UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUAN CARLOS KLEIN

OPTIMIZATION OF INTELLIGENT SENSORIZATION SYSTEMS WITH MACHINE LEARNING APPLIED TO ROBOTIC LOCALIZATION

CURITIBA

2024

LUAN CARLOS KLEIN

OPTIMIZATION OF INTELLIGENT SENSORIZATION SYSTEMS WITH MACHINE LEARNING APPLIED TO ROBOTIC LOCALIZATION

Otimização de sistemas de sensorização inteligente com aprendizado de máquina aplicado à localização robótica

Thesis presented as a requirement for obtaining the degree of Master's Degree in Applied Computing in Postgraduate Program in Applied Computing (PPGCA-CT) of the Universidade Tecnológica Federal do Paraná.

Advisor: Prof. Dr. João Alberto Fabro

Co-advisor: Prof. Dr. Jose Luis Sousa Magalhaes Lima

CURITIBA

2024

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

LUAN CARLOS KLEIN

OPTIMIZATION OF INTELLIGENT SENSORIZATION SYSTEMS WITH MACHINE LEARNING APPLIED TO ROBOTIC LOCALIZATION

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Sistemas Computacionais.

Data de aprovação: 19 de Dezembro de 2024

Dr. Joao Alberto Fabro, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Andre Schneider De Oliveira, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Felipe Nascimento Martins, Doutorado - Hanze University Of Applied Sciences - Hanze Uas

Dr. Marco Aurelio Wehrmeister, Doutorado - Universidade Tecnológica Federal do Paraná

I offer my dedication to God, whose infinite wisdom has guided me on my journey. Furthermore, I would like to dedicate this work to my parents, Méri and Leonor, without whose unwavering support, inspiration, and example, none of this would have been possible. Finally, I dedicate this work to my brother, Jean, who has inspired me greatly.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all my professors and colleagues, including UTFPR-Curitiba and IPB, in particular my advisors José Lima and João Fabro, and the advisor of the previous work in IPB, Felipe N. Martins. I would also like to thank João Braun, João Mendes, and Profa. Ana Isabel for their assistance during the research and development phase of the project. Their guidance and willingness to share their knowledge were greatly appreciated.

To become good at anything you have to know how to apply basic principles. To become great at it, you have to know when to violate those principles (KASPAROV, 2017).

RESUMO

A capacidade de se localizar de maneira eficiente é uma característica essencial para robôs autônomos. Devido à sua importância, esse tema tem sido amplamente estudado e debatido ao longo dos anos, com várias abordagens propostas. Este trabalho se baseia em pesquisas anteriores que sugerem o uso de técnicas de inteligência artificial para a localização no contexto da competição de robótica RobotAtFactory 4.0. Em particular, este estudo foca na otimização dos hiperparâmetros de uma Rede Neural Artificial do tipo *MultiLayer Perceptron* (MLP), investigando seu desempenho em diferentes situações dentro do mesmo cenário. Os resultados mostraram uma melhoria de até 60% na precisão das estimativas, quando comparado a modelos sem otimização. Outro resultado interessante foi a possibilidade de reutilizar otimizações em diferentes cenários, o que se apresenta como uma alternativa promissora em casos onde o custo computacional para encontrar a melhor configuração é muito elevado. Essa solução oferece uma abordagem eficaz para reduzir o custo computacional, ao mesmo tempo em que melhora o desempenho dos modelos de aprendizagem de máquina.

Palavras-chave: localização; aprendizado de máquina; otimizaçäo de hiperparâmetros; competiçäo de robótica; inteligência artificial.

ABSTRACT

Efficient localization is an essential feature for autonomous robots. Due to its importance, this topic has been widely studied and debated over the years, with several approaches proposed. This study builds on previous research that suggests the use of artificial intelligence techniques for localization in the context of the RobotAtFactory 4.0 robotics competition. In particular, this study focuses on optimizing the hyperparameters of the Multilayer Perceptron (MLP), evaluating its performance in different situations within the same scenario. The results showed an improvement of up to 60% in the estimates' precision compared to models without optimization. Another interesting result was the possibility of reusing optimizations between different scenarios, a promising alternative in cases where the computational cost of finding the best configuration is very high. This solution offers an effective approach to reducing the computational cost while improving the performance of machine learning models.

Keywords: localization; machine learning; hyperparameter optimization; robotics competition; artificial intelligence.

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

Pseudo-Acronyms

2D	2 Dimensions
3D	3 Dimensions
AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DGPS	Differential GPS
DL	Deep Learning
DNN	Deep Neural Networks
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
GPS	Global Positioning System
GUI	Graphical User Interface
HPO	Hyperparameter Optimization
ICP	Iterative Closest Point
IoT	Internet of Things
IPB	Instituto Politénico de Brangança
KF	Kalman Filter
KNN	K-nearest neighbors
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
ML	Machine Learning

MLP	Multilayer Perceptron
NDT	Normal Distribution Transform
RADAR	Radio Detection and Ranging
RaF	RobotAtFactory 4.0
RBIC	Rank-Based Iterative Clustering
RF	Random Forest
RFID	Radio-Frequency Identification
RMSE	Root Mean Squared Error
RPROP	Resilient Back-Propagation
SLAM	Simultaneous Localization and Mapping
SMBO	Sequential Model-Based Optimization
SVM	Support Vector Machine

SUMMARY

1 INTRODUCTION

Self-localization is a fundamental skill in several fields of knowledge, being a critical feature for animals to survive and perform activities in the wild, which can include humans from the beginning of history until the current time. In the context of autonomous robots self-localization, several approaches have been studied and developed over time, using different tools to improve accuracy and efficiency while facing the challenges of each specific scenario. In this study, the main goal is exploring hyperparameter optimization in a novel localization approach in the context of autonomous robots operating in a factory environment, using a specific robotic competition scenario to validate the approach.

1.1 Initial considerations

An Autonomous Mobile Robot (AMR) can be understood as a complex system composed of several aspects, with the fundamental behavior of each component being essential to the correct functionality and generality. One of these aspects, which deserves special attention due to its crucial importance, is localization. This concept relates to the knowledge of the robot's pose in the environment and greatly impacts the AMR's decision-making process. To estimate its pose, the robot can have an external localization system or be limited to using its sensors without external help.

In this study, the robot has to localize inside a specific environment without an external localization system. The environment selected for the evaluation and experiments is the one proposed by the RaF[1] (*Robot at Factory 4.0*) competition. This is a Portuguese competition, where an AMR has to move boxes through a limited field in the shortest possible time without external communication (except with the organization systems). In this competition, the robot has to be completely autonomous and can use any necessary tools/approaches to locate itself once it complies with the rules. One of the most used ways to perform self-localization is by using the predefined markers placed on the environment's floor, called ArUco (GARRIDO-JURADO *et al.*, 2014).

In the first part of this study, developed by the current author in their Master Thesis at Instituto Politénico de Brangança (IPB) (KLEIN, 2023), a new approach for the localization in the RaF context was proposed and validated, aiming to introduce the use of Machine Learning (ML) techniques. Furthermore, the feasibility validation of the models in the competition robot was performed (KLEIN *et al.*, 2023a). In contrast, the quality of the ML models was presented in (KLEIN *et al.*, 2023b), and the use of Deep Learning and, more specifically, CNN was presented in (KLEIN *et al.*, 2024a).

One of the points not explored in the previous study was optimizing the ML models, once they were based only on validating the possibility of use but not on increasing the models'

[1] https://www.festivalnacionalrobotica.pt/2023/robotfactory-4-0/

quality. Building on the previous work, this study aims to optimize the ML models and examine their behavior across specific variables, including the components of the pose (position and orientation) and the data collected for use by the models.

1.2 Context: RobotAtFactory 4.0 Competition

Many robot competitions worldwide have different goals and aim to encourage students and participants to develop new technologies. One of these competitions is RobotAtFactory 4.0, included in the Portuguese Robotics Open event, whose objective is to navigate a robot through a warehouse environment, with the primary goal being to transport the highest number of boxes in the shortest time. Boxes can be of three types, represented by different colored boxes of 90 x 60 x 65 mm (width x length x height).

Blue boxes represent “completely processed” pieces and should be moved to the output/outgoing warehouse. Green boxes are “Intermediate Parts” and should be carried to the Type B machines that, after processing, transform them into “completely processed” (blue) boxes. Finally, red boxes represent “raw materials”. They should first be carried to the Type A machines for processing, “transforming” them into “Intermediate Parts” (green) boxes and later follow the process of the green boxes to transform into blue boxes.

There are three rounds in the competition: (1) the robot needs to move wholly processed (blue) boxes directly from the incoming warehouse (places numbered from 50 to 53 in Figure 1) to the output warehouse (places numbered from 62 to 65); (2) the robot should move red and green boxes from the incoming warehouse to the output warehouse, in a way that the green boxes are processed in Type B machines (there are two such machines, with input positions 54 and 58, and output positions 55 and 59, respectively); (3) the robot should move red, green and blue boxes to the output warehouse. Green boxes should be first ”processed” in Type B machines; Red boxes should be processed through two kinds of processing machines, in order, first by a Type A and then by a Type B one. Type A machines have input in positions 56 and 60 and respective outputs in positions 57 and 61. After processing both machine types, the box should be moved to one free position in the ongoing warehouse. In each round, up to 4 colored boxes are in the incoming warehouse. The floor consists of a print on two A0 sheets and a flat layout, with several ArUco markers in the environment. Figure 1 presents an overview of the field. The origin of the reference frame is situated at the center of the field, where the green arrow indicates the y-axis, the red arrow indicates the x-axis and the numbers from 0 to 35 indicate the ID of each ArUco marker placed on the floor. In contrast, the blue numbers (from 50 to 65) indicate the ID of the ArUco markers placed on the walls.

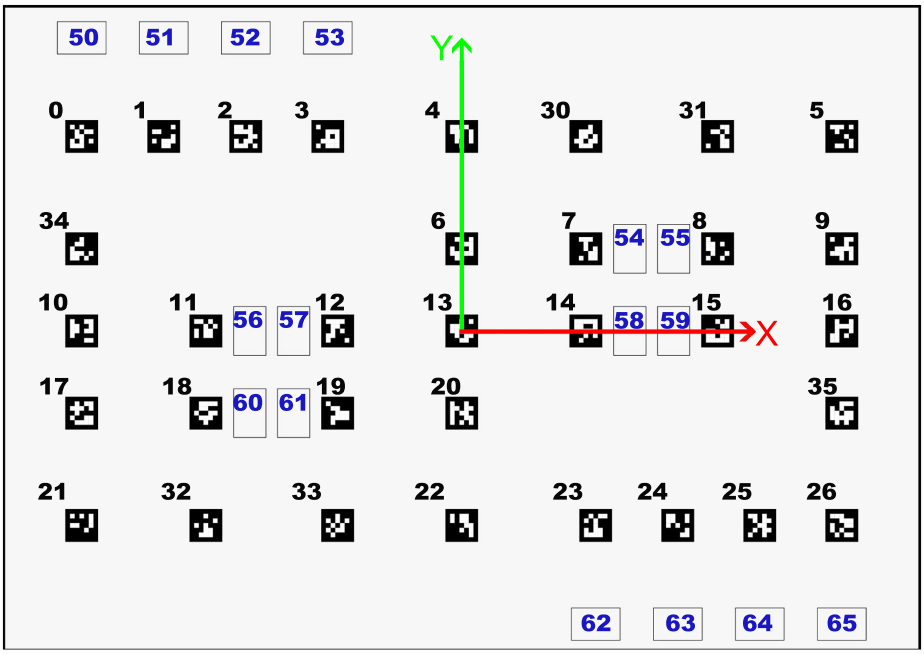


Figure 1 – The field-top view of the RaF competition, providing a visual representation of axes and the identification of ArUco markers. Source: (KLEIN *et al.*, 2023b).

1.2.1 Robot

There are several rules for the robot during the competition, such as fitting within a 30 x 30 x 30 cm cube and being fully autonomous. The robot cannot establish any communication with any external system that the organization has not explicitly provided. The organization's communication system includes information about each machine's expected average processing time and standard deviation. It informs the robots when every machine starts/ends the box processing.

An example of the robot architecture is presented in Figure 2, and was presented by (BRAUN *et al.*, 2022a) and (BRAUN *et al.*, 2022b). A Raspberry Pi is responsible for high-level control of the robot, including management of the RGB camera (which is placed in front of the robot), 2D Light Detection and Ranging (LiDAR) sensor, localization, navigation, and decision-making. Meanwhile, an Arduino handles the low-level control of the robot, such as the operation of motors, encoders, contact switches, and electromagnetic sensors used to detect the adequate adjustment of the robot to the box. Usually, the teams have an electromagnet that allows for transporting the boxes by grabbing/dragging them around. One example of the robot that was used in the competition presented in (BRAUN *et al.*, 2024) used a Raspberry PI 4B, Arduino Mega, and a camera with a resolution of 640x480p (RPI Cam V2).

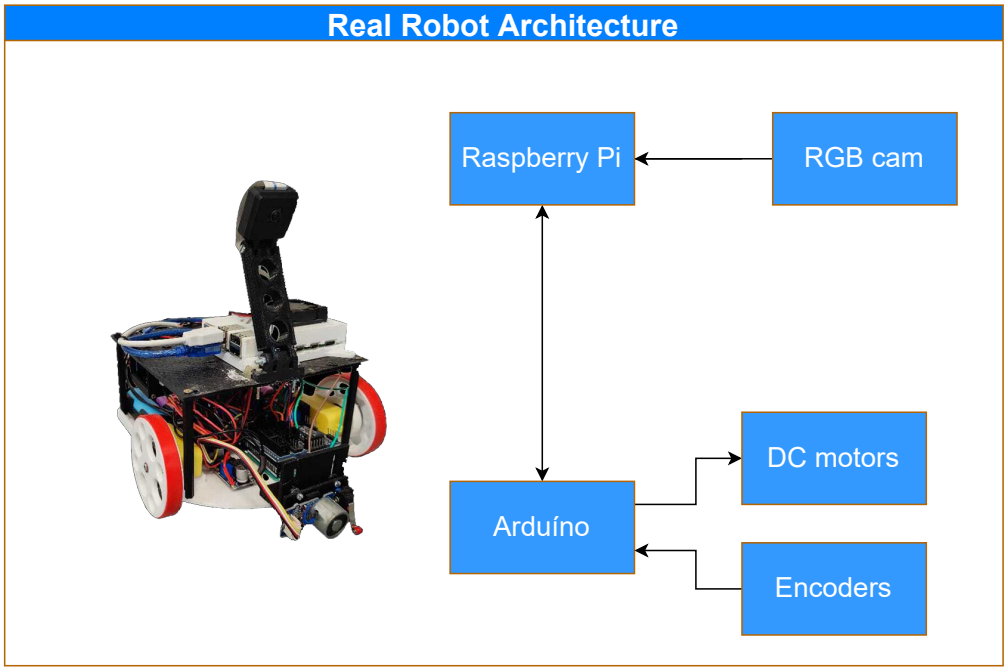


Figure 2 – Example of robot architecture. The Raspberry Pi is responsible for making decisions and controlling the RGB camera and LiDAR, while the Arduino manages additional physical components, including motors and electromagnetic sensors. Source: (KLEIN *et al.*, 2023b).

1.2.2 Real Environment

Figure 3 presents an overview of the real competition field, which, according to the official rules, had the dimension of 1.7 × 1.2 m. That way, it is always possible to identify the boxes' positions and each warehouse/machine by observing and localizing the ArUco markers in the environment. The competition organization provides each ArUco marker's ID, position, and orientation relative to the global reference frame.

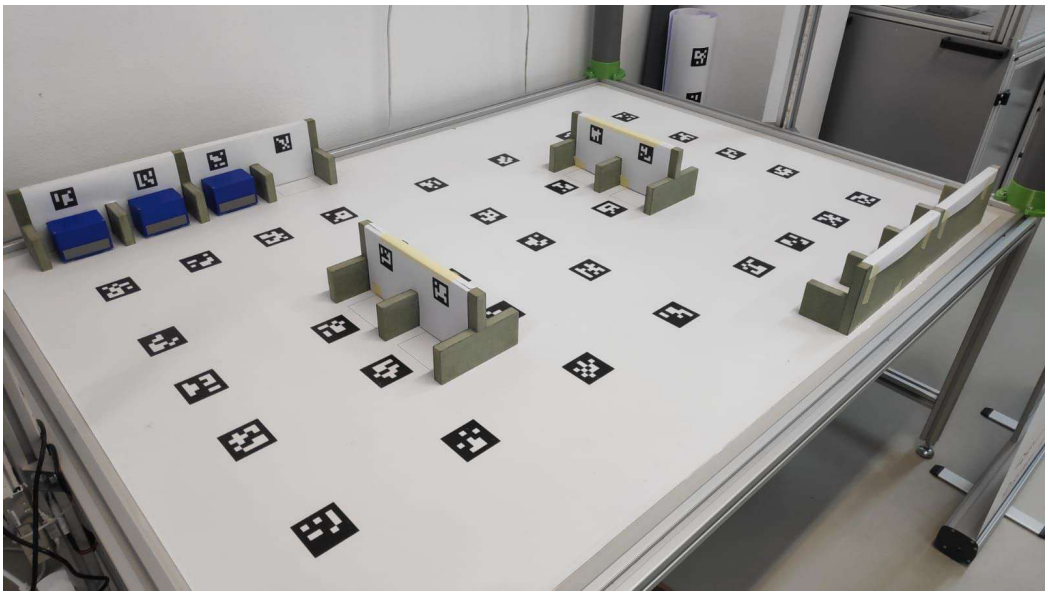


Figure 3 – Real competition field. Source: (KLEIN, 2023).

1.2.3 Realistic Simulator

The organizers of the competition have developed a simulator for the RaF competition[2] based on the real environment. The SimTwo simulator works with rigid-body dynamics interactions and constraints (COSTA *et al.*, 2011). With all its components, the simulator is presented in Figure 4. Several features are available in the simulator, such as an editor for XML files, which allows the user to define settings for the robot and its environment. A code editor in Pascal language allows the direct development of the robot programming (BRAUN *et al.*, 2022b). Developing scripts in other programming languages using a communication framework with the simulator is also possible. In addition, the image robot sensor in SimTwo is an RGB camera with a resolution of 640 × 480 pixels (COSTA *et al.*, 2011).

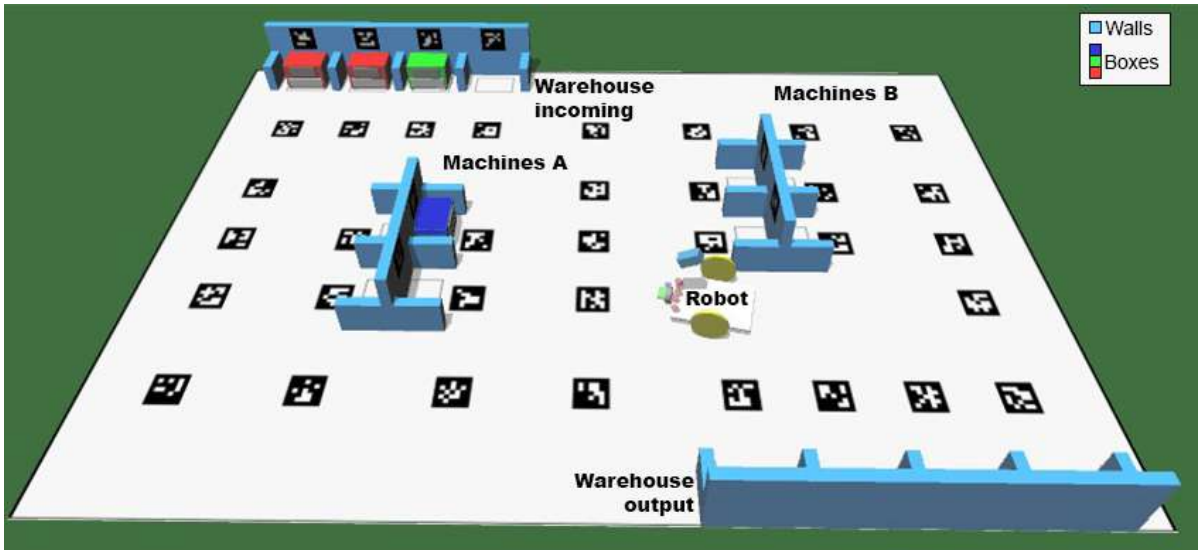


Figure 4 – Simulator scene showing the robot on the competition field and the boxes. Source: (KLEIN *et al.*, 2023b).

1.3 Motivation and Objectives

This study extends the previous master dissertation done by the author in (KLEIN, 2023). The initial goal of the earlier work was to validate and explore the newly proposed approaches to localization issues on the RaF competition using AI techniques. The main goal of that work was to avoid the need to know *a priori* (previously) the position of the ArUco Markers present in the scenario (BRAUN *et al.*, 2022a). Three papers were published based on the results of the previous work: (KLEIN *et al.*, 2023b; KLEIN *et al.*, 2023a; KLEIN *et al.*, 2024a). As an extension to the previous study, the present work aims to explore the optimization of machine learning models to check the possibility of increasing the quality of the models' pose estimations.

Three main research questions were proposed to guide the work's development:

[2] Available at https://github.com/P33a/SimTwo.

1. What are the challenges to finding an optimization for the MLP model in the RaF scenario?

2. Is there a correlation between the optimizations for different contexts, considering the RaF scenario?

3. Is it possible to apply optimizations found for one context in others, and what are the advantages and disadvantages associated?

It is important to emphasize that as one of the requirements of the reception notice (JOINT NOTICE No. 5/2023 — PROPPG/PROGRAD- reception of graduates of double degrees) of the Master's in Applied Computing at UTFPR, the paper (KLEIN *et al.*, 2024b) related to this work was published at the International Conference on Optimization, Learning Algorithms, and Applications (OL2A 2024), which took place at University of La Laguna (San Cristóbal de La Laguna - Tenerife, Spain) between 24th and 26th of July, 2024.

1.4 Structure

The work is divided into five more chapters. Chapter 2 presents state-of-the-art in localization and ML, also presenting the related works; Chapter 3 presents an overview of the previous work; Chapter 4 presents the methodology used in the current study; Chapter 5 presents the results and discussions; Finally, Chapter 6 presents the conclusions and future works.

2 LITERATURE REVIEW

This chapter presents the literature review, showing the state-of-the-art in three main parts: Section 2.1 presents the concepts and the approaches in localization; Section 2.2 presents some concepts and techniques related to the machine learning (and deep learning), with a special focus with their use in localization and; Section 2.3 presents concepts about optimizations, especially focused on the machine learning models optimizations.

2.1 Localization

Localization is a fundamental capability required by AMR in the most diverse scenarios. Knowledge of the pose (which consists of the position and orientation) is crucial in decision-making. Besides the knowledge of the pose, it is also necessary to know the uncertainly related to the estimation, since the decision of future actions based on the belief that the pose is perfect can result in catastrophic results (HUANG; DISSANAYAKE, 2016).

A fundamental concept in robotics is odometry, also known as dead reckoning (HUANG; DISSANAYAKE, 2016), which consists of sensors onboard the robot structure to track its movement. It is possible to estimate the robot's pose using mathematical motion models and the data from the sensors.

The specifics of the localization process are contingent upon the dimensions of the problem at hand (STACHNISS, 2021)[1]. For instance, in scenarios in 2 Dimensions (2D), there are 3 degrees of freedom: (*x, y,* θ), with *x* and *y* being the position and θ the orientation; in scenarios in 3 Dimensions (3D) there are 6 degrees of freedom: (*x, y, z,* α_{roll}, α_{pitch}, α_{yaw}), with *x*, *y* and *z* being the the coordinates of the position and the α_{roll}, α_{pitch}, α_{yaw} being the rotations related to the axes *x*, *y* and *z*, respectively.

The concept of a mobile robot being deployed in an uncharted environment at an unknown location and incrementally building a consistent environment map while simultaneously determining its location in this map is known as Simultaneous Localization and Mapping (SLAM) (DURRANT-WHYTE; BAILEY, 2006). Also related to the topic is the *kidnapped robot problem*, which consists of the situation where the autonomous robot in operation is moved to an arbitrary position (CHOSET *et al.*, 2005).

The localization can also be classified into several categories according to different criteria. (1) Related to the moment of the execution: offline and online. In offline localization, the data can be recorded and processed after the execution of a task, although an answer during the execution is not necessary. An example is the use of localization to update an environment map. In contrast, online localization is when the system requires knowledge of its localization at execution. This is exemplified by the navigation of an autonomous vehicle (STACHNISS, 2021);

[1] The complete course is available at http://www.ipb.uni-bonn.de/msr1-2021/. Accessed on November 1, 2024.

(2) the initial position knowledge: Global Localization and Tracking Pose (STACHNISS, 2021). In the first one, the system can be situated anywhere globally, and its initial position is unknown. Conversely, in the Tracking Pose stage, the initial system's location is already known; (3) according to outdoor and indoor environments. The most widely used approach for the first is the Global Positioning System (GPS), which presents the limitation that it may not be available indoors due to the limitations of blocked satellite signals or attenuated by structures such as walls (GREWAL; WEILL; ANDREWS, 2007). For indoors, several approaches have been developed over time, which will be discussed further in the section.

2.1.1 Markov Localization

Markov localization is an approach to the global localization issue (FOX, 1998) that discretizes the space of possible poses and manipulates discrete probability distributions (HUANG; DISSANAYAKE, 2016). The environment is presumed to be static to simplify the explanation, but this approach can also be used in a dynamic environment (FOX; BURGARD; THRUN, 1999). It is a probabilistic algorithm that maintains a probability distribution in the space of all such hypotheses, using a histogram for each degree of freedom (FOX; BURGARD; THRUN, 1999).

The Markov localization has the initial beliefs and the probability of the robot being in each position. With the interactions with the world and data collection, it updates its beliefs and, ideally, decreases the uncertainty of the pose estimates to the point that the probability converges to the effective position of the robot. Some examples can be found in (SIEGWART; CHLI; LAWRANCE, 2014).

2.1.2 Monte Carlo Localization

First introduced as a bootstrap filter in (GORDON; SALMOND; SMITH, 1993) and further explained in (ARULAMPALAM *et al.*, 2002), it is a Global Localization approach that can also be used for Pose Tracking. In the Monte Carlo approach, a sample rate is used instead of a histogram, and the scenario discretization is unnecessary; that is, it does not make assumptions on linearity. This algorithm is a type of *Particle Filter* (FOX *et al.*, 1999; DELLAERT *et al.*, 1999).

The hypothesis of the variable of interest, in the current case, the robot localization, is represented by multiple samples (particles). Each hypothesis is associated with a weight, representing the likelihood that the hypothetical estimate is true. The weighted sum of all samples provides the pose estimate[2]. The recursive particle filter operates in two stages: "*predict*" and "*update*". Prediction occurs after each action, with each particle modified according to the existing model (the "*predict*" stage). Subsequently, all weights are recalculated based on the sensory information (the "*update*" stage) (REKLEITIS; DUDEK; MILIOS, 2003).

[2] A tutorial for particle filters is available at https://www.cim.mcgill.ca/~yiannis/particletutorial.pdf. Accessed on November 5, 2024.

2.1.3 Kalman Filter

Proposed by (KALMAN, 1960), the Kalman Filter (KF) is a mathematical approach that integrates system measurements with predictions of the system's anticipated behavior. The method employs feedback control to estimate a process. At a given moment, the filter estimates the state of the process and then obtains feedback in the form of measurements, which are subject to noise (BISHOP; WELCH *et al.*, 2001; MAYBECK, 1990; RUSSELL; NORVIG, 2010).

In a simplified way, there are two inputs in the filter: The first one is the initialization, which is performed only once and inputs the initial state and the uncertainty associated with that state; the other, a recurrent input in each cycle, inputs the measurement state and the associated uncertainty.

The outputs comprise the predicted state and the uncertainty associated with that prediction. The fundamental component of the filter is the update phase, during which the internal parameters are updated on a cyclical basis, and the subsequent state is predicted[3] (BECKER, 2022). This cycle is repeated at each time interval, the duration of which depends on the specific application in question.

The KF was initially designed to be linear and is also used in linear models. However, most of the real systems are nonlinear. To address these situations, some KF variations have been developed, and one of the most promising variations is Extended Kalman Filter (EKF) (BISHOP; WELCH *et al.*, 2001). A fundamental prerequisite for EKF-based localization is the capacity to associate measurements obtained with specific landmarks present in the environment (HUANG; DISSANAYAKE, 2016).

2.1.4 Least Square

The least squares is an offline approach, meaning the system needs all data to be available beforehand. Once the route has been calculated, it is possible to determine the discrepancy between the calculated and actual values (STACHNISS, 2021; HUANG *et al.*, 2001). Moreover, this methodology employs a Gaussian belief model, such as the KF. Frequently, this approach is used as a reference solution for other localization systems, including online approaches (STACHNISS, 2021).

2.1.5 Slidding Window Least Squares

Similar to the Least Square, but instead of using all the data to calculate the localization, the sliding window uses only the most recent observations (STACHNISS, 2021). This approach aims to achieve a performance superior to that of the KF while maintaining a computational cost

[3] Tutorial with examples is available at https://www.kalmanfilter.net/. Accessed on July 17, 2024.

less than that of the Least Squares (ZHANG, 2000). An example of using this approach is for external locations, such as in autonomous vehicles (WILBERS; MERFELS; STACHNISS, 2019).

2.1.6 Perfect Match

Perfect Match, first proposed in (LAUER; LANGE; RIEDMILLER, 2006), is an image-based approach initially developed for robot localization in the Robocup Midsize League. This algorithm minimizes the matching and fitting error between the acquired data and the environment map. The algorithm can be divided into three main parts: (1) matching error and gradient computation; (2) optimization routine based on the Resilient Back-Propagation (RPROP); and (3) covariance estimation using the second derivative (SOBREIRA *et al.*, 2019).

2.1.7 Iterative Closest Point

The Iterative Closest Point (ICP) introduced by (BEST, 1992) is a map-matching technique that seeks to minimize the Euclidean distance between the input data and a reference model. In the localization context, the minimizations correspond to the sensor and the environment map, respectively (SOBREIRA *et al.*, 2019). This is an interactive process until convergence is achieved. The two main steps of the approach are as follows: (1) the matching stage, which makes the data association, matching each “real” point to the closest point in the model; (2) the transformation stage, which is based on the previous stage, and which attempts to minimize the distance between the points. Other approaches have been developed based on the ICP technique, such as Point-to-Plane (RUSINKIEWICZ; LEVOY, 2001).

2.1.8 Normal Distribution Transform

Introduce by (BIBER; STRASSER, 2003) the Normal Distribution Transform (NDT) is a method created initially for 2D scanning and was later extended to work on 3D (MAGNUSSON *et al.*, 2009; MAGNUSSON, 2009). This approach is a map-matching algorithm that generates a smooth surface representation of the environment, modeled by a set of local probability density functions. It uses a set of reference points grouped in fixed-sized cells forming a voxel grid to build this representation. Then, for each cell of the voxel grid with at least a group of 6 points, the mean and covariance matrix are calculated (SOBREIRA *et al.*, 2019).

2.1.9 Fiducial Markers approach

To provide a reference point in the environment, fiducial markers were introduced as a possible approach with applications in several activities (KALAITZAKIS *et al.*, 2021). Using

only cameras, fiducial markers localization algorithms can calculate the camera's pose related to a maker seen in the image. Numerous types of fiducial markers have been developed over time, presenting differences in the format of the markers. A comparison between four types of markers(ARTag, AprilTag, ArUco, and STag) is available in (KALAITZAKIS *et al.*, 2020).

The binary square is one of the most used types of the fiducial marker, particularly the ArUco Marker developed in (GARRIDO-JURADO *et al.*, 2014). An example of an ArUco marker is presented in Figure 5.

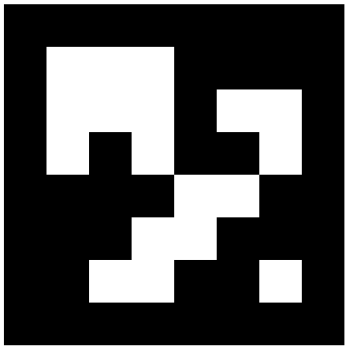


Figure 5 – Example of a Fiducial ArUco Marker[4].

The ArUco marker is implemented in the OpenCV library[5] and has specific pre-defined pose detection functions. The OpenCV library is a well-known library used in computational vision, and one of its specific uses is that it allows users to identify the ArUco and estimate its pose. In this way, the first step is the camera calibration to understand the distortion coefficients of the camera[6], and only needs to be performed once if the camera optics are not modified. Then, the library allows the ID identification, rotation, and translation vectors called *rvecs* and *tvecs* for each ArUco present in the image.

The translation (x,y,z) of the marker from the origin is represented by the *tvec*, which is expressed in units derived from the camera specifications, such as millimeters. The *rvec* represents a three-dimensional rotation vector that defines the axis of rotation and the rotation angle about that axis, thereby indicating the marker's orientation. The vector's direction indicates the rotation axis, while the vector's magnitude represents the rotation angle (in radians).

An example of using fiducial markers in localization is the RaF competition, which has several ArUco markers in the field, as presented in Section 1.2. Several approaches have been developed based on the use of ArUco markers and their pose identification competition, such as using analytical geometry (BRAUN *et al.*, 2022a) and using ML (KLEIN *et al.*, 2023b).

[4] Source to generate: https://chev.me/arucogen/. Accessed on August 24, 2024.
[5] https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html. Accessed on November 7, 2024.
[6] https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Accessed on September 15, 2024.

2.1.10 Artificial Intelligence approaches

The use of Artificial Intelligence (AI) approaches in localization has received more focus in recent years due to some aspects such as their capacity to learn from complex data, adaptation to changes, fusion sensors, and robustness against interferences. The study performed in (NESSA *et al.*, 2020) presents a survey about using ML in localization. The study explored methods that apply feature extraction and selection in the localization context. Some of the techniques presented in the study were Random Forest (RF), Support Vector Machine (SVM), K-nearest neighbors (KNN), and Artificial Neural Network (ANN). It demonstrates that this field is evolving and requires further research.

An example of the ML used in localization is in the work presented in (AHMAD *et al.*, 2024), which developed an ML approach for indoor localization in Internet of Things (IoT) environments, using techniques such as the SVM and nonlinear regression model applied with a radial basis function, achieving Root Mean Squared Error (RMSE) of 0.25m.

The application of unsupervised learning in indoor localization has been explored in (KOÇOĞLU, 2022), involving a comparative analysis of clustering methodologies, including k-Means and Fuzzy c-Means, with an accuracy range of 93% to 95%. The work presented in (MALLIK; DAS; CHOWDHURY, 2023) explored a semi-supervised approach for indoor localization, using an approach based on a Rank-Based Iterative Clustering (RBIC), with results showing accuracies between 94% to 99%. This kind of learning appears as an interesting approach besides the supervised learning (where the dataset must be labeled) and unsupervised learning (where experimenters should have some understanding of the regional divisions of the experimental area to assess the accuracy of clustering results)

However, traditional ML approaches can present weaknesses, such as KNN being computationally expensive for large datasets and having problems with high dimensionality at the same time that SVM depends on kernel choice and is sensitive to noise, decision trees may overfit, while Random Forest improves generalization but requires careful tuning and high computational cost (KERDJIDJ *et al.*, 2024).

Besides the conventional ML approaches, using Deep Neural Networks (DNN) is an interesting procedure, especially with the capability of extracting features from implicit neural models. The survey in (CHEN *et al.*, 2024) presents a deep review of the use of Deep Learning (DL) for visual localization and mapping. The authors conclude the DL can be a unique direction toward a future general-purpose SLAM system, providing advanced semantic understanding for robotic tasks and enabling autonomous learning and adaptation to new environments. In addition, the review in (KERDJIDJ *et al.*, 2024) focuses on using DL and transfer learning in indoor localization.

An example of DL is the use of the CNN, which uses images from a robot camera and other sensors that can help in the robot localization (ESFAHLANI *et al.*, 2022; ATANASYAN; ROSSMANN, 2019). The study performed by (KENDALL; GRIMES; CIPOLLA, 2015) used im-

ages to perform a localization location with six Degrees of Freedom (DoF) using GoogLeNet transfer learning (SZEGEDY *et al.*, 2015), operating indoors (obtaining approximately 2 meters and 6° accuracy for large scale) and outdoors (obtaining approximately 0.5 meter and 10° accuracy).

2.1.11 Alternatives approaches

Many other alternative approaches have been developed to address the localization issue. Some examples includes the use of infrared light (WANG; TAKAHASHI, 2018), video tracking at 6 DoF (NEBEKER, 2015), map-based probabilistic visual (BRUBAKER; GEIGER; URTASUN, 2015), acoustic beacons (OGISO *et al.*, 2015), image-based localization (SATTLER; LEIBE; KOBBELT, 2011), Radio-Frequency Identification (RFID) (PANIGRAHI; BISOY, 2022), LiDAR and Radio Detection and Ranging (RADAR) with map matching (YANASE *et al.*, 2022), and landmarks (AVGERIS *et al.*, 2019).

2.2 Machine Learning

The denomination Machine Learning refers to a computer technique in which an algorithm learns from data based on the patterns presented without explicit instructions. According to (RUSSELL; NORVIG, 2010), it is possible to define three types of learning: Unsupervised, supervised, and reinforcement. In the first one, no feedback is provided, and the learning is based on the patterns separating the data. The second, a set of input/output, is given, and the algorithm tries to represent the relations between input and output through a function. Finally, the reinforcement is closer to how humans learn, based on rewards and punishments after the actions. Plenty of techniques have been developed for each type of learning. More details about the techniques are available in (RUSSELL; NORVIG, 2010; GOODFELLOW; BENGIO; COURVILLE, 2016).

One of the most known challenges in ML development is the *underfitting* and *overfitting*. In summary, the first occurs when a model cannot learn enough about the problem, i.e., the model cannot obtain a low error value on the training set. Conversely, *overfitting* occurs when a model learns too much from the training dataset and is not able to expand the learning to other datasets of the same problem, which means there is a high difference between the training and test errors (GOODFELLOW; BENGIO; COURVILLE, 2016).

Another known challenge in ML is the bias-variance trade-off. The term “variance” describes the degree of change observed in the prediction function if it were to be estimated using a different training dataset. In contrast, the term “bias” describes the degree of error introduced when a highly complex real-life problem is approximated by a simpler model (JAMES *et al.*, 2014). The model’s objective is to achieve a low bias and low variance. However, in practice, reducing one inevitably leads to an increase in the other, resulting in a trade-off.

Several MLs have been developed over time, in special, there are Decision Tree and Random Forests (QUINLAN, 1986; KOTSIANTIS, 2013; BREIMAN, 2001), KNN (JAMES *et al.*, 2014), Gradient Boosting (FRIEDMAN, 2001; NATEKIN; KNOLL, 2013) and others. This work focuses on artificial neural networks, which will be explained further in the next section.

2.2.1 Artificial Neural Networks

To emulate the behavior of the human brain, ANN was developed with the understanding that cognitive processes are built by the interactions of neurons and synapses. The first simplified neuron model was created in 1943 by (MCCULLOCH; PITTS, 1943). The model of neurons most used nowadays, called “Perceptron”, is presented in Figure 6. This model has multiple inputs, with each input having a designated weight. The input value x_i is multiplied by the corresponding weight w_{ki}. Subsequently, the total sum of all these values is calculated, along with the addition of other values designated as bias. The final result of the sum is fed into an activation function, which generates an output. A typical implementation of a ANN is in the form of a MLP, which is a fully connected neural network, presented in (MURTAGH, 1991).

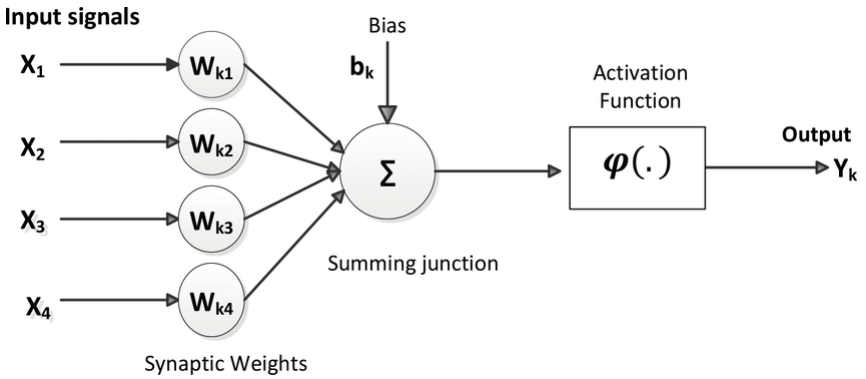


Figure 6 – A model of the neuron. Several inputs and their respective weight. The sum of all the multiplied input by the respective weight is added with a bias value, and the result feds an activation function. Source: (RUMELHART; HINTON; WILLIAMS, 1986).

A set of these neurons, disposed in layers and with each layer connected to the others, is called MLP. The term *feedforward* describes the process of providing the outputs of a given layer of neurons as the input for the subsequent layer. Furthermore, a potential approach for determining the optimal weights and biases is through the *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986) algorithm. A significant parameter of this algorithm is the *learning rate*, which determines the rate at which the network can modify the weights and biases.

The activation function obtains the node’s output and introduces non-linearity into the neural network. There are numerous activation functions, and selecting the most appropriate one depends on the context in which it is applied. The most commonly used activation function are *ReLu*, described in equation 1,

$$f(x) = max(0, x) \tag{1}$$

the *Sigmoid* activation function, described in equation 2,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

and *Tanh* (hyperbolic tangent) described in equation 3,

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

where x is the input of the function.

CNN is an approach specialized in processing data in a grid-like topology, such as images (GOODFELLOW; BENGIO; COURVILLE, 2016). The working of this kind of Neural Network is based on the utilization of filters to reduce the volume of data and the identification of patterns through the search process, which enables the retrieval of pertinent information. In essence, it can be stated that the CNNs are neural networks that employ the convolution operation instead of the conventional matrix multiplication in at least one of the layers (LECUN; BENGIO; HINTON, 2015).

Characterized by a high degree of connectivity, CNNs have architectures comprising multiple layers, which compactly represent high nonlinear functions. These layers often include convolutional, pooling, and fully connected layers (ESFAHLANI *et al.*, 2022).

The CNN technique can be broken down into two main components: feature extraction and classification. The first encompasses the convolutional layers, which are responsible for applying filters, and the latter includes the pooling layers, which contribute to reducing the representation size, improving the computation speed, and promoting the features to be more robust. An example of a well-known CNN is the VGG16, presented by (SIMONYAN; ZISSERMAN, 2015), built to work in the ImageNet[7], which is a general image classification challenge.

The high computational cost of training a machine learning model and the limitation of algorithms to specific tasks have prompted the development of a new approach called transfer learning. The objective is to transfer the knowledge acquired in one or more source tasks and use it to enhance the learning process in a related target task (TORREY; SHAVLIK, 2010). Several models[8] had already been pre-trained and can be used on Transfer Learning for other tasks, such as the GoogLeNet (SZEGEDY *et al.*, 2015), used on (KENDALL; GRIMES; CIPOLLA, 2015), and VGG16.

2.3 Machine Learning Optimization

In ML, there are two main types of parameters: model parameters and hyperparameters. Model parameters can be updated through data learning models, such as neurons' weights in

[7] Details in https://www.image-net.org/. Accessed on December 10, 2023.

[8] Some examples are available at https://keras.io/api/applications/. Accessed on October 27, 2024.

ANN. In contrast, hyperparameters are set before training the ML model and define the architecture of the model (KUHN, 2013). The ML optimization can be understood as the optimization of the training (SUN *et al.*, 2020), where the objective is to optimize the values of parameters of the ANN to obtain the best corresponding values to express the training dataset. However, the optimization also can be related to the optimization of the structure of the ML model, i.e., the Hyperparameter Optimization (HPO).

The objective of the HPO is to enhance the efficacy of machine learning algorithms by adapting them to the specific characteristics of the problem at hand. Despite its importance, the HPO faces several challenges, such as extremely expensive model evaluation and the high complexity of the model configuration space (FEURER; HUTTER, 2019).

Several approaches have been developed aiming to address the HPO in ML models (BISCHL *et al.*, 2023; FEURER; HUTTER, 2019; YANG; SHAMI, 2020). One of the most used ways to optimize the ML models is considering them as black boxes (wherein the internal behavior remains unknown despite the ability to provide an output given an input). The most basic HPO method is the Grid-Search[9] (MONTGOMERY, 2017). Based on a finite set of values for each hyperparameter to be optimized, the approach evaluates all possible combinations between the defined set, i.e., the Cartesian product of the sets. Another well-known approach is the random search, which randomly applies sample configurations until a certain budget for the search is exhausted (FEURER; HUTTER, 2019; BERGSTRA; BENGIO, 2012).

Several population-based methods are also used for the HPO, which maintains a set of configurations and improves the populations through an evolutionary process (using operators such as mutations - which are local perturbations - and crossover - combinations of different members) aiming to obtain a new generation with better configurations (FEURER; HUTTER, 2019). Some examples of these algorithms are genetic algorithms, evolutionary algorithms, evolutionary strategies, and particle swarm optimization (EBERHART; SHI, 1998; SIMON, 2013).

2.3.1 Bayesian Optimization

Bayesian Optimization was initially presented by (MOCKUS, 1974) and is part of the state-of-the-art framework for global optimization. Bayesian Optimization is also referred to as Sequential Model-Based Optimization (SMBO) (BERGSTRA *et al.*, 2011). This approach is designed to identify the maximum (or minimum) of a function (also referred to as optimization) in a more expeditious way relative to alternative approaches, such as the grid search, which evaluates all potential function values and compares all resulting outcomes. In this way, Bayesian Optimization is a good candidate for functions with costly evaluation, offering the additional benefit of accommodating black-box functions.

[9] A well-known implementation is available in: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.\GridSearchCV.html. Accessed on August 30, 2024.

To optimize a function $f(x)$, defining a surrogate model that expresses the assumptions about the function in question is necessary. This model must be created within a defined search space, representing the universe of possible inputs for the function. In essence, it is a probability representation of the objective function, a model trained on the pairs (hyperparameter, score from the evaluation function). The most common surrogate model is the Gaussian process due to its flexibility and traceability. The surrogate model, designated as $m(X, y)$, can be constructed using a subset of the $f(x)$ samples, incorporating a range of x values. In this context, the variable X represents the input values, x, while the variable y represents the corresponding function values, $f(x)$.

The next phase involves selecting the next x to be evaluated, given that some have already been evaluated in constructing the surrogate model. This is achieved using an acquisition function designated $a(x)$. To ascertain the probable variation in the function $f(x)$ across a range of x values, the acquisition function utilizes the information provided by $m(X, y)$. The value of x that corresponds to the minimum or maximum predicted value in $m(X, y)$, as determined by the objective of the optimization, is recommended as the next value of x to be assessed by $f(x)$.

Once the recommended value of x has been evaluated by $f(x)$, this value is incorporated as a new data point into the surrogate model. With the repetition of this process, the surrogate model improves. The process is terminated when the maximum number of interactions has been reached. An illustrative example of the mentioned process can be found in Algorithm 1, based on (BERGSTRA *et al.*, 2011). The optimization of the hyperparameters of a black-box function, represented by $f(x)$, is considered in the algorithm mentioned earlier. The variables H, T, f, M, a, and $x*$ have the following meanings: H is the observation history of the pair (hyperparameter, score), T is the maximum number of iterations, f is the true objective function, M is the surrogate function, a is the acquisition function, and $x*$ is the next chosen hyperparameter to evaluate. Further details about Bayesian Optimization can be found in (SNOEK; LAROCHELLE; ADAMS, 2012).

Algorithm 1 – Pseudo-code of a generic Bayesian Optimization.

SMBO (f, M_0, T, a)
$H \leftarrow \emptyset$
$t = 1$
while $t \leq T$ **do**
    $x* \leftarrow argmin_x a(x, M_{t-1})$
    $Evaluate\ f(x*)$ {Expensive step}
    $H \leftarrow H \cup (x*, f(x*))$
    $Fit\ a\ new\ model\ M_t\ to\ H$
    $t = t + 1$
end while
Return H

Source: Based on (BERGSTRA *et al.*, 2011).

In summary, this approach employs the principles of Bayes' theorem to develop a function model based on prior evaluations. Subsequently, the model is used to ascertain which search

space region should be investigated, maintaining an equilibrium between areas where the most favorable outcomes are anticipated and relatively unexplored. In this manner, the approach endeavors to ascertain the optimal result within the minimal number of evaluations of the function under optimization. Tutorials about the Bayesian Optimization are provided by (BROCHU; CORA; FREITAS, 2010; SNOEK; LAROCHELLE; ADAMS, 2012).

Bayesian Optimization has been used to optimize machine learning models in some studies, including CNN and Random Forests, gathering interesting results in terms of time and accuracy (WU *et al.*, 2019).

3 PREVIOUS WORK

This chapter presents a review of the work performed by the author in (KLEIN, 2023), which was developed as the master thesis in Master's in Informatics at IPB. It will present the proposed approaches and their specifications, as well as the results and the conclusions. It is fundamental to understand the main ideas and some details of the previous work due to the dependency of the current study on the previous one.

3.1 Motivation and Goals

This previous work, performed in (KLEIN, 2023), aimed to study approaches addressing robot localization within the context of RaF competition. The objective was to utilize images captured by an onboard camera, process these images, and then employ the data in AI-based approaches to estimate the robot's pose. Based on the results of this work, three papers were published: (KLEIN *et al.*, 2023b; KLEIN *et al.*, 2023a; KLEIN *et al.*, 2024a).

The significance of this study lies in the fact that the methodologies presented do not necessitate explicit knowledge of ArUco's pose. This represents a vital advantage of the strategies explored in the study. One illustrative example where the benefit is particularly pertinent is in situations where obtaining the precise pose of the markers is challenging or impossible, such as in hostile environments. To illustrate, the training images may be gathered by a robot aware of its location by utilizing an alternative localization system, such as a Differential GPS (DGPS), to produce the requisite dataset for the model's training. Subsequently, during the localization process, the DGPS is no longer necessary, and precise localization may be achieved through a camera alone.

Three main questions had directed the work (KLEIN, 2023):

1. Can machine learning approaches effectively estimate the position and orientation of a robot solely based on camera images in a structured environment with the presence of fiducial markers?

2. Among different types of machine learning approaches, such as MLP, Random Forest (RF), Gradient Boosting (GB), K-Nearest Neighbor Regression (KNR), and Convolutional Neural Networks (CNN), which approach leads to the smallest pose error in estimating the robot's position and orientation based solely on camera images in a structured environment with fiducial markers?

3. Considering the constraints of embedded devices with limited memory and computing power, which type of machine learning approach is better suited for achieving accurate pose estimation in a structured environment with fiducial markers?

The following sections will present an overview of the methodology applied in the work, explaining the data collection, the proposed approaches, the studies realized, and the tests, along with a brief review of the conclusions and discussions.

3.2 Methodology

In (KLEIN, 2023; KLEIN *et al.*, 2023b), three concept solutions were designed to explore the use of ML in localization in the RaF competition, and a visual demonstration of the flow of each concept is presented in Figure 7:

1. **Concept 1**: Based on the image from the robot's camera, identify all the ArUco markers on it, getting their relative poses (*rvec* and *tvec*) and their ID. Then, all the data was aggregated into a matrix comprising 49 rows and seven columns, where each row represents a distinct ArUco (ArUco with ID 0 is represented in line 1, the second row represents the ArUco with ID 1, and so on). The columns contain six elements from the *rvec* and *tvec* arrays, and the 7th element is a Boolean value indicating if that particular ArUco was identified or not. Finally, this same matrix is the input for three different prediction models, where each model is responsible for one pose component (x, y, or θ), and the output is one pose.

2. **Concept 2**: Based on the image from the robot's camera, identify all the ArUco markers on it, getting their relative poses (*rvec* and *tvec*) and their ID. These values for each ArUco are directly sent to the three ML models (one relative to each pose component), resulting in one pose estimation per ArUco (i.e., each image can have different pose estimations).

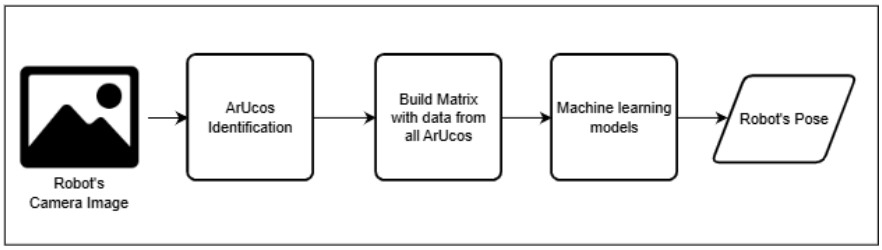
3. **Concept 3**: An image taken by the robot camera is pre-processed, and it is used as input for three CNN models, one for each component, which returns, in aggregation, the pose.

To evaluate the quality of each model, several metrics have been used, such as Mean Absolute Error (MAE), RMSE, and R square (R^2).

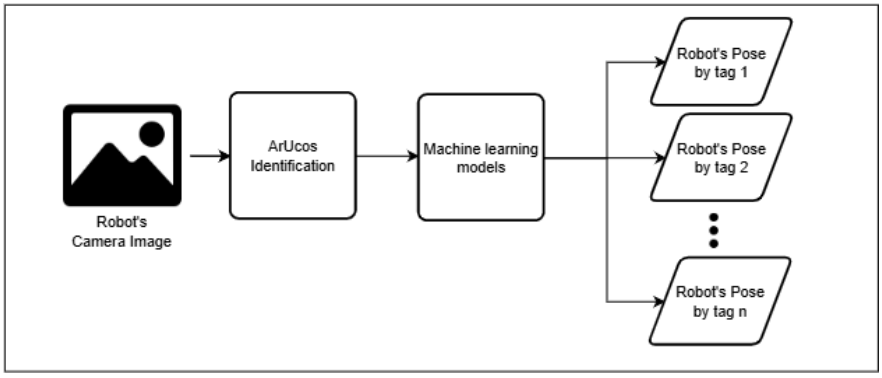
3.2.1 Data Collection

Five (5) data collections were performed in the simulator, following a similar approach. The first collection considered the whole field, which was discretized in a grid, with each square being 1 cm in size. The robot was positioned on the grid in the center of the available positions, that is to say, without obstacles. In each position, the robot takes around 60 images while performing a 360º turn to create a database (due to delays in the simulator, the number of collected

Concept 1: Using Machine Learning to Generate one Pose Estimation per Image with tags



Concept 2: Using Machine Learning to Generate one Pose Estimation per Tag Identified



Concept 3: Using CNN to Generate one Pose Estimation per Image without Tags Identification

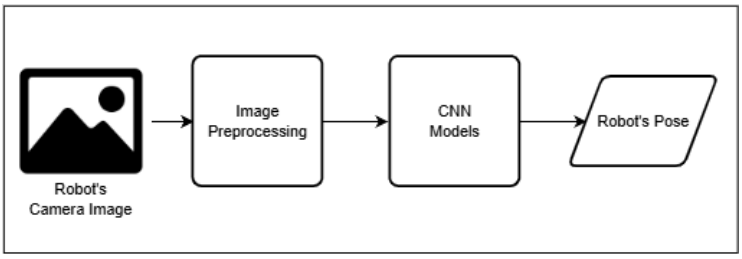


Figure 7 – Concept solutions. Based on (KLEIN, 2023).

images in each turn can vary slightly), with the total number of images collected across all positions reaching approximately 350,000. The other 4 data collections follow the same procedure but only consider a limited part of the field, a square of 10 x 10 cm in the center of the field, as presented in Figure 8. This squared was discretized in grids with 10.0 mm, 5.0 mm, 2.5 mm, and 1.0 mm, as presented in Figure 9. Subsequently, additional data collections were conducted along a randomly selected route within the field. Figure 10 presents six examples of images collected in different parts of the field.

3.2.2 Datasets Creation

Before creating the datasets to be used throughout the project, two main steps were necessary: (1) Remove the duplicated images, which happened due to delays in the simulator; (2) remove images that do not contain at least one ArUco. Figure 11 presents an example of

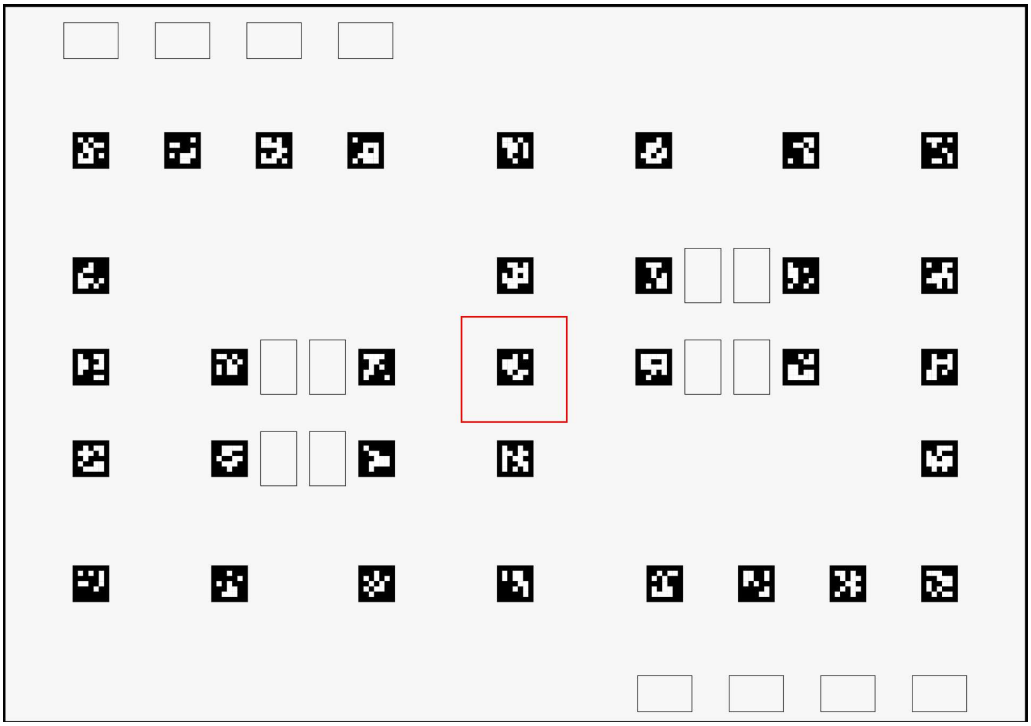


Figure 8 – Part of the field used to collect images. Source: (KLEIN *et al.*, 2023b).

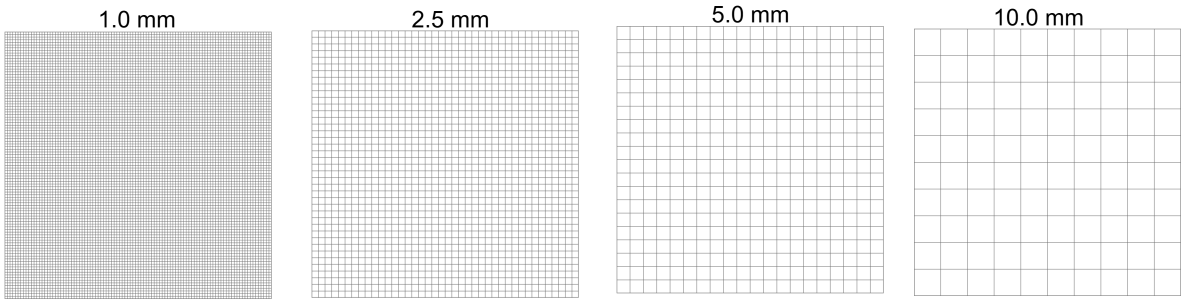


Figure 9 – Exemplification of the grids used, considering the four different grid sizes. Based on (KLEIN *et al.*, 2024b).

this kind of image, and the objective of their exclusion is to avoid the use of ambiguous images, which can be part of any part of the field and lack any clear means of identification.

The images were processed using the OpenCV library to create the datasets to identify the ArUcos and estimate each ArUco's pose relative to the camera's reference frame. The OpenCV library returns arrays for each marker, including the position and orientation of the marker to the camera. As previously mentioned, these arrays are called *tvec* and *rvec*, respectively. The following datasets were created:

1. **Dataset A** - *Considering the whole field and combining observations' attributes in a matrix format*: The images used were collected across the entire field, considering the grid resolution equal to 1 cm. Each observation represents a single image containing one feature and three targets. The feature is comprised of a matrix with 49 rows, each representing an ArUco marker, and seven columns that represent the *tvec* and *rvec*

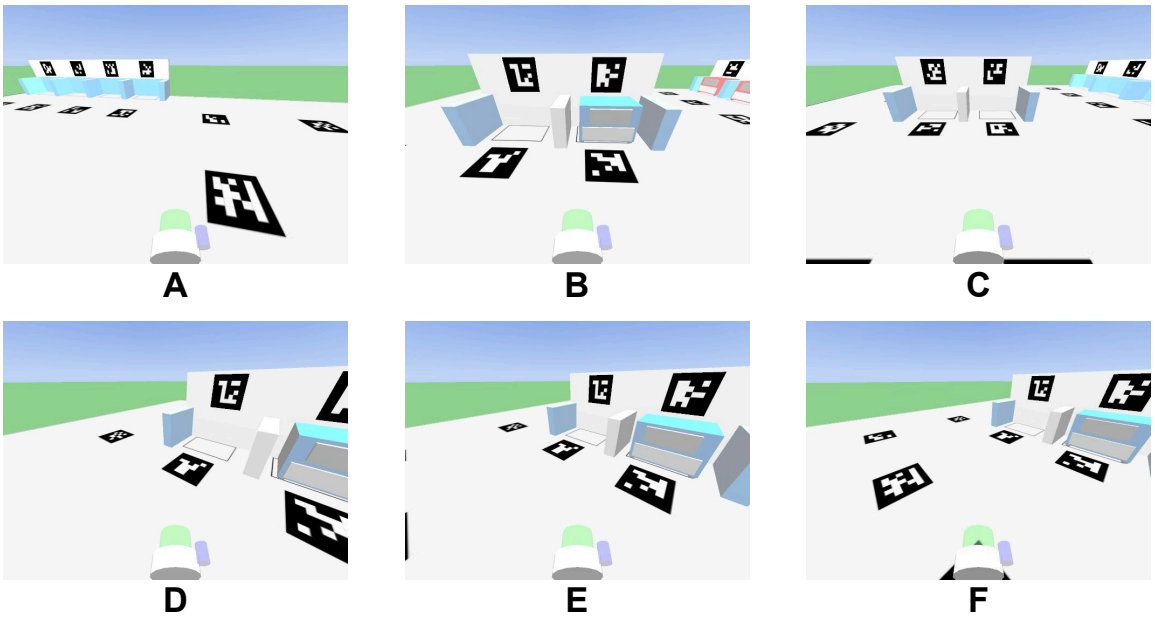


Figure 10 – Example of images collected.

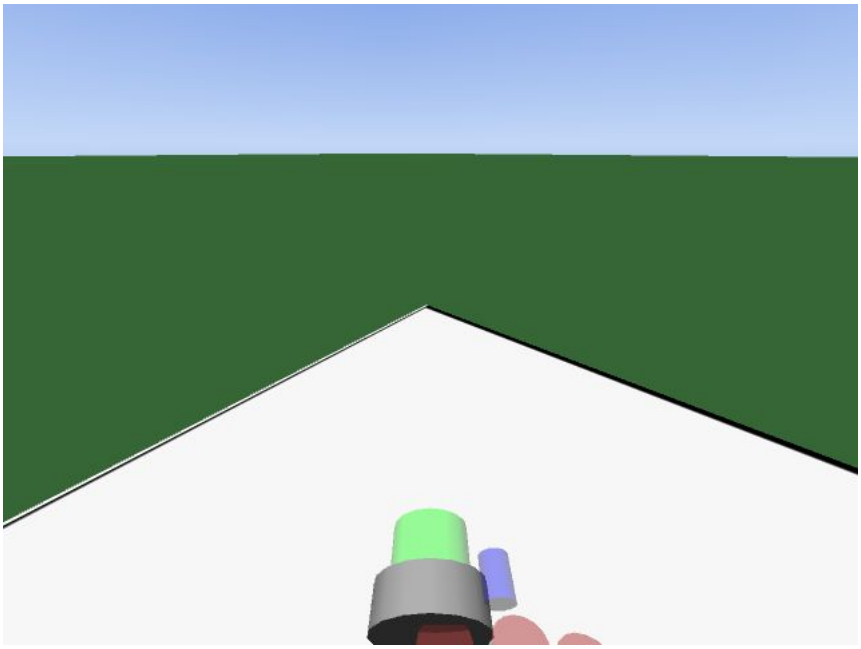


Figure 11 – Example of ambiguous image. Source: (KLEIN, 2023).

arrays, along with a boolean value indicating whether the marker was detected in the image. If a particular ArUco marker is absent from an image, the corresponding row in the matrix is filled with 0. The target variables are *x, y*, and θ.

2. **Datasets B** - *Considering the collection in part of the field with different grid resolutions and combining observations' attributes in a matrix format*: This dataset is composed of four datasets. To generate each of them, images taken in the center of the field (Figure 8) were used, varying the grid resolutions, where each dataset was composed of images from different resolutions: **B1** for 10.0 mm, **B2** for 5.0 mm, **B3** for 2.5 mm, and **B4** for 1.0 mm. All the datasets were produced using the same procedure as Dataset A.

3. **Dataset C** - *Considering the whole field in an ArUco's array*: Images taken across the entire field were used, with a grid resolution of 1 cm. However, unlike datasets A and B, each observation in this dataset represents a detected ArUco marker rather than an entire image. As such, each observation consists of seven features: the marker's ID, *rvecs*, and *tvecs*, as well as three targets: *x, y*, and θ. In this dataset, the image information is no longer relevant, as the focus is solely on the relative pose of the ArUco markers.

4. **Dataset D** - *Considering the collection in part of the field with different grid resolutions and combining the observations' attributes in ArUco's array*: Dataset D is composed of four datasets, as Dataset B. To generate these datasets, images taken at the center of the field (Figure 8) were used, varying grid resolutions. Each dataset was composed of images from different resolutions: **D1** for 10.0 mm, **D2** for 5.0 mm, **D3** for 2.5 mm, and **D4** for 1.0 mm. Each dataset was produced using the same procedure as Dataset C.

5. **Dataset E** - *Considering a random route in an ArUco's array*: This dataset was created using images captured across the entire field, using a random path. The exact process as Dataset C was used to generate this dataset.

3.2.3 Implementation

The implementation of the previous work was based on three main steps, which are discussed briefly below:

1. **Part 1: Feasibility of the ML in Embedded Systems**: To validate the feasibility of the use of Machine Learning models implemented in the localization issue at RaF in an embedded system, a Raspberry Pi 4 Model B4 was used, considering the dataset D1, with the ML techniques: MLP, SVM, and Random Forest, as also explored from others works in the same context of validation of the use ML in systems without a high computational power (YAZICI; BASURRA; GABER, 2018). The evaluation metrics

included training and execution times, energy consumption (in mWh), and model size. An Atorch USB tester was used to measure energy consumption. The complete details regarding the feasibility work are available in (KLEIN *et al.*, 2023a; KLEIN, 2023).

2. **Part 2: Quality of ML models**: The initial examination in this part involves contrasting the various methods based on Concept 1, using dataset A. The dataset was partitioned into two sections: 85% for training and 15% for validation. Figure 12 shows a visual depiction of this procedure. Then, the algorithm that presented the best results in the previous analysis was trained with datasets B1, B2, B3, and B4 with corresponding resolutions of 10.0 mm, 5.0 mm, 2.5 mm, and 1.0 mm, and an analysis of its outcomes was conducted. The concluding assessment in this study entailed comparing analytical and ML techniques. The algorithm that yielded the best results in the first analysis was now trained on dataset C and evaluated on dataset E, using Concept 2, and finally compared with the analytical technique. This comparison is significant because the analytical approach was introduced in (BRAUN *et al.*, 2022a) and provides a good benchmark for pose estimation in the RaF scenario. A further explanation of the study is available in (KLEIN, 2023; KLEIN *et al.*, 2023b).

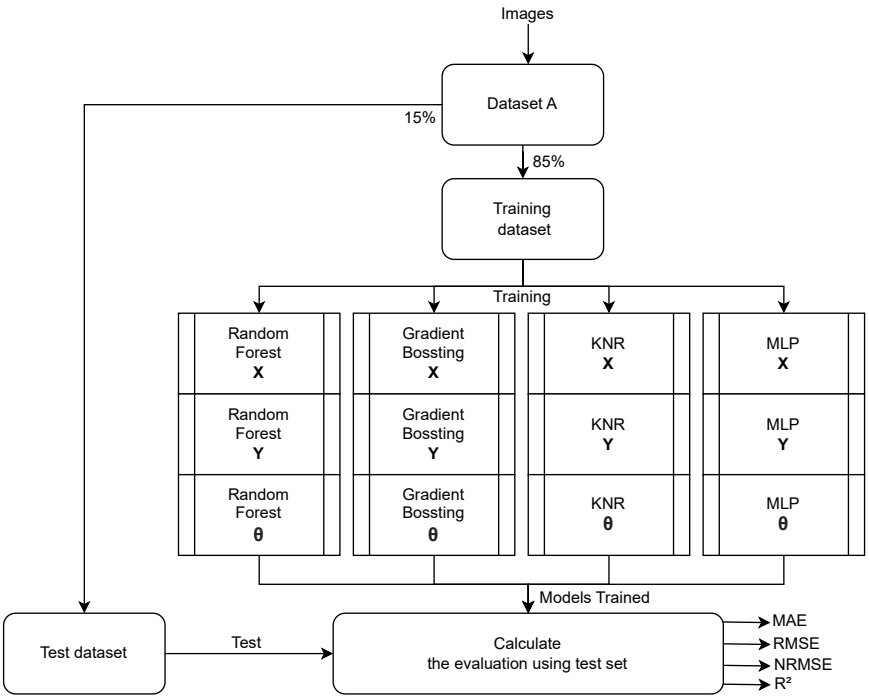


Figure 12 – Flow process. Source: (KLEIN *et al.*, 2023b).

In addition, to verify Concept 3, a CNN model based on the VGG16 transfer learning model was applied. Furthermore, an output layer was defined as a linear activation function with 1 or 2 outputs (with one output for the model to estimate θ and two outputs to estimate x and y). Images from datasets A, B1, B2, B3, and B4 were used (but without preprocessing, besides the one required by the VGG16 model). Further details

are available in (KLEIN, 2023; KLEIN *et al.*, 2024a). An overview of the CNN structure is presented in Figure 13.

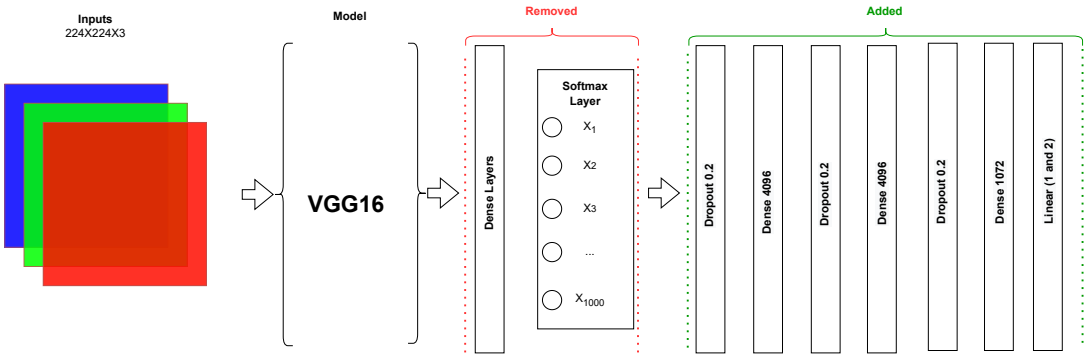


Figure 13 – Architecture the proposed CNN model. Source: (KLEIN, 2023).

3. **Part 3: Execution in the Real Scenario**

The third part of the first study was the application in the real scenario. The RaF simulator was adapted to the robot's real parameters (such as camera configurations, field of view, camera position, etc.). The real values in the robot were trying to be mimicked in the simulator, with all adaptations being done empirically. The data was recollected in the simulator, and an ML model (MLP) was trained as in solution Concept 2. The robot software was then adapted to use the MLP, compared to the ground truth, with the robot following a predetermined route. Figure 14 presents the robot, the real field, and the ground truth system used in the real scenario. More details are available in (KLEIN, 2023).

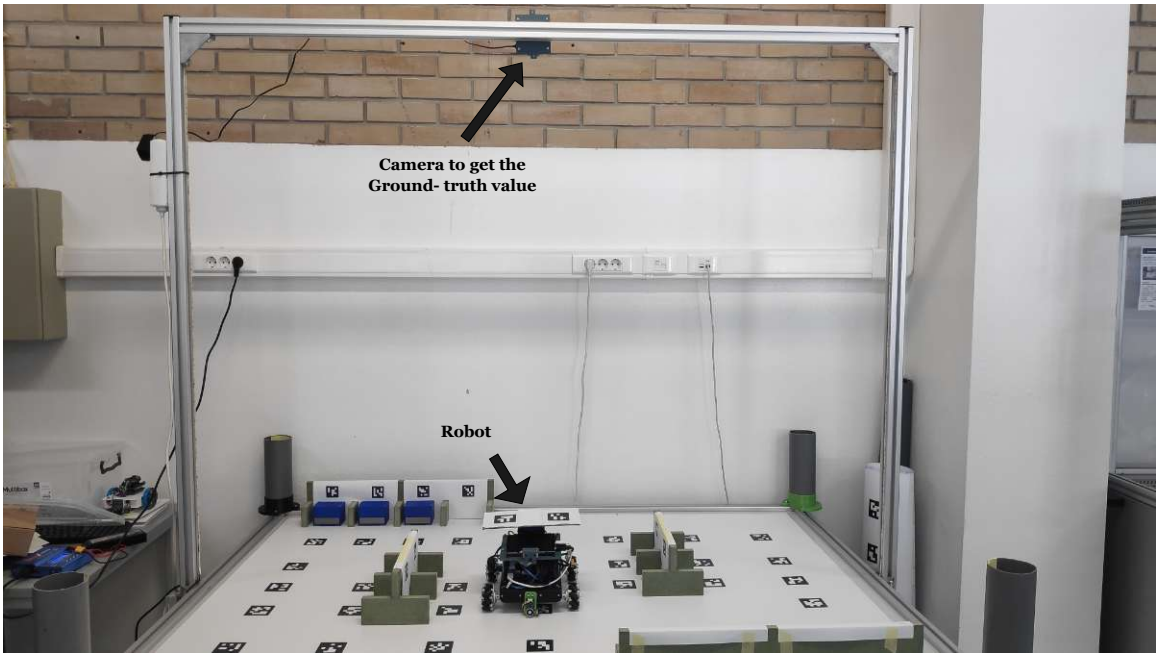


Figure 14 – Real system architecture. Source: (KLEIN, 2023).

3.3 Results and Discussions

The previous study was divided into three main parts, and the results of each will be briefly presented and discussed.

3.4 Part 1: Feasibility of ML in embedded systems

This part evaluates the feasibility of using ML in an embedded system, considering factors such as energy consumption, model size, and times to train and inference. Three ML techniques were evaluated: RF, MLP, and SVM, using the D1 dataset. The data were divided into 85% used for training and 15% for testing. The preprocessing of the training data took 157.49 seconds and consumed 180 mWh, while the preprocessing of the testing data took 28.62 seconds and 35 mWh. Figure 15 presents graphs with the comparison between models in the training and inference time in terms of speed (left) and energy consumption (right), with each color representing a different model.

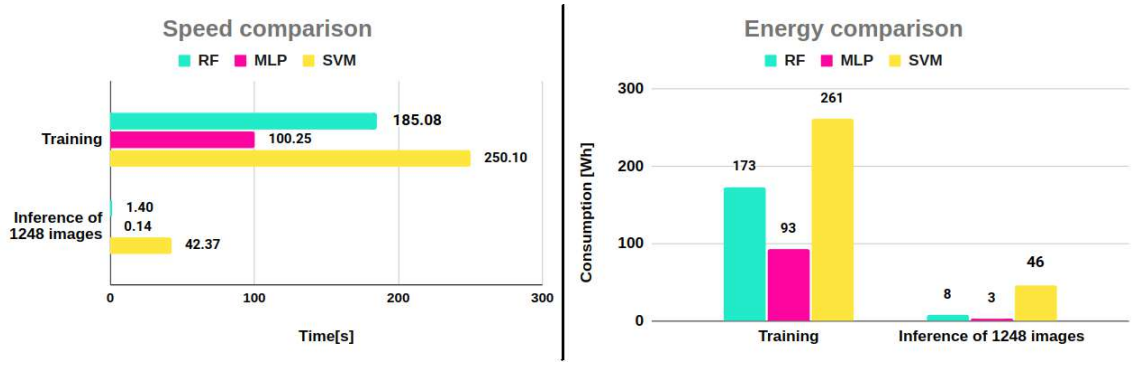


Figure 15 – Time spent (left) and energy consumption (right) by three ML methods in training and testing. These statistics do not account for the time required to preprocess the data. Source: (KLEIN, 2023).

The analysis reveals that the SVM algorithm took longer to train and execute than the other algorithms. On the other hand, the MLP algorithm was the fastest in training and execution. The energy consumption pattern follows a similar trend, with SVM being the most expensive and MLP being the most efficient. In terms of execution time, all approaches could respond to each image in no more than a few milliseconds.

Regarding model quality, the estimation errors were found to be on a centimeter scale, with RF emerging as the most effective approach. The SVM approach yielded models for the x and y targets that were notably compact, occupying a mere 1 kB, whereas the model size for θ was considerably larger, at 2.7 MB. In contrast, the three MLP models exhibited consistently small sizes, ranging between 26 and 30 kB. Notably, the RF approach resulted in the generation of models with the largest size, approximately four orders of magnitude greater than the other models, with each model requiring more than 100 MB. The analysis of the model size indicates that this is not a significant issue in the current context. However, it is essential to highlight the

stark contrast between the models, which underscores the necessity for more data (images) if the entire field is to be considered. This could lead to a notable increase in model size, which may become a challenge.

An additional experiment was conducted on a computer using the same algorithms but with the larger Dataset C. The results demonstrated that the MLP models remained relatively compact, with each model occupying a maximum of 1 MB. In contrast, RF models required between 3 and 5 GB, representing a significant difference in computational power. The computational resources needed to store and execute models of this magnitude exceed the capacity of the Raspberry Pi. Ultimately, the SVM models could not be trained due to the required time. Thus, MLP represents the optimal option for embedding an ML model in the localization system, given a trade-off between required memory, computing power, and energy consumption. While other approaches, such as RF, yield more promising results, they are not as well-suited for this particular application.

3.5 Part 2: Quality of ML models

The results of part 2 are divided in two: Section 3.5.1 presents the results for the concept solutions 1 and 2 and a comparison between ML techniques, while Section 3.5.2 presents the results for the concept solution 3, using CNN techniques.

3.5.1 Approach 1: ML techniques using fiducial markers

This study first investigated models' quality using approaches 1 and 2. Several algorithms were examined by performing on dataset A, and the most effective algorithm was identified. Subsequently, this algorithm was trained and tested on datasets B1, B2, B3, and B4 to evaluate the trade-off between the grid resolution and the accuracy of the estimates. Then, the proposed approach in Section 3.2.2 was compared to analytical (presented in (BRAUN *et al.*, 2022a)) and ML methods.

Figure 16 presents a comparison between the techniques, where the left plot illustrates the MAE error for the axes x (blue) and y (red) in meters, and the right plot shows the MAE error for θ in degrees. Analyzing this graph, it is possible to note that Random Forest was the best approach in the three axes, followed by the MLP (the second in the position and the third in the orientation). All the tested approaches were better than the baseline approach (which was the most straightforward approach).

Then, using the best approach previously found, RF, the model was trained and evaluated in dataset B. The result is presented in Figure 17. It is possible to notice a trade-off in the position: With the increase of the resolution (i.e., decrease in the grid size), the error in the estimations decreases. However, this behavior does not happen in the orientation, which keeps a similar

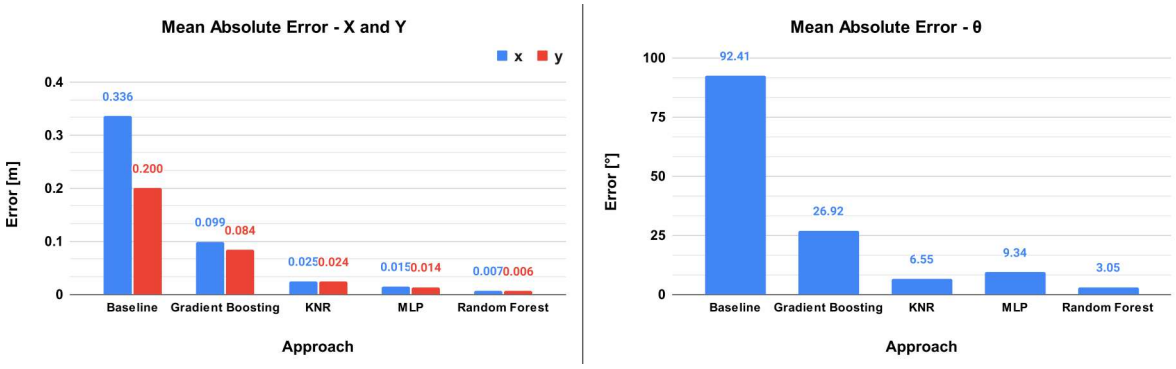


Figure 16 – Graph on the left displays the MAE for the x and y axes in meters, and the image on the right displays the MAE for the θ in degrees. Source: (KLEIN *et al.*, 2023b)

.

result for all the same grids. This is expected since the number of images increased in the position, with more images in other field positions, while the quantity taken by position in each robot loop was kept the same.

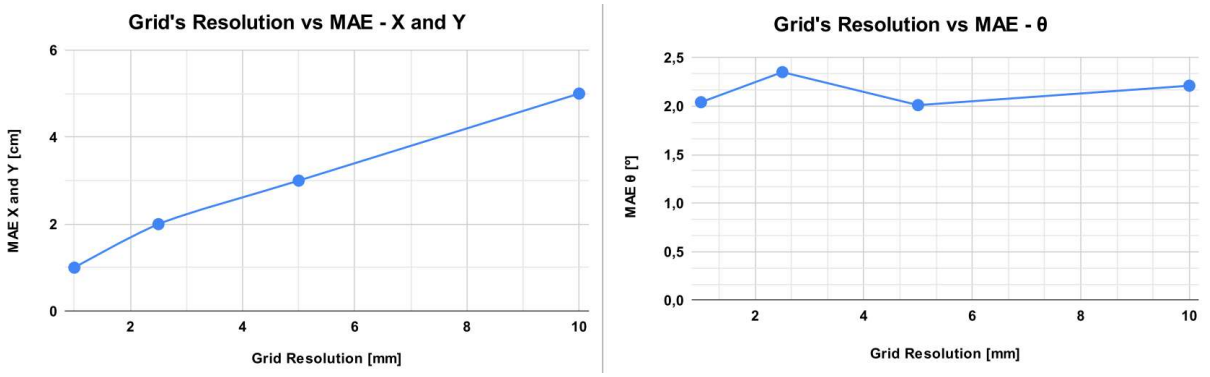


Figure 17 – Comparison of the error obtained against the decrease of the grid's resolution. The graph on the left displays the MAE for the x and y axes in centimeters (the same curve is for both values), while the image on the right displays the MAE for the θ in degrees. Source: (KLEIN *et al.*, 2023b).

Then, when comparing the results of approach 2, using RF with the analytical approach, the results presented an increase of the quality of 6 and 8 mm for x and y, respectively, and a decrease of 1.27° in θ. These results showed the possibility of using the approach as a possible solution to the problem. The complete study is available in (KLEIN *et al.*, 2023b) and (KLEIN, 2023).

3.5.2 Approach 2: CNN technique

Table 1 presents the results for the tests performed in a limited part of the environment with different grid resolutions. The columns represent the resolutions and the lines of the metrics for each pose component. All the values (except for the training time) are averages and the respective standard deviation of the three executions.

It is interesting to notice the quality of the position estimations was on a millimeter scale, and the MAE in θ was consistently lower than 5°. Also, another interesting topic to notice in this approach is the trade-off between the grid's resolution and quality in the estimations (previously presented in Section 3.5.1) does not exist here. In addition, all the results displayed low standard deviation, reflecting the liability and satisfactory performance of the models in the cross-validation process. Another important outcome of this approach is that each model is 512MB, up to 1024MB for the entire pose estimation. This is important because specific ML techniques, such as Random Forest, can significantly increase the size (KLEIN *et al.*, 2023a), as the structure of the model can change, i.e., the depth of trees, depending on the problem in the context. This study, available at (KLEIN *et al.*, 2024a), has received the Best Paper Award at OL2A 2023 (International Conference on Optimization, Learning Algorithms, and Applications).

Table 1 – Results obtained considering the limited part of the field, using different grid's resolution, with *Avg.* columns indicating the average and *Std. Dev.* indicating the standard deviation. Boldface values are the best in each metric. Source: (KLEIN *et al.*, 2024a).

Grid's Resolution		**10 mm**		**5 mm**		**2.5 mm**		**1 mm**	
Quantity of images		8306		33,291		113,596		655,130	
		Avg	**Std. Dev.**	**Avg**	**Std. Dev.**	**Avg**	**Std. Dev.**	**Avg**	**Std. Dev.**
MAE	**x[m]**	0.0026	0.0001	0.0026	0.0004	**0.0023**	**0.0002**	0.0023	0.0006
	y[m]	0.0026	0.0000	0.0027	0.0004	**0.0023**	**0.0004**	0.0023	0.0007
	θ**[°]**	2.97	0.29	2.76	0.14	**1.58**	**0.10**	4.51	3.82
RMSE	**x[m]**	0.0034	0.0002	0.0033	0.0005	**0.0029**	**0.0002**	0.0029	0.0006
	y[m]	0.0038	0.0006	0.0042	0.0002	0.0032	0.0003	**0.0030**	**0.0007**
	θ**[°]**	6.03	1.50	7.09	0.76	**4.06**	**0.71**	6.99	3.80
NRMSE	**x**	0.0307	0.0015	0.0302	0.0043	0.0293	0.0026	**0.0262**	**0.0059**
	y	0.0345	0.0053	0.0368	0.0021	0.0292	0.0031	**0.0279**	**0.0062**
	θ	0.02	0.00	0.02	0.00	**0.01**	**0.00**	0.02	0.01
R2	**x**	0.990	0.001	0.990	0.003	0.990	0.002	**0.989**	**0.005**
	y	0.985	0.005	**0.984**	**0.002**	0.990	0.002	0.990	0.004
	θ	0.996	0.002	0.995	0.001	0.998	0.001	**0.994**	**0.006**
Training time (Position) [s]		**678.63**	**80.85**	898.40	153.75	4505.48	647.51	8281.08	233.89
Training time (Orientation) [s]		**592.89**	**80.17**	1437.00	401.92	17369.99	3399.39	18509.41	8368.63
Inference time (Position) [ms]		2.95	0.74	**2.02**	**0.07**	2.44	0.35	2.06	0.15
Inference time (Orientation) [ms]		2.46	0.12	2.04	0.05	2.22	0.19	**1.96**	**0.40**

3.6 Part 3: Implementation in the Real Scenario

This part of the study explored applying a simulation-trained model to a real robot. To perform it, the simulator was adapted to resemble the real robot. The values of the robot parameters were adjusted empirically using the same parameters for the two cameras (real and simulated). The data were then recollected, and the models retrained.

Based on the previous results, the MLP technique was used as the ML model, considering approaches 1 and 2. Tables 2 and 3 present the concept solutions 1 and 2 results, respectively. The first one consists only of the direct results with the ground truth. In contrast, the

second compares with the analytical approach (used in the competition (BRAUN *et al.*, 2022a)). Also, some models with different structures, called in the previous work as optimized, were tested and showed a margin to improve the quality of the models. A video containing all executions of the analytical approach and MLP approaches (considering the concept solutions 1 and 2) was recorded[1].

Table 2 – Results of the errors obtained in the real scenario, showing the error in the estimations with the MLP models based on concept solution 1. Based on (KLEIN, 2023).

	MLP Original		
	x[m]	**y[m]**	**θ[°]**
MAE	0.047	0.060	10.73
RMSE	0.077	0.082	17.68

Table 3 – Results of differences obtained in the real scenario, considering the estimations with the MLP with the analytical method, considering the concept solution 2. Based on (KLEIN, 2023).

	MLP			**Analytical**		
	x[m]	**y[m]**	**θ[°]**	**x[m]**	**y[m]**	**θ[°]**
MAE	0.166	0.054	17.94	0.037	0.031	8.36
RMSE	0.184	0.064	34.45	0.060	0.044	20.62

Analyzing the results of both tables, it is possible to notice that the behavior of the model in the simulation was amplified in the real world (i.e., all the errors obtained by the simulator models were smaller than those obtained in the real scenario by the same models), and slight differences, such as camera parameters, can be drastically influenced by the results.

In addition, a significant difference can be observed between the MLP results and the analytical model. When analyzing the robot's behavior using the models in the recorded video, they showed that they were not ready to be implemented entirely in the robot and used in the competition. However, the proposals were promising since, even with all empirical adjustments in the simulator to represent the real scenario, the robot work is reasonably plausible considering the y axis and the θ, even with the behavior in x being the worst. On the other side, it emphasizes that the small divergences between the simulator and the real will result in high differences. More detailed information is available in (KLEIN, 2023).

3.7 Conclusions

The previous work aimed to study the localization issue of the problem of RobotAtFactory4.0 using machine learning approaches. Three concept solutions were developed in this work: one that estimates one pose per image frame using the fiducial markers, another that estimates one pose per ArUco identified in an image frame, and another that estimates one pose per image without using ArUcos.

[1] Video of the comparison: https://www.youtube.com/watch?v=-5ZmVpJobg0

The three initial questions were answered based on the results: The first research question (1) can be answered that the ML approaches can solve the localization topic, with some approaches presenting errors in millimetric scale and less than 10° in orientation. The second question (2) can be answered, based on the results, that RF and CNN were the approaches that presented the best results, with response time in milliseconds and errors in millimeter scale. The third question (3) can be answered based on the results that MLP and CNN are the best approaches due to the controlled size of the modes and the response time in milliseconds.

The work was successful, and the three research questions were answered. The ML approaches were validated in feasibility, and many possible machine-learning techniques were explored (KLEIN, 2023; KLEIN *et al.*, 2024a). The main advantage of the methods presented is the nondependency on the knowledge of the ArUcos position.

However, since the goal of the work was just an initial study related to the validation of the use of the approaches and not the final localization system, several limitations are presented, and further studies and analyses are necessary. Some future studies can better explore the application of the model trained in simulation in the real world (even if the results are promising, several improvements are needed to make the localization system complete and usable). Another relevant future work consists of exploring the optimization of the hyperparameters of the methods since the optimizations can drive a fundamental step to make the system better and closer to a final localization system.

4 METHODOLOGY

As an extension of the previous study performed in (KLEIN, 2023), whose summary was presented in Chapter 3, this chapter presents the details of the current work, that is based on the same scenario context, i.e., the RaF competition, and used the same data from the simulator. It is important to recall the three main research questions that drive the study, presented in Section 1.3. Question one aims to identify and understand the challenges in the optimization finding; Question two aims to find the relations between the optimizations for each context; and Question three is related to exploring the use of the optimization from one context to another.

The data used in this work was the same as presented before, consisting of the one collected from the limited part of the field, considering four different grid sizes: 10.0 mm, 5.0 mm, 2.5 mm, and 1.0 mm (datasets D1, D2, D3, and D4, respectively, presented in Section 3.2.2). In addition, this work uses Concept 2, presented in Section 3.2, which consists of three independent models receiving information about one ArUco marker and each model returning one pose component (x, y and θ). Based on the previous work, this decision was to have one model per pose component (as presented in Chapter 3). The exploration of using the same model to estimate more than one pose component is out of the scope of this work and will be explored in future works.

The method used to optimize the models was the Bayesian Optimization, explained in Section 2.3.1. This approach was selected due to the problems' characteristics, particularly the expensive model evaluation, once it is necessary to train all the models. The framework used was the Skopt library, which was developed for scikit-learn and based on Bayesian Optimization. Another possible approach would be verifying all the possibilities in the defined search space and then comparing all the results to get the best parameters (methodology implemented for the library Grid-Search[1]), or also a Random Search[2] on the same search space. However, due to the huge number of possibilities, these approaches are not feasible due to the large amount of time and computational demand.

According to the official documentation by scikit-learn, the MLP Regressor[3] has 23 hyperparameters. Considering all possible values for all hyperparameters, the search region is infinite since some parameters are real values. Through some empirical tests, where manual variations on the model structure were performed, some aspects were found to help restrict the number of possibilities in the search space. For instance, the activation function that gave the best results in the empirical tests was the *ReLu* function. It was kept as the "default" value in the proposed optimization. Table 4 presents the default values (from the used library) for the hyper-

[1] Available in: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed on November 1, 2024.

[2] Available in https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html. Accessed on November 1, 2024.

[3] Available in: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html. Accessed on November 5, 2024.

parameters used in the MLP models, while Table 5 presents the modified hyperparameters, i.e. the non-default.

Effectively, the only hyperparameter optimized was the hidden layer size, which consists of a list of integer values, where each position represents one layer, and the integer value represents the number of neurons in that layer. For instance, a model with hidden layers size (120, 120, 120) has 3 layers and 120 neurons in each layer. Also, theoretically, there is no restriction to the number of layers or neurons in each layer, which can differ between the layers in the same model.

Due to the high number of possibilities (infinite in theory), some restrictions to the search space were defined: (1) the number of layers must be between 1 and 11; (2) the number of neurons in each layer between 1 and 1001, and (3) all the layers in the model must have the same size, that is, the same number of neurons in each layer. These restrictions were performed after a series of empirical tests showing that most optimizations were in the defined range of layers and neurons. It is trivial to notice that these limitations can greatly impact optimization. Still, the study aims to show that it is possible to present some optimization and explore the behavior between the scenarios. If some optimal solution can be found in a restrictive space, it can also be found in a larger space. It is important to emphasize that the optimization in this work aims to improve the model structure and not the value of the hyperparameters itself. The study focuses on finding the best number of layers and neurons, not the model weights, which will be gathered through training (using the *backpropagation* method with the Adam solver defined before).

Table 4 – MLP Default Hyperparameters. Based on (KLEIN *et al.*, 2024b).

Hyper Parameter	Value
Alpha	1×10^{-4}
Beta_1	9×10^{-1}
Beta_2	9.99×10^{-1}
Epsilon	1×10^{-8}
Learning rate init	1×10^{-3}
Max iterations	$2 \times 10^{+2}$
Shuffle	True
Solver	adam
Validation fraction	1×10^{-1}

Table 5 – MLP Non-Default HyperParameters. Based on (KLEIN *et al.*, 2024b).

Hyper Parameter	Value
Batch size	$1 \times 10^{+2}$
Early stopping	True
Random state	$4.2 \times 10^{+1}$
Tol	1×10^{-3}
Validation fraction	2×10^{-1}
Warm start	True

For each scenario (related to each data collection in the limited part of the field), there are three independent ML models (two for position, x and y, and one for orientation θ), resulting in 12 independent models that will be optimized independently. Considering the number of models and the training data size (which directly impact the necessary time to train and evaluate each model), the maximum number of interactions for the Bayesian optimizer was 20, with 10 initial points. This choice aimed to minimize the time spent on the optimization and still have some exploration in the Bayesian Optimization. All the datasets were randomly divided into 70% for training and 30% for testing while finding the optimization. This division was done following a common division of training/testing data.

Once the optimization of the hyperparameters was found for each model in each scenario, they were applied to the models and evaluated using a cross-validation technique, considering a K-fold of 5. This re-evaluation of the models was done to avoid any possible *overfitting* that could be coming from the data used during the model's optimization. Finally, another test applied the optimization found for the 10.0 mm scenario to the other scenarios, aiming to work as a kind of transfer learning model structure and evaluate its performance. The flowchart in Figure 18 summarizes all the optimization processes.

Finally, it is important to emphasize that this study focuses on optimizing the hyperparameters of the ML models considering the simulation scenario of the RaF. Other issues, such as the applicability in the real scenario and aspects related to the embedded systems (such as the execution in a Raspberry Pi or any other platform), are beyond the scope of this work and will be treated as future work.

In this study, all computations were performed using a CPU, specifically an AMD[4] EPYC 7351 16-Core Processor (2.40 GHz) with 32 GB of RAM, without the use of a GPU. The implementation was carried out in a 64-bit Operating System Windows 10 Enterprise LTSC, a Python 3.10.7 [5], an open-source programming language. The analysis utilized the following open-source libraries, installed via the Python Package Index (PyPI)[6]: Pandas 1.5.0, OpenCV 4.6.0, and Scikit-learn 1.1.2.

[4] Founded: May 1, 1969, Sunnyvale, California, United States

[5] Obtained from https://www.python.org/, accessed on December 10, 2024

[6] https://pypi.org/, accessed on November 27, 2024

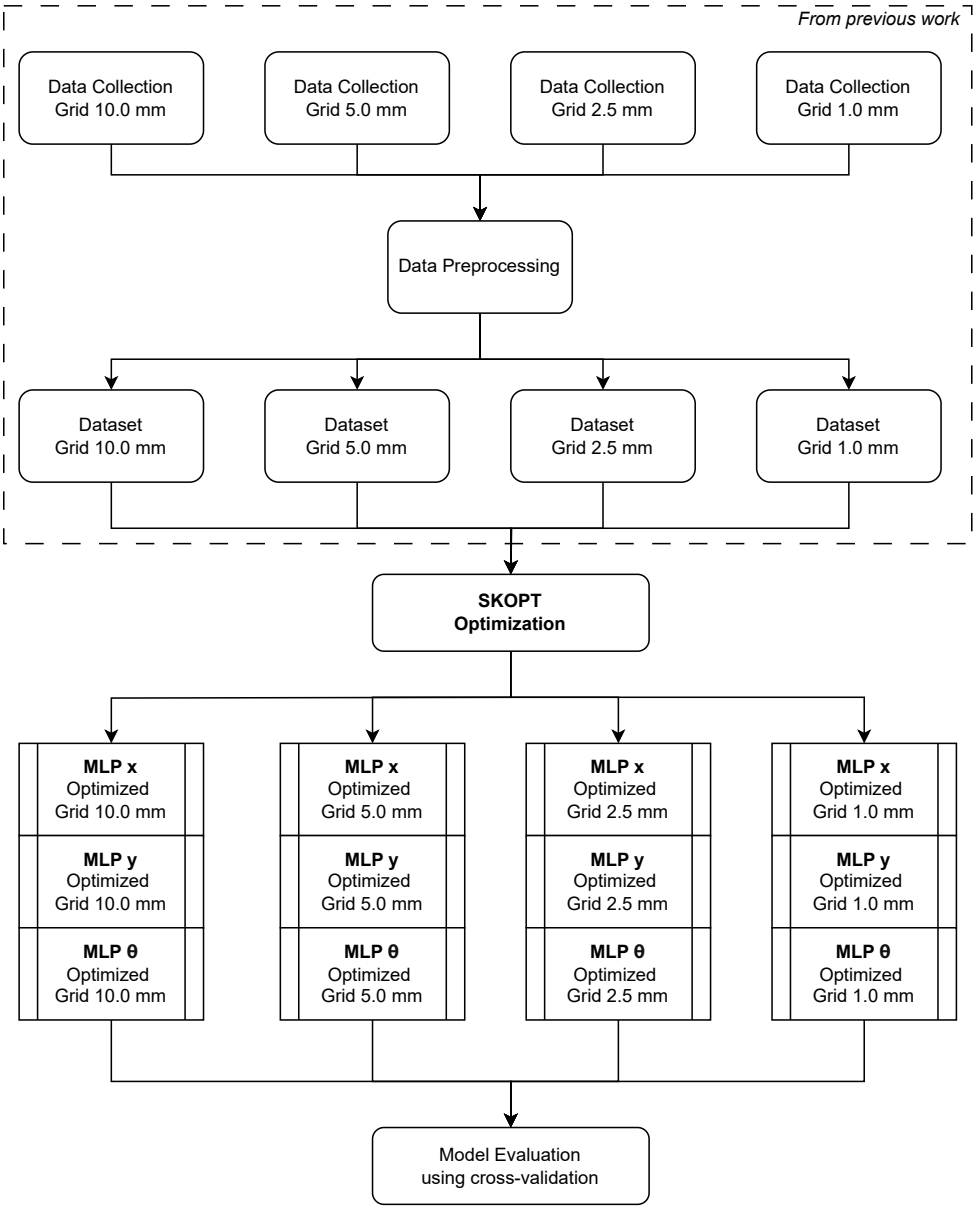


Figure 18 – Flowchart of the current work methodology. The first steps, inside of the slashed rectangle, are from the previous work. Based on: (KLEIN *et al.*, 2024b).

5 RESULTS AND DISCUSSIONS

Aiming to explore the optimization of the model structure between different scenarios of the same contexts, the MLP model, based on Concept 2, was optimized using a Bayesian Optimization in a limited search space. The methodology, presented in Section 4, presents the characteristics of the method and the tests to be performed to collect the results.

Figure 19 presents the optimization found for each scenario, where graph A shows the number of layers, while B shows the number of neurons in each layer. An interesting aspect is the different behavior of the optimization solutions for each model in the same scenario and between different scenarios. It is possible to notice that the number of layers varied between 1 and 11, as did the number of neurons in each layer, which also varied between 159 and 1001. In other words, no direct relation or pattern was observed between the optimizations, which corresponded to the expectations since the models do not directly depend on each other.

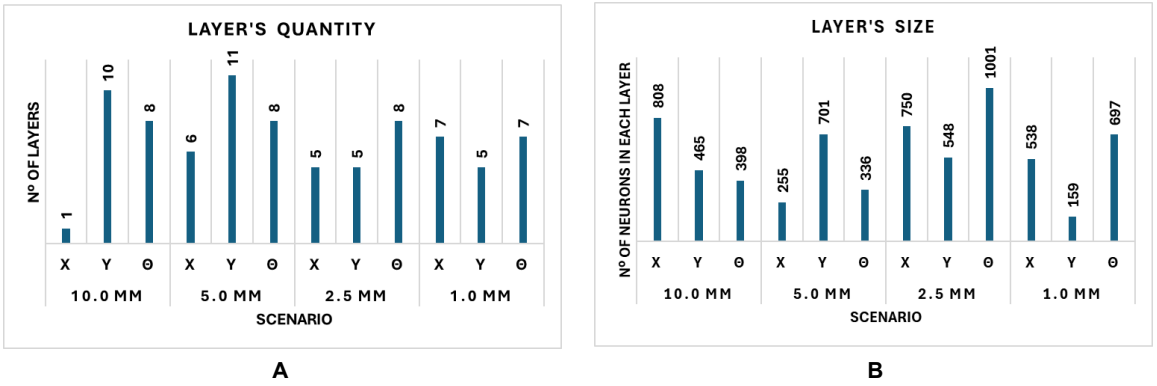


Figure 19 – Comparison between the scenarios' optimization, with the quantity of the layers and the number of neurons in each layer.

Table 6 compares the results, considering the MAE measure considering a cross-validation process of K times equal to 5, between scenarios with and without optimizations. To compute the percentage difference between the MAE from optimized and non-optimized values, Equation 4 was used. In addition, the necessary time to find each optimization is also present in the table, with the values presented in seconds. Upon analysis of the results, it is trivial to notice that the optimized models presented an improvement in consistency compared to the non-optimized model, with improvements between 10% and 65%. The last aspect of the results is the necessary time to find the optimization, which has increased considerably in the different scenarios due to the high difference in the image quantity in the dataset.

It is essential to mention that the optimization was performed using the computing resources presented in Chapter 4, and the time measurement present in the results is based on the running of that machine. No other activity (such as web browsing, video compilation, etc.) was executed during the optimizations, and the results were collected immediately after each

other. However, deactivation of the Graphical User Interface (GUI) in the operating system and other further operations over the time measurement were not performed.

$$Percentage\ Difference = \frac{|MAE_{Not\ Optimized} - MAE_{Optimized}|}{MAE_{Not\ Optimized}} \tag{4}$$

Table 6 – The optimization findings for each model, the time spent on each optimization, and the mean absolute error (MAE) obtained in the evaluation process. Based on (KLEIN *et al.*, 2024b).

	10.0 mm			5.0 mm			2.5 mm			1.0 mm		
	x [m]	y [m]	θ [°]	x [m]	y [m]	θ [°]	x [m]	y [m]	theta	x [m]	y [m]	θ [°]
MAE Not Optimized	0.0281	0.0259	30.32	0.0275	0.0282	22.34	0.0209	0.0259	19.69	0.0188	0.0178	16.41
MAE Optimized	0.0253	0.0227	20.08	0.0205	0.0155	16.80	0.0097	0.0100	7.61	0.0066	0.0087	5.88
Percentage Difference	10.02%	12.37%	33.78%	25.64%	45.02%	24.81%	53.74%	**61.29%**	61.33%	**64.94%**	51.22%	**64.14%**
Time Spent [s]	**12476**	**23906**	**20544**	65273	81491	53335	386191	258969	208331	550760	550248	417341
Nº of layers	1	10	8	6	11	8	5	5	8	7	5	7
Nº of neurons in each layer	808	465	398	255	701	336	750	548	1001	438	159	697

The second result consists of analyzing the application of the optimization found for the scenario with a 10.0 mm grid (here called the 10.0 mm optimized structure), the fastest optimization found, presented in Table 6. Table 7 presents a comparison between the application of the optimization previously found (Table 6, called Optimized Structure), against the 10.0 mm Optimized Structure.

Table 7 – A comparison of the optimized structures results with applying the 10.0 mm optimized structure in the other scenarios. Based on (KLEIN *et al.*, 2024b).

	5.0 mm			2.5 mm			1.0 mm		
	x	y	θ	**x**	**y**	θ	**x**	**y**	θ
Optimized Structure [%]	25.64	**45.02**	24.81	**53.74**	**61.29**	**61.33**	**64.94**	51.22	**64.14**
10.0 mm Optimized Structure [%]	**26.23**	37.55	**31.10**	2.88	55.90	56.82	9.82	**54.38**	60.67
Diff in time to find the optimization [%]	80.89	70.66	61.48	96.77	90.77	90.14	97.73	95.66	95.08

Observing the results presented in Table 7, it is possible to notice that the errors obtained using the optimized structure of 10.0 mm were better than compared to the non-optimized models, presenting, in the worst case (2.5 mm, x), an improvement of 2.88%, but also achieving improvements around 60% (1.0 mm, θ). In addition, the 10.0 mm optimized structure was even better than those found for the optimized structure specific for some scenarios: 5.0 mm x and θ, and 1.0 mm y (which is expected since the first optimization is not necessary for the global optimum, so improvements on that are expected). However, the Optimized Structure localization was better for most cases, specifically for 2.5 mm and 1.0 mm, where the differences were bigger than 50%.

Also, through the analysis of Table 7, it is possible to notice the great differences in finding the optimum structure. The 5.0 mm scenario showed a difference of over 60%, while the

2.5 mm and 1.0 mm scenarios demonstrated a difference of over 90%. These findings present an intriguing potential trade-off: utilizing significantly less computational power and time can identify an interesting optimization, even if one may not be as optimal as the one specifically tailored for each scenario.

6 CONCLUSIONS

This current work aimed to explore the optimization of the hyperparameters of the machine learning models for robot localization, considering the scenario of the RobotAtFactory 4.0 competition. This work is based on a previous study performed by the author in (KLEIN, 2023) and in (KLEIN *et al.*, 2023b; KLEIN *et al.*, 2023a; KLEIN *et al.*, 2024a). The first study validated the feasibility and possibility of using the ML and deep learning models, considering aspects of different models and factors such as response time and model size. The current work, based on the results of the previous study, aimed to explore the hyperparameters optimization of the ML models, particularly the MLP model, using Bayesian Optimization.

The first research question of this work (*What are the challenges to finding an optimization for the MLP model in the RaF scenario?*) can be answered based on the results of the time needed to find the optimizations, which for the scenario with the most significant amount of images (655.130), achieving around 6 days for one component of the pose. In this way, computational power is the biggest challenge for the model optimization in this work. Another related challenge observed was the correct definition of the search space, delimiting the range of the possibilities of the hyperparameters and the definition of which hyperparameters should be optimized and the values of the fixed hyperparameters (i.e., those that will not be optimized). These topics are challenging since they directly impact the values achieved by the optimization due to the restrictions of the possible optimizations to be found.

The second research question of this work (*Is there a correlation between the optimizations for different contexts, considering the RaF scenario?*) can be answered based on the analysis of the results that no direct relationship can be observed in the optimizations between the scenarios. However, 9 of 12 optimizations had between 5 and 8 layers, and all the layers had more than 100 neurons. The quality of the optimizations was of the same magnitude, varying between 10.02% to 64.94%.

The third research question of this work (*Is it possible to apply optimizations found for one context in others, and what are the advantages and disadvantages associated?*) can be answered based on the results of using the 10.0 mm optimization structure in the other scenarios. All the scenarios presented improvements compared to the non-optimized structure, even this improvement achieving only 2.88% in one case. Compared to the dedicated optimized structure, the 10.0 mm optimization structure achieved better results in 3 of 9 cases. These facts show that it is possible to apply optimizations from one context in another, with the best advantage of the less time needed to find the optimization (achieving more than 97% in some cases). The main disadvantage is the lack of quality of optimization, i.e., the improvement of the optimization can not be as good as if a dedicated optimization is found for the scenario.

In this way, once all the research questions were answered successfully, it is possible to conclude that the research was successful. The MLP models for pose estimation in the RobotAtFactory 4.0 were optimized using Bayesian Optimization, focusing on the number of layers

and their sizes, with several limitations imposed to limit the search space. Six of the 12 models presented an improvement more significant than 50%. Some optimizations require a high computational cost of around six days to generate. An exciting approach proposed involves transferring the optimization structures between scenarios, using the one found in the smallest (the 10.0 mm grid in the current work) and the larger one, presenting a good improvement in the results (even not as good as the dedicated optimization) still could be found using much less time. A paper about this work was published in (KLEIN *et al.*, 2024b).

The use of studied optimization, Bayesian Optimization, and the transfer of optimization from a small scenario to a larger one (in this work, from a small grid resolution to a large one) can appear as an interesting approach to any other similar scenario, where the optimization found for a large scenario is extremely expensive compared to a simpler scenario. Further investigation is needed on each specific scenario. Still, it can appear as an interesting approach that provides, at the same time, improvement of the results compared to models with no optimization and a feasible time to find it.

In addition, assessing whether the pre-optimization results are sufficient for the models is crucial. If not, a cost-benefit analysis of the time and computational resources required for optimization becomes essential. In the context of the RobotAtFactory 4.0 competition, it is also important to consider whether the improvements will meaningfully improve the performance during the event. In addition, time constraints could become a significant challenge if model optimization has to be performed during the competition rather than before.

It is essential to emphasize that several limitations were applied in this study. Only the MLP model was used in this study, even though several other techniques were explored in previous work. In the optimization aspects, only the number of layers and neurons in each layer was considered. Another major limitation was using the Bayesian Optimization, which directly impacted the optimization found. The current work has explored specific optimization aspects of ML in robot localization, and further study on other topics is still needed. Another point of attention is related to the limited scenario context, which only considered part of the field in the simulation of the RaF competition, lacking further exploration of the whole field in real competition and other scenarios, which may present a less controlled scenario than the one used in the work. The explicability of the models was another relevant issue that was not explored in the work.

Future works will research additional optimization strategies, including alternative approaches to Bayesian optimization. Furthermore, the search space and hyperparameters could be expanded, and other factors such as model size, robustness, and others could be incorporated into the optimization process. In addition, other studies have sought to improve the efficacy of alternative methodologies, including Random Forest and CNN, using the same model to estimate more than one pose component per time (for instance, one model to estimate the three components). Other research efforts have aimed to optimize the entire field rather than address its constraints, elucidate potential discrepancies, and examine the diverse influences of optimiza-

tion in both simulated and real-world contexts. Finally, future works include further analysis of the practical impact of the model's optimization on the RobotAtFactory 4.0 competition, discussing the real cost-effectiveness of finding optimizations in the practical application.

REFERENCES

AHMAD, T. *et al.* Location-enabled IoT (LE-IoT): Indoor Localization for IoT Environments using Machine Learning . *In*: **2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)**. Los Alamitos, CA, USA: IEEE Computer Society, 2024. p. 392–399. Disponível em: https://doi.ieeecomputersociety.org/10.1109/DCOSS-IoT61029.2024.00065.

ARULAMPALAM, M. S. *et al.* A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. **IEEE Transactions on signal processing**, Ieee, v. 50, n. 2, p. 174–188, 2002.

ATANASYAN, A.; ROSSMANN, J. Improving self-localization using cnn-based monocular landmark detection and distance estimation in virtual testbeds. *In*: **Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter**. [*S.l.*]: Springer, 2019. p. 249–258.

AVGERIS, M. *et al.* Single vision-based self-localization for autonomous robotic agents. *In*: IEEE. **2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)**. [*S.l.*], 2019. p. 123–129.

BECKER, A. **Online kalman filter tutorial**. 2022. Disponível em: https://www.kalmanfilter.net/background.html.

BERGSTRA, J. *et al.* Algorithms for hyper-parameter optimization. *In*: SHAWE-TAYLOR, J. *et al.* (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2011. v. 24. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012.

BEST, P. J. A method for registration of 3-d shapes. **IEEE Trans Pattern Anal Mach Vision**, v. 14, p. 239–256, 1992.

BIBER, P.; STRASSER, W. The normal distributions transform: A new approach to laser scan matching. *In*: IEEE. **Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)**. [*S.l.*], 2003. v. 3, p. 2743–2748.

BISCHL, B. *et al.* Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. **WIREs Data Mining and Knowledge Discovery**, v. 13, n. 2, p. e1484, 2023. Disponível em: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1484.

BISHOP, G.; WELCH, G. *et al.* An introduction to the kalman filter. **Proc of SIGGRAPH, Course**, v. 8, n. 27599-23175, p. 41, 2001.

BRAUN, J. *et al.* Design and development of an omnidirectional mecanum platform for the robotatfactory 4.0 competition. *In*: YOUSSEF, E. S. E. *et al.* (Ed.). **Synergetic Cooperation Between Robots and Humans**. Cham: Springer Nature Switzerland, 2024. p. 114–125. ISBN 978-3-031-47269-5.

BRAUN, J. *et al.* A robot localization proposal for the robotatfactory 4.0: A novel robotics competition within the industry 4.0 concept. **Frontiers in Robotics and AI**, v. 9, 11 2022.

BRAUN, J. *et al.* Robotatfactory 4.0: a ros framework for the simtwo simulator. *In*: **2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)**. [*S.l.*: *s.n.*], 2022. p. 205–210.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

BROCHU, E.; CORA, V. M.; FREITAS, N. D. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. **arXiv preprint arXiv:1012.2599**, 2010.

BRUBAKER, M. A.; GEIGER, A.; URTASUN, R. Map-based probabilistic visual self-localization. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 38, n. 4, p. 652–665, 2015.

CHEN, C. *et al.* Deep learning for visual localization and mapping: A survey. **IEEE Transactions on Neural Networks and Learning Systems**, v. 35, n. 12, p. 17000–17020, 2024.

CHOSET, H. *et al.* **Principles of robot motion: theory, algorithms, and implementations**. [*S.l.*]: MIT press, 2005.

COSTA, P. *et al.* Simtwo realistic simulator: A tool for the development and validation of robot software. **Theory and Applications of Mathematics & Computer Science**, v. 1, p. 17–33, 04 2011.

DELLAERT, F. *et al.* Monte carlo localization for mobile robots. *In*: IEEE. **Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)**. [*S.l.*], 1999. v. 2, p. 1322–1328.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **IEEE robotics & automation magazine**, IEEE, v. 13, n. 2, p. 99–110, 2006.

EBERHART, R. C.; SHI, Y. Comparison between genetic algorithms and particle swarm optimization. *In*: SPRINGER. **International conference on evolutionary programming**. [*S.l.*], 1998. p. 611–616.

ESFAHLANI, S. S. *et al.* The deep convolutional neural network role in the autonomous navigation of mobile robots (srobo). **Remote Sensing**, MDPI, v. 14, n. 14, p. 3324, 2022.

FEURER, M.; HUTTER, F. Hyperparameter optimization. **Automated machine learning: Methods, systems, challenges**, Springer International Publishing, p. 3–33, 2019.

FOX, D. **Markov localization-a probabilistic framework for mobile robot localization and navigation.** 1998. Tese (Doutorado) — Universität Bonn, 1998.

FOX, D. *et al.* Monte carlo localization: Efficient position estimation for mobile robots. **AAAI/IAAI**, v. 1999, n. 343-349, p. 2–2, 1999.

FOX, D.; BURGARD, W.; THRUN, S. Markov localization for mobile robots in dynamic environments. **Journal of artificial intelligence research**, v. 11, p. 391–427, 1999.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Disponível em: https://doi.org/10.1214/aos/1013203451.

GARRIDO-JURADO, S. *et al.* Automatic generation and detection of highly reliable fiducial markers under occlusion. **Pattern Recognition**, v. 47, n. 6, p. 2280–2292, 2014. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.*]: MIT Press, 2016. http://www.deeplearningbook.org.

GORDON, N. J.; SALMOND, D. J.; SMITH, A. F. Novel approach to nonlinear/non-gaussian bayesian state estimation. *In*: IET. **IEE proceedings F (radar and signal processing)**. [*S.l.*], 1993. v. 140, n. 2, p. 107–113.

GREWAL, M. S.; WEILL, L. R.; ANDREWS, A. P. **Global positioning systems, inertial navigation, and integration**. Hoboken, NJ, USA: John Wiley & Sons, 2007. ISBN 9780470041901.

HUANG, S.; DISSANAYAKE, G. Robot localization: An introduction. *In*: ____. **Wiley Encyclopedia of Electrical and Electronics Engineering**. Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2016. p. 1–10. ISBN 9780471346081. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W8318.

HUANG, Y. *et al.* Real-time passive source localization: A practical linear-correction least-squares approach. **IEEE transactions on Speech and Audio Processing**, IEEE, v. 9, n. 8, p. 943–956, 2001.

JAMES, G. *et al.* **An Introduction to Statistical Learning: With Applications in R**. [*S.l.*]: Springer Publishing Company, Incorporated, 2014. ISBN 1461471370.

KALAITZAKIS, M. *et al.* Fiducial markers for pose estimation. **Journal of Intelligent & Robotic Systems**, Springer, v. 101, n. 4, p. 1–26, 2021.

KALAITZAKIS, M. *et al.* Experimental comparison of fiducial markers for pose estimation. *In*: **2020 International Conference on Unmanned Aircraft Systems (ICUAS)**. [*S.l.*: *s.n.*], 2020. p. 781–789.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, v. 82, n. 1, p. 35–45, 03 1960. ISSN 0021-9223. Disponível em: https://doi.org/10.1115/1.3662552.

KASPAROV, G. **Deep thinking: where machine intelligence ends and human creativity begins**. [*S.l.*]: Hachette UK, 2017.

KENDALL, A.; GRIMES, M.; CIPOLLA, R. Posenet: A convolutional network for real-time 6-dof camera relocalization. *In*: **Proceedings of the IEEE international conference on computer vision**. [*S.l.*: *s.n.*], 2015. p. 2938–2946.

KERDJIDJ, O. *et al.* Uncovering the potential of indoor localization: Role of deep and transfer learning. **IEEE Access**, v. 12, p. 73980–74010, 2024.

KLEIN, L. C. **Intelligent sensorization system using ML applied to robotics**. 2023. Dissertação (Mestrado) — Polytechnic Institute of Bragança (IPB), 2023.

KLEIN, L. C. *et al.* Using machine learning approaches to localization in an embedded system on robotatfactory 4.0 competition: A case study. *In*: **2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)**. [*S.l.*: *s.n.*], 2023. p. 69–74.

KLEIN, L. C. *et al.* A machine learning approach to robot localization using fiducial markers in robotatfactory 4.0 competition. **Sensors**, v. 23, n. 6, 2023. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/23/6/3128.

KLEIN, L. C. *et al.* Deep learning-based localization approach for autonomous robots in the robotatfactory 4.0 competition. *In*: PEREIRA, A. I. *et al.* (Ed.). **Optimization, Learning**

Algorithms and Applications. Cham: Springer Nature Switzerland, 2024. p. 181–194. ISBN 978-3-031-53036-4.

KLEIN, L. C. *et al.* Optimization of machine learning models applied to robot localization in the robotatfactory 4.0 competition. *In*: PEREIRA, A. I. *et al.* (Ed.). **Optimization, Learning Algorithms and Applications**. Cham: Springer Nature Switzerland, 2024. p. 112–125. ISBN 978-3-031-77425-6.

KOTSIANTIS, S. B. Decision trees: a recent overview. **Artificial Intelligence Review**, Springer, v. 39, n. 4, p. 261–283, 2013.

KOÇOĞLU, F. Research on the success of unsupervised learning algorithms in indoor location prediction. **International Advanced Researches and Engineering Journal**, v. 6, p. 148–153, 08 2022.

KUHN, M. **Applied predictive modeling**. [*S.l.*]: Springer, 2013.

LAUER, M.; LANGE, S.; RIEDMILLER, M. Calculating the perfect match: An efficient and accurate approach for robot self-localization. *In*: BREDENFELD, A. *et al.* (Ed.). **RoboCup 2005: Robot Soccer World Cup IX**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 142–153. ISBN 978-3-540-35438-3.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

MAGNUSSON, M. **The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection**. 2009. Tese (Doutorado) — Örebro universitet, 2009.

MAGNUSSON, M. *et al.* Evaluation of 3d registration reliability and speed-a comparison of icp and ndt. *In*: IEEE. **2009 IEEE International Conference on Robotics and Automation**. [*S.l.*], 2009. p. 3907–3912.

MALLIK, M.; DAS, S.; CHOWDHURY, C. Rank based iterative clustering (rbic) for indoor localization. **Engineering Applications of Artificial Intelligence**, v. 121, p. 106061, 2023. ISSN 0952-1976. Disponível em: https://www.sciencedirect.com/science/article/pii/S0952197623002452.

MAYBECK, P. S. The kalman filter: An introduction to concepts. *In*: **Autonomous robot vehicles**. [*S.l.*]: Springer, 1990. p. 194–204.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–137, 1943.

MOCKUS, J. On bayesian methods for seeking the extremum. *In*: **Proceedings of the IFIP Technical Conference**. [*S.l.*: *s.n.*], 1974. p. 400–404.

MONTGOMERY, D. C. **Design and analysis of experiments**. [*S.l.*]: John wiley & sons, 2017.

MURTAGH, F. Multilayer perceptrons for classification and regression. **Neurocomputing**, v. 2, n. 5, p. 183–197, 1991. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/0925231291900235.

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. **Frontiers in Neurorobotics**, v. 7, 2013. ISSN 1662-5218. Disponível em: https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021.

NEBEKER, N. F. L. **External Robot Localization in 6DoF**. 10 2015. Dissertação (Mestrado) — Faculdade de Engenharia da Universidade do Porto (FEUP), 10 2015. Disponível em: https://repositorio-aberto.up.pt/handle/10216/80454.

NESSA, A. *et al.* A survey of machine learning for indoor positioning. **IEEE Access**, v. 8, p. 214945–214965, 2020.

OGISO, S. *et al.* Self-localization method for mobile robot using acoustic beacons. **ROBOMECH Journal**, Springer, v. 2, n. 1, p. 1–12, 2015.

PANIGRAHI, P. K.; BISOY, S. K. Localization strategies for autonomous mobile robots: A review. **Journal of King Saud University - Computer and Information Sciences**, v. 34, n. 8, Part B, p. 6019–6039, 2022. ISSN 1319-1578. Disponível em: https://www.sciencedirect.com/science/article/pii/S1319157821000550.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

REKLEITIS, I.; DUDEK, G.; MILIOS, E. Probabilistic cooperative localization and mapping in practice. *In*: IEEE. **2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)**. [*S.l.*], 2003. v. 2, p. 1907–1912.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSINKIEWICZ, S.; LEVOY, M. Efficient variants of the icp algorithm. *In*: IEEE. **Proceedings third international conference on 3-D digital imaging and modeling**. [*S.l.*], 2001. p. 145–152.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. [*S.l.*]: Prentice Hall, 2010.

SATTLER, T.; LEIBE, B.; KOBBELT, L. Fast image-based localization using direct 2d-to-3d matching. *In*: IEEE. **2011 International Conference on Computer Vision**. [*S.l.*], 2011. p. 667–674.

SIEGWART, R.; CHLI, M.; LAWRANCE, N. **Autonomous Mobile Robots MOOC**. 2014. https://www.edx.org/course/autonomous-mobile-robots-ethx-amrx-1.

SIMON, D. **Evolutionary optimization algorithms**. [*S.l.*]: John Wiley & Sons, 2013.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *In*: . [*S.l.*]: Computational and Biological Learning Society, 2015. p. 1–14.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. **Advances in neural information processing systems**, v. 25, 2012.

SOBREIRA, H. *et al.* Map-matching algorithms for robot self-localization: A comparison between perfect match, iterative closest point and normal distributions transform. **Journal of Intelligent & Robotic Systems**, v. 93, 03 2019.

STACHNISS, C. **Robot Localization - An Overview**. 2021. Disponível em: https://www.youtube.com/watch?v=8VJ-A9OlhAE&ab_channel=CyrillStachniss.

SUN, S. *et al.* A survey of optimization methods from a machine learning perspective. **IEEE Transactions on Cybernetics**, v. 50, n. 8, p. 3668–3681, 2020.

SZEGEDY, C. *et al.* Going deeper with convolutions. *In*: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.*: *s.n.*], 2015. p. 1–9.

TORREY, L.; SHAVLIK, J. Transfer learning. *In*: **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**. [*S.l.*]: IGI global, 2010. p. 242–264.

WANG, J.; TAKAHASHI, Y. Indoor mobile robot self-localization based on a low-cost light system with a novel emitter arrangement. **ROBOMECH Journal**, SpringerOpen, v. 5, n. 1, p. 1–17, 2018.

WILBERS, D.; MERFELS, C.; STACHNISS, C. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. *In*: **Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)**. [*S.l.*: *s.n.*], 2019.

WU, J. *et al.* Hyperparameter optimization for machine learning models based on bayesian optimizationb. **Journal of Electronic Science and Technology**, v. 17, n. 1, p. 26–40, 2019. ISSN 1674-862X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1674862X19300047.

YANASE, R. *et al.* Lidar-and radar-based robust vehicle localization with confidence estimation of matching results. **Sensors**, MDPI, v. 22, n. 9, p. 3545, 2022.

YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. **Neurocomputing**, v. 415, p. 295–316, 2020. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231220311693.

YAZICI, M. T.; BASURRA, S.; GABER, M. M. Edge machine learning: Enabling smart internet of things applications. **Big Data and Cognitive Computing**, v. 2, n. 3, 2018. ISSN 2504-2289. Disponível em: https://www.mdpi.com/2504-2289/2/3/26.

ZHANG, Q. Some implementation aspects of sliding window least squares algorithms. **IFAC Proceedings Volumes**, Elsevier, v. 33, n. 15, p. 763–768, 2000.