Wi-Fi based range-only constraint integration in RTAB-Map

Mathieu Nass Bsc.
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, Netherlands
m.nass@student.utwente.nl

***Abstract*— Loop-closure detection and the disambiguation of similar locations are difficult problems for simultaneous localisation and mapping (SLAM) algorithms. Wi-Fi is able to provide unique identifiers for locations, i.e. perceivable MAC-addresses. In this paper we investigate the possibility of exploiting received signal strength (RSS) in an existing visual graph-based SLAM algorithm, namely Real Time Apperance Based-Mapping (RTAB-Map). We formulate a range-only constraint and generate these constraints from perceivable Wi-Fi access-points (APs) during the SLAM process. We make the translation from RSS to distance by fitting a signal propagation model to measurement data. Finally, we evaluate performance by computing the absolute trajectory error (ATE) and relative pose error (RPE). These errors are computed by comparing with a measured ground-truth. We find that, in a controlled synthetic environment, the range-only constraints have great correctional capabilities for noisy odometry as well as great loop-closure potential. In the real-world experiments we observe limited performance due to the limitations of the used signal propagation model. In this case our addition to RTAB-map is beneficial in some of the tested robot configurations. However, the synthetic experiments show that performance increases significantly when better range measurements are available, in real-world from i.e. utra-wide-band sensors.**

***Index Terms*—SLAM, Robotics, Graph-based SLAM, Constraint types, RTABMAP, g2o, Wi-Fi**

I. Introduction

Many out of the box implementations exist for accurate SLAM in robotics. Implementations have become quite good at accurate loop-closure detection and the disambiguation of similar locations. We want to test a possible, very cheap addition to the pool of available sensors, namely Wi-Fi RSS. Wi-Fi RSS is a measurement that can be conducted on almost any device with access to the internet over a wireless connection. Signal strength, as any electromagnetic signal, generally falls off as distance from the source increases. Thus, information regarding the relative location can be extracted from this measurement. In order for RSS to be incorporated in a graph based SLAM back-end two things are required: a signal propagation model describing the relation between distance and RSS, and a range-only constraint for integration in the graph-based back-end. In this research we define such a constraint and fit a signal propagation model to incorporate these constraints in an existing visual graph-based SLAM algorithm (RTAB-Map). We test this alteration to RTAB-Map in both the areas of loop-closure assistance and odometry fault correction. As Wi-Fi RSS is not the most accurate of measurements, we are expecting a possible increase in robustness rather than in maximum achievable accuracy. The methodology of our research is based around this expectation. We conduct experiments using a synthetic dataset and deploy the proposed approach on a robot platform in a real-world setting.

In this paper we start by placing our research in context of existing research in section II. We provide research questions and some background information in III and IV respectively. We continue by introducing our methodology in section V. This methodology requires certain experiments to be conducted, these are discussed in section VI. Finally, we present our results, section VII, and conclusions in section VIII.

Thanks to DEMCON for making this research possible

II. Related research

As mentioned, many out of the box solutions for certain sensor setups and many approaches to the SLAM problem exist. Some of the latest additions to this ever growing list of implementations and libraries move towards deep-learning and graph-less approaches. ElasticFusion [1], for example, gives dense representation of surroundings without the necessity of a pose-graph by using a deformation graph generated from sub-sampling surfels. Furthermore, the authors augmented their approach with light source estimation [2] using diffuse colour surface reconstruction, ray-tracing and hough-like voting. Light source estimation could be useful for more accurately describing a lighting invariant virtual scene. Similar to this source estimation technique, acoustic SLAM (from [3]) locates audio sources with a particle filter given an estimated direction of arrival. The ray tracing voting of the last paper and particle filtering in a certain direction have a very similar result. Coming back to virtual mapping of light, when localisation is assumed fixed NerF [4] is an application for photo-realistic mapping. The authors achieve very high levels of accuracy when synthesising new views. However, this accuracy and photo-realism comes at the cost of training a neural network. This unfortunately means that it is, as of yet, not viable for SLAM purposes.

A. Graph-based SLAM and loop-closure detection

However, still many very robust and considered industry standard implementations rely on the use of graphs in contrast with the more experimental techniques described above. When adding constraints to this graph SLAM systems are generally divided in two sections: Local SLAM and Loop-closure.

Local SLAM is for generating odometry and thus for finding pose transforms between consecutive poses using the sensors available on the robot. This can be accomplished by for example measuring wheel odometry, using Iterative Closest Point (ICP) of pointclouds, bundle-adjustment or extracting depth information from cameras. Local SLAM is not as active a research environment as doing accurate loop-closure detection as this is considered more important. This because loop-closure detection limits the amount of accumulated drift that slowly builds up from the local mapping error by correcting the trajectory on a global scale. A naive method for loop-closure detection is comparing every measurement to all other measurements. Similarity between one measurement and another may indicate that a loop has been closed and the robot is revisiting a position he has been before. However, this method has roughly complexity $O(n^2)$ with the number of vertices in a graph. More specialised approaches include clustering of key frames, some different approaches are described below. Matching local frames and small maps limits the amount of individual matches that have to be conducted, [5]. Furthermore, keyframes/measurements can also be clustered based of the amount and type of visual words from a camera. A cheap bag-of-words similarity check can be performed before expensive scan matching is attempted [6] [7]. This approach ensures that keyframes that are very different in features are not attempted at matching and/or bundle-adjustment. As matching is, in most cases, an expensive operation one wants to minimise the matching attempts in order to keep an algorithm fast.

One can also cluster key-frames based of other sensors, e.g. Wi-Fi signals strength. In [8] key frames are clustered based on Wi-Fi signals and in [9] cosine similarities of Wi-Fi fingerprints are measured and thresholding is applied. Clustering is conducted by both, where this addition functions as extra layer for selection of visual keyframes. We notice, like many others, that also some information regarding position, e.g. a distance from the source, can be extracted from Wi-Fi data and other range-only sensors. When only localising, [10] use graph-based SLAM with known landmark locations for Wi-Fi SLAM. This is similar to [11], in which Kalman-filtering is applied to localisation with the inclusion of landmark identification from Wi-Fi sensing.

B. Range-only sensing

Extensive research on localisation using ultrawide band beacons has been conducted [12] [13] [14]. In these consecutive papers the authors define a smooth trajectory constraint (which is valid assumption for UAVs) and known ultra wide band (UWB) beacon locations with 2-way time of flight RSS for localising in with graph-based SLAM back-end. Furthermore, attempts to conduct SLAM based on Wi-Fi received signal strength indicator alone have been conducted [15]. Local continuity of received signal strength is assumed and constraints are added based on the weighted mean of the surrounding measurements. Also [16] [17] [18] attempt localisation with Wi-Fi RSS. All of these approaches base of the same signal propagation model which we use in this paper as well (see equation 11). These methods extend this model with knowledge based on the environment to compensate for multicasting and line-of-sight based issues. This is information we do (a priori) not have as we are trying to conduct SLAM, not just localisation on a known map.

C. Wi-Fi integration

We see an opportunity in combining Wi-Fi clustering of keyframes and localisation information from range-only sensing, i.e. the fusion of the aforementioned research from [15] and [8]. Range-only constraints can be generated based of perceivable APs and RSS, this may move locations with similar Wi-Fi fingerprints closer together. These distance constraints can be found by characterising the Wi-Fi sensor with measurements, similar to [11]. Because some SLAM systems, e.g. RTAB-map, allow for limiting key frame selection in euclidean and graph distance, moving locations with similar Wi-Fi surroundings can be advantageous. Furthermore, RTAB-map is able to optimise with the G2O graph optimiser [19] which does not include a range-only constraint but does allow for custom constraints. All of previous alterations may have the same implications on loop-closure and proximity detection as the implementation from [8] without the computational overhead of linking each frame to a Wi-Fi fingerprint.

D. Accuracy and ground-truth

The aforementioned SLAM methods can be evaluated by comparing the output map with a ground-truth map. The comparison is rather trivial using ICP [1]. Due to the "driving around"-nature of SLAM experiments it may be hard to measure a ground-truth map as the robot goes around multiple corners and through corridors. It should be noted that simulation experiments do not suffer from this problem. In [20] a method is described how a ground-truth can be obtained using scan matching by hand and Monte-Carlo localisation: this may produce a very accurate trajectory and map. Furthermore, when the sensors that are required to produce a highly accurate map are not available on the robot platform, the trajectory can be based of external sensors. For example, motion capture may be used [21]. The comparison method often used for trajectories is the absolute trajectory error. Another proposed metric is the relative pose estimate error as described and used in [21], [22] and [23]. The authors argue that the relative pose error is a more accurate measurement for performance than the absolute trajectory error. This because small deviations in trajectory may cause sub-map misalignment, this in turn causes a very big increase to the total measured absolute trajectory error. A very large error can be observed with a small misalignment as the cause. This is why both errors should be considered when evaluating SLAM performance.

III. Research questions

As discussed in section I and II, we are interested in integrating Wi-Fi in an existing SLAM framework. In order to narrow the scope and create a clear approach, next follow some concrete research questions:

1) What is the state of the art for SLAM and what sensor layouts do these techniques generally use?
2) What spatial information can be extracted from the RSS of Wi-Fi signals and how can this be exploited in a graph-based SLAM back-end?
   a) What is the relation between Wi-Fi RSS and distance from the source?
   b) How does adding Wi-Fi-related constraints to a graph change the computational load or accuracy for a given graph SLAM system?
   c) What levels of accuracy can be achieved in estimating the positions of the robot using RSS sensing and the proposed SLAM integration of those signals?
3) How do the accuracy of localisation of synthetic and real world environment change with respect to the initialisation methods of the graph optimisation process?
4) How should a ground-truth be defined for a indoor SLAM system with Wi-Fi sensing capabilities?

IV. Background

To provide context for the Method (section V) a short introduction to graph SLAM is provided below.

A. Graph-based SLAM

This section is based of the work in [24].

Consider a position in a map to be a vertex in a 2D/3D-graph, each edge in this graph describes the required translation and/or rotation from one vertex to another. The difference between the current position of the connected nodes, and the required transformation as described by an edge can be formulated as an error. Furthermore, these edges have an information matrix associated to them which functions as a weight for the respective error functions. Static objects that are observed by sensors can also be added to the graph with an edge describing the observation from the current robot position. After the construction of the graph, optimisation of the poses is required. This to minimize the sum of weighted errors as generated by the constraints. This optimisation problem can be described as a maximum likelihood problem in matrix F:

$$\boldsymbol{x}^* = \arg\min_x(\boldsymbol{F}(\boldsymbol{x})) \tag{1}$$

Where $\boldsymbol{x}^*$ are the positions of the nodes that minimize the localisation error and F is the negative log likelyhood of all observations (i.e. constraints):

$$\boldsymbol{F}(\boldsymbol{x}) = \sum_{\langle i,j\rangle \in C} \boldsymbol{e}_{ij}^T \boldsymbol{\Omega}_{ij} \boldsymbol{e}_{ij} \tag{2}$$

Here e_{ij} is the error function described by the edge from node i to node j and Ω is the information matrix. This error function describes the difference between the expected observation, i.e. our current estimate, and the actual measurement. Generally not all nodes are connected, thus for most combinations of i and j, Ω is zero. This results in F being a sparse matrix. Minimizing F is achieved by a first order Taylor expansion, differentiation and solving for zero. We start by expanding:

$$\boldsymbol{F}(\tilde{\boldsymbol{x}} + \boldsymbol{\Delta x}) = c_{ij} + 2\boldsymbol{b}^T \boldsymbol{\Delta x} + \boldsymbol{b}^T \boldsymbol{\Delta x}^T \boldsymbol{H}_{ij} \boldsymbol{\Delta x} \tag{3}$$

where $\boldsymbol{H}$ is known as the Hessian matrix. the Hessian matrix is constructed using the Jacobian and information matrix Ω of the constraints:

$$\boldsymbol{H_{ij}} = \boldsymbol{J_{ij}^T \Omega_{ij} J_{ij}} \tag{4}$$

Equation 3 can be minimised by solving the linear system:

$$\boldsymbol{H \Delta x}^* = -\boldsymbol{b} \tag{5}$$

The solution to this equation is incrementally added to the initial guess in order to minimize $\boldsymbol{F}$. As mentioned in [24] Gauss-Newton and Levenberg-Marquardt solvers are particularly well equipped to handle these problems. This process is known as multivariate function minimization. What should be noted is that for graph-optimisation the Jacobian and cost-function (or error) are required for each constraint, such as a range-only constraint (see section V for the definition of these properties).

B. RTAB-Map

As mentioned, the chosen framework in which we integrate the range-only constraint is RTAB-Map in 2D mode. RTAB-Map was chosen for its modularity and flexibility regarding sensor layouts. To enable the proposed integration we describe some specifics regarding the back-end of RTAB-map. RTAB-Map generates "Signatures" as poses in the aforementioned graph. These signatures also include all measurement data from all sensors used to construct that pose, i.e. all measurements conducted between the construction of the previous and the current pose. Signatures are connected via "Link" objects, which describe the constraint (e.g. translation and/or rotation) between two signatures. Observed landmarks are handled as measurements and are later converted to nodes in the graph at the back-end of RTAB-Map. The type of constraint by which the landmarks are connected depends on the covariance of the observations. A very large rotational covariance (i.e. Cov$>=$ 9999.0) causes a constraint with just an XY-translation. If the covariance is smaller (i.e. Cov$<$ 9999.0) a full SE2 transform is used.

V. Method

We integrate range-only constraints from Wi-Fi RSS into RTAB-map. We evaluate the output of RTAB-map with this new integration and compare it to a baseline.

A. The Wi-Fi Integration

RTAB-map is able to use several different back-ends for graph optimisation, one of which is G2O. No native 1D-constraint exists within any of the back-ends that are supported by RTAB-map. This constraint type is required due to the nature of the measurement, the RSS. The merit of G2O is that the user can create a new constraint type. Therefor G2O was the back-end of choice. We create a new 1D-constraint that is able to convey the spatial distance in a 2D-environment. G2O allows for the Jacobian, for optimisation, to be determined either numerically or analytically. For speed purposes, we did this analytically. We begin by defining our error as follows:

$$e = r - r_0 \tag{6}$$

where r_0 is the required euclidean distance and r is the current distance estimate from the graph:

$$r = \sqrt{\Delta x^2 + \Delta y^2} \tag{7}$$

The Jacobian is now formed by:

$$\begin{aligned}
\frac{\partial e}{\partial x_1} &= \frac{-\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}} \\
\frac{\partial e}{\partial y_1} &= \frac{-\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} \\
\frac{\partial e}{\partial \theta_1} &= 0 \\
\frac{\partial e}{\partial x_2} &= \frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}} \\
\frac{\partial e}{\partial y_2} &= \frac{\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}}
\end{aligned} \tag{8}$$

With this Range-Only constraint RTAB-map is now better equipped to handle Wi-Fi measurements. To enable RTAB-map to generate these constraints an alteration in the back-end is required. As mentioned in the previous section (IV-B) the type of constraint depends on the covariance of the measurement. We use the same technique to detect a single dimensional measurement (i.e. range-only) by setting a y-covariance larger than 9999.0. However, this covariance matrix and the distance measurement must be generated from the RSS measurement. For this a kernel function is needed which is described in section V-C. Contact the author for the implementation.

B. Initialisation of Wi-Fi landmarks

A single range-only constraint does not provide a single location where a node should be initialised before graph-optimisation occurs. Also, when multiple constraints are added this may still be ambiguous. We will refer to this as the disambiguation problem from now on. During the early stages of this research, different initialisation methods have been tested for accuracy and computational load. An example, the results, and further description of this problem can be found in appendix E. To summarise this appendix, the most successful initialisation that was tested is taking the weighted mean of the connected poses, as follows:

$$(x_{init}, y_{init}) = R \sum_{i=0}^{n} \frac{1}{r_i^2} \mathrm{trans}(p_i) \tag{9}$$

Here $\mathrm{trans}(p_i)$ is the translational component of the connected pose p_i and r_i is the range measurement of the constraint. 'R' is the total squared range for normalisation:

$$R = \sum_{i=0}^{n} r_i^2 \tag{10}$$

This way the translational part of each of the connected poses is weighted by $1/r_i^2$.

C. Kernel function

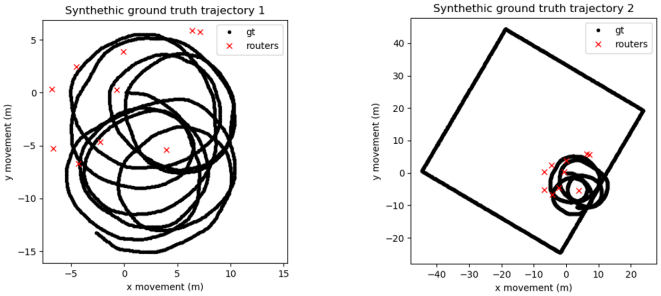
To determine the relation between RSS from Wi-Fi sensing and distance, as discussed in section II, we use the following signal propagation model:

$$\begin{aligned}
P(r) &= P(r_0) - 10\alpha \log \frac{r}{r_0} && (11)\\
P(r) &= P(r_0) - 10\alpha \log r - \log r_0 && (12)\\
P(r) &= C - 10\alpha \log r && (13)\\
\frac{C - P(r)}{10\alpha} &= \log r && (14)\\
r &= 10^{\frac{C-P(r)}{10\alpha}} && (15)
\end{aligned}$$

with:

$$C = P(r_0) - \log r_0 \tag{16}$$

Here $P(r)$ is the measured power, or RSS at a distance r. C is a reference constant determined by a reference measurement and α is the path-loss coefficient. Both C and α are empirically determined.



(a) Simulated trajectory 1: the robot stays near the simulated APs for testing the trajectory correctional power.

(b) Simulated trajectory 2: the robot temporarily is out of range of the APs for testing the loop-closure potential.

Fig. 1: The ground-truth trajectories that are used for the synthetic data experiments. Simulated APs visualised in red.

VI. Experiments

A. Synthetic experiments

To test the range-only constraints in a controlled environment, we conduct synthetic data experiments. We simulate a robot trajectory (random walk) and add drift to a virtual odometry source. This drift consistently turns the perceived robot orientation slightly counterclockwise relative to the actual position of the robot. Furthermore, we generate ten simulated Wi-Fi APs randomly near the origin. The exact specification of the trajectory and data generation can be found in appendix B. Range-only constraints are generated as if from the actual position of the robot, thus possibly correcting the noise added to the odometry.

In the first simulated trajectory (trajectory 1) the robot traverses an area within the vicinity of the simulated APs. Thus, range-only measurements are available over the entire trajectory. This in order to test the correctional power of the range-only measurements. In the second trajectory (trajectory 2) the robot temporarily leaves the range of the simulated APs. The robot is allowed to accumulate drift from the noise added to the odometry as no AP is perceived within this part of the trajectory. This second test is conducted to find whether loop-closure would become easier due to our addition. Both trajectories can be seen in respectively figures 1a and 1b.

At first, we generate the range-only measurements that are 100% accurate and we limit the range of these measurements to twenty meters. This experiment is conducted to expose any weaknesses or strengths of our implementation. Furthermore, a covariance of one meter was given to these measurements such that they would have a large influence on the graph optimisation process.

To conduct a more realistic synthetic experiment we add Gaussian noise to the range-only measurements. The amount of noise is similar to the kernel function shown in figure 8, i.e. a standard deviation of 2.3 meters. Furthermore, the range-only measurements are generated roughly every 2.5 seconds, similar to the real-world experiment.

B. Real-World experiments

To establish the influence of the proposed integration in the real-world we compare multiple robot configurations with and without the Wi-Fi measurements. Below we list the available options of deployable sensors on our platform:

- Odometry sources:
  - Wheel Odometry
  - PointCloud/ICP Odometry
  - RGBD Camera
- Available extra options:
  - Visual loop-closure from RGBD Camera
  - Proximity matching from Velodyne/Ouster converted to Laser-Scan
  - Wi-Fi sensing

All these combinations result in a total of 24 possible experiments per dataset. It might not make sense to have RTAB-Map use the camera for estimating odometry, but not have it use this for loop-closure detection. However, if our alteration to RTAB-map proves accurate enough, where loop-closure detection is not used, the argument that loop-closure detection is obsolete may be considered. Not requiring loop-closure detection may speed up and simplify the SLAM process significantly. The complete list of these 24 configurations is available in appendix A.

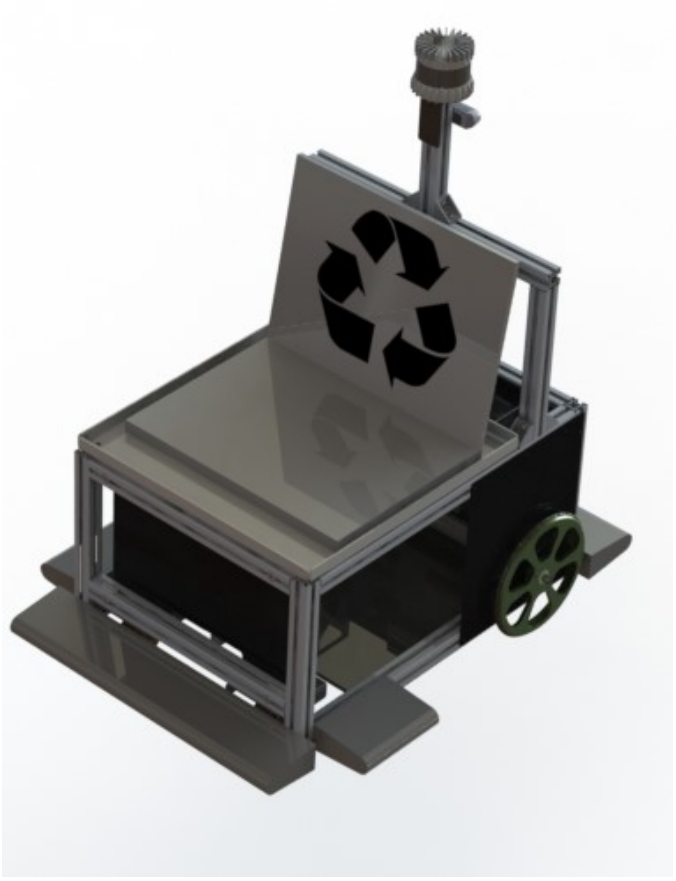


Fig. 2: Rendering of the robot the experiments are conducted on. Camera and Lidar can be seen on top. Safety bumpers mounted lower on the robot.

We collect three datasets, similar in nature to the synthetic data experiments. The first two traverse the area within range of the used Wi-Fi APs, with different traversal speeds. We vary the traversal speed because we suspect that this may influence the accuracy of the achievable kernel function. We collect a third dataset where the robot leaves the range of the APs. Again, like in the synthetic data experiments, we test the trajectory correctional capabilities and potential for assisting in loop-closure respectively. The entire process is conducted in 2D, therefore the Wi-Fi APs that are logged are in roughly the same plane as traversal as the robot, e.g. on the same floor.

1) Ground-truth: to find a ground-truth for real-world experiments a configuration has to be found which has minimal drift in an environment similar to our testing environment. This ground-truth configuration is required for evaluation of other configurations and for generating the Wi-Fi kernel function.

The configuration that we used and suspected to be optimal for our robot is listed in table I.

	Algorithm	Sensor Used
Odometry Generation	ALOAM	Ouster OS0 or Velodyne VLP-16
Loop-Closure Detection	RTAB-Map	Realsense
Proximity Detection	RTAB-Map	Laser Scan from Lidar

TABLE I: The Ground-Truth configuration for the robot. Note that proximity detection is conducted with laser-scans. Laser-scans are generated by converting the pointclouds generated by the Lidar scanner.

ALOAM is a highly accurate iterative closest point algorithm that is able to generate odometry. It was qualitatively determined to be significantly more accurate than the ICP-odometry source mentioned in section VI. We have used the ALOAM implementation that was developed in [25].

This configuration is compared to the external measurements from a tracking camera. This tracking camera has an accuracy of 1cm. This way it can be determined whether the accuracy of the ground-truth configuration is sufficient for our purposes. In our case this external measurement is only available over a very small space in which the trajectory can be evaluated. Because of this limitation we are required to make the following assumption: if the accumulated drift, as measured externally, between the beginning and the end of the trajectory is small, the error in between is also small. This is, in our opinion, true as the accuracy of the current pose estimate depends on all previous poses. We want to extend the argument further with the following: if the accumulated drift is consistently small, all trajectories produced by this setup, regardless of the presence of an external measurement device, are accurate. This accurate trajectory is considered the ground-truth configuration for the real-world measurements.

2) The Wi-Fi kernel function: measurements are collected by the aforementioned ground-truth configuration where the location of the Wi-Fi access points is known. The distance to these access points, as measured from the ground-truth trajectory, is plotted as a function of the measured RSS. Only 2.4 GHz signals can be detected by our robot setup, thus no distinction between 5 GHz and 2.4 GHz signals is necessary. This method does rely on the accuracy of the ground-truth setup. However, as RSS measurements are very noisy and quantised, roughly 3.5 dBm gaps, we argue that the error induced due to these properties is higher than the error induced due to the localisation of our ground-truth setup. Furthermore, two observations during experimentation caused change in our experimentation. There RSS measurements are capped at 100% when close to the AP. This results in the indistinguishability of the first 7.5 meters. For this reason, measurements with a 100% signal strength are not used. We also observed that the kernel functions differ significantly per AP and do not generalise well. For this reason each AP was given its own kernel function. This experiment is conducted at two different traversal speeds as we suspect this may influence the accuracy of the measurement.

C. Evaluation

We evaluate by comparing the output trajectory of the different configurations of the SLAM algorithm to the ground-truth. For this comparison, aliasing is required which can be conducted in time. This is possible because the robot is at one and only one position at any point in time. RTAB-Map includes timestamps in the generated poses. These timestamps are equal to the timestamp of the last data-point received for constructing that pose and thus to the timestamp of when this pose was constructed. As this timestamp may deviate from the timestamps in the ground-truth, linear interpolation is used. The ground-truth trajectory is interpolated such that all poses coincide with the poses of the estimated trajectory that we want to evaluate. This way the error due to aliasing is minimised. Once we have aliased the graphs we compute the following metrics for evaluation: the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE). These error metrics are shown below as described in [21]:

$$\mathrm{ATE}(\mathbf{F}_{i:n}) := \Big(\frac{1}{n}\sum_{i=1}^{n} ||\mathrm{trans}(\mathbf{F}_i)||^2\Big)^{\frac{1}{2}} \tag{17}$$

Here "trans" specifies it is the translational part of $\mathbf{F}$, where $\mathbf{F}$ is defined as:

$$\mathbf{F}_i := \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{18}$$

Here, $\mathbf{S}$ is the Horn transform [26], i.e. the rigid-body transformation from the estimated trajectory $\mathbf{P}_{1:n}$ onto the ground truth trajectory $\mathbf{Q}_{1:n}$ The ATE can be described as the standard deviation of the entire trajectory relative to the ground truth. The RPE is defined as:

$$\mathrm{RPE}(\mathbf{E}_{1:n}, \Delta) := \Big(\frac{1}{m}\sum_{i=1}^{m} ||\mathrm{trans}(\mathbf{E}_i)||^2\Big)^{\frac{1}{2}} \tag{19}$$

Where $\mathbf{E}_i$ is defined as:

$$\mathbf{E}_i := \big(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta}\big)^{-1}\big(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}\big) \tag{20}$$

Here again $\mathbf{P}_{i:n}$ is estimated trajectory and $\mathbf{Q}_{1:n}$ is the ground truth trajectory. The RPE can be normalised over all possible time intervals Δ as follows:

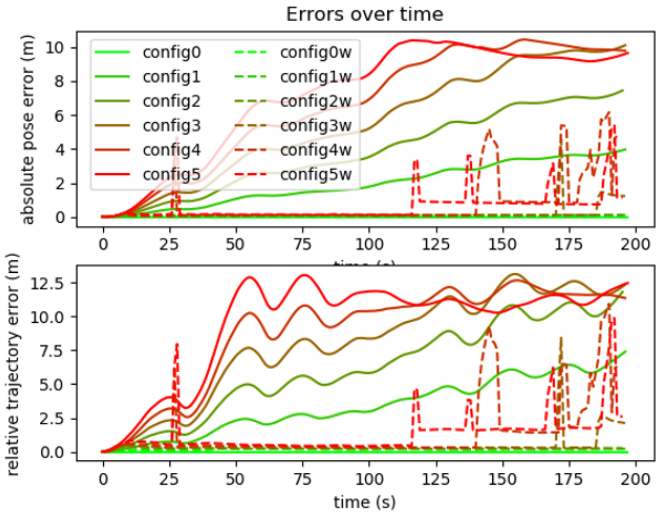
$$\mathrm{RPE}_{\mathrm{norm}}(\mathbf{E}_{1:n}) := \frac{1}{n}\sum_{\Delta=1}^{n} \mathrm{RPE}(\mathbf{E}_{1:n}, \Delta) \tag{21}$$

The RPE can be described as the standard deviation of the movement of the robot relative to the ground truth trajectory.

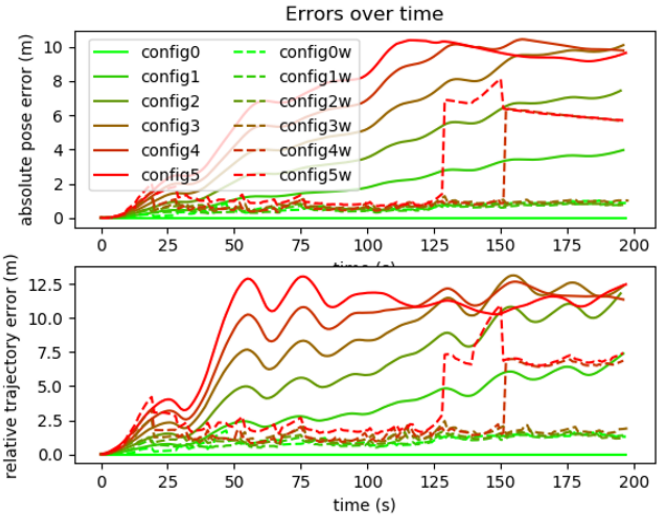
VII. Results and discussion

A. Synthetic experiments

1) Correction potential: for trajectory 1, the trajectory where the robot stays within range of the APs, the 100% accurate limited-range measurements contribute a lot to the performance of the SLAM system. Almost all of the noise that was added to the odometry can be recovered by the range-only measurements, see figure 3a. However, some non-zero

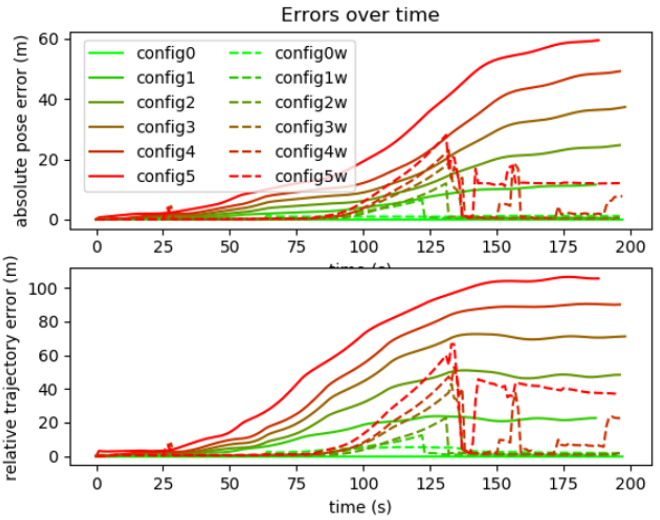


(a) ATE and RPE for the near 100% accurate measurements. The range-only measurements (dashed lines) correct the error from the noisy odometry significantly.

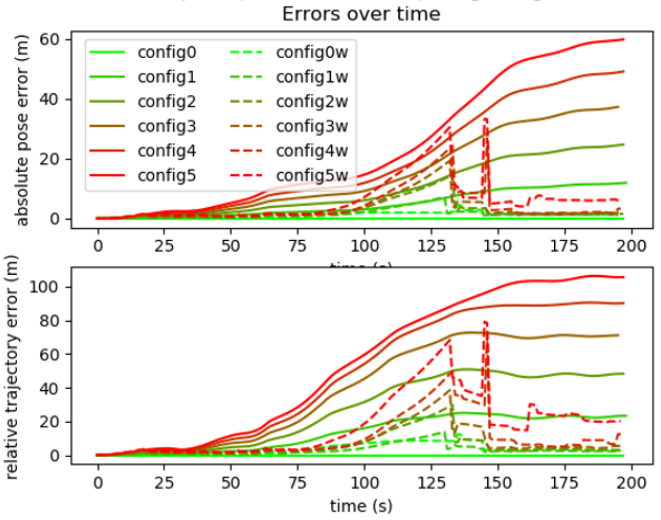


(b) ATE and RPE for the realistic measurements. The added Wi-Fi measurements are able to compensate for the added noise less effective than in the perfect measurement setup.

Fig. 3: ATE and RPE as function of time. During this trajectory the robot stays within the range of the simulated APs. The number displays the level of noise and the dashed lines (with "w" in the name) display the range-only integrated versions of RTAB-map. The bump early on is related to the disambiguation problem. The sudden increase in error later (at around 110) is related to the effect from figure 5.



(a) ATE and RPE for the near 100% accurate limited range measurements. With the exception of a very small bump at roughly $t = 25$, the addition of range-only constraints always improve performance.



(b) ATE and RPE for the realistic simulated measurements. The results, other than a slight increase in error overall, are similar to figure 4a.

Fig. 4: ATE and RPE as function of time. In this trajectory the robot temporarily leaves the range of the APs. The number displays the level of noise and the dashed lines (with "w" in the name) display the range-only integrated versions of RTAB-map.

error was still observed early on in the process; at around $t = 25s$ and at $t >= 110s$. The bump at $t = 25s$ can be related to the disambiguation problem as explained in section V and appendix E. At this time the robot is near one of the APs and has collected little information of that AP orthogonal to the current direction of travel. This causes the disambiguation problem to be solved incorrectly by the graph-optimiser and the measured error increases. This error is recovered almost instantly as soon as more spatial information is available, i.e. measurements orthogonal to this trajectory. At around $t = 110s$ we again see the error increase, this time more permanently. We relate this to the tension in the graph that is built up by the consistent drift of the odometry and the attempted correction of the range-only measurements. We observed the trajectory doing a counterclockwise loop (see figure 5), even though the full simulated trajectory is only right turns (as can be seen in figure 1a). This releases tension in the graph and decreases the error for the graph optimisation process. However, this same effect causes the ATE and RPE to increase.

In the more realistic case (figure 3b) we also observe the early disambiguation problem as well as the tension release. However, as the range-only measurements are noisy, the maximum achievable performance is decreased. This can be seen as the minimum achievable error is higher compared to the 100% accurate measurements (figure 3a).

2) Loop-closure potential: for the second trajectory we see some interesting behaviour (see figures 4a and 4b). During the first part of the trajectory the APs are localised and the trajectory is continuously corrected similar to figures 3a and 3b. Next, during the traversal of the square, when the APs are out of range, the robot is allowed to accumulate error. As soon as the robot re-enters the range of the APs, the measured error decreases as the trajectory is pulled to the correct position and the accumulated error is partly recovered. However, the trajectory may return to the APs under an angle causing the APs to be localised on the wrong side of the returning robot. This results in an increased measured error. After which, when more information about the relative location of the APs becomes available to the returning trajectory, the AP is localised correctly again. The total trajectory is corrected and the error drops significantly. This is precisely what we see for most of the range-only integrated configurations. This misalignment did not occur as much for the lower levels of noise (configs 0-3). Furthermore, the accumulated drift for configuration 5 was too large for the range-only measurements to recover completely. A visual comparison of the final trajectories (config 3) can be seen in figure 6.

B. Real-world experiments

1) Ground-truth setup: we compare the output of the ground-truth setup of the robot to the output of the base station (see figure 7). The length of the total trajectory is 57.4 meters, as calculated by the robot. In this time the total accumulated drift (as measured compared with the base station) is given by the RPE-1 error. This error is the non-normalised RPE as described, eq. 19 with $\Delta = 1$.

ATE	RPE-N	RPE-1
0.046	0.193	0.118

TABLE II: The errors when the ground-truth setup is compared to the base-station.

The total accumulated drift is only 11 cm over almost 60 meters and thus sufficiently accurate for our purposes (table II). Any other configuration that is used deviates at least 1m from this ground-truth and is thus less accurate. Due to the small size of the area covered by the base station there are only six poses at the beginning of the trajectory and seven poses when returning. As this is not a full trajectory comparison, the ATE or normalised RPE have no real significant meaning but are given for completion sake. Furthermore, unfortunately due to time restrictions, no more measurements were conducted which can confirm that this setup produces a low drift consistently. However, a qualitative analysis was also conducted. The comparison between the output of the SLAM program and the building plans can be seen in figure 19 in appendix D. This figure shows a very good correspondence, so the error can be assumed to always be within 1 m across the whole trajectory.

2) Kernel function: in table III and table IV we report the estimated parameters of the kernel function for the APs. The kernel function for AP1 for the slow traversal is displayed in figure 8, notice that the x-axis is reversed. These counter-intuitive axes are chosen for the reason that they describe the function we are looking for, i.e. distance as a function of RSS rather than the other way around. The standard deviation in these plots was calculated per signal strength level as the receiver produces quantised data. The least square error for the an unbiased estimator was used:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{N} |D_i - \text{fit}(RSS_i)|^2}{N-1}} \tag{22}$$

Access Point	C	α	$\bar{\sigma}$
AP0	1.99	4.77	2.90
AP1	15.16	6.19	3.85
AP2	29.73	7.13	4.81
AP4	34.59	6.92	4.67

TABLE III: C, α and the mean standard deviation of the measurements for the four APs within the area of traversal as determined by the fit from measurement data and the ground-truth configuration whilst travelling fast.

Access Point	C	α	$\bar{\sigma}$
AP0	-7.31	4.16	4.06
AP1	19.80	6.36	4.04
AP2	9.40	5.40	4.37
AP4	31.26	6.56	6.62

TABLE IV: C, α and the mean standard deviation of the measurements for the four APs within the area of traversal as determined by the fit from measurement data and the ground-truth configuration whilst travelling slow.

Furthermore, when comparing the estimated distance from RSS measurements with the actual distance, we observe a shift in time (not visualised). In the case the robot travelled towards the AP, the estimated distance from the kernel function would often be slightly long and vice versa. Synthetic data experiments show that Gaussian noise may not hurt the performance as much, however as will become clear later, consistent noise may be detrimental to the achievable performance. This offset was observed to decrease slightly when traversal was slower. However, as can be seen from the mean standard deviation (see table III and table IV) the models did not become significantly more accurate due to the noisiness of the measurements.

The measured standard deviations for the Wi-Fi sensor when we use this signal propagation model are high compared to the other available sensors. The model does not describe any multipathing, interference or loss due to obstacles in the line-of-sight to the AP. For all of these effects to be compensated information is required that we a priori do not posses. However, estimates may be refined after localisation with this information as collected from the SLAM process. This was however out of the scope of this research.

3) Experimental results Wi-Fi integration: We see somewhat similar results in the real-world experiments compared to the synthetic data experiments. In figure 9 the configurations are listed that benefit from our integration. Figures 10 and 11 are shown for visual context. These figures show the ground-truth trajectory, the configuration without our integration and with our integration. Note that only wheel-odometry is noisy/faulty enough to benefit from the noisy range-only measurements. For the configurations where the Wi-Fi implementation was not beneficial either the performance dropped mildly or very extremely (see figure 13). These increases are respectively caused by mild disturbances from noisy range-only measurements and wrong solutions to the disambiguation problem, causing extreme deformations in the graph. We suspect that the consistency of the errors made in estimating the distance to the APs was the main contributor, as this issue is rarely observed in the synthetic environment.

When we look at the errors over time, there are some effects that should be addressed. Erratic behaviour, instantly increasing and decreasing the error values, was sometimes observed, see figure 14. This occurs when with the current information not one clear solution exists to the disambiguation problem. A possible solution to this problem would be that APs are only integrated into the graph until sufficient information is collected orthogonal from the general direction of travel. Finally, in the third trajectory evaluation (figure 14) in all Wi-Fi integrated configurations a drop can be seen at around $t = 700s$. This drop is related to the first detection of an earlier seen, and localised, AP. Due to this the robot is able to correct some of the accumulated drift. The second drop, at around $t = 880s$, is related to a visual loop-closure detection.

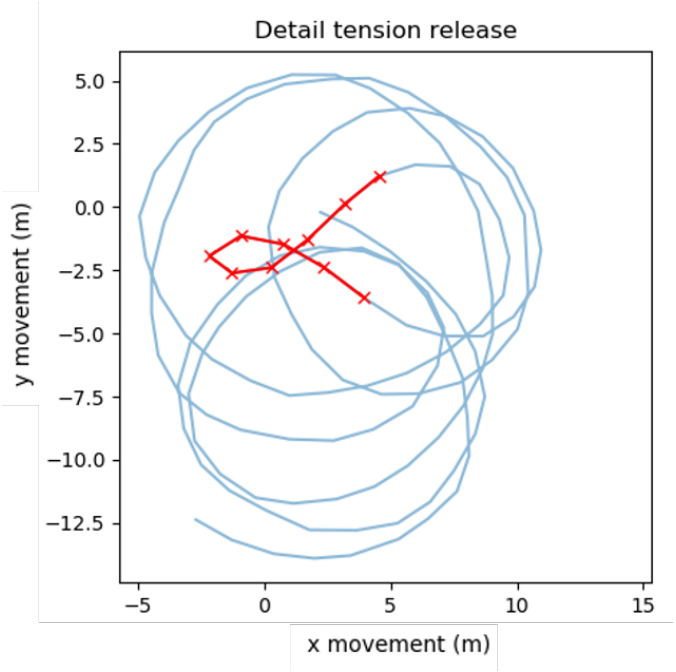


Fig. 5: Due to the constant correction of the noisy odometry by the range-only constraints tension is built up in the graph. This tension is released resulting in the red marked left turn. This turn decreases the error for the graph optimisation but causes a sudden increase in ATE and RPE.

Fig. 6: Synthetic data experiment: Comparison of the Ground-Truth and the final trajectory with and without the 100% accurate range-only constraints. The range-only constraints were able to compensate a large portion of the added noise.

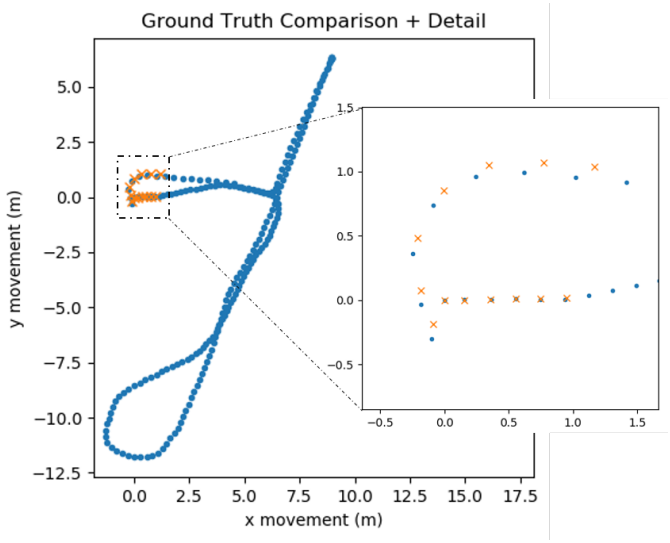


Fig. 7: The ground-truth setup compared to the base-station. The full trajectory is shown with a detailed overlay of the area covered by the base-station.

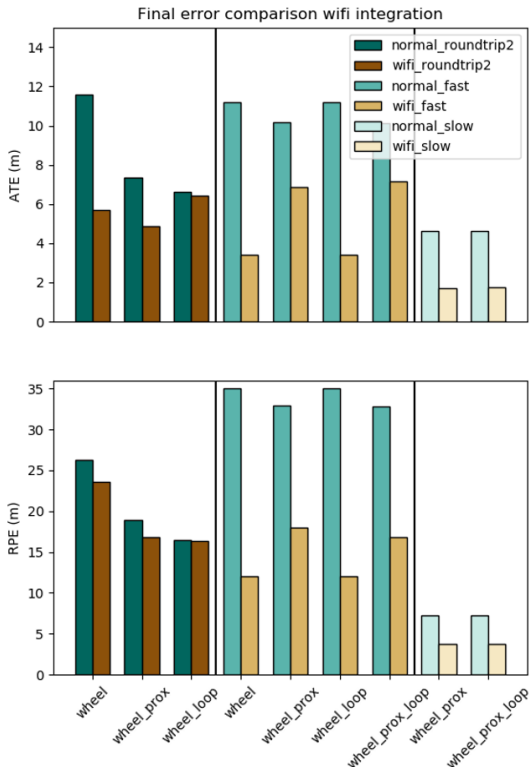


Fig. 9: ATE and RPE for the configurations that benefit from our implementation. The graph is divided in three parts, each part for one data-set (specified in the legend). Note that only wheel odometry is noisy/faulty enough to noticeably benefit from the implementation.

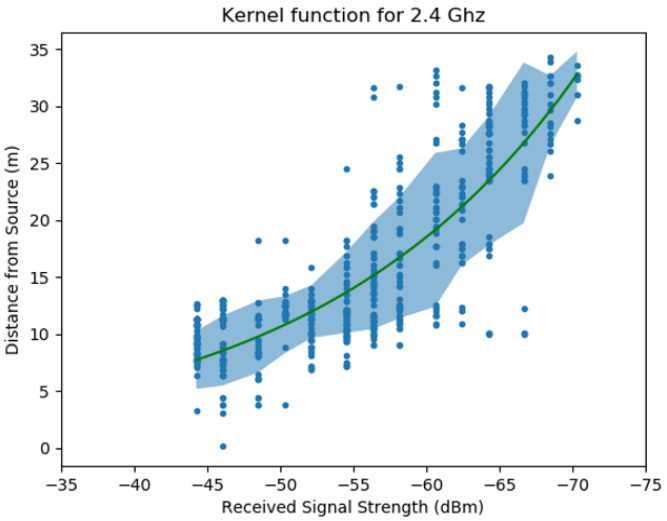


Fig. 8: The kernel function for AP0 measured from the slow trajectory that remains within the range of the APs.

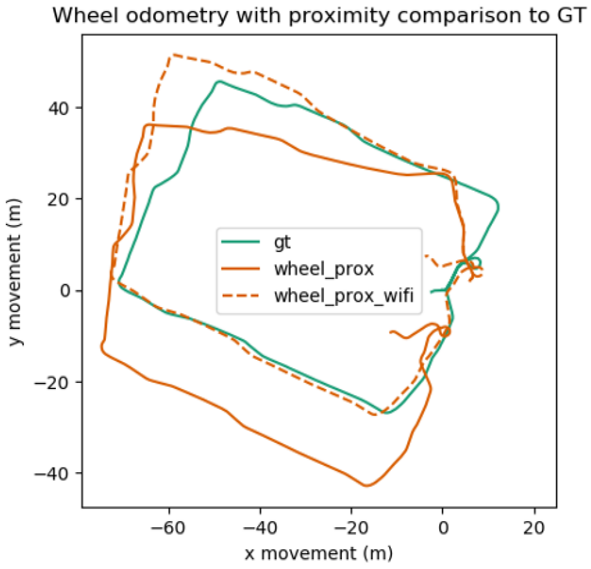


Fig. 10: Comparison to the ground-truth for wheel odometry with proximity matching enabled. The accumulated drift causes a noticeable gap between the start- and end-point of the trajectory. In the Wi-Fi enabled configuration this gap is not present and the estimated trajectory is generally closer to the ground-truth.

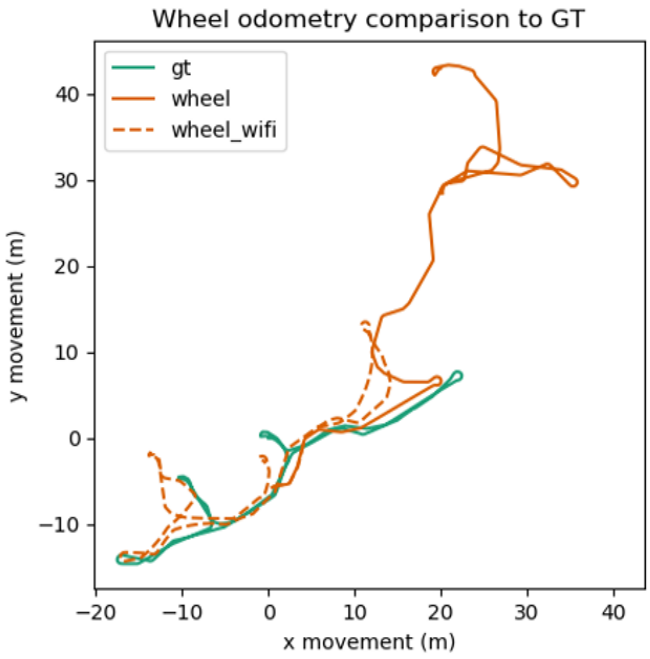


Fig. 11: Comparison of ground-truth to a wheel odometry configuration. The Wi-Fi integration results in a return to the start whereas the baseline wheel odometry makes a mistake and curves right when backtracking towards the starting location.

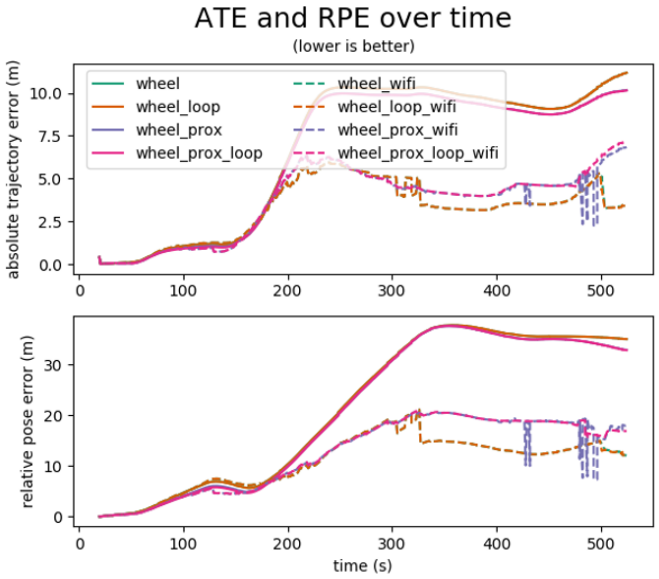


Fig. 12: ATE and RPE over time for the fast traversal of a trajectory not leaving the range of the Wi-Fi APs. The wheel odometry source makes a mistake not turning around to return in the opposite direction. This mistake is reflected as the high increase in errors at around $t = 200$.

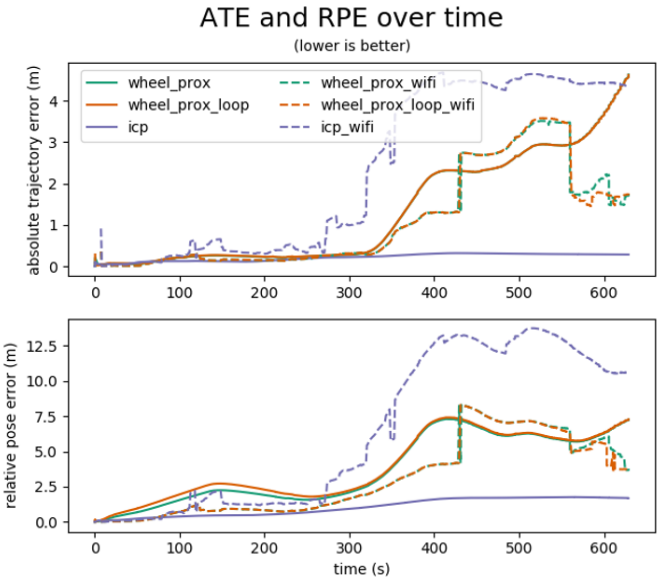


Fig. 13: ATE and RPE over time for the slow traversal of a trajectory not leaving the range of Wi-Fi APs. An incorrectly solved disambiguation of one of the detected APs, at $t = 450s$, causes a steep increase in error for the ICP-odometry which is never recovered. A similar thing occurs at $t = 570s$ for the other configurations, however this error is recovered at $t = 700$.

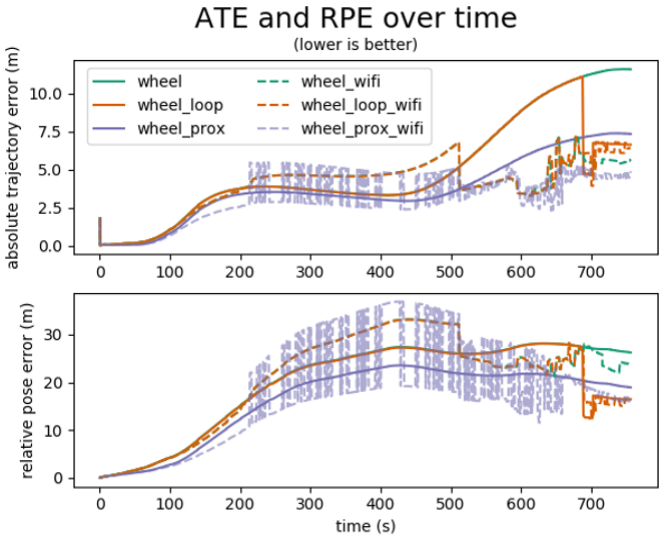


Fig. 14: ATE and RPE over time for the third trajectory, i.e. temporarily leaving the range of the APs. For the "wheel_loop_wifi" and the "wheel_wifi" configurations we clearly see the sudden decrease in error when the first AP is detected after accumulating drift ($t = 500s$). For the "wheel_prox_wifi" configuration the disambiguation problem causes the error to fluctuate significantly.

VIII. Conclusion

As presented, the addition of range-only constraints can be very effective at providing extra information for a SLAM process. This information can be used to accurately correct noisy odometry. However, the disambiguation problem persists, even though the initial poses for the APs are optimised to minimise the occurrence of wrong optimisation. This problem arises especially when too little information orthogonal to the general direction of travel is collected. In this case multiple solutions may exist to the graph optimisation, which hurts performance a lot. When the range-only constraints are sufficiently accurate and enough orthogonal information is collected, the disambiguation problem was often solved quickly, as seen in our synthetic environment.

In the real-world experiments we observed that limited spatial information can be collected from RSS. This was due to the signal propagation model and the measurement setup that we used. Several effects, such as multipathing, line-of-sight and especially the temporal offset of measurements were not compensated for in this setup. This limited the achievable accuracy of the SLAM process for the real-world experiments. Regardless, in real-world experiments the less accurate robot configurations still benefit significantly from the RSS measurements. Specifically when wheel-odometry is used. The synthetic experiments show that, with accurate range measurements, very high levels of noise and fault correction can be achieved. Ultra-wide band or other time of flight sensing may be feasible candidates for this in the real-world. Our approach enables the use of such sensors in RTAB-map, with very little extra work required.

References

[1] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. ElasticFusion: Dense SLAM without a pose graph. *Robotics: Science and Systems*, 11, 2015.

[2] Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[3] Christine Evers and Patrick A. Naylor. Acoustic SLAM. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(9):1484–1498, 2018.

[4] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. d:1–21, 2020.

[5] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2D LIDAR SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1271–1278, 2016.

[6] Mathieu Labbe. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[7] Emilio Garcia-Fidalgo and Alberto Ortiz. IBoW-LCD: An Appearance-Based Loop-Closure Detection Approach Using Incremental Bags of Binary Words. *IEEE Robotics and Automation Letters*, 3(4):3051–3057, 2018.

[8] Zakieh S. Hashemifar, Charuvahan Adhivarahan, Anand Balakrishnan, and Karthik Dantu. Augmenting visual SLAM with Wi-Fi sensing for indoor applications. *Autonomous Robots*, 43(8):2245–2260, 2019.

[9] Ran Liu, Sumudu Hasala Marakkalage, Madhushanka Padmal, Thiruketheeswaran Shaganan, Chau Yuen, Yong Liang Guan, and U. Xuan Tan. Collaborative SLAM Based on WiFi Fingerprint Similarity and Motion Information. *IEEE Internet of Things Journal*, 7(3):1826–1840, 2020.

[10] Piotr Mirowski, Tin Kam Ho, Saehoon Yi, and Michael MacDonald. SignalSLAM: Simultaneous localization and mapping with mixed WiFi, bluetooth, LTE and magnetic signals. *2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013*, (October):1–10, 2013.

[11] Zhenghua Chen, Han Zou, Hao Jiang, Qingchang Zhu, Yeng Chai Soh, and Lihua Xie. Fusion of WiFi, smartphone sensors and landmarks using the kalman filter for indoor localization. *Sensors (Switzerland)*, 15(1):715–732, 2015.

[12] Chen Wang, Handuo Zhang, Thien Minh Nguyen, and Lihua Xie. Ultra-wideband aided fast localization and mapping system. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:1602–1609, 2017.

[13] Xu Fang, Chen Wang, Thien Minh Nguyen, and Lihua Xie. Model-free Approach for Sensor Network Localization with Noisy Distance Measurement. *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018*, pages 1973–1978, 2018.

[14] Xu Fang, Chen Wang, Thien-Minh Nguyen, and Lihua Xie. Graph Optimization Approach to Range-Based Localization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12, 2020.

[15] Joseph Huang, David Millman, Morgan Quigley, David Stavens, Sebastian Thrun, and Alok Aggarwal. Efficient, generalized indoor WiFi GraphSLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1038–1043, 2011.

[16] Ghz Wireless Lan, Rafael S De Souza, and Rafael D Lins. A New Propagation Model for. (5), 2008.

[17] Yong Guang Chen and Xiu He Li. Signal strength based indoor geolocation. *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, 32(9):1456–1458, 2004.

[18] Jie Yang and Yingying Chen. Indoor localization using improved rss-based lateration methods. *GLOBECOM - IEEE Global Telecommunications Conference*, (January), 2009.

[19] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[20] Oliver Wulf, Andreas Nüchter, Joachim Hertzberg, and Bernardo Wagner. Ground truth evaluation of large Urban 6D SLAM. *IEEE International Conference on Intelligent Robots and Systems*, pages 650–657, 2007.

[21] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. *IEEE International Conference on Intelligent Robots and Systems*, (September 2015):573–580, 2012.

[22] Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Rainer Kummerle. On Measuring the Accuracy of SLAM Algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[23] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kümmerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D. Tardós. A comparison of SLAM algorithms based on a graph of relations. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 2089–2095, 2009.

[24] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[25] CAO Shaozu QIN Tong. A-LOAM. https://github.com/HKUST-Aerial-Robotics/A-LOAM, 2019. [Online; accessed mar-2020].

[26] Berthold Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4:629–642, 04 1987.

APPENDIX A
CONFIGURATIONS

All robot configurations that are used for real-world experiments are listed below.

Odom	Camera	Laser	Wi-Fi
wheel	0	0	0
wheel	0	1	0
wheel	1	0	0
wheel	1	1	0
Camera	0	0	0
Camera	1	0	0
Camera	0	1	0
Camera	1	1	0
Laser	0	0	0
Laser	0	1	0
Laser	1	0	0
Laser	1	1	0
wheel	0	0	1
wheel	0	1	1
wheel	1	0	1
wheel	1	1	1
Camera	0	0	1
Camera	1	0	1
Camera	0	1	1
Camera	1	1	1
Laser	0	0	1
Laser	0	1	1
Laser	1	0	1
Laser	1	1	1

APPENDIX B
SYNTHETIC DATA GENERATION SPECIFICS

Startup: Ten Wi-Fi access-points are generated in the vicinity of the origin of the robot, namely $(x = 0, y = 0, \theta = 0)$.
Movement: Odometry is generated as a random walk with 10 hz messages. The y-translation is zero as robots do not drive sideways. The x translation is 0.1 meters (per message) with a uniformly distributed random addition of 0 to 0.1 meters. The yaw of the robot is uniform distribution between -5.85 and 4.05 degrees per message. Thus there is a slight bias to the right. This odometry is saved as ground-truth.
Noise addition: Noise is added to the relative translation and rotation per timestep, which is saved and accumulated over time. The noise is normally distributed with respective covariances 0.02 and 0.007 (radians) multiplied by the noise level. The noise level is specified per configuration and is displayed in the name of that configuration (see VII-B). A comparison of these trajectories can be seen in figure 15.
Wi-Fi measurements: range-only measurements are generated from the ground-truth, i.e. position of the robot and the access-points. The covariance set for these measurements is 1m. For the more realistic setup noise is added to these measurements with similar covariance as found in the kernel function VII-B, i.e. 2.3m.

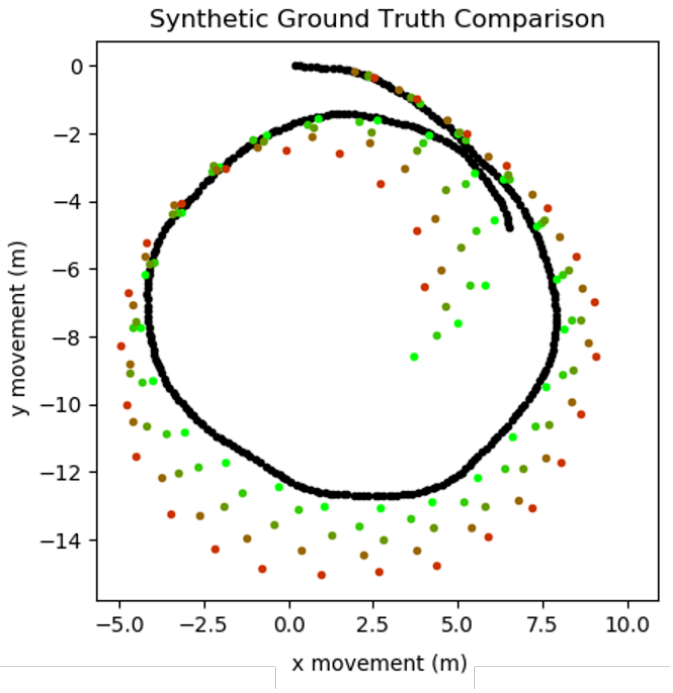


Fig. 15: The first 32 seconds of trajectory compared. In this configuration the robot stay's within the vicinity of the simulated access points. The number of the configuration indicate the level of noise as specified in B

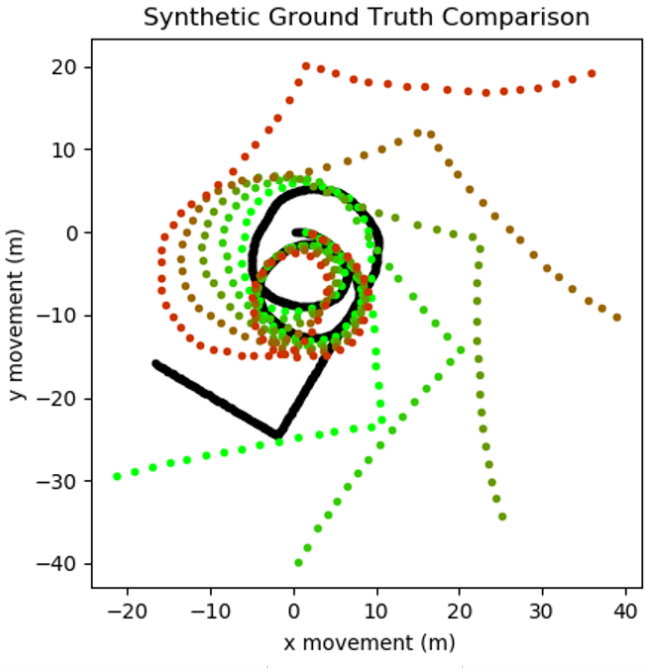
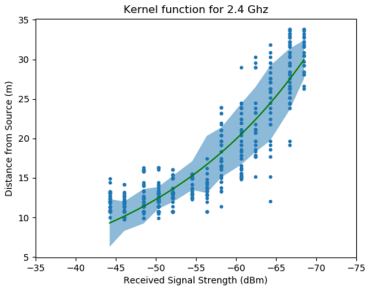
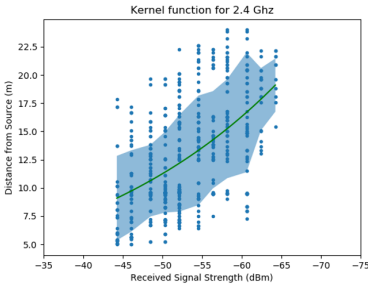


Fig. 16: Ground-truth trajectory compared to the trajectory with level 1 till level 5 noise for the trajectory leaving the vicinity of the simulated access points.

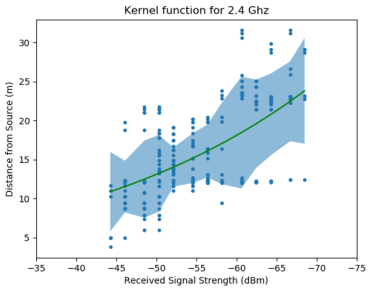
APPENDIX C
KERNEL FUNCTIONS VISUALISED



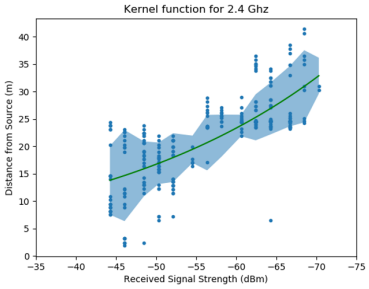
(a) AP 0 with C = 1.99 and α = 4.77.



(b) AP 1 with C = 15.16 and α = 6.19.

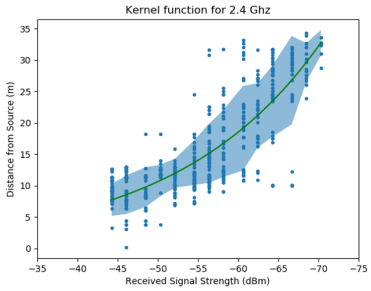


(c) AP 2 with C = 29.73 and α = 7.13.

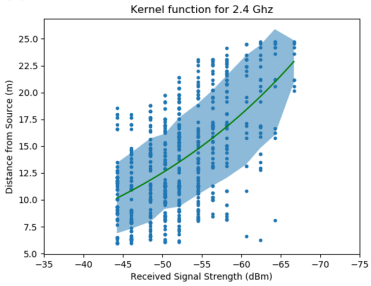


(d) AP 3 with C = 34.59 and α = 6.92.

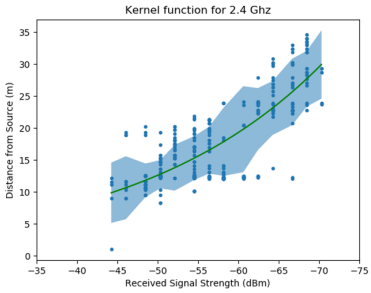
Fig. 17: Kernel function for the 4 used APs. Blue area is binned-covariance from the fit. This kernel function was generated traveling ”fast” in the vicinity of the APs. This kernel function was also used for dataset where the vicinity of the APs was left.



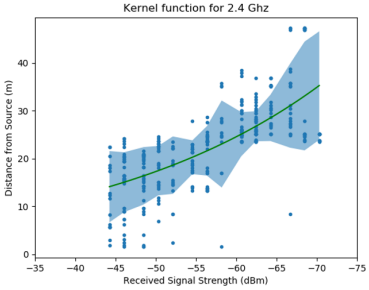
(a) AP 0 with C = -7.31 and α = 4.16.



(b) AP 1 with C = 19.80 and α = 6.36.



(c) AP 2 with C = 9.40 and α = 5.40.



(d) AP 3 with C = 31.26 and α = 6.56.

Fig. 18: Kernel function for the 4 used APs. Blue area is binned-covariance from the fit. This kernel fucntion was generated traveling ”slow” in the vicinity of the access-points

APPENDIX D
BUILDING PLAN COMPARISON

A qualitative analysis of the ground-truth setup is shown in the figure listed below. Here the building plans of the building are compared to the output of the SLAM algorithm. One can see that they clearly align. The length of the entire trajectory is roughly 266 meters.

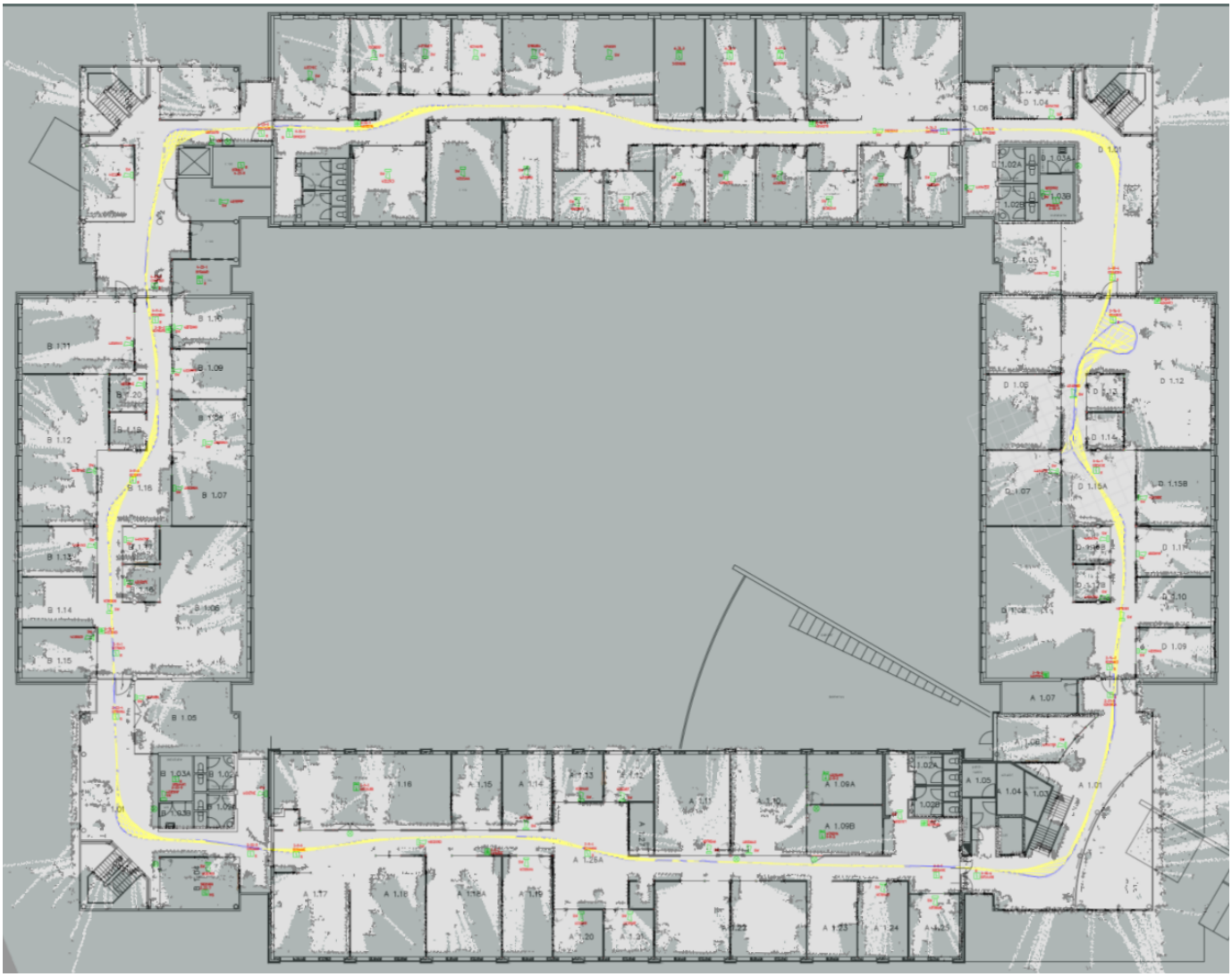


Fig. 19: The output of the ground-truth configuration of the robot in Enschede overlaid with the building plans of the same building. The largest difference can be seen on the inner side of the left side of the building. The two pictures have been matched by eye.

APPENDIX E
INITIALISATION FOR RANGE-ONLY CONNECTED NODES

A. Problem Description

For the range-only measurement, as defined in this thesis, a problem exists of multi-modality: multiple solutions may exist for a given graph. This is illustrated in figure 20. This space is expensive to calculate and is not a viable method for real-time localisation and mapping. Due to this fact an initial estimate should be given to the graph optimiser based on the rest of the graph. Three different methods were tested:

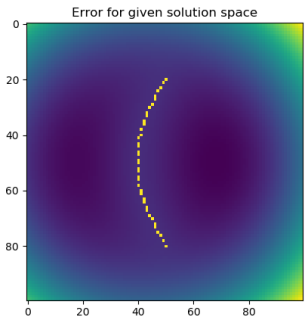
1) Initialise the landmark at the mean of all the poses at which the AP with a specific MAC-adress was detected. (m_pose)
2) Initialise the landmark at the normalised weighted mean of all the poses at which the AP with a specific MAC-adress was detected. (w_pose)
3) Initialise every single constraint with a random rotation and its own separate landmark which has a relative zero transform to each landmark with the same MAC-adress (aliasing-relation). This solution might be computational expensive and may not converge to a single solution. (rand_ori)

B. Methodology

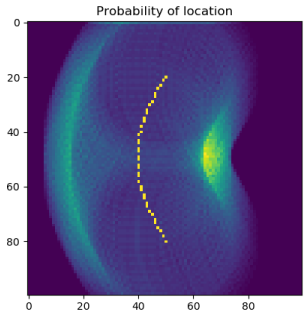
To determine the best initialisation the graph, in figure 21, was constructed. All measurements that are provided are perfect and have no noise. The poses from which the access point was measured are considered fixed. Every possible subsection of poses, minimum size 2, was optimised after each different optimisation method. Optimisation was deemed correct when the landmark was between 1.9 and 2.1 on both axes after optimisation.

C. Results

The results (see table V-VIII) list the performance of the different configurations as specified above. When failed, the amount that diverge and converge is also specified. In turn, the amount of measurements that were straight lines of those converging or diverging measurements is also specified. These amounts are specified because divergence can cause the entire graph to crash and no optimisation would be available, so convergence is the preferred behaviour in the case of bad optimisation. Furthermore, the straight lines still have a multi-modal solution space and are because of that more prone to crashing.



(a) The squared error space for the position of the access points measurement which should be localised on the right of this trajectory. A local minimum can be seen on the left



(b) The probability space of the same trajectory and range only measurements. In this space the solution is less ambiguous but also less smooth.

Fig. 20: A small piece of simulated trajectory with an access point at (60, 50).

D. Conclusion

The weighted mean pose as initial estimate performs consistently the better over all solvers compared to the other initial estimates. With over 94% of graphs converging to the correct solution we conclude that using the mean_weighted method is the most robust way of accurately optimising graphs with range-only constraints. Furthermore, only a very small portion of the simulations diverge (less than 4‰) for the Levenberg-Marquardt version of the different solvers. Also, both the "m_pose" and "w_pose" implementations are comparatively (roughly one third) faster than the "random_orientation" implementation.

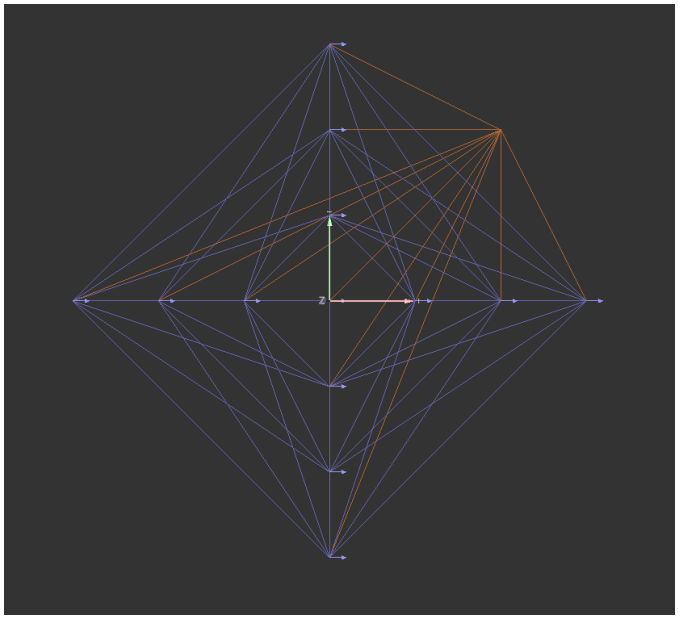


Fig. 21: The goal graph for the different initialisation methods. Different poses on the x- and y-axis with a simulated AP at (2,2).

mean_pose	succes	fail	conv	lines	div	lines
gn_pcg	90,89%	9,11%	71,54%	7,50%	28,46%	74,53%
lm_pcg	90,89%	9,11%	93,02%	34,63%	6,98%	0,00%
gn_var	90,93%	9,07%	63,88%	8,44%	36,12%	74,63%
lm_var	90,89%	9,11%	93,02%	34,63%	6,98%	0,00%
gn_var_cholmod	90,91%	9,09%	63,66%	8,46%	36,34%	74,07%
lm_var_cholmod	90,89%	9,11%	93,02%	34,63%	6,98%	0,00%
gn_var_eigen	90,91%	9,09%	63,66%	8,46%	36,34%	74,07%
lm_var_eigen	90,89%	9,11%	93,02%	34,63%	6,98%	0,00%

TABLE V: Performance of the mean_pose initialisation method for different linear solvers.

mean_weighted	succes	fail	conv	lines	div	lines
gn_pcg	94,82%	5,18%	49,76%	4,74%	50,24%	85,45%
lm_pcg	94,90%	5,10%	93,05%	61,86%	6,95%	0,00%
gn_var	94,84%	5,16%	33,41%	4,26%	66,59%	83,27%
lm_var	94,90%	5,10%	93,05%	61,86%	6,95%	0,00%
gn_var_cholmod	94,84%	5,16%	32,94%	4,32%	67,06%	82,69%
lm_var_cholmod	94,90%	5,10%	93,05%	61,86%	6,95%	0,00%
gn_var_eigen	94,84%	5,16%	32,94%	4,32%	67,06%	82,69%
lm_var_eigen	94,90%	5,10%	93,05%	61,86%	6,95%	0,00%

TABLE VI: Performance of the mean_weighted initialisation method for different linear solvers.

random_ori	succes	fail	conv	lines	div	lines
gn_pcg	80,64%±0,16%	19,36%±0,16%	08,72%±2,59%	01,31%±0,96%	91,28%±2,59%	11,93%±2,06%
lm_pcg	80,36%±0,18%	19,64%±0,18%	08,85%±3,07%	01,08%±0,80%	91,15%±3,07%	12,48%±2,42%
gn_var	80,36%±0,33%	19,64%±0,33%	08,41%±0,47%	00,58%±0,92%	91,59%±0,47%	08,39%±0,52%
lm_var	79,81%±0,37%	20,19%±0,37%	07,53%±0,51%	00,16%±0,37%	92,47%±0,51%	08,86%±1,37%
gn_var_cholmod	80,36%±0,33%	19,64%±0,33%	08,41%±0,47%	00,58%±0,92%	91,59%±0,47%	08,39%±0,52%
lm_var_cholmod	79,81%±0,37%	20,19%±0,37%	07,53%±0,51%	00,16%±0,37%	92,47%±0,51%	08,86%±1,37%
gn_var_eigen	80,36%±0,33%	19,64%±0,33%	08,41%±0,47%	00,58%±0,92%	91,59%±0,47%	08,39%±0,52%
lm_var_eigen	79,81%±0,37%	20,19%±0,37%	07,53%±0,51%	00,16%±0,37%	92,47%±0,51%	08,86%±1,37%

TABLE VII: Performance of the random_orientation initialisation method for different linear solvers.

method	mean_pose	mean_weighted	random_ori
gn_pcg	11,3us±0,61us	11,3us±0,69us	16,7us±0,90us
lm_pcg	11,2us±0,67us	11,2us±0,69us	16,8us±1,36us
gn_var	11,4us±0,98us	11,5us±1,29us	16,8us±1,45us
lm_var	11,4us±1,00us	11,3us±1,07us	17,0us±1,53us
gn_var_cholmod	11,4us±0,91us	11,3us±0,93us	16,8us±1,16us
lm_var_cholmod	11,4us±0,91us	11,4us±1,04us	17,1us±1,63us
gn_var_eigen	11,3us±0,75us	11,2us±0,71us	16,7us±0,74us
lm_var_eigen	10,9us±0,83us	10,9us±0,71us	16,1us±1,03us
avg	11,3us	11,3us	16,8us

TABLE VIII: Timing results of the graph optimisation for the different initialisation methods and linear solvers.