Sensor-Specific Parameter Optimization for Google Cartographer

Pascal Fina, Raimund Leitner and Armin Berger

***Abstract*— We present a SLAM parameter optimizer that integrates with Google's Cartographer framework to automatically identify configurations minimizing discrepancies between generated maps and ground truth. A cost function based on 2D alignment and geometric error over point clouds is optimized using Gaussian Process Expected Improvement (GPEI), balancing exploration and exploitation under noise. We evaluate the approach on three environments: a large office ($570\ m^2$) and two apartments (each $54\ m^2$ and $48\ m^2$). The optimizer reduces mapping error by 25–55% compared to trial-and-error, achieving final errors as small as $2\ \mathrm{cm}$.**

A key contribution is demonstrating that parameters optimized in smaller, controlled environments transfer effectively to larger and structurally different settings when the same range sensor is used. This transferability indicates that parameter behavior is primarily sensor-driven rather than environment-specific, reducing the need for repeated optimization. Overall, the framework lowers SLAM error while making parameter tuning more efficient and scalable than manual trial-and-error.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a fundamental component of robotic applications that aim to operate autonomously. As the name suggests, SLAM builds a map of a robot's surroundings from sensor data while simultaneously estimating the robot's trajectory within that map. The output is both a representation of the environment and the path traversed by the robot—essential information for enabling autonomous navigation. Among existing algorithms, Google's Cartographer [1] is one of the most widely used and stable SLAM frameworks.

Cartographer is an open-source framework that supports real-time 2D and 3D mapping by fusing data from range sensors and, optionally, inertial measurement units (IMUs). It employs pose graph optimization to refine the estimated trajectory and the resulting map as new sensor data is incorporated.

In previous research, a team from Infineon Technologies successfully integrated indirect Time-of-Flight (iToF) range data into Cartographer, generating promising indoor maps [2]. Unlike Light Detection and Ranging (LiDAR), iToF sensors emit modulated electromagnetic waves and measure the phase shift between emitted and reflected signals to estimate distances. Achieving robust results in Cartographer requires careful tuning of its SLAM parameters [3].

Although we configured Cartographer to be as deterministic as possible by enabling landmarks and following the recommended settings (including providing an empty `LandmarkList` and collating sensor data) [4], we still observed non-deterministic behavior across repeated runs and thus variations in the generated maps. To address this, we adopt Bayesian Optimization (BO), a family of Sequential Model-Based Optimization (SMBO) methods [5], to automatically tune SLAM parameters. Specifically, we employ Gaussian Process Expected Improvement (GPEI) to explore the parameter space efficiently. GPEI models the unknown objective function (in this case, SLAM error) as a Gaussian process and uses the Expected Improvement acquisition function to balance exploration and exploitation. This makes GPEI particularly well suited for optimizing many parameters simultaneously, handling noise in evaluations, and reducing the number of costly map-generation runs [6].

A critical step in this optimization is providing a robust cost function. To do so, the generated SLAM maps must be aligned with the ground truth as accurately as possible before error calculation. We therefore combine Google's Ceres Solver [7] with the 2D map alignment approach of Shahbandi and Magnusson [8], [9], achieving reliable alignment and error measurement.

It is important to note that our evaluation deliberately avoids standard benchmark datasets. While such benchmarks are widely used to compare SLAM systems, they are not well suited for our research question. The focus of this work is not on establishing state-of-the-art benchmark performance, but on analyzing how optimized parameters transfer across environments when the same range sensor is used. By holding the sensor constant and varying the environment, we isolate sensor-specific effects, which would be obscured in cross-sensor benchmark comparisons.

II. RELATED WORK

Recent research shows a growing interest in systematic parameter optimization for 2D SLAM systems, with a variety of optimization strategies being explored. Early work by Teame et al. (2020) [10] systematically analyzed the effects of parameter optimization in Gmapping on mapping accuracy in simulation. Their experiments revealed that while adjusting individual parameters produced only modest improvements, jointly optimizing multiple parameters—particularly those related to scan matching and resampling—yielded maps that aligned more closely with the ground truth. This work highlights the importance of collective parameter tuning and identified map update interval as a critical factor in optimization.

Expanding the methodological landscape, Nagarajan (2020) [11] investigated a multi-objective genetic algorithm (MOGA) for automatic parameter optimization of the RTAB-Map SLAM package in 2D indoor environments. By jointly optimizing three key RTAB-Map parameters, and validating through both simulation and real-robot (QBot2) experiments,

the study demonstrated that MOGA can effectively balance multiple map quality metrics. These results underscore the potential of evolutionary multi-objective optimization for robust and efficient SLAM parameter tuning across diverse environments and hardware platforms.

In a similar effort to automate and generalize the optimization process, Koide et al. (2021) [12] proposed an automatic hyperparameter tuning framework for LiDAR SLAM algorithms using SMBO, treating the SLAM system as a black box and employing surrogate models for efficient parameter search. They further introduced LiDAR-specific data augmentation to mitigate overfitting, demonstrating improved accuracy and robustness compared to manual tuning on both KITTI and real-world datasets. Their results confirmed that automatic SMBO-based tuning reduces manual effort while yielding superior, generalizable parameter sets.

Beyond algorithm-specific approaches, Trejos et al. (2022) [13] introduced a statistical methodology for characterizing, calibrating, and comparing five ROS-based 2D SLAM algorithms, including Cartographer and Gmapping. Using Plackett–Burman and factorial designs, they identified and tuned the most influential parameters to optimize pose accuracy, map quality, and computational efficiency. Their work demonstrated that systematic parameter calibration can significantly improve SLAM performance across multiple metrics, providing a rigorous and reproducible framework for parameter optimization.

Building on these advancements, Ahmed and Raafat (2023) [14] conducted an extensive experimental study, systematically fine-tuning all 32 initialization parameters of the Gmapping algorithm on a TurtleBot3 Burger in real indoor environments. Using image registration and similarity metrics, they significantly improved map accuracy, further emphasizing the substantial impact of careful parameter selection and tuning on SLAM performance.

III. Methodology

A. Optimization Framework Overview

To efficiently optimize the SLAM parameters, we adopt a two-phase approach. In the first phase, Sobol [15] sequences are used to broadly and evenly sample the parameter space. In the second phase, Bayesian Optimization with Gaussian Process Expected Improvement (GPEI) iteratively refines the search and identifies parameter sets that minimize map error.

B. Parameter Initialization and Space Exploration

Given the curse of dimensionality in high-dimensional parameter spaces, it is important to begin with a set of well-distributed samples. We use Sobol sequences to generate an initial design, ensuring that the space is evenly explored and guiding the optimizer toward promising regions. In our experiments, we found that the number of initial samples should scale quadratically with the number of parameters for effective coverage. While this scaling relationship may differ in other applications, it provides a practical balance between exploration and computational cost for our use case.

C. Bayesian Optimization with Gaussian Process Expected Improvement

Bayesian Optimization is a model-based method for optimizing expensive black-box functions. Specifically, we use GPEI, which employs a Gaussian Process surrogate to model the objective function and an Expected Improvement (EI) acquisition function to balance exploration and exploitation. At each iteration, GPEI proposes the next parameter configuration by maximizing EI, efficiently navigating toward regions of lower error. The process is repeated for a predetermined number of iterations, depending on the dimensionality of the parameter space.

Mathematically, the Expected Improvement at a candidate point x is defined as:

$$EI(x) = \mathbb{E}\left[\max\left(y^* - y(x), 0\right)\right] \tag{1}$$

where y^* denotes the best (minimum) observed value so far, and the expectation is taken over the predictive distribution of the Gaussian Process at x. This formulation quantifies the expected gain from sampling at x, given both the surrogate's predicted mean and its uncertainty. By always selecting the parameter set that maximizes EI, GPEI naturally balances exploration and exploitation, making it well suited for problems where evaluations are expensive or noisy [6].

D. Handling Non-Determinism and Noisy Evaluations

A major challenge in tuning SLAM parameters is the non-determinism of the evaluation metric. Even when Cartographer is configured to be as deterministic as possible by enabling landmarks and providing an empty `LandmarkList` as recommended in the Lua reference documentation [4], repeated runs with the same dataset and parameter configuration may still produce slightly different maps. This residual non-determinism may arise from multi-threading, floating-point effects, and randomization in components such as scan matching, making repeated evaluations necessary for robust parameter optimization. We then use the mean and standard deviation of the error as inputs to the GPEI model. Assuming Gaussian-distributed noise, GPEI can appropriately account for uncertainty and prioritize configurations likely to yield robust improvements.

E. Evaluation Function

1) Alignment Pipeline: A key requirement for optimizing SLAM parameters with GPEI is a robust evaluation function. A major challenge lies in accurately aligning the generated SLAM map with its ground truth in order to compute a meaningful error metric. Since this is a two-dimensional alignment task, both translation and rotation must be estimated.

After Cartographer generates a SLAM map, it is first converted into an occupancy grid, and spurious obstacle fragments mistakenly identified as walls (e.g., desks, chairs, and other furniture) are removed to simplify the alignment process. Following this preprocessing step, we apply the 2D map alignment method by Shahbandi and Magnusson [8], which segments both maps into meaningful regions using geometric decomposition.The algorithm generates candidate

alignments by matching regions of similar shape, estimates transformations for each candidate, and scores how well the regions overlap. The best-scoring transformation is selected as the final alignment [9]. However, experiments show that this approach alone does not always yield the best alignment, particularly when map quality is suboptimal.

To improve robustness, we incorporate Google's Ceres Solver as a refinement step. Ceres minimizes the sum of squared residuals by repeatedly linearizing the nonlinear alignment problem around the current estimate and solving the resulting linear system using algorithms such as Levenberg–Marquardt or Dogleg. As a local optimizer, Ceres requires a good initial guess to ensure convergence [7]. As an initial guess, we use the Shahbandi and Magnusson alignment result and then refine it using ceres Solver. To further enhance robustness, we also explore a small neighborhood around this guess, running Ceres for each candidate configuration. For each result, we compute the error between the aligned SLAM map and the ground truth. This combined strategy consistently outperforms region-based alignment alone [8].

2) Error Calculation Between SLAM Map and Ground Truth: To quantitatively assess the accuracy of the SLAM-generated map, we compute the geometric error between the aligned SLAM map and its ground truth reference. Both maps are first converted to 2D point clouds. Instead of assuming point-to-point correspondences, we employ a symmetric nearest-neighbor approach: for each point in the SLAM map, we calculate the Euclidean distance to its nearest neighbor in the ground truth map, and vice versa. These distances are computed efficiently using KD-trees [16].

We then aggregate these distances in both directions and report two metrics:

- the mean of all symmetric nearest-neighbor distances,
- the median of these distances,

with both metrics averaged over the two directions (SLAM $\rightarrow$ ground truth and ground truth $\rightarrow$ SLAM). This approach, sometimes referred to as the symmetric Chamfer distance [17], provides a robust and interpretable measure of how well the SLAM map matches the reference, accounting for differences in point density and map resolution.

IV. Experiments

A. Optimizing 2 Parameters on a large office floor

For the first experiment, we ran Google's Cartographer on the Infineon office floor in Graz, Austria, which covers an area of approximately 570 m^2 (*Office floor 1*). In this trial, we focused on optimizing two parameters: the sensor's minimum and maximum range, denoted as `min_range` [18] (*MinR*) and `max_range` (*MaxR*) in Cartographer. Sensor data falling outside this range is filtered out and excluded from the mapping process. Optimizing these two parameters reduced the average mapping error to 11.3 cm, compared to 22.4 cm with manual trial-and-error, corresponding to an improvement of nearly 50%. Figure 1 shows the lowest error achieved at each iteration, illustrating the progression of the optimization process. It should be noted that the reported values represent the best errors found during optimization and do not necessarily correspond to the absolute global minimum. This observation applies equally to Figure 3, Figure 6, and Figure 9.

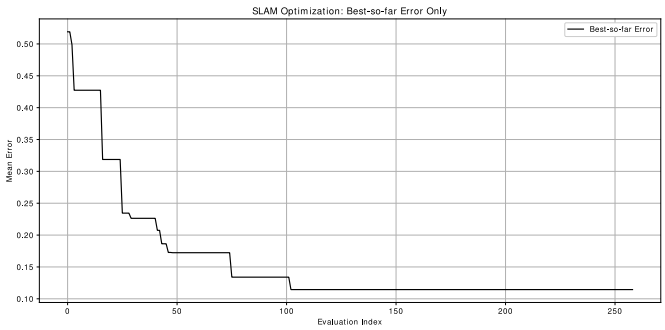

Fig. 1. *Office floor 1*: Best error (in meter) achieved at each iteration

As shown in Figure 2, the optimization process with two parameters is considerably less complex than the five-parameter case (Figure 4). The rolling median decreases rapidly during the early iterations and stabilizes after around iteration 100, at which point the minimum error is already reached (see Figure 1). Beyond this point, additional iterations do not provide further improvements, and the rolling median remains relatively constant.

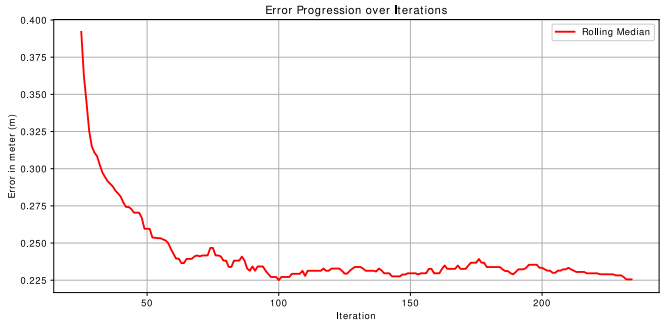

Fig. 2. *Office floor 1*: Rolling median of all recorded errorsß

From this experiment we also conclude that when the maximum range parameter is set too low, Cartographer is unable to build a map or track the trajectory. This underscores the importance of the `max_range` parameter in Google Cartographer.

B. Optimizing 5 Parameters on a Large Office Floor

Using the same dataset as before, we extended the optimization by tuning three additional parameters, resulting in a total of five. In particular, we adjusted *constraint_builder.min_score* (*MinSc*), which defines the minimum scan match score required for Cartographer to accept a constraint between a scan and a submap, thereby filtering out poor matches that could reduce map quality. We also tuned *optimize_every_n_nodes* (*OptN*), which

determines how frequently Cartographer performs global optimization to update all poses and submaps, influencing how quickly corrections such as loop closures are incorporated. Lastly, we included $submaps.num\_range\_data$ (`SubmapN`), which sets the number of range data (laser scans) added to each submap before a new one is started, thereby controlling submap size and indirectly affecting both the map's granularity and computational load [18]. Tuning these parameters reduced the mapping error to 10.2 cm (*Office floor 2*), as shown in Figure 3.

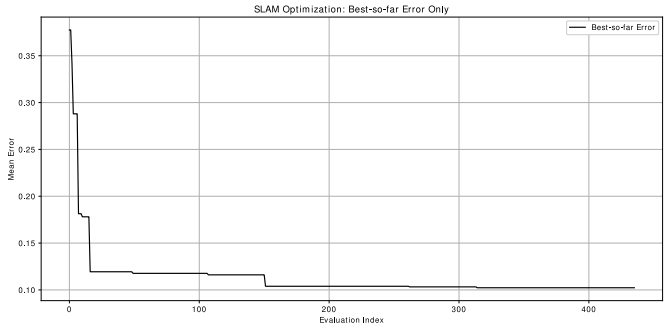

Fig. 3. *Office floor 2*: Best error (in meter) achieved at each iteration

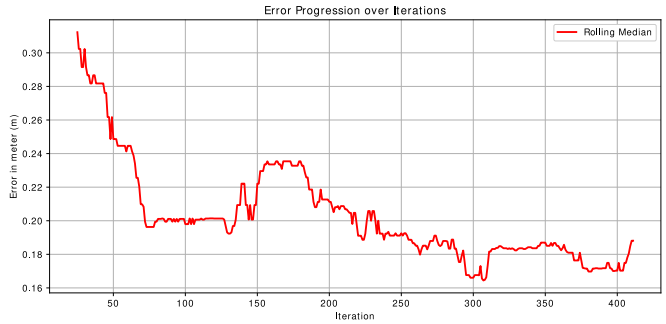

Fig. 4. *Office floor 2*: Rolling median of all recorded errors

Figure 4 shows the moving median (with a window size of 10) applied to all recorded data, providing a smoother view of the optimization process. Since this optimization is performed in a five-dimensional hyperparameter space, significantly more iterations are required to identify a suitable minimum compared to the two-parameter case. Notably, the best-found error drops below 11 cm after around 150 iterations; however, at this point, the moving median remains relatively high. With continued iterations, the overall level of the moving median decreases, ultimately reaching a final best-found error of 10.2 cm, which is consistent with Figure 3. Compared to Figure 2, it is clear that optimizing five parameters enables both a lower median error and a smaller best error, albeit with increased computational effort. This highlights that while more parameters require more iterations to converge, they also allow for improved performance. Once this local minimum is reached, no further improvement is observed in subsequent iterations.

Figure 5 shows the final overlay at the smallest achieved error, where the blue point cloud represents the generated SLAM map and the red point cloud represents the ground truth (same principle for Figure 8 and 11). Notably, in one of the rooms some obstacles remained after filtering, indicating that the pre-processing step was not entirely successful.

Fig. 5. *Office floor 2*: Overlay between the generated SLAM map and its ground truth for a 570 m^2 office floor

C. Optimizing 5 Parameters in a Small Apartment

Following the large-scale office floor experiment, we evaluated the optimization procedure in a smaller environment: a compact apartment with an area of approximately 54 m^2 (*Apartment 1*). For this experiment, we optimized the same five parameters as in the office floor trial (see Subsection IV-B). Figure 6 illustrates the progression of the best error achieved throughout the optimization process.

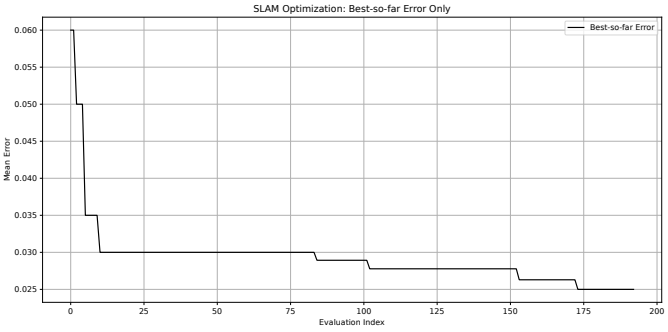

Fig. 6. *Apartment 1*: Best error (in meter) achieved at each iteration

From Figure 6, we observe that the optimization achieves an average error of 2.5 cm between the generated map and the ground truth. Figure 7 shows the rolling median of all recorded errors, providing a smoothed representation of the optimization trajectory.

Fig. 7. *Apartment 1*: Rolling median of all recorded errors

The plot in Figure 7 clearly shows a consistent decrease in error as the optimization progresses. Compared to the larger office environment (see Figure 4), convergence is achieved in fewer iterations. This result is expected, as the smaller map size generally leads to improved SLAM accuracy. Figure 8 shows the resulting map overlay between generated SLAM map and its corresponding ground truth.

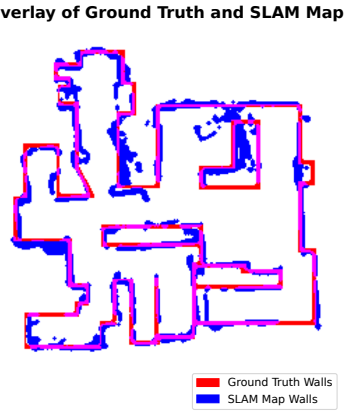

Fig. 8. *Apartment 1*: Overlay of SLAM map and its ground truth of a 54 m^2 apartment

Additionally, we recorded a second apartment with an area of 48 m^2 (*Apartment 2*) and optimized the parameters, achieving an average error of approximately 2.2 cm, as shown in Figure 9.

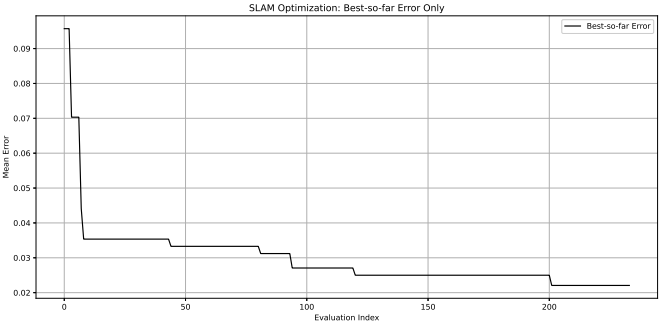

Fig. 9. *Apartment 2*: Best error (in meter) achieved at each iteration

Comparing Figure 7 with Figure 10, we observe that both experiments converged to similar minima, after 180 iterations for *Apartment 1* and 203 iterations for *Apartment 2*. However, when examining the progression of the functions, the optimizer appears to have faced greater difficulty with the apartment in Figure 8 before a stable minimum was established. In contrast, far fewer oscillations are visible for the second apartment shown in Figure 11. This behavior can likely be attributed to the higher structural complexity of the first apartment compared to the second.

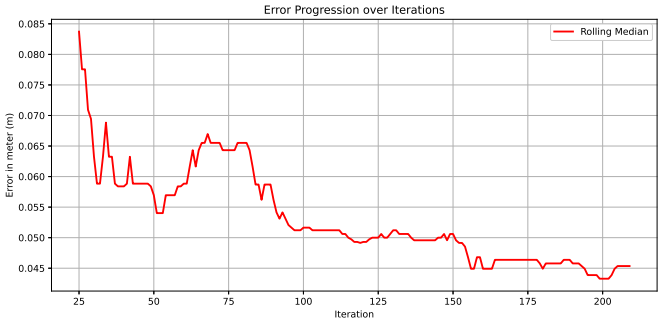

Fig. 10. *Apartment 2*: Rolling median of all recorded errors

Fig. 11. *Apartment 2*: Overlay of SLAM map and its ground truth of a 48 m^2 apartment

TABLE I

OPTIMIZED PARAMETERS FOR DIFFERENT DATASETS

Dataset	MinSc	OptN	MaxR	MinR	SubmapN
Parameter Minimum	0.5	50	1.0	0.01	50
Parameter Maximum	1.0	2000	15.0	0.1	400
Office floor 1	–	–	5.603	0.0459	–
Office floor 2	0.739	2000	4.688	0.0527	323.4
Apartment 1	0.900	2000	4.641	0.0478	400.0
Apartment 2	0.786	2000	5.909	0.0597	400.0
Average	0.808	2000	5.210	0.0515	374.5
Std (%)	8.36%	0.00%	10.68%	10.35%	9.64%

TABLE II

MAPPING ERRORS AND ITERATION COUNTS

Dataset	Trial-and-Error [cm]	Optimization Error [cm]	Optimization Iterations
Office floor 1	15.37 ± 10.0%	11.40 ± 9.0%	103
Office floor 2	14.85 ± 11.3%	10.20 ± 8.9%	315
Apartment 1	3.64 ± 8.9%	2.53 ± 8.5%	180
Apartment 2	4.83 ± 9.2%	2.20 ± 7.8%	203

V. CONCLUSIONS

This work introduced a parameter optimization framework for Google Cartographer, combining Sobol initialization, GPEI-based Bayesian optimization, and a robust alignment pipeline. Across both large-scale (office) and small-scale (apartment) datasets, the optimizer consistently reduced SLAM error, achieving mean errors as low as 2 cm. The evaluation relied on the symmetric Chamfer distance, which proved well suited for this task: the overlay plots clearly demonstrated that smaller Chamfer distances correspond to closer alignment between the generated map and the ground truth. Using the mean Chamfer distance as the primary error measure was particularly effective, since maximum error values are often dominated by outliers, as also evident in the plots. Averaging thus provided a more robust and representative quantification of SLAM map accuracy.

We further showed that the optimized parameter values remained highly consistent across environments when using the same range sensor. Specifically, the dispersion across parameters such as MinSc, OptN, MaxR, MinR, and SubmapN was modest, with standard deviations below 11%. This indicates that once a reliable parameter set is established for a sensor, it can typically be transferred to new environments with only minor adjustments—significantly reducing the need for repeated optimization.

For larger environments, we found that optimizing fewer parameters yields a better trade-off between computational cost and accuracy, while smaller environments benefit from more extensive parameter exploration. Most importantly, we demonstrated that parameters optimized in small, controlled settings can be reliably transferred to larger and structurally different environments when the same range sensor is used. This transferability reduces the need for repeated optimization, highlighting that parameter behavior is largely sensor-driven rather than environment-specific. Taken together, these findings underline the importance of sensor-specific calibration in SLAM tuning and demonstrate that automated optimization can substantially outperform manual trial-and-error approaches.

ACKNOWLEDGMENT

This publication is a result of research conducted at Infineon Technologies AG, whose support and collaboration made this work possible.

REFERENCES

[1] 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)*. IEEE, 2016, pp. 1271–1278.

[2] R. Leitner, E. Steurer, A. Berger, and W. Bell, "Hybrid time-of-flight (tof) based slam: Hybrid fusion for robust and accurate mapping and localization," Infineon Technologies AG, Tech. Rep., 2022. [Online]. Available: https://www.infineon.com/real3

[3] G. C. Authors, "Google cartographer: Tuning methodology," https://google-cartographer-ros.readthedocs.io/en/latest/tuning.html, accessed: 2024-06-18.

[4] ——, "Cartographer ros lua configuration reference documentation," https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html, accessed: 2025-06-18.

[5] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, pp. 2951–2959, 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[6] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[7] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 10 2023. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[8] S. G. Shahbandi, "Map-alignment-2d: 2d map alignment with region decomposition," https://github.com/saeedghsh/Map-Alignment-2D, 2017, gitHub repository. Archived Oct 8, 2024.

[9] S. G. Shahbandi and M. Magnusson, "2d map alignment with region decomposition," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 597–610, 2020. [Online]. Available: https://arxiv.org/abs/1709.00309

[10] W. G. Teame, W. Zhongmin, and Y. Yu, "Optimization of slam gmapping based on simulation," in *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 4. IJERT, 2020, pp. 74–81, iSSN: 2278-0181. [Online]. Available: https://www.ijert.org/research/optimization-of-slam-gmapping-based-on-simulation-IJERTV9IS040107.pdf

[11] N. Nagarajan, H. Zhang, W. Liu, and J. Gu, "Multi-objective optimization of rtab-map parameters using genetic algorithm for indoor 2d slam," *Procedia Computer Science*, vol. 250, pp. 172–181, 2024. [Online]. Available: https://doi.org/10.1016/j.procs.2024.11.022

[12] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Automatic hyper-parameter tuning for black-box lidar odometry," in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 356–13 363. [Online]. Available: https://ieeexplore.ieee.org/document/9561789

[13] K. Trejos, L. Rincón, M. Bolaños, J. Fallas, and L. Marín, "2d slam algorithms characterization, calibration, and comparison considering pose error, map accuracy as well as cpu and memory usage," *Sensors*, vol. 22, no. 18, p. 6903, 2022. [Online]. Available: https://doi.org/10.3390/s22186903

[14] Z. A. Ahmed and S. M. Raafat, "An extensive analysis and fine-tuning of gmapping's initialization parameters," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 3, pp. 126–138, 2023. [Online]. Available: https://doi.org/10.22266/ijies2023.0630.10

[15] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "Botorch: A framework for efficient monte-carlo bayesian optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [Online]. Available: https://arxiv.org/abs/1910.06403

[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, 1975. [Online]. Available: https://dl.acm.org/doi/10.1145/361002.361007

[17] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," *arXiv preprint arXiv:1612.00603*, 2016. [Online]. Available: https://arxiv.org/abs/1612.00603

[18] G. C. Authors, "Google cartographer: Configuration reference," https://google-cartographer.readthedocs.io/en/latest/configuration.html, accessed: 2024-06-19.