Eindhoven University of Technology

MASTER

Towards achieving synergy in multiple SLAM algorithms via pose-graph optimization

Walk, B.W.M.

Award date:
2021

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Department of Mechanical Engineering
Dynamics and Control Group

Towards achieving synergy in multiple SLAM algorithms via pose-graph optimization

MSc in Systems and Control

Bjorn W.M. Walk
0964797

Supervisor(s):
Dr. Ömür Arslan, Mechanical Engineering, Dynamics & Control

Committee:
Prof. Nathan van de Wouw (Chair), Mechanical Engineering, Dynamics & Control
Dr. René van de Molengraft, Mechanical Engineering, Control Systems Technology

DC 2021.091

Eindhoven, October 13, 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

04-10-2021

Name

B.W.M. Walk

ID-number

0964797

Signature

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Towards achieving synergy in multiple SLAM algorithms via pose-graph optimization

B.W.M. Walk
0964797
DC 2021.091

***Abstract*— Simultaneous Localization and Mapping (SLAM) is one fundamental challenge of mobile robotics and has many algorithmic and technical solutions. Unfortunately, there is no best solution, and each SLAM approach has its advantages and disadvantages over alternatives under certain circumstances. However, identifying the limitations of SLAM algorithms is a research challenge itself. In recent years, due to the many available SLAM approaches, combining the existing SLAM algorithms instead of creating new ones received significant research interest. This paper aims to contribute towards achieving synergy in multiple SLAM algorithms running in parallel, by proposing a pose-graph optimization approach to autonomously self-assess the localization performance of each SLAM algorithm in the integration process. This allows automatic determination of which SLAM data should be used/ignored in fusing multiple SLAM methods. The conducted experiment performed with the F1Tenth miniature racecar shows that the proposed approach leads to an accurate determination of the localization performance of each considered SLAM method. This indicates that after the implementation of the proposed approach, it is possible for a mobile robot to online self-assess the localization performance of different SLAM algorithms without a reference/ground truth.**

***Index Terms*— autonomous driving, localization, SLAM, pose-graph optimization, self-assessment of localization**

I. Introduction

Perception algorithms are key components of mobile robots and provide functionalities such as estimating the state of the robot, building a map of its surroundings, and detect obstacles. One mobile perception approach is Simultaneous Localization and Mapping (SLAM), which grants mobile robots the ability to construct a map of the perceived environment, while simultaneously localizing themselves with respect to this map [1]. SLAM has been researched extensively over the past decades and many different methods have been proposed [2]. SLAM empowers several mobile robot applications and has been implemented in many different domains from an indoor office robot to autonomous submarines [3]. In general, SLAM methods can be defined by means of two paradigms: filter-based SLAM, and Graph-based SLAM. The first paradigm is a filtering technique that incorporates the new information as it comes available, due to this incremental behavior they are also known as online SLAM techniques. On the other hand, graph-based SLAM estimates the full trajectory and map from all available information and solves, therefore, the full SLAM problem [4]. The choice of the SLAM method usually depends on the application and other factors such as computational load and equipped sensors. Based on these factors each SLAM algorithm has certain advantages over the other. However, in practice identifying these advantages usually requires extensive research of each specific SLAM method [5]. Thus, substantially limiting the SLAM capabilities and increasing the deployment costs. Additionally, the tools and framework available to assess the performance of SLAM are limited and generally require human interaction and/or external devices. Furthermore, due to the extensive research regarding single-robot SLAM, application of multiple SLAM algorithms running in parallel received significant research attention [6]. Generally, two approaches are considered; the first approach applies multiple identical SLAM algorithms with multiple data sources, e.g. multi-robot setting, and the second approach considers multiple different SLAM methods with one shared data source, e.g. single-robot setting. Their purpose is to improve the overall performance of mapping as well as localization based on map-fusion and assume each considered SLAM method is performing well, not assessing the SLAM performance.

In this paper, we present a pose-graph optimization framework that utilizes multiple different SLAM algorithms running in parallel on a single robot and averages the obtained trajectory of each SLAM algorithm to obtain an optimized reference trajectory of the mobile robot for online self-assessment of localization performance of SLAM algorithms. This is considered to be an essential step towards achieving SLAM synergy in multiple SLAM algorithms running in parallel since from this point onward we are able to assess the SLAM localization performance online, such that we can indicate how well each method is performing. Thus, being able to determine at what time instance to share information from the well-performing method to the less performing method, thereby making it possible to achieve SLAM localization synergy.

A. Motivation and Problem Formulation

As SLAM is a key component of modern-day mobile robots, many different SLAM methods are available. Especially in the commonly used framework for robotics called

The author is an M.Sc. candidate of the Systems and Control master program, Department of Mechanical Engineering, Dynamics and Control Group, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, 04-10-2021. Emails: b.w.m.walk@student.tue.nl Supervisors: dr. Ö. Arslan, prof. dr. ir. N. van de Wouw.

Robotics Operating System (ROS). This makes it challenging to choose a SLAM method for your own specific needs. Although some research has been conducted in order to merge multiple SLAM algorithms by considering multiple robots or different frameworks [7]–[10], they usually do not consider, or only partially consider, the online quality of the obtained results. To truly achieve robotic autonomy with multiple SLAM algorithms running on a single robot in parallel, it is required to enable robotic platforms to automatically self-assess the quality of the obtained data. In practice, some sort of accurate reference trajectory or map, also known as ground truth, is often used for the quality assessments of SLAM [5], [11]–[14]. Therefore, the first step in achieving synergy in multiple SLAM algorithms is to obtain an accurate reference trajectory without using external devices or human interaction. This would enable a robot to autonomously decide when to share the localization results from the well-performing localization method to the less performing localization method. In this paper, we present a pose-graph optimization approach that is able to self-assess the localization of SLAM by using multiple SLAM algorithms in parallel, given that the map is accurately estimated.

B. Contributions and Organization of the Paper

This paper proposes a pose-graph optimization approach to enable robotic platforms to automatically obtain the state of a mobile robotic platform as accurately as possible to achieve autonomy for the quality assessment of the obtained data of multiple SLAM algorithms. This paper contributes to achieving synergy in multiple SLAM algorithms as an autonomous performance measure is a necessary prerequisite for determining which data to share and when. Presently, SLAM methods are evaluated based on data obtained from external devices or human interaction, which limits the full potential of achieving robotic autonomy as well as increasing the development costs.

The remainder of this paper is organized as follows. The problem of SLAM, with in particular localization, is summarized, related work on the state-of-the-art methods and improvements are reviewed, and a more in-depth explanation of pose-graph optimization is given in Section II. In Section III, the approach proposed in this paper is discussed and is experimentally validated in Section IV. The paper is concluded in Section V along with future research directions.

II. Background

In this section, a brief overview of widely used localization methods is presented. Furthermore, we provide an in-depth introduction to the pose-graph optimization framework as it is the main topic of this paper.

A. Localization Problem

The SLAM problem may be divided into two subproblems: 1. the localization problem and 2. the mapping problem; which are closely connected, like the 'chicken and egg' causality dilemma. The localization problem can be described as the estimation of the state belief of the mobile robot's trajectory $x = [x_0, ..., x_N]$ for time $t = [t_0, ..., t_N]$ given all observations $z = [z_0, ..., z_N]$, control inputs $u = [u_0, ..., u_{N-1}]$, and map m, represented by a probability density function such that:

$$p(x|z,u,m). \tag{1}$$

This can be extended with the mapping problem m to become:

$$p(x,m|z,u), \tag{2}$$

which is also known as the SLAM problem. However, in this paper we are only interested in the localization problem of SLAM.

B. Localization Methods

Localization is essential for a mobile robot as it estimates the mobile robot's pose (position and orientation) relative to the environment (map). If a mobile robot does not know where it is relative to the environment (map), it is difficult to decide what to do next. Two common SLAM approaches for solving the localization problem of a mobile robot are filter-based localization and graph-based localization [1]. Since we only focus on the localization problem of SLAM, we are not limited to use only SLAM methods. Therefore, we also consider scan-matching-based odometry. It is a widely used method for estimating the relative pose and given the initial pose can solve the localization problem of a mobile robot.

1) Filter-based Localization: Filter-based localization often uses parametric Kalman filtering, nonparametric particle filtering, and their combination for estimating the robot pose in a known map. They are generally straightforward to implement and do not have high computational complexity. However, they only recursively estimate the current robot pose and do not take into account the whole trajectory, making them vulnerable for drift.

The Kalman filter is one of the earliest and widely used filtering algorithms [1], in which the belief of the robot's pose is represented by a Gaussian [4]. Additionally, Kalman filters assume that the control input and sensor information are subjected to Gaussian noise. The main advantage of using Kalman filters for localization is that they are relatively simple to implement, and thus do not require high computational power. However, since beliefs are represented by a Gaussian, which is uni-modal, its representation capabilities are limited. Kalman filtering also requires a proper initialization, since a Gaussian function does not allow a multimodal belief representation.

Particle filters, on the other hand, are nonparametric and represent the belief of the robot pose by a set of samples, called particles [1]. Each particle is weighted and given an importance factor based on the likelihood of the robot's measurements being taken at the sample pose. Over time the particles are resampled in a more likely pose of the mobile robot, based on the new measurements and updated weight of the particles. The main advantage of particle filters is that they do not make the Gaussian assumption and represent any

probability distribution using sample particles. However, the key problem of particle filters is that in general, in order to accurately localize, the number of particles is scaled with the dimension of the environment. Therefore, for relatively large-scale environments particle filters are computationally demanding.

Over the past years, several improvements are introduced to minimize the disadvantages of the filtering techniques by improving the accuracy and reducing computational complexity [15]–[18]. Furthermore, both filtering techniques are provided in ROS [17], [19]–[21].

2) Graph-based Localization: In contrast to the filter-based localization technique, graph-based localization techniques estimate the entire trajectory of a robot [22]. Although graph-based localization is computationally more expensive than filtering-based alternatives, recent algorithmic developments enable fast graph-based localization that is suitable for real-time operations [23]. Graph-based techniques involve constructing a graph, whose nodes represent poses of the mobile robot. In this graph, an edge between two nodes represents a relative rigid transformation, that constrains the connected poses. The geometric constraints between adjacent nodes are obtained from observations of the environment or odometry information between two consecutive poses. The crucial problem of graph-based techniques is to find the optimal rigid transformation between nodes, such that the measurements are maximally consistent with a given map. Because graph-based techniques take into account all available sensor information, its accuracy is generally higher than filter-based techniques as there is less drift. However, performing the optimization can be computationally demanding.

Improvements of graph-based techniques are mainly focused on reducing the computational complexity by solving the optimization problem more efficiently [23]–[27]. The ROS implementations of graph-based techniques can be found in [25]–[27].

3) Scan-Matching-based Odometry: In contrast to previously mentioned localization methods, which provide the absolute pose of the robot, we consider the widely used odometry method based on scan-matching. The goal of scan-matching-based odometry is to find the relative translation and rotation between two laser scans taken by the robot at the reference position and the current position [28]. Scan-matching is an optimization problem that looks for the best match between the scans starting from the initial guess transformation. Most common solutions to scan-matching problem are Iterative Closest Point (ICP) [29], [30] and its variant based on point-to-line metric (*PL*ICP) [31]. In ICP, each point in the reference scan is associated with the corresponding point in the current scan according to an Euclidean distance [32]. Hence, the optimal transformation is considered as a solution to the minimization of the distance between correspondences. However, not all points in the reference scan might have reasonable corresponding points in the current scan, and to overcome this issue the point-to-line metric is suggested. In *PL*ICP, the points in the reference scan are associated with the lines extracted from the points in the current scan, thus, achieving faster convergence and minimizing the number of iterations. Furthermore, to minimize the computational complexity, most of the algorithms rely on the dead-reckoning systems to predict the transformation and, thus, to decrease the search-space [28]. Although it provides the results faster, the robot's dead-reckoning error affects the robustness of the algorithm by predicting the initial guess far from the global maximum [32]. The ROS implementation of a scan-matching-based localization method can be found in [33].

C. Pose Graph Optimization

Consider a list of poses in the planar setup $\mathbf{p} = [\mathbf{p}_0, ..., \mathbf{p}_N]$ containing $n+1$ poses $\mathbf{p}_i = [x_i, y_i, \theta_i]^T$, describing the position (x_i, y_i) and orientation (θ_i) of a mobile robot at consecutive time instances $\mathbf{t} = [t_0, ..., t_N]$. Suppose we can obtain for each pair of poses (p_i, p_j) a relative pose measurement $\bar{p}_i^j$ describing a measurement of the relative pose between p_i and p_j by considering that p_i and p_j are the absolute poses defining a coordinate system and p_i^j is a change of coordinate system. Then, the measured relative pose between p_i and p_j is given by:

$$\bar{x}_i^j = (x_j - x_i)\cos\theta_i - (y_j - y_i)\sin(\theta_i) \tag{3}$$

$$\bar{y}_i^j = (x_j - x_i)\cos\theta_i + (y_j - y_i)\sin(\theta_i) \tag{4}$$

$$\bar{\theta}_i^j = \theta_j - \theta_i \tag{5}$$

which is also known as the inverse of compounding $\mathbf{p}_i^j = \mathbf{p}_i \ominus \mathbf{p}_j$ [34]. However, it is more convenient to describe the pose measurements as rigid transformation matrices from the robot frame r to the world frame W:

$$\bar{\mathbf{p}}_i = \bar{\mathbf{T}}_{r_i}^W = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & x_i \\ sin(\theta_i) & cos(\theta_i) & y_i \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

for $i = 0, ..., n$, where $\bar{\mathbf{p}}_i$ comprises a translation vector $\mathbf{t} \in \mathbb{R}^2$ described by Cartesian coordinates $(x_i, y_i)^T$ and a rotation matrix $\mathbf{R}_i \in SO(2)$ described by corresponding orientation θ_i such that $\bar{\mathbf{p}}_i \in SE(2)$. In this paper we switch between both representations regularly. The relative pose between $\mathbf{p}_i$ and $\mathbf{p}_j$ can then be represented by $\mathbf{T}_i^W$ and $\mathbf{T}_j^W$ as:

$$\bar{\mathbf{T}}_j^i = \mathbf{T}_j^W \cdot (\mathbf{T}_i^W)^{-1}. \tag{7}$$

In general, these relative pose measurements are considered to be the constraints between poses assumed by the mobile robot at different instants of time and can be described by odometry and loop closure constraints. Odometry constraints are based on the sequential measurements of the ego-motion of the robot, usually provided by proprioceptive sensors (wheel odometry, GPS, or IMU), or by exteroceptive sensor-based techniques (scan matching, feature registration, etc) [35]. Loop closure constraints are non-sequential and set up when the current observation is matched with a past measurement. They require the use of exteroceptive sensors (LiDAR, vision sensors, etc).

Given the constraints for all available pairs $(\mathbf{p}_i, \mathbf{p}_j)$, the objective of pose graph optimization is to estimate the configuration of poses $\mathbf{p}^*$ that maximizes measurements (constraints) satisfaction [36]. A particularly insightful way of modeling the pose estimation problem is using graph formalism. Generally, a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists out of vertices $\mathcal{V}$ (also called nodes) and edges $\mathcal{E}$ (also called constraints), and can be either directed or undirected, see Figure 1. Directed graphs have edges that can only traverse in one direction, while undirected graphs indicate that each edge can traverse in both directions. The metrical properties of the graph, such as the length of edges and the positions of nodes, are irrelevant. In the pose-graph framework, each vertex in the graph represents an absolute pose, and each edge in the graph represents a relative pose between two arbitrary vertices. By convention, if an edge is directed from node i to node j, the corresponding constraint (relative transformation) is expressed in the reference frame of node i. The objective of pose-graph optimization via graph formalism is to associate an absolute pose to each vertex of the graph based on the constraints. As the constraints are given as relative poses and not absolute poses, according to standard procedure, we set the initial pose of the robot as the origin of the optimization reference frame, i.e $\mathbf{p}_0 = [0,0,0]$. Thus, our objective is to estimate the configuration $\mathbf{p}^* = [\mathbf{p}_1^*, ..., \mathbf{p}_n^*]$ that maximizes constraints satisfaction while minimizing constraint violation, in which we exclude $\mathbf{p}_0$ from the to-be-optimized configuration $\mathbf{p}^*$. We consider the optimization problem to be [37]:

$$T^* = \arg\min_T \sum_{(i,j)\in\mathcal{E}} ||\mathbf{T}_j^i - \bar{\mathbf{T}}_j^i||^2, \tag{8}$$

which is a non-convex optimization due to the non-linear terms in the orientation of the robot. Roughly speaking, the optimization problem looks for the estimate $\mathbf{T}_j^i$ that minimizes the mismatch with respect to the measurements $\bar{\mathbf{T}}_j^i$.

(a) (b)

Fig. 1: An simple example of an undirected (1a) and directed (1b) graph with three nodes, and three edges of which one is a loop closure

III. Proposed Pose-graph Optimization Approach

We employ the current pose-graph framework to formulate new relations and constraints for a pose-graph optimization strategy that averages the obtained poses, such that the optimized pose is as accurate as possible. For this purpose, a new library in python is created. The following section describes how to build a pose-graph, the optimization strategy, windowing of incoming data and afterward weighting the optimized results.

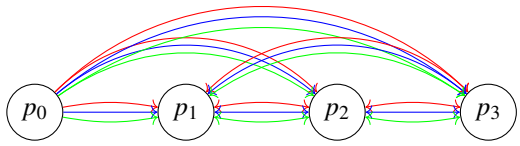


Fig. 2: Example of a pose-graph containing $\mathcal{N} = 4$ nodes with $\mathcal{K}$ constraints set up by k different localization methods indicated by the colors red, blue and green. The prior node is set up as the first node of the graph and each node in $p = [p_0, p_1, p_2, p_3]$ is initialized with an initial estimate $p(0) = [p_0(0), p_1(0), p_2(0), p_3(0)]$.

A. Building the Pose-Graph

Consider k different localization methods. Each localization method is synchronized and sampled such that they provide an equal amount of poses composing a trajectory of the mobile robot for the same time instances. In the pose-graph, nodes of the graph represent the absolute pose estimate, which is later optimized, and constraints represent the relative transform between nodes estimated by the k localization methods. In total, we have $\mathcal{N}$ nodes and $\mathcal{K}$ repetitive bidirectional constraints. Given the aforementioned settings, we build the pose-graph as follows:

1) The first step is to initialize the graph by introducing the nodes and setting up the origin. For example, in Figure 2, we set up the nodes $p = [p_0, .., p_3]$, where p_0 is the origin of the graph. The origin determines to which node all other nodes are optimized and remains constant during optimization.
2) Next, based on the localization method, edges of the graph are constructed. The edges from origin are unidirectional and between other nodes the edges are bidirectional. For example, in Figure 2, we consider three localization methods, so there are three edges between each node indicated by red, blue and green.
3) After, the pose graph is set, we initialize each node such that $p = p(0)$.
4) Finally, we assign each edge with the corresponding relative transformation derived from the absolute poses obtained by the localization methods.

By following these steps, we have build the pose-graph necessary for pose-graph optimization. An illustrative example is given in Figure 2.

B. Pose-graph Optimization Strategy

Given the pose-graph $\mathcal{G}$, we can define the optimization problem as follows:

$$\arg\min_{\{\mathbf{t}_j, \mathbf{R}_j | j=0,...,\mathcal{N}\}} \sum_{(i,j)_k} d(\mathbf{T}_i^j[k], \mathbf{T}((\mathbf{t}_i, \mathbf{R}_i), (\mathbf{t}_j, \mathbf{R}_j)), \tag{9}$$

where $\mathbf{t}_0$ and $\mathbf{R}_0$ are related to the origin and, therefore, known, and d denotes a distance metric. However, each optimized node influences the result of the other nodes in the pose-graph. Thus, to ensure that each node results in the most optimal value, instead of optimizing a whole set

of nodes at once, we optimize sequentially by considering all nodes fixed except for the to-be-optimized node. Thus, we can rewrite the optimization problem, considering $\mathbf{t}_i, \mathbf{R}_i$ is fixed, except for j, as:

$$\underset{\{\mathbf{t}_j,\mathbf{R}_j\}}{\arg\min} \sum_{i\in\mathcal{K}_j} d(\mathbf{T}_i^j[k], \mathbf{T}((\mathbf{t}_i,\mathbf{R}_i),(\mathbf{t}_j,\mathbf{R}_j))). \tag{10}$$

If we consider that:

$$\begin{aligned} \hat{\mathbf{t}}_j[k] &= \mathbf{T}_i^j[k]\mathbf{t}_i \\ \hat{\mathbf{R}}_j[k] &= \mathbf{R}_i^j[k]\mathbf{R}_i, \end{aligned} \tag{11}$$

we can describe the optimization problem as the minimization of the cost function:

$$C(\mathbf{t},\mathbf{R}) = \underset{\{\mathbf{t}_j,\mathbf{R}_j\}}{\arg\min} \sum_{i\in\mathcal{K}_j} \underbrace{d_{eucl}(\mathbf{t}_j,\hat{\mathbf{t}}_j[k])^2}_{1} + \underbrace{d_{\angle}(\mathbf{R}_j,\hat{\mathbf{R}}_j[k])^2}_{2}. \tag{12}$$

where $d_{eucl}(\mathbf{t}_a,\mathbf{t}_b)$ denotes the Euclidean distance between translation vectors $\mathbf{t}_a,\mathbf{t}_b \in (\mathbb{R}^2 \vee \mathbb{R}^3)$, and $d_{\angle}(\mathbf{R}_a,\mathbf{R}_b)$ denotes a distance metric between the rotations $\mathbf{R}_a,\mathbf{R}_b \in SO(3)$. $SO(3)$ is a Special Orthogonal Group denoting that the set of rotation forms a Lie-group:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^3 | \mathbf{R}^\top\mathbf{R} = \mathbf{I}^{3x3}, \det(\mathbf{R}) = 1\}. \tag{13}$$

Assuming that the position estimate $\mathbf{t}$ and rotation estimate $\mathbf{R}$ are independent, the cost function can be described by a translation minimization problem and rotation minimization problem, Equation (12): 1 and 2, respectively [35]. The translation minimization problem is a linear least-square problem and can be easily solved by calculating the mean of translations:

$$\mathbf{t} = \frac{1}{\mathcal{N}} \sum_{i=0}^{\mathcal{N}} \mathbf{t}_i. \tag{14}$$

However, the rotation minimization problem is a non-convex problem. Hence, rotation averaging is more difficult to solve [38]. We approach this problem by recalling that $\mathbf{R} \in SO(3)$, which is associated with Lie-algebra consisting of the set of all skew-symmetric 3x3-matrices. A skew-symmetric matrix Ω may be represented in terms of a 3-rotation vector $\mathbf{v} = (v_1, v_2, v_3)^\top$ by:

$$[\mathbf{v}]_x = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{15}$$

Every rotation in SO(3) can be represented by the angle-axis representation:

$$\mathbf{v} = \theta\hat{\mathbf{v}}, \tag{16}$$

where θ is the angle about an axis represented by a unit 3-vector $\hat{\mathbf{v}}$. The angle-axis representation is not unique and an alternative representation is given by $(2\pi - \theta)(-\hat{\mathbf{v}})$. To be unique, every rotation can be represented by a rotation through an angle by at most π. The connection between the two representations of a rotation ($\mathbb{R}^3$ and $SO(3)$) is giving by the exponential and logarithmic mapping. Using Rodrigues' formula the exponential map on $exp[\cdot]_x : \mathbb{R}^3 \to SO(3)$ can be computed as:

$$exp(\theta\hat{v}) = I + \sin(\theta)[\hat{\mathbf{v}}]_x + (1-\cos(\theta))([\hat{\mathbf{v}}]_x)^2. \tag{17}$$

And its inverse, the logarithmic map on $log(\cdot) : SO(3) \to \mathbb{R}^3$ by:

$$log(R) = \begin{cases} \arcsin(||\mathbf{y}||_2 \frac{\mathbf{y}}{||\mathbf{y}||_2}, & \text{if } \mathbf{y} \neq 0. \\ 0, & \text{if } \mathbf{y} = 0. \end{cases} \tag{18}$$

where $\mathbf{y} = (y_1, y_2, y_3)$ can be derived from the skew-symmetric matrix Ω:

$$\Omega = \frac{1}{2}(R - R^\top) = \begin{bmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{bmatrix}. \tag{19}$$

In this paper, we consider single rotation averaging in SO(3) under the geodesic distance metric by using a geodesic L_2-mean algorithm [39]. This solves the rotation minimization problem $C(R)$ given by:

$$C(R) = \sum_{i=1}^{n} d_{\angle}(R, R_i)^2 \tag{20}$$

and is also known as the Karcher mean of rotations [40]. We utilize a Karcher mean algorithm provided by [41] which is known to be convergent. The algorithm [1] is as follows:

Algorithm 1 Geodesic L_2-mean

Input: Set of rotations $\mathbf{R} = [\mathbf{R}_0, ..., \mathbf{R}_N]$, tolerance ε and maximum number of iterations i_{max}.
Output: The average rotation R.

Set the first rotation in the set of rotations $\mathbf{R}$ as initial value, $R = \mathbf{R}_0$.

1: **for** the set of rotations $\mathbf{R}$ **do**
2: Compute the mean of rotations $r = mean(log(R^\top\mathbf{R}))$.
3: **end for**
4: **while** $norm(r) \geq \varepsilon$ and the number of iterations $i \leq i_{max}$ **do**
5: Update $R = R\exp(r)$.
6: Increase the number of iterations $i = i + 1$.
7: **end while**

Where we consider the following commonly used distance metrics between two rotations $\mathbf{R}_a$ and $\mathbf{R}_b$:

- **Angular Distance:** The angular distance, also known as geodesic distance, is defined to be the angle corresponding to the relative rotation $\mathbf{R}_a\mathbf{R}_b^T$, which can always be chosen such that $0 \leq \theta \leq \pi$ by, if necessary, reversing the direction of axis. Thus,

$$d_{angle}(\mathbf{R}_a,\mathbf{R}_b) = \theta = ||log(\mathbf{R}_a,\mathbf{R}_b^T)||_2, \tag{21}$$

where $log(\mathbf{R})$ denotes the logarithmic map, see Equation (18). We could equally write this for relative

[1] For the weights introduced in Section III-D, line 2 should be updated to $r = \frac{1}{w}\sum_{i=0}^{len(\mathbf{R})}(w[i] * log(R^\top\mathbf{R[i]}))$.

rotations $\mathbf{R}_a^T\mathbf{R}_b$, $\mathbf{R}_a\mathbf{R}_b^T$ or $\mathbf{R}_a^T\mathbf{R}_b$ since it is all the same due to the distance metric being bi-invariant ($SO(3) \times SO(3) \to \mathbb{R}^+$).

- **Chordal Distance:** The chordal distance is the Frobenius norm of $\mathbf{R}_a - \mathbf{R}_b$:

$$d_{chord}(\mathbf{R}_a, \mathbf{R}_b) = ||\mathbf{R}_a - \mathbf{R}_b|| \tag{22}$$

and is related to θ using Rodrigues Formula, see Equation (17), as follows:

$$d_{chord}(\mathbf{R}_a, \mathbf{R}_b) = 2\sqrt{2}\sin\frac{\theta}{2}. \tag{23}$$

- **Quaternion Distance** The quaternion distance between two rotations is defined by:

$$d_{quat} = (\mathbf{R}_a, \mathbf{R}_b) = ||\mathbf{r}_a - \mathbf{r}_b||, \tag{24}$$

which describes the Euclidean distance between two quaternion representations $\mathbf{r}_a, \mathbf{r}_b$ of $\mathbf{R}_a, \mathbf{R}_b$ respectively. As this distance metric is also bi-invariant, we can relate this to the angular distance as follows:

$$d_{quat} = 2\sin(\frac{\theta}{4}). \tag{25}$$

Since we consider the angular distance d_{angle} to be the rotation angle between rotation $\mathbf{R}_a\mathbf{R}_b^T$, the intrinsic metrics induced by these three metrics ensure that all these metrics are essentially the same, except for a scaling factor, for small errors:

$$\begin{aligned} d_{angle} &= \theta, \\ d_{chord} &\approx \sqrt{2}\theta, \\ d_{quat} &\approx \frac{\theta}{2}. \end{aligned} \tag{26}$$

Thus, due to the strong coupling in Equation (26), Algorithm 1 allows to select any of these angular metrics for solving the average rotation. For convenience, we consider the distance metric $d_\angle$ to be the angular distance d_{angle}. Thus, the optimization problem becomes:

$$\underset{\{\mathbf{t}_j, \mathbf{R}_j\}}{\arg\min} \sum_{j\in\mathcal{N}} d_{eucl}(\mathbf{t}_j, \hat{\mathbf{t}}_j[k])^2 + \sum_{j\in\mathcal{N}} d_{angle}(\mathbf{R}_j, \hat{\mathbf{R}}_j[k])^2, \tag{27}$$

which is giving by the following optimization algorithm:

Algorithm 2 Pose-graph Optimization with Translation and Rotation Averaging

Input: A pose-graph containing prior node N_p, to-be-optimized nodes N, initial estimates E and constraints K spanned by localization methods k, tolerance ε and maximum number of iterations i_{max}.
Output: The optimized graph G containing optimized poses.

Initialize the *error*, $error[N] = \infty$ and $error[N_p] = 0$, and the graph $G = E$.

while $abs(max(error)) \geq \varepsilon$ and the number of iterations $i \leq i_{max}$ **do**
  for each node in N **do**
    Calculate the poses-to-average $P[N] = K_k[N] \cdot N$.
    Average the poses-to-average $P_{avg} = mean(P[N])$.
    Calculate the new error: $new\_error = norm(P_{avg} - G[N])$.
    Update the graph: $G[N] = P_{avg}$.
    Update the $error[N] = new\_error$.
  end for
  Increase iterations $i = i + 1$.
end while

C. Sliding Through Data

As the mobile robot progresses through the environment and receives new pose estimations over time, our incoming data keeps growing. In order to achieve real-time optimization of the trajectory of the robot, we consider a sliding window through the obtained data. This sliding window w is an arbitrary variable that determines the amount of data that should be optimized, from the current time instance t to the past $t-(w-1)$. Note that we have $(w-1)$ since we start from time instance t_0. For the given pose estimates of the sliding window, we optimize locally by building the pose-graph of window w. The size window w should be chosen carefully. When the window size is chosen relatively big, the noisy measurements in the obtained data will go unnoticed, as well as increasing the computational time drastically. However, when it is chosen too small, the noisy measurements might dominate the optimization and the results can become biased. Consider the following example of an mobile robot traversing through the environment obtaining in total $p = [p_0, ..., p_N]$ consecutive pose estimates over time instance $t = [t_0, ..., t_N]$. From the time instance that $t = t_{w-1}$, we start to locally optimize the received poses by building the pose-graph of $p = [p_{t-(w-1)}, ..., p_t]$ given the constraints obtained by k localization methods and the initial pose estimates equal to zero-pose ([0,0,0]). From the local optimization we obtain optimized poses $p^* = [p^*_{t-(w-2)}, ..., p^*_t]$ with respect to the prior node $p_{t-(w-1)}$ in the window w. After optimization we receive new data from the mobile robot and the window moves one time instance, such that $t > t_{w-1}$. If available, the obtained locally optimized poses from the previous window are used as the initial pose estimate for the pose-graph optimization of the next window. In contrast, the newly received pose is initially estimated as zero-pose. This process

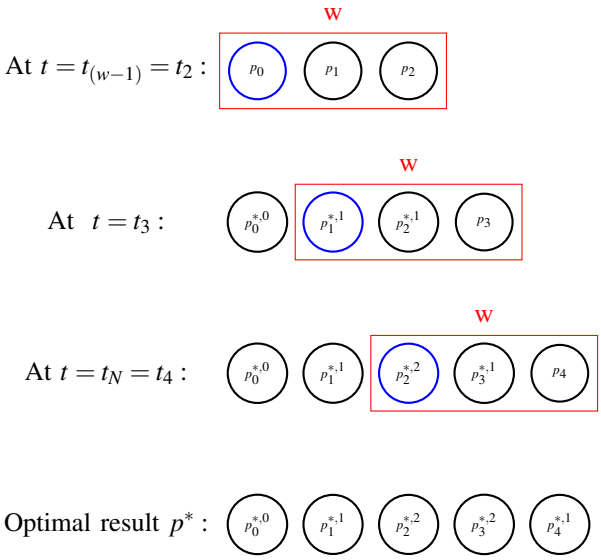


Fig. 3: Illustrative example of the functioning of a sliding window $w = 4$ through the data $N = 5$ for local pose-graph optimization.

continues until the last pose p_N is received at time instance $t = t_N$, thus, performing $N-(w-1)$ local optimizations to obtain optimized poses $p^* = [p^*, ..., p_N^*]$. Note, that due to the fact that we only have $N-(w-1)$ optimizations and not N, depending on the amount of poses p and the window w, the considered poses are optimized from 0 up to $w-1$ times. An illustrative explanation is given in Fig. 3, where we consider $w = 3$ (see red square) and $N = 4$. Furthermore, we introduce a variable i, which keeps track of how many times each pose is optimized. The first local optimization is performed at $t = t_{w-1} = t_2$ for the poses p_1 and p_2 with respect to the prior node p_0 (blue). After optimization we have obtained poses $p^* = [p_0^{*,i}, p_1^{*,i}, p_2^{*,i}]$ with $i = [0,1,1]$. Next, we obtain a new pose at $t = t_3$ and the window w shift by one time instance, such that $p_2^{*,1}$ and $p_3^{*,1}$ become the to-be-optimized poses with respect to the new prior pose $p_1^{*,1}$. Thus, we obtain $p^* = [p_0^{*,i}, p_1^{*,i}, p_2^{*,i}, p_3^{*,i}]$ with $i = [0,1,2,1]$. Finally, we optimize at time instance $t = t_N$ to obtain the final optimized poses $p^* = [p_0^{*,i}, p_1^{*,i}, p_2^{*,i}, p_3^{*,i}, p_4^{*,i}]$ with $i = [0,1,2,2,1]$. Hence, the amount of optimizations of each optimized pose p^* is as follows:

$$\begin{cases} i = 0, & \text{if time } t = t_0. \\ 1 \leq i < w-1, & \text{if time } t_1 \leq t < t_{w-1}. \\ i = w-1, & \text{if time } t_{w-1} \leq t < t_{N-(w-1)}. \\ N-(w-1) \leq i \leq 1, & \text{if time } t_{N-(w-1)} \leq t \leq t_N. \end{cases} \tag{28}$$

D. Quantifying the Localization Performance by Weighting

Based on the local pose-graph optimization we can address the quality of each considered localization method k at t_N based on the window of instances $t = [t_{N-(w-1)}, ..., t_N]$ by evaluating the error with respect to the obtained optimized poses at the same time instances. The error of the translation ($\mathbf{t}$) and rotation ($\mathbf{R}$) components for each separate localization methods k is as follows:

$$\begin{aligned} err_{trans}[k] &= \frac{1}{\mathcal{N}} \sum_{i=0}^{\mathcal{N}} (d_{eucl}(\mathbf{t}_i^*, \mathbf{t}_i[k])^2 \\ err_{rot}[k] &= \frac{1}{\mathcal{N}} \sum_{i=0}^{\mathcal{N}} (d_{\angle}(\mathbf{R}_i^*, \mathbf{R}_i[k])^2, \end{aligned} \tag{29}$$

such that we obtain k translation and rotation errors for each node $i \in \mathcal{N}$. Based on these errors the weights for each localization method are calculated accordingly by:

$$weight[k] = W_0 + \frac{(1-W_0) * ((\sum_{i=0}^{k} err[i]) - err[k])}{\sum_{i=0}^{k} err[i]}, \tag{30}$$

where W_0 is a constant factor which describes the minimum value of the weight, such that when $W_0 ! = 0$ we do not disregard a localization method completely. Thus, the weights can be a value between W_0 up to 1. As we obtain these weights, we want to use them for the next incoming data, therefore the optimization algorithm becomes:

Algorithm 3 Weighted Pose-graph Optimization with Translation and Rotation Averaging

Input: A pose-graph containing prior node N_p, to-be-optimized nodes N, initial estimates E and constraints K spanned by localization methods k, tolerance ε, maximum number of iterations i_{max}, the weights w labeled to the localization methods.
Output: The optimized graph G containing optimized poses.

Initialize the *error*, $error[N] = \infty$ and $error[N_p] = 0$, and the graph $G = E$.

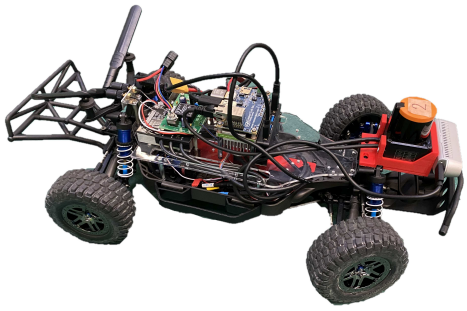
1: **while** $abs(max(error)) \geq \varepsilon$ and the number of iterations $i \leq i_{max}$ **do**
2: **for** each node in N **do**
3: Calculate the poses-to-average $P[N] = K_k[N] \cdot N$.
4: Label each pose-to-average with a weight based on Equation (30): $w[P] = w[k] \in K_k[N]$.
5: Average the poses-to-average $P_{avg} = sum(w[P] * P[N])/sum(w[P])$.
6: Calculate the new error: $new\_error = norm(P_{avg} - G[N])$.
7: Update the graph: $G[N] = P_{avg}$.
8: Update the $error[N] = new\_error$.
9: **end for**
10: Increase iterations $i = i + 1$.
11: **end while**

IV. Experimental Validation

In this section, we perform the experimental validation of the approach introduced in the previous section. We start by introducing the experimental setup, including the platform and considered localization methods. Afterward, we compare the quality of the weighting with the 'ground truth' provided by a motion capture system.

A. Experimental Setup

The experiment is conducted with a teleoperated mobile robot, F1tenth platform [42], following an s-shaped trajectory with a speed of $1m/s$ in a rectangular work-area for a typical indoor environment, see Figures 4. The considered mobile robot is a 1-to-10th scaled miniature racecar, equipped with Light Detection and Ranging sensor (LiDAR). The laser data obtained by the LiDAR is used as the input for the considered localization methods provided in ROS, in this case, Hector SLAM (filter-based) [43], Gmapping (particle-filter based) [44], Google Cartographer (pose-graph based) and Laser Scan Matcher (LSM) (scan-matching based) [33]. We synchronize the poses received from each localization method at a sampling frequency of half the frequency of the slowest localization method by using the package message_filters available in ROS [45]. This package subscribes to messages from multiple sources and outputs them only if it has received a message on each of those sources with approximately the same timestamp. After we have received the synchronized poses, we apply our approach with a window size $w = 5$ to obtain the averaged pose estimation based on our pose-graph optimization approach. The 'ground truth' is provided by the motion capture system, Optitrack, which realizes high-speed low-latency (millimeter) tracking of the vehicle `https://optitrack.com/applications/robotics/`.



(a) The considered F1Tenth platform equipped with LiDAR

(b) The experimental work-area with drawn motion

Fig. 4: The experimental setup of the proposed experiment

The considered s-shape trajectory can be described as follows: First, the vehicle drives straight for approximately 5 meters; Then it makes a u-turn in a counterclockwise direction with maximum steering angle; Afterward, it continues driving straight for approximately 5 meters; Next, it makes a u-turn in a clockwise direction with maximum steering angle; Finally, it drives straight again for approximately 5 meters and stops.

B. Discussion

To validate if the weighting of each localization method accurately represents the real-world scenario, we compare the trajectory of each localization method, including our approach, with the trajectory obtained by the motion capture system based on the Absolute Trajectory Error (ATE) for the translation and rotation components separately:

$$\begin{aligned} ATE_{trans} &= |d_{eucl}(\mathbf{t}_a^*, \mathbf{t}_b[k])^2| \\ ATE_{rot} &= |d_{\angle}(\mathbf{R}_a^*, \mathbf{R}_b[k])^2|. \end{aligned} \tag{31}$$

We use statistical metrics such as Root Mean Square Error (RMSE), mean, median, standard deviation, and maximum error, to represent the evaluation as well as the ideal weighting calculated from the absolute trajectory errors using Equation (30). Furthermore, based on the obtained trajectories, we judge if the changes occurring in weighting are in line with what we can observe from the trajectories.

For the conducted experiment the trajectories estimated by each localization method are depicted in Figure 5a along with the optimized trajectory (optimized) and trajectory given by the motion capture system (mocap). The optimized trajectory is obtained via our pose-graph averaging optimization approach, given the weighting in Figure 6 for the poses from laser scan matcher (lsm), hector slam (hector), gmapping (gmap), and Google Cartographer (google). The trajectory depicted in Figure 5a can be divided into three regions in which significant changes in weighting occur. In the first region, see Figure 5b, we can observe that after 5 seconds, google cartographer starts to deviate from the alternative localization methods. Therefore, we expect that the weighting of google cartographer becomes less. It can be observed in Figure 6d that at this time instance google cartographer is indeed starting to be weighted less. However, at the start of the second region, indicated by Figure 5c google cartographer is able to recover and, on the contrary, gmapping starts to fail with respect to alternatives. This shift of accuracy is indicated by the weighting, as it can be seen from Figures 6c and 6d that gmapping starts to perform less while google cartographer recovers. Finally, at the last region of the trajectory, we can notice that gmapping slowly starts to recover, and google cartographer slowly starts to deviate at the end. This behavior is supported by our estimated weighting as gmapping obtains a higher weighting and google cartographer a lower. Meanwhile, both lsm and hector show relatively high performance throughout the whole trajectory and, thus, are weighted higher (see Figures 6a and 6b). Remarkably, the rotation weight of lsm, see Figure 6a is 1 or very close to 1 throughout the whole trajectory. Thus, indicating that the optimized rotation is equal to the estimated rotation of lsm. This seems highly unlikely, but can be explained by revising the Karcher mean algorithm (Algorithm 1). In this algorithm, the average rotation is initialized as

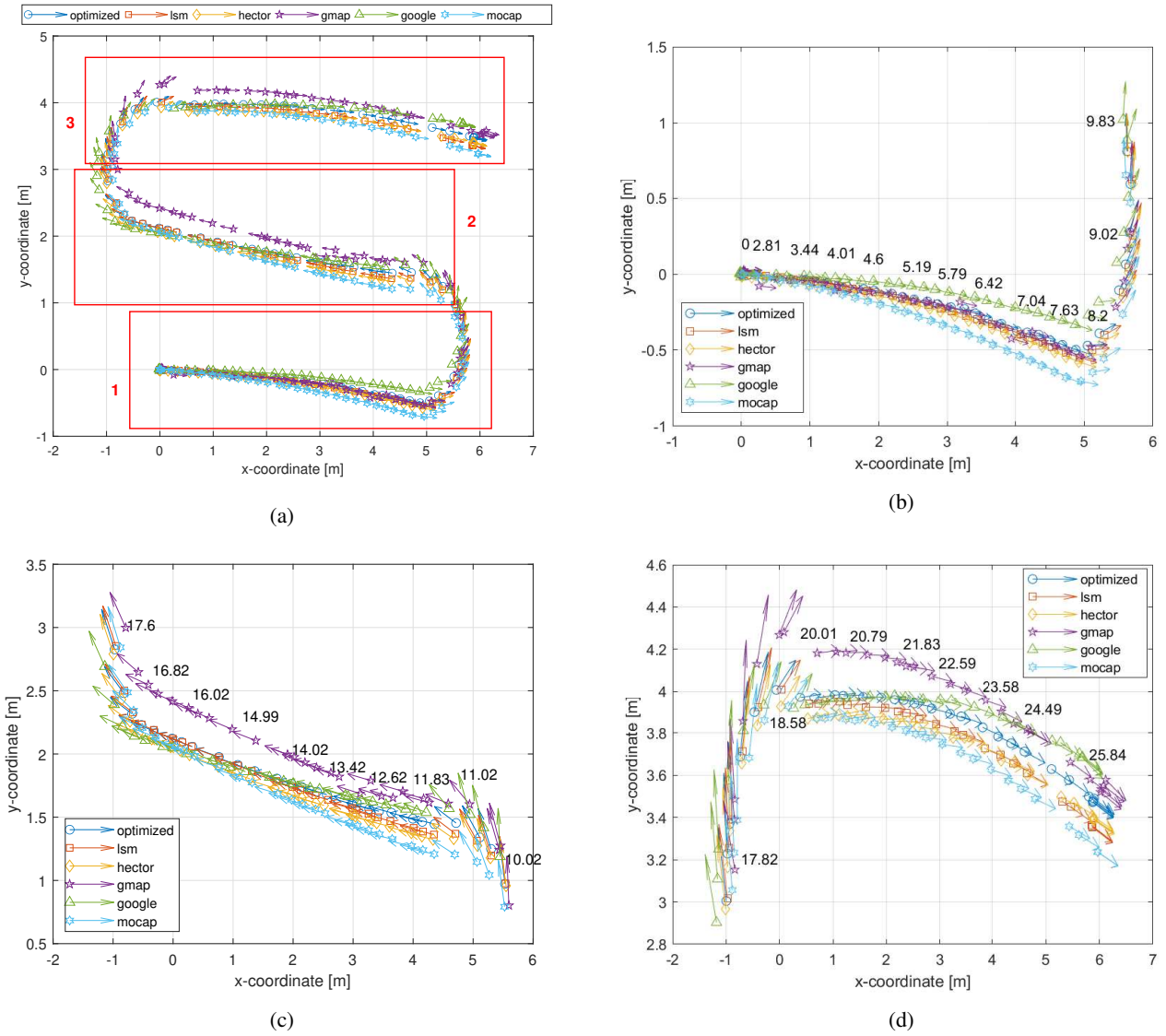


Fig. 5: (a) The obtained trajectories of the considered localization methods given by a configuration of poses indicated by laser scan matcher (lsm), hector SLAM (hector), gmapping (gmap), and google cartographer (google), as well as our approach and the motion capture system indicated by graph and mocap respectively. The red regions indicate the instances when significant changes in performance happen. (b) The zoomed version of the first region. Every third illustrated pose is labeled with the corresponding time instance at which the pose is obtained. (c) The zoomed version of the second region. (d) The zoomed version of the third region.

the first rotation of a given list of rotations, which in this case is lsm. When the tolerance is chosen relatively high or/and the maximum number of rotations is chosen relatively low, the algorithm terminates too early and the result does not fully converge. However, when decreasing the tolerance or/and increasing the maximum number of iterations, the trade-off occurs that the computational time of rotation averaging drastically increases. For example in Figure 7, in which we have increased the number of iterations for the Karcher mean algorithm from 10 to 100 while remaining the same tolerance, our approach results in significantly less optimized poses during rotation. Meanwhile, the weighting of the methods does not significantly change. Hence, the proposed weighting represents the actual picture, which we can observe from the trajectories, and is comparable with the ideal weighting profile obtained by the motion capture system.

Overall, based on the observed trajectories, gmapping and google cartographer estimate the poses quite poorly in comparison with hector slam and laser scan matcher. This is confirmed by the evaluation metrics related to the ATE, see Table I. From the presented table we can observe that from the localization methods, hector slam can be considered the best localization method in the proposed environment. Closely followed by laser scan matcher, which overall performs very similar to hector slam, whereas gmapping

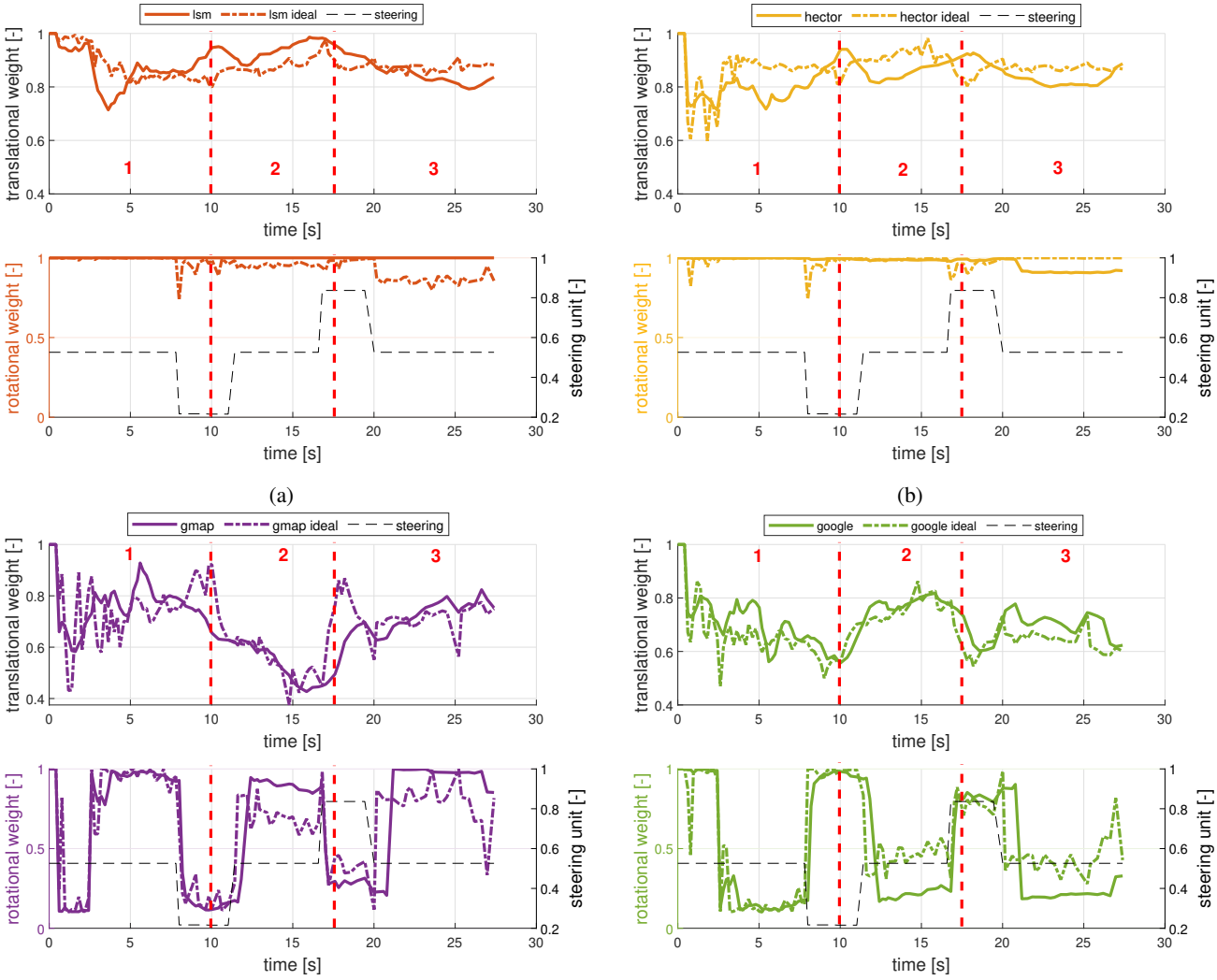


Fig. 6: The estimated weights at each time instance for: (a) laser scan matcher (lsm), (b) hector SLAM (hector), (c) gmapping (gmap), (d) google cartographer (google), versus ideal weighting given the motion capture system.

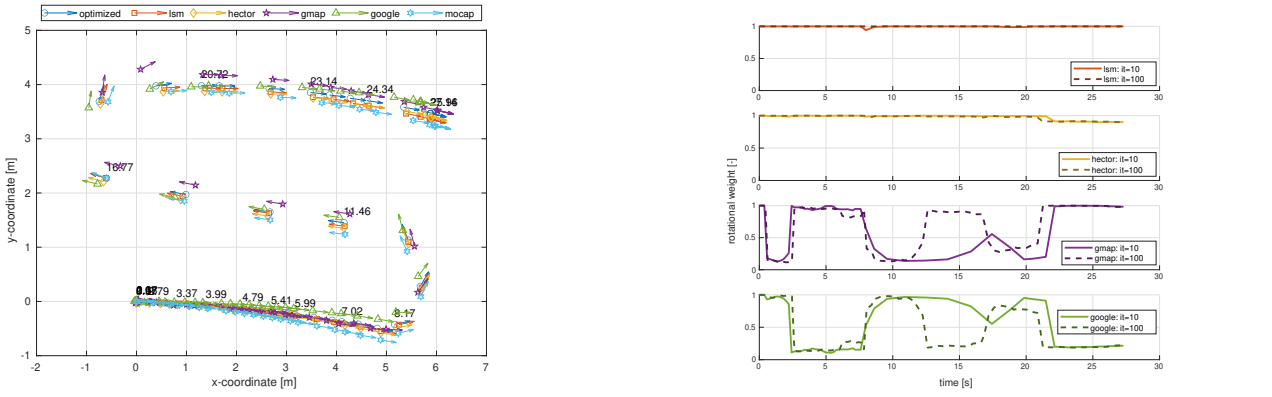


Fig. 7: The comparison of experimental results with maximum rotational iterations of 10 and 100: (a) The obtained trajectories of the considered localization methods given for a maximum rotational iterations of 100, with similar threshold as Figure 5a. (b) The weighting of each localization method with a maximum rotational iterations of 10 and 100.

Absolute Trajectory Error (ATE)	Laser Scan Matcher	Hector SLAM	Gmapping	Google Cartographer	Standard average	Optimized uniformly weighted	Optimized weighted	Optimized ideal weighted
RMSE translation	0.322	0.306	0.477	0.511	0.404	0.384	0.380	0.366
RMSE rotation	0.019	0.016	0.086	0.048	0.042	0.024	0.020	0.022
Mean translation	0.104	0.094	0.228	0.261	0.172	0.147	0.144	0.134
Mean rotation	3.787e-4	2.623e-4	7.344e-3	2.270e-3	2.564e-3	5.669e-4	4.093e-4	4.676e-4
Median translation	0.112	0.089	0.238	0.246	0.171	0.147	0.145	0.133
Median rotation	1.394e-4	1.555e-06	6.354e-4	1.991e-3	6.919e-4	1.352e-4	1.324e-4	6.015e-05
STD translation	0.065	0.060	0.147	0.166	0.110	0.100	0.093	0.092
STD rotation	7.744e-4	7.474e-4	0.017	2.710e-3	5.332e-3	1.441e-3	8.033e-4	1.035e-3
Max translation	0.230	0.214	0.637	0.570	0.413	0.339	0.320	0.318
Max rotation	5.450e-3	3.652e-3	6.853e-2	1.898e-2	2.415e-2	9.898e-3	5.450e-3	5.868e-3

TABLE I: Statistical evaluation via absolute trajectory error for translation and rotational components of the full trajectory of laser scan matcher, hector slam, gmapping, google cartographer, the standard average of the 4 mentioned localization methods, the uniformly weighted pose-graph optimized average of the 4 mentioned localization methods, the pose-graph optimized average with self-assessed weights for the 4 mentioned localization methods, the pose-graph optimized average with ideal weights obtained by the motion capture system for the 4 mentioned localization methods

and google cartographer demonstrate worse results for our experiment. Notably, in rotation laser scan matcher performs better than hector slam, while in translation the opposite is observed according to both STD, RMSE, and maximum error.

The significance of the proposed optimization approach can be highlighted by comparing it with the standard average given the four localization methods. We can notice from the table I that by applying optimization and weights the statistical measures are closer to the results from the motion capture system (optimized ideal weighting). Hence, we can conclude that we can assess the performance of localization without using external devices or human interaction as intended. We can see that when a localization method starts to ill-perform, the weighting noticeable change; indicating that the localization performance of that specific method starts to lack. We can also observe that the failing of a method does not mean it is completely excluded from the pose-graph optimization, therefore being able to recover from a low weighting.

V. Conclusions

This paper presents an online pose-graph optimization approach for self-assessing the localization performance of multiple localization methods running in parallel. By applying the proposed approach, we enable robotic platforms to autonomously self-assess the obtained localization poses from each localization method as well as the final result necessary for achieving SLAM synergy. Thereby, being able to answer the question, when to share the localization data from the well-performing localization method to the less well-performing. The approach has been proven to accurately represent the localization performance of each implemented localization method by determining weights based on the translation and rotation error with respect to the pose-graph optimization approach. The potential future research includes extensive experiments for validation of the proposed approach for more and/or different localization methods available on ROS, and more challenging environments. Advanced rotation averaging algorithms are necessary for more accurate optimization of the rotation with a reduced computational load. Furthermore, since we have answered the question of when to share data, the new research question regarding achieving SLAM synergy becomes: how to share the localization result from the well-performing localization method to the less performing localization method?

Appendix I
Localization methods implemented in ROS

A. Gmapping

Gmapping is a ROS Debian package that provides particle filter based SLAM [44], and needs, in contrast to Hector SLAM and Googe Cartrographer, odometry information to function. As the system setup in this project does not provide such information, the Laser Scan Matcher package is used to provide the necessary odometry information. Gmapping uses a Rao-Blackwellized Particle Filter for localization, in which each particle carries an individual map of the environment [46]. It computes an accurate proposal distribution by not only taking the robots movement, as well as the the most recent observation into account. Furthermore, it uses an adaptive resampling technique to maintain a reasonable variety of particles and thus reducing the risk of particle depletion.

B. Google Cartographer

Google Cartographer is an pose-graph optimization based SLAM method provided as a open-source library with ROS wrapper that can work with and without odometry information [47]. It is a system that provides real-time SLAM in 2D

and 3D compatible for multiple sensor configurations and platforms [27]. Additonally, Google Cartographer is a combination of local and global SLAM where both approaches try to optimize the optimal transform between the scans. In the local SLAM approach each consecutive scan is matched against a submap, which is a piece of the environment, using a non-linear least squares problem based scan matcher [48]. The submap construction is an iterative process of continuously aligning scan and submap coordinate frames, where the submap is created from a few consecutive scans. However, before inserting scans into a submap, the Ceres-based scan matcher [48] is used to optimize the scan pose of the current local submap. The global SLAM periodically reaarranges the sub-maps tp reduce the localization error and tries to prevent drift. In contrast to Hector SLAM's multi-resolution grid maps, google Cartographer introduces a branch and bound algorithm to efficiently compute the optimal grid-accurate match over large search windows. Finally, the global map is constructed by combining the current submap with the finished submaps.

C. Hector SLAM

Hector SLAM is a EKF filter and pose graph optimization based SLAM method provided as a ROS Debian package that can work with and without odometry information [43]. It combines a robust scan matching approach using a LIDAR system with a 3D attitude estimation system based on inertial sensing [49]. However, in this project we are only interested in the 2D SLAM of this package as no Inertial Measurement Unit (IMU) for the Extend Kalman Filter (EKF) based 3D state navigation is present. In this package, the scan information of the LIDAR is converted into a point cloud of scan endpoints. These point clouds are then preprocessed by only taking into account endpoints within a threshold of the intended scan plane. As occupancy grid maps have a discrete nature, the precision that can be achieved is limited. Therefore an interpolation scheme through bilinear filtering is introduced for both estimating occupancy probabilities as well as derivatives. The Hector SLAM scan matching approach is based on an optimization of the alignment of beam endpoints with the map discovered so far. The scans get aligned to the existing map and thus matching is implicitly performed with all preceding scans. Hector SLAM is based on gradient ascent optimization approach and is potentially prone to get stuck in a local minima. This problem is alleviated by introducing a multi-resolution map representation. Essentially Hector SLAM is using multiple occupancy grid maps with each coarser map having half of the resolution of the preceding map. These different maps are kept in the memory and simultaneously updated using the pose estimates provided by the alignment process. The alignment process starts at the coarsest map level and the resulting pose estimate is used as the start estimate of the next level.

D. Laser Scan Matcher

Laser Scan Matcher is a Debian package of ROS [33], which provides the state of the mobile robot by using *PLICP* scan matching. The notable feature of this package is that it does not use scan-to-scan *PLICP*, but instead uses scan-to-keyframe *PLICP*. As noise in the scans are inevitable, the error over time accumulates even in stationary position, known as drift. By introducing a keyframe, which only changes when the vehicle has moved or rotated a certain distance or angle, this stationary error is minimized as the keyframe does not change.

REFERENCES

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Springer Publishing Company, Incorporated, 2016.

[2] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The slam problem: a survey," *Artificial Intelligence Research and Development*, pp. 363–371, 2008.

[3] H. Durrant-Whyte and T. Bailey, ""simultaneous localisation and mapping (slam): Part i the essential algorithms"," *Robotics and Automation Magazine*, vol. 13, 01 2006.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[5] K. Krinkin, A. Filatov, A. Filatov, A. Huletski, and D. Kartashov, "Evaluation of modern laser based indoor slam algorithms," vol. 426, 05 2018, pp. 101–106.

[6] M. A. Abdulgalil, M. H. El-Alfy, M. M. Nasr, and A. Khamis, "Multi-robot slam: An overview."

[7] M. Pfingsthorn, B. Slamet, and A. Visser, "A scalable hybrid multi-robot slam method for highly detailed maps," in *RoboCup 2007: Robot Soccer World Cup XI*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 457–464.

[8] S. Joo, U. Lee, T. Kuc, and J. Park, "A robust slam algorithm using hybrid map approach," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, pp. 1–2.

[9] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, vol. 94, pp. 1384 – 1397, 08 2006.

[10] P. Buschka, "An investigation of hybrid maps for mobile robots," 01 2005.

[11] M. Rojas-Fernández, D. Mújica-Vargas, M. Matuz-Cruz, and D. López-Borreguero, "Performance comparison of 2d slam techniques available in ros using a differential drive robot," in *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 2018, pp. 50–58.

[12] R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map comparison of lidar-based 2d slam algorithms using precise ground truth," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1979–1983.

[13] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardös, "A comparison of slam algorithms based on a graph of relations," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2089–2095.

[14] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," in *2018 International Conference on Intelligent Systems (IS)*, 2018, pp. 400–407.

[15] J. Leonard and H. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," 2000.

[16] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," 2002.

[17] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," *Proc. IJCAI Int. Joint Conf. Artif. Intell.*, 06 2003.

[18] A. I. Eliazar and R. Parr, "Dp-slam 2.0," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 2, 2004, pp. 1314–1320 Vol.2.

[19] B. Steux and O. E. Hamzaoui, "tinyslam: A slam algorithm in less than 200 lines c-language program," in *2010 11th International Conference on Control Automation Robotics Vision*, 2010, pp. 1975–1979.

[20] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[21] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[22] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, no. 4, p. 333–349, Oct. 1997. [Online]. Available: https://doi.org/10.1023/A:1008854305733

[23] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *I. J. Robotic Res.*, vol. 25, pp. 403–429, 05 2006.

[24] H. Durrant-Whyte, N. Roy, and P. Abbeel, *A Linear Approximation for Graph-Based Simultaneous Localization and Mapping*, 2012, pp. 41–48.

[25] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," 10 2010, pp. 22–29.

[26] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160.

[27] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[28] O. Bengtsson and A.-J. Baerveldt, "Robot localization based on scan-matching—estimating the covariance matrix for the idc algorithm," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 29–40, 2003. [Online]. Available: https://doi.org/10.1016/S0921-8890(03)00008-3.

[29] J. Minguez, F. Lamiraux, and L. Montesano, "Metric-based scan matching algorithms for mobile robot displacement estimation," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3557–3563.

[30] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[31] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 19–25.

[32] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 4387–4393.

[33] A. C. Ivan Dryanovski, William Morris, "Laser scan matcher," 2019 (accessed January 21th, 2021), http://wiki.ros.org/laser_scan_matcher.

[34] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[35] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014. [Online]. Available: https://doi.org/10.1177/0278364914523689

[36] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 12 2010.

[37] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2262–2269, 2006.

[38] K. Wilson, D. Bindel, and N. Snavely, "When is rotations averaging hard?" in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 255–270.

[39] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[40] H. Karcher, "Riemannian center of mass and mollifier smoothing," *Communications on Pure and Applied Mathematics*, vol. 30, no. 5, pp. 509–541, 1977. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160300502

[41] J. Manton, "A globally convergent numerical algorithm for computing the centre of mass on compact lie groups," in *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, vol. 3, 2004, pp. 2211–2216 Vol. 3.

[42] F1Tenth, "F1tenth," 2020 (accessed January 15th, 2021), https://f1tenth.org/.

[43] J. M. Stefan Kohlbrecher, "Hector slam," 2014 (accessed January 20th, 2021), http://wiki.ros.org/hector_slam.

[44] B. Gerkey, "Hector slam," 2019 (accessed January 21th, 2021), http://wiki.ros.org/gmapping.

[45] D. T. Josh Faust, Vijay Pradeep, "message_filter," 2018 (accessed October 15th, 2021), http://wiki.ros.org/message_filters.

[46] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," 01 2005, pp. 2432–2437.

[47] Google, "Google cartographer ros," 2021 (accessed Februari 12th, 2021), https://google-cartographer-ros.readthedocs.io/en/latest/.

[48] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[49] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.