WiFi Assisted Loop Closure Detection

Faris Hajdarpasic, Shashank Dammalapati

University of Bonn
s25fhajd@uni-bonn.de
s42damm@uni-bonn.de

Abstract. In indoor environments, LiDAR based SLAM can have potential cons, due to the computational efficiency and featureless parts of environment. One way of approaching this issue is integrating some other types of sensors or available data. In our approach, we are trying to use WiFi data, to reduce computation time and make SLAM independent of geometry of the environment. Each pose of the robot will be assigned one WiFi fingerprint, and based on their similarity loop closures can be determined. Since WiFi data is fluctuating due to the different reasons, simple similarity comparison is not enough and we are trying to implement fingerprint sequence-based approach, which would take neighbouring poses of the current poses for which we are trying to estimate whether they are loop closures. Reliable loop closures, using WiFi data, can be found in very fast and simple manner. Besides having only WiFi SLAM, integration into existing LiDAR SLAM is possible as well, improving the efficiency and preserving the accuracy of that SLAM system.

Keywords: SLAM · Loop Closure · WiFi Fingerprint

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a technique for localizing the robot in an unknown environment, while simultaneously constructing the map of the environment. Usually in outdoor environments, GPS can be used for solving SLAM problem, but, when it comes to indoor environments, GPS cannot be used. Other sensors need to be taken into consideration, in order to solve localization (and mapping) problem.

Two most common sensors used nowadays for this purpose are LiDAR (Light Detection and Ranging) and different setups of cameras. LiDAR usage can have two potential problems, related to environment properties:

- In large environments, LiDAR SLAM can be computationally expensive, because of the many scan matching executions
- In geometrically-degraded environments, accurate loop closure can be hard to estimate, since it can be challenging for scan matching to identify proper matches for loop closure

The goal of this project is to investigate usage of WiFi signals to tackle these problems. Comparing two WiFi signals is much faster than scan matching, and also loop closure detection using two WiFi fingerprints in geometrically-degraded environments should be more accurate than scan matching in the same setup.

In following sections, we will explain our approach for WiFi SLAM, present obtained results and discuss possible further improvements.

2 Overview of graph-based SLAM

There are different approaches to solving SLAM problem. The one that is mostly used nowadays, is graph-based approach. In graph-based SLAM, nodes and edges represent two main components.

Node is defined by the pose of the robot and its observation, at a specific timestamp. In our case, instead of general 3D pose of the robot, we will use 2D position, and as a measurement, we will use WiFi fingerprint.

Edge is connection between two nodes. It is defined as relative transformation. It is also called constraint. There are two types of edges/constraints:

- Consecutive edge - relative transformation between two consecutive nodes, which can be computed using some source of odometry.
- Non-consecutive edge - relative transformation between two non-consecutive nodes, which can be computed using some other sensor, whose observations can be used to compute similarity between two nodes (e.g. LiDAR, WiFi signals, etc). It is also known as loop-closure.

Loop closure means that robot has seen again certain part of the environment, and therefore we can establish a relationship (i.e. relative transformation) between two different nodes, from which robot saw same (or overlapped) area.

Since odometry is prone to drifting, role of loop closure is to reduce that drift, and improve trajectory. This is achieved by having many nodes and edges between those nodes, and optimizing based on that, which will be explained in one of the following chapters. On the image below, the red represents trajectory (therefore the map) without loop closures and optimization, and blue represents trajectory with loop closures and after optimization.

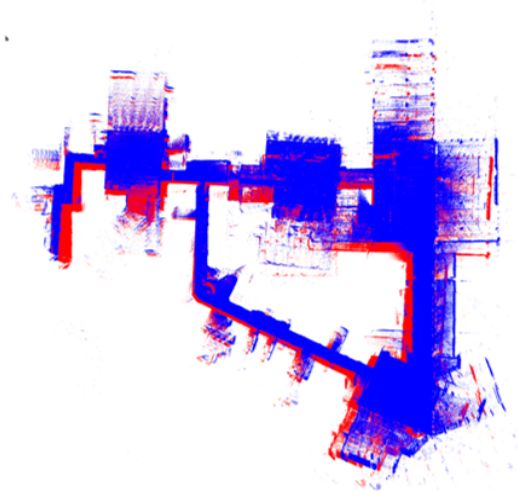

Fig. 1: Map with (blue) and without (red) loop closures

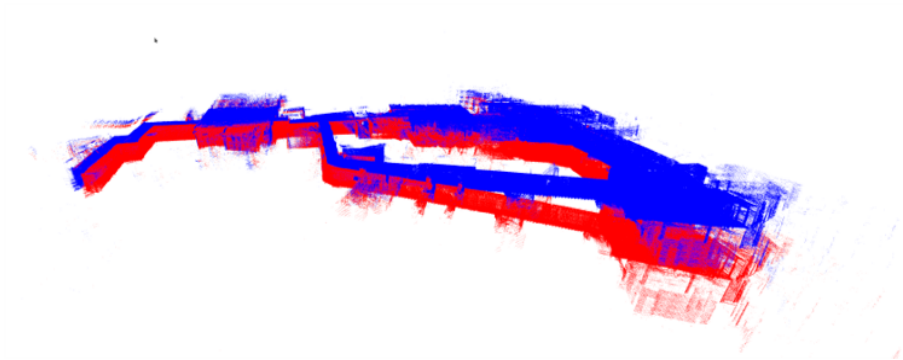

Fig. 2: Map with (blue) and without (red) loop closures

For generating these trajectories, which can be later used in detecting additional loop closures based on WiFi signals, CLINS[1] is used. CLINS stands for Continuous-Time Trajectory Estimation for LiDAR-Inertial System. It is open-source odometry algorithm, which needs LiDAR data and IMU as input. In the images above, we drove a robot on the ground floor of the Institute of Informatics building, University of Bonn.

3 WiFi fingerprint similarity

In previous chapter, we mentioned that, as observation, we use WiFi fingerprint. which is defined for each node. WiFi fingerprint is a list of all RSS values from all visible APs for each node. Each AP is defined by its own MAC address, and there can be multiple APs in one physical router.

$$\mathbf{f}_t = (f_1, f_2, ..., f_n) \tag{1}$$

After gathering WiFi data, we have one WiFi fingerprint for each node. In order to detect loop closures, we need to know node correspondences that have similar WiFi fingerprints. WiFi signals are not perfect and cannot be considered reliable by themselves, since they can fluctuate even though the robot does not move, we cannot claim that loop closure has been detected, only based on the two nodes fingerprint similarity. So we will use fingerprint similarity as a condition, to select loop closure *candidates*, and in the next chapter, the methodology of filtering those candidates and selecting only (hopefully) true loop closures will be explained.

So, given two fingerprints $\mathbf{f}_i$ and $\mathbf{f}_j$, the similarity between those two can be computed using the following formula[2]:

$$sim(f_i, f_j) = \frac{H}{L_i + L_j - H} \cdot \frac{1}{H} \cdot \prod_{n=1}^{H} \exp\left(-\frac{(f_{i,n} - f_{j,n})^2}{2\sigma^2}\right) \tag{2}$$

where, L_i and L_j are total number of visible APs at respective timestamps, and H is number of common APs visible from both timestamps.

This similarity function, has one issue. When there is a lot of commonly visible APs (i.e. H is high number), product will become very small (near zero) since many numbers that are lower than 1.0 will be multiplied together. This can be seen below, where there is a lot of common APs, and therefore similarity score is low.

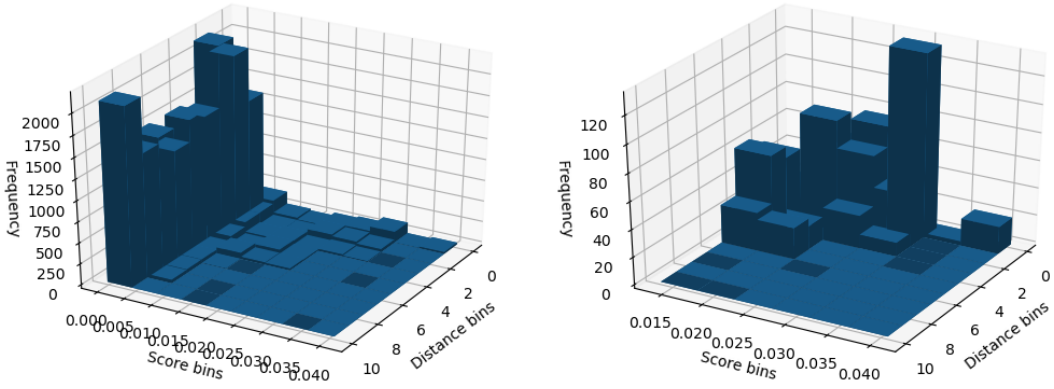

Fig. 3: Similarity score function issue

To avoid this problem, and make the similarity score more robust and less dependent to the number of H (i.e. common APs), we changed a bit previous function, and now it is:

$$sim(f_i, f_j) = \frac{H}{L_i + L_j - H} \cdot \prod_{n=1}^{H} \exp\left(-\frac{(f_{i,n} - f_{j,n})^2}{2\sigma^2}\right)^{\frac{1}{H}} \tag{3}$$

4 Loop Closure detection using WiFi

Due to the unpredictability of WiFi signal propagation, such as non-line of sight and multipath effect, same poses, at different timestamps can have different fingerprints. Therefore we cannot simply take two nodes as a loop closure, only based on their similarity score. Similarity score can be used as as condition to find potential loop closures i.e. loop closure candidates. For final decision, different approach is needed. It can be called *sequence-based* approach. Explanation of this approach is as follows.

Let's suppose that we have two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, together with their fingerprints as observations $\mathbf{f}_i$ and $\mathbf{f}_j$. Let's assume that their similarity score is high enough, and now they are loop closure candidates. To decide wether they are loop closures, and not only candidates, we have to apply sequence-based approach. Let's take window or width w centered around node i. Let's do the same for the node j. Window size w is a hyperparameter. Now we have list of nodes around node i and list of nodes around node j. Using this, we can find best relative transformation between these two tracks, by minimizing the following

cost function[2]:

$$\mathbf{T}^* = \frac{1}{w} \cdot \sum_{w/2}^{w/2} dist(T(x_i^{-1} \cdot x_{i+\tau}), x_j^{-1} \cdot x_{j*}) \tag{4}$$

where *dist* represents Euclidean distance, and x_{j*} is best possible correspondence in track j for a node in track i. x_{j*} is calculated as follows[2]:

$$x_{j*} = \frac{1}{\sum_{l=1}^{k} sim(f_{i+\tau}, f_{\pi(l)})} \cdot \sum_{l=1}^{k} sim(f_{i+\tau}, f_{\pi(l)}) \cdot x_{\pi(l)} \tag{5}$$

This formula is saying that, for some node from track i, we calculate fingerprint similarities with all nodes in track j. Then, we take k nodes from track j with highest similarity values, and then we can estimate x_{j*} by using above expression. This can be seen in the image below. For each node in track i we have its correspondence in track j.

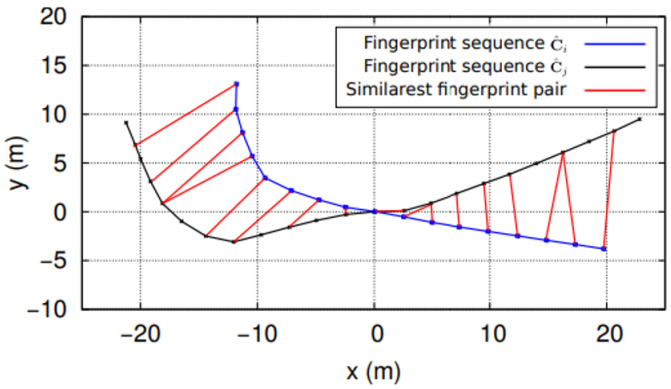

Fig. 4: Two fingerprint sequences, and correspondences between nodes in those two sequences

So now, for each node in track i, we have its correspondence in track j. Supposing these are correct correspondences, we can apply ICP with known correspondences (which is solved via SVD) and obtain direct and optimal solution (i.e. relative transformation) between those two tracks.

One final step is to calculate MSE (mean squared error) between those correspondences applying this calculated transformation, and if error is lower than certain threshold, we can take nodes i and j as a loop closures. Summarized WiFi loop closure detection is shown in pseudocode below:

```
for i ← 1 to T do
    // Check accumulated distance and
       fingerprint similarity
    for j ← 1 to i with acc(x_i, x_j) ≥ 50 and
     sim(f_i, f_j) ≥ 0.3 do
        ▷ Compute the relative pose T* between x_i and
          x_j according to Equation 3
        if Average distance computed for T* is smaller
          than 3 meters then
            ▷ Add < x_i, x_j > as loop closure
        end
    end
end
```

Fig. 5: WiFi Loop Closure Detection

5 Optimization

Previously we mentioned what are nodes and edges, as well as types of edges there are (consecutive and non-consecutive). Each of those edges represent relative transformation. Since now we have all those edges and nodes, graph is constructed. This is also known as SLAM front-end. After obtaining this, we can optimizing graph, which is known as SLAM back-end. Goal of SLAM optimization is to find node configuration that minimize the error introduced by the constraints.

Let's suppose we have two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. We will have one constraint between those two nodes according to the current graph configuration and there will be one constraint based on observation from two nodes. Following image depict explained situation:

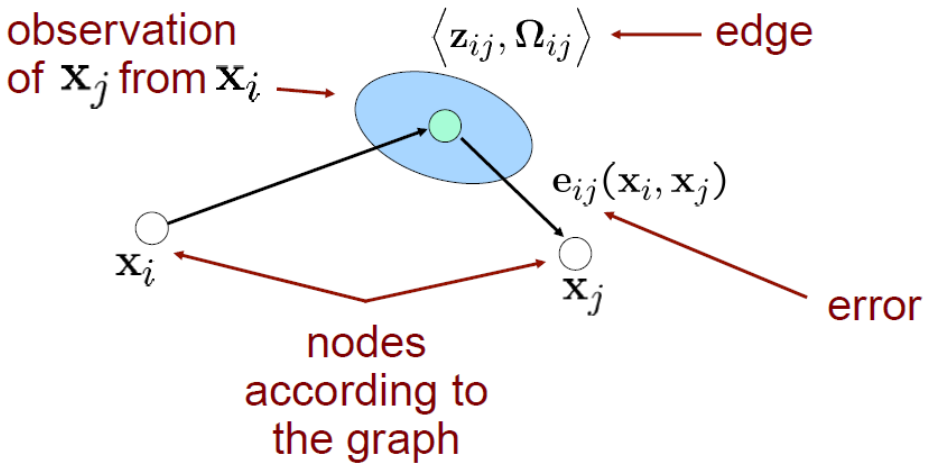

Fig. 6: Graph optimization illustration [3]

So the goal of SLAM optimization, is to optimize following cost function:

$$\mathbf{x}^* = argmin \sum_{ij} e_{ij}^T \cdot \Omega_{ij} \cdot e_{ij} \tag{6}$$

where $\mathbf{x}$ denotes all nodes in the graph, and e is error between what graph tells us and what observations tell us, and Ω is information matrix, which encodes uncertainity of the edge.

Error function (of the whole state vector) is defined as:

$$e_{ij}(\mathbf{x}) = t2v(\underbrace{\mathbf{Z}_{ij}^{-1}}_{\text{Measurement}} \cdot \underbrace{(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j)}_{\text{Current graph configuration}}) \tag{7}$$

To minimize this cost function, the Gauss-Newton method can be used. Steps needed for implementing Gauss-Newton method are:

1. Define the error function
2. Linearize the error function
3. Compute its derivative
4. Set the derivative to zero
5. Solve the linear system
6. Iterate until convergence

Due to the lack of space, all the math will not be written here, but we encourage you to look at the referenced literature[3], where all these steps are explained in detail.

6 Data Collection

6.1 Tools Used for WiFi Fingerprinting

1. **iwlist** - A straightforward command-line utility that provides a static snapshot of available Wi-Fi networks at the time of the scan but lacks real-time updates.
2. **airodump-ng** - A powerful Wi-Fi monitoring and packet capturing tool that continuously scans and updates information about Wi-Fi networks, making it ideal for real-time monitoring.
3. **pywifi** - A Python library for Wi-Fi network operations, which, while useful, does not inherently support continuous monitoring and real-time updates.
4. **nmcli** - A command-line tool for managing NetworkManager in Linux, capable of gathering information about Wi-Fi networks but with a relatively slow update rate of approximately 1/10 Hz.

In our Wi-Fi fingerprinting data collection process, we employed several distinct tools, each with its own set of capabilities and limitations. Firstly, iwlist served as a straightforward utility, providing a static snapshot of available Wi-Fi networks but falling short in terms of real-time updates. On the other end of the spectrum, airodump-ng proved to be a robust choice for real-time monitoring, continuously scanning and updating data about nearby Wi-Fi networks and their associated devices. Meanwhile, pywifi, a Python library, offered flexibility but lacked native support for continuous monitoring, requiring manual polling. Lastly, nmcli, the NetworkManager Command-Line Interface, while capable of gathering Wi-Fi information, operated at a relatively sluggish update rate of about 1/10 Hz. The selection of the appropriate tool hinges on project-specific requirements, with airodump-ng being the preferred choice for real-time monitoring, while iwlist, pywifi, and nmcli can find utility in simpler scenarios or where a single snapshot suffices.

The tools used for Wi-Fi fingerprinting in this project offer a range of options to cater to different monitoring needs. For applications demanding real-time updates and continuous monitoring of Wi-Fi networks, airodump-ng emerges as the most suitable choice, providing comprehensive and up-to-the-moment data. In contrast, iwlist, pywifi, and nmcli may find utility in scenarios where real-time monitoring is not imperative, and a static snapshot of network information suffices. The key takeaway is to select the tool that aligns with the project's objectives and monitoring requirements, ensuring the most effective and efficient data collection and analysis for the given application.

6.2 Robot Platform

Our data collection efforts were effectively facilitated by the Nimbro_home platform. This comprehensive platform consists of key components, including the Tiago Robot, which provided mobility and navigation capabilities essential for

data collection. Additionally, we utilized an Ouster Lidar - OS Dome 64 Channel with an integrated IMU to capture high-resolution point cloud data and inertial measurements, crucial for our Simultaneous Localization and Mapping (SLAM) tasks using CLINS. For the purpose of assessing the impact of wifi-based loop closures, we integrated a USB Wifi Receiver into our platform, enabling the collection of wifi fingerprinting data at every timestamp during our experiments. Furthermore, a Brio Fisheye Camera was employed to capture fisheye imagery, serving as a valuable baseline for future performance comparisons.

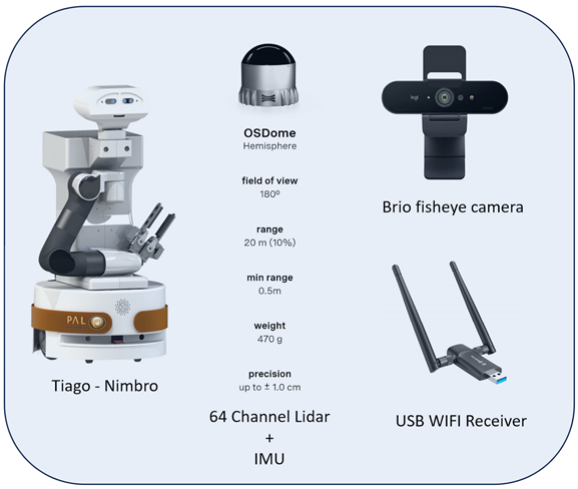

Fig. 7: Data Collection

6.3 Robot Environment

Our data collection was conducted within the confines of our project lab, located at Friedrich-Hirzebruch-Allee 8, 53115 Bonn. The lab's layout and characteristics naturally encouraged the formation of diverse loop closures of varying sizes, making it an optimal environment for conducting SLAM experiments. This choice of location allowed us to create a dataset that encompasses a wide range of real-world scenarios and challenges, enabling comprehensive evaluations and validations of our SLAM algorithms and methodologies.

Fig. 8: Robot environment

7 Results

7.1 WiFi Signal Quality and Distance Estimation

One crucial aspect of our data analysis involves assessing Wifi Signal Quality and its impact on distance estimation. We observed that Wifi signal strength does not follow a consistent pattern, particularly in scenarios where multipath interference and line-of-sight (LOS) conditions vary. To address this issue, we explored the Free Space Path Loss (FSPL) model for distance estimation under clear LOS conditions. The FSPL model is expressed as follows:

$$FSPL(dB) = 20\log 10(d) + 20\log 10(f) + K \tag{8}$$

where:

- d - distance
- f - frequency
- K - constant dependent on the units used for d and f.

For our specific units of kilometers for distance (d) and megahertz for frequency (f), the formula becomes:

$$FSPL(dB) = 20\log 10(d) + 20\log 10(f) + 32.44 \tag{9}$$

However, it's important to note that this formula may not always provide accurate distance estimations due to several factors. First, there's a significant difference in frequency between low-frequency Wifi fingerprinting (1 Hz) and Lidar frequency (10 Hz), leading to multiple poses associated with the same signal strength. Additionally, in cases involving non-LOS conditions and multipath interference, we observed unexpected results.

To visualize and understand these discrepancies, we utilized Open3D to draw lines from the poses in the trajectory to the WIFI routers. The visualizations revealed a consistent pattern wherein the estimated distances calculated using the FSPL model often underestimated the actual distances. This discrepancy is particularly evident in scenarios where multipath interference and non-LOS conditions are prevalent.

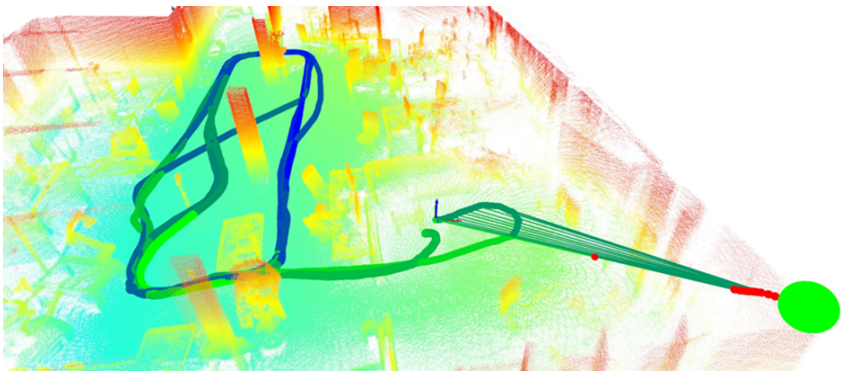

Fig. 9: Distance estimates for time stamps 0-10 seconds

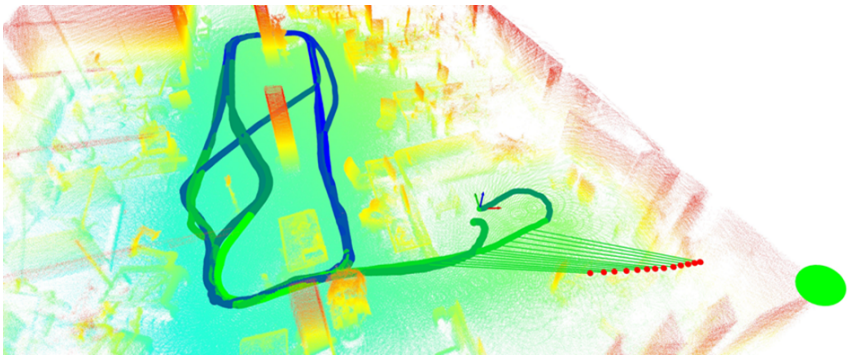

Fig. 10: Distance estimates for time stamps 12-20 seconds

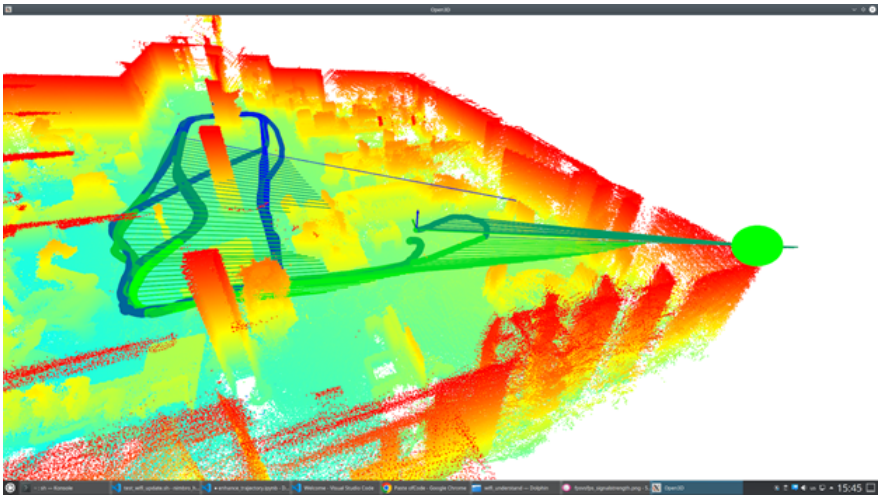

Fig. 11: Distance estimates for time stamps 0-60 seconds

These findings underscore the need for caution when relying solely on Wifi signal strength for distance estimation in complex and dynamic environments. While the FSPL model can provide reasonable estimations under LOS conditions, its accuracy diminishes in scenarios with multipath interference and non-line-of-sight conditions.

7.2 Assessing Wifi Fingerprint Stability for Loop Closure Detection

To leverage Wifi fingerprints for loop closure detection, we aimed to identify consistent patterns in signal strength across different poses with a minimum distance gap. Our investigations involved creating a looping trajectory within our lab, repeated four times. This allowed us to observe signal strength changes as we revisited locations. Notably, we found that while signal strength displayed periodic behavior, the specific numerical values did not precisely match previous measurements.

This variability in signal strength values can be attributed to dynamic environmental factors, including scene changes and variations in the robot's orientation, affecting the wifi receiver's orientation. Although periodicity in signal strength showed promise for loop closure detection, we recognized the dynamic nature of wifi fingerprinting, emphasizing the need to account for these variations in our loop closure algorithms. The exponential part of the similarity calculation between two fingerprints takes care of these slight changes in the signal strength for loop closure candidates.

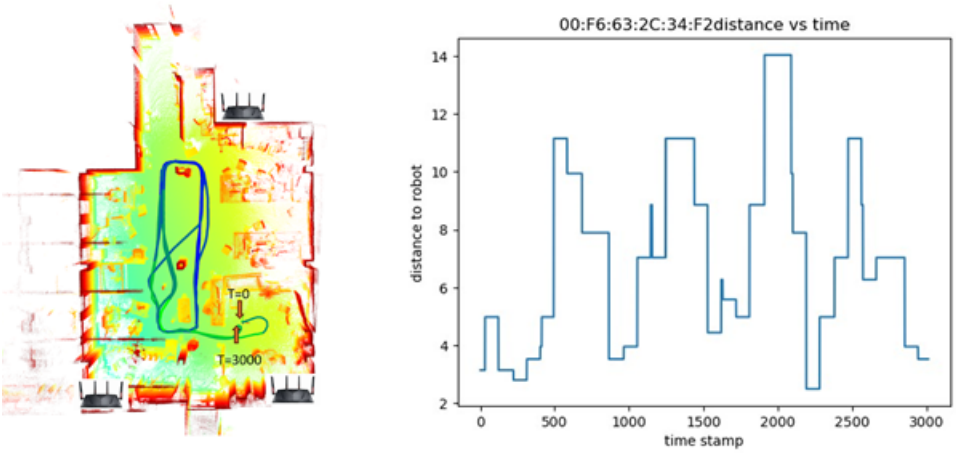

Fig. 12: Distance for one MAC address

7.3 Evaluating Similarity Scores for Loop Closure Detection

For loop closure detection using Wifi fingerprints, we conducted a study comparing similarity scores between a specific fingerprint and all other fingerprints collected along the same trajectory, which we discussed in the previous section regarding fingerprint stability. Our analysis revealed a recurring pattern in the similarity scores, aligning with the periodic behavior observed in signal strength. However, the crucial observation was that these similarity scores did not consistently reach the desired high value (ideally 1).

In our experimentation with different sequences and varying base fingerprints, we encountered substantial variations in similarity score values. Unlike Lidar-based SLAM, where a fixed threshold often suffices for loop closure candidate selection, Wifi-based SLAM presented a challenge. Identifying a single universal threshold that reliably caps potential loop closure candidates across different sequences proved elusive. The dynamic nature of Wifi fingerprints, influenced by changing environmental conditions and robot orientations made the results hard to interpret or useful.

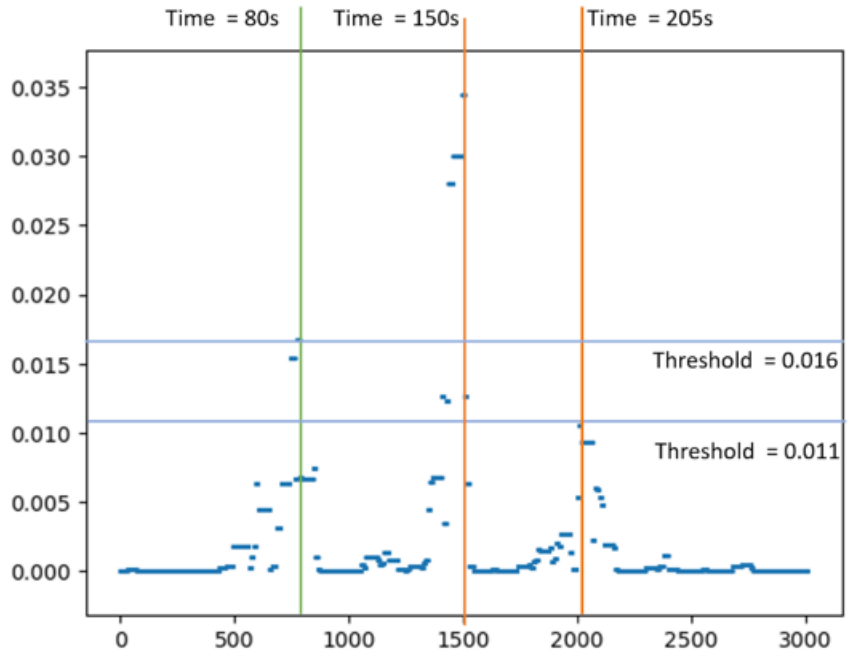

Fig. 13: Similarity score between timestamp 1500 and all other timestamps

7.4 Controlled Experiments for Wifi-Based Loop Closure Detection

To rigorously assess the efficacy of our similarity score and the entire Wifi-based loop closure detection methodology, we recognized the need for controlled experiments. The real-world data collected on the Nimbro platform yielded suboptimal results, prompting us to create simulated data with deliberately introduced noise characteristics.

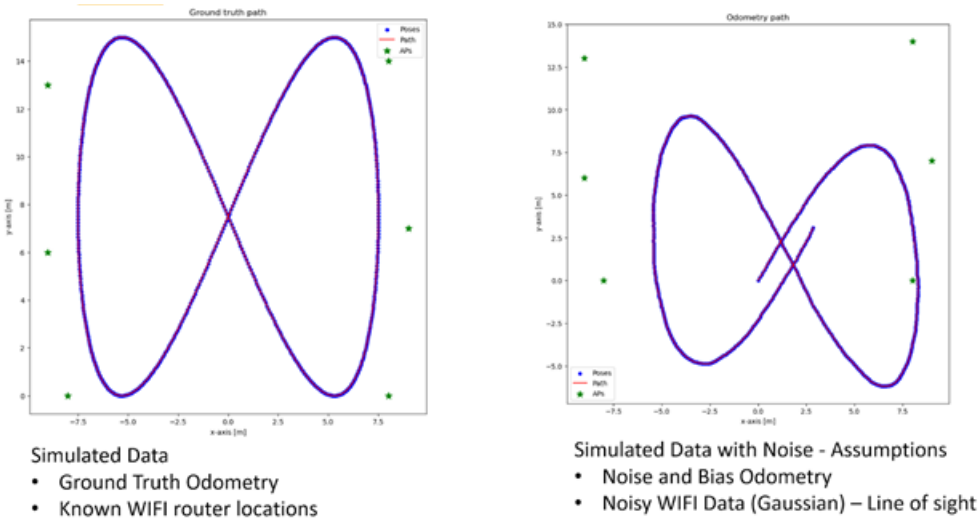

Fig. 14: Simulated data

In our simulated data, we introduced noise to both the trajectory data and the recorded wifi fingerprints. This allowed us to test the robustness of our loop closure detection approach under controlled conditions. We leveraged the methodology described in Section 4 of this report, which involves two crucial stages:

– **Identifying Possible Loop Closure Candidates**
In this stage, our approach identifies potential loop closure candidates based on similarity scores. These candidates serve as initial candidates for further scrutiny. For the simulated data, we plotted data in a manner similar to "Evaluating Similarity Scores for Loop Closure Detection" and observed ideal results. This analysis enabled us to determine a threshold value that consistently works across different scenarios. In perfect cases, we even achieved similarity scores of 1.

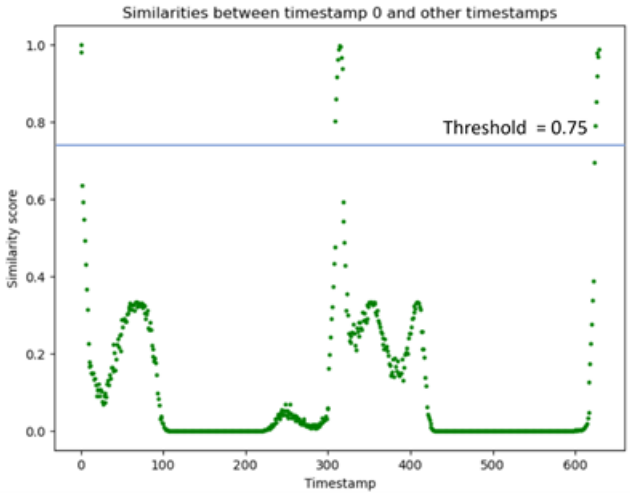

Fig. 15: Similarities between timestamp 0 and other timestamps

– **Refining Valid Loop Closure Candidates**
The subsequent step involves narrowing down the candidates to valid loop closures and determining their relative transformations. We achieved this by applying a Singular Value Decomposition (SVD) based Iterative Closest Point (ICP) method.

The following plot illustrates the calculated loop closure edges, showcasing the results of our controlled experiments with simulated data. These experiments allowed us to assess the robustness and reliability of our Wifi-based loop closure detection method under controlled conditions, ensuring its effectiveness for real-world Wifi-based SLAM applications.

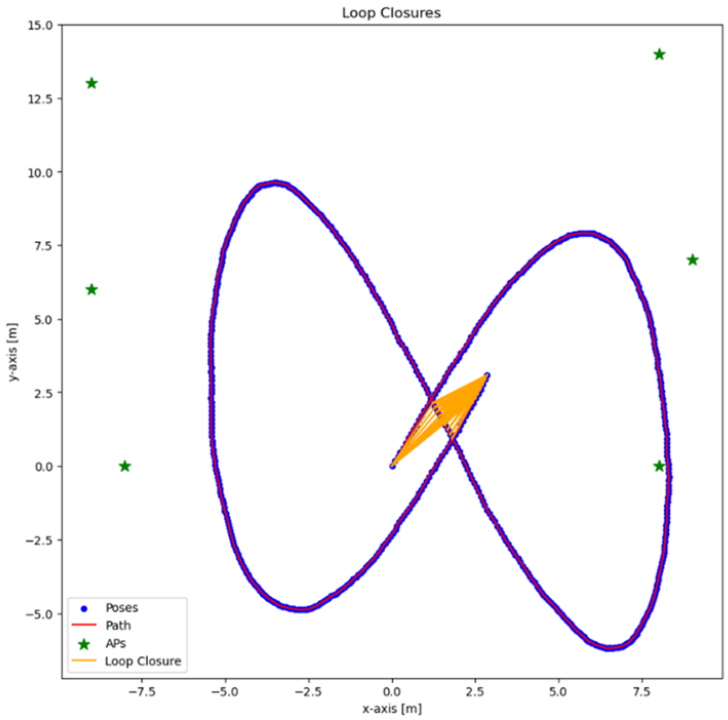

Fig. 16: WiFi Loop Closures on Simulated Data

Furthermore, through experimentation, we varied the parameter of the window size utilized in estimating loop closure validity and transformation. Remarkably, we found that a window size of 20 consistently yielded the lowest average Root Mean Square (RMS) error.

Window Size	Average Error (GT – Estimated transformation)	N(Loop Closures)
6	2.25m	61
12	1.65m	51
20	1.13m	84
30	1.14m	61

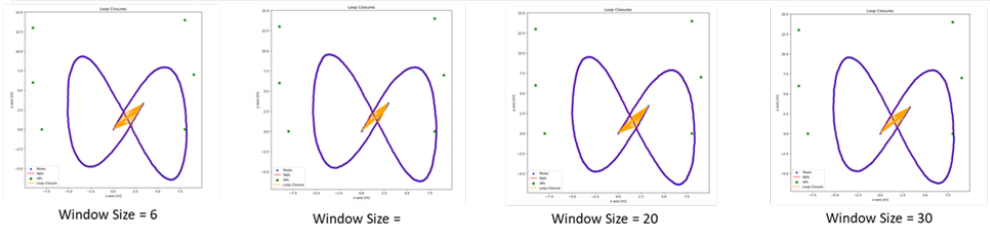

Fig. 17: WiFi Loop Closures and different window size

7.5 Testing on Real Data

In the conclusive phase of our Wifi-Based Loop Closure Detection method evaluation, we transitioned from simulated data to real-world assessments. This real-world testing involved the creation of a new dataset, covering the entire ground floor of the Friedrich-Hirzebruch-Allee 8 building, with a carefully designed trajectory that necessitated successful loop closures for a return to the initial point. We incorporated the initial trajectory estimate from the CLINS pipeline and employed it in conjunction with wifi fingerprints for every pose. This integration allowed us to systematically identify possible loop closure candidates. Our results, visualized through plots showcasing these candidates, affirm the practical effectiveness and reliability of our method in real-world scenarios. While we did not succeed in establishing a systematic method for determining the correct threshold for similarity, in the final case, we resorted to using an arbitrary number as a threshold as shown in the figure. This evaluation represents a significant milestone in Wifi-based SLAM, emphasizing the potential for further refinement and application.

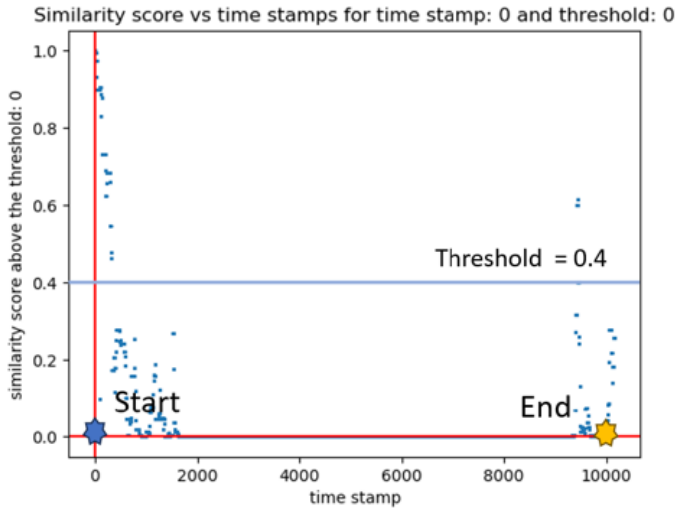

Fig. 18: Similarity score between timestamp 0 and all other timestamps

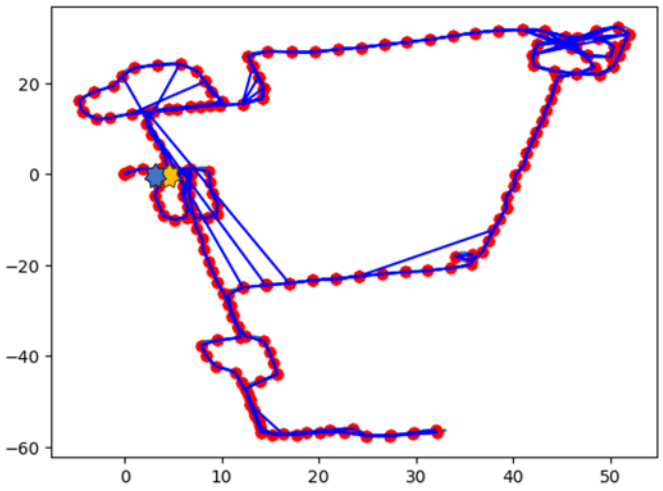

Fig. 19: Loop closure candidates detection using WiFi fingerprint similarity

8 Conclusion and future work

In summary, our exploration of Wifi-based SLAM has yielded valuable insights into its potential and challenges. While we haven't proposed specific solutions,

our research represents progress in understanding Wifi-based SLAM. Our experiments in controlled and real-world settings demonstrate the promise of Wifi-based loop closures for expediting pose graph optimization. Notably, we modified the similarity score calculation method to enhance results.

As we look ahead to future work, our focus remains on adaptable threshold determination mechanisms and further exploration of pose graph optimization and its integration into SLAM systems, such as CLINS.

References

1. J. Lv, K. Hu, J. Xu, Y. Liu, X. Ma and X. Zuo, "CLINS: Continuous-Time Trajectory Estimation for LiDAR-Inertial System," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 6657-6663, doi: 10.1109/IROS51168.2021.9636676.

2. K. Ismail et al., "Efficient WiFi LiDAR SLAM for Autonomous Robots in Large Environments," 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 2022, pp. 1132-1137, doi: 10.1109/CASE49997.2022.9926530.

3. G. Grisetti, R. Kümmerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM," in IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31-43, winter 2010, doi: 10.1109/MITS.2010.939925.