- Master Thesis -

Adaptive Local Planning for Improved Pose Certainty in Active SLAM

Robin Dominic Steiger

21. February 2025
Verified: 21.02.2025

University of Freiburg
Department of Computer Science
Chair of Computer Networks and Telematics
Prof. Dr. Christian Schindelhauer

in cooperation with

Universidade de Lisboa
Instituto Superior Técnico
Institute for Systems and Robotics
Prof. Dr. Pedro Manuel Urbano de Almeida Lima

Candidate

Robin Dominic Steiger

Matrikel number

4146672

Working period

21. 08. 2024 – 21. 02. 2025

First Examiner

Prof. Dr. Christian Schindelhauer

Second Examiner

Prof. Dr. Pedro Manuel Urbano de Almeida Lima

Supervisors

AIS Freiburg: Dr. Lukas Luft, Dulce Adriana Gómez Rosal;
ISR Lisboa: Rui Bettencourt, Rodrigo Serra;
CoNe Freiburg: Sneha Mohanty

DECLARATION

I hereby declare that I have independently authored this thesis, used no sources or aids other than those stated, and have identified as such all passages that are taken either literally or in spirit from published writings. Furthermore, I declare that this thesis has not been submitted, in whole or in part, for any other examination.

Place, Date	Signature

Acknowledgments

blablabla thank you :-* **(DRAFT: add jose)**

Abstract

This thesis addresses the challenge of autonomous navigation in unknown environments while concurrently actively enhancing the accuracy of Simultaneous Localization and Mapping. It introduces an advanced modification to the Dynamic Window Approach Planner within the Robot Operating System, designed to dynamically balance goal-directed navigation with essential pose improvement actions.

The core innovation lies in the planner's ability to evaluate and modify the trajectory based on real-time pose certainty, which is assessed through a novel pose certainty approximation method leveraging an advanced map representation. This adaptive mechanism allows the planner to scale the influence of pose improvement actions in response to varying levels of localization certainty, thereby optimizing navigation efficiency and reliability.

Extensive evaluations in simulated and real-world settings demonstrate that our approach notably improves navigation accuracy in unknown environments. The results indicate significant implications for service robotics and similar sectors which require robust autonomous navigation, demonstrating the practical and efficient nature of the integrated approach in handling the dual objectives of achieving navigation goals while ensuring high localization accuracy.

Contents

1 Introduction **12**

1.1 Motivation . . . 12
1.2 Research Objectives . . . 13
1.3 Contributions . . . 14
1.4 Thesis Structure . . . 15

2 Background **17**

2.1 Simultaneous Localization and Mapping . . . 17
2.2 Particle Filters . . . 18
2.2.1 Fast SLAM 2.0 . . . 19
2.2.2 GMapping . . . 19
2.3 Map Representation . . . 20
2.3.1 Occupancy Grid Map . . . 20
2.3.2 Full Map Posterior Maps . . . 21
2.4 Entropy in Particle Filters . . . 23
2.5 Map Entropy Calculation . . . 24
2.5.1 FMP Map Entropy . . . 24
2.6 Pose Entropy . . . 25
2.6.1 Approaches for Particle Filters . . . 25
2.6.2 Expected Map Mean Information . . . 26
2.7 Robot Operating System . . . 29
2.8 ROS Navigation Stack . . . 29
2.8.1 Local Planners . . . 30

3 State-of-the-Art **35**

3.1 Active SLAM Approaches . . . 35
3.2 Approaches in RoboCup@Home . . . 36
3.3 Towards a Combined Approach . . . 36

4 Adaptive Local Planning for Improved Pose Certainty in Active SLAM 38
4.1 Pose Improvement . . . 39
4.1.1 Strategies for Pose Improvement . . . 39
4.1.2 The new PoseBoost cost function . . . 41
4.2 Pose Entropy . . . 44
4.2.1 Current Approach in GMapping . . . 44
4.2.2 Limitations and Rationale for Improvement . . . 44
4.2.3 Review of Alternative Approaches . . . 45
4.2.4 Expected Map Mean Information . . . 46
4.3 Balancing Goal-Directed Navigation and Pose Improvement . . . 52
4.3.1 System and Variable Analysis . . . 53
4.3.2 Adaptive Weighting Mechanism . . . 55
4.3.3 Managing conflicting objectives . . . 56
4.3.4 System Configuration and Parameter Integration . . . 56
4.3.5 Final PoseBoost cost function Output . . . 57
4.3.6 Full DWA Optimization Formulation . . . 58

5 Implementation 59
5.1 Pose Improvement . . . 59
5.1.1 GMapping Framework and Requirements . . . 59
5.1.2 Implementation of Components . . . 60
5.2 Pose Entropy . . . 65
5.2.1 GMapping Framework and Overview of Modifications . . . 65
5.2.2 Implementation of the New Approach . . . 66

6 Experiments and Evaluation 70
6.1 Comparison of Planning Strategies . . . 70
6.2 Testing Metrics . . . 71
6.3 Experimental Setup . . . 71
6.4 Results . . . 75
6.5 Discussion . . . 78

7 Conclusion and Future Work 80
7.1 Realization of Contributions . . . 80
7.2 Challenges and Limitations of the Study . . . 81
7.3 Future Research Directions . . . 82
7.4 Concluding Remarks . . . 83

8 Appendices **84**

Appendices **84**

Bibliography **5**

Acronyms

DWA Dynamic Window Approach. 14, 32, 40, 41, 43, 52, 54–56, 58–61, 70, 78–83

E-Band Elastic Band. 34, 40

EM Expected Map (Average Map). 26, 27, 46, 51

EMMI Expected Map Mean Information. 26–28, 45, 46, 48, 50, 51, 53, 58, 65, 66, 71, 75, 80, 84–86

FMP Full Map Posterior. 21–23, 46, 47, 49, 50, 53, 56, 66, 68, 74, 83

GM GMapping. 10, 19, 20, 23, 41, 44, 45, 47, 51, 53, 60, 61, 65–68, 70, 71, 75, 79, 80

ISR Institue For Systems and Robotics Lisboa. 73, 75, 77

MPC Model Predictive Control. 34, 40

OGM Occupancy Grid Map. 20, 21, 29, 41, 42, 46, 60, 66, 68

PBF PoseBoost cost function. 10, 41–43, 52–58, 60, 61, 63, 70, 78–82, 84

RBPF Rao-Blackwellized Particle Filters. 18–20, 23, 26, 27, 36, 44, 45

ROS Robot Operating System. 19, 29, 30, 40, 56, 59, 68, 74, 87

RViz ROS Visualization. 29, 51, 63, 64, 67, 68

SLAM Simultaneous Localization and Mapping. 12, 14, 15, 17–23, 35–37, 39, 41, 44, 45, 65, 67, 68, 70, 80, 82

TEB Timed Elastic Band. 34, 40

List of Figures

1 Overview of the ROS navigation stack. 30

2 Interaction between the different components 38
3 Flowchart of the EMMI entropy calculation process 46
4 FMP vs. Shannon Entropy 49
5 Low Expected Map Mean Information Visualization 51
6 High Expected Map Mean Information Visualization 51
7 Overview - PoseBoost cost function (PBF) Impact 53

8 DWA Overview 59
9 DWA Local Planner: Classes involved in the PoseBoost Cost Function . . 60
10 DWA: RViz visualizer 64
11 DWA: Cell visualizer 64
12 GMapping Overview 65
13 New GMapping Functions 66
14 GMapping: Particle Publisher 67
15 Averaged Map Publisher in GMapping (GM) 68

16 Gazebo Simulation – Restaurant Environment 72
17 Simulation Environment – Subway Station 73
18 ISR Testbed 73
19 Position Error - Simulation 76
20 Strategy comparison - Simulation 76
21 Position Error - real-world 77
22 Strategy comparison - real-world 77

23 Rotational Error - Real World 85
24 EMMI - Real World 85
25 Rotational Error - Real World 86

26 EMMI - Simulation 86

List of Algorithms

1 Particle Filter Algorithm 19
2 Simplified ROS Navigation Stack Loop 31
3 Dynamic Window Approach (DWA) Planner with Full Cost Functions . . 33

List of Configurations

8.1 `local_planner.yaml` 87
8.2 `move_base.yaml` 88
8.3 `gmapping.yaml` 88
8.4 Output of `Particle and Map Service` 89

1 Introduction

[Rob says: "Dear Reviewer,
thank you for dedicating your time to review my first Master's thesis. I appreciate any constructive feedback or any thoughts you might have while reading.
Please note that this is not the final version. Also tests in simulation are still ongoing, so the results presented are just placeholders.
Thanks again :-*"]

[PL says: "you never leave clear at some point in the text the exact method you are using, with all the steps of the algorithm, the cost function of the modified DWA, and the calculation formulas o the term you added to check the entropy of the reached alternative locations. Do that!"] [PL says: "related to the previous item, sometimes you dive into different alternatives and it does not result clear whether this was just for comparison or which one you have used."]

1.1 Motivation

Simultaneous Localization and Mapping (SLAM) is a foundational problem in robotics, enabling autonomous systems to operate in previously unknown environments by estimating both their pose and a map of the surroundings. Traditionally, this task is treated as a standalone objective. However, in scenarios where a robot must navigate to a specific goal position in an unmapped area, a second imperative arises: to reach the target efficiently without compromising pose accuracy.

This dual objective—balancing precise pose estimation with effective goal-directed navigation—presents significant challenges. Achieving this balance requires intelligent decision-making to prioritize when to focus on pose improvement and when to advance toward the goal.

The genesis of this research was inspired by challenges encountered during RoboCup@Home (see RoboCup@Home [2024]), a prominent platform for advancing service robot capabilities in domestic environments. One key scenario is the *Restaurant Task* [Hart et al., 2024], where robots must serve clients in an unfamiliar environment relying solely on real-time data, without the benefit of a preexisting map. Competing teams often encounter critical issues, such as navigation failures or difficulties returning to the designated start location, due to inaccuracies in pose estimation.

Beyond research competitions, this problem extends to everyday life. As robots increasingly assist with household tasks—delivering items, cleaning, or supporting elder care, failures in mapping or localization can undermine their usefulness and reliability. Ensuring that robots can navigate safely and efficiently in new or changing environments is essential not only for the *Restaurant Task*, but also for broader integration of robotics into homes, hospitals, and public spaces.

Motivated by these limitations of existing autonomous systems, this thesis focuses on enhancing robotic navigation and localization strategies without requiring additional or specialized hardware. By strategically balancing movement toward the goal with selective re-localization actions, robots can achieve higher robustness and a lower risk of navigation failure, crucial capabilities for improving autonomous systems.

1.2 Research Objectives

Our research estimates the pose certainty in real-time to dynamically balance the two objectives and adjust the robot's path accordingly, enhancing both the robustness and reliability of autonomous systems in complex environments. Such advancements are crucial for applications where robots must operate independently in unknown areas, such as in disaster response, space exploration, and service robotics.

The overall goal of this thesis is to refine the capabilities of autonomous robots navigating accurately to a goal in unknown environments. To achieve this, the following specific research objectives have been identified:

1. **Pose Estimation Enhancement via a local Planner:**
   Modify the local planner to effectively improve the certainty of the pose, particularly in unknown or dynamic environments.

2. **Real-Time Pose Accuracy Measurement:**
   Develop and integrate real-time metrics to assess the accuracy of the estimation of the pose. This metric will facilitate signaling when strategic relocalizations are essential to maintain navigational precision.

3. **Balanced Navigation Strategy:**
   Create an algorithmic enhancement to balance the dual imperatives of reaching navigational goals and enhancing pose accuracy. The enhancement will allow adaptive trajectory adjustments based on real-time pose accuracy evaluations, ensuring optimal navigation decisions that respond to environmental uncertainties.

1.3 Contributions

This thesis aims to advance the field of robotics and Simultaneous Localization and Mapping by addressing key challenges in autonomous navigation through enhanced pose estimation, mapping accuracy, and adaptive planning. The primary anticipated contributions are as follows:

- **Enhanced SLAM-GMapping Algorithm:**
  Introducing key extensions to SLAM-GMapping, an implementation of the Fast SLAM 2.0 algorithm for ROS, to strengthen pose certainty estimation. These enhancements incorporate advanced mapping techniques to achieve a more reliable pose estimation and more accurate environmental representation.

- **Dynamic Window Approach Planner Enhancement:**
  Enhancing the Dynamic Window Approach (DWA) planner by including a new critic that actively promotes revisiting actions and loop closures to enhance the pose estimate. This modification aims to dynamically balance the trade-off between efficient path following and necessary pose corrections, based on a real-time pose certainty estimation, thus enabling the robot to maintain high pose accuracy while efficiently reaching its goals.

- **Experimental Validation and Practical Applications:**
  The proposed modifications and methodologies will be validated through comprehensive experiments in both simulated and real-world environments. The experiments are expected to demonstrate enhancements in navigation accuracy, a reduction in

pose entropy, and the effectiveness of entropy-based decision-making within the local planning process.

1.4 Thesis Structure

The thesis is organized in eight chapters. Here is a brief overview of each chapter:

- **Chapter 1: Introduction**
  This chapter presents the motivation, objectives, and contributions of the research. It outlines the real-life problem that inspired the study and introduces the structure of the document.

- **Chapter 2: Background**
  Provides technical details on the foundational concepts and technologies used in the research, including Simultaneous Localization and Mapping principles, the Dynamic Window Approach, and entropy metrics.

- **Chapter 3: State-of-the-Art**
  Reviews the existing literature and previous research related to this study, approaches trying to combine navigation with SLAM and applied strategies in the RoboCup@Home problem establishing the context and background for the innovations proposed in this thesis.

- **Chapter 4: Adaptive Local Planning for Improved Pose Certainty in Active SLAM**
  This chapter serves as the core of the thesis, detailing the strategies, concepts, and algorithms developed to balance goal-directed navigation with pose improvement. It provides an in-depth explanation of the methods used, the rationale behind their design, and the dynamic decision-making processes.

- **Chapter 5: Implementation**
  Details the modifications made to the existing framework and navigation components to implement the concepts discussed in the previous chapter, along with the addition of new functionalities.

- **Chapter 6: Experiments and Evaluation**
  Presents the experimental setup, methodology, and results of the testing performed in both simulated and real-world environments, providing a critical analysis of the system's performance.

- **Chapter 7: Conclusion and Future Work**
  Summarizes the findings of this thesis, discusses the implications of the research, and suggests areas for further study and potential improvements.

- **Chapter 8: Appendices**
  Contains supplementary material not included in the dissertation body text.

2 Background

This chapter provides a brief introduction to the various algorithms, papers, and software packages used in our research. The focus is on summarizing the essential aspects of each method used, to establish the necessary notation, and to ensure the document is self-sufficient.

2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics and computer science. It involves enabling a robot or autonomous vehicle to simultaneously map an unknown environment while estimating its pose within that environment. This dual challenge is crucial for autonomous navigation in various applications, including self-driving cars, unmanned aerial vehicles, and mobile robots operating in dynamic environments.

SLAM algorithms can be generally divided into two main categories: passive SLAM and active SLAM [Cadena et al., 2016].

In traditional passive SLAM, the robot's task is to map and localize within an environment using available sensor and odometry data, following a pre-defined or operator-controlled trajectory [Ahmed et al., 2023]. The focus in passive SLAM is to accurately estimate the pose of the robot and to build a map without automatically influencing the movement strategy to improve these estimates.

Active SLAM, by contrast, integrates decision-making mechanisms that allow the robot to autonomously control its path to achieve specific goals, such as exploring unknown regions or minimizing uncertainty in its pose or map estimates. According to Thrun et al., active SLAM leverages information gathering strategies to make navigation decisions that improve both localization accuracy and map quality by balancing exploration and exploitation.

Most research relies on information-theoretic approaches, such as maximizing the expected information gain or minimizing the expected uncertainty [Ahmed et al., 2023]. These methods use metrics such as entropy or mutual information to quantify the value of potential actions. For example, a common approach is to use the Bayesian information criterion to evaluate and select the most informative actions [Sim and Roy, 2005].

2.2 Particle Filters

Sequential Monte Carlo methods, commonly referred to as Rao-Blackwellized Particle Filters (RBPF), are extensively employed in SLAM to perform probabilistic estimation of states. RBPFs represents the posterior distribution of the robot's state (position and orientation) using a set of weighted particles. Each particle represents a potential state hypothesis, and the weights correspond to the likelihood of the observed data given that state. Each particle possesses a pose trajectory and a map estimate [Thrun et al., 2005].

The algorithm, as visualized in Algorithm 1, proceeds through three main steps [Stachniss, 2012b]:

1. **Prediction**: The particles are propagated according to the motion model, incorporating the robot's control inputs and noise to simulate motion uncertainty.

2. **Update**: The weights of the particles are updated on the basis of the likelihood of the observed sensor data, given each particle's predicted state. This typically involves computing the measurement likelihood using a sensor model.

3. **Resampling**: Particles with low weights are discarded and new particles are sampled from the high-weight particles, allowing the filter to focus on the most probable states.

RBPFs are particularly well-suited for SLAM as they can represent complex, non-Gaussian distributions and handle multi-modal distributions, which are common in real-world environments [Doucet et al., 2000].

Algorithm 1 Particle Filter Algorithm

1: **Input:** Particle set $\mathcal{X}_{t-1} = \{x_{t-1}^{[i]}, w_{t-1}^{[i]}\}_{i=1}^{N}$, control input u_t, sensor observation z_t
2: **for** $i = 1$ to N **do**
3: **Prediction:** Propagate each particle $x_{t-1}^{[i]}$ according to the motion model:

$$x_t^{[i]} \sim p(x_t | x_{t-1}^{[i]}, u_t)$$

4: **end for**
5: **for** $i = 1$ to N **do**
6: **Update:** Update weight $w_t^{[i]}$ of particle based on the likelihood of observation z_t:

$$w_t^{[i]} = p(z_t | x_t^{[i]})$$

7: **end for**
8: **Resampling:** Normalize the weights $w_t^{[i]}$ and resample N particles from the current set, favoring particles with higher weights.
9: **Output:** Updated particle set $\mathcal{X}_{t-1} = \{x_t^{[i]}, w_t^{[i]}\}_{i=1}^{N}$

2.2.1 Fast SLAM 2.0

While particle filters handle the global state estimation, it is often necessary to maintain and update feature-level information within the environment. To address this, methods commonly incorporate variants of the Extended Kalman Filter (EKF), a well-known estimator used for nonlinear state estimation[Ribeiro and Ribeiro, 2004].

One notable approach that combines RBPF for pose estimation with EKFs for landmark mapping is FastSLAM 2.0 [Montemerlo and Thrun, 2007]. This method factors the SLAM problem into separate components, robotic trajectory estimation and feature-based mapping, updated incrementally and efficiently. By maintaining EKFs for individual map features within each particle, it achieves computational tractability even in large-scale, complex environments.

2.2.2 GMapping

The package GM is a widely used RBPF-based SLAM implementation for Robot Operating System (ROS) (see section 2.7). Drawing on principles similar to those in approaches like FastSLAM 2.0, GM uses particle filters for the robot trajectory estimation and integrates feature-based updating techniques into an occupancy grid mapping framework.

This setup enables a robot to create a 2D Occupancy Grid Map (OGM) (see section 2.3) of its environment while simultaneously estimating its pose.

Designed to work with laser range finders and odometry data, GM is suitable for a wide range of robotic platforms. Its robustness, computational efficiency, and accuracy—arising from these probabilistic filtering techniques—have made it a popular choice for SLAM in various applications.

2.3 Map Representation

In SLAM, several types of map representation are commonly used, each suited to different application needs. These include grid cell maps, feature-based maps, and topological maps [Thrun et al., 2005].

2.3.1 Occupancy Grid Map

A prevalent map representation used in RBPF is the OGM. An OGM represents the environment as a grid of cells, where each cell stores a probability value that indicates the probability that the cell is occupied by an obstacle. This approach enables efficient storage and retrieval of spatial occupancy information, facilitating real-time updates and integration with sensor data.

The map is constructed by dividing the environment into a uniform grid, with the state of each cell (occupied or free) estimated using sensor measurements (e.g., laser scans) and the robot's pose. The probability of a cell c_i being occupied, denoted $p(c_i = \text{occ} \mid z_{1:t}, x_{1:t})$, is calculated based on observations $z_{1:t}$ and robot poses $x_{1:t}$ up to the current time t.

Occupancy Probability Calculation

The probability of occupancy for each cell can be computed using the recursive Bayesian update [Stachniss, 2012a]:

$$p(c_i \mid z_{1:t}, x_{1:t}) = \frac{p(z_t \mid c_i, x_t) p(c_i \mid z_{1:t-1}, x_{1:t-1})}{p(z_t \mid z_{1:t-1}, x_{1:t})} \tag{1}$$

where:

- $p(c_i \mid z_{1:t}, x_{1:t})$ represents the posterior probability indicating that the cell c_i is occupied, given all measurements $z_{1:t}$ and poses $x_{1:t}$ up to time t.
- $p(z_t \mid c_i, x_t)$ is the likelihood of observing z_t given the pose of the robot x_t and the occupancy state of c_i.
- $p(c_i \mid z_{1:t-1}, x_{1:t-1})$ is the prior probability of occupancy for c_i based on previous observations and poses.
- $p(z_t \mid z_{1:t-1}, x_{1:t})$ is a normalizing factor to ensure that the probabilities sum up to 1.

To facilitate efficient computations, the log-odds representation is often used. Defining the log-odds of occupancy l_{ij} as:

$$l_{ij} = \ln\left(\frac{p(c_i = \text{occ} \mid z_{1:t}, x_{1:t})}{1 - p(c_i = \text{occ} \mid z_{1:t}, x_{1:t})}\right) \tag{2}$$

The update rule can be simplified to:

$$l_{ij} = l_{ij} + \ln\left(\frac{p(z_t \mid c_i = \text{occ}, x_t)}{p(z_t \mid c_i = \text{free}, x_t)}\right) \tag{3}$$

where:

- l_{ij} is the updated log-odds of occupancy for cell c_i.
- $p(z_t \mid c_i = \text{occ}, x_t)$ and $p(z_t \mid c_i = \text{free}, x_t)$ are the probabilities of observing z_t given the robot's pose x_t and the occupancy or free state of c_i.

The log-odds update method simplifies the occupancy calculation and is widely used due to its computational efficiency.

The simplicity and efficiency of OGMs make them a standard choice for grid-based SLAM applications, as they support straightforward integration of sensor data and effective representation of occupied and free areas in the environment [Thrun et al., 2005, Elfes, 1989].

2.3.2 Full Map Posterior Maps

An advanced map representation, as introduced by Luft et al. [2017], uses Full Map Posterior (FMP). Whereas traditional OGMs only indicate the probability of a cell

being occupied, FMP leverages state-of-the-art beam-based lidar models to provide full posterior distributions about the certainty of the belief.

FMPs aim to determine the full posterior distribution over all possible maps given the sensor data, rather than just computing the most likely map. This approach preserves information on the certainty of the map estimates, which is typically lost in traditional mapping methods. In the case of small structures or low map resolution, FMP maps can provide significantly more information. For example, when the probability of a cell being occupied is 50% and there are many observations, the likelihood of the cell being partially occupied is high.

To calculate the full posteriors, Luft et al. proposes two different models: the reflection model and the decay-rate model. Both models only need to store two parameters per cell and are able to create closed-form solutions for these posteriors without any significant additional computational costs, thus greatly improving the performance of SLAM algorithms. For simplicity, the following section introduces only the reflection model.

Reflection Model

The reflection model represents each cell in the map based on its probability of reflecting an incident laser beam. This approach models the probability of reflection μ_i of a given map cell i using a posterior distribution, capturing the uncertainty inherent in the mapping process.

The posterior distribution for μ_i is defined as a Beta distribution:

$$bel(\mu_i) \propto \mu_i^{H_i}(1-\mu_i)^{M_i} p(\mu_i), \tag{4}$$

where:

- H_i denotes the number of times a laser beam was reflected in cell i (hits),
- M_i represents the number of times the beam passed through the cell without reflection (misses),
- $p(\mu_i)$ is the prior distribution of the reflection probability.

If the prior $p(\mu_i)$ is modeled as $Beta(\alpha, \beta)$, the posterior distribution can be expressed as:

$$bel(\mu_i) = Beta(H_i + \alpha, M_i + \beta). \tag{5}$$

This formulation allows for closed-form computation of the posterior distribution, requiring no additional computational complexity compared to traditional maximum likelihood methods. This property makes it particularly suitable for real-time SLAM applications [Luft et al., 2017].

FMPMapping

Building upon the concepts introduced in Full Map Posterior Maps, a modified version of the GM package, known as `FMPMapping`, has been developed by Arce y de la Borbolla [2021]. This implementation integrates the FMP distributions, allowing a more accurate representation of the uncertainty of the environment.

2.4 Entropy in Particle Filters

Entropy measures uncertainty or randomness within a system. In the context of SLAM, entropy is used to quantify the uncertainty in both the map and the robot's pose. Lower entropy values correspond to higher certainty, whereas higher entropy values indicate greater uncertainty. For our approach, we want the robot to automatically decide where to go next, improving its localization, and moving to the goal. As an estimation to decide whether it is necessary to improve the pose certainty, we use the concept of entropy. This section provides an overview of how the entropy can be calculated in SLAM.

Calculation

Given that in a RBPF the posterior distribution is represented by a set of weighted particles, the integral calculation of the entropy can be transformed into a summation. The entropy $H(p(m, x_{1:t} \mid d_t))$ over the map m and the robot's trajectory $x_{1:t}$ given the odometry and laser-measurements d_t can be approximated by [Stachniss et al., 2005]:

$$H\left(p(m, x_{1:t} \mid d_t)\right) \approx H\left(p(x_{1:t} \mid d_t)\right) + \sum_{i}^{\#particles} \omega_t^{[i]} H\left(p(m^{[i]} \mid x_{1:t}^{[i]}, d_t)\right) \tag{6}$$

Here, $\omega_t^{[i]}$ represents the weight of the i -th particle at time step t.

According to the principle of Rao-Blackwellization, the overall entropy of the system can be divided into two components:

1. The map entropy associated with each particle's map, weighted by the likelihood of the corresponding trajectory.

2. The entropy of the posterior distribution over the robot's trajectory, $H(p(x1:t \mid dt))$.

2.5 Map Entropy Calculation

When using the traditional Occupancy Grid Maps the calculation of the map entropy is straightforward. The entropy $H(c_i)$ of a single cell c_i is calculated by:

$$H(c_i) = -p(c_i)logp(c_i)(-(1-p(c_i))log(1-p(c_i)) \tag{7}$$

where $p(c_i)$ is the probability that the cell i is occupied [Stachniss et al., 2005]. The formula is derived from Shannon's entropy, which quantifies the expected amount of information (or uncertainty) in the probability distribution.

Usually, each cell is treated as an independent binary random variable. The entire map entropy $H(m)$ can then be obtained by simply summing up the entropy values of all the cells:

$$H(m) = \sum_i (H(c_i)) \tag{8}$$

A high map entropy indicates that many cells in the grid have probabilities near 0.5, meaning there is significant uncertainty about whether these cells are occupied or free. Conversely, low map entropy suggests that most cells are either clearly occupied or clearly free, reflecting a high degree of certainty in the map.

2.5.1 FMP Map Entropy

For the reflection model, the posterior distribution of $bel(\mu_i)$ is represented by a Beta distribution (see Equation 5). The entropy $H(\mu_i)$, which quantifies the uncertainty in the reflection probability, is calculated as [Luft et al., 2018]:

$$H[\text{Beta}(\alpha,\beta)] = \ln[B(\alpha,\beta)] + (\alpha+\beta-2)\psi(\alpha+\beta) - (\alpha-1)\psi(\alpha) - (\beta-1)\psi(\beta) \tag{9}$$

Here, $B(\cdot, \cdot)$ refers to the Beta Function, and $\psi(\cdot)$ denotes the digamma function, the derivative of the natural logarithm of the gamma function. The total map entropy $H(m)$ for the entire grid map is then obtained again by summing the entropy values of all cells.

2.6 Pose Entropy

Pose entropy is a critical metric for understanding the robot's localization performance. However, computing the uncertainty $H(p(x_{1:t} \mid d_t))$ for the robot's trajectory $x_{1:t}$, given odometry and laser measurements d_t, is more complex, as each pose x_t depends on the previous locations of the trajectory $x_{1:t-1}$. A common approach is the approximation of the entropy considering only the last pose of the robot [Stachniss et al., 2005], thus approximating $H(p(x_{1:t} \mid d_t))$ by $H(p(x_t \mid d_t))$.

2.6.1 Approaches for Particle Filters

- **Number of Effective Particles:**
  The number of effective particles (N_{eff}) is a simple yet effective method to approximate how well the particles represents the posterior [Grisetti et al., 2005].

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (\omega_t^{[i]})^2} \tag{10}$$

  where $\omega_t^{[i]}$ represents the weight of the i-th particle at time step t. This approach is computationally efficient and provides a quick estimate of the uncertainty in the particle distribution. However, this method only takes into consideration the weight of the single particles, ignoring the differences in their positions or the information about their maps.

- **Weighted Average of Pose Entropies:**
  Roy et al. [1999] proposes another way of estimating the pose certainty by averaging over the uncertainties of the different poses (of different timesteps t) of the path:

$$H(p(x_{1:t} \mid d_t)) \approx \frac{1}{t} \sum_{t'=1}^{t} H(p(x_{t'} \mid d_t)) \tag{11}$$

where d_t are the odometry and laser measurements at timestep t. And in a second step taking a weighted average over all the particles (i):

$$H_{\text{avg}} = \sum_{i=1}^{N} \omega_t^{[i]} H\left(p(x_t^{[i]} \mid d_t)\right) \tag{12}$$

where $\omega_t^{[i]}$ is the weight of particle i.

- **Covariance Matrix-Based Entropy:**
  For Gaussian approximations of the particle distribution, the entropy can be calculated using the covariance matrix Σ. This method is straightforward and effective when the particle distribution is approximately Gaussian [Stachniss et al., 2005]. The entropy H for a n-dimensional Gaussian distribution $G(\mu, \Sigma)$ is given by:

$$H(G(\mu, \Sigma)) = \log\left((2\pi e)^{\frac{n}{2}} |\Sigma|\right) \tag{13}$$

  Here, Σ represents the covariance matrix, while n denotes the dimensionality of the pose space, and $|\Sigma|$ refers to the determinant of Σ.

2.6.2 Expected Map Mean Information

Expected Map Mean Information (Expected Map Mean Information (EMMI)) is a metric introduced by Blanco et al. to address challenges associated with traditional entropy calculations in the context of RBPFs.

Calculation

EMMI is derived from the concept of Expected Map (Average Map) (EM) which is a weighted average of all map hypotheses generated by particles in an RBPF. The key idea is that by averaging the maps, we capture inconsistencies between different map hypotheses as blurred areas, which represent uncertainty.

The averaged map ($\bar{m}$) is calculated taking the weighted average of the occupancy probabilities across all particles [Blanco et al., 2008]. For each cell c in the map, the average occupancy probability is calculated as follows:

$$\bar{m}(c) = \frac{\sum_{i=1}^{N} \omega_i \cdot m_i(c)}{\sum_{i=1}^{N} \omega_i} \tag{14}$$

where:

- N is the number of particles
- ω_i refers to the weight of the i-th particle (the likelihood of the pose of that particle and the map hypothesis)
- $m_i(c)$ is the occupancy probability of cell c in the map of the i-th particle.

In a second step its entropy is calculated:

$$\text{Mean Entropy}(\bar{m}) = \frac{1}{N_{\text{obs}}} \sum_{c \in \text{ obs}} (1 - H(\bar{m}(c))) \tag{15}$$

where:

- $H(\bar{m}(c))$ is the entropy of every cell of the EM which was at least observed from one particle
- N_{obs} the number of observed cells.

The entropy $H(\bar{m}(c))$ of a cell is computed using the standard Shannon entropy formula:

$$H(\bar{m}(c)) = -p(\bar{m}(c)) \log p(\bar{m}(c)) - (1 - p(\bar{m}(c))) \log (1 - p(\bar{m}(c))) \tag{16}$$

To calculate the information I, which represents certainty, the entropy is simply inverted.

$$\text{EMMI} = 1 - \text{Mean Entropy}(\bar{m}) \tag{17}$$

Eligibility Although EMMI does not consider the pose of the particles of RBPF, it approximates the pose entropy and is well suited to determine the need to improve the pose (see Figure 5 & 6):
By averaging map hypotheses, EMMI inherently penalizes particles that produce dissimilar maps, often indicative of incorrect pose estimates. Additionally, unexplored cells do not contribute to the calculation, ensuring that exploration of an unknown area does not affect the calculation.

Advantages

EMMI offers several advantages that enhance their suitability to estimate the need for pose improvement.

- **Efficiency:**
  The calculation of EMMI is computationally less intensive compared to the explicit calculation of the pose entropy for each particle over the entire trajectory. This efficiency makes EMMI feasible for real-time applications.

- **Stability:**
  Unlike traditional pose entropy, which can fluctuate significantly due to the resampling effects of particle filters, EMMI provides a more stable measure of uncertainty. This stability is achieved by incorporating map similarity across particles, ensuring that closely clustered particles do not artificially reduce the certainty.

- **Exploration and Resolution Independence:**
  The EMMI entropy calculation is independent of the resolution of the map and the extent of the mapped area, as it is normalized by the total number of observed cells. This normalization ensures consistent and reliable uncertainty measurement, even in large or sparsely explored environments.

- **Robustness in Unmapped and Open Areas:**
  EMMI handles unmapped and open areas more robustly than traditional pose entropy. In such areas, where environmental features are sparse, particle drift does not immediately decrease EMMI Information. This prevents premature revisiting actions, allowing the robot to focus on exploration until new features are detected. When the robot encounters new features, EMMI effectively incorporates these into the calculation, allowing better navigation decisions and ensuring that pose improvement actions are only triggered when truly beneficial.

[PL says: "move ros, ros nav stack and optitrack to the experimental setup chapter 5"]
[Rob says: "I think it is necessary to introduce the ros navigation stack and therefore ros as well. We need to know how the different planners interact whith each other. also we are introducing different local planners. this is essential for the reaasoning later, why we decided to modify the dwa planner and not another one. The optitrack definetely belongs to the experiental setup, you are right :-)"]

2.7 Robot Operating System

The ROS is a popular open-source framework, is widely used in robotics research and development. It was initially developed by the Willow Garage research lab, and provides a flexible architecture to build and control robotic systems. It offers a range of tools and libraries facilitating the creation of complex robot behaviors, from basic sensor integration to advanced autonomous navigation [ROS Wiki, 2024b].

ROS uses a modular design in which different parts of a robot's software stack are implemented as separate processes known as *nodes*. These nodes communicate with each other using a publish-subscribe messaging model, allowing asynchronous data exchange across the system. It also supports synchronous communication through *services* and *actions*, enabling nodes to perform remote procedure calls and execute long-running tasks with feedback, respectively [ROS Wiki, 2024a].

The key components of the ROS framework include:

- **Nodes**: Independent processes that perform specific functions within a robotic system.
- **Topics**: Named communication channels over which nodes exchange messages.
- **Services and Actions**: Mechanisms for synchronous and goal-oriented communication between nodes.
- **Packages**: The fundamental units of organization consisting of nodes, libraries, and configuration files.

RViz

ROS Visualization (RViz) is a visualization tool within the ROS framework, it can be used to display sensor data, state information, and other output of ROS nodes in a 3D environment. It offers an intuitive interface for real-time visualization of complex data sets, including laser scans, point clouds, OGMs, and robotic models.

2.8 ROS Navigation Stack

The ROS Navigation Stack is a widely used framework that enables autonomous navigation for robots [MoveBase Wiki, 2024]. It provides multiple modules for path planning, obstacle avoidance, and motion control, ensuring safe, goal-directed navigation. As

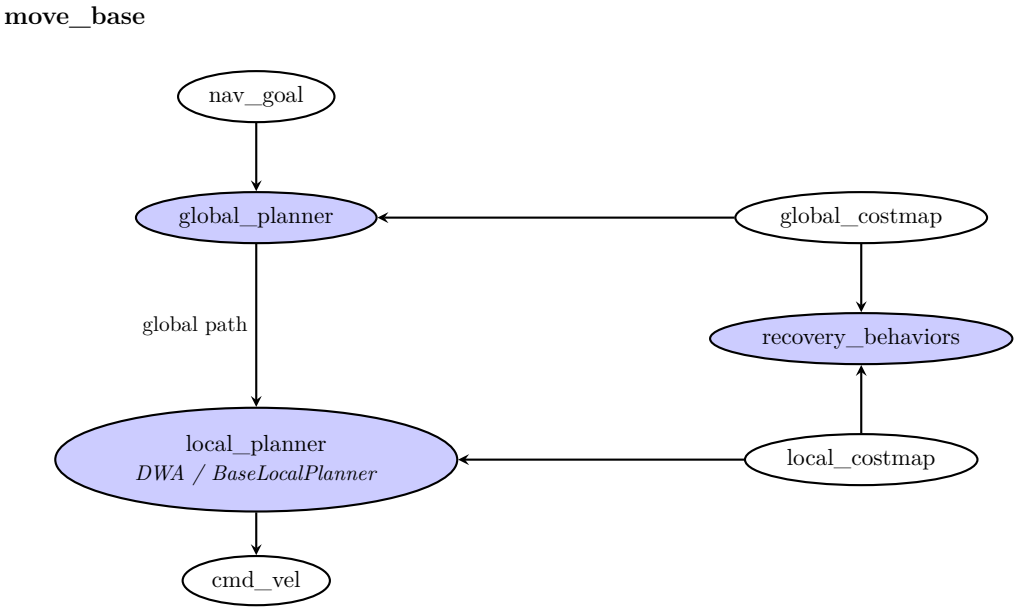

Figure 1: Overview of the ROS navigation stack.

shown in Figure 1, the navigation stack comprises the `global_planner`, `local_planner`, `costmaps`, `recovery_behaviors`, and other supporting components. Conceptually, it operates in a continuous loop rather than as a strictly linear workflow. The high-level process, outlined in Algorithm 2, starts when the robot is given a navigation goal (`nav_goal`). The `global_planner` computes an initial path using the static information in the `global_costmap`, while the `local_planner` iteratively refines the plan and generates velocity commands (`cmd_vel`) for the robot to follow. Should the path become blocked or no valid trajectory is found, the `recovery_behaviors` are triggered to unblock or reset the navigation process.

By repeating the local planning and recovery steps, the robot adjusts its trajectory until it successfully reaches the navigation goal. Such iterative loops are fundamental to reactive navigation frameworks like the ROS Navigation Stack, ensuring continuous adaptation to changing conditions in real-world scenarios.

2.8.1 Local Planners

Local planners are responsible for short-term navigation, considering the robot's motions constraints and immediate surroundings, ensuring fast reactions to unforeseen obstacles or changes in the environment. They are responsible for generating feasible and safe

Algorithm 2 Simplified ROS Navigation Stack Loop

Input: Navigation goal (`nav_goal`), robot's current position
Step 1: `global_planner` receives `nav_goal`
`global_planner` generates a high-level path from current position to goal using the `global_costmap`
while goal not reached **do**
    Step 2: `local_planner` updates detailed movement commands
    `local_planner` uses `local_costmap` to avoid dynamic obstacles
    if path is blocked or no valid trajectory found **then**
        Step 3: Invoke `recovery_behaviors`
          (e.g., rotate in place, clear costmaps, etc.)
        Retry or replan the path after recovery
    end if
    Step 4: `local_planner` sends new `cmd_vel` to robot's actuators
    Robot moves according to `cmd_vel`
end while
Output: Robot arrives safely at `nav_goal`

In this loop:

- **`global_planner`**: Calculates an initial path from the robot's current location to the goal based on static environmental data (e.g., walls, furniture).
- **`local_planner`**: Continuously refines the path, creating real-time velocity commands and adjusting for new obstacles or changes in the environment.
- **`recovery_behaviors`**: Called if the robot becomes stuck or the local planner cannot find a valid path, attempting to recover by resetting costmaps or rotating in place.
- **`cmd_vel`**: The velocity command messages controlling the robot's linear and angular velocities.

paths that guide the robot towards its goal while avoiding obstacles. A local planner operates in real-time and is beside the global map also using sensor data to dynamically adjust the robot's trajectory.

1. Dynamic Window Approach Planner

The DWA is a widely utilized local planner in mobile robotics, valued for its ability to avoid collisions efficiently while maintaining real-time performance. Originally introduced by Fox et al. [1997], it has become a foundational approach for local navigation in robotic systems.

The core principle of DWA lies in evaluating the robot's feasible velocities, both translational and rotational, within a constrained time frame. A significant advantage of this method is its consideration of the robot's dynamic properties, such as velocity and acceleration limits, during trajectory planning. By restricting the search space to a "dynamic window" of feasible velocities, the planner ensures real-time selection of safe trajectories tailored to the robot's current state and capabilities.

A key feature of the DWA is that the trajectories evaluated during planning are arcs of circles, determined by the relationship:

$$r = \frac{v}{\omega},$$

where r is the radius of the trajectory, v is the linear (tangential) velocity, and ω is the angular velocity. This representation aligns with the robot's motion constraints, making it particularly effective for generating smooth and natural trajectories in real-world environments.

As illustrated in Algorithm 3, the DWA planner evaluates short-term trajectories, enabling smooth and reliable navigation even in cluttered or dynamic environments. By considering the robot's motion constraints and representing trajectories as arcs of circles, the planner generates feasible paths that adhere to the robot's physical limitations. Extensive experiments conducted in various scenarios have consistently demonstrated the DWA planner's robustness and reliability, particularly for high-speed, short-range navigation tasks.

Algorithm 3 Dynamic Window Approach (DWA) Planner with Full Cost Functions

Input: Robot's current state (position, velocity), navigation goal, `local_costmap`

Step 1: Search Space Restriction

Limit velocities to the *dynamic window*, including only velocities achievable within a short interval based on the robot's current state and acceleration limits.

Step 2: Admissible Velocities

Filter velocities to include only those that allow the robot to stop safely before encountering obstacles, ensuring collision-free navigation.

Step 3: Cost Function Evaluation

for each admissible trajectory tr_i in the dynamic window **do**

Compute cost $c_{\text{tot}}(tr)$ as the weighted sum of individual cost functions:

$$c_{\text{tot}}(\text{tr}_i) = \omega_1 \cdot \text{cf}_1(\text{tr}_i) + \omega_2 \cdot \text{cf}_2(\text{tr}_i) + ... \tag{18}$$

end for

Step 4: Optimization

Select the velocity command pair $(v^\star, w^\star)$ that minimizes the total cost:

$$(v^\star, w^\star) = \underset{(v,w)\in\mathcal{V}_{\text{feasible}}}{\arg\min} \; c_{\text{tot}}(\text{tr}(v, w)) \tag{19}$$

Here, $\mathcal{V}_{\text{feasible}}$ represents the set of velocity commands permissible under the robot's kinematic and dynamic constraints.

Step 5: Output: Execute the selected trajectory $tr(v^\star, w^\star)$ by generating `cmd_vel` commands to move the robot.

All cost functions used for the evaluation of a trajectory are (see source code):

- $c_{\text{osc}}(\text{tr}_i)$: Penalizes oscillatory motions (assigns cost -1 for oscillations).
- $c_{\text{obs}}(\text{tr}_i)$: Discards trajectories that move into obstacles or get too close to them.
- $c_{\text{gf}}(\text{tr}_i)$: Prefers trajectories that align the robot's front (nose) toward the local goal.
- $c_{\text{al}}(\text{tr}_i)$: Prefers trajectories that keep the robot aligned along the local path (nose on path).
- $c_{\text{path}}(\text{tr}_i)$: Encourages trajectories that follow the global path.
- $c_{\text{g}}(\text{tr}_i)$: Rewards trajectories that progress toward the local goal, based on wave propagation.
- $c_{\text{tw}}(\text{tr}_i)$: Optionally discourages spinning motions in trajectories.

2. Elastic Band (E-Band) Planner

The E-Band Planner as proposed by Quinlan and Khatib [1993] optimizes a robot's path by representing it as a series of interconnected elastic bands that adjust dynamically based on environmental constraints. As the robot progresses, these bands "shorten" by pulling the path towards the goal while simultaneously expanding to avoid obstacles. The E-Band planner is simple and able to generate smooth and adaptable paths.

3. Timed Elastic Band (TEB) Planner

Rösmann et al. [2015] introduced a modified version of the classic E-Band planner by incorporating time as an additional dimension in the path optimization process. This capability allows the TEB planner to optimize not only the geometric shape of the path but also the timing of the robot's movements. Consequently, the planner facilitates more efficient navigation, especially in dynamic environments where precise timing plays a critical role. To accomplish this, the TEB planner minimizes a cost function that integrates various objectives, such as obstacle avoidance, trajectory smoothness, and additional relevant criteria. By balancing these factors, the planner produces trajectories that prioritize both safety and efficiency.

4. Model Predictive Control (MPC)

MPC or Receding Horizon Control a as proposed by Mayne et al. [2000] optimizes the robot's trajectory over a finite time horizon, solving a control problem at each time step. It considers the robot's dynamics, constraints, and predicted future states, generating smooth, feasible trajectories. It uses a cost function that balances goal progress, control effort, etc.

3 State-of-the-Art

This chapter reviews current research and methodologies related to Active Simultaneous Localization and Mapping (see section 2.1), particularly in the context of integrating navigation with real-world tasks, such as those presented in the RoboCup@Home competition (see section 1.1).

3.1 Active SLAM Approaches

In recent years, Active SLAM has received significant attention; a comprehensive review of recent and notable studies can be found in *Active SLAM: A Review on Last Decade* [Ahmed et al., 2023].

One of the pioneering contributions in this field was made by Stachniss et al. [2005], who developed a decision-theoretic framework that allows robots to evaluate potential actions by balancing the costs of execution against the expected information gain. This approach facilitates actions that enhance the quality of the map while reducing pose uncertainty, exemplifying the integration of exploration, mapping, and localization strategies.

Although numerous studies have explored various aspects of Active SLAM in recent years, few have directly addressed the specific problem of integrating active exploration with goal-directed navigation. A notable contribution in this domain is presented by Chaplot et al. [2020], who introduced a neural framework that combines learned modules with classical path planning. Their focus on maximizing exploration efficiency is commendable; however, it primarily emphasizes exploration over navigation towards specific goals.

In the context of improving exploration strategies, the work by Zhelo et al. [2018] offers valuable insights. This paper investigates curiosity-driven exploration for mapless navigation using Deep Reinforcement Learning (DRL). The authors propose augmenting traditional extrinsic rewards with intrinsic rewards derived from the agent's curiosity, enabling it to explore more effectively in environments without predefined maps. Although the focus is on mapless navigation, the principles of intrinsic motivation to encourage

exploration can inform strategies in SLAM, particularly in balancing exploration and goal-oriented navigation.

3.2 Approaches in RoboCup@Home

In this context it is important to examine approaches that have been specifically designed to address the challenges of navigation in unknown environments. The RoboCup@Home competition provides a unique benchmark for evaluating these approaches, particularly with the restaurant scenario (see section 1.1), which involves tasks requiring robust navigation and high localization accuracy. Therefore, comparing the navigation and localization approaches used by RoboCup teams offers valuable insight into the effectiveness and real-world applicability of traditional navigation strategies and their limitations.

A common approach among competing teams in the restaurant scenario is to rely heavily on odometry for navigation. Although this method simplifies initial movement and positioning, it can accumulate errors over time, especially in environments with obstacles or uneven terrain. These errors lead to disorientation, undermining the robot's ability to return to starting positions accurately, which is a critical failure in tasks requiring precise localization.

On the other hand, teams employing SLAM algorithms face distinct challenges that need to be addressed to improve reliability. For instance, RBPF, commonly used in SLAM-based systems, can experience instability during the resampling phase. This process, though essential for computational efficiency, may cause sudden shifts in the robot's estimated pose. These instabilities can be particularly problematic in high-stakes tasks, such as those in the restaurant scenario, where accurate localization is necessary to fulfill the task.

Thus, examining both odometry-based and SLAM-based approaches in the RoboCup-@Home context highlights the critical need for solutions that can seamlessly integrate navigation with pose improvement.

3.3 Towards a Combined Approach

The challenges faced by both odometry-based and SLAM-based approaches underscore the necessity for a more integrated strategy.

Although significant progress has been made in Active SLAM, it is usually considered as an isolated problem from navigation. This gap presents a significant opportunity for further investigation in the simultaneous need for effective and reliable navigation in unexplored areas (see section 1.1).

4 Adaptive Local Planning for Improved Pose Certainty in Active SLAM

In the following sections, we detail the methodology and approach used to address our main goal of dynamically balancing our two objectives of an effective goal-directed navigation while maintaining a good pose estimate.

To accomplish this, we need to solve three challenges:

1. The improvement of pose estimation
2. An accurate pose certainty estimation
3. Balancing Goal-Directed Navigation and Pose Improvement

The following graphic visualizes how the different components work together in order to achieve the overall goal.

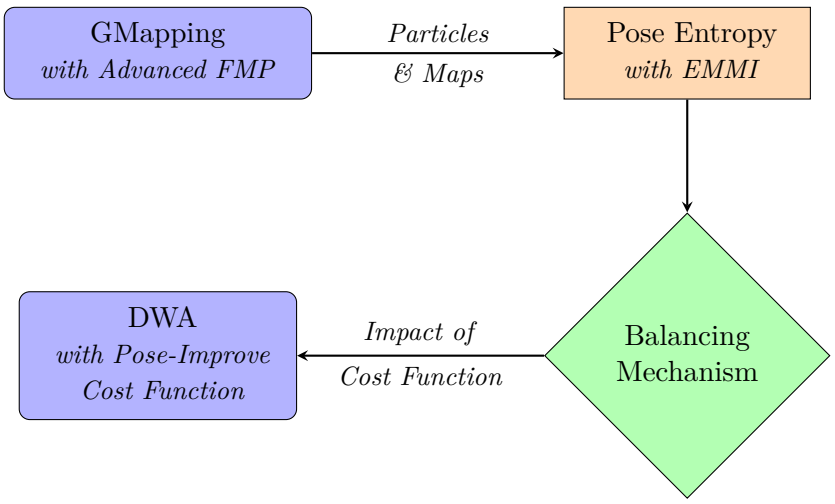

Figure 2: Interaction between the different components

[PL says: "I do not understand. If there is a tradeoff, it must be between 2 alternatives. But there are no alternatives here, apparently. You could have an arrow directly from "Pose Entropy with EMMI" to "DWA(...)""] [Rob says: "You are absolutely right, Trade-Off is not the corret term. Still there is a whole mechanism which does not only

calculate the weight parameter of pbf dynamically but also tries to identify oscillation and also the walked distance after oscillation. The code to accomplish that does not only lie in dwa but also in move_base"]

4.1 Pose Improvement

The first challenge is to enhance the robot's pose estimation accuracy. Within the SLAM framework, several strategies are identified to achieve this, such as:

- Revisiting previously mapped areas,
- Implementing loop closure techniques,
- Minimizing movement through unknown or unexplored spaces.

Controlling the robot's movement through modifications to the navigation system allows these strategies to be effectively implemented.

4.1.1 Strategies for Pose Improvement

The ROS Navigation Stack provides two approaches for altering the robot's movement: by adjusting either the local or the global trajectory. In the following, we will explore the advantages of each approach in detail.

Global Approach A global approach to pose improvement involves altering the overall trajectory to achieve higher-level navigation goals, either by modifying the global planner or by setting intermediate goals that guide the robot along a more desirable path.

The advantages of this method include planning with foresight, which optimizes the entire trajectory and reduces the risk of the robot getting trapped in local minima. However, the global approach has drawbacks. Since the SLAM algorithm continuously updates the map, assessments of whether a movement revisits known areas or explores unknown space can change dynamically. Furthermore, the pose entropy, representing the uncertainty in the robot pose, evolves as the map updates. Consequently, the need for revisiting actions varies dynamically, but global plans typically do not update frequently enough to capture these changes, leading to potentially suboptimal decisions.

Local Approach Given the limitations of a global approach, this work adopts a more adaptable local strategy. The main advantage of this approach is its responsiveness to rapid environmental changes and variations in pose uncertainty, allowing the robot to adapt its path based on the most recent map updates.

Implementing this local strategy requires modifications to the local planner responsible for generating real-time trajectories. In the ROS framework (see section 2.7), several local planners are available (see subsection 2.8.1), including:

- TEB (Timed Elastic Band),
- E-Band (Elastic Band),
- MPC (Model Predictive Control),
- and Dynamic Window Approach (DWA) planners.

Comparison of Local Planners

- The **TEB and E-Band** models represent the robot's trajectory as an elastic band, which dynamically extends to avoid obstacles, maintaining a smooth navigation. These planners heavily rely on the global plan, meaning that revisiting actions conflicts with their core design principles. Implementing an uncertainty-based decision-making approach in this planner would require substantial modifications to the algorithms, which may compromise their effectiveness.

- **MPC** computes optimized trajectories by addressing control problems at every time step, considering the dynamic model's constraints. Despite its effectiveness, MPC usually demands significant computational resources and is difficult to adapt. Evaluating trajectories in real time for pose improvement is impractical with MPC due to high complexity and computational demands, making it less suitable for scenarios that need real-time adaptability.

- **DWA** evaluates potential trajectories within the robot's velocity space and integrates various cost functions. This modularity makes it well suited to include new metrics for assessing pose improvement potential. The DWA planner's ability to dynamically adjust cost function weights allows it to respond to changing pose improvement needs based on real-time entropy estimates. Due to these advantages, this work focuses on modifying the DWA planner.

4.1.2 The new PoseBoost cost function

To incorporate pose improvement actions into the DWA planner, this work introduces a new cost function, the PBF. This function contributes to the overall cost c_{tot} by evaluating the potential of a local trajectory to improve the robot's pose estimate.

The updated calculation of the total cost c_{tot} for a local trajectory traj_i can be expressed as:

$$c_{\text{tot}}(\text{traj}_i) = \omega_{\text{pbf}} \cdot \texttt{PoseBoost}_{\text{cost}}(\text{traj}_i) + \omega_1 \cdot \text{cf}_1(\text{traj}_i) + \omega_2 \cdot \text{cf}_2(\text{traj}_i) + ... \tag{20}$$

Here, ω_x represents the weighting factors for individual cost functions.

While $\texttt{PoseBoost}_{\text{cost}}(\text{traj}_i)$ is the newly introduced term specifically designed to enhance pose improvement actions, the other terms $(\text{cf}_1, \text{cf}_2, \ldots)$ correspond to the original cost functions of the DWA planner. These include objectives such as moving toward the goal, obstacle avoidance, and following the global plan (see Equation 18), ensuring that the planner retains its foundational functionality. The dynamic calculation of ω_{pbf}, which balances pose improvement actions with the existing cost functions, is explained in section 4.3.

Methodology The primary objective of the PBF is to evaluate how much a local trajectory contributes to an improvement of the current pose estimate of the robot.

Generally, revisiting actions of mapped areas removes uncertain pose hypotheses and corrects accumulated drift, which enhances the accuracy of the pose estimate (see section 4.1). To quantify whether a trajectory explores or revisits, a ray-casting operation is performed at its endpoint, counting the number of known occupied cells in the OGM. Each ray is traced until it either intersects with an occupied cell or reaches the maximum range of the laser. Regions with a high number of occupied cells provide stable features, which are essential for accurate scan matching. When revisited, these features enable the SLAM algorithm (e.g. GM) to better align the current sensor data with the map, thus improving pose certainty. Trajectories leading to such areas with a high number of occupied cells are therefore assigned a lower cost, as they support localization accuracy and reliable navigation.

Ray-casting

For the ray-casting operations we would need to efficiently iterate over all the cells intersected by the laser rays. To do so, we decided to employ an adapted version of the line drawing algorithm originally proposed by Bresenham [1965].

While the traditional Bresenham algorithm typically identifies a single point per axis along the line, our modified version returns all the grid cells intersected by the ideal line throughout its path, ensuring a more accurate representation of the laser beams' interaction with the grid cells [Dedu, 2001].

Cost Calculation

The PBF is designed to seamlessly integrate with the base local planner by implementing the required cost function interface.

To generate a meaningful output, we opted to calculate a percentage representing the proportion of potentially occupied cells encountered during the ray-casting operation. This percentage is derived by dividing the observed number of occupied cells N_{occ} by the estimated maximum number of cells that could potentially be encountered along the ray-cast Max_{occ}:

$$\texttt{PoseBoost}_{\text{cost}}(\text{traj}_i) = \frac{N_{occ}}{\text{Max}_{occ}} \tag{21}$$

Estimation of Max_{occ} Since the exact calculation of Max_{occ} is complex and unnecessary for our purposes, we opted for a rough estimation.

To estimate the maximum number of grid cells intersected by a circle of radius r in an OGM with a resolution of res, we developed a perimeter-based heuristic formula. The perimeter of the circle, $2\pi r$, divided by the resolution of the grid cells, provides an approximate count of the cells along the edge of the circle:

$$\text{Estimated cells} \approx 2\pi \cdot \frac{r}{\text{res}} \tag{22}$$

For our application, this formula was adapted to account for the laser's angular range α. The angular range α represents the portion of the circle's perimeter covered by the

laser scan. In the limit case, when the laser covers the full 360 degrees, $\alpha = 2\pi$, and the estimation reverts to the original formula:

$$\text{Estimated cells} \approx \frac{\alpha}{2\pi} \cdot \frac{r}{\text{res}} \tag{23}$$

To ensure that the estimate Max_{occ} consistently exceeds the actual number of ray-casted cells, N_{occ}, a safety factor was introduced based on empirical testing. After extensive evaluation, a factor of 2.5 was deemed appropriate, leading to the final formula:

$$\text{Max}_{occ} \lesssim 2.5 \cdot \frac{\alpha}{2\pi} \cdot \frac{r}{\text{res}} = \frac{5\alpha \cdot r}{4\pi \cdot \text{res}} \tag{24}$$

Challenges in Safety and Trajectory Selection

Although the primary objective of the PBF is to encourage revisiting actions, testing revealed a potential issue in partially mapped areas. When the robot faces a mapped obstacle, trajectories directing the robot away from these obstacles often lead toward sparsely mapped or unmapped regions. In such cases, the ray-casting operation returns a low count of N_{obs} due to the lack of mapped features in these areas.

This scoring bias causes the PBF to inadvertently favor trajectories that keep the robot facing already mapped obstacles, preventing it from exploring unmapped regions. Consequently, the robot may stick to paths near obstacles, or even move closer to them, in an effort to maximize the count of observed cells N_{obs}.

This behavior poses a risk of unsafe maneuvers, as the robot may navigate dangerously close to obstacles. To mitigate this issue, we introduced the `pose_boost_min_distance` parameter in the DWA configuration. This safety parameter defines a critical boundary around the robot, ensuring that the proximity of obstacles is explicitly considered during trajectory evaluation. If PBF detects that a proposed trajectory violates this minimum distance, it assigns the highest possible cost to that trajectory, effectively preventing its selection.

This mechanism not only enhances the safety of the navigation process, but also ensures that pose improvement efforts do not compromise the primary goal of maintaining a safe operational boundary for the robot. Further details on how these parameters interact within the broader planning framework are discussed in section 4.3 (Balancing Goal-Directed Navigation and Pose Improvement).

4.2 Pose Entropy

An accurate and stable estimation of the pose entropy is essential to assess the reliability of the robot pose estimate, which allows dynamic balance between the goal-directed navigation and pose improvement actions.

4.2.1 Current Approach in GMapping

GMapping Overview

GM (see subsection 2.2.2), a widely used implementation of SLAM based on RBPF (see section 2.2), allows a robot to build a 2D map while estimating its pose. Multiple particles represent potential trajectories, each with an individual map estimate that is continuously updated through sensor inputs and odometry.

Baseline Pose Entropy Calculation

The standard method for calculating pose entropy in GM can be found in the source code (access here) and is expressed as:

$$H(p(x_{1:t}|d_t)) \approx -\frac{1}{N}\sum_{i=1}^{N} w_i \log(w_i) \tag{25}$$

where $H(p(x_{1:t}|d_t))$ represents the pose entropy, w_i is the normalized weight of the i-th particle, and N is the number of particles.

4.2.2 Limitations and Rationale for Improvement

The standard pose entropy calculation in GM is often performed where particle weights may not be properly updated, especially post-resampling. This can lead to inaccurately constant entropy values that do not reflect the true uncertainty in the pose estimate. Furthermore, relying solely on particle weights for entropy calculations overlooks the spatial distribution of the estimated poses. Particles with identical weights but significantly different spatial distributions should not result in the same entropy value. Specifically, particles that are far apart should produce a higher entropy, reflecting greater

uncertainty, compared to tightly clustered particles. Addressing these limitations is crucial to improving the reliability and accuracy of pose entropy calculations.

4.2.3 Review of Alternative Approaches

To address these limitations, several new methods were evaluated to improve pose entropy accuracy and stability without significantly increasing computational complexity (see subsection 2.6.1).

Identified Challenges

Some specific challenges addressed in the new approach include:

- **Temporal Fluctuations:**
  Resampling in RBPF, while crucial to maintaining computational efficiency, introduces significant temporal fluctuations in pose entropy calculations, causing inconsistencies that do not always represent reality.

- **Representation of Hypothesis Diversity:**
  Calculations based only on particle weights may overlook the spatial distribution of particles, potentially misrepresenting the diversity of hypotheses.

- **Computational Complexity:** More accurate entropy calculations involving full trajectory comparisons would increase computational demands, posing challenges for real-time applications.

Testing and Selection of EMMI

All the approaches mentioned in subsection 2.6.1 were implemented within the GM framework and tested in various scenarios to assess their performance in terms of precision, computational efficiency, and stability to represent the uncertainty of the pose.

The EMMI entropy (see subsection 2.6.2) emerged as the most effective approach, offering a balanced method with better accuracy in representing pose certainty and manageable computational requirements, suitable for real-time SLAM applications.

4.2.4 Expected Map Mean Information

The baseline approach to calculate EMMI entropy involves a sequence of steps, as shown in Figure 3, and begins with the use of a standard OGM. For a detailed introduction of EMMI please refer to subsection 2.6.2.

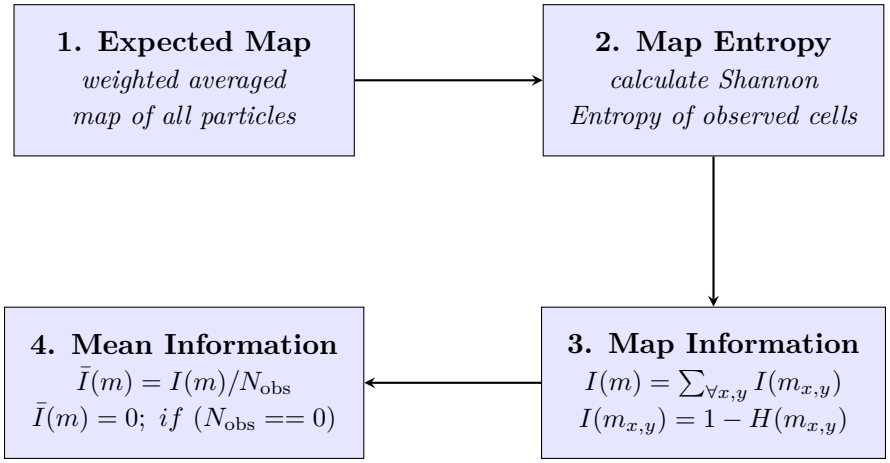

Figure 3: Flowchart of the EMMI entropy calculation process

The process includes:

1. Calculation of the Expected Map (EM), which is an average map based on the current particle weights.
2. Computation of the Shannon Map Entropy $H(m)$ of observed cells in the EM.
3. Conversion of entropy to map information $I(m)$, reflecting certainty rather than uncertainty.
4. Normalization of map information $I(m)$ by the number of observed cells N_{obs} to ensure independence from map resolution and extent of exploration.

Integration of Full Map Posterior

To tailor EMMI entropy to FMP requirements, the following modifications were implemented:

1. Rather than directly summing the weighted probabilities, we aggregate α and β values weighted by the confidence of each particle observing the cell. This ensures

that each cell's contribution is proportionate to the particle's confidence in its observations.

$$\bar{\alpha}_{x,y} = \sum_i w^i \cdot \alpha^i_{x,y} \tag{26}$$

$$\bar{\beta}_{x,y} = \sum_i w^i \cdot \beta^i_{x,y} \tag{27}$$

where $\bar{\alpha}_{x,y}$ and $\bar{\beta}_{x,y}$ represent the expected α and β values of one cell, respectively. Each $\alpha^i_{x,y}$ and $\beta^i_{x,y}$ are contributions of the particle i reflecting the hits and misses observed by that particle at the cell coordinates x, y.

2. After aggregation, normalize the values of α and β by the sum of weights that contribute to each cell, followed by the calculation of entropy using the FMP model (see subsection 2.5.1):

$$H(c_{x,y}) = H\left[\text{Beta}\left(\frac{\bar{\alpha}_{x,y}}{\bar{w}} + 1, \frac{\bar{\beta}_{x,y}}{\bar{w}} + 1\right)\right] \tag{28}$$

where $\bar{w}$ is the sum of weights of particles that have observed the current cell $c_{x,y}$.

3. The entropy of the Beta distribution asymptotically approaches negative infinity (see Figure 4), which requires normalization to ensure meaningful values.

   The Shannon entropy remains zero once a cell is either fully hit or fully missed, as no further uncertainty exists. However, this model does not account for partially observed cells, where uncertainty persists.

   In contrast, the FMP entropy captures the uncertainty associated with partially observed cells. Figure 4 illustrates how FMP entropy decreases continuously as a cell accumulates consistent observations (e.g., repeated hits). This divergence highlights the need for normalization to ensure that the entropy remains bounded and interpretable during runtime.

   An intuitive approach to normalization would involve using a gradient threshold. However, this method requires access to the previous entropy values of all cells, making it computationally impractical. As an alternative, we define a fixed threshold parameter, `entropy_threshold`, within GM, serving as a reference for absolute certainty.

Empirical observations show that the entropy $H[\text{Beta}]$ reaches -4 after approximately 150 consistent measurements. Therefore, the default value for `entropy_threshold` is set to 4. The normalized entropy is then computed as:

$$\bar{H}[\text{Beta}] = \max(H[\text{Beta}], -\texttt{entropy\_threshold}) \tag{29}$$

4. The worst entropy value for unobserved cells remains at 0. For observed cells:

$$\hat{H}(c_{x,y}) = \bar{H}(c_{x,y}) \cdot \bar{w} + 0 \cdot (1 - \bar{w}) \tag{30}$$

5. Transform expected Entropy $\hat{H}$ into information $\texttt{I}$, representing certainty as a probability. Hence we normalize the minimum entropy to a fixed value of -3 we can deduct the information in the following way:

$$\texttt{I}(c_{x,y}) = -\frac{\hat{H}(c_{x,y})}{3} \tag{31}$$

(DRAFT: also add pictures of the published particles in rviz together with the calculated pose entropy to proof :))

Accurate Local Representation

In many real-world scenarios, a robot's localization can degrade over time, especially as it traverses areas with fewer features or more dynamic obstacles. If the pose entropy is calculated over the entire map, previously well-localized areas may still contribute to a favorable entropy value, despite the uncertainty on the robot's current position increase due to the lack of local features. To prevent such misleading results and ensure a more accurate entropy estimation, the EMMI entropy calculations were restricted to a local radius r around the robot.

Limiting the entropy calculation to a smaller region around the robot is crucial for maintaining the accuracy of pose uncertainty. However, it is important to ensure that the radius is not too restrictive, as this could exclude important map areas, especially when the robot's particles are spatially distributed across a large area. A non-representative subset of the map could lead to skewed entropy values, which would undermine the accuracy of the system.

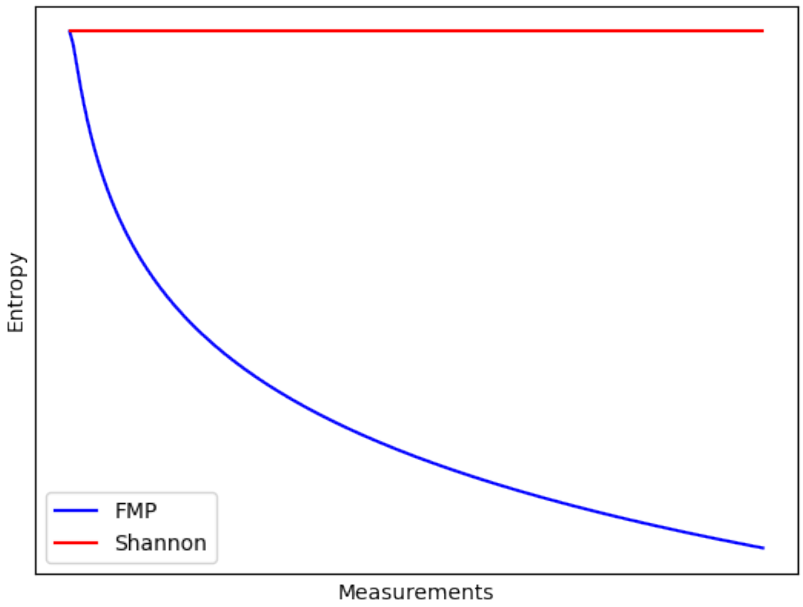

Figure 4: Comparison of FMP and Shannon entropy over consistent measurements. This graph compares the entropy development of the FMP model (blue curve) and the standard Shannon entropy (red curve). The x-axis represents the number of consistent measurements (e.g., repeated hits in a cell), and the y-axis shows the corresponding entropy values. While the Shannon entropy , appearing as an almost flat line, remains at zero once a cell is fully observed, FMP entropy decreases continuously, capturing the remaining uncertainty for partially observed cells.

To strike a balance between limiting the calculation area and ensuring that the entropy measure reflects the robot's actual pose uncertainty, we define the radius r based on two key parameters:

- the laser range l_r (the maximum distance at which the robot's sensors can measure the environment) and
- the update distance d_u (the distance the robot travels before a map update is triggered).

This radius is defined by the following formula:

$$r = 3 \cdot d_u + l_r \tag{32}$$

Incorporating both the laser range l_r and the update distance d_u in this way ensures that the entropy calculation remains relevant to the robot's immediate surroundings. [Rui says: "in my experience all these constants you are atributing numerical values and justifying through testing, usually a letter is assigned to this parameter and in the beggining of the results the values of these parameters are shown and explained that they were obtained empirically"] The factor of 3 in this formula was determined empirically based on the robot's typical particle distribution (obtained by Extended Functionalities in GMapping) and its behavior observed during tests. It ensures that the entropy calculation includes enough of the surrounding environment to account for relevant changes in the robot's trajectory and pose, without being influenced by distant map areas that no longer reflect the robot's current state.

While limiting the calculation to a local radius improves pose uncertainty estimation, it also results in a computational benefit, as fewer areas need to be processed.

Stability with Increased Measurements

In scenarios where a map region has been extensively surveyed with consistent measurements, the Full Map Posterior approach tends to maintain a high Information (I) value, even if local pose estimates temporarily diverge. This stability is beneficial for

- **Stability in Decision-Making:** High stability in Information (I) means that the system is less likely to react abruptly to short-term deviations in pose estimates. This is crucial in maintaining a consistent strategy for navigation and map exploration, Stability in (I) values prevents abrupt responses to short-term pose deviations supporting consistent navigation strategies.
- **Avoidance of Data Confusion:** Avoiding redundant revisits based on minor errors prevents deterioration of pose and map estimates by introducing noise or conflicting data, maintaining map clarity.
- **Resource Efficiency:** By curbing redundant revisits, the system conserves computational resources prioritizing pose improvement actions in less certain areas, enhancing overall navigation effectiveness.

Using EMMI with FMPs improves the balance between goal-directed navigation and pose accuracy.

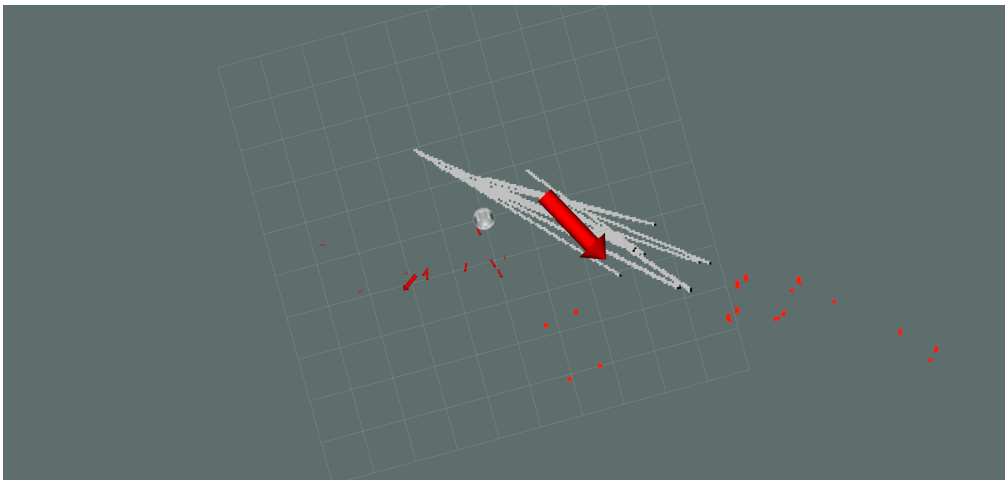

Figure 5: A low EMMI scenario in RViz, illustrating the particle distribution from GM via the subsubsection 5.2.2. Each red arrow corresponds to a particle, with arrow size proportional to its weight and likelihood. The white circle marks the robot's actual position. In this example, 14 out of 30 particles yield an EMMI value of 0.040922, indicating very low pose certainty. This low value stems from the wide spatial dispersion of particles and an inconsistent EM, highlighting the effectiveness of EMMI in capturing spatial discrepancies and quantifying pose certainty.

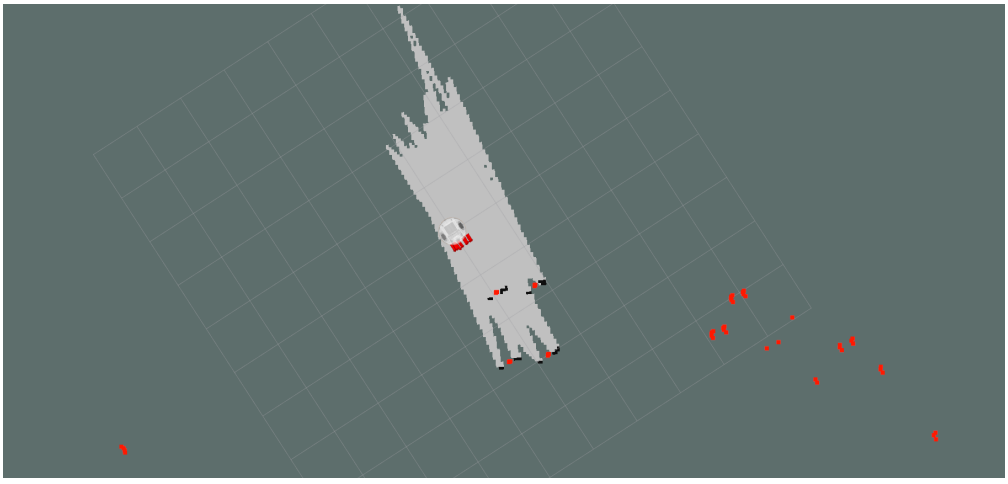

Figure 6: A high EMMI scenario featuring a tight clustering of all 30 particles near the robot's true position (white circle). The uniform arrow sizes imply a well-balanced weight distribution among the particles. In contrast to what a classic Shannon entropy measure (subsubsection 4.2.1) might suggest (i.e., high uncertainty), the EMMI value of 0.532546 here indicates robust pose certainty. The elevated value reflects a consistent EM, underscoring EMMI's utility in evaluating the need for pose-improvement actions.

4.3 Balancing Goal-Directed Navigation and Pose Improvement

In this section, we focus on achieving a balance between goal-directed navigation and pose stability within the DWA framework.

Building on the foundations of pose improvement actions and pose entropy estimation, this section addresses how these elements integrate to create a cohesive navigation strategy. We analyze the baseline DWA path selection process and discuss the interfaces and communication between components necessary to effectively balance these objectives.

Additionally, we explore the challenges encountered in aligning navigation goals with pose reliability, presenting strategies to overcome these issues and optimize the system's overall performance.

The diagram below provides an overview of the mechanisms, settings, and packages involved in controlling the PBF and its influence on trajectory selection.

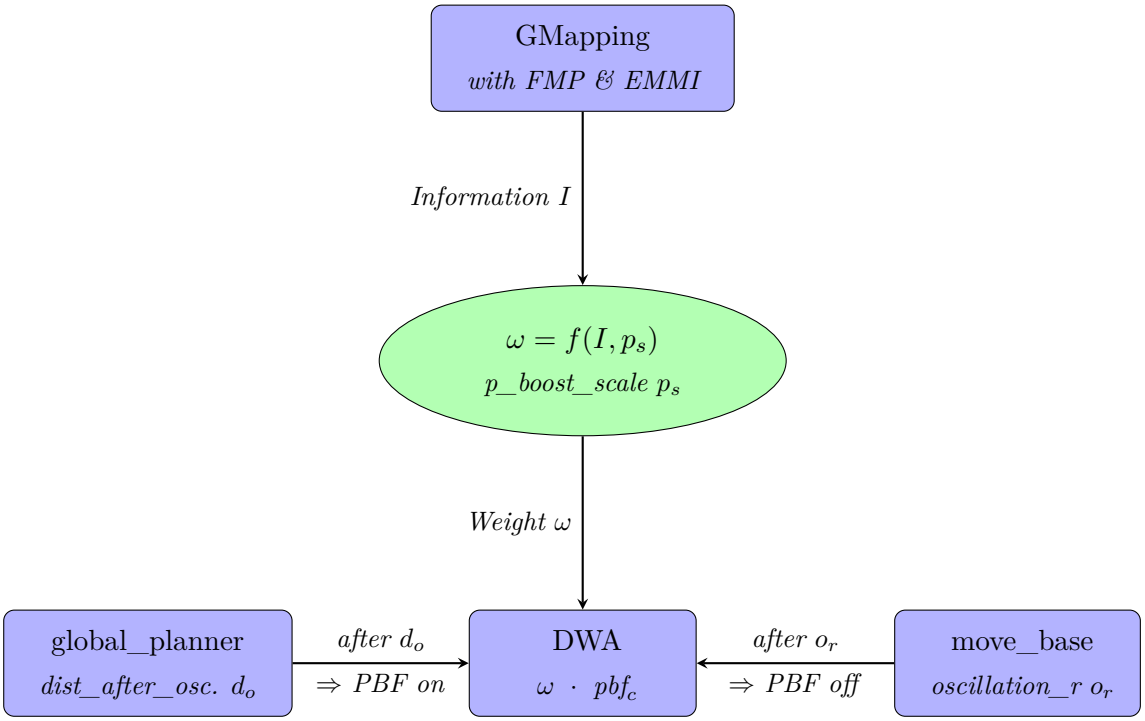

Figure 7: Overview of the automatic PBF impact mechanism.
The influence of the PBF is dynamically adjusted based on the user-defined parameter `pose_boost_scale` and the EMMI, as provided by the modified GM package. When move base detects a robot oscillation lasting longer than the parameter `oscillation_reset` seconds, the PBF is temporarily deactivated. It is then reactivated once the robot has traveled a distance defined by `distance_after_oscillation` meters. This ensures that the mechanism adapts to the robot's current navigation state while maintaining operational safety and efficiency.

The following subsections will examine the parameters, system components, and real-time computation of ω that contribute to harmonizing navigation and pose stability.

4.3.1 System and Variable Analysis

To appropriately calculate the weighting factor ω_{pbf} for the PBF, we analyze the dynamics of trajectory selection and the influence of existing cost functions. Understanding these interactions is necessary to assess the impact of new components such as the PBF.

Information I - Dynamic Window Approach

The DWA selects trajectories by evaluating each with cost functions that score aspects such as safety, efficiency and goal alignment (see Equation 18). The trajectory with the lowest cumulative score is chosen.

Integrating the PBF requires careful calibration to ensure it contributes to decision-making without overshadowing essential cost functions like collision avoidance. Initial analyses of existing cost values guided the calibration of the PBF output. Typically, a viable trajectory tr_i scores in the range:

$$0 \leq c_{\mathrm{tot}}(\mathrm{tr}_i) \lesssim 100 \tag{33}$$

Output Range of the PBF

To determine a suitable weight for the PBF, we examined the raw output range it produces.

The PBF is designed to influence the trajectory costs based on possible improvements in pose entropy (see section 4.1).

For a trajectory tr_i, the cost is calculated as follows:

$$\mathrm{c}_{\mathrm{pbf}}(\mathrm{tr}_i) = 1 - \frac{N_{occ}(\mathrm{tr}_i)}{\mathrm{Max}_{occ}} \tag{34}$$

where:

- Max_{occ} represents the maximum number of cells that could be observed by the robot,
- $N_{occ}(\mathrm{tr}_i)$ counts the occupied cells that were encountered in the ray-casting operation at the last pose of the trajectory.

The $\mathrm{c}_{\mathrm{pbf}}$ yields a value between 0 and 1, with lower values indicating that the trajectory tr_i could significantly contribute to improving pose entropy because many cells have been found to re-localize. This probabilistic output is logical given that the DWA planner opts for the trajectory with the lowest total cost. Consequently, PBF effectively penalizes trajectories that do little to revisit known areas, enhancing the likelihood of improving the pose.

PoseBoost Scale Parameter

To allow control over goal-directed navigation versus pose improvement, we introduce `p_boost_scale` (p_s) in the DWA configuration. This parameter scales the PBF impact on trajectory costs, encouraging trajectories that enhance pose certainty at higher values, while lower values focus on efficient navigation.

By tuning p_s, users can adapt navigation behavior to suit specific needs or environmental conditions, ensuring an optimal trade-off between rapid goal attainment and robust pose accuracy.

4.3.2 Adaptive Weighting Mechanism

This section outlines the methodology for adaptively calculating the PBF weight ω.

Desired Values of PBF

To integrate the influence effectively without overwhelming other crucial cost factors, the $\texttt{PoseBoost}_{\text{cost}}$ should not excessively exceed the accumulated cost of the the existing cost functions.

To ensure effective influence without overwhelming other cost factors, the target value for $\texttt{PoseBoost}_{\text{cost}}$ is set around 35:

$$\text{Target}_{\text{PoseBoost}}(\text{tr}_i) \approx 35 \tag{35}$$

Weight Calculation Formula

The weight ω for the PoseBoost cost function is calculated using Information I (representing the pose certainty) and the user-defined `p_boost_scale` (p_s) .

$$\omega = f(I, p_s) = (1 - I) \cdot p_s \tag{36}$$

where:

- Information: $0 \leq I \leq 1$
- `p_boost_scale`: $0 \leq p_s$ (default: 200)

4.3.3 Managing conflicting objectives

Integrating PBF into the DWA planner, which is designed to increase revisiting actions to improve the pose, presents several operational challenges. These challenges are particularly prominent when balancing potentially conflicting navigation objectives.

A primary challenge is oscillation, which arises when the robot is poorly localized and the navigation goal is in an open, yet-to-be-explored area. In such cases, the drive to reach the goal may conflict with the need to revisit known areas, leading to counteracting cost functions that cause the robot to oscillate.

To mitigate this, a timeout mechanism using the `oscillation_timeout_pose_boost` parameter (o_t) temporarily disables the PBF upon detecting oscillation, allowing the robot to focus on reaching its goal without interference from conflicting cost functions.

The PBF is reactivated only if a new global plan is generated and the robot has moved a minimum distance, as specified by the `pose_boost_min_distance` parameter. This configurable parameter ensures that the PBF is used strategically, prioritizing stability and preventing excessive or premature pose improvement actions.

4.3.4 System Configuration and Parameter Integration

Integrating the PBF enhancements required modifications in several ROS modules. The key components with their settings and parameters for effective PBF integration include:

- FMPMapping

  Calculating pose certainty information I

  · `entropy_threshold`: Normalizing FMP Entropy (see subsubsection 4.2.4) (default: $4 \mathrel{\widehat{=}} 150$ consistent measurements).

- DWA

  Estimation of Pose Improvement Potential

  · `p_boost_scale`: Adjusts the overall influence of PBF on trajectory selection, allowing users to control the emphasis on pose improvement (default: 70).

  · `p_boost_min_distance`: Specifies the minimum safe distance from obstacles within which PBF does not apply additional costs, thus maintaining safe navigation distances (default: 0.3 m).

- `pose_boost_distance_after_oscillation`: Defines the minimum distance the robot must travel after an oscillation event before the PBF can be reactivated, preventing premature reactivation (default: 0.4 m).

Additional options allow users to manually adjust the PBF laser topic and toggle visualization functions (see subsubsection 5.1.2).

- Move Base

  Detection of oscillation
  - `pbf_oscillation_reset`: Sets a timeout period, after which PBF is temporarily disabled if oscillation is detected, ensuring stable navigation (default: 2.5 seconds).

- Global Planner

  Reactivates the PBF if previously deactivated due to oscillation, whenever a new global plan is generated (after traveling `pose_boost_distance_after_oscillation` meters).

These parameters are designed for dynamic reconfiguration during runtime, allowing the system to adapt to changing environmental conditions and operational demands. ROS messages facilitate seamless communication between components, ensuring flexible and responsive navigation strategies.

4.3.5 Final PoseBoost cost function Output

The final PBF cost value for a trajectory tr_i at time step t results from combining the dynamically computed weight with the trajectory's ability to improve pose accuracy. Formally, it is expressed as:

$$\mathrm{PBF}_{\mathrm{cost}}(\mathrm{tr}_i) = \omega(t) \cdot \mathrm{c}_{\mathrm{pbf}}(\mathrm{tr}_i) \tag{37}$$

where:

- $\omega(t)$ indicates the required extent of pose improvement at time step t (see Equation 36),

- $\mathrm{c_{pbf}}(\mathrm{tr}_i)$ represents the trajectory-specific potential to enhance pose certainty (see Equation 34).

By expanding these terms, we obtain the complete formula:

$$\mathrm{PBF_{cost}}(\mathrm{tr}_i) = (1 - I(t)) \cdot p_s \cdot \Big(1 - \frac{N_{occ}(\mathrm{tr}_i)}{\mathrm{Max}_{occ}}\Big), \tag{38}$$

where:

- I denotes the EMMI (pose certainty estimation) at time step t,
- p_s is the user-defined parameter that scales the influence of the PBF,
- Max_{occ} represents the maximum number of cells that could be observed by the robot,
- $N_{occ}(\mathrm{tr}_i)$ counts the occupied cells that were encountered in the ray-casting operation at the last pose of the trajectory.

This formulation ensures that each trajectory's cost is adjusted according to both the robot's present need for re-localization and the trajectory's ability to contribute to that re-localization.

4.3.6 Full DWA Optimization Formulation

At each control cycle, DWA generates and evaluates a set of feasible trajectories by sampling linear and angular velocities (v, w) from the robot's dynamic window (see subsubsection 2.8.1). Each candidate trajectory tr_i is then assigned a total cost, denoted by $c_{\mathrm{tot}}(\mathrm{tr}_i)$ (see Equation 18).

Additionally to the standard cost functions (see Algorithm 3) the DWA planner now also considers the $\mathrm{PBF_{cost}}$.

$$\begin{aligned} c_{\mathrm{tot}}(\mathrm{tr}_i) = {} & \mathrm{oscillation_{cost}} + \mathrm{obstacle_{cost}} + \mathrm{goal\_front_{cost}} + \mathrm{alignment_{cost}} + \\ & \mathrm{path_{cost}} + \mathrm{goal_{cost}} + \mathrm{twirling_{cost}} + \mathrm{PBF_{cost}} \end{aligned} \tag{39}$$

The robot ultimately selects the velocity command pair $(v^\star, w^\star)$ that minimizes the new total cost:

$$(v^\star, w^\star) = \underset{(v,w)\in\mathcal{V}_{\mathrm{feasible}}}{\arg\min} \; c_{\mathrm{tot}}(\mathrm{tr}(v, w)) \tag{40}$$

5 Implementation

In this chapter, we detail the practical implementation of the concepts and strategies discussed in chapter 4. The following sections provide an in-depth explanation of the methods and tools used to realize these ideas in practice.

5.1 Pose Improvement

To incorporate a new cost function in the DWA planner a thorough understanding of its structure within the ROS navigation stack framework is required.

5.1.1 GMapping Framework and Requirements

The DWA planner acts as a wrapper around the base local planner and is responsible for generating accessible trajectories. The base local planner then evaluates each trajectory tr based on different objectives represented by the cost functions (see subsubsection 2.8.1).

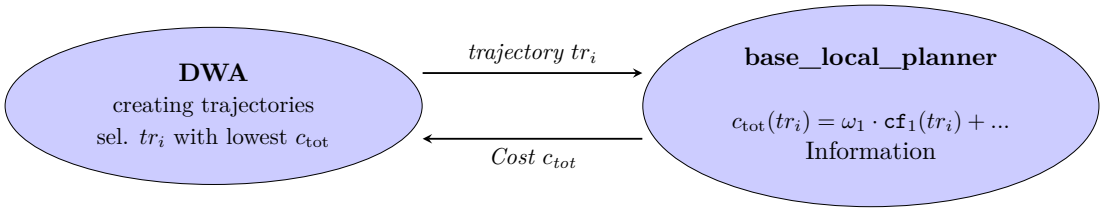

Figure 8: Trajectory selection - Interaction between the DWA and the base_local_-planner.

Requirements For a successful implementation, the following challenges will be addressed:

- Integrating the new cost function in the framework
- Accessing the map data in real-time
- Performing the ray-casting operation

- Tracking occupied cells
- Visualizing the outcome to validate correctness and accuracy

5.1.2 Implementation of Components

The components designed to meet the specified requirements are illustrated in Figure 9. In what follows, we will describe the purpose of each component and clarify how the PBF functions within the DWA planner.

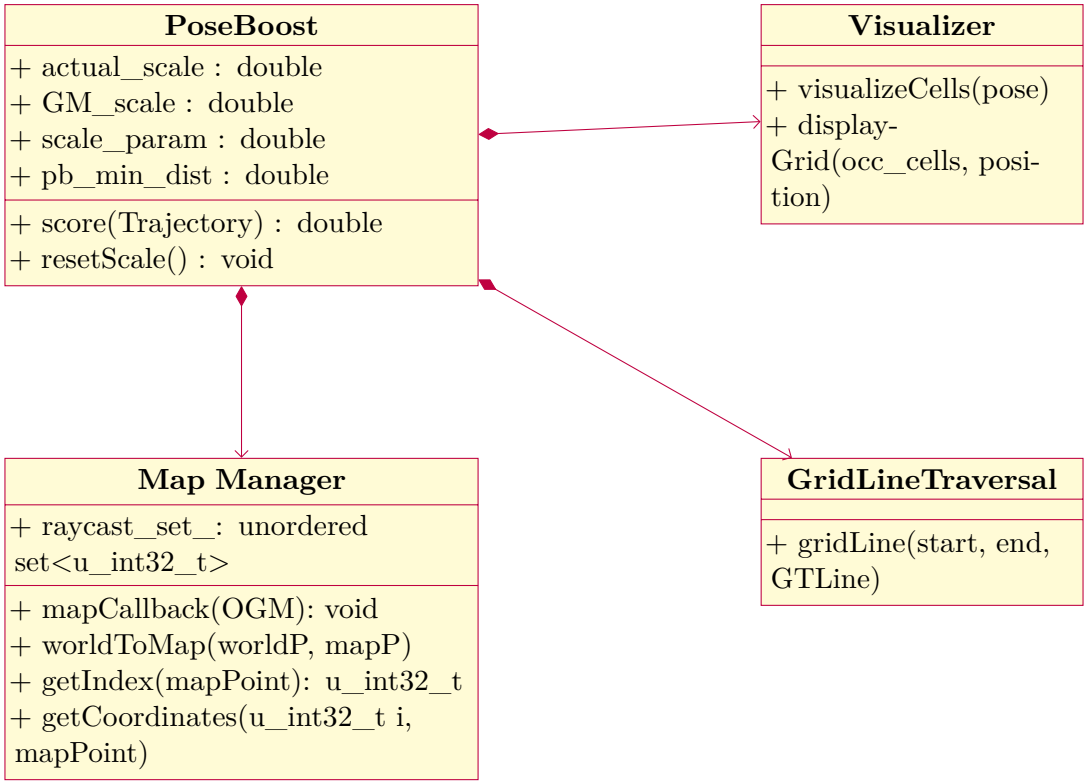

Figure 9: DWA Local Planner: Classes involved in the PoseBoost Cost Function

PoseBoost cost function

All cost functions in the DWA planner must implement the `TrajectoryCostFunction` interface. During trajectory evaluation, the `base_local_planner` invokes the `score` function to retrieve the cost values for a given trajectory.

To integrate our new PoseBoost cost function (PBF), we implemented the `TrajectoryCostFunction` interface and incorporated it into the DWA planner's trajectory evaluation process. This addition ensures that the PBF contributes to the overall cost calculation, allowing the planner to factor in the potential for pose improvement when selecting optimal trajectories.

The trajectory scoring process requires efficient ray casting operations and access to the most up-to-date map. To achieve this, the PBF leverages external modules:

- The `Map Manager` handles map updates and tracks occupied cells in real time.
- The `namerefsubsubsec:gltraversal` module efficiently performs ray casting operations to evaluate the trajectories.

Map Manager

The Map Manager plays an important role in managing the map data and tracking cells during ray-casting operations for each trajectory. It subscribes to the `/map` topic published by the GM package, ensuring real-time access to the latest map data.

Challenges in Computational Efficiency Updating local plans at high frequencies for each trajectory poses computational challenges. Consider our base configuration as an example:

- Laser range $r = 5m$
- Laser angle $\alpha = 200°$
- Number of beams $b = 605$
- Map resolution $res = 0.05$
- Frequency $f = 20\text{Hz}$

Required number of cells per second:

$$b \cdot \frac{r}{res} \cdot f = 726\ 000\ 000\ \frac{\text{cells}}{\text{second}} \tag{41}$$

Managing such a vast number of cells per second can degrade the planner's responsiveness, potentially compromising its functionality.

Optimizing Data Management To address the high computational demands of the application, various data structures were evaluated to determine the most efficient solution for managing and accessing map indices. The following approaches were tested:

1. `unordered_set<tuple<int, int»:`
   This data structure stores map indices as tuples in an unordered set, providing several advantages:
   - Straightforward implementation with minimal complexity.
   - Ensures element uniqueness without additional checks.
   - Facilitates direct retrieval of the number of cells by querying the set size.
   - Efficient reset capabilities, enabling quick re-initialization between trajectory evaluations.

   However, the computational overhead associated with hashing and comparing tuples proved to be a limitation, especially for high-frequency operations, rendering this approach suboptimal for real-time applications.

2. `int8[]:`
   This method represents a cropped region of the map, restricted to areas within the laser range of the robot. Although it offers constant-time access and eliminates the need for runtime memory allocation, the following drawbacks were identified:
   - Significant time overhead due to copying the array during each iteration.
   - The need to iterate through the entire array to count cells introduces inefficiencies.

   These limitations reduced the practicality of this approach for the given application.

3. `unordered_set<u_int32_t>:`
   This method amalgamates the x and y coordinates into one `u_int32_t` value using either multiplication or a more rapid bit-shifting technique. This composite value is subsequently stored within an unordered set. This process preserves the advantages of a tuple-based implementation, which include simplicity and rapid access, while significantly lowering the computational expense linked to hashing and comparing two distinct values.

Following comprehensive analysis and performance evaluations, the `unordered_set<u_int32_t>` was selected for its superior performance as the most suitable data structure for this application. This choice ensures scalability and responsiveness during high-frequency operations, reflecting a careful assessment of both theoretical and practical performance aspects.

Grid Line Traversal

The last integral part of the ray casting process is the efficient and precise implementation of the Bresenham-based supercover line algorithm. To optimize performance, the `GridLineTraversal` class utilizes static memory allocation upon its creation. This approach minimizes the overhead associated with dynamic memory allocation during run-time, ensuring that the ray-casting operations can be performed at the high frequency required by the planner.

For each laser ray, the `gridLine` function receives the coordinates of the start and end cells from the PBF. It then computes and returns all cells intersected by the line connecting these points (`GTLine`). This set of intersected cells is subsequently passed on to the Map Manager, which tracks the cells and their occupancies encountered along the trajectory.

Visualizer

To confirm that cells encountered during ray casting were correctly identified and counted, a visualizer was developed. This tool provides insight into how the results of the ray casting influenced the value of PBF for pose improvement.

The Visualizer is capable of displaying the set of cells penetrated during a ray casting operation for a given position (see Figure 11). This feature was particularly useful for debugging and fine-tuning the ray-casting process. Furthermore, the Visualizer publishes messages that can be viewed in RViz, showing the pose and range of the trajectory currently evaluated (see Figure 10). This immediate visualization was valuable not only for identifying faults but also for comprehending the spatial interplay between the robot's pose and the areas analyzed for pose improvement.

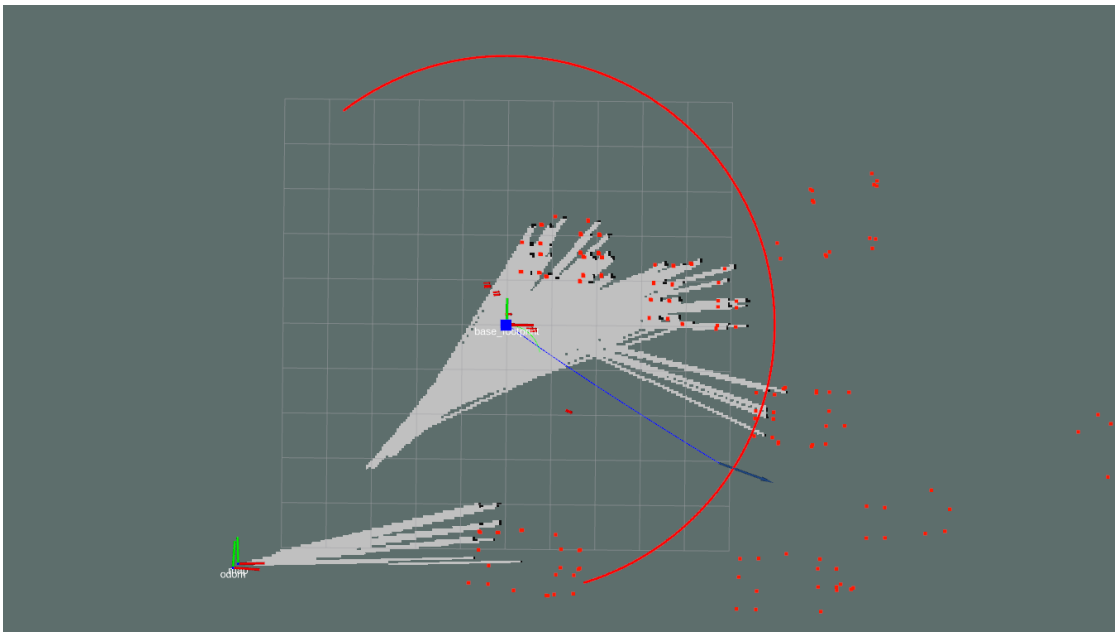

Figure 10: RViz visualizer: The blue dot in the middle represents the end of the local trajectory which is being evaluated. From there we will perform the ray casting operation till the robot hits an occupied cell or reaches the maximum range of the laser.

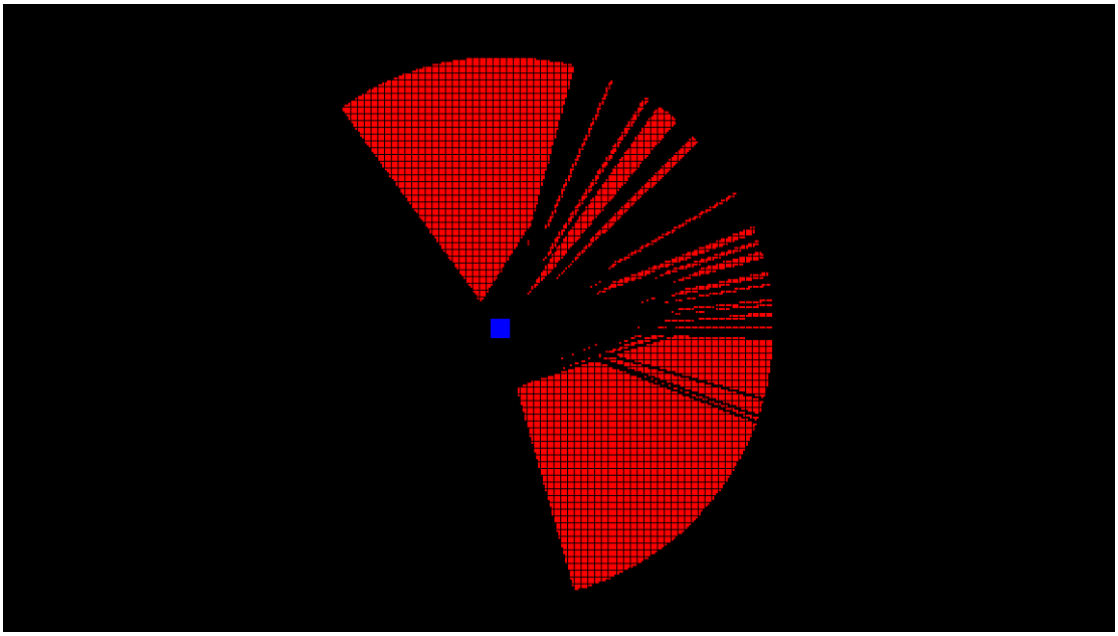

Figure 11: The blue point is again the position of the laser. In this visualization, unknown cells are highlighted in red.

5.2 Pose Entropy

Integrating new components into the GM framework required a thorough analysis of its existing structure and functionality.

5.2.1 GMapping Framework and Overview of Modifications

This work focuses on the `SLAM_Gmapping` package and its core components, particularly the `GridSlamProcessor`, which is responsible for managing particle states and updating the map.

To identify optimal points for integrating the EMMI calculation, a detailed examination of data and control flows was performed, spanning from data ingestion to map generation (see Figure 12).

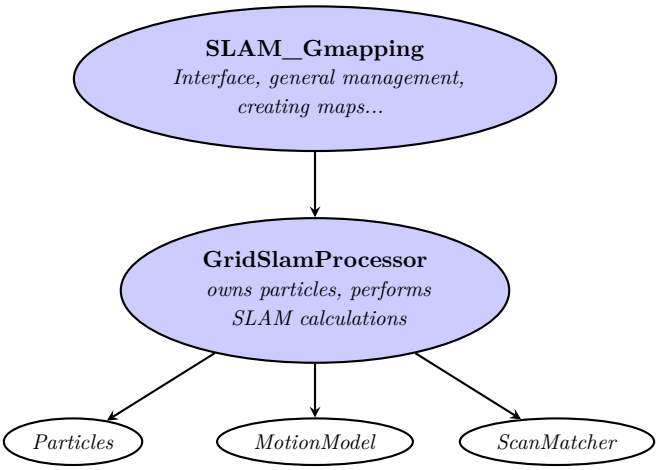

Figure 12: Overview of the GMapping package interactions. *For more details please refer to Arce y de la Borbolla [2021]*

In the `SLAM_Gmapping` package, the main program serves as the primary interface for users, handling overall functionality, and managing interactions with the system. It acts as a wrapper around the `GridSlamProcessor`, which performs the core computations of the SLAM algorithm.

5.2.2 Implementation of the New Approach

This section details the implementation of the new functionalities (see Figure 13) as described in section 4.2.

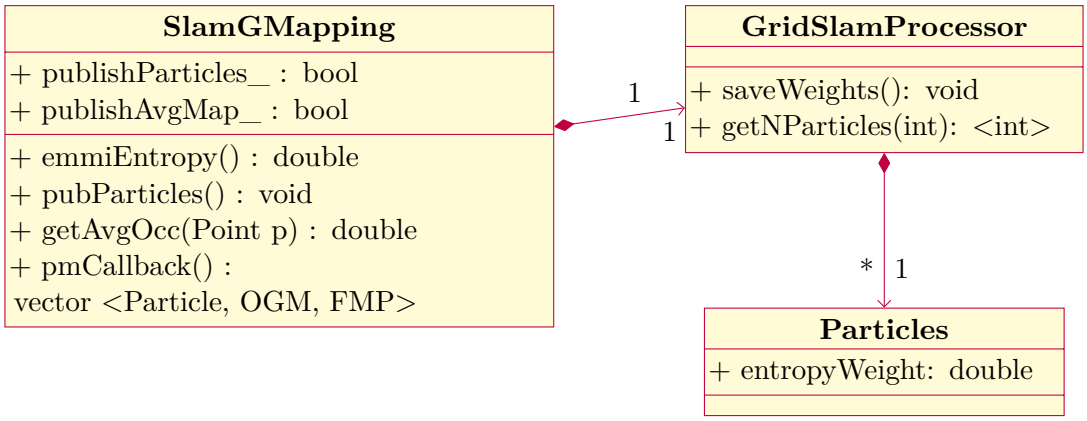

Figure 13: Diagram showing the newly implemented functions and classes in the pose entropy calculation system. This diagram highlights the interconnections and data flow between components, providing a broad visual representation of the system architecture.

Obtaining an accurate representation of the particle weight distribution is feasible only during scan matching. Consequently, the EMMI calculation was implemented within the `SLAM_Gmapping` package to occur whenever a scan update is performed. This ensures that the entropy calculations are both precise and aligned with the most recent map data.

Capturing Representative Weights

As highlighted in subsection 4.2.2, one of the primary challenges in the calculation of pose entropy within the GM framework is the transient nature of the particle weights, which are updated during resampling.

To address this, a new parameter was introduced into the particle data structure (see Figure 13). This parameter stores the weight of each particle during the scan matching phase, ensuring that these values remain accessible for the subsequent EMMI calculation before being overwritten in the resampling step.

During calculation, the stored weights are normalized across all unique particles to produce a meaningful and representative distribution. This approach ensures that the entropy calculations accurately reflect the current pose uncertainty and are not skewed by transient weight changes during resampling.

Extended Functionalities in GMapping

Besides the functions implemented for enhanced entropy estimation, several other features were integrated into the GM package extending its functionality:

- **Particle Publisher**
  The Particle Publisher (see Figure 14) module enables monitoring and debugging of particle states by publishing the pose and weight for real-time visualization in RViz. This tool helps to understand the particle distribution and verify pose entropy calculations.

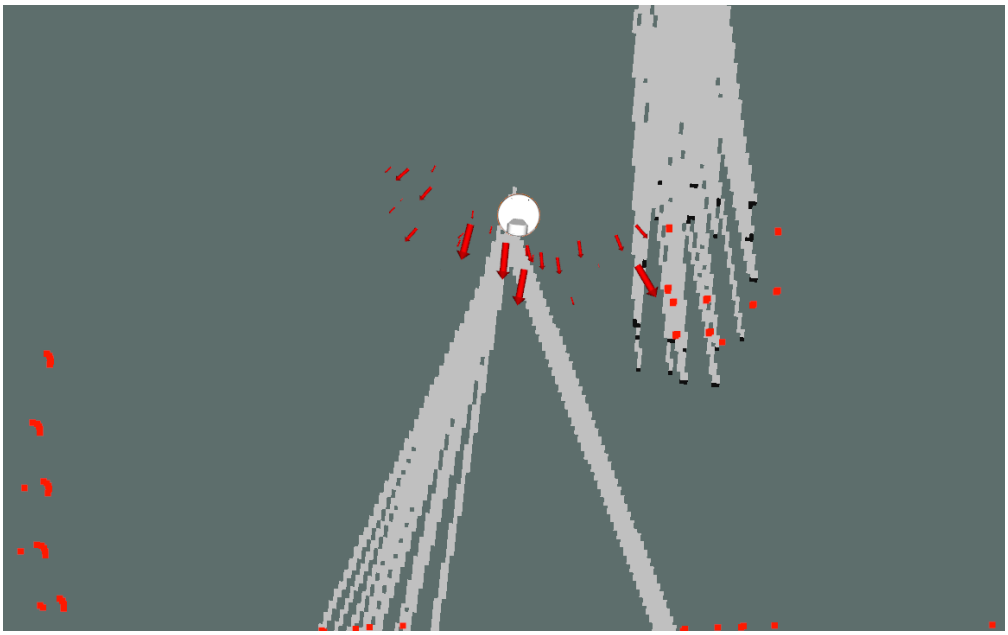

Figure 14: Particle Publisher Visualization: This image displays the distribution of particles within the GM algorithm, represented by red arrows where each arrow's size corresponds to the particle's normalized weight. The actual position of the robot is denoted by the white circle, providing a reference for assessing the spread and alignment of the particle weights relative to the robot's true location. The visualization helps to understand how the SLAM system perceives and adjusts to environmental data.

- **Average Map Calculation**
  The Average Map Calculation function computes the Expected Map (Average Map) from all particles, reflecting the consensus among trajectory hypotheses. By visualizing the averaged maps in RViz (see Figure 15), we can evaluate the convergence of the SLAM process identifying areas where the robot's understanding of the environment is still uncertain.

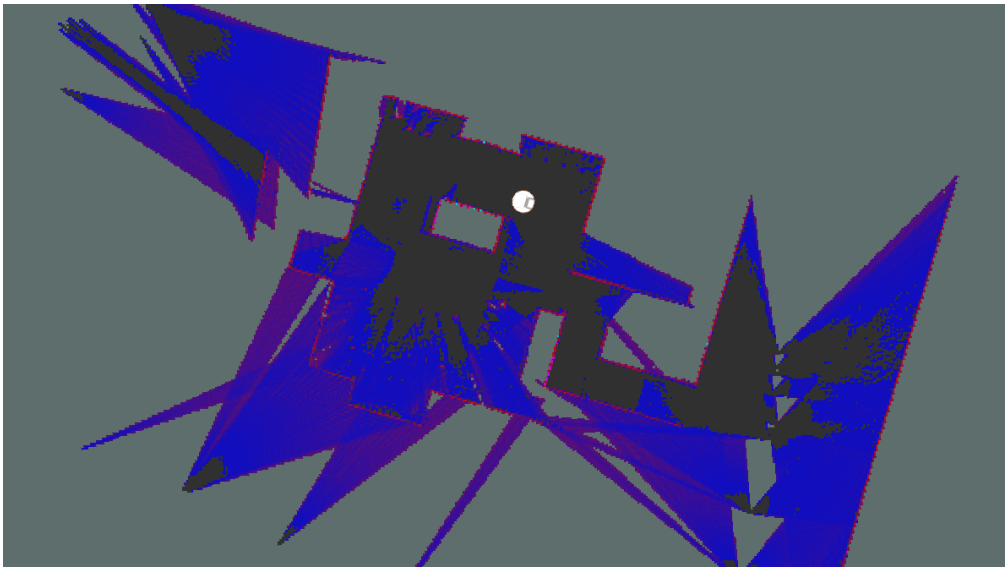

Figure 15: Visualization of the Averaged Map in RViz: This screenshot from RViz displays the averaged probability map generated by the enhanced GM package. The map aggregates occupancy probabilities across all particles, illustrating areas with varying levels of certainty. The white circle indicates the real position of the robot. Grey areas represent unexplored cells; red cells denote high occupancy probability, signifying likely obstacles. Blue and purple areas indicate cells with uncertain occupancy, with blue suggesting a higher likelihood of being empty. This coloration highlights the influence of the prior from the Full Map Posterior, especially in regions with sparse sensor data.

- **Particle and Map Service**
  The `Particle and Map Service` enables efficient data access for other ROS components. Upon request, the `GridSlamProcessor` selects and publishes n randomly sampled particles, including their OGMs and associated poses, for use by external modules. The service leverages multithreading to optimize performance and speed. In addition, it supports FMP maps, offering a comprehensive probabilistic repre-

sentation of the environment. An example output can be found in the Appendices (see section 8).

6 Experiments and Evaluation

This chapter examines how effective the modifications to the DWA planner and the GM package are. We executed tests in several scenarios to verify whether these improvements truly enhance pose estimation in autonomous navigation.

6.1 Comparison of Planning Strategies

To evaluate the influence on navigation efficiency and pose accuracy, we contrasted our method with the standard DWA planner and a straightforward strategy designed to enhance pose estimation.

- **DWA**
  The standard DWA Planner serves as the baseline and is often used in typical navigation strategies in known environments. It focuses on achieving navigation objectives efficiently, omitting functionalities to actively improve pose estimation.

- **PBF**
  The modified DWA planner incorporates the PoseBoost cost function. It assimilates the certainty of the pose in real time, derived from the modified SLAM GM package, into the cost functions. This feature aims to balance efficient navigation with the need for accurate pose estimation.

- **360**
  Given the absence of existing methods that concurrently optimize navigation and pose estimation, we implemented a straightforward experimental approach. This technique employs the standard DWA, enabling the robot to execute a 360-degree rotation after advancing each meter towards its target. The goal is to enhance the frequency of scan updates periodically, supplying additional data to the GM, thereby increasing scan updates. These additional scan updates are anticipated to enhance map quality and thus increase scan-matching operations, which should subsequently improve pose estimation.

6.2 Testing Metrics

To evaluate the effectiveness of planning strategies, we employed several metrics that capture both quantitative and qualitative aspects of performance. These metrics facilitate the evaluation of improvements in pose estimation precision, navigation effectiveness, and system reliability.

- **Pose Accuracy:**
  Measures the Euclidean distance and the angular deviation between the estimated pose of the robot provided by GM and the ground truth poses. This metric directly reflects the accuracy of the localization process.

- **Expected Map Mean Information:**
  Used to quantify the certainty of the current pose of the robot and the surrounding map. A higher EMMI score indicates greater confidence in pose estimation and map quality (see 2.6.2).

- **Navigation Efficiency:**
  Assessed by recording the time taken and the distance traveled to reach the navigation goal. This metric evaluates how effectively the navigation strategies reach the goal.

- **System Reliability:**
  Monitors occurrences of navigation failure (i.e., when `move_base` aborts) that can arise from incorrect pose estimations or excessive computational demands that adversely affect system performance. This metric helps identify the robustness of the modifications implemented under various operational conditions.

6.3 Experimental Setup

We performed tests in both simulated and real world environments to model varying conditions. [PL says: "though in the introduction you refer the motivating RoboCup@Home Restaurant task, here you do not mention it anymore, and you never explain the details of the task anywhere in the dissertation. I think you should anchor the case study to that task and provide a description of it"] [Rob says: "I was just implementing this idea, when I realized, that I am not sure, whether this is a good idea. Sadly my algorithm was

not ready yet when the robocup competition took place so it waas not tested there. Also tests in the ISR-testbed were done without any furniture and not really similar to the restaurant task. It was purely navigation. Tests in simulation were 30% conducted in a szenario similar to the restaurant, but the rest was performed in other szenarios, because goal and start positions in a small restaurant szenario are limited."]

Simulation Setup

All tests in simulation were performed using the Gazebo simulator [Robotics, 2024], with a TIAGo robot from PAL Robotics. Two custom environments were created within Gazebo to challenge both localization and mapping capabilities:

- **Restaurant Environment** *(Figure 16)*: Inspired by the RoboCup@Home restaurant task (section 1.1), this environment is designed to mimic a small dining area with tables, chairs, and a bar-like station. In our tests, the robot typically starts near the entrance and must navigate to a designated service point (e.g., near the bar or a customer table), requiring it to avoid tables, chairs, and other obstacles.

- **Subway Station Environment** *(Figure 17)*: Created to simulate a larger, more open public area, this setting features sparse landmarks and a layout reminiscent of a subway platform or corridor. Here the robots start and end position were chosen randomly. This scenario helps evaluate the system's ability to maintain accurate localization over longer distances.

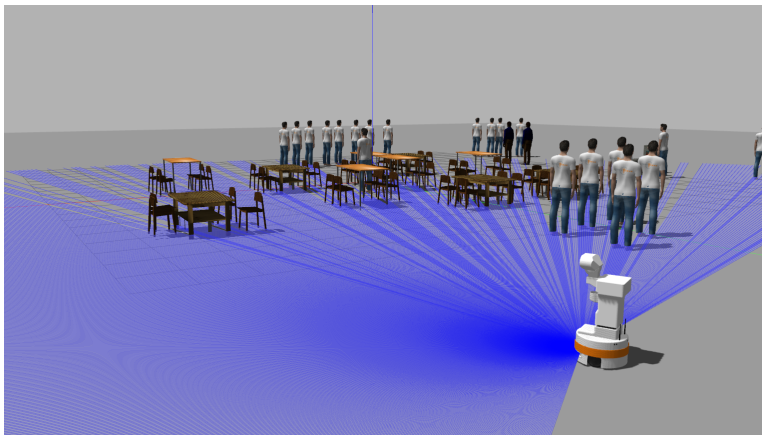

Figure 16: A screenshot of the Gazebo simulator showing the TIAGo robot navigating the restaurant-like environment. Obstacles include tables, chairs, and other furniture configured to mirror a typical dining layout.

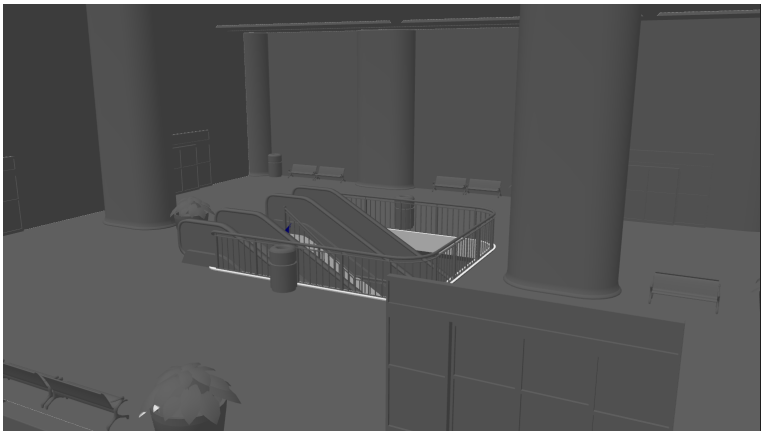

Figure 17: The second simulated environment, designed to represent a subway station or similar public area with sparse obstacles and more open space.

Real-World

Real-world tests were conducted using the TIAGo robot, nicknamed "Bobby", in the Institue For Systems and Robotics Lisboa (ISR) laboratory test bed (see Figure 18). To mimic sparse environment, most of the obstacles were cleared from the test area. For each trial, the robot was positioned at a designated start location and tasked with navigating to a goal position. A video recording of a representative test run is available at `steigerob.github.io/Thesis/videos/`, where the robot's behavior and trajectory can be observed.

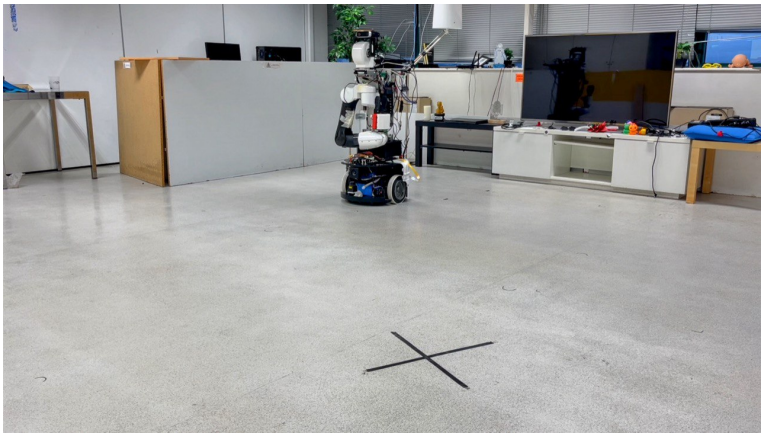

Figure 18: The TIAGo robot "Bobby" navigating in the ISR test bed, cleared of most obstacles to mirror the sparse simulation conditions.

Parameters and Variables

To improve the generalizability of our findings, we predominantly utilized the default parameter configurations.

Variations from these configurations and parameters newly introduced in this work are outlined below:

- **PoseBoost cost function**:
  - `p_boost_scale`: 400
  - `p_boost_min_distance`: 0.3 meters
  - `p_boost_oscillation_timeout`: 2.5 seconds
  - `p_boost_distance_after_oscillation`: 0.4 meters
- **GMapping**:
  - `entropy_threshold`: 4
  - `alpha0`: 0.0231
  - `beta0`: 0.6156

  For the documentation about the FMP parameters (α_0 and β_0) please refer to Arce y de la Borbolla [2021].
- **Move Base**
  - `sim_time`: 5.0 seconds (default: 1.7) to allow better prediction of potential re-localization actions.
  - `path_distance_bias`: 12.0 (default: 32.0) to decrease the influence of the global plan on local decision-making, encouraging the selection of more informative rather than merely expedient paths.
- **Noisy Movement**

  To simulate odometry errors in the simulation, random Gaussian noise was added to the robot's movements. The noise, characterized by a mean of 0 and a variance of 0.0175, was applied to both linear and angular velocities. This was implemented by subscribing to the relevant ROS topic, modifying the motion commands with the noise, and publishing the updated messages.

For the complete configuration files used for testing, please refer to section 8.

Data Collection Methods

The evaluation metrics require a comparison between ground truth poses and estimations of GM. While obtaining ground truth data is straightforward in simulations, for our real-world experiments, we utilized an OptiTrack motion capture system [NaturalPoint Corporation, 2022]. OptiTrack is a high-precision motion tracking solution that uses infrared cameras to detect reflective markers and calculate their positions in three-dimensional space. In our setup within the ISR testbed, twelve cameras tracked three reflective markers mounted on the robot. This configuration ensured accurate tracking of the robot's position and orientation, providing reliable ground truth data for comparison with the estimated poses from the GM algorithm.

6.4 Results

This section presents the results of experiments conducted in both simulated and real world environments. We compare the performance of three distinct planning strategies:

- **DWA:** The baseline Dynamic Window Approach (DWA) planner.
- **PBF:** The improved DWA planner incorporating the Pose Boost Function (PBF).
- **360:** The DWA planner supplemented by a simple pose improvement strategy.

Graphical Representations

Bar graphs (Figure 20 and Figure 22) display the following statistics for test runs:

- The mean and variance of the distance traveled from the starting point to the end of the navigation.
- The average time required to reach the destination or encounter a navigation failure.
- The percentage of navigation failures attributed to poor localization.

Line graphs 19 and 21 illustrate the changes over time in the Euclidean distance error between the estimated pose and the robot's true location. These graphs plot the average error (in meters) on the y axis against the percentage of the journey completed towards the goal on the x axis. Additional graphs detailing the angular difference and the development of EMMI are provided in section 8.

Simulation Environment Results

The following graphs present the aggregated results from 75 **(TODO: we did 75 so far... wanna do moooore?)** tests conducted in both of the simulation environment:

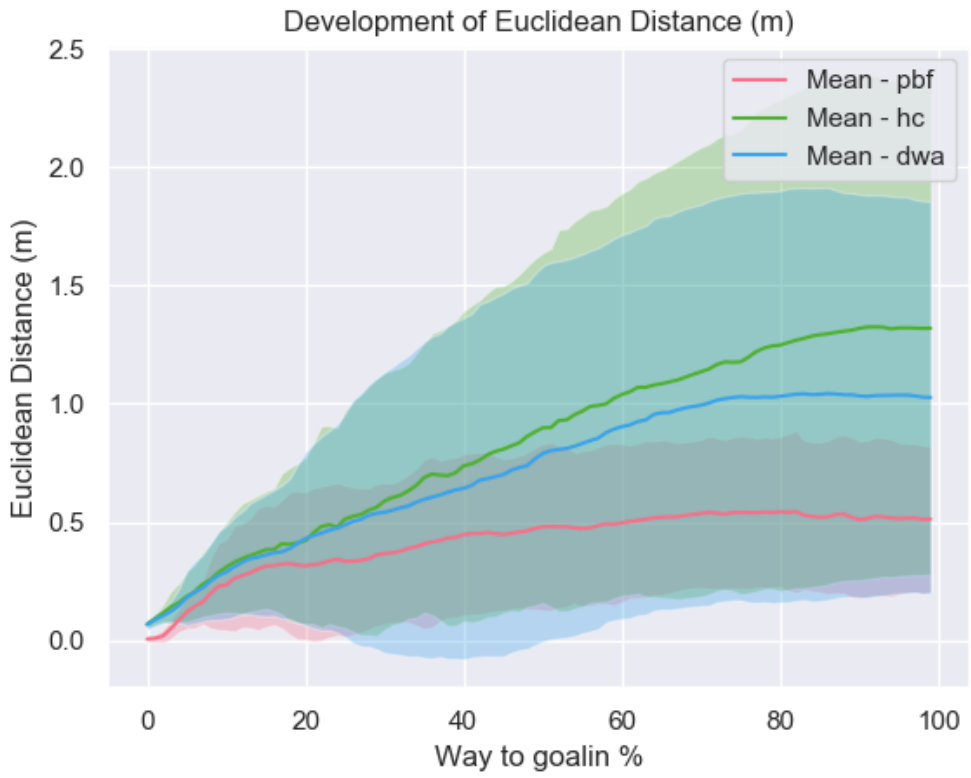

Figure 19: Mean positional error development in the simulation environment.

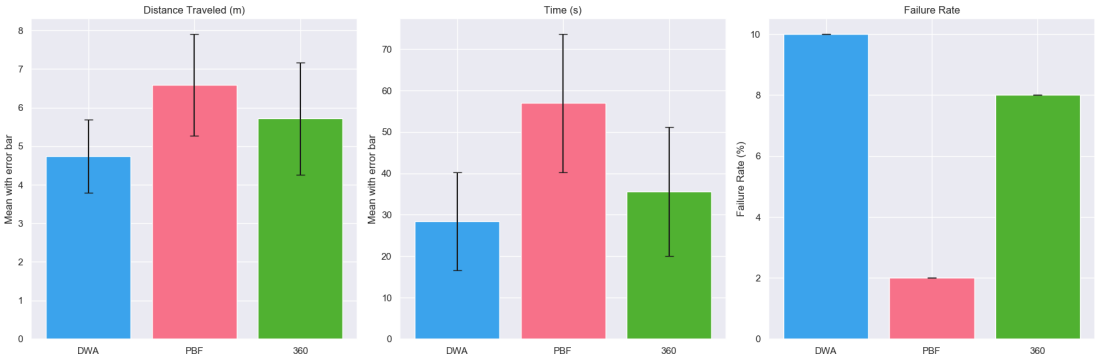

Figure 20: Distance and Time needed till end of navigation, percentage of navigation failures in simulation environment.

Real-World Environment Results

These graphs illustrate the results from a series of 50 experiments carried out in the ISR Testbed (see Figure 18):

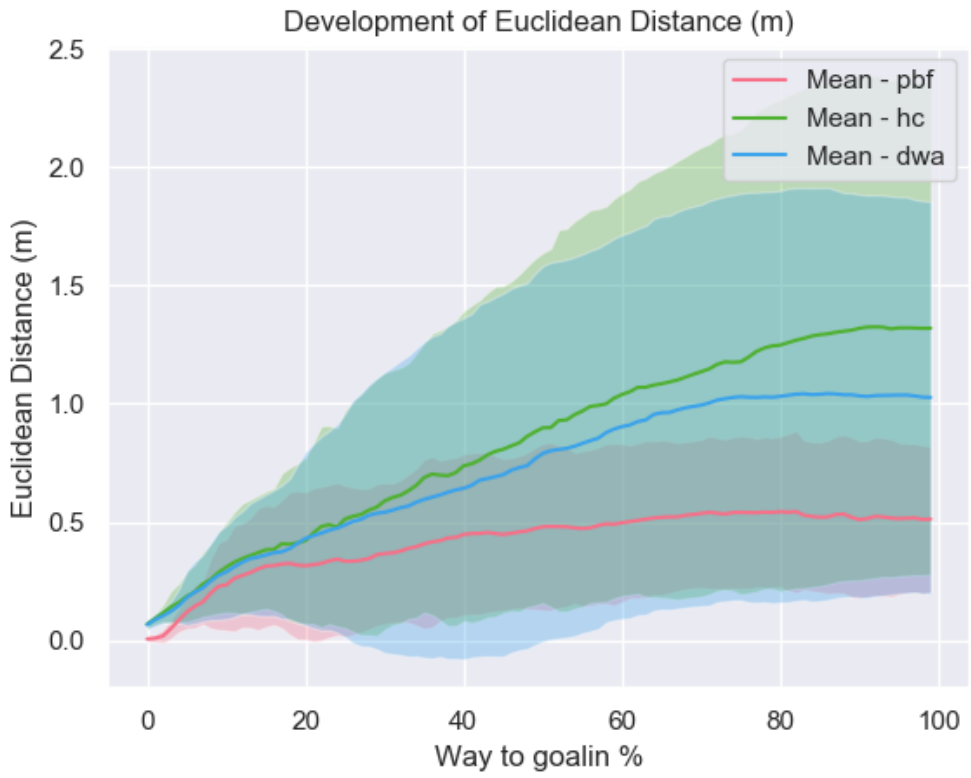

Figure 21: Mean positional error development in the real-world environment.

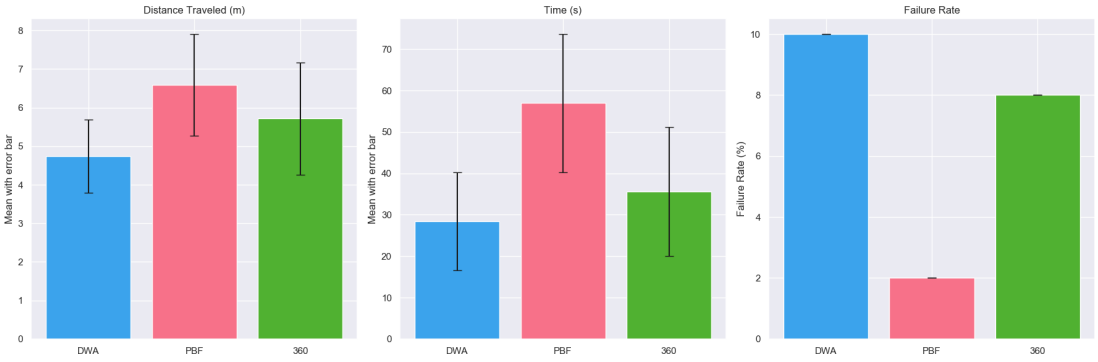

Figure 22: Distance and Time needed till end of navigation, percentage of navigation failures in real-world environment.

6.5 Discussion

- **General performance:** The experimental results reveal distinct performance characteristics across the three navigation strategies tested. The PBF-enhanced DWA planner demonstrates superior pose accuracy, significantly increasing the mapped area and its certainty compared to the baseline DWA and the simple 360-degree turn strategy. Although the PBF approach takes longer and covers more distance on average to reach the goal, its ability to decrease navigation failures (from 10% to 2%) underscores its efficacy in maintaining robust pose estimation even in challenging navigation scenarios.

- **Analysis of pose improvement strategies:** The simple pose improvement strategy, designed to periodically improve pose accuracy through deliberate orientation changes, proved less effective than anticipated. The underperformance of this strategy highlights the complexity of pose improvement, which requires more than just frequent measurement updates.

- **Performance in Different Scenarios:** The PBF modification shows the most notable advantages in environments where odometry errors are likely to accumulate due to longer travel distances or complex trajectories. In cases where the goal point was close and in front of the robot, the baseline DWA planner was able to outperform the PBF. This can be explained due to introduced route deviations creating additional uncertainty and noise. This observation suggests that the enhanced planner is particularly beneficial in scenarios where odometry errors are likely to accumulate due to non-trivial trajectories or unreliable odometry.

- **Behavioral Insights** Observational data from tests (recordings available at: `steigerob.github.io/Thesis/videos/`) indicate that the PBF strategy modifies the robot's behavior to verify its pose by referring back to previously mapped areas, especially when it detects uncertainty in its pose. This behavior mimics a 'confirmation' strategy where the robot reassures its localization by aligning its current sensor readings with the known map features provoking scan-matching, thus effectively utilizing its historical data to mitigate present uncertainties.

- **Impact of Ground-Truth Odometry in Simulation:** During early simulation tests, the PBF-enhanced DWA sometimes underperformed compared to the baseline DWA. This discrepancy arose from the simulation's reliance on perfect ground-truth

odometry: while GM artificially injects noise into particle movements, the baseline DWA gained an advantage by avoiding additional (and thus unhelpful) revisits. However, this setup is unrealistic for real-world applications where odometry errors are inevitable. Under normal conditions the PBF-based revisiting actions prove effective for correcting pose errors, as our results proofed.

Conclusion

The analysis clearly demonstrates the effectiveness of integrating pose improvement mechanisms into autonomous robot navigation planning. The PBF not only enhances pose accuracy but also contributes significantly to reducing navigation errors, highlighting its potential for applications that demand high reliability in autonomous navigation tasks.

While the increased time and distance required to reach the goal may be viewed as drawbacks, these are mitigated by the benefits of a more informative map and the exploration of a larger portion of the environment. These advantages lead to overall improvements in navigation reliability and robustness, underscoring the value of incorporating pose improvement strategies into autonomous systems.

7 Conclusion and Future Work

This chapter revisits the contributions delineated at the outset of the thesis and evaluates the methodologies employed to realize them. In addition, it examines potential research directions that could extend this study.

7.1 Realization of Contributions

The objective of this thesis was to make a significant impact in the domain of SLAM by improving autonomous navigation in unknown environments (see section 1.3). This section examines how those contributions were implemented during the research process.

1. **Advancements in the SLAM-GMapping Method:**
    - The EMMI approach was customized to meet our particular requirements, resulting in the creation and implementation of a novel method for a fast, stable and reliable estimation of pose certainty.
    - Integrating Full Map Posteriors into the EMMI computation improved the environmental representation, thereby improving both the accuracy of the map and the precision of the pose certainty metrics.
    - The ability to publish individual particles along with their maps enhanced analysis and debugging, by offering comprehensive insights into the SLAM framework, and serves as a valuable asset for future developments.
    - Generating an averaged map based on all particle predictions effectively combines environmental data, providing a more accurate representation of the probability field and optimizes the decision-making processes.
2. **Advancements in Dynamic DWA Planning:**
    - The integration of the PBF within the DWA planner allows real-time modification of the robot's local trajectory to enhance pose accuracy.
    - Achieving a balance between reaching the goal and ensuring pose reliability, leveraging the GM pose precision metric, facilitates fast but robust navigation.

- The pose improvement actions lead to a more informative map and a notable reduction in navigation errors.

3. **Empirical Validation and Practical Applications:**
   - Experimental evaluations in both virtual simulations and actual field environments validated the proposed improvements, demonstrating noticeable enhancements in navigation accuracy.
   - The practical applications of these advancements were confirmed during real-world tests, showing their potential for broad use in service robotics and autonomous systems.

7.2 Challenges and Limitations of the Study

Despite the promising outcomes, several limitations and challenges were encountered:

1. **Time Efficiency:** The modified planner often requires more time to reach the goal compared to the standard DWA planner. This increase in travel time can be problematic in scenarios where time efficiency is crucial. An optimization of the PBF parameters or strategy may be necessary to improve time efficiency without compromising pose accuracy.

2. **Reliability of Odometry vs. Scan Measurements:** The PBF aims to enhance pose estimation by encouraging the robot to revisit previously mapped areas to collect more scan data. However, this approach assumes that the scan data is more reliable than odometry. In cases where odometry is more accurate, re-visiting actions based on scan data could worsen the pose estimate rather than improve it.

3. **Navigation in Short-Range Environments:** In environments where the destination is close, the direct path typically does not accumulate significant odometry errors. However, the PBF might cause the robot to take a longer route to enhance pose certainty, inadvertently introducing more odometry errors. In such scenarios, additional movement can negate the benefits of improved pose estimation, leading to an overall reduction in navigation performance.

7.3 Future Research Directions

The advancements presented in this study highlight the potential for future research in the field of SLAM and autonomous navigation. Based on the challenges and limitations of this work future investigations might consider several enhancements of the techniques:

1. **Refinement of Pose Certainty Feedback:**
   While the dynamic feedback mechanism to adjust the PBF has proven effective, optimizing its parameters could significantly enhance performance. Further refinement of the pose certainty estimation is also possible, which could lead to even more reliable navigation decisions.

2. **Assessment of Necessity for PBF Actions:**
   As highlighted in Challenges and Limitations of the Study, there are scenarios where the standard DWA planner may outperform the new PBF version, especially when direct routes are short or odometry is particularly reliable. Future research could focus on developing algorithms that autonomously decide when to bypass pose improvement actions, perhaps through advanced parameter tuning or real-time environmental assessment.

3. **Application of Machine Learning:**
   Machine learning techniques could be used to predict the necessity and potential effectiveness of pose improvement actions more accurately. Current methods estimate the pose enhancement potential by counting known cells during a raycast; however, this could be refined by considering the informational quality of surfaces, such as flat walls, which provide limited localization cues.

4. **Optimization of Revisiting Strategies:**
   Continually revisiting the same areas can be redundant. A novel approach could involve detecting and recording areas that have been used for re-localization, assigning them a lower priority for future revisiting actions. This would help in diversifying the areas used for enhancing pose certainty and prevent over-reliance on specific spots.

5. **Cross-Platform Integration:**
   Testing the integration of the PBF with various planners and SLAM systems across different robotic platforms could provide insight into its adaptability and

effectiveness in a wider range of applications. This could facilitate the development of more versatile and robust autonomous navigation systems suitable for diverse operational environments.

These directions not only aim to refine the enhancements made in this thesis, but also explore new possibilities for advancing the state-of-the-art in robotic navigation and mapping.

7.4 Concluding Remarks

This thesis addresses the relatively underexplored challenge of balancing goal-directed navigation with the enhancement of real-time pose estimation in unknown environments. By introducing dynamic modiications to the Dynamic Window Approach (DWA) planner, particularly by incorporating the PoseBoost cost function and advanced mapping techniques (FMP), this research demonstrates improvements in both robotic pose precision and navigation reliability. These enhancements validate the efficiency of adaptive pose correction mechanisms, establishing them as essential components for precise autonomous navigation.

Experimentation and validation processes, conducted in simulated and real-world scenarios, confirm that the proposed approaches considerably enhance the robustness and efficiency of autonomous systems, demonstrating their practicality in challenging environments. These advancements not only expand the capabilities of robotic navigation, but also create advancements for more reliable robotic assistants.

Looking ahead, the contributions of this thesis suggest promising directions for further research, with applications in various areas of robotics. The insights provided here are intended to inspire further innovation and set the stage for more adaptable, reliable, and efficient autonomous systems.

8 Appendices

Remaining Results

This section presents additional results from section 6.4. Of particular interest is the observation that, despite the improved pose certainty in the PBF planner, as reflected in the Euclidean distance development (see Figure 21), the results from the EMMI approach appear less favorable (see Figure 24). This discrepancy can be explained by the way EMMI is calculated (see Equation 15). Specifically, revisiting actions, while improving pose certainty, also result in a larger portion of the map being explored. In case fewer cells are explored, the average number of scan measurements per cell increases, resulting in a higher average information value (EMMI), which can appear less favorable in comparison to other metrics.

Real World

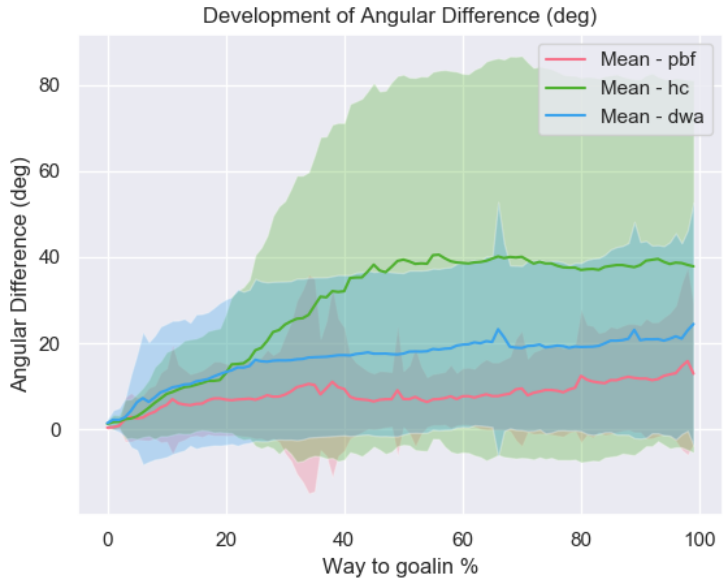

Figure 23: Development of the Rotational Error in Euler Degrees in real world tests.

Figure 24: Development of EMMI in real world tests. Initial values at zero can be disregarded, as entropy calculation had not yet commenced at that stage.

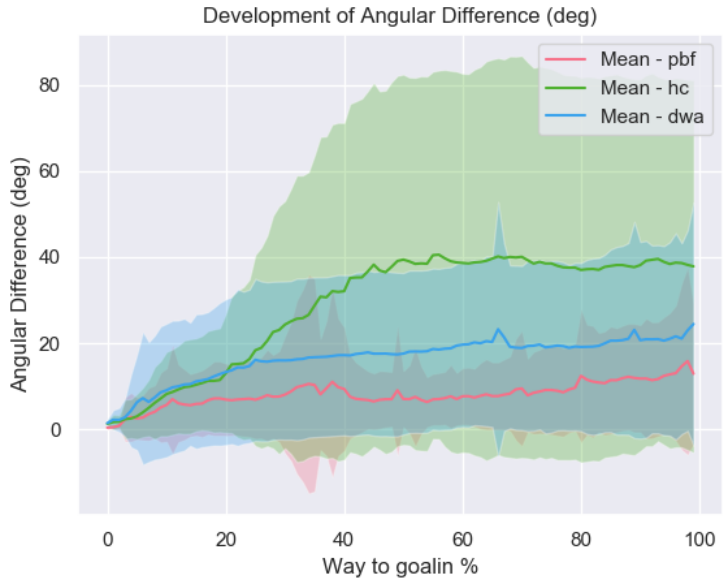

Figure 25: Development of the Rotational Error in Euler Degrees in simulation tests.

Figure 26: Development of EMMI in simulation tests. Initial values at zero can be disregarded, as entropy calculation had not yet commenced at that stage.

Configuration used for testing

Complete ROS configuration files used in our experminents. Parameters set to default values are not listed.

DWA-Planner Parameters

```
base_local_planner: dwa_local_planner/DWAPlannerROS

DWAPlannerROS:
  acc_lim_th: 0.05
  acc_lim_x: 0.05
  acc_lim_y: 0.0

  max_vel_x: 0.25
  min_vel_x: 0.0

  max_vel_trans: 0.25
  min_vel_trans: 0.0

  max_vel_theta: 0.5
  min_vel_theta: 0.0

  sim_time: 5.0
  sim_granularity: 0.25
  controller_frequency: 0.75

  path_distance_bias: 12

  oscillation_reset_dist: 0.1

  scaling_speed: 1.0

  vx_samples: 20
  vy_samples: 0
  vtheta_samples: 30

  xy_goal_tolerance: 0.2
  yaw_goal_tolerance: 0.2

  # New PBF parameters
  pose_boost_scale: 400
  pose_boost_min_distance: 0.3
```

```
  pose_boost_visualize_area: true
  pose_boost_visualize_cells: false
  pose_boost_laser_topic: "scan_front"
  pose_boost_distance_after_oscillation: 0.4
```

Listing 8.1: `local_planner.yaml`

MoveBase Parameters

```
planner_frequency: 2.0
planner_patience : 0.1

controller_frequency: 10.0
controller_patience :  3.0

oscillation_timeout : 10.0
oscillation_distance:  0.2

# New PBF parameters
oscillation_timeout_pose_boost : 2.5
```

Listing 8.2: `move_base.yaml`

GMapping Parameters

```
base_frame: base_footprint

map_update_interval: 1.0
maxUrange: 6
maxRange : 6
sigma     : 0.003
lstep     : 0.05
astep     : 0.05

linearUpdate     :  0.1
angularUpdate    :  0.2

# Parameters specifically for FMP:
publishParticles: true
publishRawAndProbMap: true

alpha0: 0.023154724941509084
```

```
beta0: 0.6156273138207687

best_entropy_threshold: 4
```

Listing 8.3: `gmapping.yaml`

Particles and Maps Service

Example output when calling the `Particle and Map Service` (see subsubsection 5.2.2)

```
socrob@dolores:~ $ rosservice call /particles_and_maps "n_particles:
  data: 3"

p_poses:
  -
    position:
      x: 22.611448397393296
      y: -1.26666752479945
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.9398000977226229
      w: 0.3417247083846126
  -
    position:
      x: 22.61096903419241
      y: -1.2685044173386688
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.9398808547943723
      w: 0.34150253116338697
  -
    position:
      x: 22.61232260228084
      y: -1.2679536840141359
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.9397988637075507
```

```
      w: 0.3417281021133563
p_weights:
  -
    data: 0.3581606212647417
  -
    data: 0.31146603999753597
  -
    data: 0.33037333873772234
probability_maps:
  -
    header:
      seq: 0
      stamp: ..........
```

Listing 8.4: Output of `Particle and Map Service`

Bibliography

Muhammad Farhan Ahmed, Khayyam Masood, Vincent Fremont, and Isabelle Fantoni. Active slam: A review on last decade. *Sensors*, 23:8097, 2023. doi: 10.3390/s23198097.

José Arce y de la Borbolla. Full map posterior slam in ros. `https://github.com/joseab10/FMP_gmapping/blob/master/doc/Report.pdf`, 2021. Accessed on: Aug. 28, 2024.

Jose Luis Blanco, J.-A Fernández-Madrigal, and Javier González-Jiménez. A novel measure of uncertainty for mobile robot slam with rao blackwellized particle filters. *I. J. Robotic Res.*, 27:73–89, 01 2008. doi: 10.1177/0278364907082610.

J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965. doi: 10.1147/sj.41.0025.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *CoRR*, abs/1606.05830, 2016. URL `http://arxiv.org/abs/1606.05830`.

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. *CoRR*, abs/2004.05155, 2020. URL `https://arxiv.org/abs/2004.05155`.

Eugen Dedu. Bresenham-based supercover line algorithm. `http://eugen.dedu.free.fr/projects/bresenham/`, 2001. Accessed on: Aug. 28, 2024.

A. Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, 2000. URL `https://api.semanticscholar.org/CorpusID:2948186`.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. doi: 10.1109/2.30720.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *Robotics & Automation Magazine, IEEE*, 4:23 – 33, 04 1997. doi: 10.1109/100.580977.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Improving Grid-based SLAM with Rao-Blackwellized Particle Filters By Adaptive Proposals and Selective Resampling*, pages 2432–2437, 01 2005. doi: 10.1109/ROBOT.2005.1570477.

Justin Hart, Alexander Moriarty, Katarzyna Pasternak, Johannes Kummert, Alina Hawkin, Vanessa Hassouna, Juan Diego Pena Narvaez, Leroy Ruegemer, Leander von Seelstrang, Peter Van Dooren, Juan Jose Garcia, Akinobu Mitzutani, Yuqian Jiang, Tatsuya Matsushima, and Riccardo Polvara. Robocup@home 2024: Regulations on the organization of the competition. `https://github.com/RoboCupAtHome/RuleBook/releases/tag/2024.1`, 2024.

Lukas Luft, Alexander Schaefer, Tobias Schubert, and Wolfram Burgard. Closed-form full map posteriors for robot localization with lidar sensors. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 6678–6684. IEEE, September 2017. doi: 10.1109/iros.2017.8206583. URL `http://dx.doi.org/10.1109/IROS.2017.8206583`.

Lukas Luft, Alexander Schaefer, Tobias Schubert, and Wolfram Burgard. Detecting changes in the environment based on full posterior distributions over real-valued grid maps. *IEEE Robotics and Automation Letters*, 3(2):1299–1305, April 2018. doi: 10.1109/LRA.2018.2797317.

D.Q. Mayne, James Rawlings, Christopher Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 06 2000. doi: 10.1016/S0005-1098(99)00214-9.

M. Montemerlo and S. Thrun. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *IEEE Rob. Autom.*, 13:99–110, 01 2007. doi: 10.1007/978-3-540-46402-0_4.

MoveBase Wiki. Ros services and actions. `http://wiki.ros.org/move_base`, 2024. Accessed on: Aug. 26, 2024.

NaturalPoint Corporation. Optitrack wiki. `https://v21.wiki.optitrack.com/index.php?title=OptiTrack_Wiki`, 2022. Accessed on: Nov. 18, 2024.

S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2, 1993. doi: 10.1109/ROBOT.1993.291936.

Maria Ribeiro and Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties, 2004. Instituto Superior Técnico.

RoboCup@Home. Available online. `https://athome.robocup.org/`, 2024. Accessed on: Sep. 12, 2024.

Open Robotics. Available online:. `https://gazebosim.org/about`, 2024. Accessed on: Oct. 09, 2024.

ROS Wiki. Ros concepts. `https://wiki.ros.org/ROS/Concepts`, 2024a. Accessed on: Aug. 25, 2024.

ROS Wiki. Ros introduction. `http://wiki.ros.org/ROS/Introduction`, 2024b. Accessed on: Aug. 25, 2024.

Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *Coastal navigation-mobile robot navigation with uncertainty in dynamic environments*, volume 1, pages 35 – 40 vol.1, 02 1999. ISBN 0-7803-5180-0. doi: 10.1109/ROBOT.1999.769927.

Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. In *2015 European Control Conference (ECC)*, pages 3352–3357, 2015. doi: 10.1109/ECC.2015.7331052.

Robert Sim and Nicholas Roy. Global a-optimal robot exploration in slam. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 661–666, 2005. doi: 10.1109/ROBOT.2005.1570193.

source code. dwa_planner.cpp. `https://github.com/ros-planning/navigation/blob/9ad644198e132d0e950579a3bc72c29da46e60b0/dwa_local_planner/src/dwa_planner.cpp#L170`, 2009.

Cyrill Stachniss. Grid mapsn, 2012a. URL `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam11-gridmaps.pdf`. AIS - Robot Mapping - WS20/21 Accessed on: Nov. 14, 2024.

Cyrill Stachniss. Short introduction to particle filters and monte carlo localization, 2012b. URL `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/`

pdf/slam09-particle-filter.pdf. AIS - Robot Mapping - WS20/21 Accessed on: Nov. 14, 2024.

Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters, 06 2005.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005. ISBN 9780262201629. Accessed on: Nov. 08, 2024. ProQuest Ebook Central.

Oleksii Zhelo, Jingwei Zhang, Lei Tai, Ming Liu, and Wolfram Burgard. Curiosity-driven exploration for mapless nhttps://irs-group.github.io/socrobwebsite/avigation with deep reinforcement learning, 2018. URL https://arxiv.org/abs/1804.00456.

ToDo Counters

To Dos: 1; 1

Parts to extend: 0;

Drafts: 2; 1, 2

Comments: 10; 1, 2, 3, 4, 5, 6, 7, 8, 9, 10