer durations and the combinations of translation and rotational movements.

In-Game Situation Paths. This scenario reproduces a similar robot soccer game situation to the Vision Blackout challenge stage II: scoring on an empty goal. The robot is positioned at the field's corner, and the ball in the opposite half. In this scenario, three movements are executed: rotating on its axis to look towards the ball, moving to the goal-to-ball line projection looking towards the ball, and moving forward to kick it to the goal.

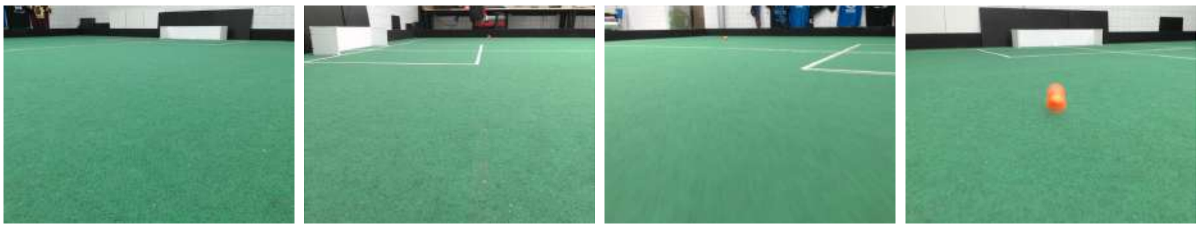
Table 3 – Dataset Characteristics.

Name	**Duration (s)**	**Avg. Trans. Vel. (m/s)**	**Avg. Rot. Vel. (deg/s)**	**FPS (Hz)**	**Odometry RMSE**
sqr_01	97.295	0.469	0.250	22.22	0.472
sqr_02	81.378	0.567	0.290	22.05	0.468
sqr_03	70.989	0.662	0.342	20.66	1.245
rnd_01	114.211	0.423	0.367	21.27	0.476
rnd_02	101.969	0.531	0.456	19.94	0.945
rnd_03	95.123	0.589	0.464	21.10	0.911
igs_01	25.402	0.325	0.268	21.37	0.284
igs_02	25.099	0.395	0.255	21.87	0.420
igs_03	25.333	0.382	0.274	21.59	0.258

Source: Author

Figure 18 – Images from the onboard-recorded dataset in three different scenarios.

(a) rnd_01. (b) sqr_02. (c) igs_03. (d) igs_03.



Source: Author

5.4 DETECTING FIELD LINES AND BOUNDARIES

In the SSL, three main classes of field elements can be used as references for localization: goals, field markings (lines), and boundaries. Goals can be detected using the approach from (MELO; BARROS, 2023). Thus, we implemented approaches for detecting field lines and boundaries using adaptations of grid-based line detection techniques from other RoboCup Leagues (RÖFER; JÜNGEL, 2004; RÖFER et al., 2019), and their processing times were evaluated running on the Jetson Nano.

5.4.1 Grid-based Line Detection

Past researches in the RoboCup showed that extracting pixels on lines instead of detecting complete lines is a faster and more robust approach, due to the robot soccer dynamism, which often causes lines to be partially covered by other robots or limited by the camera's field of view (RÖFER; JÜNGEL, 2004).

This approach sets vertical and horizontal lines in the image and performs pixel-by-pixel color segmentation along them. Then, a state machine decides whether objects of interest exist, or not, inside the segmented line, mostly by counting the number of pixels of certain colors, or the number of pixels since a certain color was detected last. For reducing computational costs, we adopt only vertical line scans and color segmentation is realized by setting thresholds that decide whether a pixel is black, white, green, or none of them.

5.4.2 Field Boundaries Detection

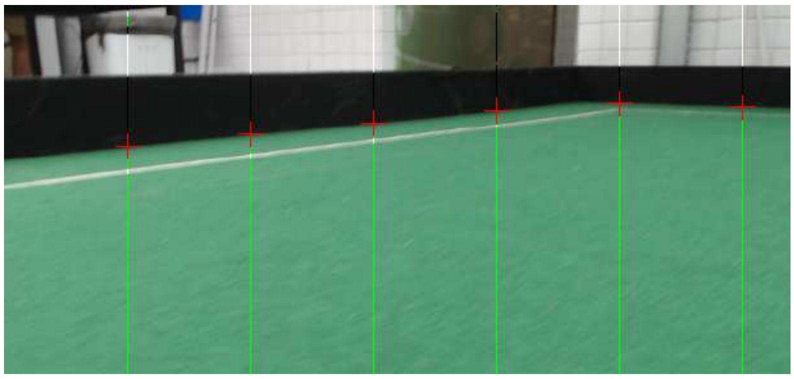
Field boundaries, or borders, are 0.1 meters tall black walls upon the green carpeted field. Therefore, a straight forward approach is to look for a sequence of black pixels after a green-to-black transition. The first pixel corresponds to the point the border touches the floor, i.e. with $Z = 0$, which we shall refer to as field boundary ground-point. We can estimate its relative position to the onboard camera and, consequently, to the robot, with the camera intrinsic and extrinsic parameters employing the formulations presented in 2.2.

Algorithm 10 presents our solution for detecting the field boundary ground-point within a vertical line of the image. A bottom-up scan is performed, making color segmentation and checking whether a black pixel exists; if yes, we check if a number of subsequent pixels, defined as min_wall_length, are also black, and if yes, the first pixel of the sequence is taken as our desired ground-point. Figure 19 illustrates the resulting frame after performing multiple vertical line scans, highlighting boundary points with red cross marks.

Algorithm 10 Field Boundary Detection

```
Require: $img$, $vertical\_lines$, and $min\_wall\_length$
  Initialize: $boundary\_pixels = \emptyset$
  for $line_x$ in $vertical\_lines$ do
      $wall\_pixels = \emptyset$
      $pixel_y = img_{height} - 1$
      while $pixel_y > 0$ do
          $pixel \leftarrow img[pixel_y, line_x]$
          if $sizeof(wall\_pixels) > min\_wall\_length$ then
              $boundary\_pixels \cup wall\_pixels[0]$
          else
              if $isBlack(pixel)$ then
                  $wall\_points \cup pixel$
              else
                  $wall\_pixels = \emptyset$
              end if
          end if
          $pixel_y \leftarrow pixel_y - 1$
      end while
  end for
  return $boundary\_pixels$
```

Figure 19 – SSL field boundary detection.



Source: Author

5.4.3 Field Markings Detection

Detecting field markings, which consist of 0.01 meters wide and white (paint, spray, white carpet or strong tape) lines, is a more challenging task in the SSL, due to their thin widths and the robot's low height. Therefore, a robust approach for detecting these elements requires more elaborate conditions after color segmentation. Once they are detected, their relative positions can also be estimated since they lay on the ground.

An approach for detecting field markings in vertical line scans is depicted in Algorithm 11. It scans the vertical line from bottom to top classifying the pixels' colors. While a black pixel is not found, it checks if the current pixel is white; if yes, it starts appending the following pixels to a $field\_line$ list. When a green pixel is detected, it checks whether the list size satisfies the min_length and max_length conditions; if yes, the mean pixel from the list is taken for representing that line's point and the list is reset.

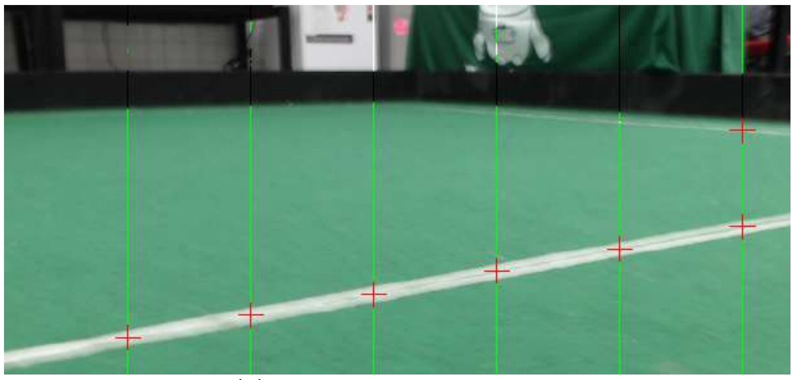
Figure 20 illustrates the presented technique applied to two different frames, with detected field markings highlighted by red cross marks. In the first, we see a successful utilization of the algorithm, on which the detected points correspond to true marking of the field. However, in the second image we note that several field (green) pixels were classified as markings (white), highlighting the limitations of a simple threshold-based color segmentation, which is prone to false classifications when the illumination changes. Also, when compared to the field boundary detection, Algorithm 11 presents a higher computational cost. We shall evaluate and compare their performances regarding processing time, which led to not using field markings as references for localization.

5.4.4 Practical Considerations

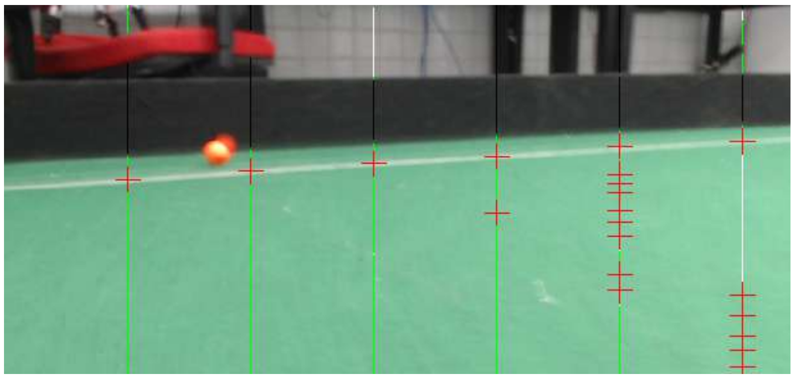
The camera transformations from 5.4 can be applied to the detected field points for computing their relative positions to the robot, by using the prior knowledge that they lay on the ground, which implies $Z = 0$. However, as we shall present in the following chapter, the results from field markings detection were not fast and reliable enough for it to be employed

Figure 20 – Field lines detection.

(a) Successful line detection.



(b) Erroneous line detection.



Source: Author

in self-localization.

For minimizing the computation costs, only a single vertical scan line was employed for acquiring measurements, performing the presented boundary detection technique. For increasing the information gain, the line's position on the screen was randomly changed at each iteration, excluding positions inside detected objects' bounding boxes.

Algorithm 11 Field Markings Detection

```
Require: $img$, $vertical\_lines$, $min\_length$, and $max\_length$
  Initialize: $line\_pixels = \emptyset$
  for $line_x$ in $vertical\_lines$ do
      $is\_field\_line = false$
      $white\_pixels = \emptyset$
      $pixel_y = img_{height} - 1$
      while $pixel_y > 0$ do
          $pixel \leftarrow img[pixel_y, line_x]$
          if $isBlack(pixel)$ then
              $white\_pixels = \emptyset$
              $break$
          else
              if $!isGreen(pixel)$ and $is\_field\_line$ then
                  $is\_field\_line = true$
              else
                  if $isGreen(pixel)$ and $is\_field\_line$ then
                      $is\_field\_line = false$
                      if $min\_length \leq sizeof(white\_pixels) \leq max\_length$ then
                          $mean\_white\_pixel \leftarrow mean(white\_pixels)$
                          $line\_pixels \cup mean\_white\_pixel$
                      end if
                      $white\_pixels \leftarrow \emptyset$
                  end if
              end if
              if $is\_field\_line$ then
                  $white\_pixels \cup pixel$
              end if
          end if
      end while
  end for
  return $line\_pixels$
```

5.5 PROPOSED METHOD IMPLEMENTATION

The proposed methods from chapter 4 contain several adjustable parameters that must be set before deployment. Also, for implementing an MCL algorithm two models are required:

- Motion model: describes how the state evolves, based on the robot's movement.
- Measurement model: estimates the sample likelihood, based on the current robot measurement and its expected value given the particle state.

5.5.1 Motion Model

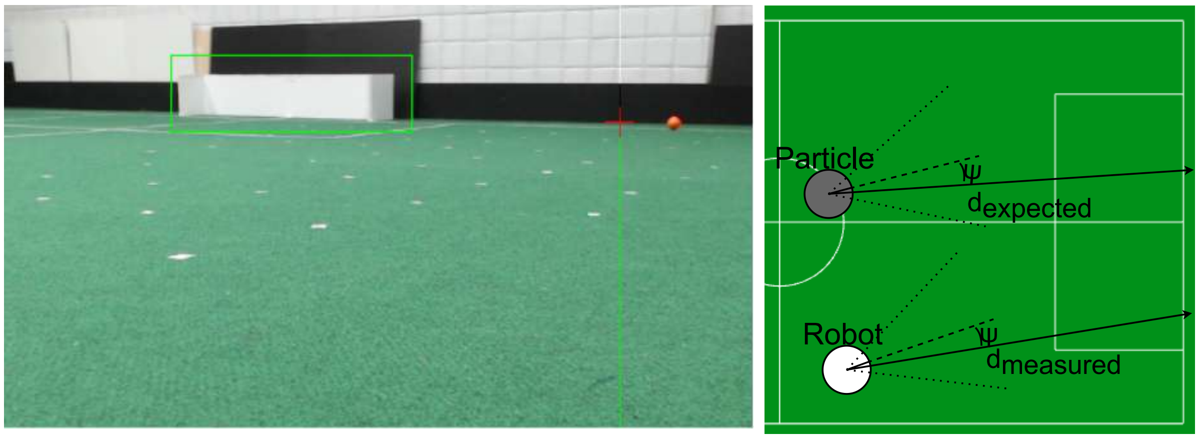
The first follows the implementation from section 4.2.1, employing the odometry acquired from the architecture presented by MELO et al.(2022), which computes the robot's translational movement from the wheels' speeds using forward kinematics, as explained in section 2.1, and estimates rotations from the gyroscope measurements.

5.5.2 Measurement Model

From the outputs of our vision processing pipeline we define measurement models for two types of observations: goals and boundary points. Boundaries can be seen from almost any location on the field and provide information about both Cartesian directions; however, the field is symmetric regarding them, which may cause mirrored pose estimations. On the other hand, goals are rarely seen from the robot's perspective but give orientation information, enabling to correct mirroring issues on a half-field setup, however, their computed relative positions are only rough estimates, since they can not be retrieved correctly from the 2D bounding box. Figure 21 illustrates a resulting frame after vision processing is applied, highlighting the detected goal's bounding box and the detected boundary point.

Figure 21 – Observations with onboard vision on a SSL field for a MCL algorithm.

(a) Resulting frame after vision processing. (b) Robot and particle visions.



Source: Author

Boundary Points Observation. Field boundary XY relative coordinates to the robot are computed using camera parameters by assuming they lay on the ground and converted to relative distance, d, and bearing, ψ. For each of the M samples, the expected distance $\hat{d}^m$ is computed by projecting a line from its position towards the measured ψ direction, and finding the first intersection with a field border, as shown in Figure 21. Similarities are computed from the measurement model presented in section 4.2 using Equation 4.2, simplified as:

$$p(z_t^{boundary}|x_t^m) = exp\left[-\alpha_d\left(1-\frac{\hat{d}^m}{d}\right)^2\right] \tag{5.1}$$

Goal Observation. As the goals' 2D bounding boxes do not provide enough information for estimating their relative positions accurately, we express similarities from goal observations as booleans, i.e., 1 or 0. The particle's vision field-of-view is projected onto the field, as in Figure 21, determining whether a goal is inside its vision range. If the robot detects a goal and the particle's vision does not cover it, the similarity is set to 0; otherwise, it is 1. This logic in summarized by the truth table 4, on which the first columns represents if a goal was detected or not by the robot's vision, and the "Particle" column expresses if the sample's field-of-view covers the goal or not.

Table 4 – Truth Table for Goal Observations.

Robot	**Particle**	$p(z_t^{goal} \mid x_t^m)$
0	0	1
0	1	1
1	0	0
1	1	1

Source: Author

Computing Likelihoods. The sample likelihood is expressed by the product of boundary and goal observations, and the importance weights are updated by Equation 4.3, resulting in:

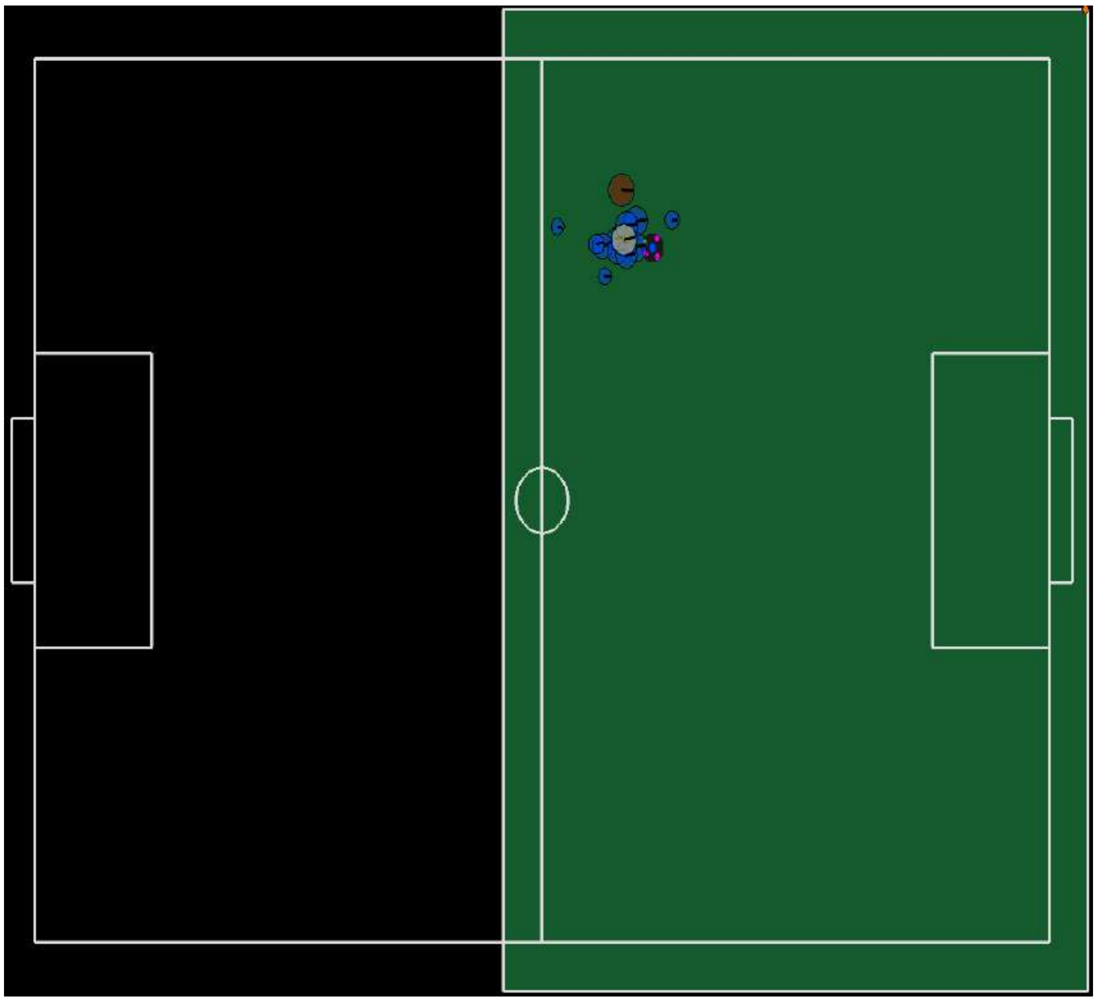
$$w_t^m = w_{t-1}^m p(z_t^{boundary} \mid x_t^m) p(z_t^{goal} \mid x_t^m) \tag{5.2}$$

5.6 ALGORITHM IMPLEMENTATION

We applied the motion and measurement models to algorithms 8 and 9, implementing them as Python scripts. The methods were first run on an AMD Ryzen 5 4500 CPU, containing 6 CPU cores at 2.3GHz, employing the rSoccer framework for visualizing the robot, particles, and odometry positions (MARTINS et al., 2022). Figure 22 illustrates self-localization being performed on the igs_03 scenario from the recorded dataset. The green-background area corresponds to the valid positions of the SSL field from the dataset, and blue, yellow, and red circles correspond to the particles, the average particle, and the odometry, respectively, while the robot with colored patterns on the top represents the ground-truth robot position. This visualization allows to observe the particles' behaviours and how they are affected by the parameters of the algorithm, thus, facilitating their fine-tuning process before deploying the algorithm to the robot hardware.

After the algorithms' were validated and parameters were adjusted, the methods were run on the 4GB NVIDIA Jetson Nano, the target hardware of our application, using the presented dataset. We evaluate the complete system's processing time with and without adapting the sample set sizes of the MCL, showing that the second approach drastically reduces the computational costs of the algorithm without losing its accuracy.

Figure 22 – Proposed method behaviour visualization in rSoccer (MARTINS et al., 2022).



Source: Author

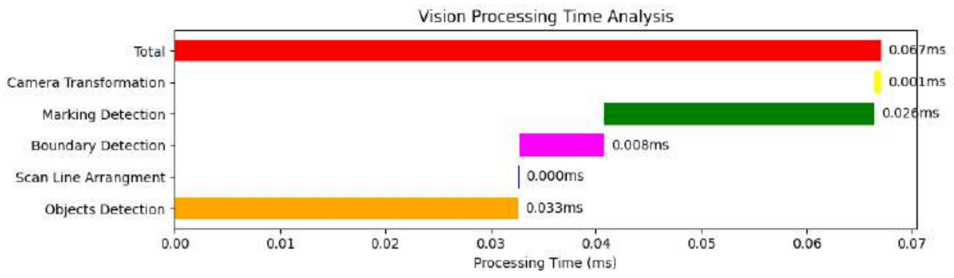
6 EXPERIMENTS AND RESULTS

This chapter aims to validate our proposed self-localization solution. Experiments were conducted employing the onboard recorded data and evaluated running on the hardware presented in sections 5.3 and 5.1. Firstly, we evaluate our vision processing pipeline, highlighting the performance of our proposed grid-based field detection algorithms regarding processing time on the Jetson Nano. Then, we present results from the MCL algorithm 8 regarding accuracy and computation speed, comparing its performance with and without adapting the set sizes.

6.1 EMBEDDED VISION EVALUATION

Algorithms 10 and 11 were run on the Jetson Nano using a single randomly positioned vertical scan at each iteration. Figure 23 shows the resulting processing times of each method, along with other capabilities from the vision processing pipeline, and examples of processed images, highlighting the field points are presented in figures 24 and 25.

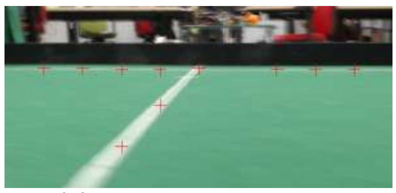
Figure 23 – Evaluating processing times of vision modules.



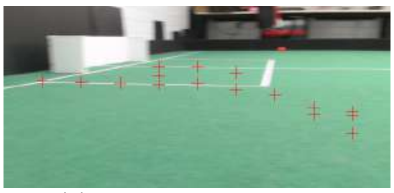
Source: Author

Figure 24 – Resulting frame after performing field markings detection on ten uniformly spaced vertical scan lines.

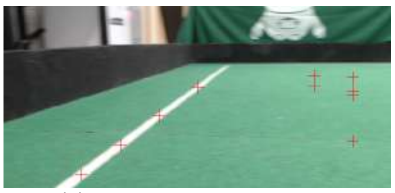
(a) Successful detection.



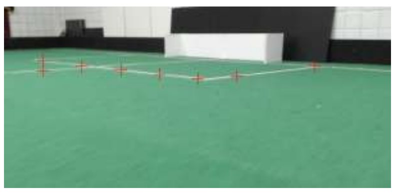
(b) Erroneous detection.



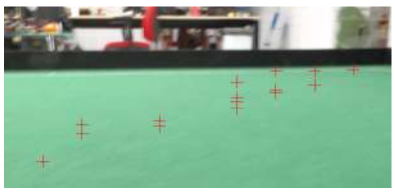
(c) Erroneous detection.



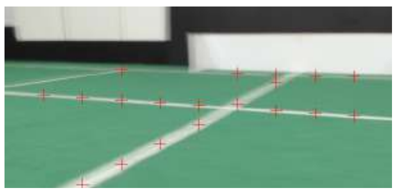
(d) Successful detection.



(e) Erroneous detection.



(f) Successful detection.

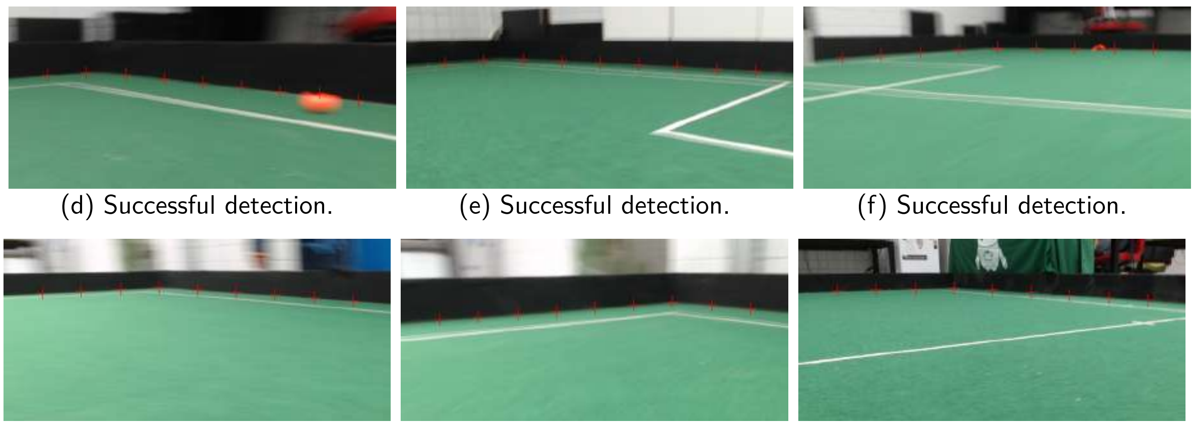


Source: Author

The system takes 26 milliseconds on average for scanning a vertical line performing marking detection, more than 3 times slower than boundary detection. This result yields that detecting field boundaries on 3 different line scans would be faster than detecting a single marking, for example. In addition, Figure 24 highlights the method's sensibility to illumination changes, mainly due to the color segmentation causing field (green) pixels to be classified as white, but also due to the field lines being too thin, requiring low min_length and max_length thresholds. Therefore, the method was not employed in the system, and markings were not used as references for self-localization.

Figure 25 – Resulting frame after performing field boundary detection on ten uniformly spaced vertical scan lines.

(a) Successful detection. (b) Successful detection. (c) Successful detection.



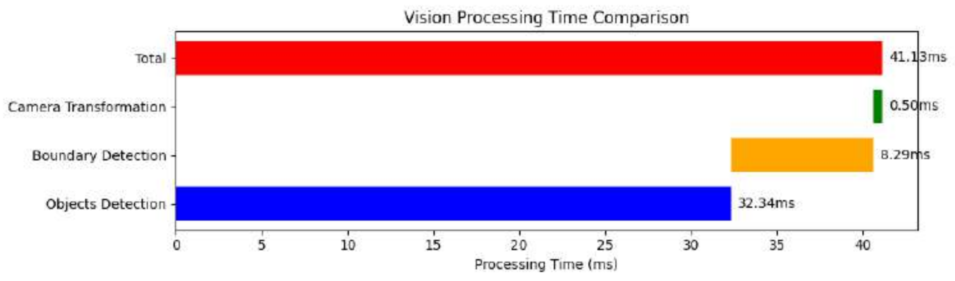
(d) Successful detection. (e) Successful detection. (f) Successful detection.

Source: Author

Boundary detection presented more reliable results while also achieving faster processing speeds. The system takes 8 milliseconds on average for scanning a vertical line performing the boundary detection algorithm 10. In addition, it is less prone to false-positives, which are usually caused by wrong color segmentation due to imperfections on the field border walls, as illustrated in Figure 25.

The complete module takes 41 milliseconds to process, on average, resulting in an approximate rate of 24 frames per second. We depict the processing times from the vision processing tasks in Figure 26.

Figure 26 – Complete vision processing pipeline processing times.



Source: Author

6.2 SELF-LOCALIZATION EVALUATION

We evaluate the accuracy of our proposed self-localization solution within three scenarios of the dataset: rnd_01, sqr_02, and igs_03. Also, the experiments were conducted under two assumptions: a 1 meter radius seed of the robot's initial position is known (I), as in the SSL Vision Blackout Challenge (SMALL SIZE LEAGUE TECHNICAL COMMITTEE, 2022a); and no information of the initial robot pose is known (II). Since the MCL algorithm relies on probabilistic functions, these experiments were run three times on each scenario. Thus, each algorithm was evaluated in a total of eighteen runs. Accuracy was measured by the Root Mean Square Error (RMSE) over the entire trajectory and we also report if the algorithm was able to regress the robot's pose correctly or not.

6.2.1 Localization Given a Seed Position

Note that, in the cases where a seed position is given, the algorithm still has no information of the the robot's initial orientation, which directly impacts the particles' propagation and likelihood updates. Thus, even though this prior knowledge enables to reduce the initial search space, it is not sufficient for addressing the localization problem as position tracking.

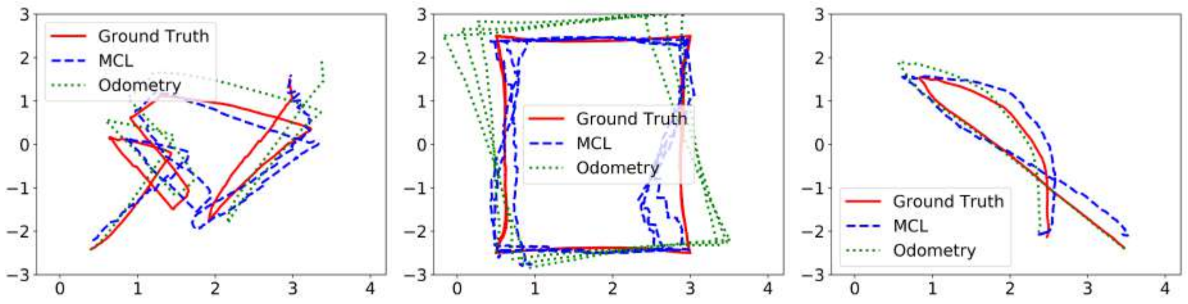
Firstly, we measure the accuracy and processing times of the self-localization using a fixed number of 100 particles, i.e., we employ Algorithm 9, but the M_{min} and M_{max} are set to 100. Figure 27 shows the resulting trajectory over one round of each scenario, demonstrating that the algorithm was able to find and maintain the robot's pose. However, the particles tend to spread when the robot is far from the borders, mostly due to the observations becoming less accurate for higher distances. Even so, the self-localization algorithm was able to recover the robot's tracking in all cases, reducing its distance to the ground truth position, as illustrated in Figure 28, which shows the odometry (green) and MCL (blue dashed) distances to ground truth over time, in meters and seconds. It highlights that the MCL trajectory was able to maintain a lower distance than odometry to the ground-truth during most of the trajectory, and specially at the end, when the robot stops and the particles converge to its state, resulting in errors lower than 0.2 meters (the robot's diameter is 0.18 meters).

For adapting the number of particles, M_{min} and M_{max} were set to 20 and 200, respectively, and we plot the same comparisons for a qualitative analysis. Figure 29 shows that similar trajectories were achieved using adaptive set sizes. However, the algorithm took more steps to recover from deviations, which were also higher, mostly due to the reduced number of samples. The higher errors are more evident in Figure 30, which plots the distances to the ground-truth over time. In comparison to the fixed sample sets, not only the deviations during the trajectory were higher, but also the final distance to the ground-truth.

Quantitative results are presented in Table 5. The reduced accuracy from Adaptive MCL is an expected consequence from reducing the number of particles. Even so, results show that our adaptive MCL algorithm was able to localize and maintain the robot's tracking, while

Figure 27 – Trajectory comparisons from MCL, odometry and ground-truth in 3 scenarios, using a fixed number of 100 particles, and starting from a 1 meter seed of the robot's position.

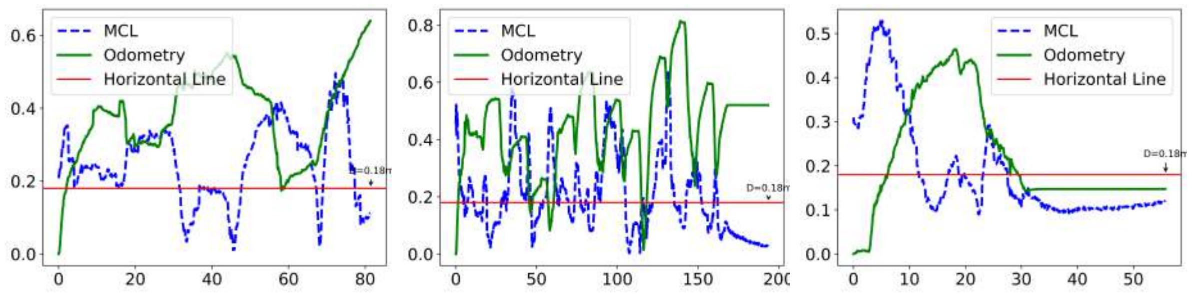
(a) rnd_01. (b) sqr_02. (c) igs_03.



Source: Author

Figure 28 – Distances from MCL and odometry to ground-truth over time in 3 scenarios, using a fixed number of 100 particles, and starting from a 1 meter seed of the robot's position. The x- and y-axis indicate the elapsed time and distances in seconds and meters, respectively.

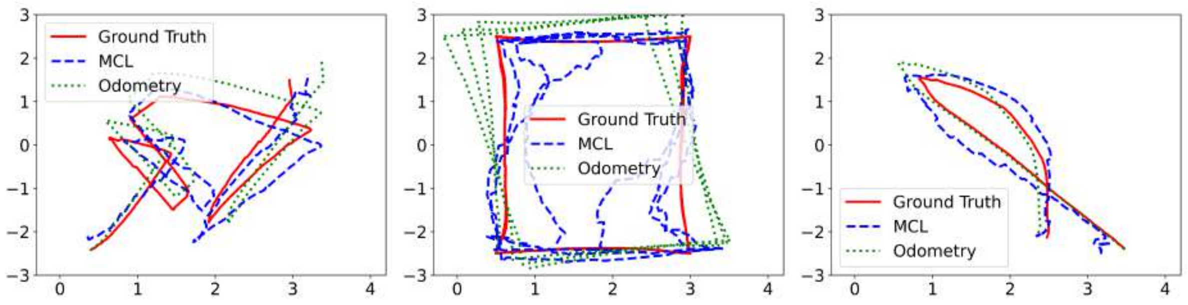
(a) rnd_01. (b) sqr_02. (c) igs_03.



Source: Author

Figure 29 – Trajectory comparisons from MCL, odometry and ground-truth in 3 scenarios, using adaptive set sizes from 20 to 200 samples, and starting from a 1 meter seed of the robot's position.

(a) rnd_01. (b) sqr_02. (c) igs_03.



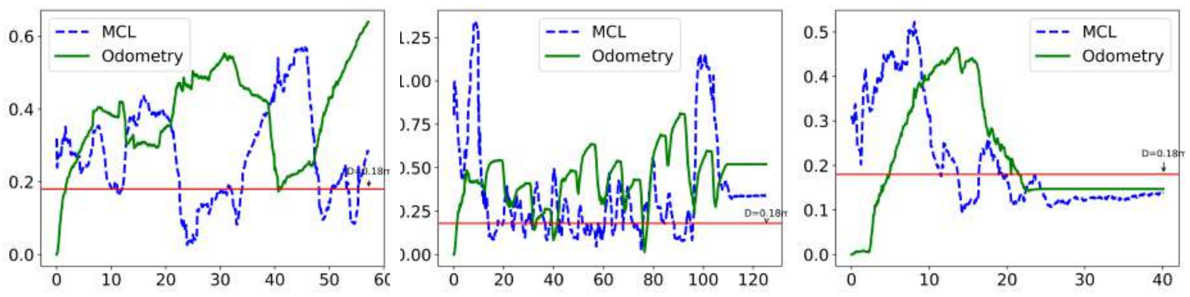
Source: Author

also achieving lower errors than the odometry trajectory. Also, we share the average rates for processing the complete system's pipeline in frames per second (FPS), which includes the onboard vision and self-localization, demonstrating that adapting the number of particles drastically increases the processing speed of self-localization.

Note that the results in Table 5 include the times from vision processing, which was shown to consume approximately 41 milliseconds. A detailed analysis of self-localization processing times over iterations of the sqr_02 scenario is shown in Figure 31. The dashed plots correspond

Figure 30 – Distances from MCL and odometry to ground-truth over time in 3 scenarios, using adaptive set sizes from 20 to 200 samples, and starting from a 1 meter seed of the robot's position. The x- and y-axis indicate the elapsed time and distances in seconds and meters, respectively.

(a) rnd_01. (b) sqr_02. (c) igs_03.



Source: Author

Table 5 – Accuracy and Total Processing Times Comparison.

Scenario	Odometry	MCL (M=100)		Adaptive MCL	
	RMSE (m)	RMSE (m)	FPS (Hz)	RMSE (m)	FPS (Hz)
rnd_01	0.454	**0.323**	9.782	0.381	**13.845**
sqr_02	0.474	**0.396**	9.168	0.447	**14.333**
igs_03	0.251	**0.220**	9.805	0.239	**13.619**

Source: Author

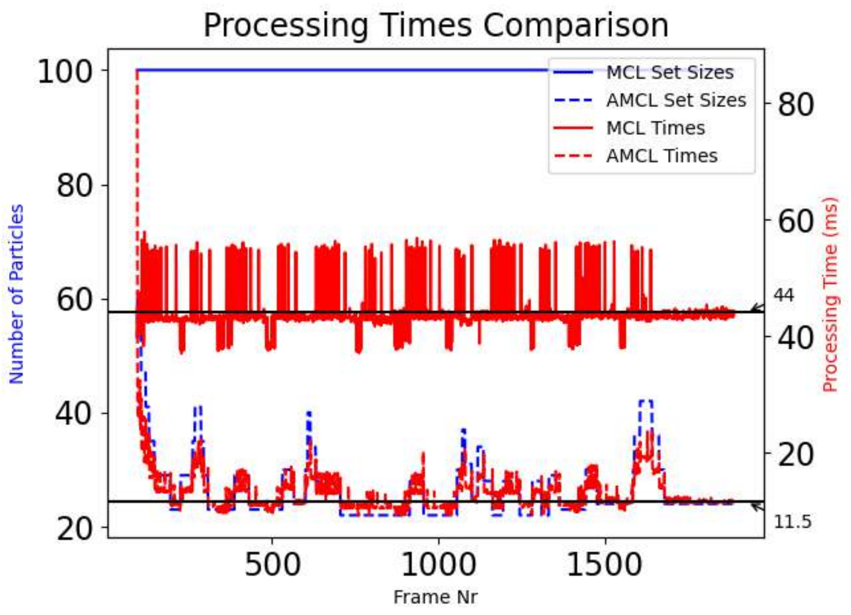
to the adaptive MCL algorithm and the red color indicates plots of processing times, while blue plots are the number of particles. We highlight the 44 ms and 11.5 ms lines, showing that the AMCL was able to achieve almost 4 times the speed of MCL, due to the reduced number of samples.

A more in-depth performance evaluation is presented in Figure 32. It depicts the times required for processing each functional block from the self-localization schematic presented in Figure 12 with 100 and 20 particles. The set size most affected the resampling and likelihood estimation steps; however, the first had a much lower impact on the system's pipeline (approximately 1 ms with 100 particles). Even though the propagation of the particles is also affected by the number of samples, it has shown to suffer less from the set size adaptation, mostly due to its implementation being paralyzed as matrix operations. Therefore, exploiting parallelism in the likelihood estimation may bring benefits to the self-localization pipeline, since they are also independent for each particle.

6.2.2 Localization Without Prior Information

Results from the previous section demonstrated that our proposed methods were capable of converging to the correct localization and maintaining the robot's tracking when the particles are initialized within a range of the initial ground-truth position. Therefore, in this section, we evaluate the methods' capability to correctly localize the robot without prior information about its initial coordinates. For that, we present the algorithms' success rates in converging

Figure 31 – Self-localization processing times and number of particles over iterations on the sqr_02 scenario. The left y-axis accounts the set size and the right y-axis shows the processing time in milliseconds, while the x-axis is the number of the corresponding frame from the dataset.



Source: Author

to the correct pose on multiple attempts and discuss what impacts the most on these results.

The algorithm could converge to the correct pose in 6 of the 15 attempts using a fixed number of 100 samples. We depict the details and characteristics observed in the three types of scenarios: igs, rnd, and sqr.

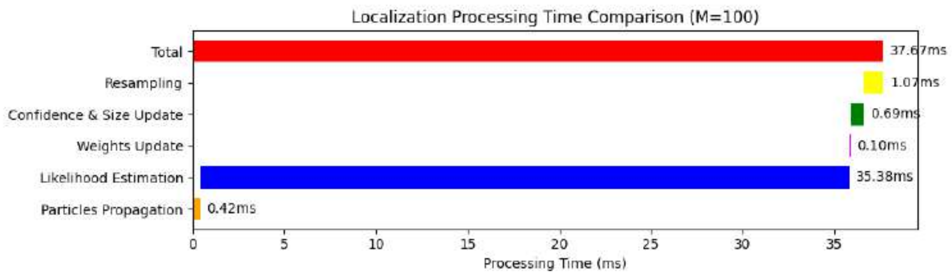
In the igs scenario the robot starts near the field corner and performs a rotation around its axis. Therefore, its initial measurements have high accuracy, and it acquires information from multiple borders. These characteristics cause the algorithm to rapidly converge, however, in some situations it might converge to symmetric poses, since the goal was not observed, which is the only reference for fixing mirrored poses in our method. This behavior is highlighted in Figure 33, which presents the errors of MCL and odometry over time.

In contrast, the rnd scenario starts by performing a linear forward movement looking towards the goal, but acquiring little and imprecise information from the field borders. We observe that the goal observation rapidly discards a wide range of states, but, in most cases, it was insufficient for converging to the correct pose, which diverged as soon as the robot started looking towards other directions, as presented in the distance comparisons in Figure 34. Based on the results from this scenario, we believe that observing the goal for a longer time with more varied movements would enable the robot to correctly self-localize.

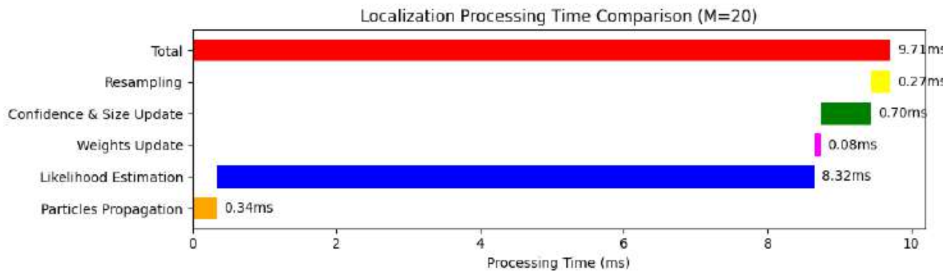
The highest success rate was achieved in the sqr scenario. It has almost no goal observations, but contains more varied and precise measurements, since the robot approximates to the field borders and rotates looking towards them. The measurements from field boundaries allow the algorithm to converge to a pose that corresponds to them, however, the lack of goal observations cause mirrored pose estimations. Therefore, we observe that, in the cases where

Figure 32 – Complete self-localization pipeline processing times with 100 and 20 particles.

(a) Processing times with 100 samples.



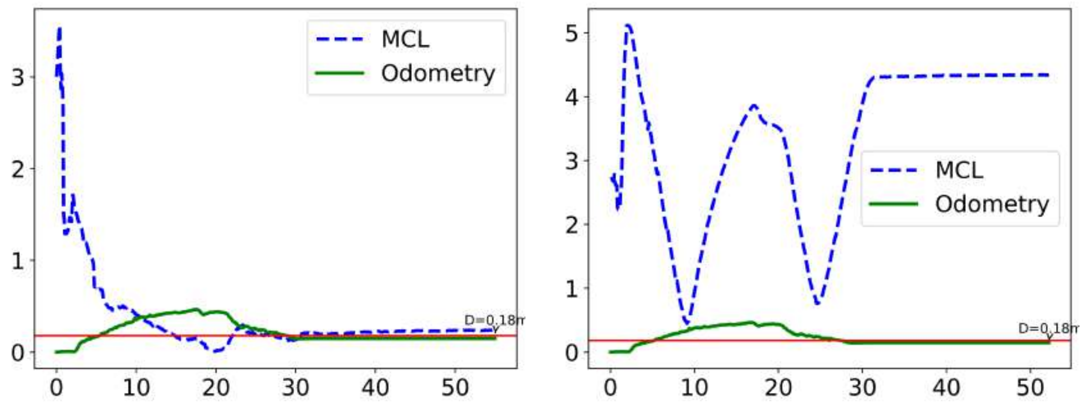
(b) Processing times with 20 samples.



Source: Author

Figure 33 – Distances from MCL and odometry to ground-truth over time in 2 rounds of the igs scenario, using fixed set sizes of 100 samples, without prior localization knowledge. The x- and y-axis indicate the elapsed time and distances in seconds and meters, respectively.

(a) Successful localization. (b) Erroneous localization.



Source: Author

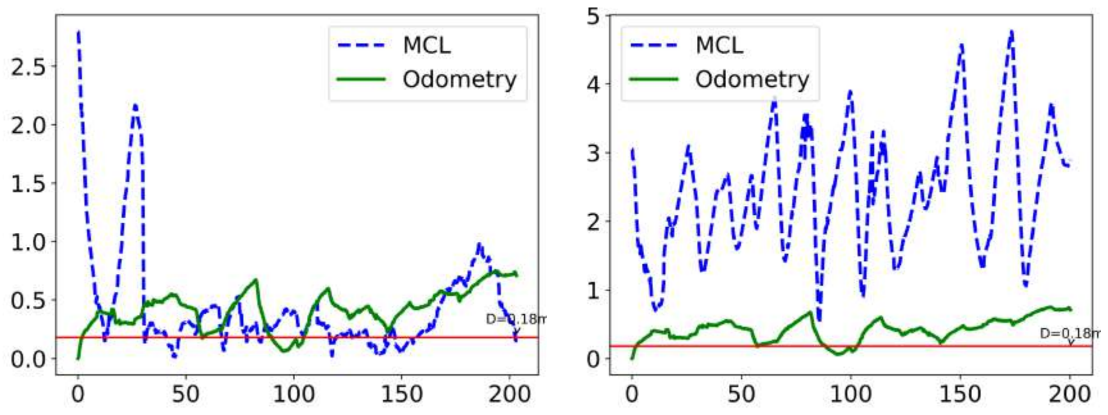
localization did not converge correctly, it was mostly due to mirroring issues, which can be seen in Figure 35.

Similar results were achieved using adaptive set sizes in the sqr and igs scenarios. In contrast, the algorithm converged correctly in all attempts of the rnd scenario, resulting in a total of 9 successful localization. Table 6 depicts the success rates of each scenario using fixed and apative set sizes, showing the number of times the algorithm correctly converged among the 5 attempts executed in each scenario.

Results from sqr and igs scenarios are derived from the same analysis presented for fixed set sizes, which showed that most erroneous localization occurred due to mirrored pose estima-

Figure 34 – Distances from MCL and odometry to ground-truth over time in 2 rounds of the rnd scenario, using fixed set sizes of 100 samples, without prior localization knowledge. The x- and y-axis indicate the elapsed time and distances in seconds and meters, respectively.

(a) Successful localization. (b) Erroneous localization.



Source: Author

Table 6 – Localization success rates on each scenario using adaptive and non-adaptive set sizes.

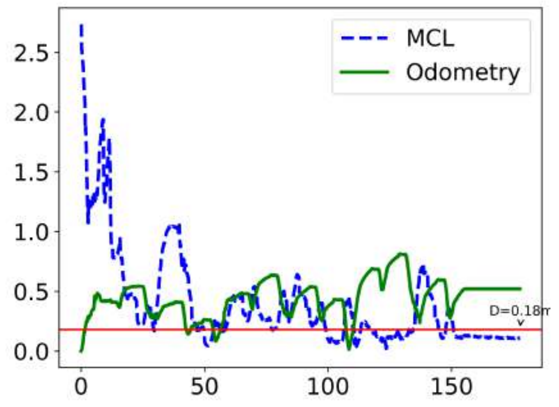
Scenario	MCL (M=100)	Adaptive MCL
rnd_01	1/5	5/5
sqr_02	3/5	2/5
igs_03	2/5	2/5

Source: Author

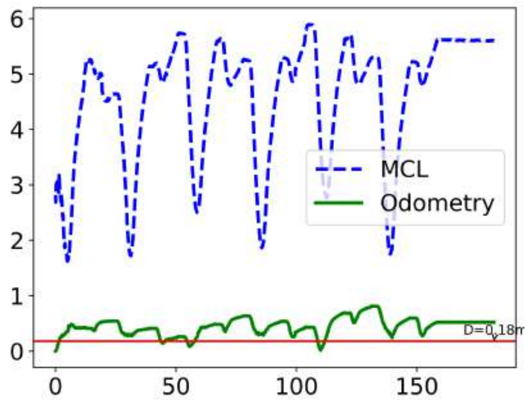
tions. As for rnd scenarios, in comparison to the non-adaptive MCL, we observe that the initial number of 200 particles provides more information for converging to the correct pose while the robot is looking towards the goal, avoiding mirroring issues. While the goal is detected, the algorithm's confidence grows and particles not looking towards the goal are removed, reducing the search space and improving the robustness against mirrored poses.

Figure 35 – Errors and trajectories comparison from MCL and odometry to ground-truth over time in 2 rounds of the sqr scenario, using fixed set sizes of 100 samples, without prior localization knowledge. The x- and y-axis indicate the elapsed time and distances in seconds and meters, respectively.

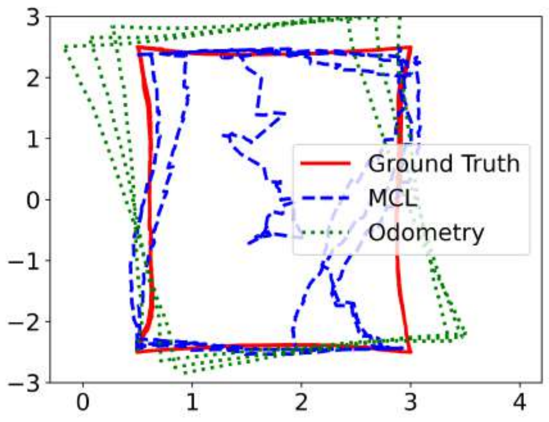
(a) Distances from successful localization.



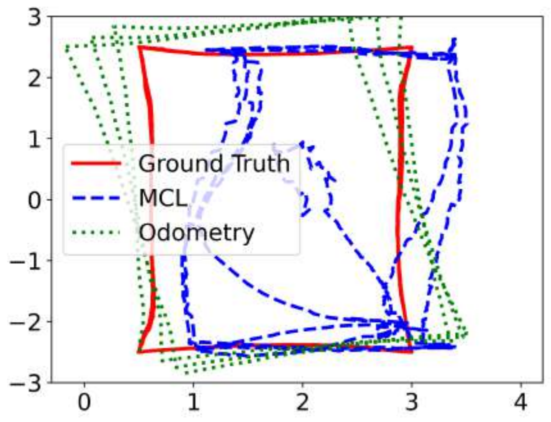
(b) Distances from erroneous localization.



(c) Trajectories from successful localization.



(d) Trajectories from erroneous localization. The algorithm converged to a mirrored pose.



Source: Author

7 FINAL CONSIDERATIONS

This thesis presented a solution for estimating a mobile robot's self-localization in dynamic environments under a hardware constrained platform. An onboard monocular camera is used for acquiring information, employing a CNN-based method for detecting objects and vertical scans for detecting points of landmarks. Detected elements are transformed to relative positions using the camera intrinsic and extrinsic parameters relative to the robot's axis. Also, the robot's odometry is estimated from the wheels' encoders, applying forward kinematics transformations, and gyroscope measurements. Information from odometry and onboard vision are combined into a Monte Carlo Localization (MCL) algorithm for regressing the robot's pose over time.

MCL was enhanced based on techniques proposed by other researches from RoboCup leagues, which were proven to increase the algorithm's robustness regarding erroneous measurements and imprecise odometry tracking in the robot soccer environment. In addition, we propose a novel measure of confidence for estimating the quality of the current MCL particles by applying the measurement model to the resulting state from self-localization, which in case was computed from the samples' weighted average. We also implement a mapping for adapting the number of particles during execution based on the current confidence, increasing the algorithm's overall processing speed while also maintaining its capability to track the robot's state.

We evaluate these methods within the RoboCup Small Size League environment, running our proposed onboard vision and self-localization pipelines on a 4GB NVIDIA Jetson Nano, which receives odometry updates from an ARM Cortex-M7 microcontroller. The resulting system was able to localize the robot's pose using the SSL soccer field boundaries and goals as references, while also detecting dynamic objects of the SSL environment, namely balls and other robots, achieving up to 15 FPS processing speeds.

In this chapter we review the conclusions of this research, which key insights and drawbacks are summarized in Table 7. In section 7.1, we discuss to what extent the defined objectives were accomplished. Section 7.2 depicts the main advantages and scientific contributions of this thesis. In contrast, section 7.3 presents drawbacks and limitations of our proposed methods and experiments. Lastly, section 7.4 suggests future work for enhancing the methods from this research.

Table 7 – Review of main contributions and drawbacks from our work.

Platform	**Environment**	**Key Insights**	**Drawbacks**
SSL Omnidirectional Robot (hardware specifications in Table 2)	RoboCup Small Size League	- Adapts modelings from multiple RoboCup soccer leagues to the SSL - Proposes conditions for resampling - Estimates the localization confidence based on measurements from multiple iterations - Adapts the set size and resample deviation based on the current confidence	- Does not use field lines for localization - Converges to symmetric poses - Requires specialized knowledge for tuning parameters - Experiments conducted in non-dynamic situations with a reduced environment

Source: Author

7.1 OBJECTIVES CONCLUSION

Three main objectives were defined within the Introduction of this thesis: introducing a solution for estimating an SSL robot's self-localization using onboard sensing and processing only (I); proposing a confidence metric for evaluating the localization quality during execution (II); and proposing an approach for increasing the processing speed of MCL-based self-localization algorithms (III).

The architecture and pipelines presented in section 4.1 were built upon the hardware and functional modules from previous work that proved to fit the needed requirements for building an autonomous SSL robot (MELO et al., 2022). We improve the onboard vision processing pipeline with detection of more SSL elements and add a new self-localization module, showing how it can be integrated into the system.

Results showed that our proposed architecture was able to localize and maintain the robot's true pose in scenarios where a seed of its initial position are given, which is the case for the RoboCup SSL Vision Blackout self-localization challenge. However, the experiments were conducted within a reduced version of the SSL field, which does not provide enough information to solve the SSL self-localization problem to its full extent.

Objectives II and III were addressed in section 4.3, which presents a novel method for computing the quality of the current distribution and how to adapt the number of particles of the MCL algorithm based on this estimation. The approach was able to drastically increase the system's computation speed, while also maintaining the capability to track the robot's pose. Also, the confidence measure may be useful for implementing active localization, i.e., making decisions and movements based on the current localization confidence.

7.2 IMPROVEMENTS AND CONTRIBUTIONS

This thesis presents techniques for pushing the state-of-the-art towards building more autonomous robots in the RoboCup SSL. We improve the architecture proposed by MELO et al. (2022), integrating a self-localization capability that allows to implement more effective state machines and planning more efficient paths for solving soccer tasks autonomously.

A public dataset containing onboard-recorded and ground-truth navigation data was generated and used for evaluating the methods proposed in this thesis. This data can be used not only for developing self-localization and tracking algorithms, but also for evaluating onboard vision processing solutions. It allows for testing and evaluation under various conditions, and objective comparisons can be realized by using the presented metrics and baseline results.

Section 4.2 presents how solutions from other RoboCup leagues can be combined and integrated into the MCL algorithm for improving its robustness regarding erroneous measurements and imprecise odometry estimations. Also, we propose conditions for choosing whether resampling shall be performed or not. A systematic resampling algorithm was employed due to its reduced computational complexity.

Lastly, a method for measuring the MCL distribution quality was presented, based on the likelihood of the current average particle, i.e., the weighted sum of the MCL samples. We propose a technique for adapting the samples' number and spreading during execution, based on the current confidence. Even though the approach was validated within the SSL environment, it can be easily reproduced and may also bring benefits to other applications.

We seek to expand this work and apply it as a conference paper or journal. By the time this thesis was written, the following paper have published during our researches:

- **MELO, J. G.**; MARTINS, F.; CAVALCANTI, L.; FERNANDES, R.; ARAúJO, V.; JOAQUIM, R.; MONTEIRO, J. G.; BARROS, E. Towards an autonomous robocup small size league robot. In: 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE). [S.l.: s.n.], 2022. p. 1–6.
- **MELO, J. G.**; BARROS, E. An embedded monocular vision approach for ground-aware objects detection and position estimation. In: RoboCup 2022:. [S.l.]: Springer International Publishing, 2023. p. 100–111. ISBN 978-3-031-28469-4.

7.3 DRAWBACKS AND LIMITATIONS

One drawback from our proposed confidence estimation and set size adaptation approaches is that they require parameters to be empirically tuned, not presenting an analytical estimation for their values. However, other researches in the self-localization problem showed that applying particle filters demands a proper configuration of a variety of different parameters (BURCHARDT; LAUE; RÖFER, 2011). Therefore, this drawback is a common characteristic in MCL-based algorithms.

The implemented system performed poorly in the self-localization problem when no information of its initial position was given, achieving low success rates using both adaptive and non-adaptive MCL. We believe this result yields from the lack of information due to using a single scan line detecting field boundaries only. Thus, field markings and multiple scan lines are necessary for improving the self-localization accuracy and success rates.

The experiments were limited to non-dynamic situations in a reduced half-field SSL configuration. Employing other moving robots and ball would evaluate our capability to detect these objects on movement and avoid them during line scans for detecting field elements. Using a complete field configuration causes more distant measurements, specially for field boundaries, highlighting again the importance of field markings detection. Also, it allows to evaluate our algorithm's robustness regarding mirrored pose estimations. Therefore, the ultimate validation for a self-localization technique in the SSL would require experiments on complete field with other moving objects.

7.4 FUTURE WORK

As discussed in the previous section and presented in experiments results, the perception of SSL field elements (goals, markings, and boundaries) still leaves room for several improvements. Firstly, more reliable and faster approaches for detecting field markings must be investigated. More robust color segmentation and more precise camera parameters can be learned from data, allowing to improve field points detection and relative position estimation. Also, the CNN-based objects detection approach can be trained for detecting field crossings as well (SZEMENYEI; ESTIVILL-CASTRO, 2019).

Analysis from time computation of self-localization showed that likelihood computation is the most time consuming part of the algorithm. This step is independent for each sample of the MCL and, thus, can be implemented as parallel operations. This approach may bring benefits to the algorithm's processing speed without causing accuracy losses.

The proposed self-localization algorithms require tuning of several parameters. In this thesis, these parameters were adjusted by observing the distribution's behavior in multiple tests. BURCHARDT; LAUE; RÖFER (2011) showed that Particle Swarm Optimization (PSO) can be employed for finding a set of parameters that leads to more precise position estimates than hand-tuned ones, and we shall investigate this approach for tuning our proposed methods.

Lastly, the confidence estimation and set size adaptation techniques presented in section 4.3 can be easily implemented and evaluated in other environments and platforms. Evaluating their performances within other applications would enhance the proposed methods' contribution the scientific community, specially regarding MCL-based algorithms.

REFERENCES

ABBENSETH, J.; OMMER, N. Position control of an omnidirectional mobile robot. In: . [S.l.: s.n.], 2015.

ALBUQUERQUE, D.; CASTRO, J.; RIBEIRO, S.; HEINECK, T. Requirements engineering for robotic system: A systematic mapping study. In: . [S.l.: s.n.], 2017.

BEN-ISRAEL, T. N. E. G. A. *Generalized Inverses*. 2nd. ed. [S.l.]: Springer New York, NY, 2003. ISBN 9780387002934.

BURCHARDT, A.; LAUE, T.; RÖFER, T. Optimizing particle filter parameters for self-localization. In: SOLAR, J. R. del; CHOWN, E.; PLOEGER, P. G. (Ed.). *RoboCup 2010: Robot Soccer World Cup XIV*. [S.l.]: Springer; Heidelberg; http://www.springer.de/, 2011. (Lecture Notes in Artificial Intelligence, v. 6556), p. 145–156.

CAVALCANTI, L.; JOAQUIM, R.; BARROS, E. Optimized wireless control and telemetry network for mobile soccer robots. In: ALAMI, R.; BISWAS, J.; CAKMAK, M.; OBST, O. (Ed.). *RoboCup 2021: Robot World Cup XXIV*. Cham: Springer International Publishing, 2022. p. 177–188. ISBN 978-3-030-98682-7.

COVER, T. M.; THOMAS, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006. ISBN 0471241954.

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. Monte carlo localization for mobile robots. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. [S.l.: s.n.], 1999. v. 2, p. 1322–1328 vol.2.

DOUCET, A.; GODSILL, S.; ANDRIEU, C. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, Kluwer Academic Publishers, USA, v. 10, n. 3, p. 197–208, jul 2000. ISSN 0960-3174. Available at: <https://doi.org/10.1023/A:1008935410038>.

ELFRING, J.; TORTA, E.; MOLENGRAFT, R. van de. Particle filters: A hands-on tutorial. *Sensors*, v. 21, n. 2, 2021. ISSN 1424-8220. Available at: <https://www.mdpi.com/1424-8220/21/2/438>.

FOX, D. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, v. 22, p. 1003 – 985, 2003.

FOX, D.; BURGARD, W.; THRUN, S. Markov localization for mobile robots in dynamic environments. *J. Artif. Int. Res.*, AI Access Foundation, El Segundo, CA, USA, v. 11, n. 1, p. 391–427, jul 1999. ISSN 1076-9757.

GOUAILLIER, D.; HUGEL, V.; BLAZEVIC, P.; KILNER, C.; MONCEAUX, J.; LAFOURCADE, P.; MARNIER, B.; SERRE, J.; MAISONNIER, B. The nao humanoid: a combination of performance and affordability. *CoRR*, abs/0807.3223, 07 2008.

GUTMANN, J.-S. Markov-kalman localization for mobile robots. In: *2002 International Conference on Pattern Recognition*. [S.l.: s.n.], 2002. v. 2, p. 601–604 vol.2.

GUTMANN, J.-S.; BURGARD, W.; FOX, D.; KONOLIGE, K. An experimental comparison of localization methods. In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. [S.l.: s.n.], 1998. v. 2, p. 736–743 vol.2.

GUTMANN, J.-S.; FOX, D. An experimental comparison of localization methods continued. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2002. v. 1, p. 454–459 vol.1.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2. ed. USA: Cambridge University Press, 2003. ISBN 0521540518.

HE, S.; SONG, T.; WANG, P.; DING, C.; WU, X. An enhanced adaptive monte carlo localization for service robots in dynamic and featureless environments. *J. Intell. Robotics Syst.*, Kluwer Academic Publishers, USA, v. 108, n. 1, may 2023. ISSN 0921-0296. Available at: <https://doi.org/10.1007/s10846-023-01858-7>.

JOHNSON, N.; KOTZ, S. *Continuous Univariate Distributions*. J. Wiley, 1970. (Continuous Univariate Distributions, v. 1). ISBN 9780471446262. Available at: <https://books.google.com.br/books?id=SQ7vAAAAMAAJ>.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, I.; OSAWA, E. Robocup: The robot world cup initiative. In: *Proceedings of the First International Conference on Autonomous Agents*. New York, NY, USA: ACM, 1997. (AGENTS '97), p. 340–347. ISBN 0-89791-877-0. Available at: <http://doi.acm.org/10.1145/267658.267738>.

KITANO, H.; FUJITA, M.; ZREHEN, S.; KAGEYAMA, K. Sony legged robot for robocup challenge. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. [S.l.: s.n.], 1998. v. 3, p. 2605–2612 vol.3.

KLODMANN, J.; SCHLENK, C.; HELLINGS-KUß, A.; BAHLS, T.; UNTERHINNINGHOFEN, R.; ALBU-SCHäEFFER, A.; HIRZINGER, G. An introduction to robotically assisted surgical systems: Current developments and focus areas of research. *Current Robotics Reports*, v. 2, p. 1–12, 09 2021.

KOLAR, P.; BENAVIDEZ, P.; JAMSHIDI, M. Survey of datafusion techniques for laser and vision based sensor integration for autonomous navigation. *Sensors*, v. 20, n. 8, 2020. ISSN 1424-8220. Available at: <https://www.mdpi.com/1424-8220/20/8/2180>.

KOSE, H.; AKIN, H. L. The reverse monte carlo localization algorithm. *Robotics and Autonomous Systems*, v. 55, p. 480–489, 06 2007.

KUKA. Hello industry 4.0 we go digital. *KUKA Aktiengesellschaft*, 2016.

LAUE, T.; RÖFER, T. Getting upright: Migrating concepts and software from four-legged to humanoid soccer robots. In: PAGELLO, E.; ZHOU, C.; MENEGATTI, E. (Ed.). *Proceedings of the Workshop on Humanoid Soccer Robots in conjunction with the 2006 IEEE International Conference on Humanoid Robots*. Genoa, Italy: [s.n.], 2006.

LAUE, T.; RÖFER, T. Particle filter-based state estimation in a competitive and uncertain environment. In: *Proceedings of the 6th International Workshop on Embedded Systems*. [S.l.]: Vaasa, Finland, 2007.

LI, T.; BOLIC, M.; DJURIC, P. M. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, v. 32, n. 3, p. 70–86, 2015.

MARTINS, F. B.; MACHADO, M. G.; BASSANI, H. F.; BRAGA, P. H. M.; BARROS, E. S. rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: ALAMI, R.; BISWAS, J.; CAKMAK, M.; OBST, O. (Ed.). *RoboCup 2021: Robot World Cup XXIV*. Cham: Springer International Publishing, 2022. p. 165–176. ISBN 978-3-030-98682-7.

MELO, J. G.; BARROS, E. An embedded monocular vision approach for ground-aware objects detection and position estimation. In: *RoboCup 2022:*. [S.l.]: Springer International Publishing, 2023. p. 100–111. ISBN 978-3-031-28469-4.

MELO, J. G.; MARTINS, F.; CAVALCANTI, L.; FERNANDES, R.; ARAúJO, V.; JOAQUIM, R.; MONTEIRO, J. G.; BARROS, E. Towards an autonomous robocup small size league robot. In: *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. [S.l.: s.n.], 2022. p. 1–6.

MUZIO, A.; AGUIAR, L.; MáXIMO, M. R.; PINTO, S. C. Monte carlo localization with field lines observations for simulated humanoid robotic soccer. In: *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*. [S.l.: s.n.], 2016. p. 334–339.

PANIGRAHI, P.; BISOY, S. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University - Computer and Information Sciences*, v. 34, 03 2021.

ROBOCUP. *RoboCup Standard Platform League (NAO) Rule Book*. 2023. Available at: <https://spl.robocup.org/wp-content/uploads/SPL-Rules-2023.pdf>.

ROBOTICS, S. *Nao v6 Datasheet*. 2018. Available at: <https://www.generationrobots.com/media/Specifications_NAO6.pdf>.

RÖFER, T.; JÜNGEL, M. Fast and robust edge-based localization in the sony four-legged robot league. In: POLANI, D.; BROWNING, B.; BONARINI, A.; YOSHIDA, K. (Ed.). *RoboCup 2003: Robot Soccer World Cup VII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 262–273. ISBN 978-3-540-25940-4.

RÖFER, T.; LAUE, T.; BAUDE, A.; BLUMENKAMP, J.; FELSCH, G.; FIEDLER, J.; HASSELBRING, A.; HASS, T.; OPPERMANN, J.; REICHENBERG, P.; SCHRADER, N.; WEISS, D. *B-Human Team Report and Code Release 2019*. 2019. Only available online: <http://www.b-human.de/downloads/publications/2019/CodeRelease2019.pdf>.

RÖFER, T.; LAUE, T.; HASSELBRING, A.; MONNERJAHN, L. M.; MATSCHULL, N.; PLECHER, L. B-Human 2021 – playing soccer out of the box. In: ALAMI, R.; BISWAS, J.; CAKMAK, M.; OBST, O. (Ed.). *RoboCup 2021: Robot World Cup XXIV*. [S.l.]: Springer, 2022. (Lecture Notes in Artificial Intelligence, v. 13132), p. 302–313.

RÖFER, T.; LAUE, T.; THOMAS, D. Particle-filter-based self-localization using landmarks and directed lines. In: BREDENFELD, A.; JACOFF, A.; NODA, I.; TAKAHASHI, Y. (Ed.). *RoboCup 2005: Robot Soccer World Cup IX*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 608–615. ISBN 978-3-540-35438-3.

SEEKIRCHER, A.; LAUE, T.; RÖFER, T. Entropy-based active vision for a humanoid soccer robot. In: SOLAR, J. R. del; CHOWN, E.; PLOEGER, P. G. (Ed.). *RoboCup 2010: Robot Soccer World Cup XIV*. [S.l.]: Springer; Heidelberg; http://www.springer.de/, 2011. (Lecture Notes in Artificial Intelligence, v. 6556), p. 1–12. Best Paper Award.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to Autonomous Mobile Robots*. 2nd. ed. [S.l.]: The MIT Press, 2011. ISBN 0262015358.

SMALL SIZE LEAGUE TECHNICAL COMMITTEE. *RoboCup 2022 SSL Vision Blackout Technical Challenge Rules*. 2022. <https://robocup-ssl.github.io/technical-challenge-rules/2022-ssl-vision-blackout-rules.pdf>, last checked on 2023-05-23.

SMALL SIZE LEAGUE TECHNICAL COMMITTEE. *Rules of the RoboCup Small Size League 2022*. 2022. <https://robocup-ssl.github.io/ssl-rules/2022/sslrules.pdf>, last checked on 2022-07-11.

SRINIVAS, S.; RAMACHANDIRAN, S.; RAJENDRAN, S. Autonomous robot-driven deliveries: A review of recent developments and future directions. *Transportation Research Part E: Logistics and Transportation Review*, v. 165, p. 102834, 2022. ISSN 1366-5545. Available at: <https://www.sciencedirect.com/science/article/pii/S1366554522002150>.

STRAKA, O.; ŜIMANDL, M. A survey of sample size adaptation techniques for particle filters. *IFAC Proceedings Volumes*, v. 42, n. 10, p. 1358–1363, 2009. ISSN 1474-6670. 15th IFAC Symposium on System Identification. Available at: <https://www.sciencedirect.com/science/article/pii/S1474667016388401>.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN 1848829345.

SZEMENYEI, M.; ESTIVILL-CASTRO, V. ROBO: robust, fully neural object detection for robot soccer. *CoRR*, abs/1910.10949, 2019. Available at: <http://arxiv.org/abs/1910.10949>.

THRUN, S. Particle filters in robotics. In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. (UAI'02), p. 511–518. ISBN 1558608974.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. [S.l.]: The MIT Press, 2005. ISBN 0262201623.

VELOSO, M.; UTHER, W.; FIJITA, M.; ASADA, M.; KITANO, H. Playing soccer with legged robots. In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. [S.l.: s.n.], 1998. v. 1, p. 437–442 vol.1.

YU, L.; YANG, E.; REN, P.; LUO, C.; DOBIE, G.; GU, D.; YAN, X. Inspection robots in oil and gas industry: a review of current solutions and future trends. In: *2019 25th International Conference on Automation and Computing (ICAC)*. [S.l.: s.n.], 2019. p. 1–6.

ZACHIOTIS, G. A.; ANDRIKOPOULOS, G.; GORNEZ, R.; NAKAMURA, K.; NIKOLAKOPOULOS, G. A survey on the application trends of home service robotics. In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.: s.n.], 2018. p. 1999–2006.

ZICKLER, S.; LAUE, T.; BIRBACH, O.; WONGPHATI, M.; VELOSO, M. Ssl-vision: The shared vision system for the robocup small size league. In: BALTES, J.; LAGOUDAKIS, M. G.; NARUSE, T.; GHIDARY, S. S. (Ed.). *RoboCup 2009: Robot Soccer World Cup XIII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 425–436. ISBN 978-3-642-11876-0.