ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision and
Geometry Group
Prof. Marc Pollefeys

Semester Thesis

Active Mapping with Known Floor Plans for Precise Re-Localization

Autumn Term 2023

Supervised by:
Eric Vollenweider
Jeff Delmerico
Juan Nieto

Author:
Elisa Hoskovec

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

- (●) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].
- ( ) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].
- ( ) I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Active Mapping with Known Floor Plans for Precise Re-Localization

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Hoskovec	Elisa

With my signature I confirm the following:

- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zurich, 29.02.2024	

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard

Contents

Abstract

Cutting-edge devices like augmented reality headsets or micro aerial vehicles require precise localization to complete a variety of tasks. An approach to enable an agent to localize itself in a particular area is to first create a map and then re-localize itself in the created map. The autonomous creation of such maps is a non-trivial and widely researched problem. The numerous active exploration and mapping approaches may pursue different objectives, such as accurate 3D reconstruction, surface inspection, or the aforementioned localization, but often share a framework structure.
Many traditional approaches do not consider any prior information. However, in real-world scenarios, 2D floor plans are widely available and can be used to influence the decision of the planner with the information at hand.
This project aims to implement an exploration planner that maps an environment with the goal of precise re-localization. To that end, the frequent availability of 2D floor plans is exploited for a more informed exploration and mapping process. A novel exploration planner is introduced in four variants trying to revisit drifty regions during the mapping process to increase the localization ability.
The four variants of the proposed planner are compared against three different planners using a custom evaluation framework quantifying the re-localization accuracy of the created map. Further analysis is conducted regarding the precision of pose estimation during the mapping process and the efficiency of the exploration. The analysis shows that the novel exploration planner improves the re-localization accuracy compared to the other planners. However, the prior information gained from the 2D floor plan could not be exploited to its full potential.

Symbols

Symbols

$\mathbf{p}$	pose
$\mathbf{x}$	position
R	rotation
g	gain function
c	cost function
v	value function

Acronyms and Abbreviations

SLAM	Simultaneous Localization and Mapping
IMU	Inertial Measurement Unit
VIO	Visual-Inertial Odometry
MAV	Micro Aerial Vehicle
RRT	Rapidly-exploring Random Tree
TSP	Travelling Salesperson Problem

Chapter 1

Introduction

Active exploration of unknown spaces have been subject to extensive research in the past years. The underlying problem is to efficiently plan and execute trajectories through an unknown environment for an agent to explore the region of interest at hand. Especially sampling-based approaches have gained popularity in this field and are successfully applied in a wide range of scenarios, like object search [1], classification [2] and infrastructure modeling [3].

Traditional sampling-based approaches often assume no prior knowledge of the area of interest. However, in a real-world setting, 2D floor plans are often available and can therefore be utilized to gain prior information on the shape and structure of the region. This information can be used to not only make more informed decisions to cover the whole environment efficiently, but also help localize the agent against the floor plan.

This project introduces an active mapping policy aiming to construct a map for precise re-localization. It utilizes floor plans for an efficient and more informed mapping of unmapped spaces. An evaluation framework is created to compare the success of our method to three other planners.
The work builds on a former semester project conducted by Baumgartner [4]. His work includes creating an active mapping simulation and evaluation framework based on the work of Schmid et al. [5], which among other contributions implements a structure for sampling-based exploration planners. In contrast to most work done in the field of active exploration and mapping, the main focus of the framework and therefore this project is the re-localization quality of the created map rather than the reconstruction quality. To contribute to this goal, this project extends the work of Baumgartner mainly in three different ways:

- The evaluation framework introduced by Baumgartner is extended to support multi-frame localization. To that end, a new evaluation metric is defined capturing the re-localization error of a trajectory segment.
- Based on a suggestion by Baumgartner, a drift-aware planner is implemented focusing on its re-localization quality. It is compared against other state-of-the-art planners to evaluate the success of its focus.
- The use of a floor plan is added to the framework. Since floor plans are often available in real-world scenarios, the information gained can be used for a better incentive on planning trajectories.

All contributions are available open-source [1].

[1] `https://github.com/ehosko/Active-Mapping-with-Known-Floor-Plans-for-Precise-Re-Localization`

It is important to note, that while the current framework is simulation-based, the simulation environment is interchangeable with real devices. This might be a robot like a micro air vehicle (MAV) mapping the space autonomously or a mixed-reality device navigating the user through the environment.

1.1 Related Work

Different approaches have been developed to fulfill the objective of autonomous exploration. Apart from reinforcement learning, most of these methods can be separated into frontier- and sampling-based approaches. Frontier-based methods introduced by Yamauchi [6] look for the boundaries between mapped and unmapped space and choose these frontiers as goals to iteratively explore the whole space. Sampling-based methods sample different viewpoints and act in a next-best-view fashion [7] often following a gain formulation [8].

Many traditional sampling-based approaches make use of rapidly-exploring random trees (RRT) [9] to sample, store and execute trajectories [10],[1],[11]. Schmid et al. [5] utilize RRT*, a refined version of RRT introduced by Karaman and Frazzoli [12] and proven to be asymptotically optimal. They extend the algorithm further by preserving a single tree throughout the exploration instead of discarding the tree and extending it from scratch after every iteration.

Previous work concerning the localization quality of the built map includes the approach introduced by Merzić et al. [13]. They propose an algorithm predicting the localization success for a given pose by computing the observable landmarks from that pose and weighing them based on their likeliness to be observed. Zhang and Scaramuzza [14] use Fisher Information to quantify how much the information of an image taken at a certain pose benefits localization. By separating the rotation-invariant and rotation-dependent parts of the Fisher Information, they achieve an efficient approach to calculate the Fisher Information during mapping.

The work of Boniardi et al. [15] proposes a system for long-term localization using CAD floor plans. They combine graph-based mapping techniques and Bayes filtering to maintain a globally consistent map. Recent work conducted by Gao and Kneip [16] introduces an efficient approach for drift-free localization given a floor plan. In a first stage, a single frame is localized by comparing extracted features to the geometric elements of the floor plan. The pose estimate is then refined by jointly aligning multiple frames. Both approaches use LiDAR sensors for accurate localization. Watanabe et al. [17] propose a robust localization system given a floor plan and a depth camera. Their algorithm estimates the robot's position by matching the point cloud from the depth image with the floor plan.

Chapter 2

Framework

The initial framework of this project is taken from a semester project conducted by Baumgartner [4]. The repository is publicly available [1]. Apart from the *Evaluation Framework*, the entire project is integrated and works within the Robot Operating System (ROS) [2].The main body of the framework implements a simulation of a MAV mapping an unknown environment using multiple planners. Additionally, an evaluation framework provides feedback in the re-localization capabilities of the used planner. Figure 2.1 visualizes an overview of the framework with the individual building blocks. The following sections describe the different parts of the framework.

Figure 2.1: **Framework Overview:** The sketch outlines the individual components of the framework and their communication with each other. The Figure is taken from [4].

[1] `https://github.com/michbaum/optimal_active_guidance_in_mixed_reality_using_prior_floorplans`

[2] `https://www.ros.org/`

2.1 Exploration Planner

The planner node is based on the work of Schmid et al. [5]. It determines a trajectory to explore based on a specific exploration planner. The underlying method is explained in more detail in chapter 3. The planner operates on the pose estimates from the SLAM system, which might differ significantly from the ground truth. Therefore, the trajectories from drifty poses have to be transformed into the ground truth simulation frame by a trajectory translation module.

2.2 Simulation Environment

Given the trajectory from the planner, a MAV is simulated to follow the trajectory. The simulation environment is divided into two parts: a physics simulation and a graphics simulation. In a real-world scenario, this part is changed to a real device.

2.2.1 Physics Simulation

The physics simulation uses Gazebo[3] with the RotorS package [18] to realistically simulate the MAV's behavior conditioned on maximum yaw rate, acceleration, and speed. The simulated inertial measurement unit (IMU) is passed on to the SLAM system.

2.2.2 Graphics Simulation

NVIDIA Isaac Sim [4] is utilized as the graphics simulator. A custom interface [5] enables the embedding in the project's framework. The graphics simulator renders the sensor data along the given trajectory resulting in RGB and depth images.

2.3 SLAM System

The simultaneous localization and mapping (SLAM) system takes the simulated IMU data and simulated images as input and computes a pose estimate of the agent. To that end, the maplab framework by Schneider et al. [19] and Cramariuc et al. [20] is utilized. Maplab implements a collection of important tools for mapping and localization. The visual-inertial mapping front-end for maplab - ROVIOLI (Robust Visual Inertial Odometry with Localization Integration) - builds a map from raw visuals and inertial data by tracking features and utilizing the visual-inertial odometry framework ROVIO [21] to estimate transformations from the raw global pose estimates and odometry constraints. The resulting map contains the keyframe poses, inertial constraints, visual features and landmarks. Furthermore, maplab offers the ability to localize an input image against the map by looking for loop closures based on binary descriptors. This feature is mainly used for the evaluation framework.

2.4 Evaluation Framework

The evaluation framework aims to quantify the re-localization success of the planner. To that end, an adapted evaluation metric is implemented using a custom maplab plugin as well as the trajectory alignment library provided by Zhang and Scaramuzza [22]. The evaluation metric is discussed in detail in chapter 5.

[3] `https://gazebosim.org/home`
[4] `https://developer.nvidia.com/isaac-sim`
[5] `https://github.com/michbaum/isaac_cv_ros`

Chapter 3

Method

In this project, a new exploration planner is introduced. The *drift-aware planner* aims to be aware of the accumulated drift of the predictions from the SLAM system and account for it. As mentioned above, the project is initially based on the work of Schmid et al. [5]. Their code is made publicly available[1] and provides an adaptive framework to implement different exploration planners. The planners consist of two main components: a trajectory generator and a trajectory evaluator. They are closely intertwined. The trajectory generator samples new poses as nodes V_i for the agent to visit and connects them to create trajectories. The trajectory evaluator assigns a value to each of these nodes V_i, by combining a gain and cost function. The agent's trajectory is then determined by following the path with maximal value.

3.1 Trajectory Generator

The proposed planner uses the trajectory generator implemented by Schmid et al. [5]. To that end, a tree structure is used to continuously build and execute trajectories. The tree $G = (V, E)$ represents the sampled nodes $\{V_i\}_{i=1,\dots n}$ and edges E. The tree's root is determined by the agent's position. Each node V_i contains a cost c_i, gain g_i, value v_i and a corresponding trajectory τ_i. The tree is continuously expanded by sampling a new pose and adding it as a node to the tree by applying an algorithm similar to RRT* by Karaman and Frazzoli [12], an extension of the RRT introduced by LaValle [9]. The new node considers all nodes of the tree within the distance of l_{max} and chooses the node, which maximizes its value as a parent. The remaining neighbors are rewired, to increase their value. If the goal position is reached, the tree is updated by setting that node to the root and re-estimating the values based on the change. The next trajectory is then determined by choosing the one with the maximal value.

3.2 Trajectory Evaluator

The trajectory evaluator assigns value v_i to each node V_i by combining a cost and gain formulation. The following section describes the policy used by the drift-aware planner.

3.2.1 Drift-Aware Policy

The proposed drift-aware planner (DAP) adapts the reconstruction planner introduced by Schmid et al. [5]. Since the reconstruction planner is designed with the

[1] `https://github.com/ethz-asl/mav_active_3d_planning`

goal of reconstruction, the drift-aware planner implements a new cost function to serve the purpose of building a map for precise re-localization. To achieve this goal, the planner considers the drift from the pose predictions of the VIO system. It tries to account for it by remaining in the region revisiting poses and therefore gaining more information, to reduce the drift before moving on. The following sections outline the computation of g_i, c_i, and v_i.

Gain Function

The gain formulation is determined in two steps. First, the underlying computation is derived from a frontier-based formulation, introduced by Yamauchi [6].

$$g_{front}(\xi) = \sum_{m \in Vis(\xi)} \begin{cases} 1, & \text{if } m \in U \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

The set U includes all unobserved voxels near an observed surface. The view ξ is determined by the agent's position $\mathbf{x}$ and yaw φ. Therefore, the gain of a view ξ, is determined by counting the unobserved voxels near a surface of all visible voxels in that view, denoted by $Vis(\xi)$.
The second step for the gain computation is maximizing the gain over the agent's yaw φ.

$$g(V_i) = \max_{\varphi} g(\xi(\mathbf{x}, \varphi)) \tag{3.2}$$

This formulation is introduced by Selin et al. [23].

Cost Function

The cost function consists of two parts. The first part, $c_{temporal}(V_i)$, measures the time it takes to execute the trajectory corresponding to V_i. The time is estimated with maximal velocity and constant sampling rate.

$$c_{temporal}(V_i) = t_{end}(\tau(V_i)) - t_{start}(\tau(V_i)), \tag{3.3}$$

This formulation is taken from the implementation by Schmid et al. [5].
The second part of the cost function is designed to account for the drift in the prediction of the agent's pose from the VIO system. To that end, the estimated pose of the agent is compared to its ground truth pose to measure the drift. The cost function adds the translation error and the angular error between the poses.

$$c_{pose\_err}(V_i) = ||\mathbf{x}(V_i) - \tilde{\mathbf{x}}(V_i)||_2 + \beta * \theta^2 \tag{3.4}$$

$\tilde{\mathbf{x}}(V_i)$ represents the ground truth pose of the agent and θ the angle between the two poses. β is a hyperparameter weighing the impact of the angular difference. By accumulating the cost along the trajectory, this formulation aims to steer the agent to previous positions with high drift rather than exploring further.
The final cost formulation combines the two components.

$$c_{drift}(V_i) = (1 - \lambda)c_{temporal}(V_i) + \lambda c_{pose\_err}(V_i) \tag{3.5}$$

λ is a hyperparameter regulating the trade-off between the two parts.

Value Function

The value function combines the gain and cost to measure the utility of each node. The following formulation is again taken from the implementation of Schmid et al. [5].

$$v(V_i) = \max_{V_k \in \text{subtree}(V_i)} \frac{\sum_{V_l \in R(V_k)} g(V_l)}{\sum_{V_l \in R(V_k)} c(V_l)} \tag{3.6}$$

Subtree(V_i) indicates the tree originating from V_i and $R(V_k)$ is the sequence of nodes connecting the root of the tree to V_k. Therefore, the value function captures the global context of each node by accounting for all subsequent children of any given node.

However, the cost function utilizes the agent's ground truth pose, which is not given in a real-world scenario. Therefore, another information source is needed to approximate this value and therefore make this policy applicable in real life.

3.3 Floor Plan Incorporation

If a floor plan of the environment is available, we can use it to guide the agent more informed. We utilize the floor plan in two different ways: On the one hand, we compute an a priori optimal trajectory through the area of interest, which can be used as a guideline in the later active mapping process. On the other hand, by refining the pose estimation of the SLAM system with the floor plan, we approximate its ground truth position.

3.3.1 Reference Trajectory

The floor plan is used to lay out a reference trajectory to travel through the whole area of interest efficiently. To that end, a graph is built by sampling multiple locations in the floor plan as nodes and connecting them with edge weights corresponding to traveling distance. Given this graph, we want to find the optimal trajectory to visit all nodes - this is equivalent to the Traveling Salesperson Problem (TSP).

Building the Graph

First, the given floor plan needs to be represented as a graph $G_{plan} = (V^{plan}, E^{plan})$. To build the graph, the floor plan is first transformed into an occupancy grid. n grid cells are randomly sampled from the grid. If the sampled cell is not occupied, the position is added as a vertex V_i^{plan} to the resulting graph. The vertices are connected with Bresenham's line algorithm. Therefore, an edge is added between two vertices, if a line can be drawn between the two positions without intersecting a wall or any obstacle. The edge weights are determined by the Euclidean distance between the two vertices.

Traveling Salesperson Problem

Finding the shortest path through the floor plan and therefore the graph, while visiting each node exactly once, is known as the Traveling Salesperson Problem. There are several libraries implementing the solution to TSP. In this project, the LKH solver, introduced by Helsgaun [24], is used to find the optimal path through the floor plan.

Cost Formulation

The policy needs to be adapted to incorporate the knowledge of an optimal trajectory. To that end, an additional term can be added to the cost formulation of the policy. The goal hereby is to reward the agent for following the optimal trajectory. Therefore, once the agent visits the next vertex of the optimal path, it is marked as visited and the new cost is computed with respect to the next vertex in the optimal trajectory.

$$c_{traj\_err}(V_i) = ||\mathbf{x}_{proj}(V_i) - \mathbf{x}_{traj}(V_k^{plan})||_2 \tag{3.7}$$

$\mathbf{x}_{proj}(V_i)$ is the position estimate of the node V_i from the VIO system projected onto the floor and $\mathbf{x}_{traj}(V_k^{plan})$ is the position of the next vertex (V_k^{plan}) in the optimal trajectory.
However, the agent might choose to move away from the next vertex, if the policy decides on optimizing other costs momentarily. In that case, if the distance to the next vertex exceeds a certain threshold, the path is rewired. To that end, the next vertex is found by looking for the nearest vertex (V_l^{plan}) to (V_i), which has not been visited yet. The optimal trajectory is given as before, but following the path starting from the vertex (V_l^{plan}).

3.3.2 Floor Plan Localization

The floor plan can be utilized to improve the pose estimation of the VIO system and therefore approximate a ground truth pose for the cost formulation. To that end, a localization pipeline with the floor plan is implemented, based on the algorithm proposed by Watanabe et al. [17]. It combines the information from the floor plan with the point cloud generated by the depth camera and the pose estimate of the VIO system to improve the estimation of the agent's pose. The localization is therefore local and not global and the origin pose of the agent on the floor plan is assumed to be known. The algorithm can be divided into three parts: generating the source cloud from the depth image, generating the target cloud from the floor plan, and matching the two point clouds to estimate the agent's pose on the floor plan. The following algorithm 1 gives an overview of one localization iteration.

Algorithm 1 Floor Plan Localization

Require: $M, Z_t, \mathbf{x}_t, \mathbf{q}_t$
  $P_{source} \leftarrow \text{sourceCloudGen}(Z_t, \mathbf{q}_t)$
  $\Delta\mathbf{x}_{t-1,t} \leftarrow ||\mathbf{x}_t - \mathbf{x}_{t-1}||_2$
  if $\Delta\mathbf{x}_{t-1,t} < d_l \wedge \Delta\theta_{t-1,t} < \theta_l$ **then**
    return
  end if
  $\hat{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \Delta\mathbf{x}_{t-1,t}$
  $P_{target} \leftarrow \text{targetCloudGen}(M, \hat{\mathbf{x}}_t, \mathbf{q}_t)$
  $\tilde{\mathbf{x}}_t \leftarrow \text{floorPlanMatching}(\hat{\mathbf{x}}_t, P_{source}, P_{target})$

Here, M denotes the floor plan, and Z_t is the depth cloud generated by the depth camera. x_t and q_t are the estimated position and orientation of the agent. d_l and θ_l are thresholds for the translation and rotation change between time $t-1$ and t.

Source Cloud Generation

To generate the source cloud P_{source}, the point cloud obtained from the depth image is first transformed using the known camera-to-world transformation and

pose estimation from the VIO system. The point cloud is then projected onto the floor by computing the normal vector of the floor based on the estimated agent's rotation. The resulting point cloud is stored as P_{source}.

Target Cloud Generation

First, all the contours of the floor plan are obtained using the algorithm of Suzuki et al. [25]. The contours are then divided into segments of maximal length l_{max} and the endpoints of each segment are stored as candidate points $A = \{a_j\}_{i=1}^{M}$.
To generate the target cloud P_{target}, the candidate points are filtered. To that end, each point from the candidate points is evaluated whether it lies within the Field of View (FoV) by ray casting from the agent's estimated pose. The visible points are stored as valid points $F = \{f_k\}_{k=1}^{L}$. These valid points are then interpolated by applying A* [26] between every two successive points f_k and f_{k+1}, where $||f_k - f_{k+1}||_2 < l_{max}$. If a path between the two points is found, all the points in the path are added to the resulting target cloud P_{target}.

Cloud Matching

Before matching the two point clouds, P_{source} and P_{target} are independently thinned out. To that end, a grid is generated over the point cloud with a given cell size $l \times l$. The points in each voxel are represented by their centroids, which are stored in the resulting filtered point cloud.
After the filtering process, P_{source} and P_{target} are matched with the Generalized Iterative Closest Point (GICP) algorithm, proposed by Segal et al. [27]. GICP computes correspondences between the two clouds and calculates a transformation T, that minimizes the distance between the corresponding points in P_{source} and P_{target}. The final agent's position on the floor plan $\tilde{\mathbf{x}}_t$ is then estimated by applying the transformation on the prior position estimate $\mathbf{x}_t$.

$$\tilde{x}_t = T x_t \tag{3.8}$$

Furthermore, the estimated orientation is refined by the rotation given by the transformation matrix.

3.3.3 Drift Aware Planner using a Floor Plan

By using the floor plan instead of the ground truth, the drift-aware planner is slightly adapted. Its underlying policy only differs in the cost formulation, the gain and the value are computed as described in Equation 3.2 and Equation 3.6 respectively. The resulting cost reads as follows:

$$\begin{aligned} c_{drift}(V_i) &= \lambda c_{pose\_err}(V_i) + (1-\lambda) c_{traj\_err}(V_i) \\ c_{pose\_err}(V_i) &= ||\mathbf{x}_{proj}(V_i) - \mathbf{x}_{floor}(V_i)||_2 + \beta\theta^2, \end{aligned} \tag{3.9}$$

where $c_{traj\_err}(V_i)$ is given in Equation 3.7.$\mathbf{x}_{floor}(V_i)$ is the refined position estimate on the floor plan approximating the ground truth position. The angular error θ is computed between the estimated and refined orientation. Since $c_{temporal}(V_i)$ and $c_{traj\_err}(V_i)$ are both a measure of efficiency, only the latter one is considered in this formulation.

Chapter 4

Experiments

This chapter describes the experiments conducted to evaluate the performance of the proposed planner. To that end, multiple planners are used to map two different environments. Each simulation is repeated five times to account for the probabilistic nature of the planners.

4.1 Planners

The code from Schmid et al. [5] provides three different exploration planners: the receding horizon NBV planner (RH-NBVP), the autonomous exploration planner (AEP), and their newly introduced reconstruction planner (RCP). All three planners are sampling-based and use the gain formulation described in chapter 3.
It is important to note, that all three planners are not designed for re-localization purposes, but rather for exploration and reconstruction.

4.1.1 Receding Horizon NBV Planner

The RH-NBVP is based on the work of Bircher et al. [10]. In contrast to the other planners, it only uses RRT [9] to locally expand trajectory candidates and is rebuilt after reaching the first node of the best trajectory.

4.1.2 Autonomous Exploration Planner

The AEP is proposed by Selin et al. [23] and builds on the RH-NBVP. This planner utilizes RRT more globally to reuse tree branches and grow the tree continuously.

4.1.3 Reconstruction Planner

Schmid et al. [5] introduce the RCP. It differs from the drift-aware planner in its cost function by only considering the estimated time of a trajectory as the cost associated with a node V_i.

4.1.4 Drift-Aware Planner

The drift-aware planner is described in detail in chapter 3. To evaluate the impact of the floor plan, multiple variants of this planner are considered. On the one hand, we consider the ground truth to be known - a non-realistic scenario. In that case, we use the drift-aware planner (DAP) as introduced in subsection 3.2.1 and the drift-aware planner with ground truth, but also following the optimal trajectory provided by the floor plan (DAP-TSP). On the other hand, a real-life scenario is

considered by using the floor plan localization as an approximation to the ground truth. In this scenario, we also consider the drift-aware planner without (DAFP) and with the optimal trajectory (DAFP-TSP). For all variants, λ is set to 0.5.

4.2 Environment

Two different environments are considered for the agent to map. Each of them poses different difficulties for the robot.

4.2.1 Maze

The maze environment is a well-textured map, as Figure 4.1 shows. It consists of many long, straight walls constructing multiple hallways, allowing for large drift along the straight trajectories. However, the design of this environment allows for multiple possibilities of loop closure. Therefore, the environment tests the capability of the planner to find loop closure during the exploration process. Each planner is simulated for 30 min to map the maze environment. Additionally, Figure 4.2 visualizes the optimal trajectory through the maze environment computed with the method described in subsection 3.3.1.

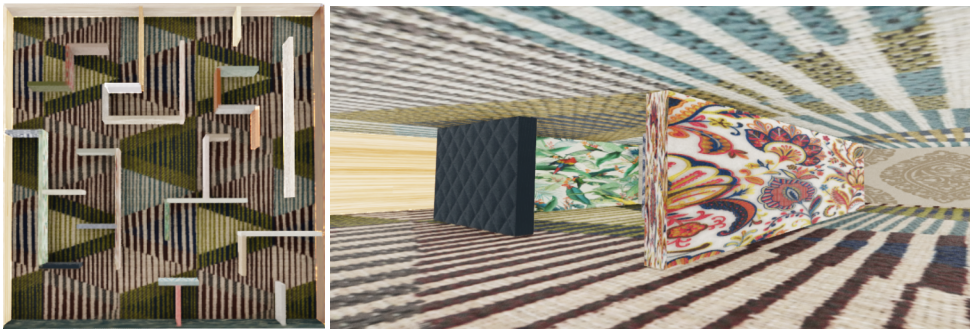


Figure 4.1: **Maze Environment:** A top-down and inside view of the maze environment, a well-textured maze, spanning over a space of 40m x 40m x 3m.

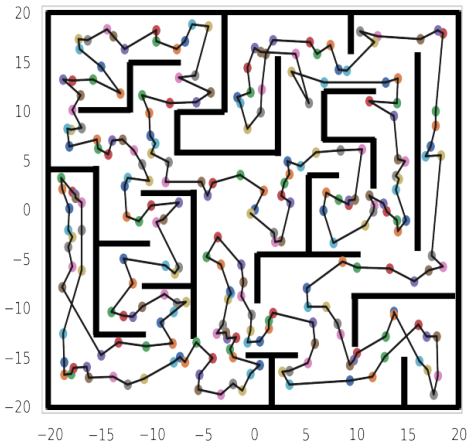


Figure 4.2: **Optimal Path:** The plot depicts the optimal trajectory through the maze environment.

4.2.2 Warehouse

The warehouse environment, depicted in Figure 4.3, on the contrary, lacks textured walls. It introduces the possibility of accumulating drift due to missing image features rather than large traveling distances. Thus, the agent is tested on its capability of reducing drift with ill-textured walls. Moreover, the warehouse has a height of 9 meters and therefore evaluates the planner's proficiency to map and explore in the third dimension. For the warehouse environment, a simulated time of 20 min is considered. Figure 4.4 shows the optimal trajectory through the warehouse environment. Due to the height of the environment, the whole space is traversable. Therefore, the algorithm is slightly adapted by allowing the path to traverse the contours. However, a cost is added for moving from the unoccupied to the occupied space to discourage the agent from changing height too often, an rather stay on top of the shelves once it reaches that point.

Figure 4.3: **Warehouse Environment:** A top-down and inside view of the warehouse environment, a sparse warehouse, spanning over a space of 20m x 35m x 9m.

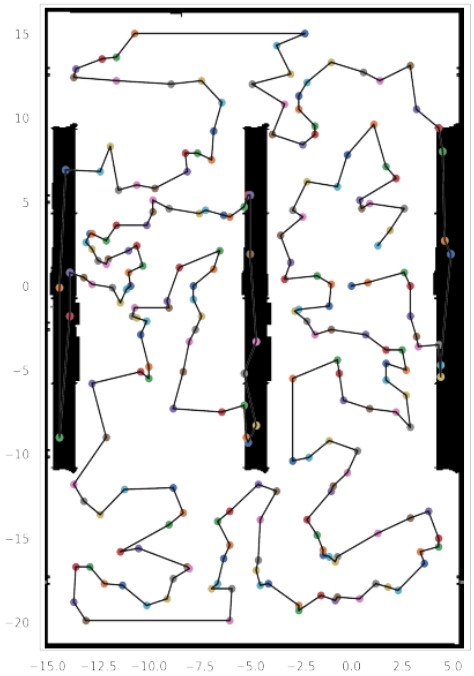


Figure 4.4: **Optimal Path:** The plot demonstrates the optimal trajectory through the warehouse environment.

Chapter 5

Evaluation

The planners are evaluated based on different metrics. Since the most important aspect of the introduced planner is the ability to localize any given position in the built map, a re-localization metric needs to be implemented. Furthermore, the overall drift and the efficiency of the planners are measured. The following sections describe the metrics used to that end.

5.1 Re-Localization

In an ideal case, the re-localization metric would be differentiable, to integrate it into the framework as an instant feedback for the policy. However, defining such a metric is out of the scope of this project. Therefore, a human-in-the-loop is assumed to interpret the results and adjust the policy accurately.

To evaluate the re-localization capability of the planner, query trajectories need to be gathered. These trajectories are generated by simulating multiple, realistic exploration runs of the environment with different policies. To ensure an even distribution of different views, they are sampled with equal probability. Each trajectory is around $7s$, which corresponds to a traveling distance of $\approx 2.5m$. Subsequently, these query trajectories are then localized in the map built by the to-be-evaluated planner. To that end, maplab [19] is utilized to search for loop closures between the built map and a query trajectory by matching features in the query frame to the 3D structures of the map. Since this process tries to localize every frame in the trajectory independently without sharing information between consecutive frames, it does not properly reflect the re-localization capabilities of the planner in a real-life scenario.

Therefore, the process is extended: First, all detected loop closure edges with their corresponding keyframes in the built map and query trajectory are gathered for all query trajectories with the aforementioned process using a custom maplab plugin. The loop closure edges are then filtered by rejecting all edges with a distance exceeding a certain threshold d_{tresh}. With the remaining matched keyframes, the two trajectories - the query trajectory and the trajectory from the built map - are aligned. To that end, the code provided by Zhang and Scaramuzza [22] implementing the alignment process is used. The transformation T' between the two trajectories is computed with a least-squares estimation of transformation patterns introduced by Umeyama [28], which solves the following minimization problem.

$$T' = \underset{T=\{R,t,c\}}{\arg\min} \sum_{i=1}^{n} ||\mathbf{x}_i^{query} - (cR\mathbf{x}_i + t)||^2 \tag{5.1}$$

$\mathbf{x}_i^{query}$ is the position of the keyframe in the query trajectory and $\mathbf{x}_i$ the position of the matched keyframe in the built map. The transformation matrix T' is built from the parameters R, t, c.
Subsequently, the keyframes of the trajectory, which were either rejected in the filtering process or could not be localized, are gathered by interpolating between the matched keyframes and transformed with T'. Finally, the positions of the transformed keyframes are compared to the ground truth positions. The final re-localization error per query trajectory is computed by averaging over the Euclidean distance between each keyframe corresponding to that trajectory and the ground truth position.

$$\begin{aligned} e_{query} &= \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}_i^{aligned} - \mathbf{x}_i^{gt}||_2 \\ \mathbf{x}_i^{aligned} &= T'\mathbf{x}_i \end{aligned} \tag{5.2}$$

$\mathbf{x}_i^{aligned}$ is the position of the transformed keyframe and m the number of keyframes considerd after interpolation. $\mathbf{x}_i^{gt}$ is the corresponding ground truth position.
This metric provides a re-localization error considering multiple frames.

5.2 Drift

To measure the drift, the pose estimate of the SLAM system is compared to the ground truth pose. During the simulation, the estimated poses and ground truth poses of the agent are collected. A pose $\mathbf{p}_i = \{\mathbf{x}_i, R_i\}$ consist of the position $\mathbf{x}_i \in \mathbb{R}^3$ and orientation $R_i \in \mathbf{SO}(3)$. The implementation of Zhang and Scaramuzza [22] is used, which provides a library for commonly used evaluation metrics for visual(-inertial) odometry. To evaluate the quality of the estimated trajectory, first the trajectory estimate $\mathbf{P} = \{p_i\}_{i=1}^N$ and the ground truth $\mathbf{P}^{gt} = \{p_i^{gt}\}_{i=1}^N$ are aligned by solving the same minimization problem as Equation 5.1, resulting in the aligned trajectory $\mathbf{P'} = \{p_i'\}_{i=1}^N$. Subsequently, the error $\Delta\mathbf{p}_i$ is calculated:

$$\begin{aligned} \Delta R_i &= R_i^{gt}(R_i')^T, \\ \Delta\mathbf{x}_i &= \mathbf{x}_i^{gt} - \Delta R_i \mathbf{x}_i' \end{aligned} \tag{5.3}$$

Finally, the quality of the whole trajectory is computed by taking the root mean squared error (RMSE):

$$\begin{aligned} \text{ATE}_{rot} &= (\frac{1}{N} \sum_{i=0}^{N-1} ||\angle(\Delta R_i)||^2)^{\frac{1}{2}} \\ \text{ATE}_{pos} &= (\frac{1}{N} \sum_{i=0}^{N-1} ||\Delta\mathbf{x}_i||^2)^{\frac{1}{2}} \end{aligned} \tag{5.4}$$

$\angle(\Delta R_i)$ symbolizes the conversion of the rotation matrix to angle-axis representation and taking the rotation angle as error.

5.3 Efficiency

Another important aspect of a planner is its runtime. It is desirable to map the environment as efficiently as possible. To measure this aspect, the coverage of the environment is compared against time. To that end, the floor plan is utilized to calculate the percentage of the environment covered at a certain time.

Chapter 6

Results

This chapter describes and visualizes the results of the experiments described in chapter 4 with respect to the evaluation metrics reported in chapter 5. The results are discussed in the following chapter 7.

6.1 Maze Environment

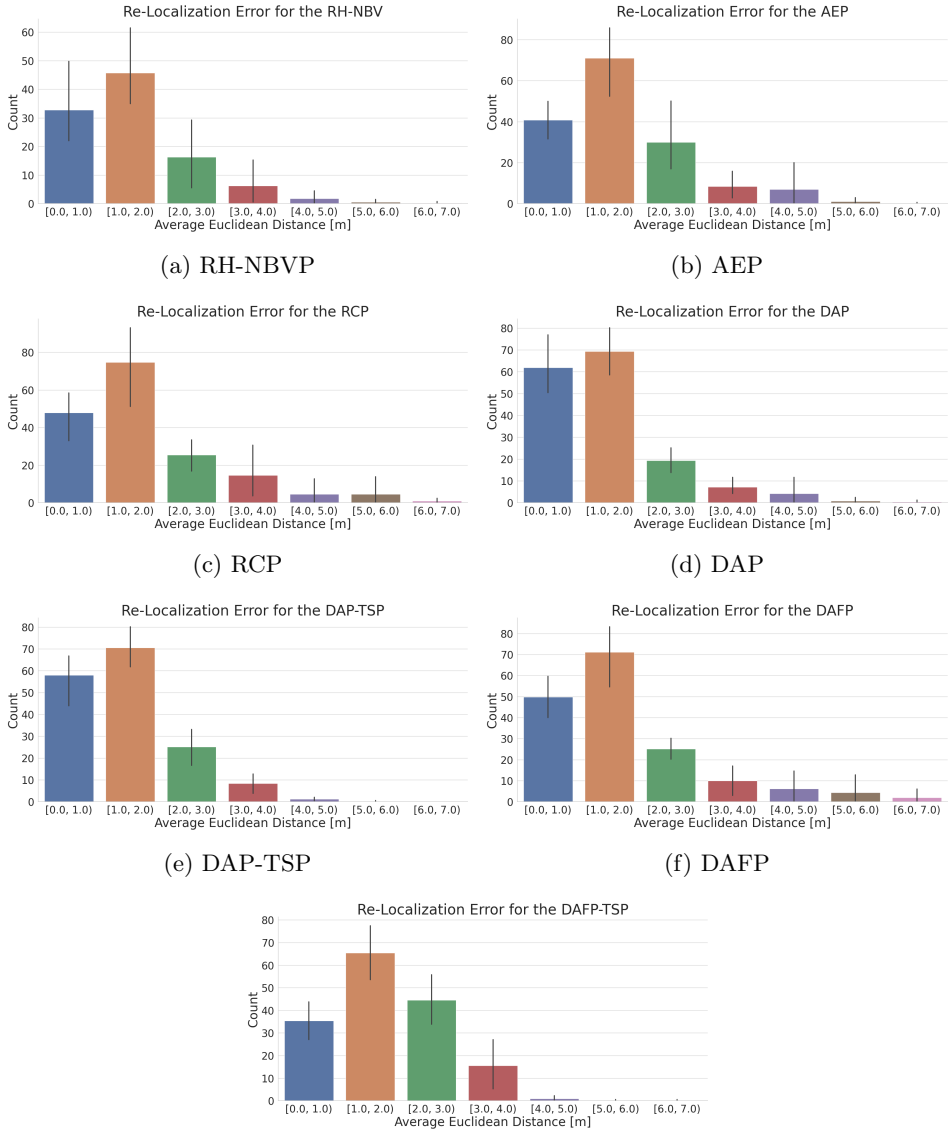
6.1.1 Re-Localization

The re-localization error is represented as a bar plot. As described in chapter 5, the positions of the aligned keyframes are compared to the ground truth positions. The mean of the Euclidean distance between the two positions is calculated over the corresponding query trajectory and classified into a bin. Figure 6.1 depicts the number of trajectories with respect to their mean error for all seven planners over the five runs. The threshold to reject the loop-closure edges is set to $d_{tresh} = 5m$. Overall, the maplab plugin tries to localize the built map against 258 query trajectories.
Additionally, Table 6.1 presents the mean number of query trajectories that are localized in the built map over all 5 runs for each planner.

planner	# query trajectories localized
RH-NBVP	104.75 ± 20.42
AEP	160.0 ± 9.03
RCP	174.0 ± 5.61
DAP	166.0 ± 8.49
DAP-TSP	164.4 ± 11.28
DAFP	170.6 ± 6.78
DAFP-TSP	163.6 ± 14.06

Table 6.1: **Maze Re-Localization Table:** The table presents the mean and the standard deviation of the number of trajectories that are localized in the built map over the five runs. A trajectory is considered as localized if at least three loop-closure edges are found since a minimum of three matched keyframes are needed to estimate the transformation, and at least one loop-closure edge is not rejected by d_{tresh}.

The RH-NBVP showcases the least amount of localized query trajectories as well as the poorest performance concerning the re-localization error. The AEP and RCP



(a) RH-NBVP

(b) AEP

(c) RCP

(d) DAP

(e) DAP-TSP

(f) DAFP

(g) DAFP-TSP

Figure 6.1: **Maze Re-Localization Error:** Figure 6.1a - Figure 6.1g show the average re-localization error of each planner over 5 runs. The bars depict the mean and the black lines show the 95% confidence interval of the number of trajectories inside the bins over the 5 runs.

behave very similarly regarding their re-localization precision. Both show a re-localization error of less than $1m$ for about 40 query trajectories and a re-localization error between $1m$ and $2m$ for around 70. However, while the RCP results in an error of more than $5m$ for up to 15 trajectories, the AEP hardly exceeds an error of $5m$. The DAP provides a re-localization error of less than $1m$ for almost one and a half times as many trajectories as the RCP and AEP. Similar to the AEP, it barely surpasses an error of $5m$. The results for the DAP-TSP hardly differ from the one of the DAP besides the number of query trajectories with an error exceeding $4m$, which are even more diminishing for the DAP-TSP. The DAFP shows very similar behavior to the RCP and does not compare to the re-localization quality of the DAP. The DAFP-TSP performs better than the DAFP regarding the number of trajectories with a higher re-localization error than $4m$. However, clearly, fewer query trajectories provide an error of less than $1m$ for the DAFP-TSP compared to the DAFP.

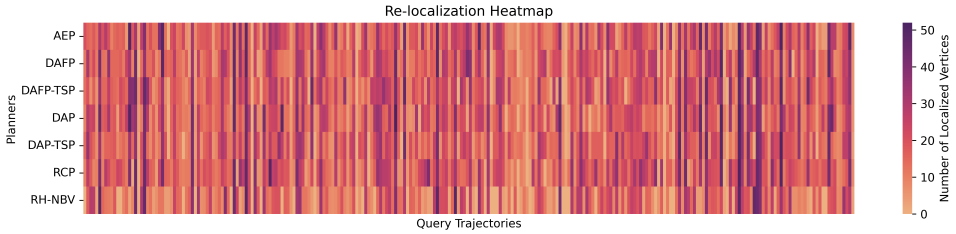


Figure 6.2: **Maze Re-Localization Correlation:** The heat map shows the mean number of vertices localized in a query trajectory over the five runs depending on the planner.

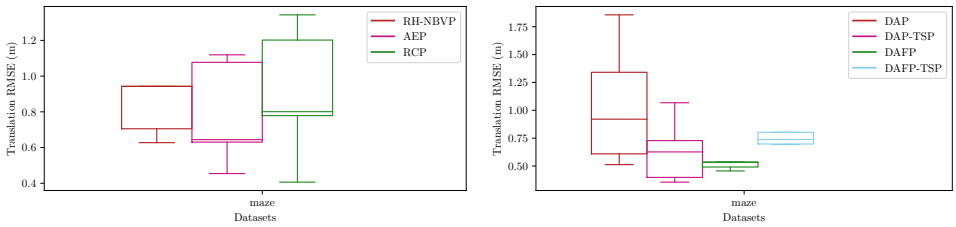
Moreover, Figure 6.2 demonstrates the correlation between the number of vertices localized in a query trajectory averaged over the five runs and the planners used. One can see, that while most query trajectories show similar localizability for all planners, some are only picked up by certain planners.

6.1.2 Drift

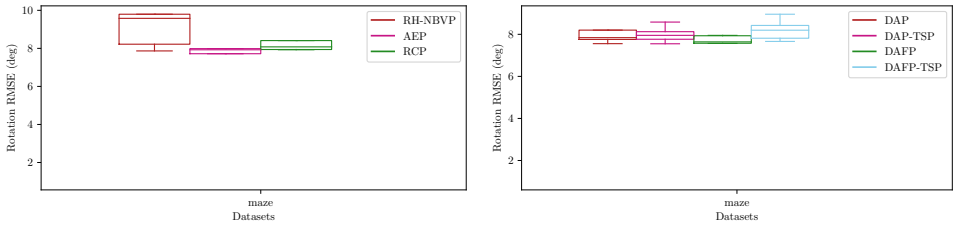
As mentioned in chapter 5, the drift of the planners is evaluated by the RMSE described in Equation 5.4. Figure 6.3 depicts the RMSE of all seven planners of the five runs.
One can see in Figure 6.3 that the RMSE of all planners stays within $2m$ for the translation error and 10 deg for the rotation error. The RCP and DAP demonstrate the most variance in their translation RMSE, while DAFP and DAFP-TSP show hardly any variance. The rotation RMSE is very similar for all planners with a median of around 8 deg.

6.1.3 Efficiency

The efficiency of the planners is measured by observing the area covered over time. Figure 6.4 portrays the mean coverage for the simulated time of 30 minutes for all planners over the five runs. Since the timestamps may differ between runs, the mean coverage is not necessarily continuously increasing.
Figure 6.4 exhibits a very similar behavior for all planners besides the RH-NBVP. Apart from the RH-NBVP, the planners cover around 20% of the area in five minutes, which accounts for around $320m^2$ for the maze environment. However, the RH-NBVP maps the space significantly slower and is only able to cover around 50% during the 30 minutes.



(a) Translation RMSE.



(b) Rotation RMSE.

Figure 6.3: **Maze Translation and Rotation RMSE:** The figure shows the translation and rotation RMSE for all seven planners over the five runs.

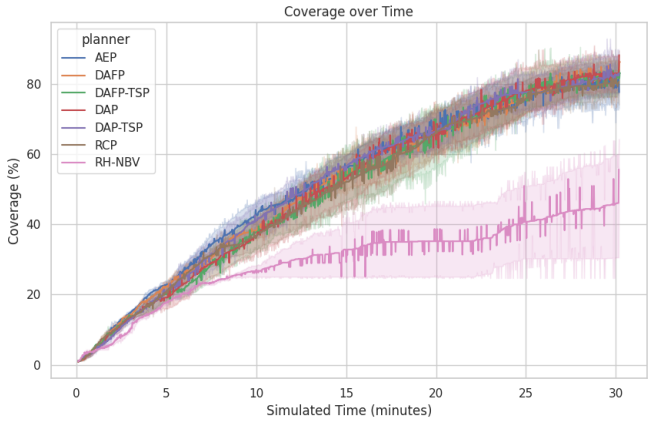
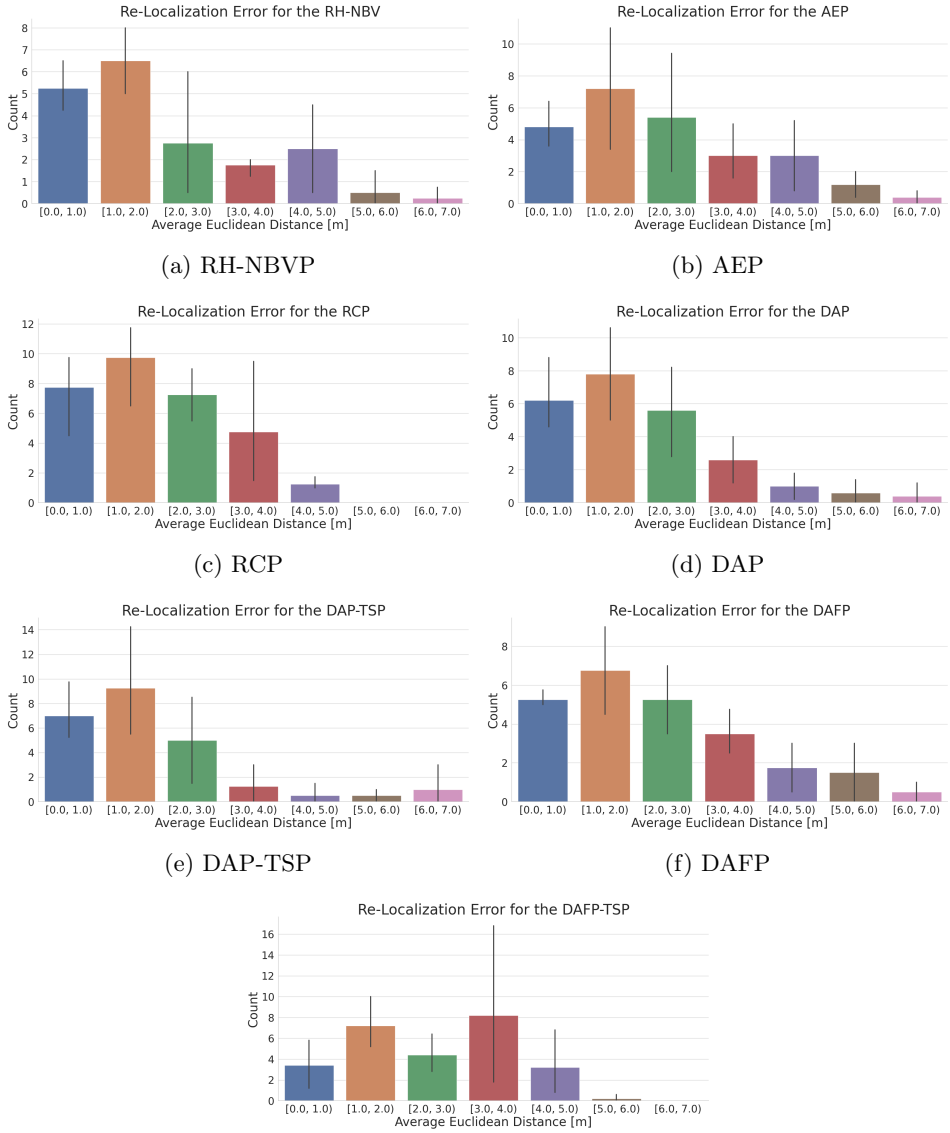


Figure 6.4: **Maze Coverage over Time:** The lines represent the mean coverage of the corresponding planner over the five runs and the shaded area around demonstrated the standard deviation.

6.2 Warehouse Environment

6.2.1 Re-Localization

Figure 6.5 depicts the re-localization error for the warehouse environment for the seven planners over five runs. As for the maze environment, the bars symbolize the number of trajectories with the re-localization in the associated bin. For the evaluation, again 258 query trajectories are considered and $d_{thresh} = 5m$.
Furthermore, Table 6.2 shows the mean number of query trajectories localized over the five runs as well as the standard deviation. Additionally, it states the number of runs that diverged in the 20 min simulation time.
The low mean number of localized trajectories and the high standard deviation in



(a) RH-NBVP

(b) AEP

(c) RCP

(d) DAP

(e) DAP-TSP

(f) DAFP

(g) DAFP-TSP

Figure 6.5: **Warehouse Re-Localization Error:** Figure 6.5a - Figure 6.5g show the average re-localization error of each planner over 5 runs. The bars depict the mean and the black lines show the 95% confidence interval of the number of trajectories inside the bins over the 5 runs.

planner	# query trajectories localized	# runs diverged
RH-NBVP	20.5 ± 5.0	0
AEP	26.4 ± 13.58	2
RCP	31.5 ± 7.94	2
DAP	24.8 ± 4.66	2
DAP-TSP	26.5 ± 9.95	3
DAFP	26.75 ± 2.36	1
DAFP-TSP	27.6 ± 13.58	1

Table 6.2: **Warehouse Re-Localization Table:** The table presents the mean and the standard deviation of the number of trajectories that are localized in the built map over the five runs. The third column specifies the number of runs that diverged.

both the number of localized trajectories, but also in the re-localization error, minder the informative value of the results. Here, the RH-NBVP shows more similar behavior to the other planners. While all planners perform nearly equally badly in the number of localized trajectories, the RCP presents the most number of trajectories with a re-localization error of less than one and does not exceed the $5m$ mark.
Moreover, Figure 6.6 visualizes the correlation between the planners and the number of keyframes localized in a query trajectory.

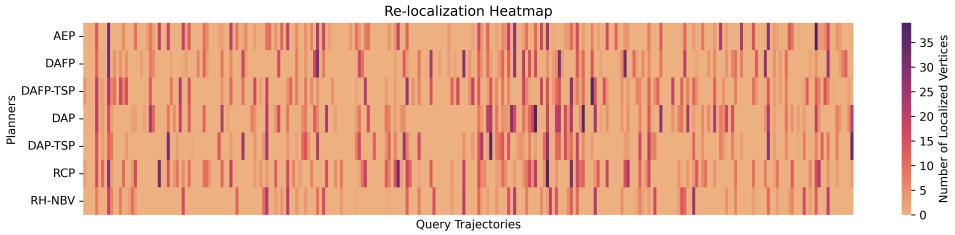
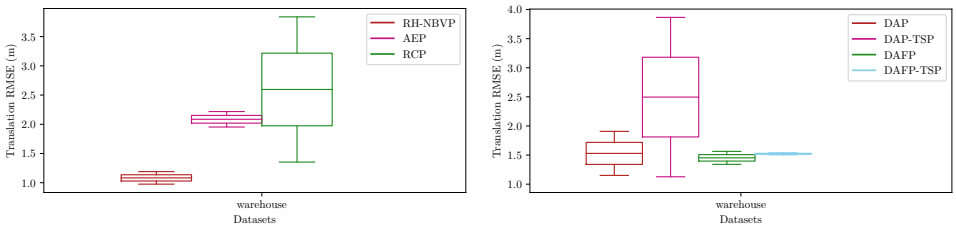


Figure 6.6: **Warehouse Re-Localization Correlation:** The heat map shows the mean number of vertices localized in a query trajectory over the five runs depending on the planner.

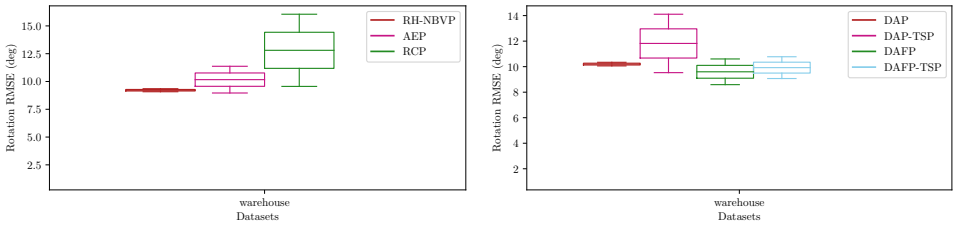
One can see, that the number of vertices localized is not necessarily comparable between the different planners. While some planners cannot localize the query trajectory at all, others localize it very well. However, considering the scale of the colorbar compared to the one of Figure 6.2, it becomes evident, that fewer keyframes in a trajectory are localized in general.

6.2.2 Drift

The drift of the pose estimates is again evaluated per planner by the RMSE. Figure 6.7 visualizes the RMSE of all seven planners. It is important to note, that only two runs per planner are considered, since the diverged runs cannot be included.
In comparison to the maze environment, the translation RMSE is only bounded by $4m$ in the warehouse environment. The RCP shows again the most variance compared to the RH-NBVP and AEP. For the variants of the drift-aware planner, the DAP-TSP exhibits the most variance, while the other planners perform very similarly. However, since only two runs are considered, the difference between the



(a) Translation RMSE.



(b) Rotation RMSE.

Figure 6.7: **Warehouse Translation and Rotation RMSE:** The figure shows the translation and rotation RMSE for all seven planners over two runs.

planners is not as significant. The rotation RMSE is again rather similar between all planners.

6.2.3 Efficiency

The efficeny is again evaluated by the coverage over the 20 min simulated time. The results are visualized in Figure 6.8.

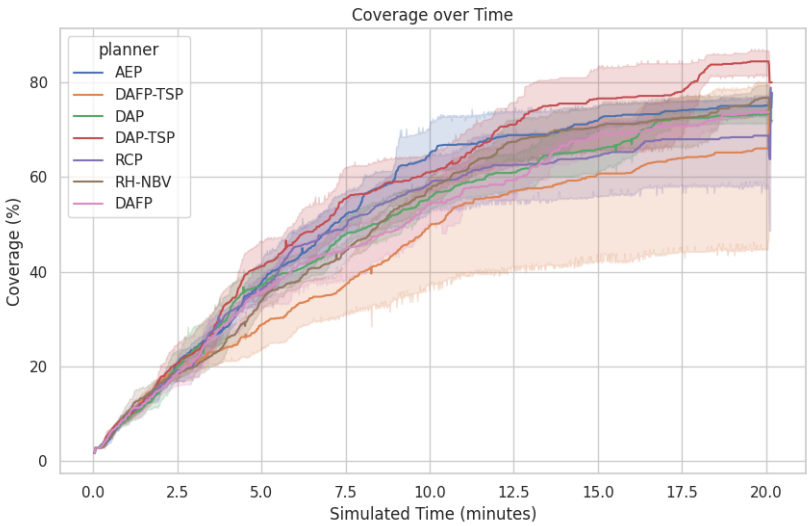


Figure 6.8: **Warehouse Coverage over Time:** The line plot represents the mean coverage per time over the 5 runs. The standard deviation is shown as the shaded are around the line.

Due to the diverged runs, the standard deviation is quite significant for all planners. A striking difference to the maze environment is the performance of the RH-NBVP, which is consistent with the efficiency of the other planners. Figure 6.8 shows, that the agent covers around 40% in the first 5 min for most planners, but slows down significantly afterward. While the DAFP-TSP indicates the worst performance, the DAP-TSP almost consistently outperforms the other planners.

6.3 Floor Plan

6.3.1 Floor Plan Localization

Finally, the drift of the floor plan localization is evaluated. Figure 6.9 depicts the translation RMSE for the estimated position on the floor plan against the ground truth position.

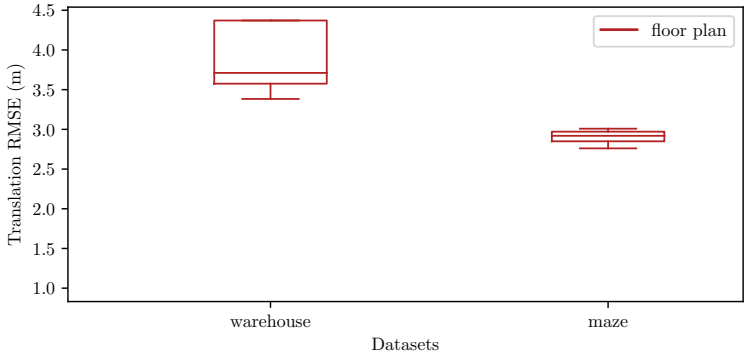


Figure 6.9: **Translation RMSE for Floorplan Localization:** The figure shows the translation RMSE for both the maze and warehouse environment.

As Figure 6.9 demonstrates, the localization on the floor plan significantly underperforms the performance of the SLAM estimate. The maze environment indicates a better localizability with a median of $2.919m$, while for the warehouse environment, only a median of $3.711m$ can be achieved.

Chapter 7

Discussion

In this chapter, the results are discussed and analyzed.

7.1 Localization Accuracy

The localization accuracy includes the re-localization ability of the planners, which is considered the main focus behind the design of the proposed planners, as well as the floor plan localization, which aims to approximate the ground truth. Both are discussed in the following subsections.

7.1.1 Re-Localization

Considering both environments, it is obvious that the warehouse environment is much more difficult to localize in. The reason behind this behavior is the absence of well-textured features and therefore landmarks. This can further lead to the SLAM system diverging completely, as shown in Table 6.2. The RH-NBVP is the only planner, that does not diverge for all five runs. The design behind the RH-NVBP is very local compared to the other planners, meaning that while the RH-NVBP only plans trajectories in the near surroundings of the agents' position, the other planners maximize over trajectories throughout the whole environment. Therefore, the RH-NVBP avoids long trajectories and thus decreases the possibility of divergence.
The DAP and DAP-TSP outperform the other planners regarding re-localization accuracy for the maze environment. Even if the difference to the other planners is not immense, the design of the DAP seems to give the agent the right incentive to improve the accuracy of the re-localization. In terms of the number of query trajectories localized per planner, tabulated in Table 6.1, all planners beside the RH-NVBP show comparable behavior. Since they are based on one-frame localization, they are not as expressive as the re-localization accuracy. However, the DAFP and DAFP-TSP cannot match the performance of the DAP, which suggests poor behavior in approximating the ground truth. Moreover, the DAFP and DAFP-TSP demonstrate quite different behavior in precise re-localization. This might stem from the DAFP-TSP rather concentrating on following the reference trajectory than considering the drift from the pose prediction of the floorplan localization.

7.1.2 Floor Plan Localization

As shown in Figure 6.9, the floor plan localization is unable to approximate the ground truth and even underperforms the SLAM prediction. The behavior in localizing the agent's position in the warehouse environment is no surprise due to

the lack of contours as input for the target cloud. While the floor plan localization performs better in the maze environment, the approximation to the ground truth is still much worse than expected. This behavior might arise from the GICP failing to grasp the underlying shared structure between the source and target cloud. A reason for that could be the different sizes of the clouds, as well as the area represented by them. While the target cloud only represents the coordinates on the contours, the source cloud includes the whole depth image projected onto the floor. This could cause a poor estimate of the transformation between them.

7.2 Drift

The RH-NVBP can keep the pose error quite low for both environments due to its design explained above. Remaining rather local improves the pose estimates since the drift tends to increase with the distance traveled. This behavior is also reflected by the coverage of the RH-NVBP in the maze environment shown in Figure 6.4, where it only maps about 50% of the space during the simulated time. Interestingly, the DAP presents the highest maximum translation error, even though it showcases the best re-localization accuracy. However, considering the median translation RMSE of the planners, they do not differ profoundly from each other, but all stay within $0.5m - 1m$. It is also important to note, that the figures showcase an absolute and not relative trajectory error. ATE is shown to be sensitive to the time of the error occurring [29]. The decision to use ATE is based on a simpler comparison between the planners.

7.3 Efficiency

Due to the height of the warehouse environment, the whole area is traversable when operating with an MAV. Moreover, the simple shape of the warehouse leads to the fact, that an optimal path through the environment is not unique. Therefore, the advantage of following the reference trajectory is questionable. Similarly, the structure behind the maze environment does not necessarily offer the possibility of a distinct optimal path and makes it inevitable to revisit certain locations multiple times. This explains the similar coverage behavior of all planners besides the RH-NBVP. Moreover, all planners consider a measure of efficiency in their costs, which due to the design of the environments may show the same effect as following the reference trajectory.

Chapter 8

Conclusion

This work designed and implemented an active mapping policy to build a map with precise re-localization ability. A new evaluation method is implemented to assess the re-localization of the planners more precisely by approximating a multi-frame with an underlying single-frame localization. Moreover, the simulation framework was extended with a floor plan node utilizing a 2D floor plan to make more informed decisions on the exploration and mapping process.

The proposed drift-aware planner was compared to three different planners. Utilizing the new evaluation metric, it was shown, that the drift-aware planner successfully improved the re-localization accuracy in comparison to the other planners for the maze environment. The floor plan enabled the suggestion of an optimal trajectory as a reference for the planner, a localization of the agent's position on the floor plan and an evaluation of the planner's efficiency by considering the area covered over time.

The current implementation of the floor plan localization could not meet its expectation and failed to provide and approximation of the ground truth. Moreover, the two environments considered could not provide sufficient information to evaluate the benefits of following the optimal trajectory. Additionally, the value function of the proposed policy was not adapted to its focus and therefore is very likely not the optimal choice.

8.1 Future Work

The simulation framework and floor plan utilization leave a lot of possibilities for future work.

Firstly, both environments considered are edge-case scenarios and do not provide the best setting to compare the planners. While the warehouse environment lacks visual features for localization, the maze environment consists of long corridors of the same textured walls, which introduces the high possibility of accumulating drift. It would be interesting to evaluate the planners in less edge-case environments. This may provide more information on the different strengths of the planners and how to adapt them accordingly.

Secondly, adapting not only the cost, but also the value function of the proposed planner may further improve its performance. This might be achieved by weighing the gain and the cost against each other rather than normalizing the gain by its cost.

Thirdly, the floor plan localization needs to be improved to be able to approximate the ground truth. A first possibility would be to only consider contours when matching the source and target cloud to enable a more precise matching.

In summary, the framework offers the implementation and evaluation of an exploration planner mapping an environment with the goal of precise re-localization by considering prior information from a 2D floor plan and leaves many options for future work and improvement.

Bibliography

[1] T. Dang, C. Papachristos, and K. Alexis, "Autonomous exploration and simultaneous object search using aerial robots," in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–7.

[2] M. Popovic, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using uavs," *arXiv preprint arXiv:1609.08446*, 2016.

[3] L. Yoder and S. Scherer, "Autonomous exploration for infrastructure modeling with a micro aerial vehicle," in *Field and Service Robotics: Results of the 10th International Conference*. Springer, 2016, pp. 427–440.

[4] M. Baumgartner, "Optag: Optimal active guidance with known floorplans," 2023.

[5] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[6] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[7] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[8] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3477–3484.

[9] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[10] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[11] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using mavs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] H. Merzić, E. Stumm, M. Dymczyk, R. Siegwart, and I. Gilitschenski, "Map quality evaluation for visual localization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3200–3206.

[14] Z. Zhang and D. Scaramuzza, "Fisher information field: an efficient and differentiable map for perception-aware planning," *arXiv preprint arXiv:2008.03324*, 2020.

[15] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "A pose graph-based localization system for long-term navigation in cad floor plans," *Robotics and Autonomous Systems*, vol. 112, pp. 84–97, 2019.

[16] L. Gao and L. Kneip, "Fp-loc: Lightweight and drift-free floor plan-assisted lidar localization," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4142–4148.

[17] Y. Watanabe, K. R. Amaro, B. Ilhan, T. Kinoshita, T. Bock, and G. Cheng, "Robust localization with architectural floor plans and depth camera," in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 133–138.

[18] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 595–625, 2016.

[19] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[20] A. Cramariuc, L. Bernreiter, F. Tschopp, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena, "maplab 2.0–a modular and multi-modal mapping framework," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 520–527, 2022.

[21] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

[22] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.

[23] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-d environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[24] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.

[25] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[27] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[28] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[29] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, pp. 387–407, 2009.