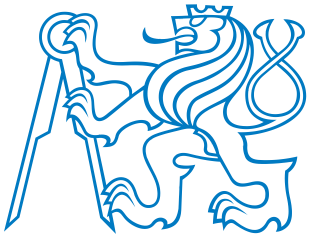

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Control Engineering

Michal Jirků

Advanced localization methods for ground robots using WiFi signals

Bachelor thesis

2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Jirků Michal**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Bachelor Project: **Advanced localization methods for ground robots using WiFi signals.**

Guidelines:

The aim is to design, implement and experimentally evaluate a system for localization of mobile robotic platform, developed as part of the TRADR project (http://www.tradr-project.eu/), with rich sensory equipment: LIDAR, IMU, odometer and WiFi receiver. Localization system should primarily use the WiFi signal available in the environment, processed by machine learning techniques (Gaussian processes are recommended). Trajectory should be considered in 3D, a 6-DOF, and to its estimation, it is recommended to use the Monte Carlo Localization (MCL) or Kalman filter. Implementation for the robotic platform should be realized using combination of Python (ROS framework) and MATLAB.

Bibliography/Sources:

[1] B. Ferris, D. Haehnel, and D. Fox, Gaussian processes for signal strength-based location estimation,in In proc. of robotics science and systems. Citeseer, 2006.
[2] Y. Sun, M. Liu, and M.-H. Meng, WiFi signal strength-based robot indoor localization, in Information and Automation (ICIA), 2014 IEEE International Conference on, 2014, pp. 250-256.
[3] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann, GPPS: A gaussian process positioning system for cellular networks, in NIPS, 2003.
[4] F. Duvallet and A. Tews, WiFi position estimation in industrial environments using Gaussian processes, in Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, 2008, pp. 2216-2221.

Bachelor Project Supervisor: Ing. Michal Reinštein, Ph.D.

Valid until the summer semester 2017/2018

L.S.

prof. Ing. Michael Šebek, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 30, 2017

Acknowledgments

I would like to thank my bachelor thesis supervisor Ing. Michal Reinštein, Ph.D. for advices and support during the work and Vladimír Kubelka for his introduction to the problem, initial guidance with running up the system and his help with collecting the dataset.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 25. May 2017

ABSTRACT

This bachelor's thesis focuses on the design, implementation and experimental evaluation of a localization system for mobile robotic platform, developed as part of the TRADR project (http://www.tradr-project.eu/), with rich sensory equipment: LIDAR, IMU, odometer and WiFi receiver. The localization system primarily uses the WiFi signal available in the environment, processed by machine learning techniques (Gaussian processes), since high WiFi coverage provides a way to localize even low-cost robots that do not need to be equipped with excessive computational power to run visual-based SLAM algorithms or expensive exteroceptive sensors. Trajectory is considered in 3D, a 6-DOF, and to its estimation, it uses Gaussian processes followed by Monte Carlo Localization. The proposed application is to use one SLAM robot to map WiFi signal strength in the working area and provide it to a low-cost robot to correct drift of its 6-DOF gyro-odometry localization system. Implementation of this localization system is realized using MATLAB.

KEYWORDS

WiFi localization, WiFi signal strength, Gaussian processes, Particle filter, 6-DOF gyroodometry localization system, mobile robots

ABSTRAKT

Cílem této práce je návrh, implementace a experimentální ověření lokalizačního systému pro robotickou platformu, která je vyvíjena jako součást projektu TRADR (http://www.tradr-project.eu/) a je vybavena širokou škálou senzorů jako je LIDAR, IMU, odometr a WiFi příjmač. Lokalizační systém předně využívá dostupný WiFi signál, který je zpracován pomocí strojového učení (Gaussovské procesy). Díky tomu, že je dnes většina budov pokryta hustou WiFi sítí, je možné využít sílu WiFi signálu pro spolehlivou lokalizaci levných robotů, kteří tak nemusí mít excesivní výpočetní výkon pro provádění SLAM algoritmů a nemusí být vybaveni drahými exteroceptivními senzory. Trajektorie je uvažována ve 3D, 6 stupňů volnosti a k jejímu určení jsou využity Gaussovské procesy následované Monte Carlo lokalizací. Uvažovaná aplikace systému je taková, že jeden SLAM robot zmapuje prostředí a sílu WiFi signálu v něm. Tato data budou zpracována a poskytnuta lévnému robotovi, který díky tomu bude moci opravovat výchylku svého gyro-odometrického lokalizačního systému se šesti stupni volnosti. Implementace navrhnutého lokalizačního systému je provedena v Matlabu.

Klíčová slova

WiFi lokalizace, síla WiFi signálu, Gaussovské procesy, částicový filtr, gyroodometrický lokalizační systém, mobilní roboti

Contents

List of abbreviations:

AP	Access Point
DOF	Degrees Of Freedom
GP	Gaussian Processes
IMU	Inertial Measurement Unit
LP	Logarithmic Probability
MAC	Media Access Control
PF	Particle Filter
RMSE	Root Mean Square Error
RSSI	Received Signal Strength Indication
SE	Squared Exponential
SLAM	Simultaneous Localization And Mapping

1 Introduction

The bachelor thesis builds on author's publication in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)[1]. The thesis focuses on reliable localization in dynamic environment, where nowadays only robots equipped with expensive exteroceptive sensors and excessive computational power to run visual-based simultaneous localization and mapping (SLAM) algorithms succeed. The motivation is to provide also low-cost robots, which are popular and commonly used these days, with reliable localization system, enabling them to perform tasks in large and dynamic environment such as warehouses, airports or factories using only their gyro-odometry and WiFi receiver.

Almost every building is covered by multitude of WiFi networks. The idea is to use this already built infrastructure for localization purposes, reducing the hardware requirements only to a WiFi receiver. In this thesis one branch of WiFi localization methods based on WiFi signal mapping that uses Gaussian processes (GP) is adopted. It is further combined with a 6-Degrees of freedom (DOF) gyro-odometry for continuous localization in complex 3D environment. Two different approaches are implemented and tested. One approach estimates robot's position by selecting the place with highest probability (HP). The second approach implements particle filter (PF) and estimates the position as the mean of resampled particles.

At the beginning state of the art approaches in this field are presented, then they are compared with our approach. In the next section is an overview on our localization system design given. Moreover advanced localization methods used in this thesis are introduced there.

After presenting the theoretical background comes the practical part. The first section of this part describes how the trajectories and WiFi signal strengths for our experiments are acquired. It further presents our testing environment and dataset.

The next section deals with our implementation and is divided into two parts. One part explains the data preprocessing and WiFi signal strength model creation in the offline phase. The second part presents our approach to odometry correction.

The last practical section deals with experimental setting and the results of our two different position estimation approaches. Each approach was experimentally evaluated with and without yaw correction. The experimental results are summarized afterwards.

2 Related work

This section introduces localization approaches suggested by various research papers and compares them with our approach.

2.1 State of the art

In general two major methods for localization based on wireless networks are proposed in the literature. These are:

- Signal propagation modeling methods.
- Signal propagation mapping methods.

The first approach [2, 3, 4] requires the position of a radio transmitter (typically WiFi access point), walls and other obstacles to be known or estimated. Using these information a signal distribution model for the environment can be created. But the real not modeled effects such as signal attenuation, reflection or dispersion are the cause for lower accuracy of localization. This approach does not require direct collection of training data in the environment to create a signal distribution map.

On the other hand the signal mapping approach needs spatially localized radio signal strength training set. Using this training set the signal distribution map can be approximated. There is no need to have prior knowledge on the position of access points (AP), obstacles and effects on the path between the radio transmitter and receiver in order to create the map [1]. Moreover one approach is presented, which handles possible minor changes in the signal levels in the environment [5].

Some research papers, which focus on localization techniques using signal map, explicitly build a graph, where the recorded samples belong to the vertices of the graph and subsequent localization is done by interpolating between those vertices [6].

There are also few approaches which focus on classification of new radio signal strength measurements. One part uses the classified samples to generate a set of new artificial learning measurements in order to increase the localization accuracy [7]. The other part uses the classified samples to obtain a rough estimate of location further using other localization techniques[8, 9].

The character of measured radio signal strength is similar to Gaussian distribution. Therefore many approaches [10, 11, 12, 13, 14] use the framework of Gaussian processes (GP) as it is suitable for non-parametric modeling of the signal distribution over the environment. It further

allows to create probabilistic model of the environment and omit location labels. In several papers dealing with mobile robot localization are GP often followed by Bayes particle filter (PF) [12, 13, 11, 6]. This allows to incorporate a-priory knowledge from previous sequence of odometry and WiFi measurements [1].

2.2 Comparison

In this thesis is the signal mapping approach adopted. It uses GP to model the WiFi signal propagation. For localization is GP followed by particle filter (PF) used to incorporate a-priori knowledge from previous sequence of WiFi and odometry measurements as [12, 13, 11, 6]. Moreover we combine it with a 6-DOF gyro-odometry. This allows mobile robots to continuously localize themselves. With this approach the localization is possible even in a complex 3D environment or outdoor, under the assumption that in the mapped area is sufficient WiFi coverage. Compared to a classical 3-DOF problem, additional degrees of freedom allow continuous localization which can spread over multiple floors and capture movement in full 3D. We mainly focus on the problem of drifting position and yaw angle, and propose an option how to correct it.

To our best knowledge, no localization technique was yet published in context of mobile robot localization, which extends the classical 3-DOF problem to 6-DOF and uses localization based on WiFi signal strength. We found only [8, 11] which focus on 3D person localization in graph-based map using WiFi reciever. We show that extending the 3-DOF problem to the 6-DOF problem is possible.

For the space representation we use rather continuous over graph based representation [6], because it is more suitable for the character of the 3D mobile robot localization. Experimental verification was done in environment with presence of people and on different times of the day and floors to mimic real application scenarios.

3 Methodology

This section gives an overview on the general workflow of our localization system and further presents advanced methods used this work.

3.1 Workflow

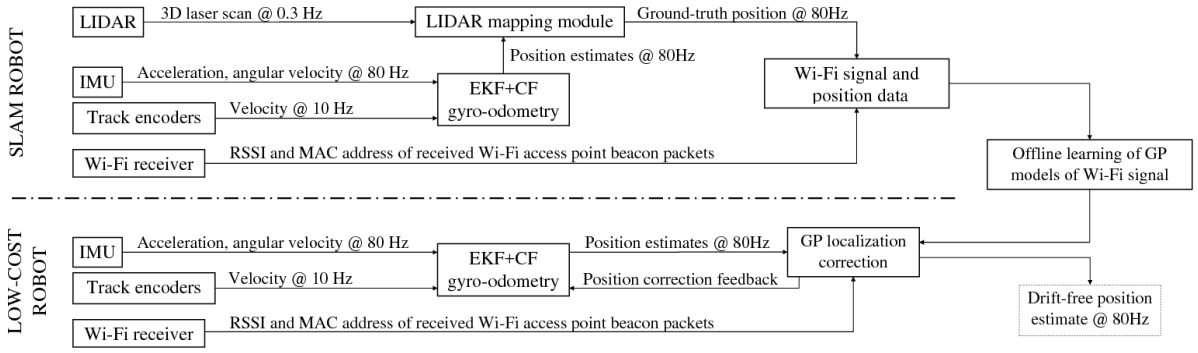

Figure 1: The localization system diagram is divided into SLAM robot part and low-cost robot part. The SLAM robot precisely maps the environment and the mapping data are used to create GP model of WiFi signal. Then a low cost-robot equipped only with gyro-odometry, WiFi receiver and WiFi signal model can be applied with drift-free position estimation [1].

Our workflow for Wi-Fi signal based robot localization is schematically shown in Fig. 1. In general it consist of one SLAM robot capable of reliable localization and a low-cost robot with gyro-odometry that is prone to drift[1]. In order to estimate drift free position of the low-cost robot, these 3 stages need to be completed:

1. Wi-Fi signal strength mapping with SLAM robot
2. Offline Gaussian process learning
3. Low-cost robot Wi-Fi localization

In the first stage the SLAM robot records surrounding Wi-Fi signal levels from access points and performs SLAM to reliably localize itself in the environment. Then in the second stage the acquired Wi-Fi signal levels from access points are paired with the corresponding positions. These data are then used for learning Gaussian process models for each access point (recognized by its physical MAC address). The last stage utilizes results of the two previous ones; the basic gyro-odometry (prone to drift) is corrected by position inferred from the GP models (based on signal strength of received Wi-Fi packets). The result is that the low-cost robot is capable of reliable localization.

[1]For the purpose of this theses the same robotic platform was used as SLAM robot, to obtain reliable position estimates, as well as low-cost robot, recording odometry data for localization system evaluation.

Boxes in Fig. 1 have the following meaning:

- **LIDAR**: Our robotic platform on Fig. 4 is equipped with a laser range finder (SICK LMS-151) that is rotated by a servo mechanism and creates 3D map of the environment at rate of 0.3 Hz.
- **EKF + CF gyro-odometry**: EKF means Extended Kalman Filter and CF means complementary filter. it processes the velocity vector from track encoders. Further EKF uses complementary filter and processes the IMU measurements to probabilistically estimate the position [15, 16].
- **IMU**: The inertial measurement unit of the robot (Xsens MTi-G) measures at 80 Hz the acceleration and angular velocity which is later utilized by the EKF + CF gyro-odometry.
- **Track encoders** are sensing the velocity at 10 Hz. They are the source of velocity vector further utilized by the EKF + CF gyro-odometry.
- **WiFi receiver**: 2.4 GHz WiFi adapter is attached to the robot and sniffs beacon packets in monitor mode. The MAC address and the RSSI of the packet is further used to create WiFi signal and position data.
- **LIDAR snapping module** utilizes the data from LIDAR and EKF+CF gyro-odometry. It creates the groundtruth with the temporal resolution of EKF + CF gyro-odometry and spatial accuracy resulting from LIDAR.
- **WiFi signal and position data**: This data are created by merging received WiFi signal packets RSSI and MAC address with the corresponding position.
- **GP localization correction**: Implementation of our approach to correct odometry drift using WiFi signal strength and GP models of WiFi signal.
- **Offline learning of GP models of WiFi signal**: The training phase where GP model of WiFi signal is created from WiFi signal and position data.
- **Drift-free position estimate**: Desired result is to have a drift free position with the temporal resolution of IMU, using only measurements from IMU, track encoders, WiFi receiver and GP models of WiFi signal.

3.2 Gaussian processes

A Gaussian process is a collection of random variables, any finite number of which have a consistent joint Gaussian distribution [17]. It defines probability distribution over functions.

GP are suitable for WiFi signal based localization, as they are non-parametric, continuous and provide uncertainty estimation for predictions at any given location [12, 18]. The standard GP regression model assumes that dataset $D := \{\mathbf{X} := [\mathbf{x}_1,...,\mathbf{x}_n]^{\mathrm{T}}, \mathbf{y} := [y_1,...,y_n]^{\mathrm{T}}\}$ was generated so that $y_i = h(\mathbf{x}_i) + \epsilon_i$, where $h : \mathbb{R}^D \to \mathbb{R}$ is a random function, $\epsilon_i \sim N(0,\sigma_n^2)$ is independent Gaussian measurement noise, $\mathbf{x}_i$ is in our case the position and y_i the corresponding signal strength. Gaussian distribution is specified by a mean vector and a covariance matrix. GP is similar to the Gaussian distribution as it is fully specified by a mean function $m_h(\cdot)$ and a covariance function [19]

$$k_h(\mathbf{x},\mathbf{x}') := \mathrm{cov}_h[h(\mathbf{x}),h(\mathbf{x}')], \mathbf{x},\mathbf{x}' \in \mathbb{R}^D \tag{1}$$

specifying the covariance between any two function values. The covariance function $k_h(\cdot,\cdot)$ is in terms of GP called kernel. In connection with WiFi signal strength modeling a prior mean function $m_h = 0$ and squared exponential (SE) kernel determined as [19]

$$k_{SE}(\mathbf{x}_p,\mathbf{x}_q) := \alpha^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^{\mathrm{T}} \mathbf{\Lambda} (\mathbf{x}_p - \mathbf{x}_q)), \mathbf{x}_p, \mathbf{x}_q \in \mathbb{R}^D \tag{2}$$

are used. Matrix $\mathbf{\Lambda}$ has diagonal elements set to the respective length scale so that $\mathbf{\Lambda} = \mathrm{diag}([\ell_1^2,...,\ell_D^2])$. The parameters α^2 and ℓ are called hyperparameters, where α^2 is the signal variance and $\ell_i,...,\ell_D$ are the characteristic length scales. As the WiFi signal strength y is a noisy observation of $f(\mathbf{x})$, it is necessary to add noise covariance function $\sigma_n^2\delta_{pq}$ in the covariance matrix resulting in [12]

$$k_h = k_{SE} + \sigma_n^2\delta_{pq} \tag{3}$$

where $\delta_{pq} = 1$ if $p = q$, else $\delta_{pq} = 0$. For entire set of values $\boldsymbol{X}$, covariance matrix can be rewritten as $K_h = K_{SE} + \sigma_n^2\boldsymbol{I}$ where K_{SE} is a covariance matrix resulting from k_{SE}. The collection of hyperparameters is denoted as vector $\boldsymbol{\theta}$ and contains α^2,$\boldsymbol{\ell}$ and σ_n^2. This set of hyperparameters denotes the smoothness of the functions estimated by GP. They are learned from training data by maximizing the log marginal likelihood of the observations conditioned on the hyperparameters. Learning of the hyperparameters can be completed offline after the training dataset was collected [12, 18].

The posterior distribution can be then generated over functions for arbitrary points $\mathbf{x}_\star$ given the training data $\boldsymbol{X}$ and $\mathbf{y}$ as [12]

$$p(f(\mathbf{x}_\star)|\mathbf{x}_\star,\boldsymbol{X},\mathbf{y}) \sim N(\mu_{\mathbf{x}_\star},\sigma^2_{\mathbf{x}_\star}) \tag{4}$$

where the prediction means and variance are [12]:

$$\begin{aligned} \mu_{\mathbf{x}_\star} &= \mathbf{k}_\star^{\mathrm{T}}(K+\sigma_n^2 I)^{-1}\mathbf{y} \\ \sigma^2_{\mathbf{x}_\star} &= k_{SE}(\mathbf{x}_\star,\mathbf{x}_\star) - \mathbf{k}_\star^{\mathrm{T}}(K+\sigma_n^2 I)^{-1}\mathbf{k}_\star \end{aligned} \tag{5}$$

where $\mathbf{k}_\star$ is a vector of covariances between $\mathbf{x}_\star$ and n training inputs [12, 18].

Example of simplified use of GP can be seen on Fig.2 and 3. On Fig. 2 is a dataset with samples determined by the function $y = f(x)+\sigma^2$, where $f(x)$ is a function that is going to be estimated by GP and σ^2 is noise. On Fig. 3 is the estimation of the function $f(x)$ using hyperparameters learned by minimizing the negative log marginal likelihood of the corresponding dataset and using squared exponential kernel. Using this dataset and learned parameters the probability of a function value $f(x_\star)$ at any arbitrary position $x_\star$ as in equation 4 can be obtained [20].

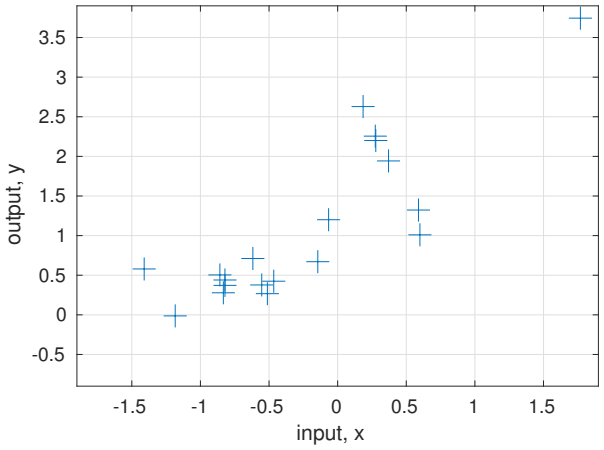

Figure 2: One dimensional example dataset with 20 samples where $y = f(x) + \sigma^2$, $f(x)$ is an unknown function and σ^2 is noise [20].

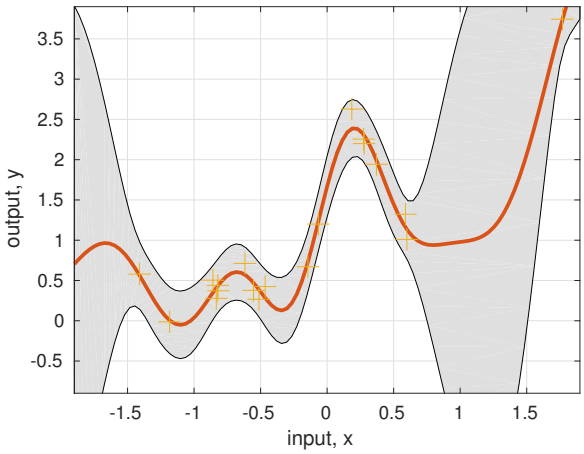

Figure 3: GP approximation of the unknown function $f(x)$ from Fig.2 with hyperparameters learned by minimizing the negative log marginal likelihood of the dataset corresponding to $f(x)$ using squared exponential covariance function. The solid orange line represent the mean function, the shaded areas represent the 95% confidence intervals of the GP distribution [20].

3.3 Particle filter

The particle filter is a nonparametric Bayesian filtering technique, which is able to handle any arbitrary probability density function. It approximates the posterior by a finite number of parameters. The main idea of this approach is to represent the posterior believe $bel(x_t)$ by a set of random state samples drawn from this posterior. The samples of posterior distribution are called particles and are in the following form [21]:

$$\chi_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{6}$$

where M is a number of particles in set χ_t and $x_t^{[m]}$ $(1 \leq m \leq M)$ is a particle with concrete instantiation of the state at time t [21]. The beliefs $bel(x_t)$ are constructed recursively from the beliefs $bel(x_{t-1})$ occurring one time step earlier.

Simple particle filter algorithm is shown bellow on in Alg. 1 [21]. The algorithm takes as input the previous belief χ_{t-1}, an actuation command u_t and data received from sensors z_t. The output is the new belief χ_t at current time t. At line 4 occurs the motion update where the

Algorithm 1 Simple particle filter algorithm [21]

```
1:  procedure PARTICLE_FILTER(χ_{t-1}, u_t, z_t)
2:      χ̄_t = χ_t = ∅
3:      for m = 1 to M do
4:          sample x_t^[m] ~ p(x_t | u_t, x_{t-1}^[m])          ▷ Motion update
5:          w_t^[m] = p(z_t | x_t^[m])                          ▷ Importance factor (weight) update
6:          χ̄_t = χ̄_t + ⟨x_t^[m], w_t^[m]⟩
7:      end for
8:      for m = 1 to M do                                       ▷ Resampling step
9:          draw i with probability α w_t^[i]
10:         add x_t^[i] to χ_t
11:     end for
12:     return χ_t
13: end procedure
```

particles are moved according to the robot's movement. At line 5 the weight of each particle is updated based on the data received from sensor. In our case the sensor data are the log likelihoods acquired from the WiFi signal map given the particle position and RSSI. So the weights are updated by the exponentiation of our log likelihoods [21].

On line 8-11 occurs the resampling step. In this step it is iterated over particles from set $\overline{\chi}_t$ and particles having probability over certain level are chosen and added to the set χ_t. In the resampling phase is often the low variance algorithm used in order to select the particles. Our approach uses this algorithm too [21].

The basic idea of low variance algorithm (Alg. 2 [21]) is that it rather involves sequential stochastic process than selecting the samples independently of each other. The algorithm takes as its input the particle set χ_t and the the corresponding weights W_t of the particles in the set. It chooses random number r as on line 3 and selects particle by repeatedly adding the fixed amount M^{-1} to r and by choosing particle that corresponds to the resulting number [21].

Algorithm 2 Low variance sampling algorithm [21]

1: **procedure** LOW_VARINACE_RS(χ_t, W_t)
2: $\overline{\chi}_t = \emptyset$
3: $r =$rand$(0; M^{-1})$ ▷ Choose random number r from the given interval
4: $c = w_t^{[1]}$
5: $i = 1$
6: **for** $m = 1$ to M **do**
7: $u = r + (m-1) \cdot M^{-1}$ ▷ Add fixed M^{-1} to r
8: **while** $u > c$ **do**
9: $i = i + 1$
10: $c = c + w_t^{[i]}$
11: **end while**
12: add $x_t^{[i]}$ to $\overline{\chi}_t$
13: **end for**
14: **return** $\overline{\chi}_t$
15: **end procedure**

4 Data collection

This section presents the data collection phase where the SLAM robot collects data about nearby APs. The robot also records its trajectory, which we consider as the real trajectory that is used as reference or so called ground truth. As our robotic platform serves as both SLAM robot and low-cost robot, drift prone gyro-odometry trajectory is also recorded at this phase.

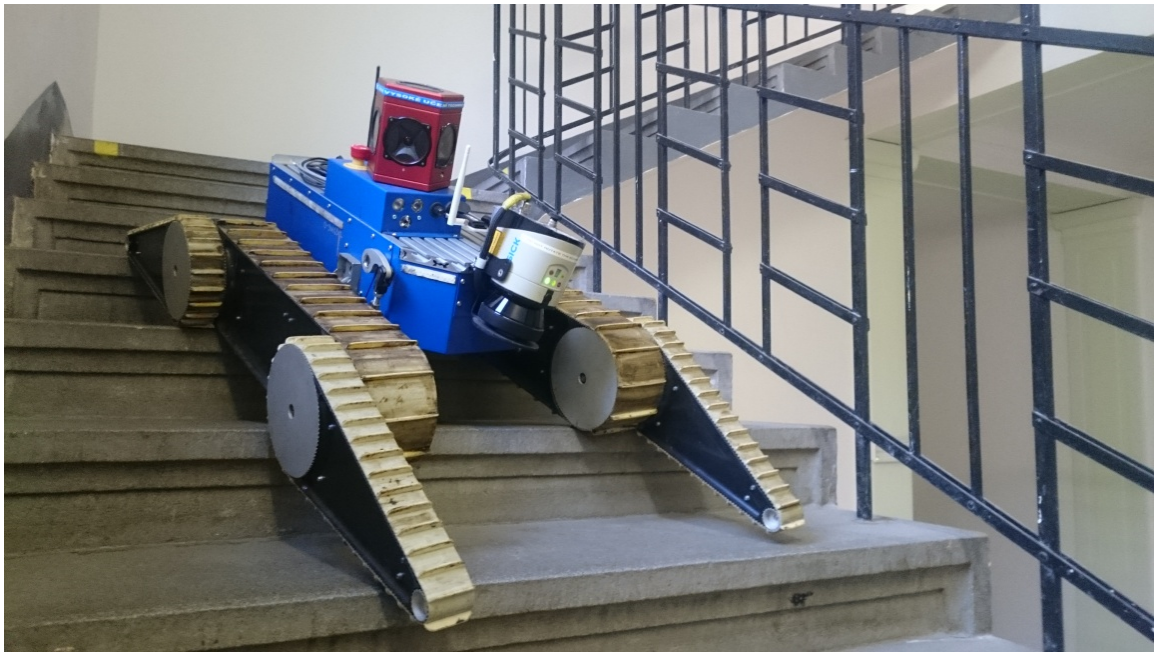

Figure 4: Our skid-steered search and rescue robotic platform that was used to record data both in the role of a SLAM robot as well as a low-cost robot. The robot is developed as part of the *TRADR* project[1] and is equipped with a laser range-finder used for 3D SLAM [1].

4.1 Localization data collection

For robots it is important to know their position, therefore they use odometry to keep track of their movement. As the position derived from odometry is actually the position one time step earlier incremented by a known movement over elapsed time and course (known as dead reckoning), it is subject to cumulative errors [22]. The error is integrated over time and it is necessary to periodically correct it by other sources of position estimation.

Our mobile robot (Fig. 4) is equipped with a laser range-finder (SICK LMS-151). This laser range-finder is rotated by a servo mechanism to create 3D scans with relatively low frequency around 0.3 Hz. To obtain reliable position estimates, the scans are processed by the *ethz-asl/libpointmatcher* [23] SLAM algorithm. Since the position estimation using SLAM algorithm

[1] `http://www.tradr-project.eu`

depends on the 3D scans arriving every 3 seconds, the gyro-odometry sampled at 80 Hz is used to support it. The position estimation from SLAM algorithm interpolated with gyro-odometry is used to localize received WiFi-packets.

4.2 WiFi signal acquisition

To record WiFi packets, a 2.4GHz Wi-Fi adapter switched to the *monitor* mode is used. This mode allows to inspect all packets the adapter receives.

Since the adapter can receive and transmit only on one channel at given time, it is necessary to iterate over WLAN channels and sniff beacon frames. By selecting only these frames, traffic transmitted by client devices is ignored.

Beacon frames are periodically transmitted by public APs to announce the presence of WLAN. These frames contain all the necessary information about the AP from which they were sent. For localization purpose is only the MAC address of the AP and the signal strength (RSSI) at given time and position needed.

As 2.4 GHz Wi-Fi adapter is used, it is iterated over all 14 WLAN channels of that frequency range in order to capture beacon frames from APs. These channels are spaced by 5 MHz except the last 14th channel which is spaced by 12 MHz.

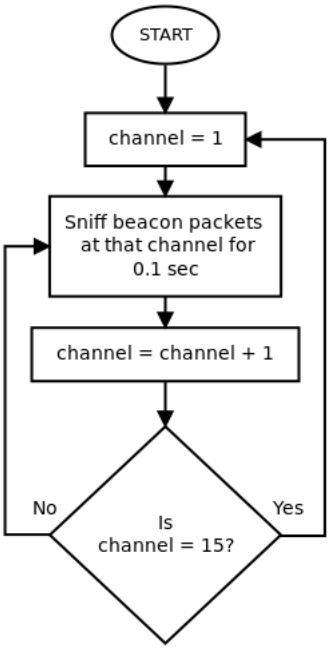

Figure 5: The flowchart shows sniffing of WiFi beacon packets. Sniffer starts at channel 1 and sniffs for 0.1 on that channel, then the channel is switched and the process is repeated.

In practice is the WiFi adapter set to monitor mode and it is iterated over the 14 channels of

2.4GHz frequency range while fetching the incoming packets as in Fig. 5. Every packet is checked if it is a 802.11 packet and further of type Management Frame and subtype Beacon Frame. If the packet has desired attributes specified above, then the MAC address of the originating AP and the RSSI is extracted. This is then saved with the current time stamp into a .csv file.

4.3 Dataset description

Testing and training data were collected on different dates spanning nine months. The data were collected also on different times of the day spanning from noon until late afternoon hours in order to cover the variations in appearance of the environment.

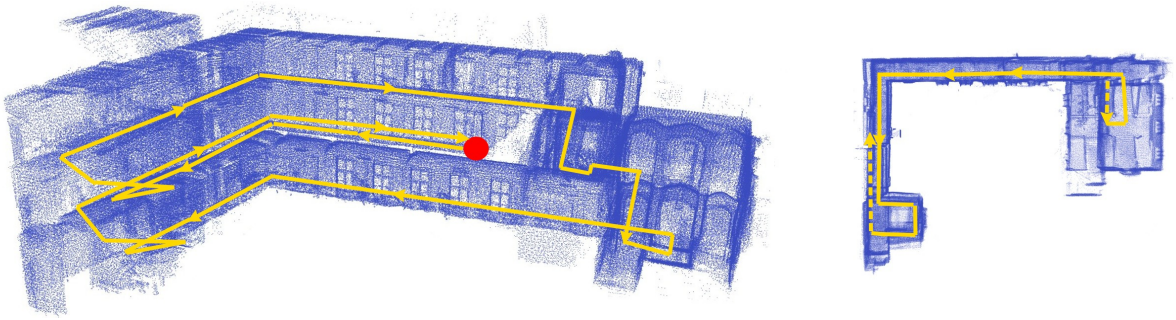

Figure 6: Output of the *ethz-asl/libpointmatcher* [23] SLAM algorithm showing the three floor testing environment of CTU department building. Red circle is the starting position of the testing data in the dataset. Yellow lines denote one of the testing trajectory from the dataset. On the right top view image is only the trajectory on the bottom floor showed. [1]

Experiments were performed with presence of people on three floors of the CTU campus building on Karlovo náměstí in Prague (Fig. 6) to mimic real uncontrolled environment.

Table 1: Dataset overview

Dataset overview					
ID	**Date & time**	**Changed axes**	**Covered floor**	**Lenght**	**Note**
20151	21.10.2015 14:22	Yes	1, 0	133 m	
20152	21.10.2015 15:03	Yes	1, 0	133 m	
20161	6.1.2016 14:48	Yes	0, -1	95 m	floor 0 only partially covered
20162	6.1.2016 15:08	No	1, 0, -1	155 m	floor -1 only partially covered
20163	6.1.2016 15:46	No	1, 0, -1	254 m	
20164	6.1.2016 16:12	Yes	1, 0	75 m	
201621	8.2.2016 12:31	No	1, 0,-1	265 m	
201622	8.2.2016 12:54	No	1, 0,-1	281 m	
201631	25.2.2016 17:33	No	0, -1	147 m	
201632	25.2.2016 17:50	No	1, 0	133 m	
201633	25.2.2016 18:03	No	1, 0, -1	261 m	
201671	25.7.2016 12:59	No	0+	179 m	
201672	25.7.2016 13:25	No	0+	178 m	

Tab.1 summarizes our dataset for GP map learning and experimental evaluation. Changed axis column means that the start of the collection was performed in different direction and for evaluation we need to invert the x and y axes. The covered floor specifies which floors were covered in the data collection. The starting floor is 0, the upper floor is 1, bottom floor is -1 and 0+ stands for the starting floor covering some extra area on that floor. In the length column is the length of the ground truth trajectory for which we have the corresponding WiFi signals.

5 Implementation

In this section is the implementation of 3D WiFi based localization discussed. This section consists of two subsections dealing with offline training phase and correction phase. The repository with source codes and dataset for this project is publicly available here [24].

5.1 Offline phase

In the offline phase are the recorded data paired and the ground truth is checked and corrected if needed. As the ground truth data are recorded at much slower rate than gyro-odometry, it is necessary to extend it to the same length in order to use in the next step, where the WiFi map for each AP is created using GP.

5.1.1 Data preprocessing

While preprocessing the recorded data, it is necessary to check if the outcoming ground truth trajectory is approximately the same as the real trajectory. This is done in order to use it for pairing with WiFi packets and in order to compute the Root Mean Square Error (RMSE) later in the experiments. If there is a drift in roll, pitch or yaw angle, then it needs to be manually corrected so that it refers to the real trajectory. After manual correction the ground truth can be used to pair corresponding WiFi packet with position.

As every measurement of the ground truth is recorded every 3 seconds (0.3 Hz), it is necessary to extend the its temporal resolution to match the length of the gyro-odometry (80 Hz) in order to use it for WiFi signal map learning see Fig. 7. For the extension is linear interpolation between every two subsequent ground truth (0.3 Hz) records used [25]. The extended ground truth (80 Hz) will be further called just ground truth.

Before pairing the ground truth position with WiFi packets, MAC addresses which occur less than 10 times are removed from the corresponding WiFilog file. The threshold of 10 measurements from one MAC address is set based on observations of several WiFi signal maps created by different number of measurements, where those containing less than 10 measurements were more likely to be highly imprecise and were decreasing the localization accuracy. Every WiFi packet is paired with the corresponding ground truth position in a way that the resulting position is the one whose time stamp is nearest to the packet time stamp as in Fig. 8.

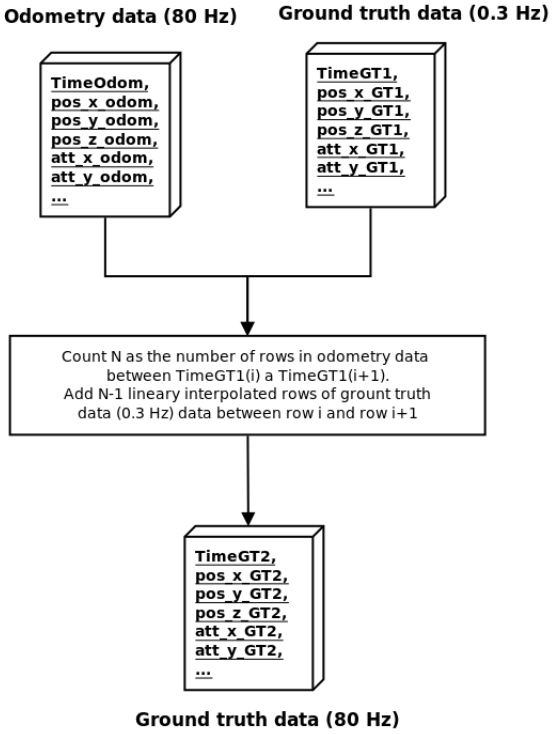

Figure 7: Workflow where ground truth (0.3 Hz) and odometry data (80 Hz) are processed to create ground truth with temporal resolution of 80 Hz. Data files contain time stamp, positions on x, y and z axes and attitudes.

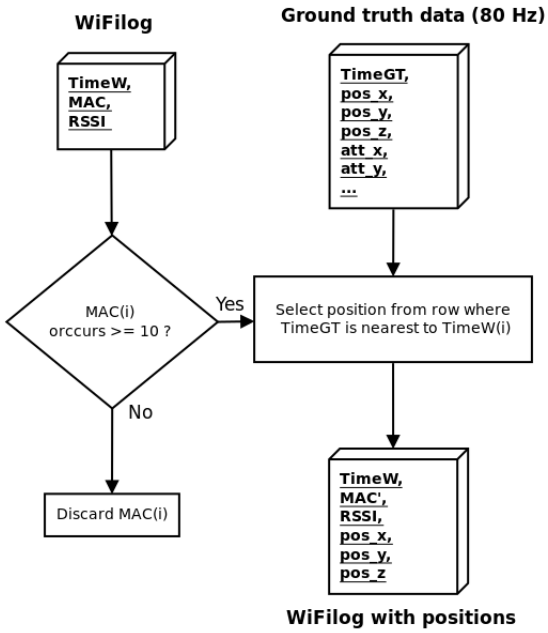

Figure 8: This workflow shows pairing of WiFi packets with position. Packets with MAC address which occurs less than 10 times in the WiFilog are discarded. The remaining packets containing time stamp (TimeW), MAC address and RSSI are paired with the ground truth position according to the nearest ground truth time stamp. The result is a WiFilog with time stamp TimeW, MAC address and RSSI extended with x, y and z position for every packet. The ground truth contains time stamp (TimeGT), positions on x, y and z axes and attitudes.

5.1.2 WiFi map creation

Before creating the WiFi signal map, all unique MAC addresses of the corresponding WiFilog with position file are found and concatenated with their position and RSSI. Then it is iterated over each unique MAC address and the WiFi signal map parameters are trained [20]. Zero mean prior, squared exponential kernel and Gaussian likelihood are chosen to train the hyper-parameters of the GP WiFi model. The resulting structure contains unique MAC addresses, their corresponding position, RSSI and hyper-parameters. The resulting signal distribution map of two APs in the CTU department building can be seen on Fig. 9

In order to make more generalized WiFi signal maps, more wifilogs with position are merged together and used as the input for the GP training algorithm.

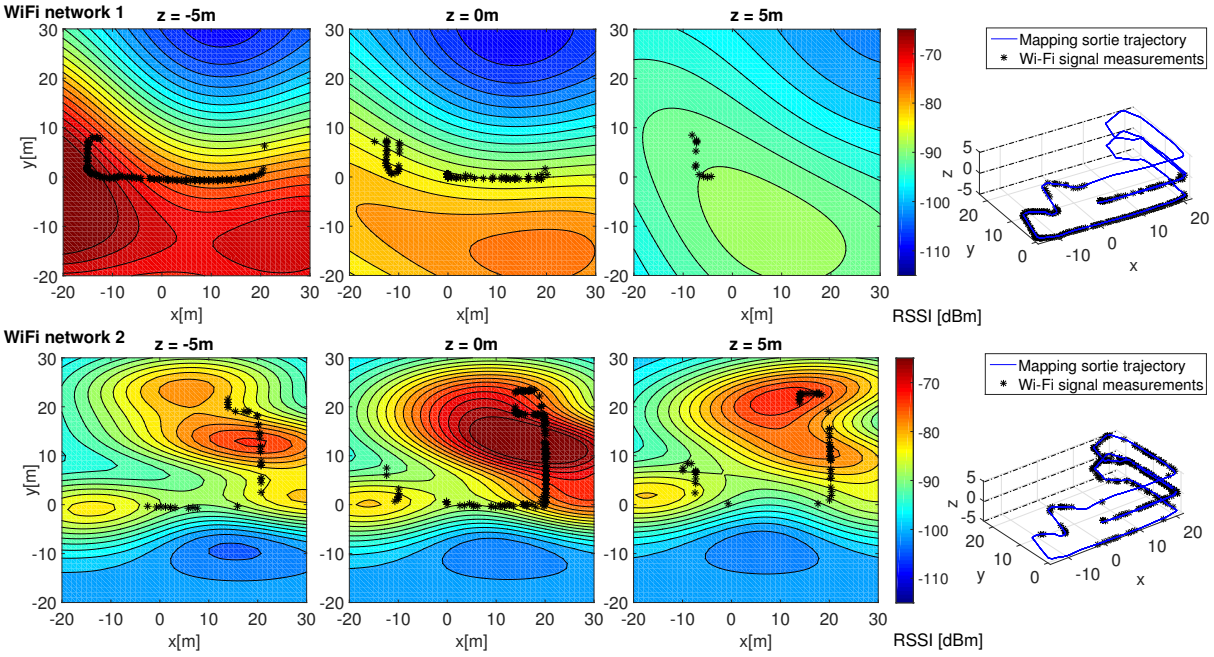

Figure 9: Cuts through WiFi signal strength maps modeled by GP for two different APs on the three floors of CTU department building. Red regions show areas of high WiFi signal strength and the potential position of the AP. Black stars show the measurement locations used to learn the models [1].

5.2 Odometry correction

Proposed scenario consist of low-cost robot equipped with gyro-odometry localization system and WiFi adapter. The goal is to combine these two sources of information with created WiFi signal map and improve the localization accuracy of the robot. As stated before, the main focus is on the correction of the gyro-odometry drift, therefore the evaluation starts always at the same position at (0,0,0) coordinates and in the same direction. First a general overview on the odometry correction process is given and then is the implemented method described in more detail.

The localization consists of multiple steps. The first one is WiFi packet buffering, which runs until N packets are reached. Then the position probability distribution using our WiFi map is estimated [20] and model constrains with prior position are added. There are two approaches implemented to extract the position from the probability distribution. One approach is naive and selects the position with highest probability. Second approach is more sophisticated and uses particle filter and selects the mean position of resampled particles. We wanted to test our assumption if estimating the position over more particles would lead to better accuracy and be more resistant to biases than the first naive approach.

After estimating the position, the current position is corrected by the estimated position. The

new position after each correction is buffered and after reaching M points, it is used for yaw correction.

A simplified correction workflow scheme is on Fig. 10.

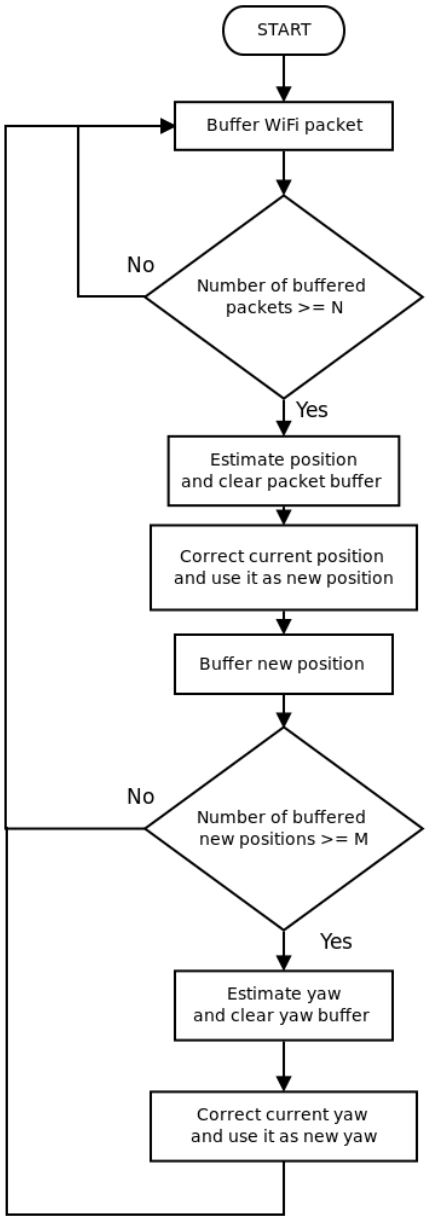

Figure 10: Odometry correction workflow scheme.

Odometry correction and position estimation require around 10 parameters to be set. Finding the optimal parameters, which would work best over multiple testing data, is very time consuming process as every simulation takes around 3 to 6 minutes. Most of the parameters are estimated from several observations and cannot be considered as optimal.

5.2.1 Position estimation

For each fetched packet the learned WiFi signal maps are used to compute the logarithmic probability (LP) distribution over signal strength in the environment. As resulting LP distribution might be biased, typically more packets are fetched and their LP distributions are added. LP is in range from $\langle -\infty, 0 \rangle$ and has a useful attribute that combining multiple probabilities can be done by summation instead of multiplication. The addition of LP distributions can be seen in Fig. 11, where the series under **a)** display LP distribution of a single packet on the floors -1,0,1 (the difference between floors is 5 m). Series **b)** show the distribution after adding the LP distributions of 10 consecutive packets on the same floors as in a). Series **c)** show the same but from the summation of 20 LP distributions of 20 consecutive packets. Note that the more packets are added the more the LP peak at position (10, 0, 0) is distinguishable. This corresponds to the position of the mobile robot that can be seen on the trajectory plot as a red cross.

Since the prior knowledge of the robot's position is provided, it is sufficient to use Bayes filter which reuses prior knowledge. The combination of Bayes filter with the LP distribution derived from fetched packets creates the posterior probability. As it can be assumed that the robot trajectory can only lie within the mapped space, a prior constraint LP distribution is added. The process can be seen on Fig. 12. Series **a)** show the prior constraint map, where the region within the mapped space has 0 log probability and the rest goes to $-\infty$. Series **b)** show used Bayes filter, which creates a Gaussian like LP distribution over the prior robot position. The variance of the Gaussian is set the same for all 3 dimensions by diagonal matrix $\mathbf{\Sigma}$. Series **c)** show the LP distribution resulting from summation of 20 LP distributions from the fetched packets. The last series **d)** show the summation of LP distributions from a), b) and c). The position of the robot is marked as a red cross on the trajectory.

In one approach an equidistant grid of points around the prior position is used as the set of arbitrary points $\mathbf{X}_\star$. The log probability at the position of each point in this grid is computed given the RSSI signal $f(\mathbf{x}_\star)$, training set $\boldsymbol{X}, \boldsymbol{y}$ and the hyper-parameters $\boldsymbol{\theta}$ [20]. This goes through process displayed in Fig. 12. Then the point with highest probability is selected as the estimated position. New position $\boldsymbol{x}_{new}$ is then computed from the estimated position $\boldsymbol{x}_{estimated}$ and current position $\boldsymbol{x}_{current}$, so that $\boldsymbol{x}_{new} = k \cdot \boldsymbol{x}_{estimated} + (1 - k) \cdot \boldsymbol{x}_{current}$, where k is a correction constant.

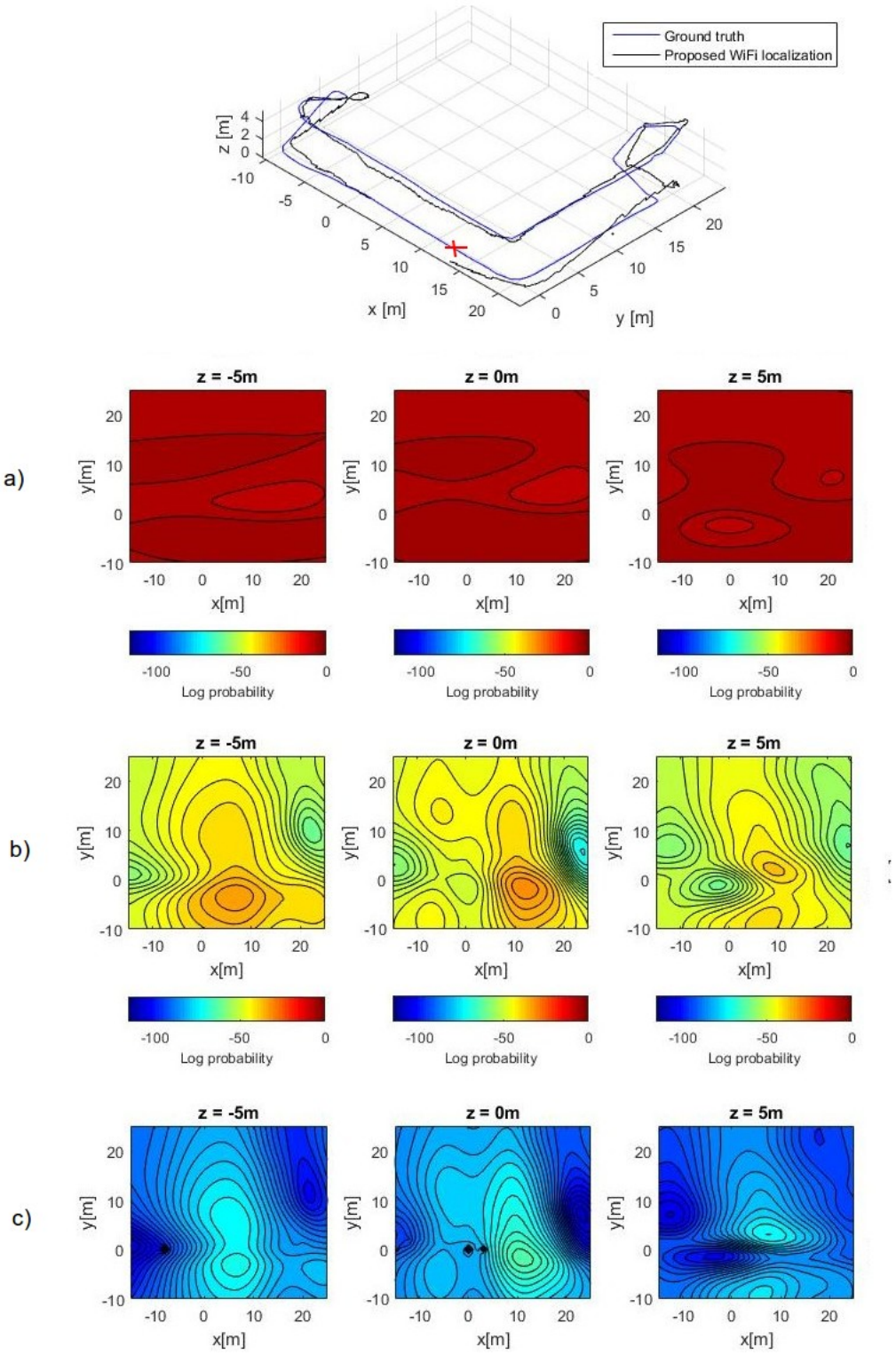

Figure 11: Log probabilities maps on the three floors of the CTU building. a) show the log probabilities given one packet, b) show log probabilities resulting from 10 consecutive packets, c) show log probabilities resulting from summation of 20 consecutive packets. The position of the robot can be seen on the trajectory and is marked as a red cross.

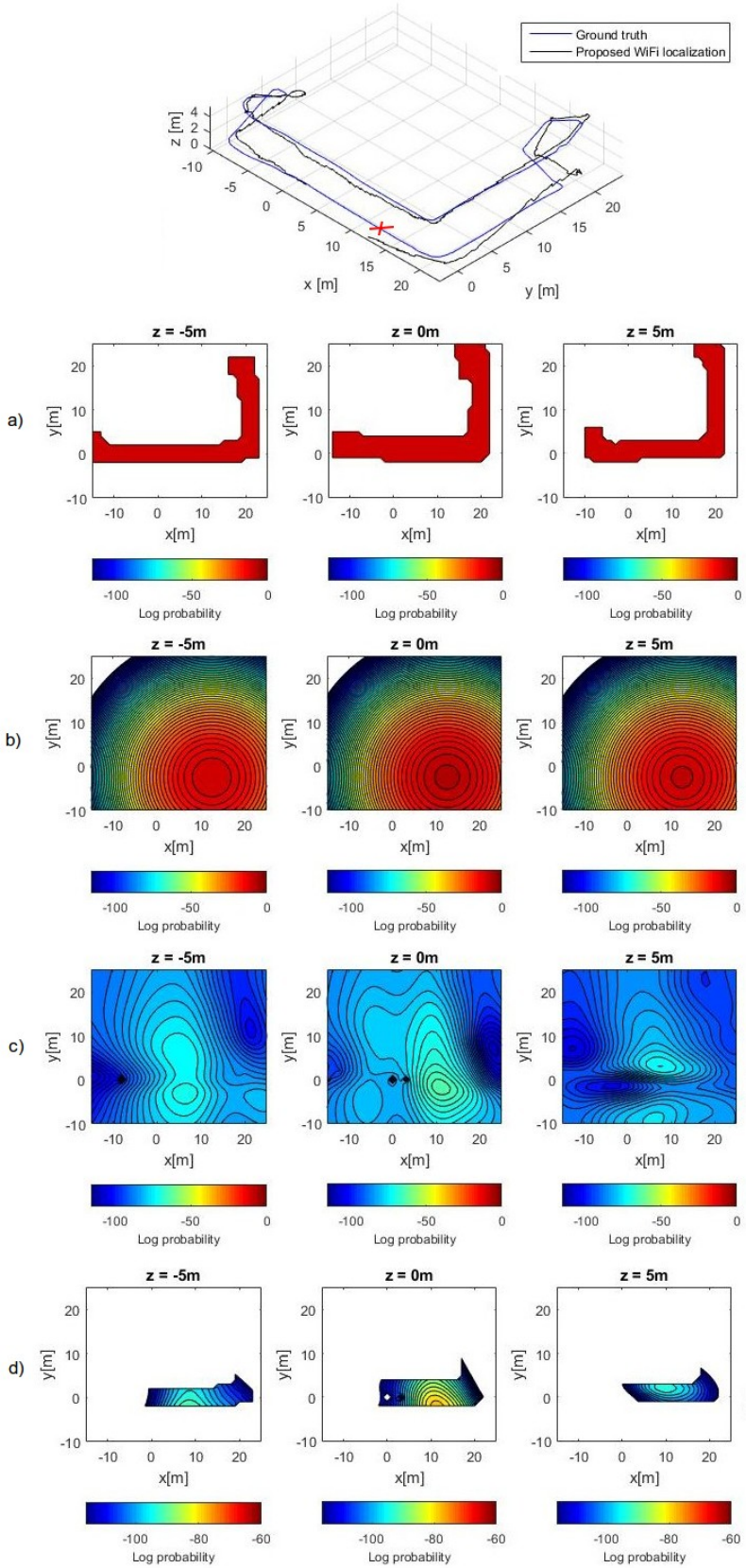

Figure 12: Log probability maps on the three floors of the CTU building. **a)** show the log probabilities of our model constraints, **b)** show Gaussian probability distributions over the prior robot position, **c)** show the probability map from Fig.11 c) and **d)** show the resulting summation of the three maps above. The position of the robot is marked on the trajectory as a red cross.

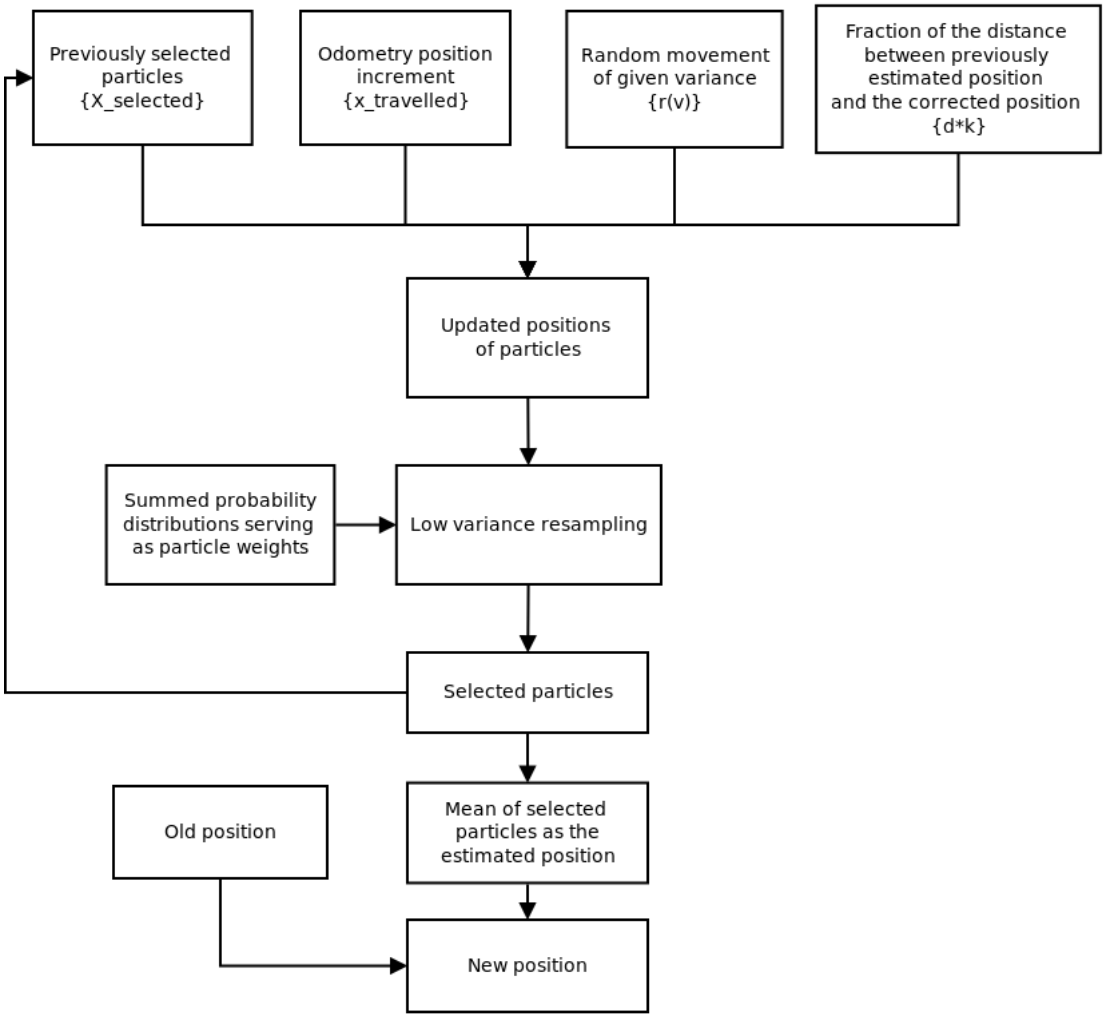

Figure 13: Estimating position using GP and particle filter

Second approach uses particle filter to estimate the position, so instead of an equidistant grid, the positions of the particles are used as arbitrary points $\mathbf{X}_\star$. The same process as in Fig. 12 occurs to obtain the LP for each particle. The exponentiated LP is used as the weight for the particles. Low variance argorithm (Alg. 2) is used to resample the particles in the resampling phase. Then the mean of the resampled particles is used as the estimated position. New position is corrected in the same way as in the previous approach.

Moreover the update of each particle by the actuation command u_t is computed as follows:

$$\mathbf{X}_{updated} = \mathbf{X}_{resampled} + \mathbf{x}_{travelled} + \mathbf{r(v)} + \mathbf{d} * k \tag{7}$$

where $\mathbf{X}$ is a matrix containing positions of particles, $\mathbf{x}_{travelled}$ distance traveled from last estimated position, $\mathbf{r(v)}$ is a random number of given variance $\mathbf{v}$ on x, y and z axes and $\mathbf{d}$ is the difference between estimated position and the current position weighted with a constant

k. As the particle cloud may drift away on some places, the term $\mathbf{d}$ is used to move the particles back to the actual position. The actuation command itself is then $u_t = \mathbf{x}_{travelled} + \mathbf{r(v)} + \mathbf{d} * k$. The workflow for estimating position using particle filter is on Fig.13 .

5.2.2 Yaw estimation

For 6-DOF localization it is necessary to consider not only the position of the robot, but also the attitude. Since roll and pitch angles are observable due to gravitational force, only correction for yaw angle is needed (see Fig. 14). We propose to correct yaw angle drift using buffered new positions at trajectory segments, where the robot moves in a straight line.

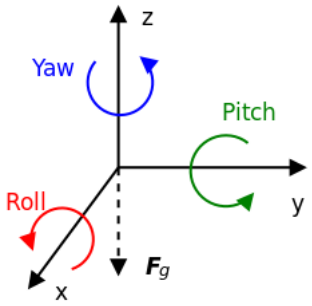

Figure 14: Coordinate system with x, y, z axes with their corresponding euler angles roll, pitch and yaw. Dashed vector $\mathbf{F}_g$ shows the direction of gravitational force.

After estimating and buffering enough new positions $\{\boldsymbol{x}_{k-d},...,\boldsymbol{x}_k\}$, the difference between the oldest $\boldsymbol{x}_{k-d}$ and the newest $\boldsymbol{x}_k$ buffered position is checked to assure that the robot moved some minimal distance.

The vector pointing in the robot heading is then computed from the set $\{\boldsymbol{x}_{k-d},...,\boldsymbol{x}_k\}$ by fitting a line through the points using the least square method and only the x and y position. The resulting line is described as [26]:

$$y = a + bx \tag{8}$$

and parameters a and b are determined as [26]

$$\begin{aligned} a &= \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n\sum x^2 - (\sum x)^2} \\ b &= \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2} \end{aligned} \tag{9}$$

where n is the number of buffered positions.

To assign at which end of the line is the robot heading, the vector from $\boldsymbol{x}_{k-d}$ to $\boldsymbol{x}_k$ is used. Estimated yaw angle can be computed using the *atan2* function.

$$yaw_k = atan2(y_2 - y_1, x_2 - x_1) \tag{10}$$

where y_2,y_1,x_2,x_1 are coordinates of two points lying on the fitting line respecting the direction of the robot motion [1].

The estimated yaw is further checked with the previous new yaw to allow only corrections in a straight line up to some tolerance set empirically to $\pm\frac{\pi}{5}$ rad. This also avoids correcting the yaw by too big yaw estimations, which could be potential source of error. The current yaw is then corrected by an amount of the estimated yaw and the resulting new yaw is further used. The correction amount was determined by observing multiple values and choosing the one, which performed with the lowest Root mean square error (RMSE) in position over multiple testing data.

6 Experimental evaluation

Several experiments were done to show differences between proposed localization approaches and evaluate their performance experimentally. Five WiFi signal maps were created and tested on data which were different then those on which the maps were learned. Maps and their corresponding training data can be seen on Tab. 2.

Table 2: Maps used for experimental evaluation

Map name	**Data used for GP modeling by ID (see Tab. 1)**	**Total length [m]**	**Number of training samples**
MAP 1	201622	281	15149
MAP 2	20163	254	8443
MAP 3	201633	261	17627
MAP 4	201622, 20163	535	23592
MAP 5	201622, 20163, 201633	796	41219

We performed three WiFi mapping sorties of $796\,m$ of total length with almost one month difference in between experiments. We also compared impact of using data from single mapping sortie, data from two merged sorties and data from three merged sorties.

To compare performance of the bare gyro-odometry to the one estimated and corrected by WiFi, the *Root-Mean-Square Error* (*RMSE*) is used:

$$RMSE(k) = \sqrt{\frac{\sum_{i=1}^{k}(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 + (\hat{z}_i - z_i)^2}{k}} \tag{11}$$

where x, y and z are estimated coordinates, k is discrete time and $\hat{x}_i$, $\hat{y}_i$ and $\hat{z}_i$ is the ground truth position.

The performance of the gyro-odometry to estimated and corrected yaw is also calculated using the RMSE:

$$RMSE_{yaw}(k) = \sqrt{\frac{\sum_{i=1}^{k}(y\hat{a}w_i - yaw_i)^2}{k}} \tag{12}$$

where yaw is the estimated yaw, k is discrete time and $y\hat{a}w_i$ is the reference yaw.

There are four experiments made with two different position estimation approaches, each with and without yaw correction. The structure of each experiment is:

1. Definition of the parameters for the experiment and comments.

2. Table of results for RMSE in position for the given approach.

3. Experiments with yaw correction have table of results for RMSE in yaw for the given

approach.

4. One testing trajectory corrected by the given approach. To make it comparable, there are always the same testing data with the same WiFi signal map showed in all approaches. Top view of the trajectory is chosen in order to clearly show the performance of the yaw correction.

5. RMSE in position graph for the testing trajectory above.

6. Experiments with yaw correction have RMSE in yaw graph for the testing trajectory above.

The localization was implemented in *MATLAB*. **X** value in table means that the corresponding test data were used as the training data for the given WiFi signal map and therefore are excluded. Fail cases of our localization system are in red.

6.1 Highest probability

This Wi-Fi localization approach estimates the position by selecting the position with highest probability. 20 Wi-Fi packets are fetched (that yields cca. $0.5\,m$ traveled by the robot, standard gyro-odometry is running and localizing during this period) before computing the localization correction.

Other parameters for the experiment are: variance on the Σ diagonal is $\sigma^2 = 6\,m^2$, position correction smoothing weight is set to $0.045 \cdot \boldsymbol{x}_k^* + (1-0.045) \cdot \boldsymbol{x}_{odom,k}$. We sample the probability $p(\boldsymbol{x}_k|\boldsymbol{s}_{1:k})$ by samples $1\,m$ equidistant in the x,y dimension up to $8\,m$ far from the robot and $0.2\,m$ equidistant in the z dimension up to $2.5\,m$ far from the robot. The parameters were derived from observations of the behavior depending on the parameter change.

It is assumed that initial position of the robot is known, otherwise sampling the whole map for initialization would be necessary.

Table 3: RMS Error in position [m] using HP approach. **X** means that the corresponding test data were used as the training data for the given WiFi signal map. Fail cases of our localization system are in red.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	4.4	1.6	1.7	2.3	1.5	1.7
20152	3.7	1.2	1.5	1.9	1.4	1.5
20162	6.1	3	2.5	2.9	2.7	3
20163	8	4.7	**X**	4.7	**X**	**X**
201621	7.6	2.7	3.2	2.6	2.5	2.7
201622	9.4	**X**	5.3	4.2	**X**	**X**
201631	7.1	2	3	1.9	2.4	1.9
201632	3.6	1.8	2.5	1.2	1.7	1.4
201633	6.9	2	2.7	**X**	2	**X**

In Tab. 3 are 4 localization fail cases in red. They occurred in the same area on the bottom floor. This might mean that the area is poorly covered by WiFi signal or the WiFi signal model has some biases in there.

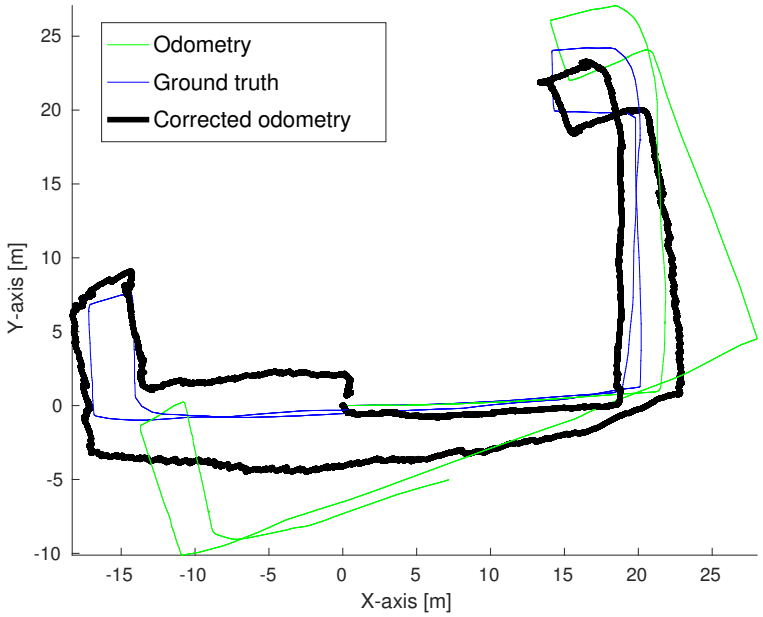

Figure 15: Top view on a successful localization experiment in data 201631 using MAP 5 and HP approach.

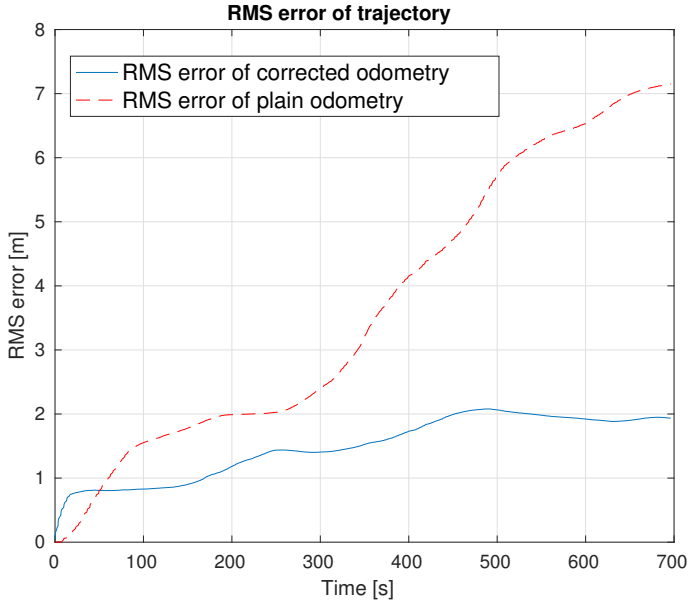

Figure 16: RMSE in position of a successful localization experiment in data 201631 using MAP 5 and HP approach.

6.2 Highest probability and yaw correction

This approach applies our proposed yaw correction from section 5.2.2. It is expected that the yaw correction will make the corrected odometry stick to the ground truth. This should then yield smaller RMSE in the corrected position. The experiment parameters are the same as in the previous experiment, but some new connected to yaw correction are added. Yaw correction weight is set to $0.23 \cdot yaw_k + (1 - 0.23) \cdot yaw_{odom,k}$ and yaw angle is estimated from 4 previous position estimations. The yaw correction weight was determined by observing multiple values and choosing the one, which performed with the lowest RMSE in position over multiple testing data.

Table 4: RMS Error in position [m] using HP approach and yaw correction. **X** means that the corresponding test data were used as the training data for the given WiFi signal map. Fail cases of our localization system are in red.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	4.4	1.8	1.5	2.3	1.6	1.7
20152	3.7	1.5	1.9	1.9	1.8	2.4
20162	6.1	2.3	2.1	3.1	2.6	2
20163	8	3.1	**X**	5.6	**X**	**X**
201621	7.6	3	3.1	3.6	2.6	3
201622	9.4	**X**	4.3	3.1	**X**	**X**
201631	7.1	1.5	2.9	2	1.6	1.4
201632	3.6	2.4	2.8	2	2.3	1.7
201633	6.9	2.6	4.4	**X**	3	**X**

Table 5: RMS Error in yaw [rad] using HP approach and yaw correction. **X** that the corresponding test data were used as the training data for the given WiFi signal map.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	0.16	0.23	0.11	0.18	0.13	0.16
20152	0.13	0.16	0.17	0.17	0.21	0.24
20162	0.22	0.24	0.2	0.33	0.22	0.2
20163	0.28	0.14	**X**	0.4	**X**	**X**
201621	0.33	0.27	0.21	0.3	0.25	0.26
201622	0.41	**X**	0.28	0.28	**X**	**X**
201631	0.26	0.22	0.21	0.34	0.12	0.17
201632	0.18	0.22	0.14	0.3	0.21	0.2
201633	0.21	0.26	0.3	**X**	0.4	**X**

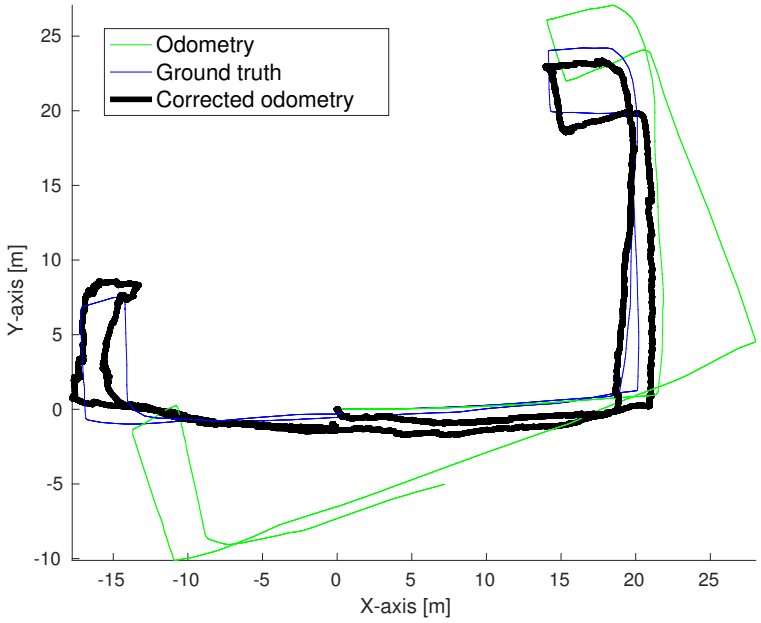

Figure 17: Top view on a successful localization experiment in data 201631 using MAP 5 and HP approach with yaw correction.

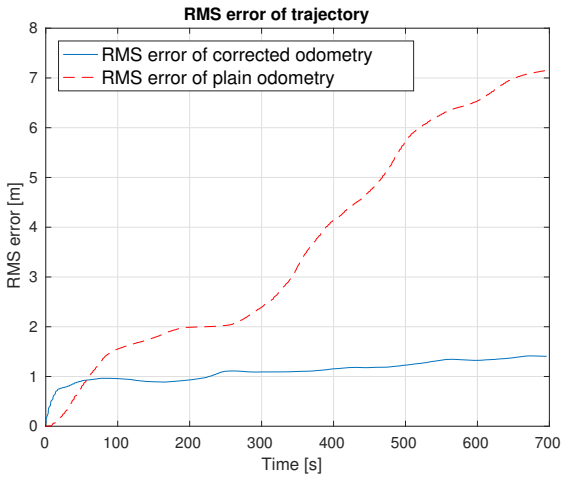

Figure 18: RMSE in position of a successful localization experiment in data 201631 using MAP 5 and HP approach with yaw correction.

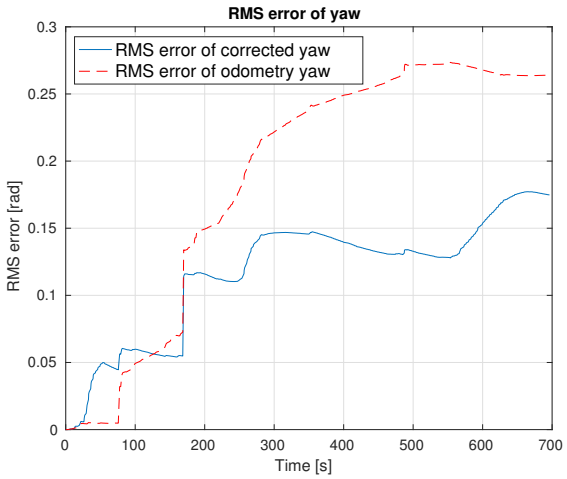

Figure 19: RMSE in yaw angle of a successful localization experiment in data 201631 using MAP 5 and HP approach with yaw correction.

6.3 Particle filter

In this approach is the estimated position selected as the mean of the resampled particles. 10 WiFi packets are fetched before computing the localization correction (that yields cca. $0.25\,m$ traveled by the robot). This is the only difference in the parameter setting as it showed to achieve better results, than using 20 WiFi packets as in the HP approach.

The particle filter uses 1000 particles, the update variance for random number $r(v)$ on x, y and z axis is $v = [1, 1, 0.1]$ m, update correction constant k is set to 0.15. The variance vector was determined based on our robot movement capabilities. Correction constant k was estimated from observations. Other parameters are same as in HP approach.

It is expected, that the particle filter will be more robust to WiFi signal strength biases and the estimation of the position over several particles will be more accurate.

Table 6: RMS Error in position [m] using PF approach. **X** means that the corresponding test data were used as the training data for the given WiFi signal map. Fail cases of our localization system are in red.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	4.4	1.3	1.5	2	1.3	1.6
20152	3.7	1.2	1.7	2	1.2	1.4
20162	6.1	2.7	2.2	3.2	2.4	2.6
20163	8	15.9	**X**	7.5	**X**	**X**
201621	7.6	2.2	2.8	2.4	2	2.3
201622	9.4	**X**	4.1	3.5	**X**	**X**
201631	7.1	1.6	2.5	1.6	1.8	1.6
201632	3.6	1.7	2.4	1.2	1.7	1.2
201633	6.9	1.9	2.6	**X**	1.9	**X**

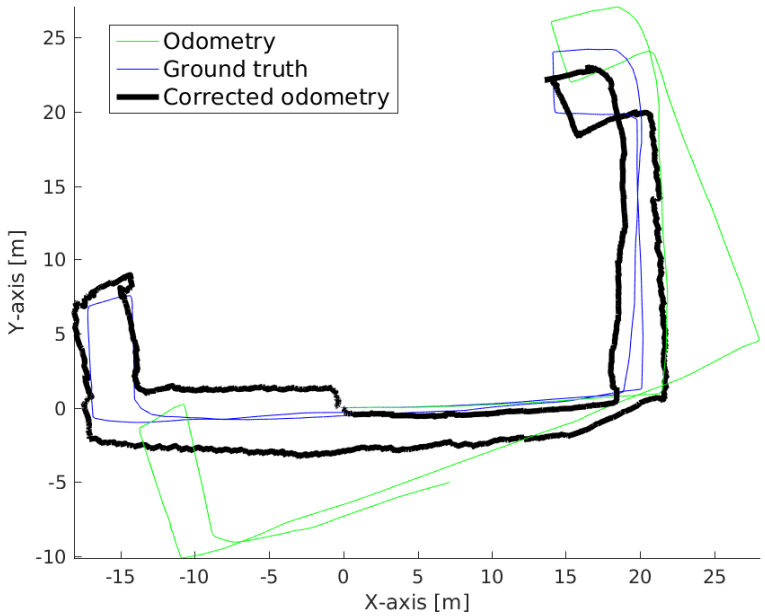

Figure 20: Top view on a successful localization experiment in data 201631 using MAP 5 and PF approach.

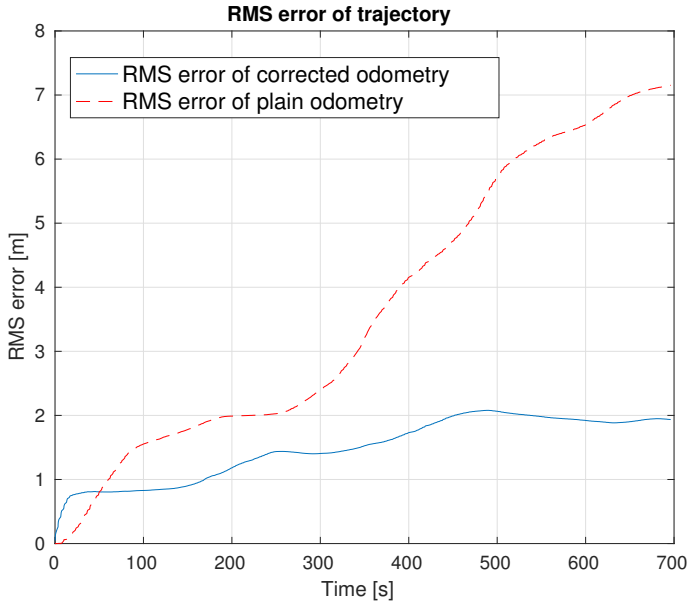

Figure 21: RMSE in position of a successful localization experiment in data 201631 using MAP 5 and HP approach.

6.4 Particle filter and yaw correction

In this experiment the PF approach for position estimation is used together with yaw correction. Again the same parameters for PF as in the previous experiment are used. The general experimental parameters are the same as the HP with yaw correction approach.

Table 7: RMS Error in position [m] using PF approach and yaw correction. **X** means that the corresponding test data were used as the training data for the given WiFi signal map. Fail cases of our localization system are in red.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	4.4	2	1.7	2.3	1.7	2.4
20152	3.7	1.6	1.7	2	1.4	1.6
20162	6.1	3.6	1.8	2.5	2.5	2.6
20163	8	8.8	**X**	4.1	**X**	**X**
201621	7.6	2	2	2.3	1.9	2
201622	9.4	**X**	2.6	2.7	**X**	**X**
201631	7.1	1.5	2.7	1.2	1.5	1.3
201632	3.6	2.1	3.3	1.6	2.1	1.3
201633	6.9	2.1	12.2	**X**	2.2	**X**

Table 8: RMS Error in yaw [rad] using PF approach and yaw correction. **X** value in table means that the corresponding test data were used as the training data for the given WiFi signal map.

ID	Odometry	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	0.16	0.23	0.27	0.24	0.28	0.29
20152	0.13	0.17	0.15	0.17	0.2	0.22
20162	0.22	0.26	0.16	0.23	0.23	0.17
20163	0.28	0.4	**X**	0.28	**X**	**X**
201621	0.33	0.24	0.21	0.25	0.22	0.23
201622	0.41	**X**	0.34	0.3	**X**	**X**
201631	0.26	0.17	0.38	0.14	0.19	0.16
201632	0.18	0.22	0.23	0.22	0.24	0.18
201633	0.21	0.26	0.43	**X**	0.26	**X**

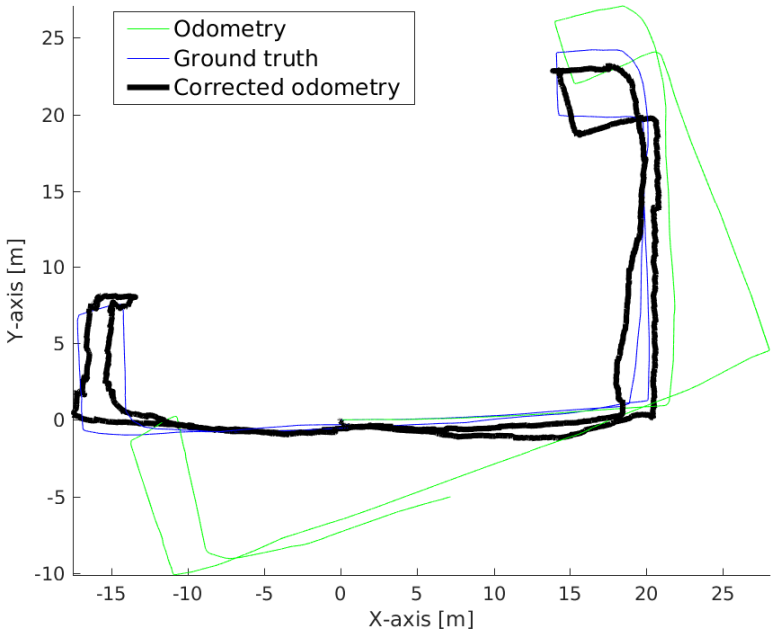

Figure 22: Top view on a successful localization experiment in data 201631 using MAP 5 and PF approach with yaw correction.

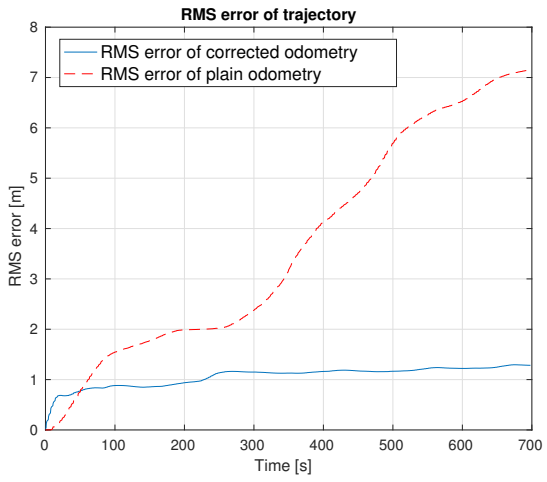

Figure 23: RMSE in position of a successful localization experiment in data 201631 using MAP 5 and PF approach with yaw correction.

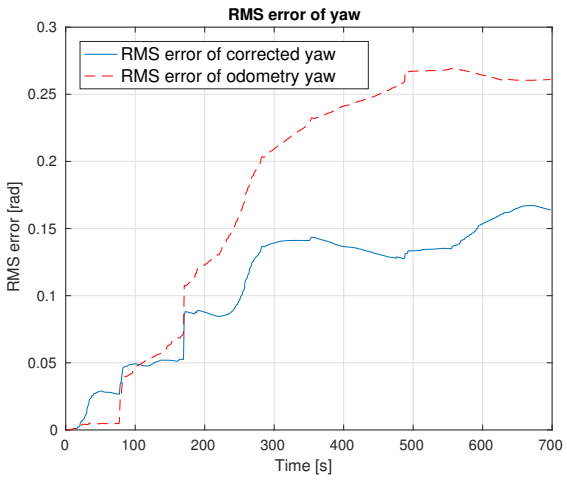

Figure 24: RMSE in yaw angle of a successful localization experiment in data 201631 using MAP 5 and PF approach with yaw correction.

6.5 Experiment summary

We performed 148 trials and the localization system was successful in 138 cases. Overall every approach was able to correct drift of the gyro-odometry in at least 89% of the trials. Every approach achieved smaller RMSE in position in all non fail cases. To make the RMSE results comparable, the RMSE in position is divided by the length of the ground truth for given testing trajectory, thus achieving the average RMSE per meter. This is further averaged over all non fail trials of the given approach. The result are in Tab. 9.

Table 9: Average RMSE in position per meter for different approaches

	Average RMSE in position per meter
Plain odometry	0.0329 m
HP	0.0131 m
HP with yaw correction	0.0138 m
PF	0.0119 m
HP with yaw correction	0.0127 m

In general the HP approach with yaw correction was less prone to fail than without yaw correction. Considering all successful testing cases, PF approach yields better results in terms of RMSE in position (Tab. 10). It has also smaller computational requirements, as the log probability is computed only over 1000 particles instead of an equidistant grid space around the last position of the robot in HP approach consisting of over 7500 points.

Table 10: RMS Error difference in position [m] between PF approach with yaw correction and HP approach with yaw correction (PF RMSE - HP RMSE). Negative value denotes smaller RMSE for PF approach. Results where the localization failed were not considered.

ID	**MAP 1**	**MAP 2**	**MAP 3**	**MAP 4**	**MAP 5**
20151	0.2	0.2	0	0.1	0.7
20152	0.1	-0.2	0.1	-0.4	-0.8
20162	1.3	-0.3	-0.6	-0.1	0.6
20163	**X**	**X**	**X**	**X**	**X**
201621	-1	-1.1	-1.3	-0.7	-1
201622	**X**	-1.7	-0.4	**X**	**X**
201631	0	-0.2	-0.8	-0.1	-0.1
201632	-0.3	0.5	-0.4	-0.2	-0.4
201633	-0.5	**X**	**X**	-0.8	**X**
Mean	-0.03	-0.4	-0.49	-0.21	-0.17

Moreover both approaches without yaw correction yield better results than with (Tab. 11 and 12). But it is not so significant and mainly caused by the first two testing data with ID 20151 and 20152.

In the example testing trajectory showed in each experiment, it can be clearly seen that the

Table 11: RMS Error difference in position [m] between HP approach with yaw correction and HP approach without yaw correction (HP yaw RMSE - HP RMSE). Negative value denotes smaller RMSE for HP with yaw correction approach. Results where the localization failed were not considered.

ID	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	0.2	-0.2	0	0.1	0
20152	0.3	0.4	0	0.4	0.9
20162	-0.7	-0.4	0.2	-0.1	-1
20163	**X**	**X**	**X**	**X**	**X**
201621	0.3	-0.1	1	0.1	0.3
201622	**X**	**X**	**X**	**X**	**X**
201631	-0.5	-0.1	0.1	-0.8	-0.5
201632	0.6	0.3	0.8	-0.6	0.3
201633	0.6	**X**	**X**	1	**X**
Mean	0.11	-0.02	0.35	0.19	0

Table 12: RMS Error difference in position [m] between PF approach with yaw correction and PF approach without yaw correction (PF yaw RMSE - PF RMSE). Negative value denotes smaller RMSE for PF with yaw correction approach. Results where the localization failed were not considered.

ID	MAP 1	MAP 2	MAP 3	MAP 4	MAP 5
20151	0.7	0.2	0.3	0.4	0.8
20152	0.4	0	0	0.2	0.2
20162	0.9	-0.4	-0.7	0.1	0
20163	**X**	**X**	**X**	**X**	**X**
201621	-0.2	-0.8	-0.1	-0.1	-0.3
201622	**X**	-1.5	-0.8	**X**	**X**
201631	-0.1	0.2	-0.4	-0.3	-0.3
201632	0.4	0.9	0.4	0.4	0.1
201633	0.2	**X**	**X**	0.3	**X**
Mean	0.33	-0.2	-0.19	0.14	0.08

yaw correction works fine and sticks the corrected trajectory to the ground truth. The RMSE for different approaches in this example can be seen on Fig. 25. The RMSE in yaw decreases in HP approach from 0.26 rad to 0.17 rad and in PF approach from 0.26 rad to 0.16 rad. This can be seen on Fig. 26

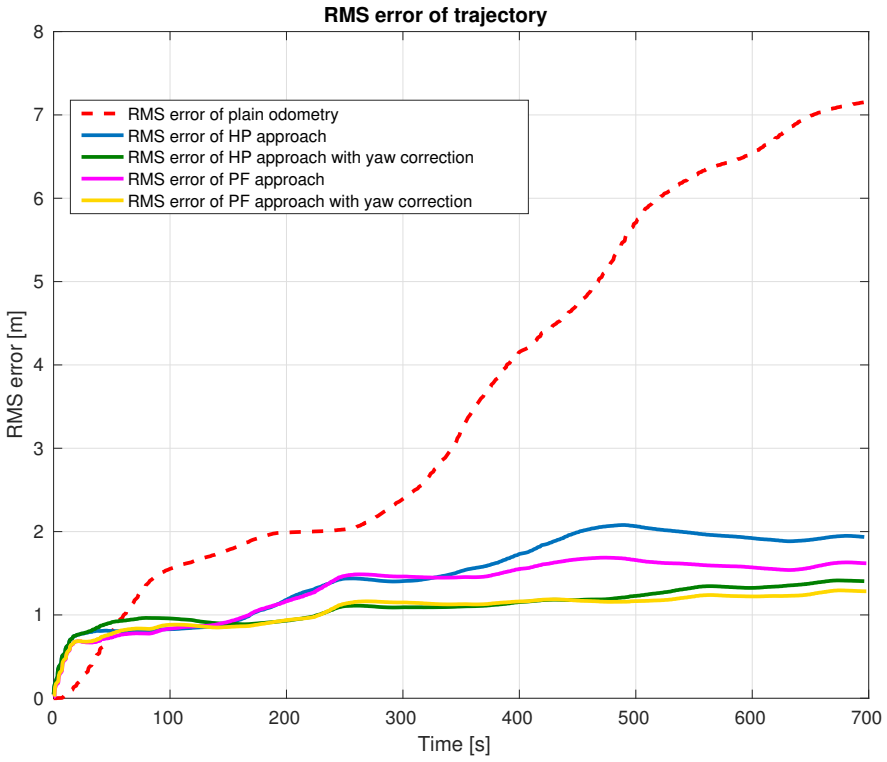

Figure 25: RMS error graph for the example data 201631 using MAP 5 comparing different approaches and plain odometry.

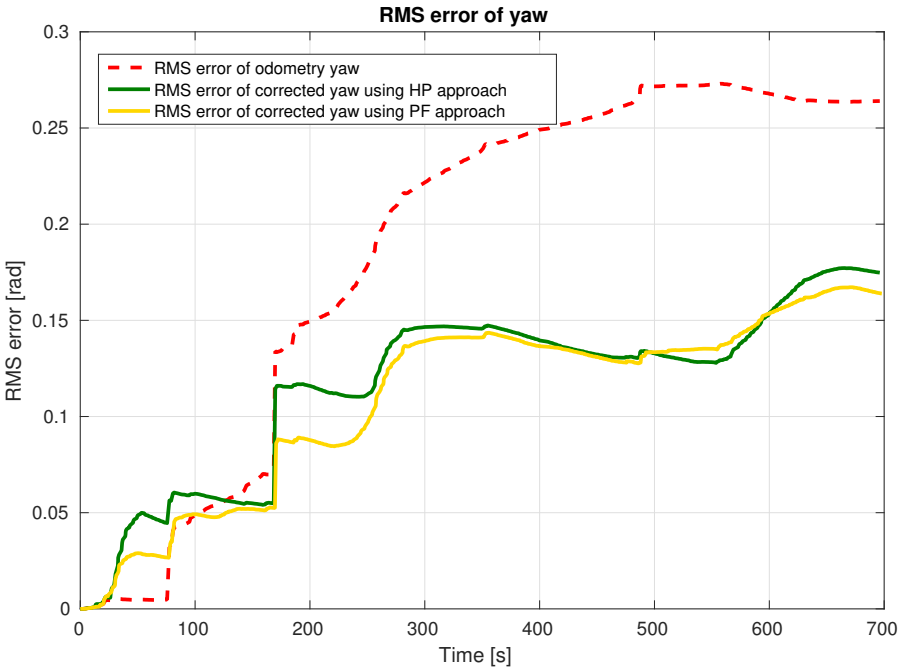

Figure 26: RMS error graph of yaw for the example data 201631 using MAP 5 comparing different approaches and plain odometry.

7 Discussion and future work

From the experimental results derive few ideas for possible improvements.

- As the first point it is necessary to reduce the chance for the localization system to fail and drift away. This can by done by computing the probability distribution not only over the location around the robot, but rather over the whole environment. As this is computationally expensive, it can be done each time after certain time period.
- Experimental results show that the yaw correction does not always provide expected result. This could be caused by parameter setting or the method itself. Anyway more methods should be tested out and compared.
- There are many parameters for the localization system that are tuned manually by observing the behavior and results. The idea for future work here is to find if there are dependencies of the parameters on each other. And reduce the number of parameters, which need to be tuned.

8 Conclusions

The thesis proposes a WiFi signal based localization system using advanced localization methods intended for a 6-DOF mobile robot. In order to accomplish this, several research papers and state of the art approaches were studied. Based on the acquired knowledge the design of the localization system was created. The main source of data for the localization system is the WiFi signal strength. WiFi signal strength over the environment was modeled using machine learning technique called Gaussian Processes. The trajectory for localization and correction is considered in 3D, a 6-DOF and for the estimation are two approaches used. One approach estimates the position by selecting the position in space with the highest probability. The second approach adopts Monte Carlo Localization also known as particle filter localization. Further an approach for correcting of the yaw angle was designed and implemented.

The implementation of the localization system is done in MATLAB. The repository with source codes and datasets for this project is publicly available here: [24]. The dataset for experimental evaluation was collect at the CTU department building on Karlovo náměstí in Prague. Four different implemented approaches were experimentally evaluated and they were capable of successful localization in 138 out of 148 trials. Moreover the RMSE in position per meter using our four tested localization approaches ranged between 0.0119 m and 0.0138 m. In comparison to the RMSE in position per meter of plain odometry, which was 0.0329 m, our approaches performed almost three times better.

Bibliography/Sources

[1] Michal Jirků, Vladimír Kubelka, Michal Reinstein. WiFi localization in 3D. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct 2016.

[2] Paramvir Bahl, Venkata N Padmanabhan, and Anand Balachandran. Enhancements to the RADAR user location and tracking system. Technical report, technical report, Microsoft Research, 2000.

[3] Anthony LaMarca, Jeff Hightower, Ian Smith, and Sunny Consolvo. Self-mapping in 802.11 location systems. In *UbiComp 2005: Ubiquitous Computing*, pages 87–104. Springer, 2005.

[4] Moisés Lisboa Rodrigues, Luiz Filipe M Vieira, and Mario FM Campos. Mobile robot localization in indoor environments using multiple wireless technologies. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 79–84. IEEE, 2012.

[5] Sinno Jialin Pan, James T Kwok, Qiang Yang, and Jeffrey Junfeng Pan. Adaptive localization in a dynamic WiFi environment through multi-view learning. In *Proceedings of the national conference on artificial Intelligence*, volume 22, page 1108, 2007.

[6] Joydeep Biswas and Manuela Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4379–4384. IEEE, 2010.

[7] R. Elbasiony and W. Gomaa. WiFi localization for mobile robots based on random forests and GPLVM. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 225–230, 2014.

[8] Andreas Haeberlen, Eliot Flannery, Andrew M Ladd, Algis Rudys, Dan S Wallach, and Lydia E Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 70–84. ACM, 2004.

[9] Yu-Cheol Lee. A reliable range-free indoor localization method for mobile robots. In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pages 720–727, 2015.

[10] Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. GPPS: A gaussian process positioning system for cellular networks. In *NIPS*, 2003.

[11] Brian Ferris, Dirk Haehnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In proc. of robotics science and systems*. Citeseer, 2006.

[12] F. Duvallet and A.D. Tews. WiFi position estimation in industrial environments using Gaussian processes. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2216–2221, 2008.

[13] Yuxiang Sun, Ming Liu, and M.Q.-H. Meng. WiFi signal strength-based robot indoor localization. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pages 250–256, 2014.

[14] Joydeep Biswas and Manuela Veloso. Multi-sensor mobile robot localization for diverse environments. In *RoboCup 2013: Robot World Cup XVII*, pages 468–479. Springer, 2014.

[15] Jakub Simanek, Michal Reinstein, and Vladimir Kubelka. Evaluation of the EKF-based estimation architectures for data fusion in mobile robots. *Mechatronics, IEEE/ASME Transactions on*, 20(2):985–990, 2015.

[16] V. Kubelka, L. Oswald, F. Pomerleau, T. Svoboda, M. Reinstein. Robust data fusion of multimodal sensory information for mobile robots. In *Journal of Field Robotics*, 32 (4):447-473, 2015.

[17] C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. In *Adaptive Computation and Machine Learning*, The MIT Press, Cambridge, MA, USA, 2006.

[18] X. He, S. Badiei, D. Aloi and J. Li. WiFi iLocate: WiFi based indoor localization for smartphone. In *Wireless Telecommunications Symposium (WTS)*, 2014.

[19] M. P. Deisenroth. Efficient Reinforcement Learning using Gaussian Processes. In *Dissertation*, pages 8-19, 2012.

[20] GPML Matlab Code. *gaussianprocess.org*. [online]. 28.10.2016 [cit. 2017-05-17]. Available at WWW: `http://www.gaussianprocess.org/gpml/code/matlab/doc/`.

[21] S. Thrun, W. Burgard and D. Fox. Probabilistic robotics.*Early draft*, pages 89-102, 2000.

[22] H. Rashid, A. K. Turuk. Dead reckoning localisation technique for mobile wireless sensor networks. In *IET Wireless Sensor Systems*, Volume 5, Issue 2, pages 87-96 , 2015.

[23] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, February 2013.

[24] M. Jirků. 3D WiFi localization repository. *gitlab.fel.cvut.cz*. [online].[cit. 2017-23-05]. Available at WWW: `https://gitlab.fel.cvut.cz/jirkumic/3d-wi-fi`.

[25] V. Kubelka, M. Reinstein, T. Svoboda. Improving multimodal data fusion for mobile robots by trajectory smoothing. In *Robotics and Autonomous Systems*, vol. 84, pp. 88-96, 2016.

[26] The Least-Squares Line. *efunda.com*. [online].[cit. 2017-23-05]. Available at WWW: `http://www.efunda.com/math/leastsquares/lstsqr1dcurve.cfm`.

List of Figures

List of Tables